Syracuse University
SURFACE

Dissertations - ALL SURFACE

December 2020

# Model-based Approach for Product Requirement Representation and Generation in Product Lifecycle Management

Omer Yaman
*Syracuse University*

Follow this and additional works at: https://surface.syr.edu/etd

Part of the Engineering Commons

## Recommended Citation

Yaman, Omer, "Model-based Approach for Product Requirement Representation and Generation in Product Lifecycle Management" (2020). *Dissertations - ALL*. 1290.
https://surface.syr.edu/etd/1290

This Dissertation is brought to you for free and open access by the SURFACE at SURFACE. It has been accepted for inclusion in Dissertations - ALL by an authorized administrator of SURFACE. For more information, please contact surface@syr.edu.

## ABSTRACT

The requirement specification is an official documentation activity, which is a collection of certain information to specify the product and its life-cycle activities in terms of functions, features, performance, constraints, production, maintenance, disposal process, etc. It contains mainly two phases; product requirement generation and representation. Appropriate criteria for the product design and further life-cycle activities are determined based on the requirement specification as well as the interrelations of product requirements with other life-cycle information such as; materials, manufacturing, working environments, finance, and regulations. The determination of these criteria is normally error-prone. It is difficult to identify and maintain the completeness and consistency of the requirement information across the product life-cycle. Product requirements are normally expressed in abstract and conceptual terms with document base representation which yields unstructured and heterogeneous information base and it is unsuitable for intelligent machine interpretations. Most of the time determination of the requirements and development of the requirement specification documents are performed by the designers/engineers based on their own experiences that might lead to incompleteness and inconsistency. This research work proposes a unique model-based product requirement representation and generation architecture to aid designers/engineers to specify product requirements across the product life-cycle. A requirement knowledge management architecture is developed to enhance the capabilities of the current Product Life-cycle Management (PLM) platforms in terms of product requirement representation and generation. After a systematic study on the categorization of product requirements, an ontological framework is developed for the specification of the requirements and related product life-cycle domain information. The ontological framework is embedded in an existing PLM system. A computational platform is developed and integrated into the PLM system for the intelligent machine processing of the product requirements and related information. This architecture supports product requirement representation in terms of the ontological framework and further information retrieval, inference, and requirement text generation activities.

# MODEL-BASED APPROACH FOR PRODUCT REQUIREMENT REPRESENTATION AND GENERATION IN PRODUCT LIFECYCLE MANAGEMENT

by

**Omer Yaman**

B.S., Istanbul Technical University, 2009

B.S., Istanbul Technical University, 2011

M.S., Syracuse University, 2013

Dissertation

Submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Mechanical and Aerospace Engineering

Syracuse University

**December 2020**

Don’t get lost in your pain, know that one day your pain will become your cure.

*Rumi*

**Copyright © 2020 Omer Yaman**

**All rights Reserved**

## ACKNOWLEDGEMENTS

I would like to thank five groups of people, without whom this thesis would not have been possible: my advisor, my thesis committee members, my lab mates, my sponsor, the Republic of Turkey Ministry of National Education, and my family.

First, I would like to express my special appreciation, and thanks to my advisor Professor Dr. Utpal Roy, you have been a tremendous mentor for me. I would like to thank you for encouraging my research and for allowing me to grow as a research scientist. Your advice on both research as well as on my career has been invaluable.

Besides my advisor, I would like to thank the rest of my dissertation committee members for their great support, invaluable advice, and valuable suggestions and comments for this dissertation.

A big thank you goes to my lab mates and friends, including Bicheng Zhu, Heng Zhang, Yunpeng Li, Hang Yin, and Kai Sun, for their continued support and good suggestions in my research work.

I am also grateful to my sponsor, the Republic of Turkey Ministry of National Education, for their financial support that I otherwise would not have been able to develop my scientific discoveries.

Finally, I would like to acknowledge the most important persons in my life, my parents Mumun Yaman and Fatma Yaman, and my wife Betul F. Yaman for their unconditional love devotion. This dissertation would not have been possible without their warm love, continued patience, and endless support.

## TABLE OF CONTENTS

**Abstract** . . . . . i

**Acknowledgments** . . . . . v

**List of Tables** . . . . . ix

**List of Figures** . . . . . xi

**1 Introduction** . . . . . 1
- 1.1 Research Background . . . . . 1
- 1.2 Problem Statement . . . . . 9
- 1.3 Scope and Overview of Thesis Approach . . . . . 10
- 1.4 Proposed Methodology and Research Objectives . . . . . 11
- 1.5 Dissertation Outline . . . . . 13

**2 Literature Review and Background** . . . . . 16
- 2.1 Information Modeling and Ontology . . . . . 16
  - 2.1.1 Ontology in Product Design and Manufacturing . . . . . 18
  - 2.1.2 Ontology Based Approach for Product Lifecycle and Knowledge Management . . . . . 20
- 2.2 Relationships in the Product Design Domains . . . . . 22
- 2.3 Product Requirement Management . . . . . 23
  - 2.3.1 Ontology in Product Requirement Management . . . . . 26
  - 2.3.2 Linguistic Methods and Information Extraction in Requirement Management 28
- 2.4 Summary and Limitation of Current Studies . . . . . 30

**3 Model-based Product Requirement Representation and Generation Methodology** . 33
- 3.1 Requirement Knowledge Management Architecture . . . . . 33
  - 3.1.1 The Role of the Ontology-Based Information Model in REKMA . . . . . 36
  - 3.1.2 The Role of the PLM System in REKMA . . . . . 38
  - 3.1.3 The Role of the Computational Platform in REKMA . . . . . 40
- 3.2 Formal Representation and Generation of the Product Requirements . . . . . 41
- 3.3 Summary . . . . . 52

**4 Development of the Ontology-Based Information Model for the Product Requirement Specification** . . . . . 53
- 4.1 General Description of the Requirement Model . . . . . 53

4.2 Modeling Methodology and Elements for the Requirement Information Model . . . 55
4.3 Information Model for Product Requirement Specification . . . . . 59
4.4 Product Requirement Actor Models: Material, Process and Shape . . . . . 67
4.4.1 Material Information Model . . . . . 68
4.4.2 Shape Information Model . . . . . 70
4.4.3 Process Information Model . . . . . 72
4.5 Summary of Model Development . . . . . 77

**5 PDROM Implementation based on Web Ontology Language . . . . . 78**
5.1 Implementation of PDROM in Protege . . . . . 78
5.1.1 PDROM Model Realized in OWL . . . . . 79
5.1.2 Modeling of OWL Components . . . . . 81
5.1.3 Use of the Reasoners . . . . . 85
5.1.4 Use of the SWRL and SQWRL . . . . . 86
5.2 Automobile Brake Requirements for the PDROM Implementation . . . . . 88
5.2.1 Description of Requirements and Rules Development . . . . . 88
5.2.2 Instantiation of the PDROM Model and Completeness Check . . . . . 93
5.2.3 Consistency Check . . . . . 96
5.3 Query Processing . . . . . 98
5.4 Summary of PDROM OWL Implementation . . . . . 99

**6 PLM System and Computational Platform in the REKMA . . . . . 101**
6.1 PLM System Development . . . . . 101
6.1.1 Customization of the PLM System based on PDROM Structure . . . . . 102
6.1.2 Representation of the Product Requirements In The PLM System . . . . . 105
6.1.3 Software Installation and Data Collection . . . . . 107
6.2 Computational Platform and Integration of the Systems . . . . . 109
6.2.1 Data and Information Exchange . . . . . 113
6.3 Requirement Data in RDF Format . . . . . 115
6.4 Summary . . . . . 116

**7 Requirement Information Extraction and Text Generation in the REKMA . . . . . 118**
7.1 Product Requirement Information Extraction and Retrieval . . . . . 118
7.1.1 Syntax Analysis of the Product Requirements . . . . . 121
7.1.2 Requirement Semantics Extraction . . . . . 125
7.1.2.1 Development of the Requirement Named Entity Recognition Model 125
7.1.2.2 Results and Evaluation of the Requirement NER Model . . . . . 130
7.1.3 Determination for Super-entity and Sub-entity Hierarchy . . . . . 132
7.1.4 Entity Inference . . . . . 134
7.2 Requirement Text Generation . . . . . 136
7.3 Summary . . . . . 140

**8 Conclusion** . . . . . 142
8.1 Dissertation Summary . . . . . 142
8.2 Research Contribution . . . . . 145
8.3 Limitation and Future Work . . . . . 148

**Appendix A Product Requirement Information Model** . . . . . 152
A.1 UML Description . . . . . 152
A.2 Material Properties in the Material Information Model . . . . . 152
A.3 Process Information Model . . . . . 153

**Appendix B PDROM implementation in Protege** . . . . . 157
B.1 SWRL and SQWRL Rules . . . . . 157

**Appendix C Development of the Computational Applications** . . . . . 161
C.1 Building of the Product Function Architecture . . . . . 161
C.1.1 Collection of Functional Data for Coffee Makers . . . . . 161
C.1.2 Building Functional Architecture for Coffee Maker . . . . . 163
C.2 Product Inspection and Maintenance Requirement Generation . . . . . 163
C.2.1 Model Development for the Application . . . . . 165
C.2.2 Overall System Architecture Of UAS Predictive Maintenance Application . 166

**Appendix D Custom Requirement NER** . . . . . 170
D.1 Requirement Texts for NER Testing . . . . . 170
D.2 NER Test Result for Performance Analysis . . . . . 172

**References** . . . . . 184

**Vita** . . . . . 185

## LIST OF TABLES


2.1 Overview of Requirement Analysis Tools & Techniques . . . . . 31

3.1 Table Representation of the Requirement Specification for an Automobile Brake Rotor . . . . . 42

3.2 Textual Description of the Requirement Specification for an Automobile Brake Rotor 43

3.3 The Proposed Format to Represent a Product Requirement Sentence . . . . . 43

3.4 Presentation of a Product Requirement Sentence . . . . . 45

3.5 Inference Entities and Representation for a Requirement Text . . . . . 46

3.6 Formalized Requirement Syntaxes and Examples for Textual Requirement Generation . . . . . 48

3.7 The Representation Format of a Requirement Sentence based on the Information Model Structure . . . . . 51

5.1 Automobile Brake System Requirements . . . . . 90

5.2 Categorization of the Automobile Brake System Requirements in PDROM . . . . . 94

7.1 Syntax Analysis of the General Requirement Form . . . . . 121

7.2 Syntax Analysis of a Requirement with Verb Phrase Modification . . . . . 122

7.3 Syntax Analysis of a Requirement with Overlapping Entities . . . . . 123

7.4 Syntax Analysis of a Non-Functional Requirement . . . . . 123

7.5 Syntax Analysis of a Requirement with Two Entities in Subject Part . . . . . 124

7.6 Syntax Analysis of a Non-Functional Requirement with Subject Complement . . . 124

7.7 Requirement Named Entity Types and Descriptions . . . . . 127

7.8 Overlapping Entities and Rules to Define Entity Type . . . . . 128

7.9 Requirement NER Output of a Product Requirement Sentence . . . . . 130

7.10 Accuracy Score of Requirement NER . . . . . 131

7.11 Requirement NER Output of a Product Requirement Sentence . . . . . 134

7.12 Examples of the Requirement Information Inference Rules . . . . . 135

7.13 Requirement Text Generation for Coffee Maker and Rules . . . . . 138

A.1 Factors Affecting Powder Metallurgy Part Quality . . . . . 155

C.1 Function Model Data of Coffee Makers . . . . . 162

C.2 Representation of the Drone Inspection Requirements . . . . . 164

## LIST OF FIGURES


1.1 Flow Chart for Product Design [1] . . . . . 3

1.2 Product Life-Cycle and Requirement Activities . . . . . 5

1.3 Design Requirements and integrated Domains . . . . . 6

1.4 Product Life-cycle and Information Flow . . . . . 7

1.5 V-model of the system development life cycle and Requirement Management [14] . 11

1.6 The Overall Structure of the Dissertation . . . . . 15

3.1 The Requirement Knowledge Management Architecture . . . . . 34

3.2 Information Models in the Requirement Knowledge Management Architecture . . . 37

3.3 The UML Activity Diagram for Requirement Sentence Analysis Process . . . . . 49

3.4 The Architecture of Requirement Information Extraction and Technologies . . . . . 51

4.1 The Integrated Information Model with Design Rationale and Product Requirement Models . . . . . 54

4.2 The Layered Modeling Methodology for the Development of the PDROM . . . . . 56

4.3 The Ontology Development Process . . . . . 57

4.4 Part I Model for the Requirement Representations . . . . . 60

4.5 Part II Model for the Requirement Rationale . . . . . 63

4.6 PDROM Part I and Part II Integrated . . . . . 66

4.7 Material Information Model . . . . . 69

4.8 Shape Information Model . . . . . 71

4.9 Process-oriented Information Model . . . . . 73

5.1 Steps for Formal implementation of PDROM . . . . . 80

5.2 PDROM Implementation in Protégé . . . . . 84

5.3 Implementing the Semantic Axioms-Necessary Condition . . . . . 85

5.4 Implementing the Semantic Axioms-Necessary and Sufficient Conditions . . . . . 85

5.5 PDROM Implementation in Protégé and Semantic Rules in the PDROM . . . . . 87

5.6 (a) Black Box Model and (b) Function&Flow Representation for Automobile Brake System . . . . . 89

5.7 Requirement Rationale Model Utilization and Product Requirement Generation for Automobile Brake System . . . . . 92

5.8 Instantiation of the PDROM Model-An Example . . . . . 95

5.9 Inference and Reasoning Mechanisms for R12 . . . . . 96

5.10 R7 Storage in Information Structure . . . . . 97

5.11 Inconsistency Checking . . . . . 97

5.12 Requirement Individuals with Necessary Information . . . . . 99

5.13 Material Property and Requirement Description Instances for Rotor Part . . . . . 99

5.14 PDROM OWL Implementation Summary . . . . . 100

6.1 PDROM Contents in the PLM System . . . . . 104

6.2 Function Model in PLM for Brake System . . . . . 105

6.3 Requirement Form When User Inputs Requirement Information to Create a New Requirement Item . . . . . 106

6.4 Requirement Form When the PLM Gets the Information from the Computational Platform . . . . . 106

6.5 Aras Innovator login window . . . . . 108

6.6 The Proposed Three-Layered Framework for Information Model Utilization . . . . 110

6.7 General CL2M Framework for the Requirement Data Management . . . . . 112

6.8 Implementation of the Communication System . . . . . 114

6.9 Requirement Data Generation in RDF Format . . . . . 116

7.1 Data Exchange and Initial clustering of Product Functions for a Coffee Maker in Computational Platform . . . . . 119

7.2 SpaCy NER Results of Sample Requirement Texts . . . . . 127

7.3 Sample of the Training Data in Json Format . . . . . 129

7.4 Sample of the Training Data in spaCy Format . . . . . 129

7.5 Convert Json File to spaCy's Trainig Format . . . . . 130

7.6 Semantic Similarity of 'coffee beans' . . . . . 133

7.7 Inference Entities in PLM System . . . . . 136

7.8 Function Requirement Generation and Representation in the PLM System . . . . . 137

7.9 Maintenance Requirement Generation . . . . . 140

A.1 UML Diagrams . . . . . 152

A.2 Expanded PIM for the Powder Metallurgy Process . . . . . 154

C.1 Structure and Function Representation for a Coffee Maker . . . . . 162

C.2 Overall System Architecture for Inspection and Maintenance Requirement Management . . . . . 167

C.3 Maintenance Requirement Generation . . . . . 168

C.4 UAV Data Repository Structure and PLM Implementation . . . . . 169

# CHAPTER 1

# INTRODUCTION

In this chapter, an overview of the research performed in this dissertation is presented. This chapter starts with the technical context and background of the research. The problem statement, thesis scope, and research methodology are then introduced. Finally, this chapter is wrapped up by outlining the structure of the overall dissertation.

## 1.1 Research Background

The technological advantages in today's marketplace require product stakeholders to be more collaborative by sharing and exchanging product-design information in order to compete with rivals. The information exchange and sharing make product design more knowledge-intensive requiring companies to develop/use integrated product modeling architectures in order to build high quality products with the lowest cost and minimum lead time while reducing risk. Such integrated modeling architecture not only supports product design but also provides necessary information for any future decision making activities in the product life-cycle such as material selection, manufacturing, usage, maintenance, disposal, and recycling, etc. The product design criteria and this necessary information are specified in the product requirements. Therefore, formal representation and generation of the product requirements, which are stated within the product Requirement Specification are very crucial to this integrated modeling architecture.

A requirement is a single statement of something the product or system must do or quality it

must have. The requirement specification document captures the set of all product requirements that guide a product design. It includes all the supporting documentation and related information necessary to justify and explain those requirements for the design, verification, and maintenance of the product. Development of the requirement specification is an activity that is considered as the first step of the product design in the product life-cycle. The product design has many stages that start with the design requirements and finishes in product development. As shown in figure 1.1, three main design stages: conceptual, embodiment, and detail design that are completed sequentially between starting and endpoints. At each stage of the product design, information mainly about materials and processes is needed while considering product requirements. Material and process selection activities are performed based on the product requirements along with the product design stages.

A requirement specification document of a product explains what the product is going to do, determines how the product is going to operate, and offers guidance to the design, production, maintenance, service, disposal, etc. teams in designing, manufacture, maintain, repair, dispose of, etc. the product. It reflects the design information of the system and states how the design will meet the product requirements, even though the design might not be a one-to-one response to the requirements. It can be defined as a documented requirement list that describes the physical, structural, financial, and other needs for a given product or part.

A requirement specification document may include hundreds of product requirements. One important point is that the requirement specification is written or contributed by different stakeholders, including customers, users, repair and maintenance teams, engineers, the design team, and others, each with different knowledge and viewpoints.

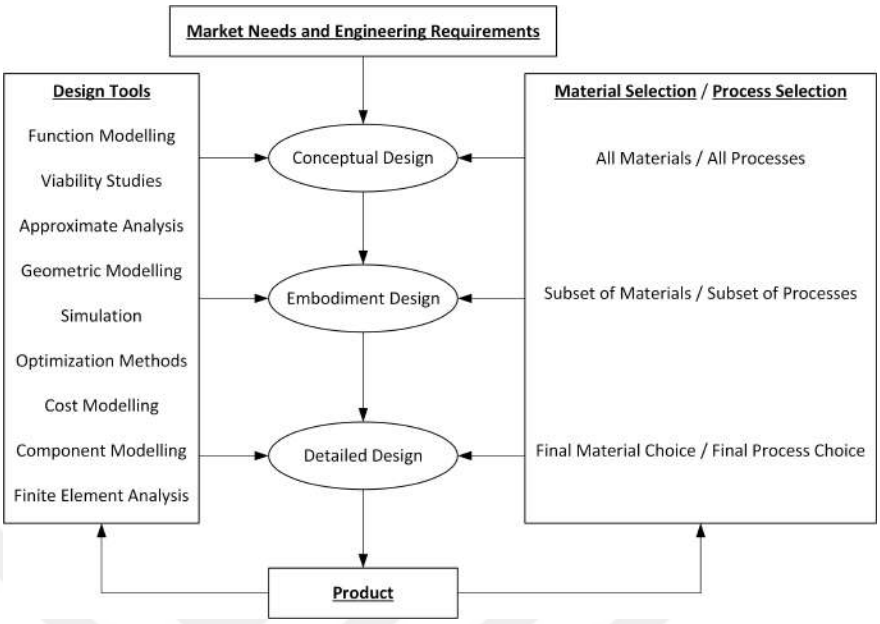


Figure 1.1: Flow Chart for Product Design [1]

The requirement specification document of a product contains crucial information for its whole product life-cycle activities and dominates these activities. For example, at the outset of the product design, immediately after the function-to-form mapping activity [2], designers are compelled to choose materials whose properties match with the design requirements of the overall product design. In another case, during product disposal stages, different regulatory requirements play important roles in deciding product recovery strategies. Matching the design requirements of a product with an appropriate material, form, and process while considering other product life-cycle activities and dependencies between them is not a trivial task; it needs specialized domain knowledge. Whatever decisions are taken they must satisfy the product requirements.

It is obvious that the importance of the product requirements is understood and addressed to certain extents [3]; however, essential gaps and challenges remain to be dealt with. First of all, the information about product requirements is broad and complex. It is difficult to identify and

capture the complete requirement information [4]. At the beginning of a design project, what a designer receives from the customer is just the high-level requirements of a product. However, for a complete design, additional and detailed requirements across the product life cycle like requirements for form, material, process, maintenance, and disposal are implicit, and it is hard to capture them all by engineers/designers.

Another aspect of this challenge is that the complexity of the product requirements may increase if any interactions among themselves occur. As an example, considering the financial requirements, the designer may want to reduce the weight of a certain part by modifying the geometry, which may lead to a compromise in the structural requirements, or by choosing lightweight material, which may lead to a compromise in the material related requirements that both affect process requirements.

In addition, variables used to describe design requirements are usually poorly understood in the beginning and are expressed in abstract and conceptual terms. Therefore, a well-structured and extensible semantic ”requirement” model is necessary for intelligent machine interpretations. The traditional approach for product design requirement storage is based on text documents, which normally yield an unstructured and heterogeneous information base, and is unsuitable for automatic machine processing.

Manually written and recorded product requirements may lead to inconsistencies and incompleteness, and this unstructured natural language component is hard to analyze by the computational systems [5]. As shown in figure 1.2, informal, inconsistent, and incomplete requirement specification may lead to unnecessary design feature conceptualization, production or structure failure, different kinds of hazards, etc. Even though some structured formats (like XML) can be used for design requirement storage, the level of machine intelligence is still limited, it can

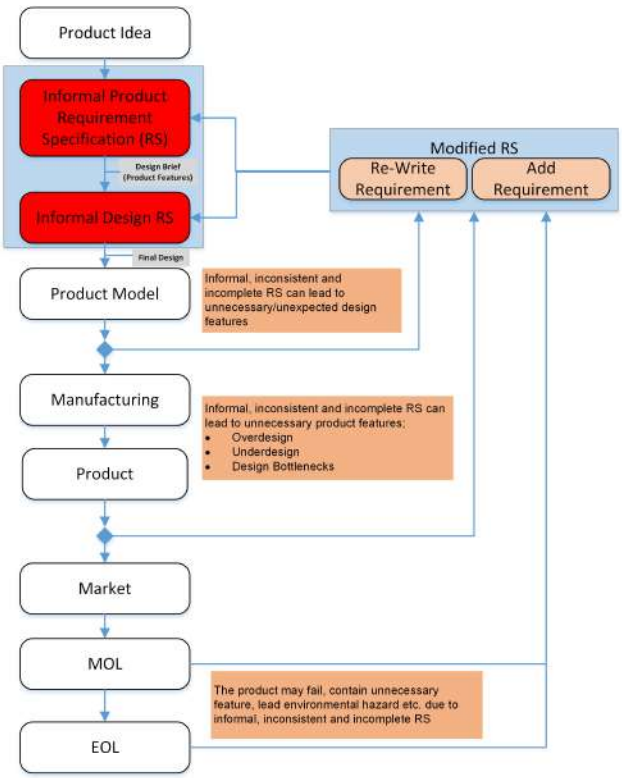


Figure 1.2: Product Life-Cycle and Requirement Activities

only ”read” the data rather than ”understand” the meaning of the data (data interpretation). For example, a good design requirement specification model should enable computers to understand the term ”Good machinability”. The term ”Good machinability” refers to the part machining process characteristics which are defined by a variety of factors like acceptable surface finishes, tool life, tool forces, and power consumption, etc.

Moreover, when developing a new product, most of the time, attention is paid to ensure that the end products can achieve specified product performances. However, the effects of product form, material, and manufacturing process on product performances might not be determined due to poor information management; representation, mapping, exchange, and sharing of design parameters as well as linking them to the design requirements which are shown in Figure 1.3 [6].

Another challenge for the requirement specification is that information flow becomes vague or

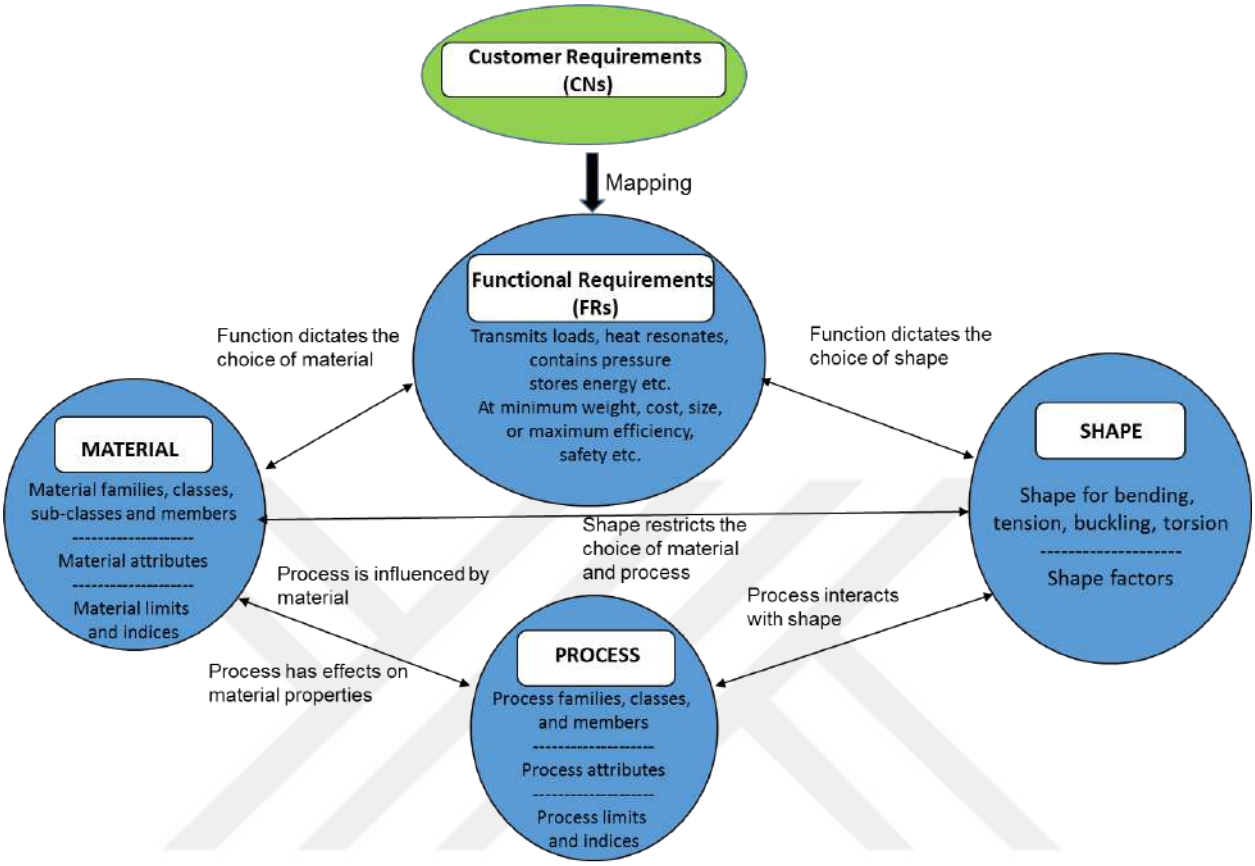


Figure 1.3: Design Requirements and integrated Domains

unrecognized during Middle-Of-Life (MOL) and End-Of-Life (EOL) stages of the product life-cycle [7]. Beginning-Of-Life (BOL) stage is supported by many information systems such as Product Data Management (PDM), CAD/CAM, Knowledge Management (KM), and requirements analysis tools such as Quality Function Deployment (QFD). These tools and information systems make the flow of the product requirement and requirement-related information quite complete in BOL. However, MOL and EOL phases have limited visibility for the requirement and product-related information flow. This issue leads to difficulty in feeding information forward through the product life-cycle stages. Controlling the flow of the requirement and product information across the product life-cycle phases is very important to develop quality product design, execute an

adequate manufacturing process, change/upgrade product requirements, etc. Figure 1.4 shows the product-related information at the different phases of the life-cycle and information flow through the product life-cycle stages. All this information is made visible over the whole product life-cycle by advanced technologies like the internet, wires communication, and Product Embedded Information Device (PEID) [8].

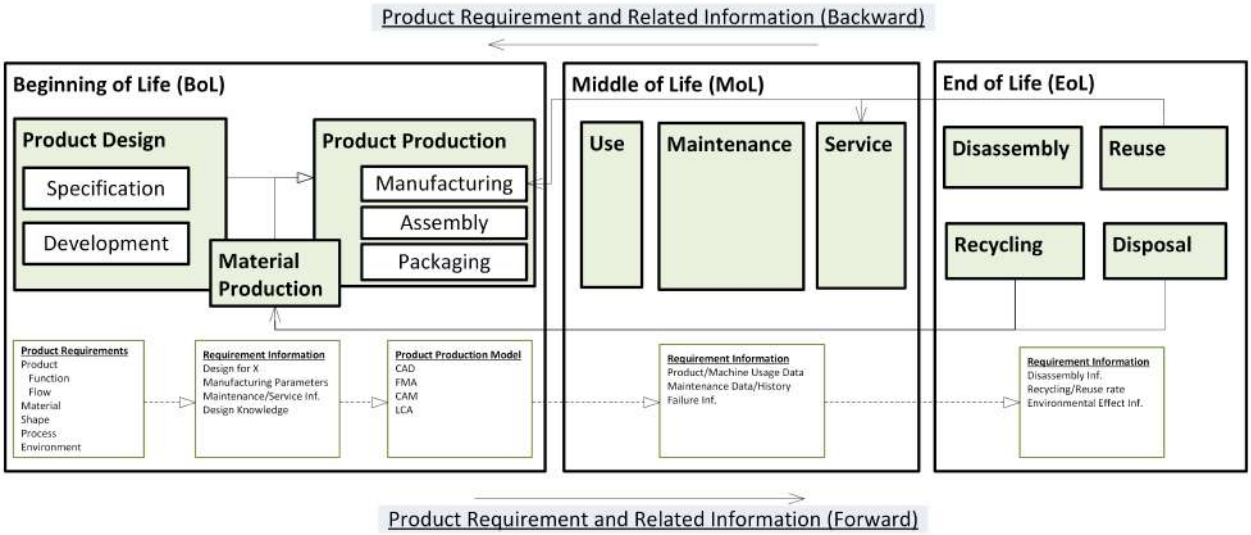


Figure 1.4: Product Life-cycle and Information Flow

The current solution for the management of product data and knowledge within the product design process is enabled by the Product Life-Cycle Management (PLM) system. The PLM system manages the product information across the entire life-cycle of a product from the beginning, through engineering design and manufacturing, to service and disposal of products. The PLM is a powerful tool that can work with advanced technologies to gather, store, and manage product-related information. However, it has document-based management that might yield interoperability and communication issues between designers, analysts, operators, and tools that are involved in product life-cycle activities [9]. It is not strongly linked to the structured and integrated

representation and generation of the product requirements.

The final challenge is to define how and which product information is used to generate product requirements through the life-cycle. Model-Based Systems Engineering (MBSE) has the ability to increase the completeness of the requirements by providing a better way to capture, analyze, share, and manage product information [10]. However, most of the MBSE tools and languages require requirement construction manually that is open to human error and time-consuming [11]. Requirement generation requires a deep understanding of semantics and structure of product life-cycle information over the whole life-cycle phases.

Therefore, an elaborate study must be done to aid designers/engineers for the formal representation and generation of the product requirements across the design and further life-cycle stages of a product in the PLM.

In order to address the challenges above, this research explores the product requirements extensively and proposes an integrated architecture for the product requirement knowledge management. It supports a new level of product requirement information extraction, retrieval, storage, generation, representation, and integration among product life-cycle stages by addressing different facets of requirement specifications. The following activities are studied and done to build the requirement knowledge management architecture. A Product Design Requirement Ontology Model (PDROM) [12] that provides rich requirement semantics is developed by using Semantic Web (SW) technologies and ontologies. The PDROM is extended with the Structure-Behavior-Function-Failure (SBFF) model to support the retirement generation with the representation of the requirement rationale. Moreover, data structures of these models are implemented in the PLM to take advantage of PLM's information management. Finally, a computational method is created to retrieve requirement information and knowledge from stored data in the PLM and streaming data that is

continuously generated from different product life-cycle phases.

## 1.2 Problem Statement

The requirement specification is carried out by the engineer/designer, and it heavily depends on his/her own experience. There is no complete automated system to take full advantage of Computer Aided Engineering (CAE) and digital thread solutions for the requirement specification. Current product requirement management tools such as PLM tolls, ReQtest [13], etc. deploy requirements as a document. This makes analysis and validation of the product requirements and execution of the design activities based on the requirements difficult. Therefore, an integrated and automated system that represents, generates, and validates the product requirements through the product life-cycle is needed.

Based on a comprehensive study and initial review of the product requirements, requirement specification development, MBSE, PLM, and IM, as the core studies of the dissertation, the following research questions are answered in this dissertation:

• How to formally represent and automatically generate the product requirements and requirement-related information across the design and further life-cycle stages of a product

• How to identify and capture the information and relationships regarding the product requirements across the product life-cycle phases

• How to extract and retrieve requirement semantics and information from informal, and textual product requirements

These questions can be broken down into four partial research questions, which guide the scope of the thesis:

• What technologies should be used to identify, represent, generate, and retrieve requirement

information

• How to develop, implement, and validate an integrated product requirement knowledge management architecture

• How to develop, implement, and validate a product requirement information model

• How to develop a computational application that retrieves and infers product requirement information

The following sections explain our scope and approach that offers solutions to these questions.

## 1.3 Scope and Overview of Thesis Approach

The requirement specification plays an important role at every stage of product development in the PLM contrary to the common misconception that requirements specification is just a single stage of the PLM [14]. Product requirement generation and specification of the requirements are mostly carried out and completed at the outset of product development but their allocation process occurs within the context of a larger system development life-cycle as well within whole PLM stages.

The requirement specification is a management and development process to define, document, and maintain product requirements. This process is integrated with the system models under the content of Model-Based Systems Engineering. In order to formalize requirement specification and represent relations between the product requirements and the system models that consist of requirement metadata, this thesis proposes the PDROM. The "vee model" or "V-model" (figure 1.5) of the product development life cycle illustrates that the system development begins in the upper left with customer requirements and continues with deriving system and subsystem requirements while showing relationships with system models and validation. While our main scope of this thesis takes shapes on the left side of the V, formalization of product requirements, developing system

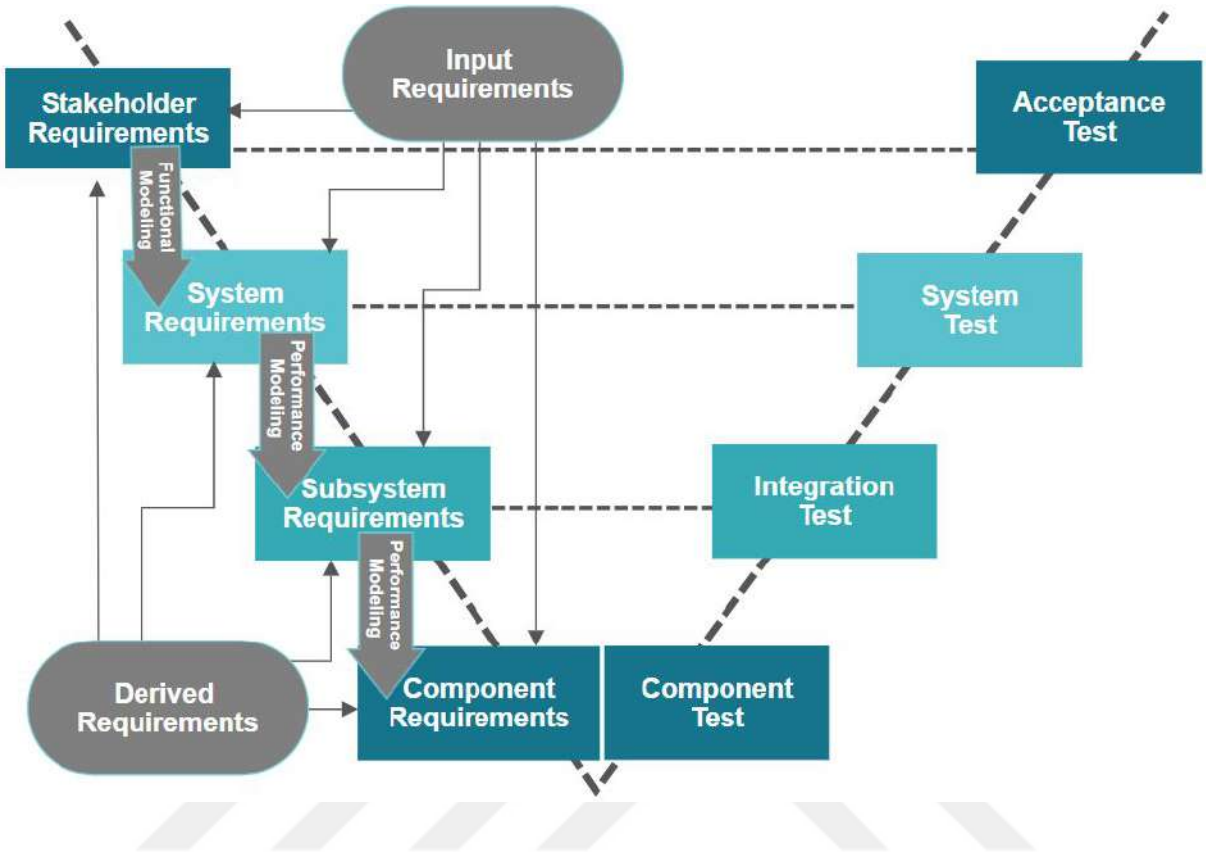


Figure 1.5: V-model of the system development life cycle and Requirement Management [14]

models, and identifying relationships can help to automate the requirement validation process. To address the thesis scope and manage (generate and formally represent) product requirements and requirement-related information across the product life-cycle phases, this thesis proposes a REquirement Knowledge Management Architecture (REKMA).

## 1.4 Proposed Methodology and Research Objectives

The thesis discusses mainly two broad topics; product requirement knowledge management and information model development. This research addresses the product requirement representation and generation in the engineering design field by focusing on the flow of the product requirements information across the product life-cycle. It explores the product requirements extensively and

proposes ontology-based information models that provide rich domain semantics for product requirements and other product life-cycle domains mainly form, material, and process. These models identify the key engineering parameters during the conceptual design and support a new level of product domain information, extraction, retrieval, storage, and analysis by addressing product requirements and requirement-related parameters. In this study, proposed models are integrated and implemented in a single platform to formally represent and generate product requirements. This research aims to improve design quality and provide a better insight into the design issues while reducing the design process' cost and time and eliminating human errors by proposing a unique requirement specification methodology.

The general objective of this research is to develop a model-based architecture for product requirement knowledge management. It is capable to extract and retrieve product requirement knowledge, generate and represent multiple types of product requirements in a single computational environment that supports by both PLM and SW technologies. It supports the development, representation, and analysis of the requirement specification in the product design and further product life-cycle phases.

In order to validate the proposed methodology, this thesis discusses the use-case analysis. Requirement specifications of an automobile brake rotor, a coffee maker, and an unmanned aerial vehicle (UAV) are studied as proof-of-concept in this thesis.

The overall research methodology of this work is organized in five logical steps which include:

i. A review of the current works, technologies, and literature (Chapter 1 & 2),

ii. The development of new concepts and methodologies (Chapter 3 & 4),

iii. The implementation and testing of the developed concepts and methods (Chapter 5, 6 & 7),

iv. The overall evaluation of the results (Chapter 5 & 7), and

v. The conclusion and possible future extensions of this work (Chapter 8).

The detailed information about each step is described in the dissertation outline:

## 1.5 Dissertation Outline

This dissertation is organized in eight chapters as depicted in figure 1.6, and description of each chapter is narrated as follows:

**Chapter 1** provides an introduction to the thesis by elaborating on the background and motivation of this research, stating problem statement and research questions, discussing the research objectives, and providing an overview of the thesis structure.

**Chapter 2** reviews literature related to the research problems that address the first research question(see figure 1.6) by establishing the background of requirement specification, IM, and closed-loop PLM. This chapter first discusses the literature about the existing requirement specification and requirement management definition methodologies along with model-based design methodologies. Next, it reviews the definition and description of IM and existing IM developments in the domain of requirement specification. It also discusses strategies, methods, and tools of the IM.

**Chapter 3** addresses the first research question and discusses the proposed methodology for the model-based requirement representation and generation. It introduces and discusses REKMA and related technologies. This chapter also presents the roles of the ontology-based information model, PLM system, and computational platform in the REKMA.

**Chapter 4** comprehensively presents the development of PDROM that addresses the second research question. The chapter starts with the product requirement-related information analysis in the domain of requirement specification, which results in a high level of domain conceptualization.

Afterward, the PDROM model is introduced to turn the product domain knowledge into standardized product requirement models to support the specification of the requirements. Submodels of the PDROM; material, form, product, and process are developed and discussed.

**Chapter 5** presents the formal implementation of the PDROM in the Protégé and the instantiation of the proposed model. It is discussed how the PDROM can store product requirements and detect incompleteness or inconsistency in the requirement specification. Also, several semantic inference and query rules are defined and discussed in this chapter.

**Chapter 6** mainly discusses the role of the PLM system and computational platform in the proposed requirement knowledge management architecture. It first presents the implementation of the PDROM data structure in an existing PLM tool. Then, it introduces the development and use of the computational platform which is integrated into the PLM system. Finally, this chapter presents a communication mechanism among a product, the PLM system, and computational platform to transport data and file and to convert relational data from the PLM system into the Resource Description Framework (RDF).

**Chapter 7** argues the third research question which is about how to extract and retrieve product requirement semantics, information, and knowledge from structured, unstructured, informal, textual product requirements, and requirement-related data. After a comprehensive study of requirement semantics and Natural Language Processing (NLP) techniques, a method for the requirement information and knowledge retrieval is developed with the use of machine learning. In addition, chapter 7 discusses the linguistic analysis of the product requirements. Predefined product requirement syntaxes to write requirements in natural language and automatical generation of the textual requirements are discussed and implemented in this chapter. This chapter also presents the analytical applications that are developed to analyze product requirements and retrieve product

requirement knowledge and information from product life cycle data. To validate the proposed method, case studies are presented. The concepts of the digital thread and closed-loop PLM for the product requirement information flow are discussed.

**Chapter 8** concludes this dissertation and proposes directions for future works and research. The dissertation is summarized, and key findings and main thesis contributions are highlighted. It starts with a summary of the whole dissertation. Then, the research contributions are described. Finally, the limitations of the research work in the dissertation are discussed.

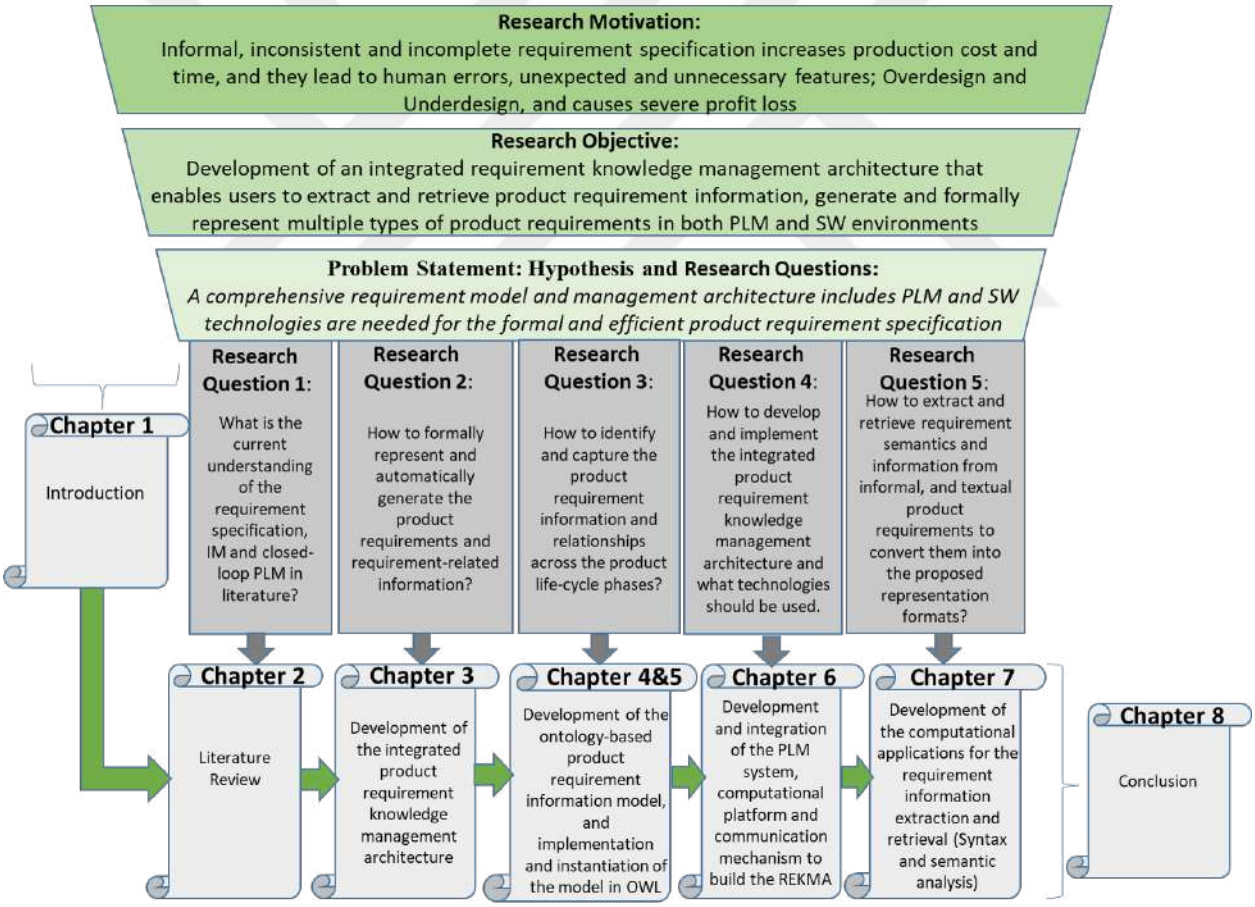


Figure 1.6: The Overall Structure of the Dissertation

# CHAPTER 2

# LITERATURE REVIEW AND BACKGROUND

This chapter highlights the gaps and weaknesses of the current requirement specification methodology, requirement management tools, and techniques. Ontology and PLM which serve as main foundations in our proposed methodology are also examined. Besides that, normally as a precursor of the Ontology model, an Information Model, which is a prevalent study area to identify and support product-related information, is presented as well. In the following paragraphs, some significant research works are discussed concerning two topics: (i) Information Model & Ontology and (ii) Requirement Management & Related Technologies

## 2.1 Information Modeling and Ontology

Information modeling [15], which identifies the concepts and relationships between these concepts in a domain of interest, originally comes from software engineering for building system architectures. It is applied in several domains including manufacturing and product design. The advantage of using an information model in this research is that it can provide a shareable and structured information specification for the product requirements, and it is good at representing complex and interrelated product domain artifacts. An information model has great abilities to provide different generalized or abstract concepts with regard to a specific domain, different relationships between concepts, constraints, rules, and operations to specify the information of the domain context [16]. In this dissertation, we sometimes use the term Ontology, Ontology Model, or Ontology-based

Information Model instead of the Information Model (IM).

A tremendous effort is devoted to Information Modeling by the National Institute of Standards and Technology (NIST). Several examples include the NIST Core Product Model (CPM) [17], which intends to capture the full range of engineering information that is commonly shared in product development. NIST has also developed an Open Assembly Model (OAM) [18] which extends CPM by enriching the information representation with more information about the function, form, and behavior of an assembly. Another recent attempt by NIST is on the modeling of the disassembly process by highlighting the information content used for disassembly sequence identification, feature modeling, equipment modeling, and inspection process modeling [19]. Information modeling is also used in product design standards. One example of this is ISO 10303 which is an open standard for representation and exchange of the product manufacturing information[20]. It is informally known as STEP, the STandard for the Exchange of Product Model data. The STEP defines a neutral representation for product data over its entire life cycle.

Ontology is a formal description of objects and their properties, relationships, constraints, and behaviors [21]. As a natural extension of Information Modeling, ontology can model richer semantics in a domain of interest with formalized Semantic Web (SW) languages like Ontology Web Language (OWL) [22] which is based on RDF (Resource Description Framework) [23]. They define a conceptual and rigorous organization of concepts about the selected domain. Concepts such as classes, attributes, functions, rules constitute a semantic network. Ontologies mainly consist of classes and their relationship but they are not limited to taxonomic hierarchies of classes. Any axioms that constrain the possible interpretations for the defined terms can be added if needed. Ontology is a very powerful tool in the application of knowledge representation and suitable to solve problems such as diagnosis, causal analysis, discovery and design, and knowledge sharing

and reuse in engineering and manufacturing. In such a context, a significant amount of research works are already done on developing semi-automated or automated ontology applications in the field of product design and manufacturing.

### 2.1.1 Ontology in Product Design and Manufacturing

In the manufacturing area, some IMs or ontologies are developed at different levels for different purposes over the years. Kim et al. [24] propose an ontology applied to the domain of the manufacturing for assembly design. They represent an OWL-based assembly design ontology, with explicit semantic rules which are written in Semantic Web Rule Language (SWRL) [25], to differentiate ambiguous assembly and joining relations. Lemaignan et al. [26] develop MASON (Manufacturing Semantics Ontology) that presents a draft ontology for the manufacturing domain to formally capture the concepts related to the manufacturing industry. They work on the semantic sufficiency check of OWL in representing manufacturing knowledge and create their ontology model based on three fundamental concepts: Entity, Operation, and Resource. Also, they propose two implementations of their ontologies in the domains of automatic cost estimation and a multi-agent system for manufacturing. Usman et al. [27] summarize the major formal and semi-formal ontologies in manufacturing domains that are published between 1996 and 2010. They propose a Manufacturing Reference Ontology (MRO) which aims at defining a core set of manufacturing concepts critical for developing interoperable manufacturing systems. Ameri and Dutta [28] develop an ontology for Manufacturing Service Description and introduce the Manufacturing Service Description Language (MSDL). It provides the primitive building blocks required for the description of a wide spectrum of manufacturing services to demonstrate an ontological approach for the representation of manufacturing services. A rule-based extension of

MSDL based on the SWRL is introduced by Ameri and Mcarthur [29] to enhance the expressivity of MSDL. It supports the automated inference and reasoning capabilities of the search engine. MSDL covers technological aspects of manufacturing capabilities and needs mainly focus on mechanical machining services such as machine tools, surface finish, etc. Ameri et al. [30] propose the metal casting process extension to the MSDL to represent the manufacturing capability of the MSDL. It shows that developed ontologies can be extended to the sub-domains of the interest. In [31], Giovannini et al. present an ontology-based system to support manufacturing sustainability. The system acts as a sustainable manufacturing expert that can automatically identify a sustainable production scenario.

Graves [32] draws attention to the reasoning capability of ontology engineering for product design. In [33], Wang et al. propose an ontological approach to support design decisions effectively and efficiently. They analyze what kind of manufacturing knowledge is needed for the product design decision. Besides, ontology models are developed to eliminate human interaction in the engineering product analysis which is performed right after creating geometry modeling of the product and assigning materials [34, 35, 36]. These analyses include Design for X (DFX), sustainability, Finite Element Method (FEM), etc. Creating an automated product analysis framework by defining design semantics and mapping it into product analysis provides benefits for engineering product design by enabling autonomous product requirement validation and execution. It also assists users to solve a complex engineering problem without having a high level of expertise to understand a requirement specification.

A different study for ontology in the manufacturing domain is published by Benjamin P., Patki M., and Mayer R. to show the advantages of the ontology management methods and tools for modeling and simulation [37]. They outline the technical challenges in distributed simulation

modeling and show how to deal with these challenges by applying ontology-based methods. They highlight the critical roles of ontology in simulation modeling.

In product design, material selection is another area of study for ontology engineering. It contributes to the development of IM for product design. Usually, the main goals of material selection are to minimize weight, cost, and environmental impact, etc. while meeting the requirements of a product. One of the notable methods for material selection is developed by Michael Ashby [1]. Besides this, there are other material selection studies using ontology. Ashiso and Fujita [38] propose a web ontology method for design-oriented material selection which formalizes the process of mapping material properties into material substances. They prove their method with an example of the mapping creep property into the design data. Another study which is an ontology-based knowledge framework for automatic material selection in the engineering domain is presented by Zhang et al. [39]. They propose an ontology which uses Semantic Query-enhanced Web Rule Language (SQWRL) and supports knowledge retrieval and reason approach for material selection.

### 2.1.2 Ontology Based Approach for Product Lifecycle and Knowledge Management

The current solution for the management of product data and knowledge within the product life is enabled by the product life-cycle management (PLM) system that is the process of managing the entire lifecycle of a product from the beginning, through engineering design and manufacturing, to service and disposal of products. However, PLM itself does not strongly support product data semantics because the PLM has document-based manipulation. It might create interoperability and communication issues between designers, analysts, operators, and tools which are involved in product life-cycle activities [40]. To support the sharing and re-use of modeled knowledge within

the PLM systems, it is useful to define a common vocabulary that can be structured based on an ontology for shared knowledge.

Advanced technologies like the internet, wireless communication, and product embedded information device (PEID) such as RFID, sensors, on-board computers, etc. make product life-cycle data and information visible and controllable across the whole product life-cycle. The PLM with these technologies becomes a much more powerful system that can access, gather, store, manage, and control the product-related information. Especially, the PLM can manage the information during MOL and EOL [41]. It allows all the actors who play a role during the life-cycle of a product such as managers, designers, service and maintenance operators, recyclers, etc. to track, manage and control product information at any phase of its lifecycle [42]. This system, called Closed-loop Lifecycle Management (CL2M) [43] allows users to control information flow across product life-cycle activities and analyzes gathered product-related data to create knowledge and make some decisions. The outcomes go back to the users, designers, manufacturers, suppliers, service operators, etc. so that the information flow can stay in the loop and be closed through the whole product life-cycle [44]. However, there is a need to standardize the knowledge representation and integration for the PLM product development solutions. Ontological knowledge-based approaches are proposed to fill this need. Recent works dealing with ontologies show that ontologies have an important role in the field of the PLM for the information exchange process. They propose a PLM solution strategy for semantic technology. In [45], Zhan presents an ontology-based knowledge architecture to integrate different CAD/CAE programs for the PLM applications.

Jun et al. [46] develop an ontology model to design the structure of metadata that describes the contents of product life-cycle data in the Closed-Loop PLM. Matsokis and Kiritsis [47, 48] propose their works for ontology in the PLM to deal with the lack of reasoning capabilities as well as lack

of interoperability and integration of elements of today's PLM systems and models. These studies encourage us to consider the requirement specification under the context of the Closed-Loop PLM because the requirement information flows back and forth through the product life-cycle phases.

## 2.2 Relationships in the Product Design Domains

In order to understand the relationship between design requirements and product related domains such as form, material, and process, some studies are reviewed in this section. Some manufacturing process activities are emphasized here because the manufacturing process is influenced by both material and form and vice versa. Many factors influence each activity of the manufacturing process. These factors must be well known and controlled during the process to create a product with desired properties. They must be indicated in the requirement specification as product requirements and considered during a product's conceptual design stage. These factors are reviewed because they are used to develop an expert system as a part of the REKMA in this research.

In a powder metallurgy process; compaction pressure, sintering temperature, compaction and sintering time, etc. are some of the factors that have effects on final product material properties [49, 50]. At the same time, powder type and characteristics are the factors that must be considered to define compaction type, sintering temperature, sintering atmosphere, etc. [51]. Injection molding is another manufacturing process that also shows some dependencies among material, form, and process. Many experimental studies investigate the effect of injection molding process parameters on material properties especially the tensile strength of the final product. Results show that process parameters such as bulk material properties, melting temperature, injection pressure, cooling time, and holding time have effects on properties of injected molded product [52, 53]. Holding pressure, injection speed, and melting temperature significantly affect the tensile strength of material [54,

55]. Process parameters not only affect the material properties but also affect the shape characteristics of the molded product. The mold temperature, melting temperature, packing pressure, packing time, and injection time are the process parameters that have effects on the surface quality of the molded product [56, 57, 58]. One of the critical problems related to the shape of the injection molding product is shrinkage. Studies show that shrinkage can be minimized by setting optimal process parameters which are basically melting temperature, injection pressure, refilling pressure and cooling time [59, 60]. Another study shows that injection speed is the main factor for the weld lines [61]. These factors are embedded in the REKMA as rules providing requirement information for users to determine additional product requirements.

## 2.3 Product Requirement Management

Requirements representation is a very difficult job that needs detailed study on related domains because of its complexity. In order to develop an ontology-based design requirement model, first, entities and relationships between those entities for the product requirements are needed to define. The below works help us understand the model-based studies on the requirement specification and develop a requirement model. Current techniques that are used for product requirement management are also reviewed below to define requirement model classes and relationships.

Requirement management consists of a set of technologies that ensure the validity of requirement documents, and satisfy the needs and expectations of customers and other stakeholders in different phases of the product life cycle. It has mainly three activities; requirement elicitation (requirement extraction and storage), analysis (classification and prioritization of requirements), and specification (modeling and specification of requirements). Several notable research works on this topic are published over the years.

The most popular technique for requirement specifications in organizing customer and technical requirements is Quality Function Deployment (QFD) [62] which is developed in the late 1960s to early 1970s, in Japan, by Yoji Akao. This management tool is used for a visual representation of mapping customer needs into appropriate engineering requirements. Later, the traditional QFD approach is extended to the Fuzzy Quality Function Deployment (FQFD) [63]. It incorporates the uncertainty issues of the requirements. Besides QFD and FQFD, Stauffer and Morris [64] discuss a taxonomy-based approach (MOOSE) for eliciting customer requirements. Similarly, Rounds and Cooper [65] study product requirements using taxonomies of environmental issues. Another approach to translating customer requirements into the design technical attributes is the Customer Optimization Route and Evaluation (CORE) model [66] which is developed by Mousavi and his colleagues. This model considers the interaction between design and market needs during the process of translating requirements. One of the notable studies for requirement specification is carried out by McAdams et al. [67]. They propose a matrix-based approach to identify the relationships between product functions and customer needs. These research works encourage the use of taxonomy for product requirements and model-based requirement specification; however, their works identify only a specific part of the product requirements, and the overall structure of the requirement specification is lacking in their models. There are many standalone tools developed to help users in requirement generation and representation. The product requirements can be represented with different forms using textual and non-textual notations. However, the needs of the stakeholder are usually expressed through the requirement statements written in natural language that is formed by human written language. While this way of textual requirement representation is easily generated and read, it creates instability and risk and it is considered informal representation. In order to increase precision and minimize problems associated with natural language within a

requirement sentence such as ambiguity, quality, etc., use of non-textual notations are promoted with proposed representation and analysis tools such as graphical notations like UML and SysML, requirement management tools like IBM Telelogic Doors and formal notations such as Z notation [68]. However, the representation in non-textual notations requires information translation from the diagrams to natural written language (textual notations) to compose requirement statements because non-textual notations might be difficult to understand by most people without having a background knowledge in systems or software. The most preferred techniques to create quality, unambiguous, consistent and complete textual requirements is the use of formalized requirement syntaxes and break up the statements into their various entities [69] which are the alternatives to the plainly written text. ARIES [70] for representation and presentation of requirements knowledge is one of the earliest projects. IBM Requirements Management tool, DOORS [71] is one of the popular requirements management and visualization software in today's marketplace which can transform unstructured texts of requirements into structured texts. It also enables users to capture, trace, analyze, and manage changes to information while it is interoperable with other tools, including life-cycle management, team collaboration, and systems/software engineering. Another requirement management tool, ReQtest [13] is a fully cloud-based tool that is used to manage requirements such as functional, non-functional, or any other requirements. Besides these, TESSI [72] is another notable requirement management tool. Different than the above tools, the TESSI composes of ontology-based components to transform the requirements specifications into a UML model. Kroha et al. [73] indicates that it is one of the earliest attempts for consistency checking of requirements and validation of the requirements specifications by using Semantic Web technology, and autonomous requirement texts generation by using Natural Language Process (NLP) advantages.

### 2.3.1 Ontology in Product Requirement Management

Ontological approaches are becoming popular to represent and interrelate many types of knowledge in the Requirement Engineering (RE) area. Many existing ontology-based studies are conducted for requirement modeling in RE [74]. Lee and Gandhi [75] develop an ontology-based active requirement engineering framework, Onto-ActRE that integrates various RE modeling techniques to analyze, represent, and model software-intensive systems and also to deal with their complexity. Another work in the domain of product requirements is done by Farfeleder et al. [76]. Their work proposes a guidance system for requirements elicitation based on domain ontology. In [77], Sommerville proposes an ontology-based framework for supporting semantic-based RE. Kücherer [78] utilizes ontologies to improve the quality of software requirements specification that contains many individual requirements and show how RE can be improved with domain ontologies. In another recent study Guizzardi et al. [79] use ontology for requirement elicitation and provide an ontological interpretation of the non-functional requirements while distinguishing between non-functional and functional requirements. It is grounded on the Unified Foundational Ontology (UFO) [80]. Kaiya and Saeki [81] emphasize the importance of domain knowledge and domain ontology on eliciting requirements and discuss the quality of requirements such as completeness and consistency with their proposed Ontology-based Requirements Elicitation (ORE) method. Avdeenko and Pustovalova [82] implement their proposed ontology that is designed for the verification of requirement specification and the completeness and consistency of the requirements in Protégé ontology editor. OntoReq [83] allows users to generate requirements while formalizing related knowledge. It provides automated validation and measurement for the requirement knowledge.

One of the earliest studies in the ontology-based requirement knowledge representation is discussed in [84]. They propose a requirement ontology for engineering design that captures design knowledge and represents design requirements to supports a generic requirements management process. In the NISTIR report, Weissman et al. [85] discuss a formal model to represent product requirements. This model relates product requirements to the design solution which is represented by CPM and OAM. They used many domain model taxonomies such as product, material, and function. Some of these taxonomies also are used in this dissertation to build the REKMA. While this model is unique for requirement representation, it still needs autonomy to build requirement specifications and capture requirement-related information. In order to provide requirement knowledge inference in the Requirements Engineering process, Riechert et al. [86] present a semantic structure for capturing requirements relevant information and developed requirements engineering ontology (SWORE) that use and interlinking with domain ontologies. Kaiya & Saeki [87] proposed a domain ontology technique for software requirements analysis that contains domain-specific concepts, relationships, and inference rules. Inference rules are used to process a requirements document semantically. Their approach detects requirement problems such as incompleteness and inconsistencies by automating semantic analyses with lightweight NLPs.

These reviews show that RE activities such as requirement elicitation, verification of requirements, completeness and consistency check, and requirement specification are implemented using ontologies and other related technologies in many studies. These studies point that the model-based representation of requirements using ontologies instead of the requirement document written in a natural language offers a more effective requirement specification, analysis, and validation. However, there is still a need to support model-based requirement specification with the requirement information extraction from a requirement document to represent them in the proposed models.

### 2.3.2 Linguistic Methods and Information Extraction in Requirement Management

In this dissertation, a computational application that is part of the proposed requirement knowledge management architecture is developed to extract requirement information from the textual requirement description. The processing of unstructured or semi-structured product requirement and related domain documents are studied to extract and structure product requirement information through natural language processing (NLP), information retrieval (IR), and information extraction (IE). Some of the studies that propose information extraction and retrieval models in the engineering domain using the above technologies and ontology are reviewed in this section. NLP tools and machine learning (ML) techniques are used to extract certain structured types of information from unstructured and/or semi-structured natural language text. Structures of the extracted information are defined with ontological models. Information retrieval (IR) that is integrated by an ontology model attempts to analyze text and extract their semantic contents [88, 89]. In many studies, authors suggest that ontology should be part of the information extraction and retrieval [90, 91]. Ontology-based information extraction and retrieval have three main activities; syntax and semantic analysis of domain knowledge, domain ontology development, and semantic rules generation. In [92], Oro and Ruffolo discuss the development of a system that processes PDF documents and presents the output in the form of an ontology. Similarly, many other studies indicate ontology has a great potential for automatic processing of the information in the natural language text and creating semantic contents.

In the manufacturing domain, ontology-based information extraction and retrieval are studied by [61]. This study describes a framework for design information extraction and retrieval and aims to automatically construct a structured and semantics-based representation from unstructured

design documents based on predefined design information. In another study, these authors propose an ontology-based query processing to improve the performance of design information retrieval [93]. In this dissertation, a similar process is followed to build the ontology-based information extraction and retrieval model for the product requirement knowledge retrieval. Verma and Kass [94] develop Requirements Analysis Tool (RAT) that automatically performs a wide range of syntactic and semantic analyses on requirements documents using domain ontologies. This tool enables users to write requirement documents in natural language based on standardized syntax. Similar to this study, requirement syntaxes will be discussed and used for autonomous requirement text generation in chapter 5.

Another powerful technique which is the core of our ontology-based information extraction application is Named-Entity Recognition (NER). NER is a subtask of information extraction and it seeks to locate and classify elements in text into pre-defined categories that are identified in domain ontologies. Ontology-based named entity recognition, and information extraction are used successfully in different domains such as the business intelligence domain [95]. Saggion et al. propose an ontology-based extraction and merging in the context of a practical e-business application. Yasavur et al. [96] develop a behavioral health ontology and design a named-entity (NE) recognizer to identify the lifestyle change information. Their named entity recognizer can automatically tag words and phrases in sentences about the lifestyle with the pre-defined names contracted in the ontology model. Similarly in this thesis, domain-specific tags of the product requirements such as product, function, flow, material, shape, etc. are identified in the proposed requirement ontology and a requirement named entity recognizer is created to automatically tags the requirement words.

## 2.4 Summary and Limitation of Current Studies

All studies, requirement management tools, and techniques reviewed above show that the use of ontology, NLP, and AI are growing fast in information extraction and knowledge management. They encourage the model-based representation and generation of the product requirements instead of the document-based specification. Information representation and integration are very difficult jobs in product design domains. They should be studied in detail because of their complexity. While in the beginning, designers/engineers start the requirement specification using customer requirements, but they must deal with very complex, detailed, and interrelated product requirements such as performance, environmental, finance, manufacturing, and recycling, etc. at the further design and product life-cycle stages. They must also consider the relevant laws and regulations when they are generating product requirements.

Our research shows many tools and techniques are developed for product requirement management. Some of their advantages and limitations are discussed in table 2.1. This review shows that a structured requirement data model for the automatic requirement knowledge management and requirement specification is still lagging. It should help the designers/engineers to identify and analyze the product requirements, represent requirements in a structured form, inference requirement information by using the relationships between product domains, and support requirement generation by showing design rationale. One key component towards the development of such a model-based approach needs the incorporation of the Information Modeling, the PLM, and the NLP into the product requirement specification.

In the current product requirement management, the development of the requirement specification has the following limitations;

Table 2.1: Overview of Requirement Analysis Tools & Techniques

| Tool&Technique | Scope | Limitation |
|---|---|---|
| SysML [97, 98] | Formal description supported by graphical models and textual description | • requires training<br>• does not support autonomous text generation which requirement text must be manually generated through capturing information from other diagrams<br>• requirements are mainly expressed using natural language |
| QuARS [99] | Semi-formal description supported by structured textual description | • only works within controlled natural language requirements<br>• does not support requirement generation and graphical representation<br>• only supports phrasal analysis |
| RAT [100] | Semi-formal description supported by structured textual description and graphical models | • requires training<br>• time consuming<br>• does not support requirement generation |
| KAOS [101] | Formal description supported by graphical models | • does not provide any taxonomy<br>• time consuming<br>• requires users to write requirements in formal description |
| TESSI [102] | Formal description supported by graphical models | • graphical model generation is limited<br>• does not support requirement generation |
| DOORS [71] | Semi-formal description supported by structured textual description | • highly customizable<br>• requires training<br>• neither build any models nor generate requirements |

- Limited autonomy and formalization in the requirement generation and representation in the product design and further life-cycle phases that require:
  - Formal representation of the product requirements
    - Both textual and ontological representations
    - Representation of the product requirements with requirement rationale

  - Autonomous requirement generation
    * Inference of the requirement information
    * Requirement text generation using formalized syntaxes

- Limited visibility of the product requirement information in the MOL and EOL phases that requires:
  - Requirement information extraction using product MOL and EOL data
  - Support for the requirement specification process by analyzing the product MOL and EOL data
    * Performing the requirement specification within the closed-loop PLM concept.

To help designers/engineers and support product life-cycle activities in terms of the requirement generation and representation, an ontology-based integrated requirement management architecture within the closed-loop PLM concept is developed in this work. It incorporates semantic knowledge of the product requirements in the product life-cycle activities, syntax, and semantic analysis using NLP tools, and computational applications supported by AI techniques.

# CHAPTER 3

# MODEL-BASED PRODUCT REQUIREMENT REPRESENTATION AND GENERATION METHODOLOGY

In this section, the REquirement Knowledge Management Architecture (REKMA) that supports the formal representation and generation of the product requirements is proposed and discussed. This chapter addresses the first research question (How to formally represent and automatically generate the product requirements and requirement-related information?) and describes the methodology of the model-based requirement representation and generation. This chapter first discusses the overview of the proposed REKMA. Then it presents the roles of the ontology-based information model, PLM system, and computational platform for the REKMA. Finally, it explains the proposed product requirement representation and generation procedures.

## 3.1 Requirement Knowledge Management Architecture

Every routine product design generally starts with the specification of a set of requirements. If these requirements are already formalized into a set of mathematical equations and constraints, it will be easy to represent them; however, requirements often remain implicit and are difficult to formalize. Most of the requirements represented in a specification document come from three sources: (1) from the product user who describes the requirement quite broadly, mostly in an informal way; (2) from the "real" world constraints; the requirements come from the domain knowledge of the product design, manufacture, product usage and maintenance, and disposal; and

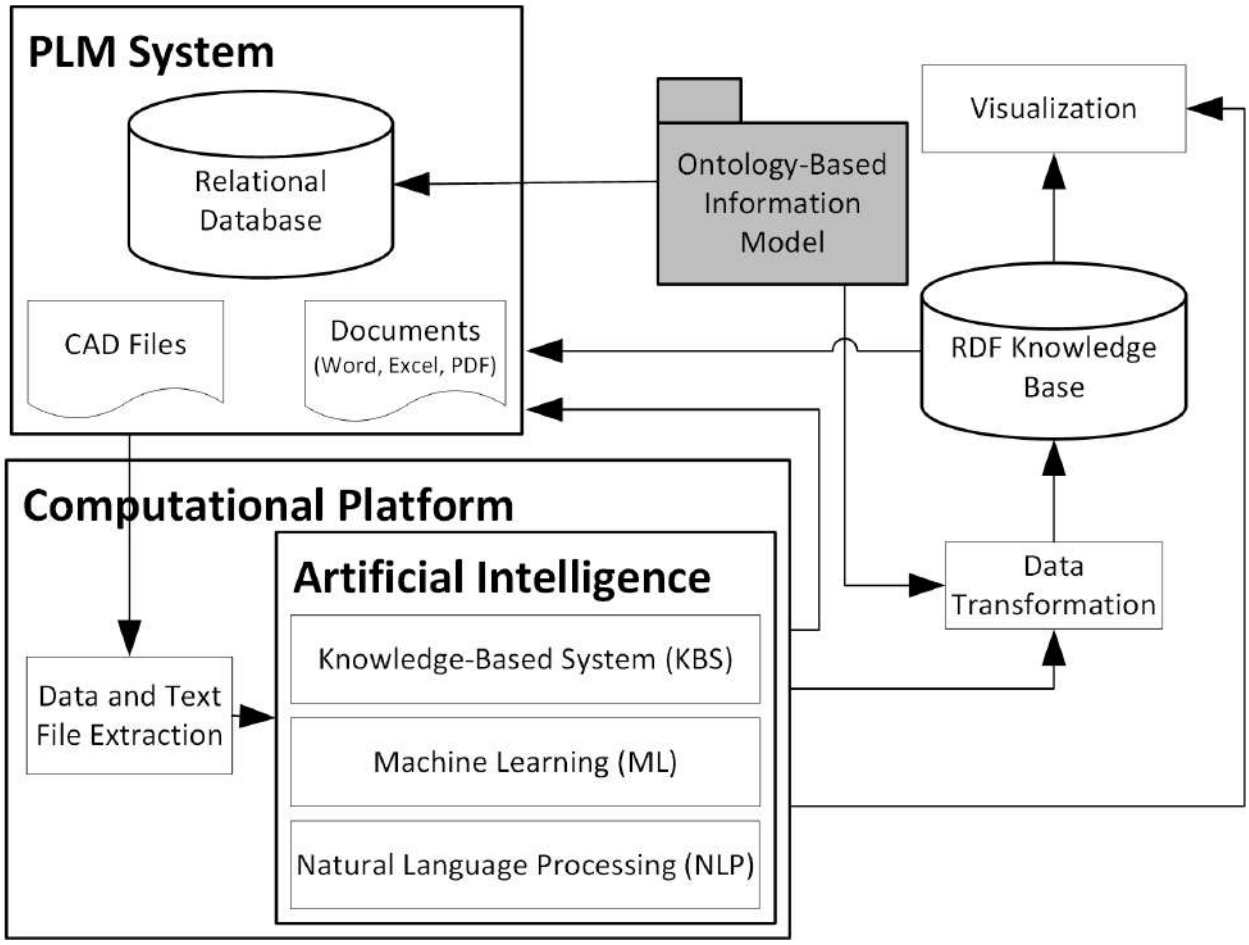


Figure 3.1: The Requirement Knowledge Management Architecture

(3) from the product synthesis and analysis phases. The requirements generated during product synthesis and analysis phases must satisfy other necessary product requirements that come from source (1) and (2) to control the design procedure. It is hard to capture all requirements while maintaining consistency between them. Therefore, any formulation of requirement specification remains difficult and becomes very knowledge-intensive. A knowledge-based approach is used to support the product requirement representation and generation. It translates an informal, vague requirement statement to a set of formal criteria.

In this dissertation, a REquirement Knowledge Management Architecture (REKMA) as shown

in Figure 3.1 is proposed. It captures product requirements and related information within the data structures of the proposed information model(s), manages them within the PLM system and computational platform, and stores them as both graph data model and relational format. In this architecture, ontology-based information models are developed to define product requirement classes and relationships between classes. They provide a huge set of predefined requirements metadata for the requirement specification. The PLM system is used as a data source and a user interface for the requirement knowledge management. It enables users to access and manage product requirement data and documents. The computational platform is created to convert any product requirement inputs into the proposed requirement representation format. It generates requirement information from the requirement and related data stored in the PLM. It also generates requirement texts by using the generated requirement information to support the development of the requirement specification document. It then helps designers/engineers to analyze and validate product requirements by checking the completeness and consistency of the requirements. Finally, this architecture has two phases to complete the requirement information flow; data and file transportation between the PLM system and computational platform and data transformation to make requirement data available in the Resource Description Framework (RDF) format.

I-Semantec platform [103] which studied the necessity of PLM and SW integration is inspired to develop the REKMA. Some reasons are discussed here to explain why this architecture is needed for requirement management. First of all, the relationships between product requirements and product life-cycle data might be complex. An ontological model, which is one of the major parts of SW technology is a good candidate to represent these kinds of relationships. Secondly, product requirement management across the product life-cycle phases needs access to many product-related data. PLM is the current solution for the management of product data and

knowledge within the product life-cycle. So, the PLM is a good candidate for the management of the product requirements and related data. Thirdly, the challenges, such as interoperability, knowledge exchange, and re-use of the product information are major problems in the PLM. Semantic Web technologies support knowledge sharing, data transfer, and information re-use. Ontologies, part of the semantic web, can be used to bridge the data interoperability gap between various software systems and support communication through product life-cycle. Lastly, the semantic web provides better cooperation between humans and machines with the interpretable semantic contents of the product requirements [104]. In order to address these issues and to manage (identify, capture, validate, store, distribute, and maintain) product requirement information in the product design and during the product life-cycle phases, REKMA is created by bringing PLM and Semantic Web technologies together for the formal representation and generation of the product requirements.

The descriptions of the ontology-based information model, PLM system, and computational platform are narrated in the following sections.

### 3.1.1 The Role of the Ontology-Based Information Model in REKMA

As discussed in chapter 2, the information model identifies the concepts and relationships between these concepts in a domain of interest. A comprehensive requirement information model is developed in this dissertation and discussed in chapter 4. Three types of product domain knowledge are needed to develop a requirement information model:

i. Domain meanings of the product requirement concepts and relationships

ii. Physical and behavioral information for product-related domains that provide insights regarding product requirements

iii. Descriptions and rules about how domain models interact with one another

To incorporate all these types of knowledge into the product life-cycle and to enable the interoperability and traceability of them, information models should be explicitly and formally captured and linked to the computational platform and PLM system.

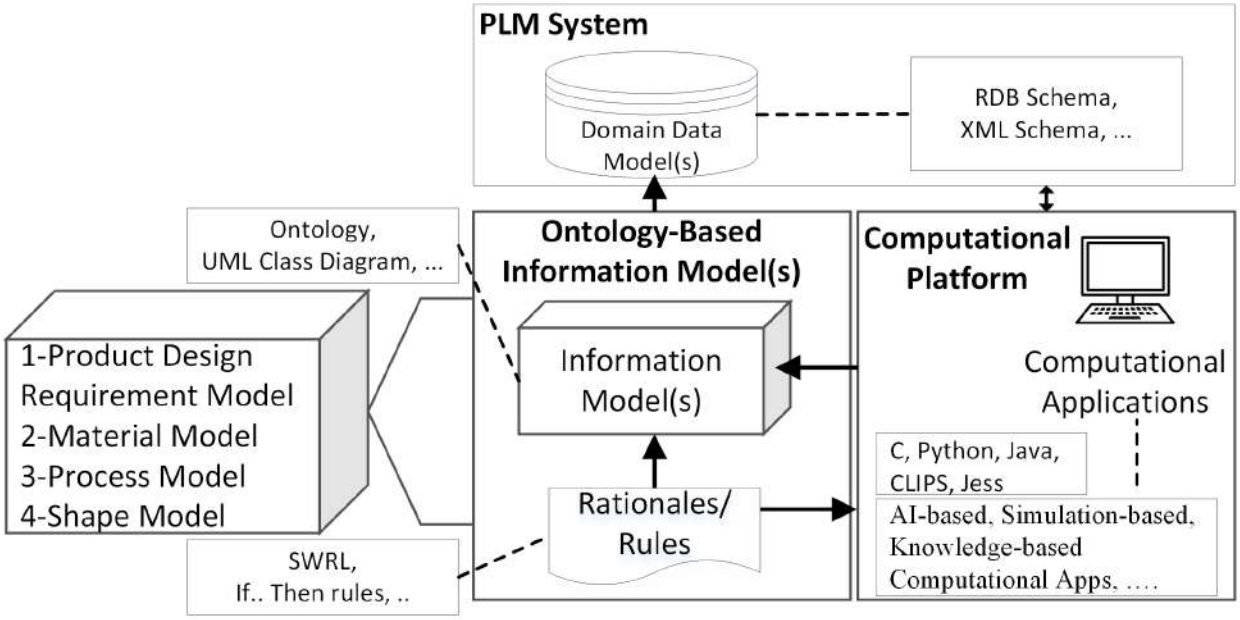


Figure 3.2: Information Models in the Requirement Knowledge Management Architecture

In this dissertation, product requirement related information, knowledge, and relationships are captured into the information model(s) and rationales as shown in Figure 3.2. The rationales are used to describe the rationality of product design and other life-cycle information with the product requirements. They are used to support information models for obtaining the semantic meaning of the domain concepts. The rationals are implemented in the computational applications to reveal the behavioral knowledge of a product in the requirement representation and generation processes. Rationales of the product requirements and related information are formally represented as rules to make them machine-processable and understandable. The SWRL (Semantic Web Rule Language)

[25] language is used to build rules in OWL. The rules can be implemented in CLIPS and Jess rule engines.

Here, the ontology-based information model(s) provides a common terminology for the application domains that are the representation of product requirements and requirement generation. The ontology-based information model(s) also represents the graph data structure for the RDF knowledge base. RDF is a suitable language for the modeling of the product requirement and requirement-related data because it can describe the requirement entities and the relationships with a simple modeling approach. RDF knowledge base stores the product requirement data for the SW applications. The detailed descriptions of the proposed information models for the product requirements and related domains are presented in chapter 4 and their formal implementations are discussed in chapter 5.

### 3.1.2 The Role of the PLM System in REKMA

The current solution for the management of product data and knowledge within the product life-cycle is enabled by the PLM systems. The PLM is the process of managing the entire life-cycle of a product from beginning to end through product design, manufacturing, service, and disposal phases. It is a powerful tool that can work with many product development technologies to gather, store, and manage product-related information. However, the PLM itself is not strongly linked to the model-based representation of the product data because PLM has document-based manipulation. This issue of the PLM might yield interoperability and communication problems between designers, analysts, operators, and tools that are involved in product life-cycle activities [40].

In PLM, all product life-cycle data and information are made visible and controllable over the

whole product life-cycle using advanced technologies like the internet, wireless communication, and product embedded information devices (PEID) such as RFID, sensors, on-board computers, etc. The PLM under these technologies became a much more powerful system that can access, gather, store, manage and control product-related information, both before and after a product delivered to the customer and up to its final destination [41]. It allows all the product stakeholders, such as managers, designers, engineers, service and maintenance operators, recyclers, etc. to track, manage, and control product information at any phase of its life-cycle [42]. This PLM system is called the Closed-loop Life-cycle Management (CL2M) [43] that allows stakeholders to control requirement information flows across product life-cycle operations and analyze product-related data in order to create knowledge and make decisions. Every downstream decision/outcome is then feedbacked to the product's earlier life-cycle phases so that stakeholders can access those decisions/outcomes. All information is now available in a closed-loop and product information flows back and forth through the whole product life-cycle [44]. Every requirement specification document as a whole and each individual requirement in the document have their life-cycles across the product life-cycle. Unlike traditional PLM systems [105], CL2M in requirement knowledge management focuses on data management and information flow, and supports the generation and reuse of the product requirement knowledge. It enables all stakeholders to access product requirement-related information on-demand throughout the product life-cycle phases.

In this architecture, the PLM system is used as a data repository, a knowledge source, a files storage, a user interface that the user can enter and access to product requirement related data and documents. It is also used as a management tool to control requirement information flow across the product life-cycle phases. In this dissertation, a PLM tool (Aras Innovator) is further enhanced using the data structure of the proposed requirement information model and integrated with a

computational platform for the requirement information extraction and retrieval. The detailed descriptions of the PLM system are presented in chapter 6.

### 3.1.3 The Role of the Computational Platform in REKMA

Another main part of the REKMA is the computational platform. It has three essential phases: extraction, recognition, and information exchange. The computational platform extracts and retrieves the product requirement semantics from the product data stored in the PLM system. The types of data stored in the PLM system and processed in the computational platform are product requirements and requirement-related data such as requirement text, bill of materials (BOM), manufacturing, assembly, or part details, etc. The goal of the extraction and recognition phases is to reorganize and formalize the product requirements and related knowledge to enable a formal requirement representation and autonomous requirement generation. The computational platform supports the two-way data exchange between the computational platform and the PLM system. It also translates the product requirement data and metadata from the PLM into the RDF knowledge-base. To support the data exchange among the computational platform, the PLM system, and the RDF knowledge-base, a communication mechanism is developed.

In this system architecture, analytical applications are created and used in the computational platform to support the requirement representation and generation. These applications are able to generate requirement information and knowledge from the product requirements and related data, and represent them in the proposed format. Requirement information and knowledge generation increases the reliability of product requirement specification and provides a better insight into the product requirements to all the stakeholders. The main analytical application developed for the computational platform is the product requirement information extraction and retrieval. It

is developed by using natural language processing (NLP) technology. The development of this application combines two main studies; semantic analysis of product requirements through the ontology models and syntax analysis of the product requirements. A Named Entity Recognition (NER) model is also created to support this application by capturing product requirement concepts automatically in a requirement sentence. Controlled requirement syntaxes are discussed next to automatically generate textual requirements.

The detailed descriptions of the computational platform and communication mechanism are presented in chapter 6 and the development of the requirement information extraction and retrieval application is discussed in chapter 7.

## 3.2 Formal Representation and Generation of the Product Requirements

In current requirement specification practice, product requirements and related data are represented most commonly with textual descriptions, textual documents, string, and numerical data types, and sometimes with tables. They include a series of requirement sentences and paragraphs written in natural language and arrangements of requirement data in rows and columns. For the development of the requirement specification documents, either with or without table representation, designers/engineers mostly prefer to use pre-defined requirement sentence templates and formalized technical vocabularies. They also prefer to follow a systematic method to organize the list of requirement sentences and document structure. However, as discussed in the previous chapters, traditional requirement specification documents remain not suitable for automated machine processing to analyze and manage the product requirements listed in the documents.

If the requirement data are represented in a table format with string and numerical types, not as a textual format, they can be processed by computational applications without applying any natural

language pre-processing. As shown in table 3.1, requirements for a brake rotor are represented in a table format. This table contains partial examples of a brake rotor requirement specification. It includes category names such as requirement characteristics, value, unit, etc. that are called *requirement entity types* and category values such as cooling time, weight, high, minute, etc. that are called *requirement entity names* in this dissertation. Even though this type of representation is much more suitable than textual sentence representation for automated machine processing, it must be supported with formalized vocabularies (taxonomies) and relationships between requirement entity names.

Table 3.1: Table Representation of the Requirement Specification for an Automobile Brake Rotor

| **Requirement Aspect** | **Characteristics** | **Value** | **Unit** |
|---|---|---|---|
| Performance | Cooling Time | <30 | Minute |
| Performance | Corrosion Resistance | High | |
| Performance | Operating Temperature | -30 - 200 | °Celsius |
| Design | Nominal Thickness | 0.40 | Inch |
| Design | Discard Thickness | .15 | Inch |
| Design | Weight | 20 - 25 | lb |
| Manufacturing | Method | Casting and Machining | |
| Maintenance | Service Life | 3 | Year |
| Maintenance | Replacement Time | <20 | Minute |

If the requirement data is represented as a textual description in a requirement specification document which is the most common preference as shown in table 3.2, pre-processing applications such as natural language process and text mining techniques become crucial for automated machine processing. Requirement entity names must be extracted from the written requirement text and classified as requirement entity types.

This dissertation concentrates on autonomously converting the requirement sentences into the proposed formal requirement representation formats and generating requirement information and

Table 3.2: Textual Description of the Requirement Specification for an Automobile Brake Rotor

| |
|---|
| **Performance**<br>• Cooling time shall be less than 30 minutes after the vehicle is parked.<br>• Rotor shall have high corrosion resistance.<br>• Operation temperature shall be between -30°Celsius and 200 °Celsius. |
| **Design**<br>• Nominal thickness shall be .4 inches .<br>• Discard thickness shall be .15 inches.<br>• Weight hall be between 20 Lbs and 25 Lbs. |
| **Manufacturing**<br>• Casting and machining operations shall be used for rotor production. |
| **Maintenance**<br>• Service life shall be at least 3 years.<br>• Replacement time shall be less 20 minutes. |

text. An NLP application is developed in this dissertation to extract terms from the requirement text and classify these terms based on the proposed ontology-based requirement information classification. When users define a product requirement in the PLM system as a requirement sentence, it is processed by the NLP application in the computational platform. Then, it is represented in the PLM system with requirement entity types and names as shown in table 3.3 in a similar way that NIST is studied in [106].

Table 3.3: The Proposed Format to Represent a Product Requirement Sentence

| |
|---|
| **Triple::** Predicate (Subject, Object)<br>**Subject Entities;**<br>Entity Type:'Entity Name'<br>Entity Type::Sub-Entity Type<br>**Object Entities;**<br>Entity Type:'Entity Name'<br>Entity Type::Sub-Entity Type<br>**Predicate Entities;**<br>Entity Type:'Entity Name'<br>Entity Type::Sub-Entity Type |

In this representation, a requirement sentence is demonstrated first in a triple format: Subject, Object, and Predicate.

***Subject*** defines the topic of discussion in a requirement sentence and tells what the sentence is about. It might be about a product, material, shape, process, property, user, environment, etc. 'Automobile brake system' as a product, 'rotor surface' as a shape, 'CO2 emission' as property, 'water pollution' as a hazard can be defined as examples of the requirement subject entities.

***Object*** corresponds to the value of subject. It is acted by the predicate. In a sentence, an object undergoes a change stated by the predicate. It might be a property of a product, material, and process, a failure, a user, a measurement, a constraint, etc. 'Density' as a material property, 'envelope size' as a product property, 'electricity' as a substance, 'corrosion' as a failure can be defined as examples of the requirement object entities.

***Predicate*** indicates the action or state in a requirement sentence. It defines a directed binary relation between subject and object. It might be a function or non-function entity. 'Resist' as a function and 'be' as a non-function can be defined as examples of the predicate entities in a requirement sentence.

After the subject, object and predicate are defined in a requirement sentence, requirement entity types of components of the sentence (entity names) are labeled with pre-defined entity tags. Then, entity type hierarchy is identified for each entity names based on ontology models and taxonomies.

Table 3.4 shows how this dissertation proposes a formal representation for a requirement text in a triple format with entity types and names in the PLM system. An example of the requirement text 'An automobile brake system shall generate friction' is represented in the table with the proposed format. First, this requirement sentence is written in the PLM system by a user. Then, a natural language processing application running in the computational platform autonomously converts this

Table 3.4: Presentation of a Product Requirement Sentence

| |
|---|
| **Requirement::** An automobile brake system shall generate friction. |
| **Triple::** shall generate (automobile brake system, friction) |
| **Subject Entities;** |
| Part:'automobile brake system' |
| Part::Automotive Part::Brake System |
| **Object Entities;** |
| Flow:'friction' |
| Flow::Energy |
| **Predicate Entities;** |
| Function:'generate' |
| Function::Function Verb::Convert |

sentence into the proposed representation format. Finally, this representation is presented in the PLM system as shown in table 3.4. A requirement NER model is created to automatically tag the requirement entity types, which are defined as requirement classes in the proposed requirement information model (PDROM). An expert system that includes semantic and syntactic rules to infer requirement entity types and names is also developed. Whenever requirement entities are created from a given requirement text, additional requirement entities are inferred by an expert system in the computational platform. Rules that only match with the entity types and names of the given requirement text are executed in the expert system to infer requirement entities. These requirement entities are called inference entities. Inference entities are represented in a format as; **Entity Type::Sub-Entity Type:'Entity Name'**. Table 3.5 shows the inference entities by an expert system in the computational platform and their representation in the PLM system for the requirement text 'An automobile brake system shall generate friction'. The computational platform is also able to create textual requirement sentences by using inference entities and formalized syntaxes as shown in table 3.6.

In order to make requirements more understandable, unambiguous, consistent, and complete,

Table 3.5: Inference Entities and Representation for a Requirement Text

| |
|---|
| **Requirement::Actor::Part:**'rotor' |
| **Requirement::Actor::Part:**'pad' |
| **Requirement::Actor::Part:**'caliper' |
| **Requirement::Actor::Material** |
| **Requirement::Actor::Shape** |
| **Requirement::Stage::MOL:**'usage' |
| **Requirement::Stage::MOL:**'material selection' |
| **Failure::Failure Mode:**'abrasive wear' |
| **Failure::Failure Mode:**'thermal stress' |
| **Failure::Failure Mode:**'thermal shock' |
| **Material::Material Property::Mechanical Property:**'wear resistance' |
| **Material::Material Property::Mechanical Property:**'friction coefficient' |
| **Material::Material Property::Mechanical Property:**'harness' |
| **Material::Material Property::Physical Property:**'thermal diffusivity' |
| **Material::Material Property::Physical Property:**'thermal conductivity' |
| **Flow::Energy:**'heat' |

formalized requirement syntaxes and boilerplates that express product requirements are studied [11, 94, 107]. These syntaxes, which are individual statements of requirements, consist of product related information entities such as product, material, function, flow, unit, key characteristics, etc. Many formalized syntaxes for the product requirements and many product requirement sentences are examined from a lexical viewpoint in this study. While syntaxes define the arrangement of words and phrases to create well-formed and structured natural language requirement statements, the lexical viewpoint examines the words and phrases used to compose the requirements [108]. Although several formalized syntaxes for both functional and non-functional product requirements are proposed, requirement syntaxes which are discussed by Lamar [109] are used when product requirement text generation is necessary. He first describes generic requirement syntax, then defines two functional requirement boilerplates for (i) Transitive Functional Requirement and (ii) Intransitive Functional Requirement, consisting of a transitive verb that has a direct object or an

intransitive verb that does not have a direct object and one non-functional requirement boilerplate as shown in table 3.6. It shows the requirement syntaxes that are used in this study. Backus-Naur Form (BNF) which is a syntactic meta-language is used to express requirement syntaxes. Four types of BNF symbols that are used to express syntaxes are: (i) brackets (<>) represent the defined entities within the syntax, (ii) two sets of colons and an equal sign (::=) separate descriptive syntactic terms from the explained term, (iii) vertical bar (|) denotes a choice can be made and (iv) curly braces ( {} ) indicates the optional element in the syntax. The requirement syntaxes are described with five main types of grammatical functions: subjects, verbs, objects, complements, and adjuncts. As discussed before, the subject is a noun phrase and it indicates what the sentence is about. The verb part includes a modal verb (shall) and a verb phrase. Direct and indirect objects are noun phrases that are acted by verbs. Complement provides syntactically necessary information about the subject or object noun phrase. Adjunct modifies words or phrases and provides additional information. In the proposed requirement representation, the verb is defined as predicate and direct object, indirect object, complement, and adjunct are defined as an object in the proposed representation format. Table 3.6 illustrates some requirement text examples that are generated by using these syntactic structures. These requirements are derived from entities of the requirement text: 'An automobile brake system shall generate friction'. This operation helps users to have a complete set of requirements represented in the requirement specification document.

The need for the formal requirement representation instead of representing requirements only with textual description in building requirement specification is addressed above. This formal specification improves the requirement understanding for the users and machines. It is pointed out that the automatic construction of structured requirement data from textual requirement description requires natural language pre-processing. NLP should explicitly and accurately capture

Table 3.6: Formalized Requirement Syntaxes and Examples for Textual Requirement Generation

| **Requirement Text Templates** |
|---|
| <**requirement**>::= |
| <subject><modal><verb phrase> |
| <**intransitive functional requirement**>::= |
| <subject><modal><intransitive verb>{<adjunct>} |
| <**transitive functional requirement**>::= |
| <subject><modal><transitive verb><direct object>{<adjunct>} |
| <**non-functional requirement**>::= |
| <subject><modal><linking verb><complement>\|{<adjunct>} |
| **Generated Textual Requirements** |
| • Rotor shall rotate with the rim. |
| <subject><modal><intransitive verb>{<adjunct>} |
| • Rotor shall resist abrasive wear. |
| <subject><modal><transitive verb><direct object> |
| • Rotor material shall have high wear resistance. |
| <subject><modal><linking verb><complement> |

the important requirement entities of requirement text. After entities are defined, entity inference can be processed by the rule-based system. Rules are defined by using relationships between entities represented in PDROM and by using expert knowledge. Then, semantic similarity and keyword-based searching processes are applied to taxonomies and PDROM data structure to find out the entity and sub-entity types. Lastly, requirement text generation using inference entities and the pre-defined text templates is processed to represent inference entities as a requirement text. Figure 3.3 illustrates these activities for the formal representation of a given requirement sentence and automated requirement entity and text generation processes. The detailed descriptions of these activities and the development of the applications to execute them autonomously that convert a requirement sentence into the proposed format, infer requirement information and generate requirement text are discussed in chapter 8.

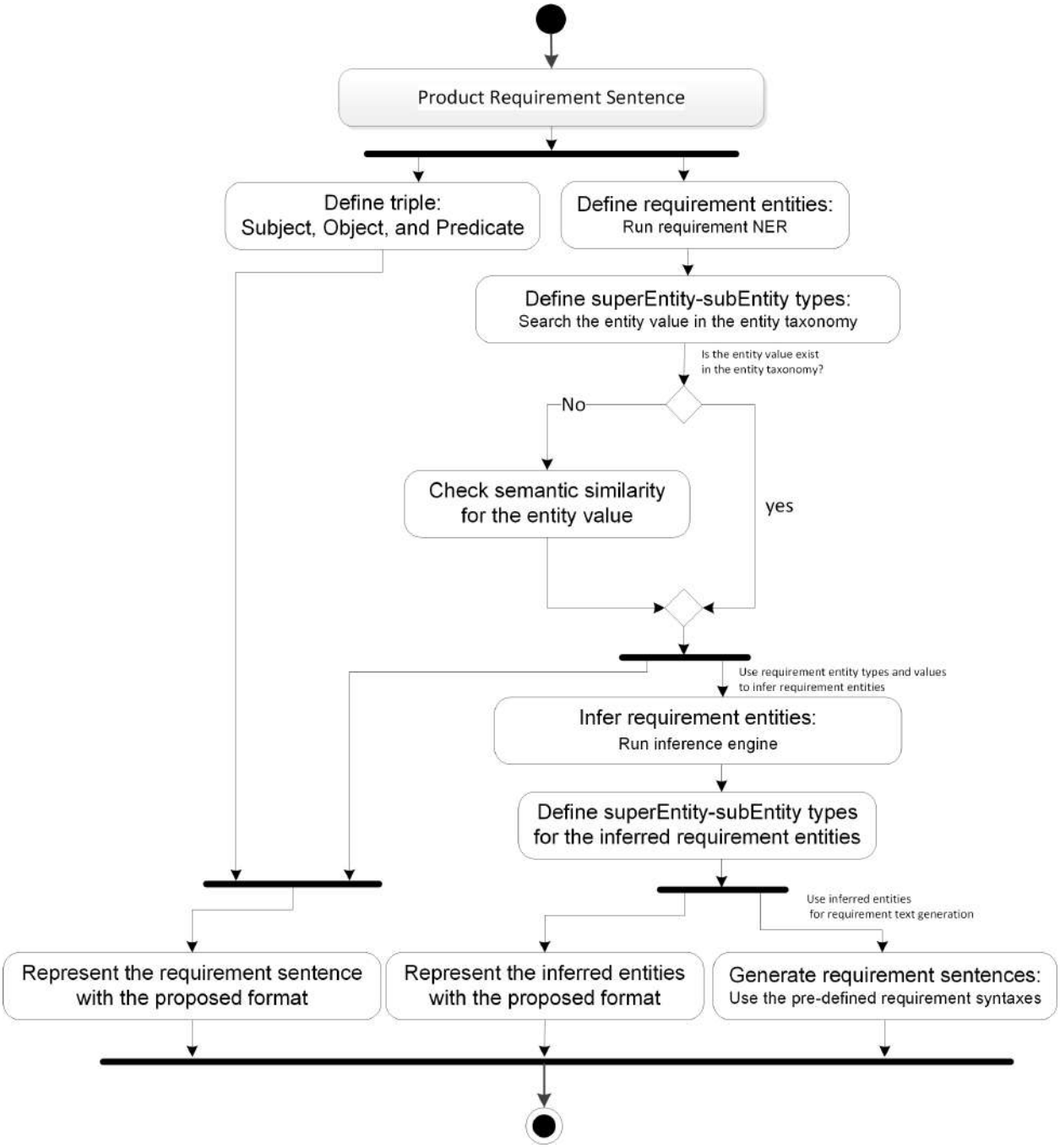


Figure 3.3: The UML Activity Diagram for Requirement Sentence Analysis Process

Automatic extraction of requirement semantics from the textual description requires mainly two types of study: (i) syntax analysis that describes the syntactic structure of requirement text and identifies entity types of requirement, and (ii) semantic analysis that reveals semantic meanings of the text based on the PDROM using NLP techniques. Requirement information extraction steps and NLP techniques that are tokenizing, part of speech tagging lemmatizing, NER, etc. are illustrated in figure 3.4. It shows that in order to make a machine-readable requirement text machine-understandable, NLP components are applied to the requirement text to practice the text mining applications such as information extraction and knowledge generation. These applications are supported by the semantic analysis of the product requirements through developing ontology models and semantic rules. Semantic and syntactic analysis and the development of these activities are discussed in the following chapters.

Requirement sentences which are described in the PLM system are also represented within the semantic format by breaking down the requirement sentence into the proposed PDROM structure. An application that converts relational requirement data into the RDF triple formats is discussed in chapter 6. The basic format of the representation of a requirement sentence in PDROM structure is proposed as shown in table 3.7. In this representation, a product requirement is defined with the requirement actor, stage, type, and measurement classes. These classes represent the instances of a requirement sentence. The requirement is also represented with requirement rationale through the structure, behavior, function, and failure models if applicable. The structure of this representation and requirement semantics are discussed in chapter 4.

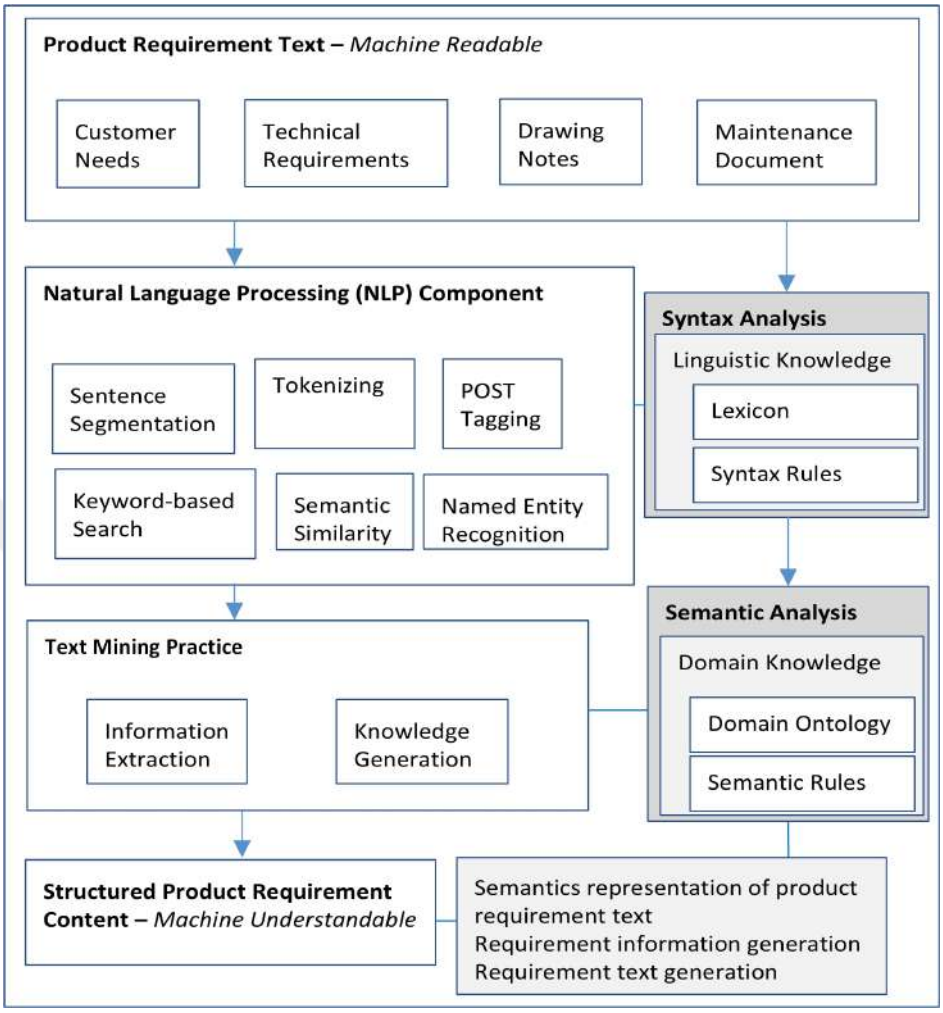


Figure 3.4: The Architecture of Requirement Information Extraction and Technologies

Table 3.7: The Representation Format of a Requirement Sentence based on the Information Model Structure

***Requirement***(requirement_id) =
***Requirement Actor***(***Sub-Class***(instance)) $\bigcup$
***Requirement Stage***(***Sub-Class***(instance)) $\bigcup$
***Requirement Type***(***Sub-Class***(instance)) $\bigcup$
***Requirement Measurement***(***Sub-Class***(instance)) $\bigcup$
***Function Model***(***Sub-Class***(instance)) $\bigcup$ ***Structure Model***(***Sub-Class***(instance)) $\bigcup$
***Behavior Model***(***Sub-Class***(instance)) $\bigcup$ ***Failure Model***(***Sub-Class***(instance)) $\bigcup$

## 3.3 Summary

In this chapter, the proposed methodology for the model-based product requirement representation and generation is discussed. An integrated architecture for the requirement knowledge management that covers the PLM and SW technologies is introduced. The roles of the ontology-based information model, PLM system, and computational platform are briefly discussed. Structured information representation and text generation templates are introduced for the formal representation and generation of the product requirements. The following chapters of the dissertation elaborately explain the topics covered in this chapter. Chapter 4 discusses the development of the ontology-based requirement information model. Chapter 5 focuses on the formal implementation of the proposed information model based on web ontology language. Chapter 6 shows how the data structure of the proposed information model is embedded into a PLM tool and how the PLM system is used for the management of the product requirement and related data. It also explains the development of the computational platform that provides integration among the PLM system, computational applications, and RDF knowledge-base. In chapter 7, the development of the applications for the requirement information extraction and retrieval and syntax analysis of the functional and non-functional requirements are discussed.

# CHAPTER 4

# DEVELOPMENT OF THE ONTOLOGY-BASED INFORMATION MODEL FOR THE PRODUCT REQUIREMENT SPECIFICATION

In this chapter, the establishment of an integrated Product Design Requirement Ontology Model (PDROM) including Material, Shape, and Process Information Models is presented in detail. Chapter starts with the general description of PDROM in the domain of product requirement specification. Next, it presents PDROM modeling methodology and modeling elements. Lastly, it introduces and represents the detailed PDROM models by using UML class diagrams.

## 4.1 General Description of the Requirement Model

The requirement information model provides a common terminology for the product requirement representation and generation. The requirement information model also captures an explicit representation of causal relationships (i.e., causes and effects of all product requirements). It represents the product design rationale that relates the product requirements to design objects to give an explanation of product requirement generation. Development of the requirement information model requires to know context information of the product requirements with regard to design, material, manufacturing, operation, maintenance, end-of-life of the product, etc. Product requirements are constructed from various product domain information which can be endless in terms of possibilities. These facts prompt us to introduce an integrated rationale-based information model to support requirement representation and generation for product design, usage, maintenance, safety,

sustainability, etc. Using this rationale-based model users are able to trace a product requirement specification to understand the requirements' dependency that may be required for any design, manufacturing, use, repair, or recycling purposes. The proposed architecture of the product requirement model, including the product domain taxonomies, design rationale, and product domain models are shown in figure 4.1.

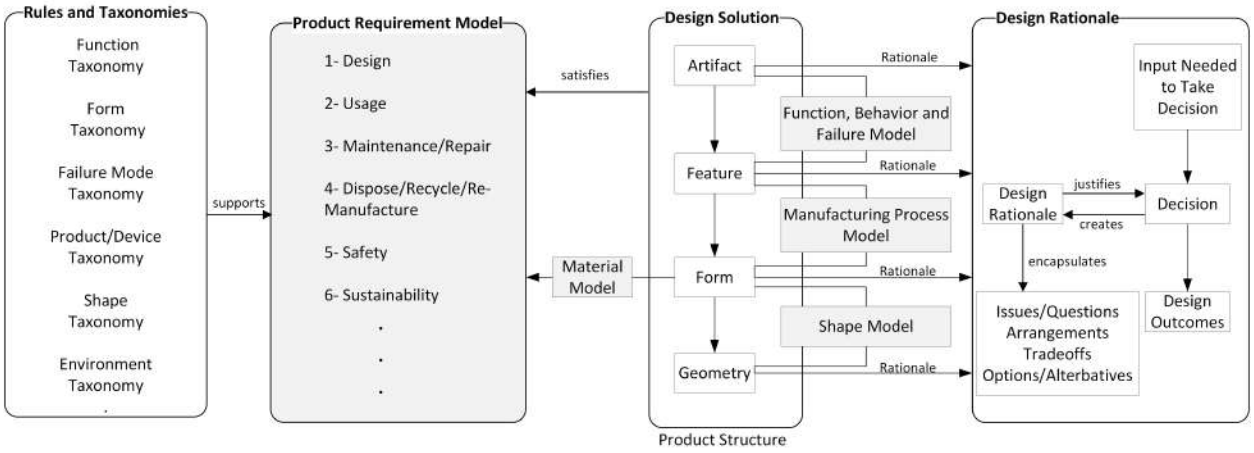


Figure 4.1: The Integrated Information Model with Design Rationale and Product Requirement Models

The reason for developing this integrated information model is to provide the "traceability" feature of the product requirement information that is needed to support understanding of existing relations and dependencies between product requirements and product domain information. Product requirements are intricately related to the design objects along with system models. The design rationale model formalizes the requirement rationale by providing background information for every generated requirement. Requirement representation and generation demand a sound understanding of the structure, function, behavior, and failure of the product assembly, and product's material, manufacturing process, shape, and also the underlying requirement rationale that will explain why requirements exist and what assumptions and constraints they depend on.

The model helps designers/engineers to generate and represent the product requirements by integrating system and product domain models and providing formalized technical vocabularies and expert knowledge with rules. In this chapter development of the product requirement model and product domain models are discussed in detail.

## 4.2 Modeling Methodology and Elements for the Requirement Information Model

In order to develop an integrated architecture for product requirements, ontology-based information models for requirements and related domains; material, manufacturing process, and shape, must be studied and developed with product life-cycle consideration. To develop these models in a systematic way, layered modeling methodology that is discussed in OntoCAPE [110] is studied. Information models are developed in a three-level framework as shown in figure (Figure 4.2). It represents product requirement information from a different level of abstraction. In this layered representation, the usability and reusability of the models are inversely correlated. The usability of the models increases from top to bottom, while reusability decreases.

In the layered framework, the **Meta Layer** is the most abstract one, represented as **Concept Abstraction** and holds the fundamental modeling concepts like Mereology (part-of relationship), Topology (connection relationship), Data_structure (array, list, graph, etc.) and Fundamental_concept (class, relationship, etc.).

A lower abstraction level, **System Abstraction** is the **Top Layer**, which is derived from the standard system theory and it is comprised of Coordinate System, Technical system and Network System. They provide a systematic structure to define concepts like system, subsystem, system boundary, etc.

Both Concept and System Abstraction layers are discussed in [110, 111]. In this dissertation,

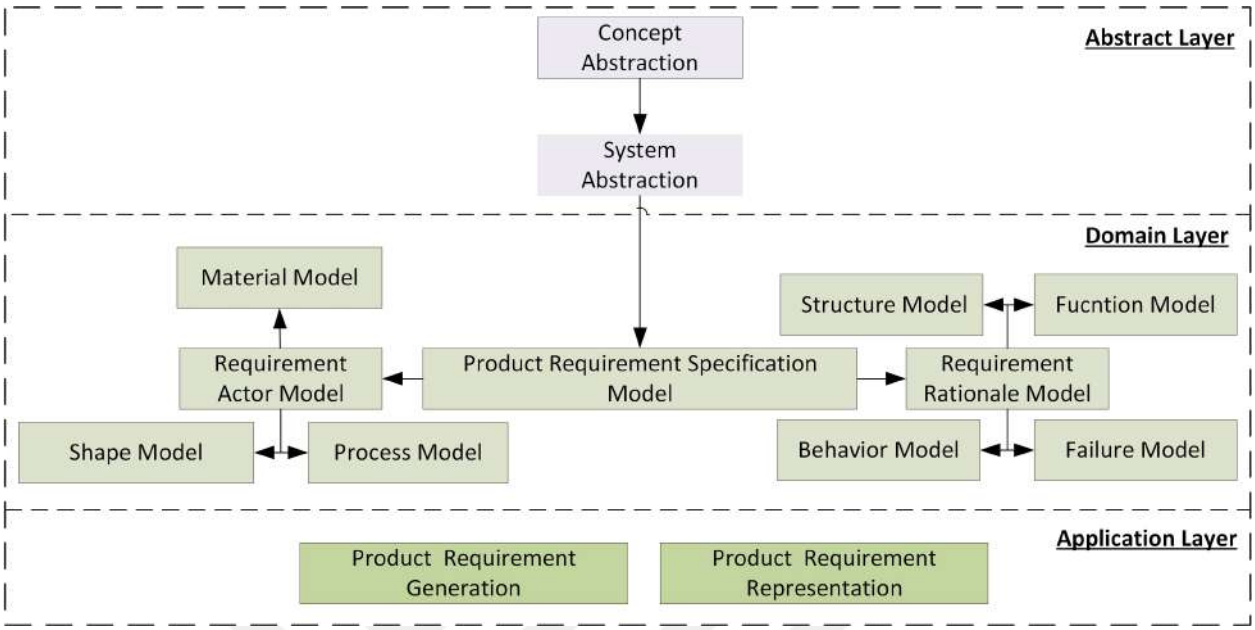


Figure 4.2: The Layered Modeling Methodology for the Development of the PDROM

we mainly discuss and implement **Domain** and **Application** layers.

**Domain Layer** comprises the core of the proposed information models for product requirement specification. It is structured into interrelated models: Material, Process, SBFF, etc. The requirement information model integrates the product domain models and requirement rationale models. The product domain models represent the product requirement related information such as shape, material, and process. The rationale models present the requirement rationale through the structure, behavior, function, and failure models.

**Application Layer** extends the information model towards a concrete application. The number of such applications could be numerous. In this study, applications for the representation and generation of the product requirements are discussed by extending the domain level information model. These ontology-based domain information models are developed in this research before discussing the integration of the models to the REKMA, requirement information retrieval, and

knowledge management processes.

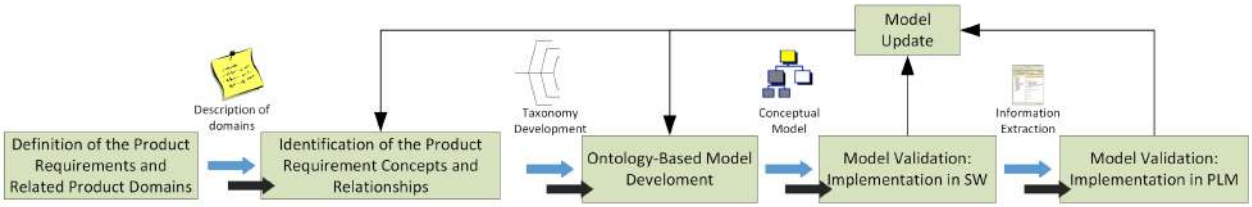


Figure 4.3: The Ontology Development Process

To develop the ontology-based information models for the requirement representation and generation, six main phases of the ontology development process for REKMA are followed as shown in figure 4.3. In the first phase, the domain of interest which is product requirement specification is defined. It includes product requirements and related product domain information. In the second phase, the domain is further analyzed. The set of concepts relevant to the domain and relationships of these concepts are identified. In the third phase, concepts and relationships are organized in a model and the domain taxonomies are developed. UML formalism is used to graphically represent the model.

Information models can be constructed with different modeling techniques, and be implemented in various kinds of languages. These languages and techniques provide constructs for classes, instances, relations, and attributes. A **class** represents a set or a category of things that have some properties or attributes. An example of a class could be ***Requirement***, ***Product***, ***Process***, ***Material***, etc. Bold italicized font with the capitalized first letter is used to represent a class in this dissertation. An attribute represents features, characteristics, or parameters of classes. It is identified by a name and it can take one or several values, which are usually restricted to a specific data type such as boolean, string, integer, etc. Underlined font with the lowercased first letter is used to represent an attribute of a class in this dissertation. For example, <u>requirementID</u> or

requirementName are attributes of **Requirement** class and they can take values of datatype string. Entities that belong to a particular class are said to be instances or members of that class. An example of an entity could be *coffeeMaker* or *UAV* that are instances of the ***Product*** class. Italicized font with the lowercased first letter is used to represent an instance of a class in this dissertation. A **Relation** describes the interrelation between classes. it can also be denoted as properties, roles, slots, or associations in other modeling paradigms. Most modeling languages support the relations only between two classes in a way that it points from a particular domain class to a designated range class. As an example, some major concepts in the proposed PDROM are: ***Requirement***, ***Stage***, ***Actor*** and ***Type***. The relationship between ***Requirement*** and ***Stage*** is ***composition***, which means that a ***Requirement*** (domain class) instance comprises of several ***Stage*** (range class) instances. Bold italicized font with the lowercased first letter is used to represent a relation in this dissertation. The UML class diagram is an ideal object-oriented tool for representing the PRDOM model since it provides classes, instances, and attributes to represent domain concepts, and relations to represent the relationships between these concepts. In a UML class diagram, the rectangular box represents classes or concepts while the diamond arrow, hollowed triangular arrow, and line identify the relationships between classes as shown in figure A.1 in the appendix A.1. The ***composition*** relationship is denoted as a filled diamond arrow, ***inheritance*** relationship is denoted as a hollowed triangular arrow, and ***association*** relationship is denoted as a straight line.

In the fourth phase, the proposed models are validated. The ontology-based information models are formally implemented in an ontology editor to instantiate the models by translating UML models into the Web Ontology Language (OWL). A set of product requirements are stored in the ontology editor and incompleteness or inconsistencies in the requirement specification are analyzed. In the fifth phase, the data structures of the ontology-based information models including

classes and relationships are implemented in the PLM by customizing a PLM tool through the models' structures. The product requirements are then stored in the customized PLM. Requirement information is extracted from the PLM system to have it in OWL to check whether the PLM system and the proposed requirement models can store and represent the product requirements and related information in the same structure. Finally, based on the models' capability for the product requirement representation in the ontology editor and the PLM system, they are released or updated.

## 4.3 Information Model for Product Requirement Specification

In order to address issues that are discussed in chapters 1 and 3, the product requirement structure is explored extensively and a Product Design Requirement Ontology Model (PDROM), which provides rich requirement semantics is proposed in this chapter. The model is for formal representation and generation of product requirements and it supports a new level of product requirement storage and analysis by addressing different facets of requirement specifications. The proposed PDROM is described in detail as follows:

PDROM consists of two main parts; (I) Part I is for requirement representation and description of a requirement and (ii) Part II is to support requirement generation and to link between product requirements and design rationale. In the Part I as shown in figure 4.4, class ***Specification***, which can be considered the root entity for requirement representation, constructs the highest level of generalization. It reflects a collection of relevant information from the ***Requirement*** class for product design.

Class ***Requirement*** is aggregated to the ***Specification*** class and it can be further divided into four categories: ***CustomerRequirement***, ***CorporateRequirement***, ***RegulatorRequirement*** and ***Tech-***

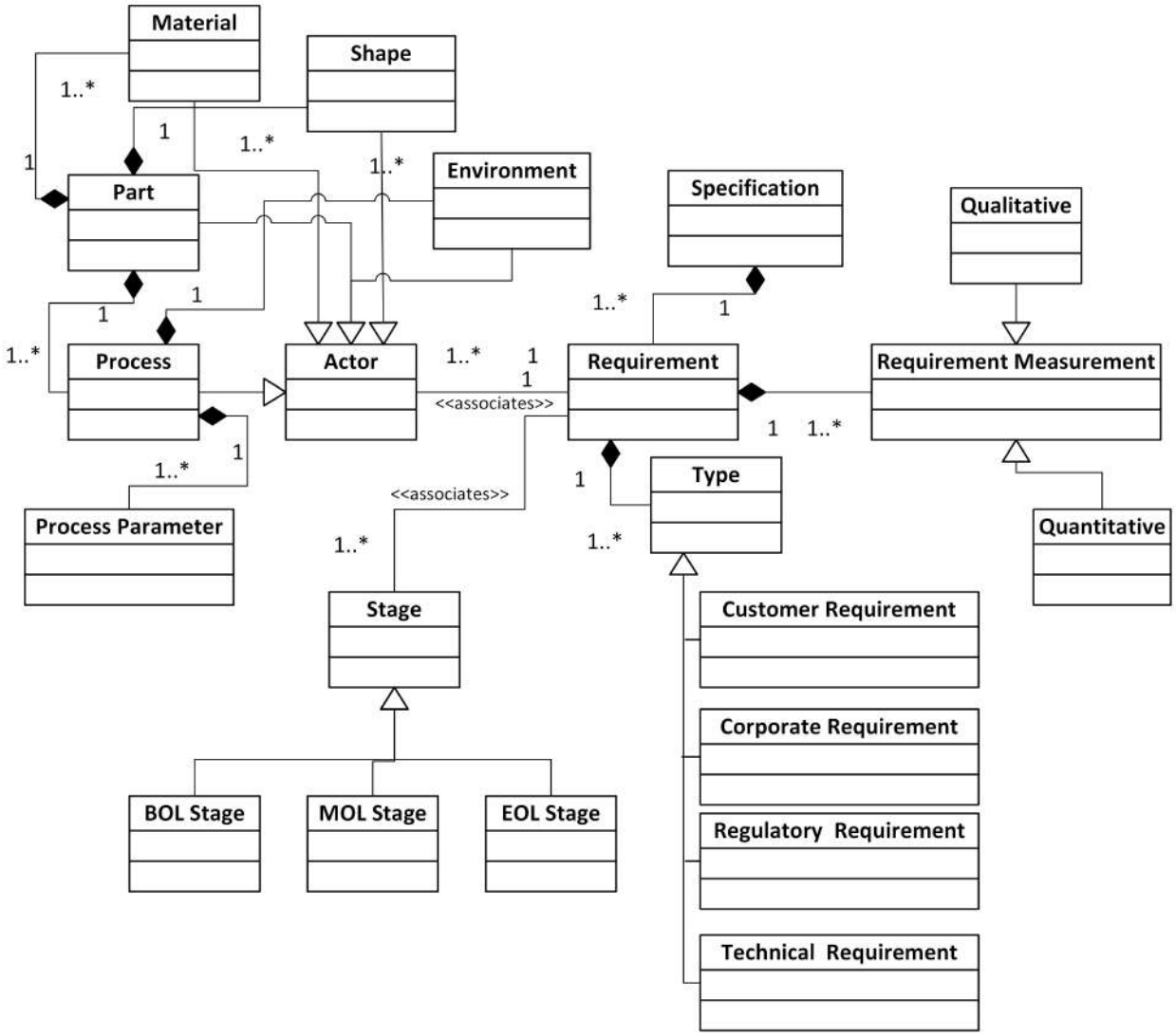


Figure 4.4: Part I Model for the Requirement Representations

***nicalRequirement***. Every Requirement instance is comprised of one or more ***RequirementMeasurement*** instance(s) and ***Stage*** instance(s). Also, every ***Requirement*** instance is associated with one or more ***Actor*** instance. ***RequirementMeasurement*** class represents the objective of the requirement sentences in either a qualitative or quantitative way. ***Stage*** class represents the product life-cycle domain through which a requirement is applied and used. It could be ***Manufacturing Stage***, ***MOL Stage*** (Middle of Life Stage) and ***EOL Stage*** (End of Life State). For example, the

dimension between two geometry entities (lid and bottle) could be specified differently (open or closed) for ***Use Stage*** under ***MOL Stage*** according to the stage of the parts.

***Actor*** class represents a diverse subject in requirements and it could be a ***Part***, ***Process***, ***Material***, etc. The product requirements provide Actor-based constraints and objectives which classify and define the ***Actor***. ***Part*** class describes a single component object which can be a product or used to construct a product. It must be formed by certain material or materials and has certain geometrical shape based on requirements. ***Material*** class and ***Shape*** class are aggregated into ***Part*** class. Specifically, ***Material*** class describes what the Part are made by and it gives the description of the internal composition of the ***Part***. The ***Shape*** class describes the shape, geometry and feature information to form a ***Part***. It usually contains the boundary representation (B-Rep) structure which is already defined in a formal data format (such as ISO STEP part 21 file) [20]. ***Process*** class is a collection of activities which are sets of sequential or parallel operations that divided into many categories: ***MaterialProductionProcess***, ***ManufacturingProcess***, ***AssemblyProcess***, ***DisassemblyProcess RecycleProcess***, ***DisposalProcess***, ***InspectinProcess***, etc. It is comprised of a set of ***ProcesParameter*** instances. ***Environment*** class describes environmental information that effects product through its lifecycle such as manufacturing process atmosphere, storing temperature, usage corrosiveness, etc. for the related ***Actor***.

The Part I demonstrates a structured way for requirement modeling and reveals that any requirement description (usually written in a natural sentence), can be broken down into the proposed PDROM structure. A simple illustrative example for the automobile brake system, R1: ”distance between surface A (brake pad surface) and surface B (rotor surface) shall be bigger than 0.3 cm during the non-braking time”, can be structured into the PDROM Part I as follows:

***Requirement***(R1)=

***Actor***(***Part***(***Shape***(surface A))) $\bigcup$

***Actor***(***Part***(***Shape***(surface B))) $\bigcup$

***Actor***(***Part***(***Shape***(distance))) $\bigcup$

***RequirementMeasurement***(***Quantitative***(>0.3 cm)) $\bigcup$

***Stage***(***MOL***(***Usage***(non-braking time))) $\bigcup$

***Type***(***Technical Requirement***(***Functional Requirement***))

In this example, a requirement sentence is divided into its phrases (requirement instances). The '*surface A*', '*surface B*', '*distance*', '>0.3 cm' and '*non-braking time*' are the instances of this requirement sentence. Then, these requirement instances that match with the requirement model classes are represented with class and sub-class names as discussed in table 3.7. The '*surface A*', '*surface B*', and '*distance*' belong to ***Shape*** class which is aggregated to the ***Part*** and ***Actor*** classes respectively. The '*non-braking time*' is classified under ***Usage*** stage which is aggregated to the ***MOL*** and ***Stage*** classes. The '*bigger than 0.3 cm*' is the instance of the ***RequirementMeasurement*** class and it is classified under the ***Quantitative*** class. This demonstration of a product requirement sentence based on the Part I is about the representation of requirements and classification of requirement instances.

There is still a need to extend this model (Part I) to show the rationale behind the requirements. As a part of the PDROM, Part I model is extended with the Part II model as shown in figure 4.5. Part II is proposed to represent the rationale and relationships between the requirements and design

objects. This model is called Structure Behavior Function Failure (SBFF) model and it consists of four system models: **Structure Model**, **Behavior Model**, **Function Model**, and **Failure Model**. It is developed based on Structure-Behavior-Function (SBF) ontology model [112] with the extension of the failure model.

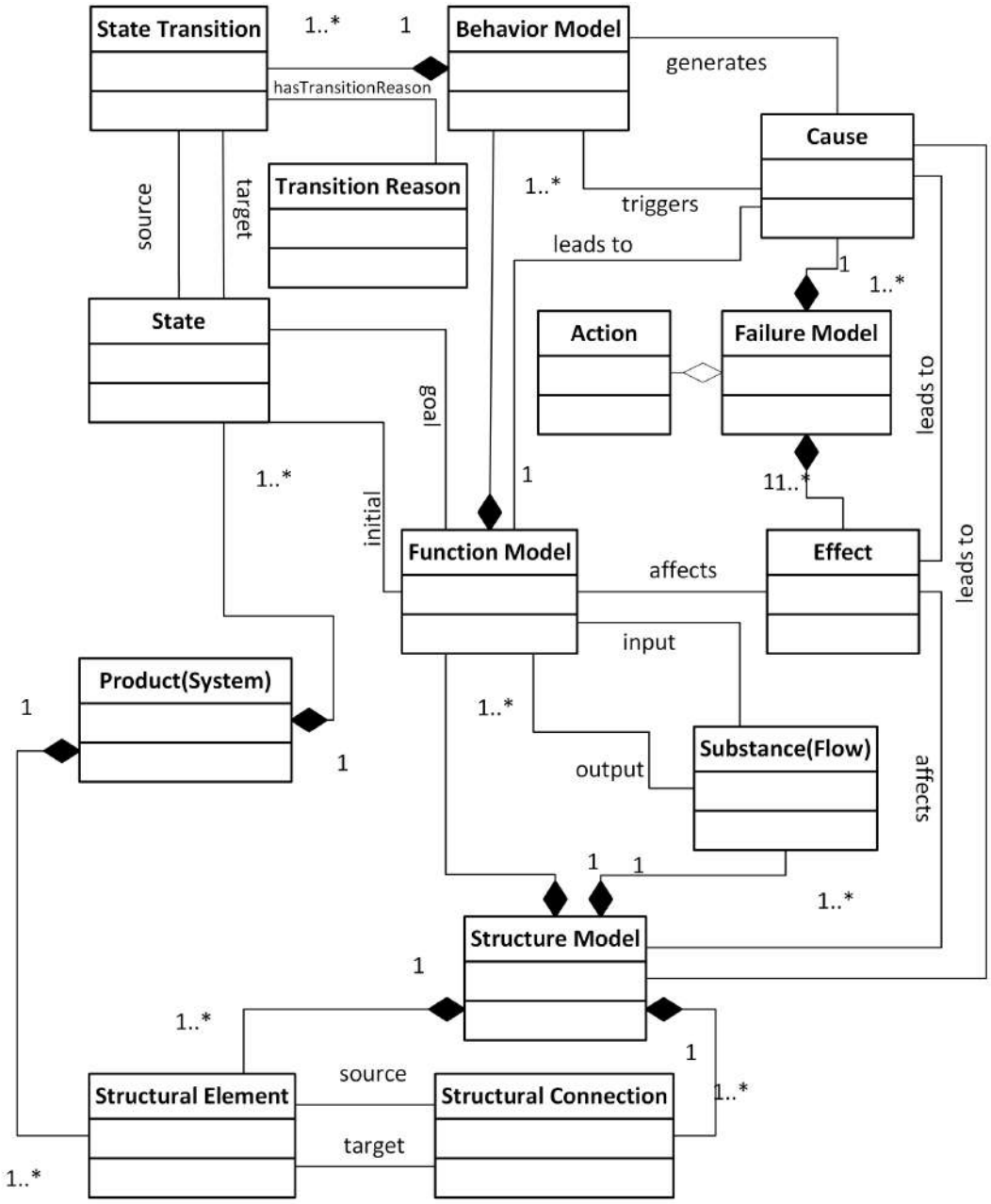


Figure 4.5: Part II Model for the Requirement Rationale

In this model, ***Structure Model*** represents the collection of things for a mechanism that is created to achieve particular purposes. It consists of ***StructureElement*** that defines physical things for the mechanism and ***Flows*** which are input and output for that mechanism. ***Flow*** is represented by *Property* and a property is represented by ***Value*** and *Unit*. ***StructuralConnection*** class describes how structure elements connect each other to complete the mechanism. ***StructureElement*** might be represented as an ***Object*** or its ***Component*** for a designed product or might represent physical things of any particular process for product life-cycle operations such as manufacturing, assembly, maintenance, disassembly processes, etc.

***Function Model*** defines a particular objective that a mechanism needs to achieve. Every function is represented by ***Initial State*** and ***Goal State***. Also, it has ***input Flow(s)*** and ***output Flow(s)***.

***Behavior Model*** describes how a mechanism works and achieves a certain function with structure attributes. A behavior might be ***Mechanical***, ***Electrical***, ***Thermal*** or ***Aerodynamic*** type and it can be described as a ***Quantitative*** or ***Qualitative***. Every behavior is represented by ***State***, ***State Transition*** and ***Transition Reason***. The state transition between source state and target state consists of four types for any behavior: ***By_Function***, ***By_Principle***, ***By_External_Stimulus***, and ***By_State***. ***External_Stimulus*** is defined as an input from outside of mechanism that create a change on it. These terms are explained in detail by Goel et al. in their SBF model.

***Failure Model*** represents possible situations that a mechanism does not meet a specific objective particularly or completely. It is comprised of ***Failure Mode*** that is an attribute and defined with a broad taxonomy [113]. Each type of failure model is represented by ***Cause***, ***Effect*** and ***Action***.

Let's look at the example discussed above for the automobile brake system, R1: "The distance between the pad surface and rotor surface shall be bigger than 0.3 cm during the non-braking time"

and its rationale is 'keep a distance between surfaces to avoid unnecessary friction and wearing' can be structured into the PDROM as follows:

***Requirement***(R1)=
***Actor***(***Part***(pad)) ⋃
***Actor***(***Part***(rotor)) ⋃
***Actor***(***Shape***(pad surface)) ⋃
***Actor***(***Shape***(rotor surface)) ⋃
***RequirementMeasurement***(***Quantitative***(>0.3 cm)) ⋃
***Stage***(***MOL***(***Usage***(non-braking time)))⋃
***Type***(***Technical Requirement***(***Functional Requirement***)) ⋃
***Structure_Model***(***Structure_Element***(***Component***(pad, rotor))) ⋃
***Function Model***(***State***(***Goal***≡***Initial***(keep distance))) ⋃
***Flow***(***Property***(distance between pad surface and rotor surface)) ⋃
***Property(Value***(>0.3)) ⋃
***Property(Value***(***Unit***(cm))) ⋃
***Behavior Model*** (***TransitionReason***(***By_Function***(function of a product (caliper))) ⋃
***FailureMode***(***Cause***(friction)) ⋃
***FailureMode***(***Effect***(thermal fatigue)) ⋃
***FailureMode***(***Effect***(fatigue wear)) ⋃

In this example, the requirement sentence is represented in the PDROM Part II model to define the requirement rationale as an addition of the requirement sentence representation based on the PDROM Part I model. Additionally, the requirement rationale is divided into instances of the structure, function, behavior, and failure models. The '*keep distance*' is the initial and goal function for this requirement. This function has a flow instance, 'distance between the pad surface and rotor surface' which is classified as a property. This property has a value instance, '>0.3' and unit instance, 'cm'. This function model is achieved by a behavioral modal which is the function of a product (caliper). If this mechanism does not meet the objective, it might yield failure which is explained by the failure model. It represents 'friction' as an instance of the cause, 'thermal fatigue', and 'fatigue wear' as instances of effect classes. The PDROM Part I and Part II models

are integrated as shown in figure 4.6. Whenever a requirement is generated, it is represented with the PDROM model and the requirement instances are classified for the following classes in the given order.

1- Actor of the requirement

2- Stage of the requirement

3- Type of the requirement

4- Measurement of the requirement

5- Structure, Function, Behavior, and Failure models of the requirement and requirement rationale

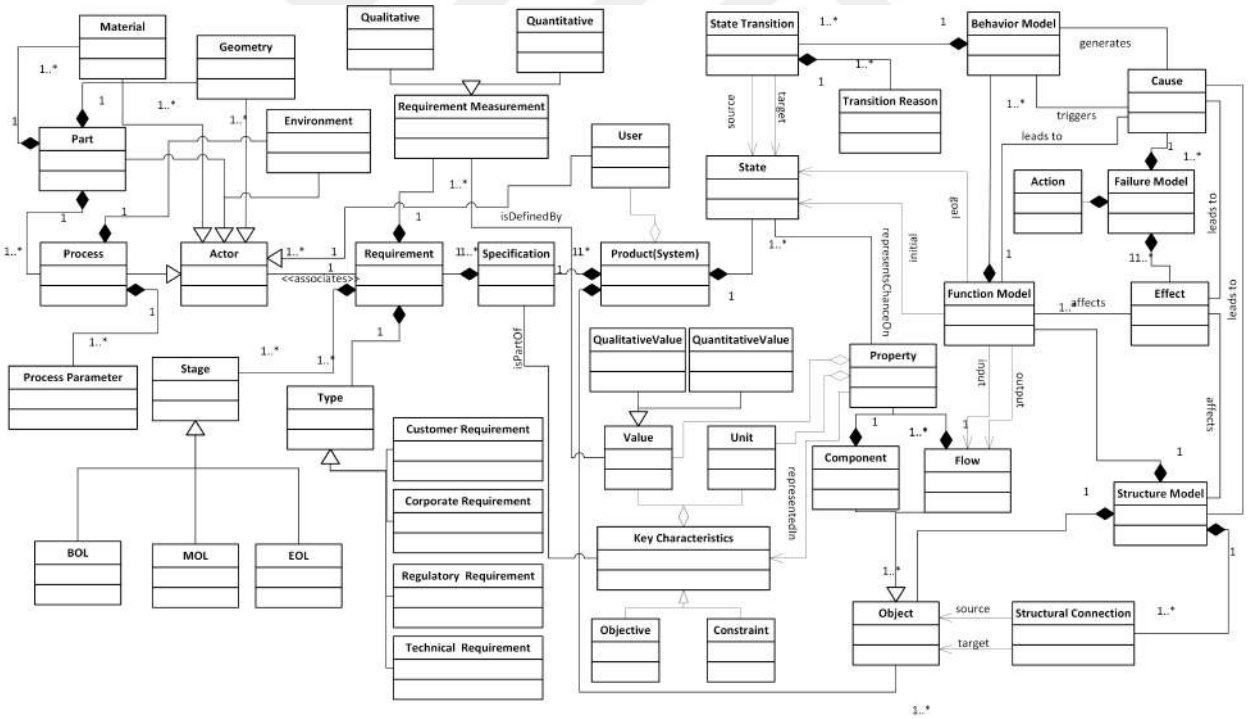


Figure 4.6: PDROM Part I and Part II Integrated

In this integrated model (PDROM), two more classes are added to the Part I and Part II to create better connection between them. The firs one is ***User*** class to represent end user who

uses a product or who is involved in the life-cycle stages of the product design, development, production, maintenance, etc. In this model, ***User*** class is considered as a requirement ***Actor***. Some of the requirements in a requirement specification document are specifically generated for the users that makes users the subject of the requirement sentence. The second class that is added to the integrated model is the ***Key Characteristics*** (KCs). It is also called as properties or performance characteristics. It describes the core of a product requirement and it can be the information about product performance, reliability, cost, weight, feature of a system, material, process, part, etc. [114, 115]. KCs class facilitates the efficient interaction between PDROM Part I and PDROM Part II through the identification and generation of critical information as a summary of a requirement specification. ***Key Characteristics*** can be further divided into two categories; ***Constraint*** and ***Objective***, and every ***Key Characteristics*** instance is comprised of one or more ***Value*** instance(s) and ***Unit*** instance(s).

## 4.4 Product Requirement Actor Models: Material, Process and Shape

There is a dire need for the development of an integrated product requirement information model to organize and manage material, form, and manufacturing related information, required for any product requirement realization by extending requirement ***Actor*** classes. Detailed representation for them is necessary to capture all product requirement related information. The role of materials, shapes, and the associated manufacturing processes needed to fabricate the materials into a specified form for product requirement realization is very important in the context of the requirement specification.

The information models of the product requirement actors capture the product domain information and knowledge for the product requirement representation and generation. For example,

***Material*** class can be extended and developed a Material Information Model (MIM) that describes the possible requirement-related information for product material. In this dissertation, some scope or themes are identified for ***Actor*** classes, and constructed to concepts that needed to represent requirements for product domain classes. Information frameworks for ***Material***, ***Shape*** and ***ManufacturingProcess*** models are proposed to support both PDROM Part I and Part II. These models consist of taxonomies and hierarchical organization for those taxonomies. The following sections describe these domain layer actor-models in detail.

### 4.4.1 Material Information Model

The material information model is to facilitate product requirements for the product material and material selection process during the beginning of the product design and also during any decision making process related to product's maintenance and replacement, and product's end-of-life activities. Information regarding material characteristics of the product must be integrated with the product's form, function, behavioral, failure models. The material package shown in figure 4.4 and figure 4.6 is further detailed and it is illustrated in figure 4.7 below. The Material Information Model (MIM) that describes the possible requirement-related information for product material.

The ***Material*** class is the core in the MIM, and it is comprised of a variety of information like ***MaterialProperties***, ***Composition***, ***Cost***, ***Application*** etc. The ***Material*** class defines a raw material, which is used to form a part or product. Certain manufacturing processes must be employed to produce the raw materials and transform the raw materials into the final part that we will discuss this under Process Model. ***Composition*** class provides information about the components of the material, their proportion, and the intrinsic characteristics (i.e. the physicochemical nature) of a material. ***HazardsMaterial*** class indicates the inherent property of a material that can cause

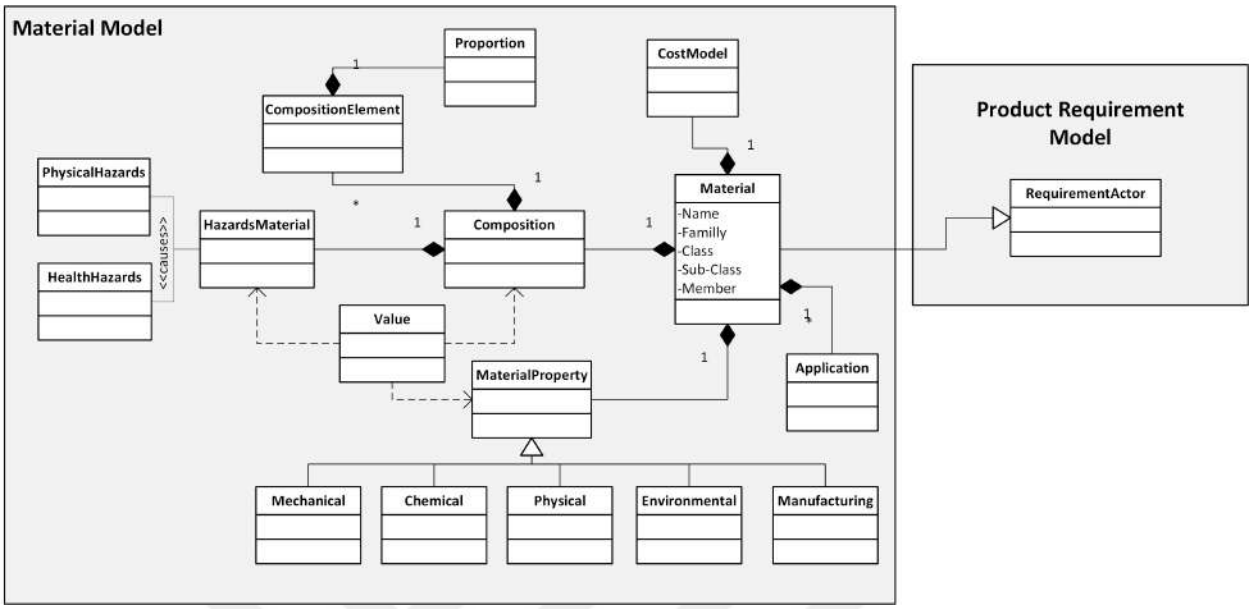


Figure 4.7: Material Information Model

adverse physical and health effects. Standards like STEP part 45 [116], IPC-1751/1752 [117, 118], etc. discuss some representation issues related to the structure-based material property and hazardous material composition identification and declaration. The material information could contain material family, material name, material properties, and so on. A material property (e.g., mass, density, etc.) is represented by the class ***MaterialProperty***. This class can be used to assign a material property either to the whole product or to the part of the product. It can be further divided into five categories: ***MechanicalProperty***, ***PhysicalProperty***, ***ChemicalProperty***, ***ManufacturingProperty*** and ***EnvironmentalProperty***. These material properties and material property taxonomy are briefly described in the appendix A.2. Any material property indicated in a requirement sentence can be represented using this taxonomy and model hierarchy in a structured way. For example, instead of defining the '*thermal conductivity*' property in a requirement sentence without any definition, it can be represented as ***Actor***(***Material***(***Material Property***(***Physical Property***(***Thermal Property***(*thermal conductivity*))))) using this model.

### 4.4.2 Shape Information Model

In this dissertation, a Shape Information Model (SIM) is proposed to identify the high-level shape information of a product which is presented in the product requirement sentences. This ontology model is for the semantic level description of geometric terms and relationships which they are usually mentioned in product requirements. The geometric product specifications are typically generated with CAD systems and represented with formal data format STEP AP or vocabularies from popular CAD software. It usually contains the boundary representation (B-Rep) structure which is already defined in a formal data format (such as ISO STEP part 21 file [119]).

Figure 4.8 illustrates the proposed Shape Information Model using the UML class diagram. The SIM is developed to represent the following major information or knowledge in a product requirement. The description of each class in the SIM is described in detail as follows:

The ***Shape*** class describes the form of an object that categorized with various form ***Type*** and it can be represented as a list of ***Features***, various ***Geometry*** and ***Topology*** entities in the objects' feature tree or a boundary representation (BRep) information. While shapes can be classified into open and closed shapes, only closes geometric shapes; two-dimensional and three-dimensional shapes are placed into shape taxonomy which is used for product requirement representation. These shapes have defining attributes; ***Vertex***, ***Edge***, ***Face***, ***Solid*** that are ***Topology*** entities and ***ClosedSurface(s)***, ***Surface***, ***Curve*** and ***Point*** that are ***Geometry*** entities. ***Shapes*** also represented by ***Feature*** that is a set of face(s) with distinct topological and geometrical characteristics such as ***Hole***, ***Cutting***, ***Bending***, ***Rib***, ***Thread***, etc. There are many studies to classify these features under different categories such as volumetric feature, deformation feature and free-form surface feature [120].

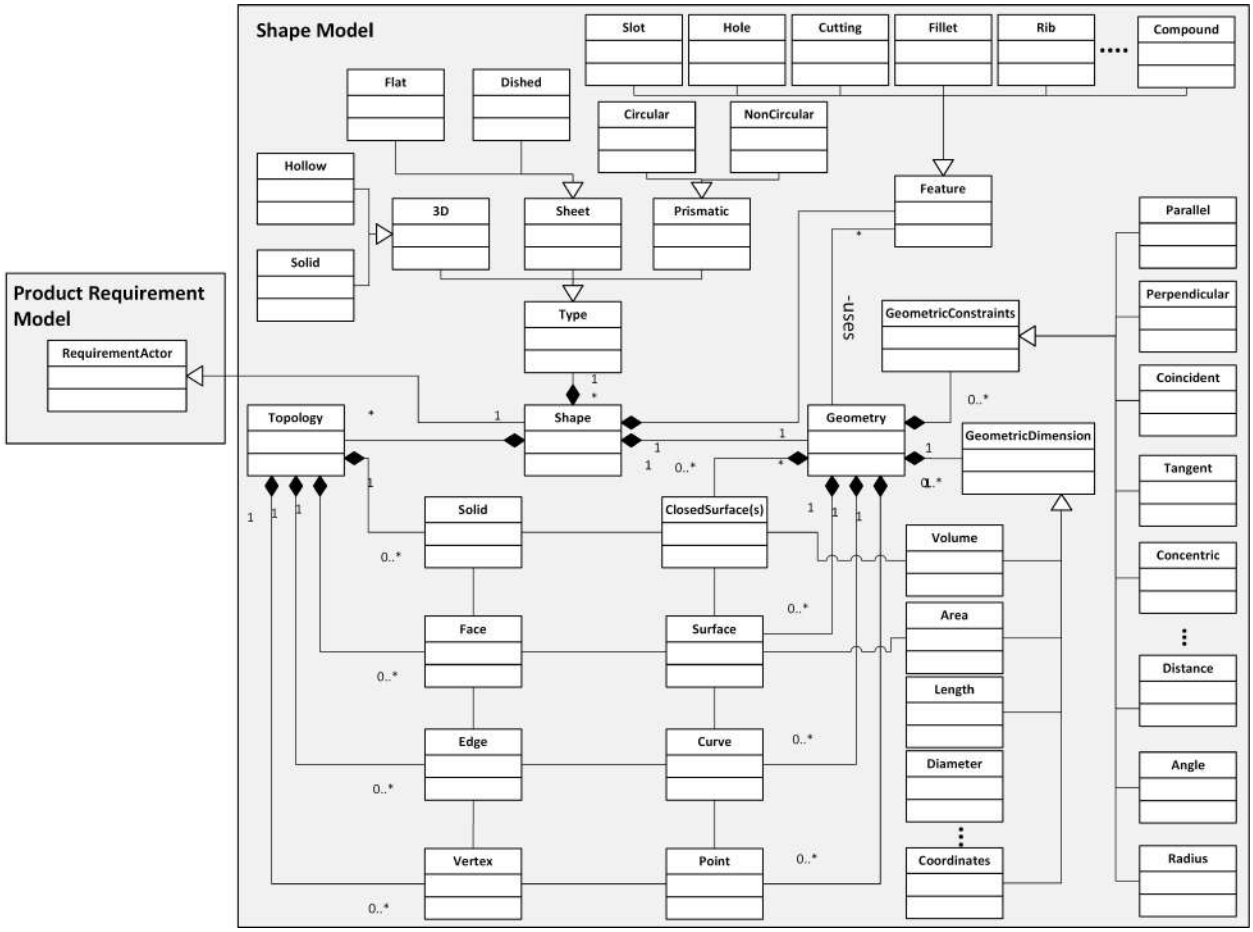


Figure 4.8: Shape Information Model

Real-life objects that named solid shapes are classified into three basic forms; ***3D***, ***Sheet*** and ***Prismatic***, according to shape and process relationship [6]. ***3D*** shape is further divided into two categories; ***Solid*** and ***Hollow***. ***Sheet*** class might be ***Flat*** or ***Dished***, and ***Prismatic*** class might be ***Circular*** and ***Noncircular***.

Geometry mainly studies the object's position and represented by ***GeometricDimension***. Its ***Lengths***, ***Angle***, ***Sizes***, ***Areas***, ***Volumes*** etc... that literally describes the objects in space. On the other hand, ***Topology*** is the study of position in an area of mathematics that deals understand shape and space without an explicit measure of distance, size, volume, angles, coordinates etc., which is

unaffected by deformation.

***GeometricConstraints*** are rules that placed limitations on objects to define their **Distance**, ***Radius***, ***location***, ***orientation*** etc. Constraints such as ***Parallel***, ***Perpendicular***, ***Coincident***, ***Tangent***, ***Concentric***, etc. develops relationships between components placed within assembly models and controls geometry during design stage.

Any shape information indicated in a requirement sentence can be represented using shape taxonomy and model hierarchy using this model. For example, the geometric constraints, '*parallel*' in the requirement sentence, '*Pad surface and rotor surface shall be parallel during the usage stage*' is represented as ***Actor***(***Shape***(***Geometry***(***Geometric Constraints***(***Parallel***(*pad surface, rotor surface*))))) in a structured way.

A detailed semantic description of CAD models, which is based on BREP, can be found in the study of Perzylo et al. [121] and a complete specification of BREP's geometric and topological representation can be found in ISO 10303-42 [122].

It should be noted that the material information model (MIM) and shape information model (SIM) are closely tied up with the product information model and the manufacturing processing information models. Since the product information models are extensively studied by the authors and their co-researchers at NIST in their past works [18, 123, 124], this study mainly emphasizes the development of the manufacturing processing model.

### 4.4.3 Process Information Model

In this dissertation, a Process Information Model (PIM) to identify and represent the high-level manufacturing process information which is presented in the process requirement sentences. This generic model serves as an information core and can be used directly for product requirement

evaluation with the expansion of a specific process and material. In the previous study [125], PIM is expanded to the powder metallurgy process model and discussed with the MIM model.

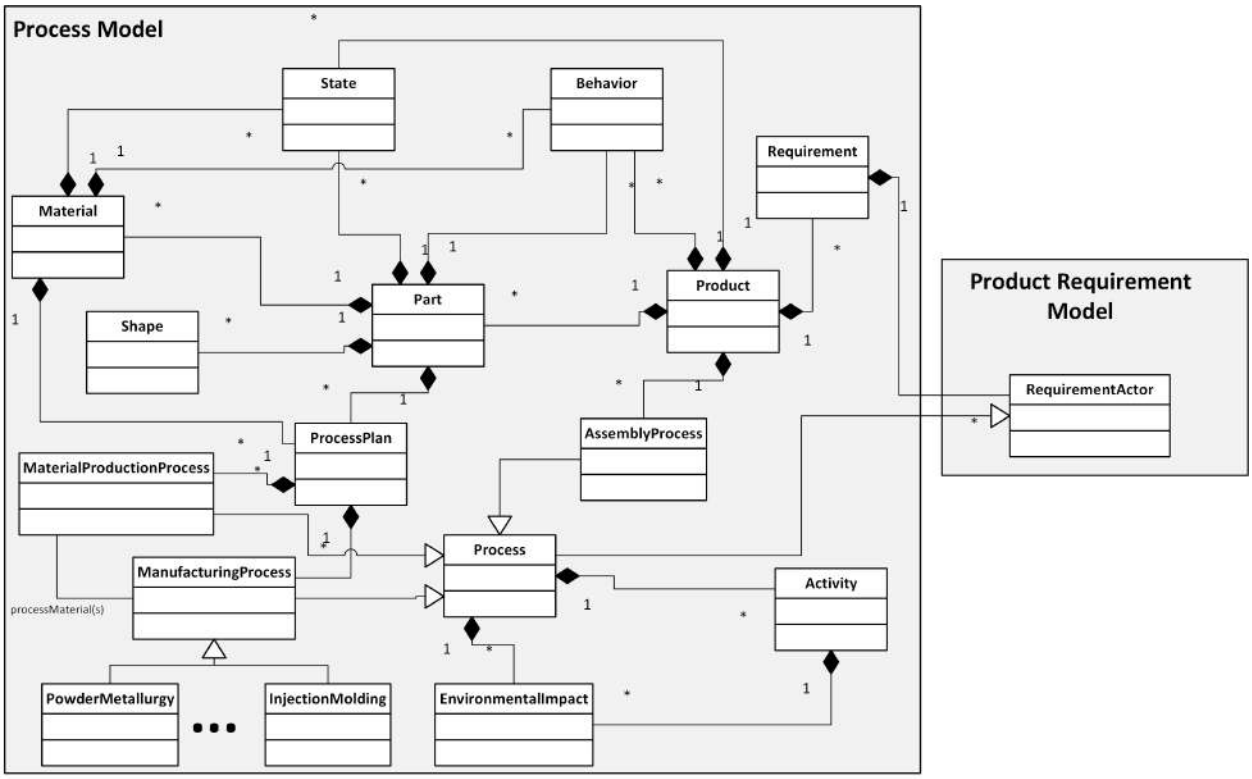


Figure 4.9: Process-oriented Information Model

Figure 4.9 illustrates the proposed Process Information Model using the UML class diagram. The PIM is developed to represent the following major information and knowledge for the manufacturing process requirement specification in a product design process. The description of each class in the PIM is described in detail as follows:

The ***Product*** class describes an object which is synthesized by a set of parts or subassemblies (each subassembly itself is also a product object). These parts are put together following appropriate spatial relationship and contact constraints. A product comprises its instances of behaviors to describe a certain product's motion pattern and functions according to the inputs to the product.

The ***Part*** class describes a single component object which can be used to construct a product. A part is a minimal functional unit of a product; thereby a part must be formed with certain material and has certain geometrical shape. In the Core Product Model (CPM) scheme, an ***Artifact*** composes its material and shape, where the ***Artifact*** conforms to the ***Part*** class in the PIM.

The ***Material*** class describes the raw material object which is used to form a part. Certain manufacturing processes must be employed to produce raw materials and transform them into the final part. The material information could contain material family, material name, material properties, and so on. The Behavior of a ***Material*** instance involves the property changes of the material under different internal or external environments (stimulus).

The ***Behavior*** class defines the behavior pattern of a certain product, part, or material. Behavior describes the reaction of an entity if certain external stimuli are applied to this entity or the interaction between the sub-entities which comprise the entity. For example, the behavior of a product can be understood as the execution of different functions according to various user input. The behavior of a part can be the shape deformation when a force is applied according to its geometry information. The behavior of a material can be the changes of material structure and properties according to the external environmental conditions such as temperature and pressure.

The ***Shape*** class describes the feature information to form a part. It usually contains the boundary representation (B-Rep) [126] structure which is already defined in a formal data format (such as ISO STEP part 21 file [119]).

The ***State*** class describes the status of a ***Product***, ***Part*** or ***Material*** at a certain time point. For example, a mechanical or a chemical property of a particular ***Material*** might have different values under different conditions or by using different measuring methods. The ***State*** class enables the PIM to capture the characteristics of any ***Product***, ***Part*** or ***Material*** at any important time point.

A ***ProcessPlan*** defines the sequence of manufacturing operations to produce a ***Part*** or a ***Material***. The types of processes, types of equipment, and operation parameters are specified in a ***ProcessPlan***.

A ***Process*** describes any of processes that would be carried out to produce the final product. As the PIM shown, a ***Process*** can be a ***ManufacturingProcess***, ***MaterialProductionProcess*** or an ***AssemblyProcess***. A ***ManufacturingProcess*** is a process that transforms a raw material into a finished or a semi-finished part. It can be a machining process, powder metallurgy process, casting process, forging process, heat treatment process, etc. All the sequentially organized ***ManufacturingProcess*** required to be carried out to produce a ***Part*** forms a ***ProcessPlan***. A ***MaterialProductionProcess*** describes the activities and their sub-activities which are required to produce ***Material***. An ***AssemblyProcess*** is a process that assembles all the parts to a ***Product*** (which can be a final product or a subassembly as discussed before). A ***Process*** can typically be decomposed into a set of sequential or parallel operations, each of which is an ***Activity***.

The ***EnvironmentalImpact*** class contains the attributes that indicate the manufacturing footprint of a certain activity, such as energy consumption, waste, and emissions. Also, the ***EnvironmentalImpact*** class describes the environmental impacts brought by an Activity. This ***Environment*** class, which is also discussed in the requirement model, represents environmental information that affects process and process activities such as process atmosphere, temperature, etc.

An ***Activity*** is a minimal operational unit of a ***Process***. For example, an ***Activity*** of a typical machining process can be setting up the machine, fastening the workpiece, positioning the cutting tool, injecting the cutting fluid, and cutting, etc. The operation parameters (e.g. feed rate, spindle speed, and depth of cut for machining) associated with an ***Activity*** are captured in this class. To

carry out an ***Activity***, some resources (e.g. raw material and/or energy) are consumed and some environmental impacts are introduced.

In order to support the development of the product requirement specification for a certain manufacturing process, the PIM model is further formalized and expanded to Powder Metallurgy (PM) and Injection Molding (IM). These expanded models are discussed in [125, 39]. The expanded PIM keeps most entities of the generic PIM but expands the ***Material*** class and the ***ManufacturingProcess*** class to serve for a specific manufacturing process and material. The differences among the expanded models are mainly on the composition of the material and process activities. There are a lot of important factors that influence each activity of the expended PIMs. Variables that affect product quality and process environmental impacts must be identified to support the development of the product requirement specification. To write a consistent and complete set of process requirements for activities of a specific process, a deep understanding of not only part geometry, size, material type, and material properties, but also the effects of these factors on the process are needed. The factors must be well known and controlled during the manufacturing process to create a part with desired properties.These factors provide critical information that must be considered during a product's conceptual design stage and listed under the product requirement specification. Some factors for the powder metallurgy process and the expanded PIM for the powder metallurgy process are represented in the appendix A.3 and discussed in [125] through each step of the powder metallurgy process from powder to finished part.

Factors that affect manufacturing processes should be defined and considered as product requirements. Failing to precisely control these factors will yield undesirable properties. These factors are implemented as relationships, constraints, and rules when PDROM is implemented in Protege based on Web Ontology Language (OWL) in the next chapter.

## 4.5 Summary of Model Development

Since there are complex relationships among material, shape, and process, a structured knowledge-base approach is discussed in this study for integrating product requirement related information. For a complex assembly system like an automobile, requirements (like 'safety', 'environmental friendliness', etc.) are not immediately obvious unless the system is broken down into several layers, and specific sub-requirements are properly understood. On top of that, identification of the product requirements is highly complicated and the knowledge base system must evolve accordingly to cope with the changes and practical needs. In order to achieve this goal, we discuss the creation of appropriate information models for requirements, material, shape, and process. The following chapter will be about the implementation of these models.

# CHAPTER 5

# PDROM IMPLEMENTATION BASED ON WEB ONTOLOGY LANGUAGE

In this chapter, implementation of the proposed requirement information model (PDROM) in Protégé and instantiation of the PDROM by translating UML models into the Web Ontology Language (OWL) are presented. This chapter discusses how the product requirements are stored and represented and how the incompleteness or inconsistencies in the requirement specification are detected with the PDROM. The Semantic Web Rule Language (SWRL) rules for rule-based reasoning and inference are represented. Lastly, the instantiation of the PDROM Model with automobile brake requirements is discussed.

## 5.1 Implementation of PDROM in Protege

The proposed PDROM is implemented in this chapter. Data structures and relationships for product requirement domains are built on top of the PDROM. Instances of the product requirements are stored according to PDROM data structures and relationships. PDROM represents a conceptual information model which is a high-level graphical description of the important information in the domain of product requirement specification, and it is implemented into the Web Ontology Language (OWL) for machine reasoning, interpretation, and web search queries.

The implementation of the PDROM is discussed in four steps as shown in figure 5.1. The development of the PDROM is discussed in the previous chapter that represents the information structures of the product requirements using the UML schema. In this chapter, first, instantiation

of the PDROM and Web Ontology Language (OWL) implementation are discussed to specify product requirement information with the proposed information architecture that can be published and accessed through the web. The PDROM classes, relationships, and rules that are contracted through expert knowledge about function, flow, material, shape, process, etc. are translated into the OWL schema and SWRL rules to take advantage of inference and reasoning mechanisms. Then, reasoning mechanisms such as Pellet, Hermit, etc. and SWRL rules are used to check the consistency of the requirement ontology and requirement information stored in the ontology.

The completeness check of a requirement is also performed by defining minimum information for a particular product requirement such as requirement actor, stage, type, measurement, etc. The completeness of the product requirement specification in terms of the requirement information is supported by inferring information for a certain class, property, or instance. Lastly, semantic queries are developed and executed using SQWRL (Semantic Query-Enhanced Web Rule Language) [127] to search and extract the requirement information in the classified ontology.

### 5.1.1 PDROM Model Realized in OWL

The PDROM is proposed to store and represent product requirements in a structured way to help users generate a more complete and consistent set of requirements, and show the requirement rationale. However, it is still deficient at the semantic level: the meaning of the concepts is not rich enough, which results in restricted capability for further requirement management activities such as detecting incompleteness or inconsistency in a requirement specification. The semantic processing of the requirements is indispensable to produce high-quality requirement specifications. The PDROM provides this utility by adding description rules and implementing it in OWL as discussed in detail in this section.

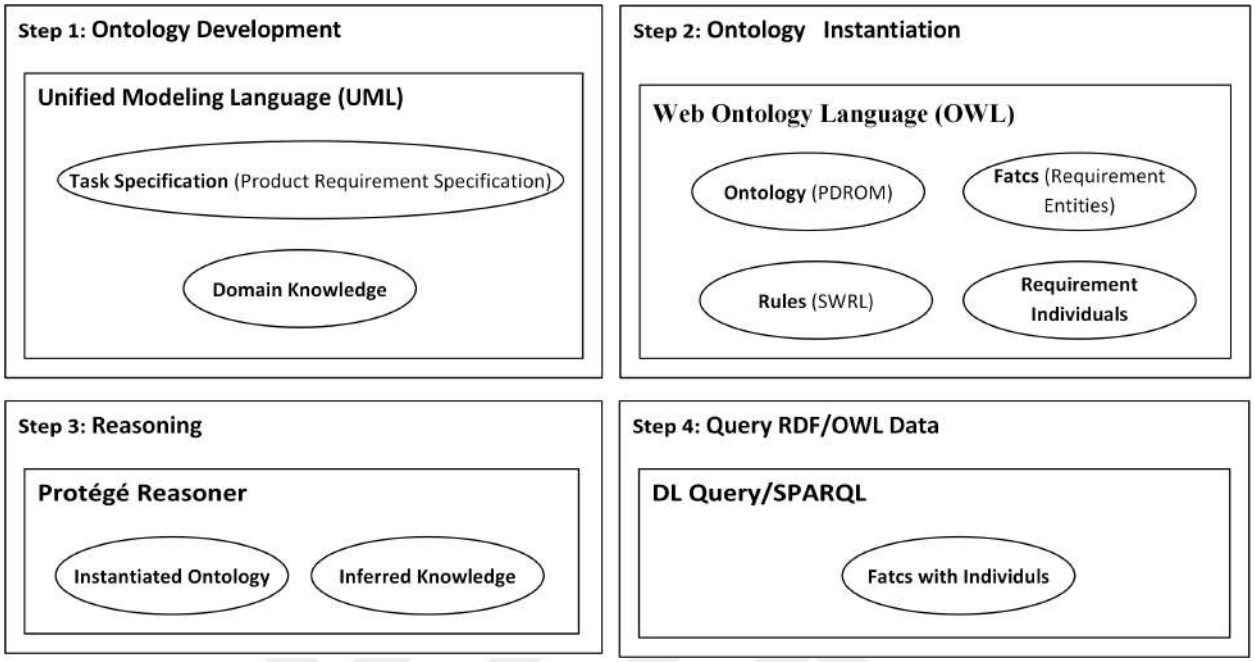


Figure 5.1: Steps for Formal implementation of PDROM

The OWL 2 Web Ontology Language is recommended by the World Wide Web Consortium (W3C) [128] for the Semantic Web to represent the information and knowledge of a domain concept. It can be used to computationally implement the PRDOM. The advantages of OWL allow us to publish and access product requirement specifications through the web and develop semantic queries for necessary information retrieval. The PDROM implementation also supports to store product requirements data in the digital memory for future analytical and intelligent applications along the digital thread. The following OWL solutions represent why OWL is used for PDROM implementation:

• The working environment for product requirement specification that considers the whole product life-cycle is distributed under the paradigm of the Internet of Things (IoT) (e.g. the sustainability evaluation of product production, usage, and disposal stages is directly linked to the product specification and this service might be online and accessible from the Sustainability

evaluation systems). The semantic web (OWL) would be preferred to implement the PDROM and results in improved efficiency, accuracy, and economic benefit.

• Every requirement specification represented by PRDOM in OWL is annotated with a Unified Resources Identifier (URI). Users for other applications can access the requirement specification content and requirement information through the web, and they can add or manipulate requirement information for certain life-cycle phases.

• A complex and huge set of product requirements in PRS can be validated and searched automatically by using existing lightweight query and reasoning plugins. Protégé 5.5.0 [129], as an OWL editor that fully supports the latest OWL 2 Web Ontology Language and RDF specifications, is used in this thesis to develop formal OWL implementation of PDROM. An OWL 2 ontology consists of individuals, properties, classes, and data values. Below, we give a brief overview of the components of OWL Ontology with examples and represent PDROM OWL implementation in Protégé.

### 5.1.2 Modeling of OWL Components

A large amount of the elements in the PRDOM are meant to be modeled using OWL components: classes, relationships, and individuals. Protégé has the same frame correspond to these components. A class represents a set or a category of individual things that are the objects in a domain, have some properties or attributes in common, and are differentiated from others by kind, type, or quality. An example of a class could be **Requirement**, which would contain all the requirement individuals in our domain of interest. Classes can be represented and organized as a superclass-subclass hierarchy based on domain taxonomy. All individuals that belong to a subclass also belong to a superclass. An example of a superclass-subclass could be

**Requirement-FunctionalRequirement**. **FunctionalRequirement** is a subclass of **Requirement**, and inversely, **Requirement** is a superclass of **FunctionalRequirement**. This means that all functional requirements are requirements. In OWL DL, an individual of a class can never belong to a disjoint class. For example, **FunctionalRequirement** and **NonFunctionalRequirement** are disjoint classes, and an individual requirement belonging to one of them can not belong to the other. These superclass-subclass relationships and disjoint classes can be automatically checked by reasoning which is one of the key features of OWL-DL.

Properties describe the interrelation between classes or class and class attributes, and it can also be denoted as relations, roles, slots, or associations in other modeling paradigms like description logics and UML. If it specifies how the class individuals relate to other individuals, it is called Object Properties. Same as most modeling languages, OWL supports the representation of relations only among two individuals in a way that it points from a particular domain class to a designated range class.

As an example, our major classes in the proposed PDROM are: **Requirement**, **Stage**, **Actor**, **Type**. The object property between **Requirement** and **Stage** is ***hasStage***, which means: a **Requirement** (its domain) individual comprises of several **Stage** (its range) individuals. Another propety type in OWL and framed in Protégé is Data Properties. It represents features, characteristics, or parameters of class individuals and data property that are identified by name and can take one or several values, which are usually restricted to a specific datatype, such as boolean, string, integer, etc. As an example, requirementID, requirementName, or requirementDescription are attributes of class **Requirement**, and it can take values of the datatype String.

In OWL, both object properties and data properties can be represented and organized as a superproperty-subproperty hierarchy based on domain taxonomy. All individuals that are related

with subproperty are also related with superproperty. An example of a superproperty-subproperty could be ***hasMaterialEntity-hasMaterialPropertyEntity***. ***hasMaterialPropertyEntity*** relates **Requirement** and **MaterialProperty** individuals and is a subproperty of ***hasMaterialEntity***. Inversely, ***hasMaterialEntity*** relates **Requirement** and **Material** individuals, and it is a superproperty of ***hasMaterialPropertyEntity***. If the ***hasMaterialPropetyEntity*** object property links two individuals of **Requirement** and **MaterialProperty**, this indicates that these two individuals are related by the ***hasMaterialEntity*** object property. Same kind of example for superproperty-subproperty of datatype properties can also be created.

Information that belong to a particular class is said to be individual or members of that class. An example of an individual could be *thermalConductivity* or *thermalDiffusivity* that are individuals of the **ThermalProperties** class. In Protégé, individuals are represented with Unique Name Assumption (UNA), but they can be specified using 'Same Individual As' and 'Different Individuals' descriptions.

The requirement classes, properties, object properties, data properties, and individuals are implemented in Protégé as shown in figure 5.2, and several semantic rules and queries are defined in the next section. Figure 5.2 shows the PDROM classes and class hierarchy on the left hand-side of the figure, the data properties of the classes and object properties between classes on the right hand-side of the figure, and the requirements generated for a product in the middle of the figure as individuals.

In OWL ontology and Protégé, both object and data properties are used with restrictions to define relationships for a member of a specified class. OWL restrictions group into three main categories: quantifier, cardinality, and hasValue restrictions. These restrictions are used in Protégé as Some (existential) and Only (universal) for quantifier restrictions and Min (min cardinality),

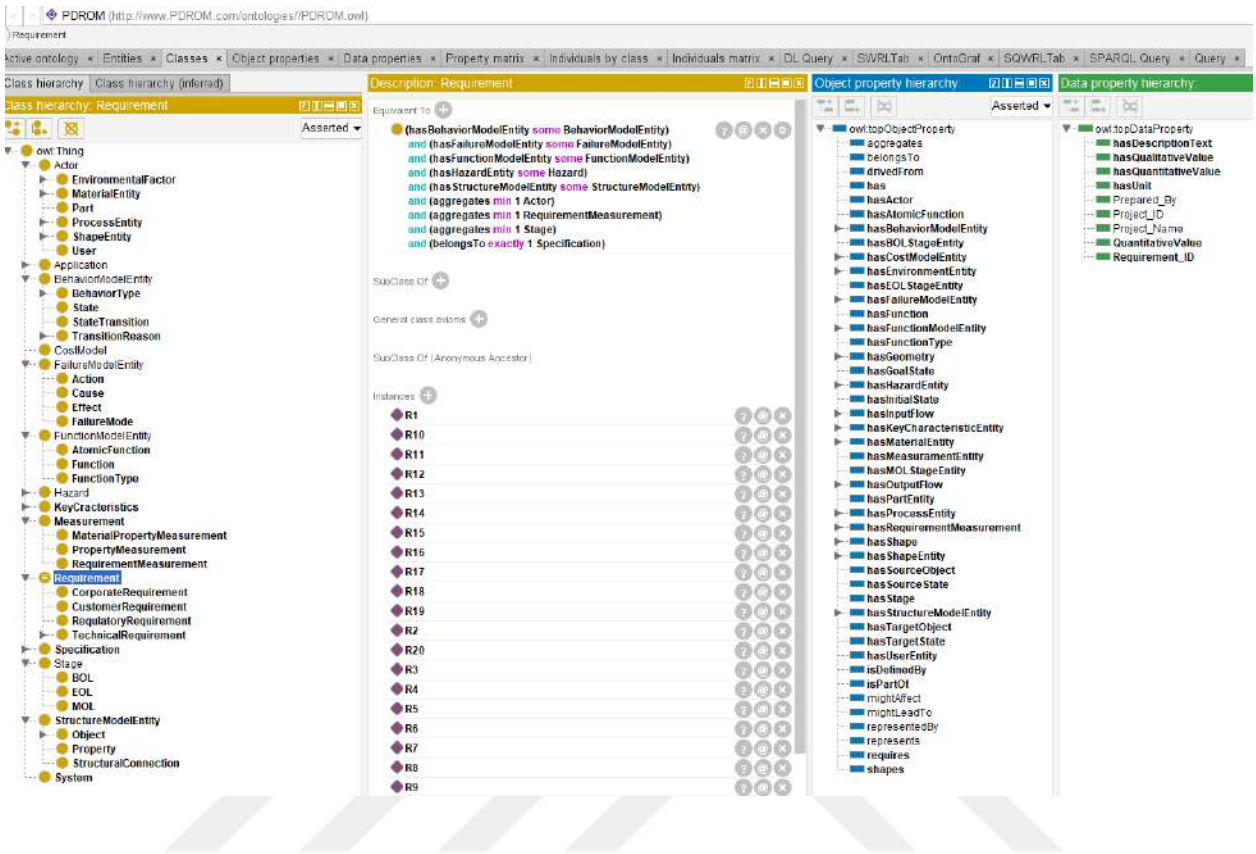


Figure 5.2: PDROM Implementation in Protégé

Max (max cardinality), and Exactly (exact cardinality) for cardinality restrictions.

Another important issue in implementing the PDROM is to include semantic axioms to represent the conditions of a certain class, which are a necessary condition called **SubClass Of** description and a necessary and sufficient condition called **Equivalent To** description in Protégé. Both types of axioms are implemented formally in OWL. Figure 5.3 represents a necessary condition that says if an individual is a member of **FunctionalRequirement**, it is necessarily a **Technical-Requirement** and it has to have minimum one **AtomicFunction** individual.

Similar to this example, **Requirement** class relates to **Component** and **Property** classes with necessary conditions, as seen in figure 5.4. But it does not mean that every class related to **Component** and **Property** classes is a **Requirement** class. In order to determine the member

Figure 5.3: Implementing the Semantic Axioms-Necessary Condition

of the class for an individual, necessary and sufficient conditions must be defined that make the class a Defined Class. As a defined class, **Requirement** class ***belongsTo*** exactly one **Specification**, ***aggregates*** minimum one **Actor**, ***aggregates*** minimum one **RequirementMeasurement**, ***aggregates*** minimum one **Stage** as shown in figure 5.4.

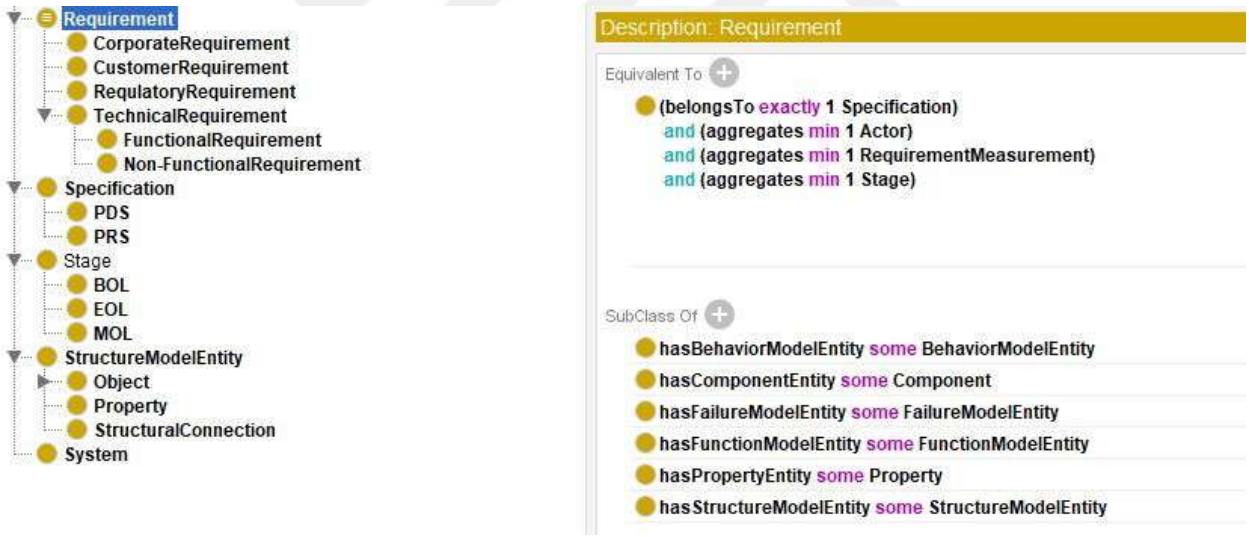


Figure 5.4: Implementing the Semantic Axioms-Necessary and Sufficient Conditions

### 5.1.3 Use of the Reasoners

The PDROM is implemented into the Web Ontology Language (OWL) above to carry out knowledge reasoning. As a piece of software, the Reasoner is one of the key features of the OWL-DL ontology, inferring logical consequences from a set of asserted axioms to automatically compute the

classification hierarchy and checking the logical consistency of the ontology. The inferred ontology class hierarchy shows us whether or not one class is a subclass of another class, and consistency checking analyzes the possibility of having any instances for a class and semantic contradiction within the definition of classes. Protégé allows users to add knowledge to an implemented PDROM in OWL by carrying out automatic validation processes with certain reasoners like Pellet or Hermit. Two types of the class hierarchy are defined in Protégé: "asserted hierarchy," which is a manually created class hierarchy, and "inferred hierarchy," which is automatically computed by the reasoner based on superclass-subclass and superproperty-subproperty relationships.

This work addresses two requirement specification criteria: (1) requirement completeness and (2) requirement inconsistency using the reasoning and inference power of the OWL ontology. Both of them can be carried out automatically by enriching the PDROM with semantic rules and semantic reasoners. These activities are illustrated below with an automobile brake case study. The last thing we will discuss before starting the case study is how semantic rules are modeled using SWRL and how semantic query rules are modeled using SQWRL in OWL.

### 5.1.4 Use of the SWRL and SQWRL

This section presents the semantic rules for the inference and query of the product requirement information for the PDROM implementation. The SWRL language in OWL is used to represent the rules for consistency validation and inference, and the SQWRL language, which is an OWL query language, is used to represent the rules for information extraction from an OWL ontology.

The consistency validation rules for revealing the meaning of several types of requirements, the inference rules for inferring types of requirement actors or stages, and the query rules for searching for requirements that belong to specific stages are developed and implemented in Protégé, as shown

in figure 5.5.

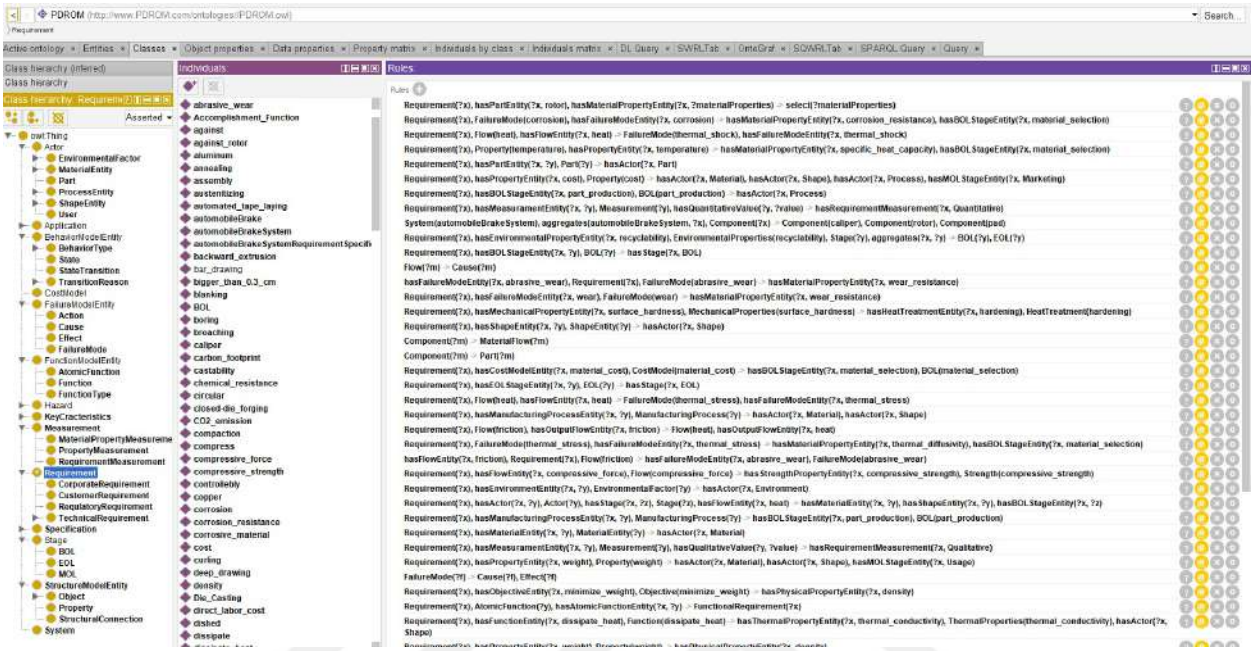

Figure 5.5: PDROM Implementation in Protégé and Semantic Rules in the PDROM

In order to reveal the requirement actor, type, stage, measurement, and rationale for the product requirements, the relationships which prescribe many restrictions are represented in the SWRL in terms of expert knowledge. These rules are empirical and subjective, and they can be extended if needed. They help users generate complete and consistent sets of requirements for a specific product during the design stage and further product life-cycle phases. Possible relationships between the material, shape, process, function, flow, failure mode, etc. must be well defined with semantic rules to generate the necessary requirements and complete the requirement set for the requirement specification.

Some query rules are also developed in Protégé for the PDROM OWL ontology to extract requirement information. These query rules search and extract the **Requirement** information from the PDROM OWL ontology based on desired information such as requirement I.D., description,

**Actor**, **Stage**, **Measurement**, **Material Property**, etc. Such queries are needed because product requirements related to a specific life-cycle activity such as material selection, manufacturing process, recycling, etc. or a specific part can be easily extracted from the requirement specification of a product.

Some of the SWRL and SQWRL rules that are created in Protégé for PDROM OWL implementation are illustrated and explained in the appendix B.1. As it is indicated before, these rules can be extended if needed. In the next case study section, the semantic incompleteness, inconsistency, and query of the populated requirement information process with the rules defined in the proposed PDROM are discussed.

## 5.2 Automobile Brake Requirements for the PDROM Implementation

An automotive brake is a system and it is an assembly of the caliper, pads, and rotor components for the purpose of slowing or stopping the motion of a wheel while the automobile runs at a certain speed. A rotor is clamped by the calipers and the brake pads when the brakes are applied. It is fastened to the vehicle's hub behind the wheel and is usually held on with two set screws, a large center nut, or simply the wheel itself. The brake example is used to demonstrate: (1) how to populate the PDROM model and (2) the consistency and completeness check on the requirement information.

### 5.2.1 Description of Requirements and Rules Development

The brake rotor is one of the main components of the automobile brake assembly. The main function of the brake is to reduce speed. It can be written in the functional requirement concept as an R1: "Automobile brake shall reduce speed safely". Figure5.6 shows the black box model for

the functional description with the brake's main function, input flows, and output flows. For the brake system, calipers compress pads against the rotor in order to create friction that retards the rotation to reduce speed. The energy of motion is converted into heat, and material is worn out, as a consequence of friction. Moreover, the represented function for the usage stage of the brake system might have input flows whose effects are not controllable, which are called "Noise Inputs." These input flows might cause crucial consequences in the long term. For example, 'water' is a noise input for the brake system, and it causes corrosion in the long term.

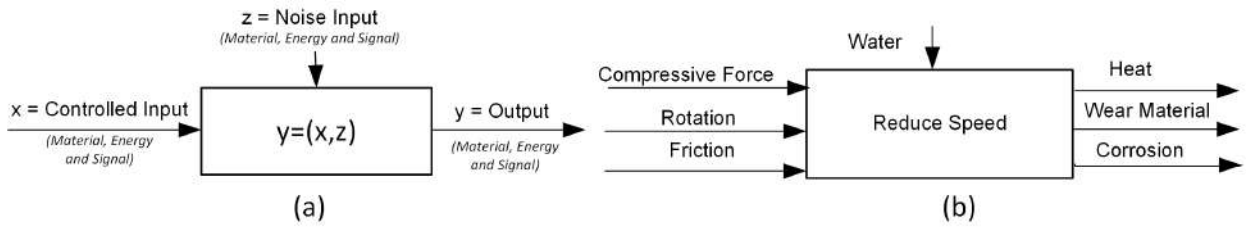


Figure 5.6: (a) Black Box Model and (b) Function&Flow Representation for Automobile Brake System

Some requirements for brake design can be defined as minimizing cost, weight, noise level, and resistances to vibrations, stresses, corrosion, and thermal shock under various conditions of load, velocity, temperature, and environment. The most important requirement consideration is the ability of the brake rotor component to withstand high friction and maintain less abrasive wear. Also, the rotor and pads must withstand high temperature, which is generated due to friction. This indicates that the working temperature of the rotor must be below the melting temperature of the components' materials.

Another design issue is that the tolerance of the rotor must be compatible with the rest of the wheel and brake parts. Ample space between the caliper and the rotor should be provided when the brake is not in action (non-braking time). Besides these requirements, some other considerations,

Table 5.1: Automobile Brake System Requirements

| Requirement ID | Requirement Description |
|---|---|
| R1 | Automobile brake shall reduce speed safely |
| R2 | Caliper shall compress pads against rotor |
| R3 | Rotor shall rotate with rim |
| R4 | Rotor shall transmit power from the caliper to the rim |
| R5 | Parts shall resist to vibrations |
| R6 | Parts shall generate friction |
| R7 | Parts shall resist to corrosion |
| R8 | Parts shall resist to thermal stress |
| R9 | Parts shall resist to thermal shock |
| R10 | Parts shall resist temperature up to 600 degree Celsius |
| R11 | Rotor and pads shall resist abrasive wear |
| R12 | Caliper shall keep distance between pad surface and rotor surface bigger than 0.3 cm during non-braking time to avoid unnecessary friction and wear |
| R13 | Rotor material shall be recyclable |
| R14 | Water pollution in part production shall be low rate |
| R15 | CO2 emission in part production shall be less than 120 g |
| R16 | Rotor shall be suitable for die casting and machining |
| R17 | Minimize weight and cost |
| R18 | Rotor shall resist compression force |
| R19 | Rotor shall dissipate heat |
| R20 | Rotor shall have high surface hardness |

which are very critical to the product design, can be; recyclability, pollution rate in production, assembly variables like time and cost, energy consumption in processing/use/reuse/recycling, and other design (geometry) constraints like good surface finish. Finally, the rotor design must consider the ease of manufacturing processes like casting and machining. The above requirement description of the case problem can be summarized into twenty requirement sentences listed in table 5.1.

These requirements are defined based on experience and expert knowledge. This knowledge and experience are converted into semantic rules and implemented in PDROM. PDROM provides

an architecture to infer some of these requirements based on pre-defined information and rules, and to represent requirements semantically. Figure 5.7 shows how requirements are generated from the given function and flow information. The main function of the brake system, 'reduce speed' is divided into sub-functions. Each sub-function is represented with input and output flows. As discussed under the semantic rules above, requirement information is inferred based on the rules and defined information. For example, the input flow 'compressive force' requires the structure to resist the force. In addition, the output flow 'heat' requires to generate information about 'heat dissipation', 'thermal stress', and 'thermal shock'. This example illustrates that PDROM represents the given and inferred requirement information with its class hierarchy and relationship. The requirement sentences as shown on the right-hand side of figure 5.7 are generated based on the inferred requirement information which is represented on the left-hand side of the figure. The reason for developing this illustration is to represent the requirement generation procedure through the semantic rules and requirements rationale through the model elements.

Requirement generation requires an understanding of the structure, function, behavior, and failure concepts of the brake assembly, as well as its material, manufacturing process, end of life information. Also, the underlying requirement rationale provides the reasoning support to explain why the requirements of the objects exist and what assumptions and constraints the designers depend on generating those requirements. In this example, the integrated brake system information provides a qualitative design reasoning mechanism by relating product requirements, design intents, design constraints, and many implicit design assumptions at different abstraction levels of the brake structure.

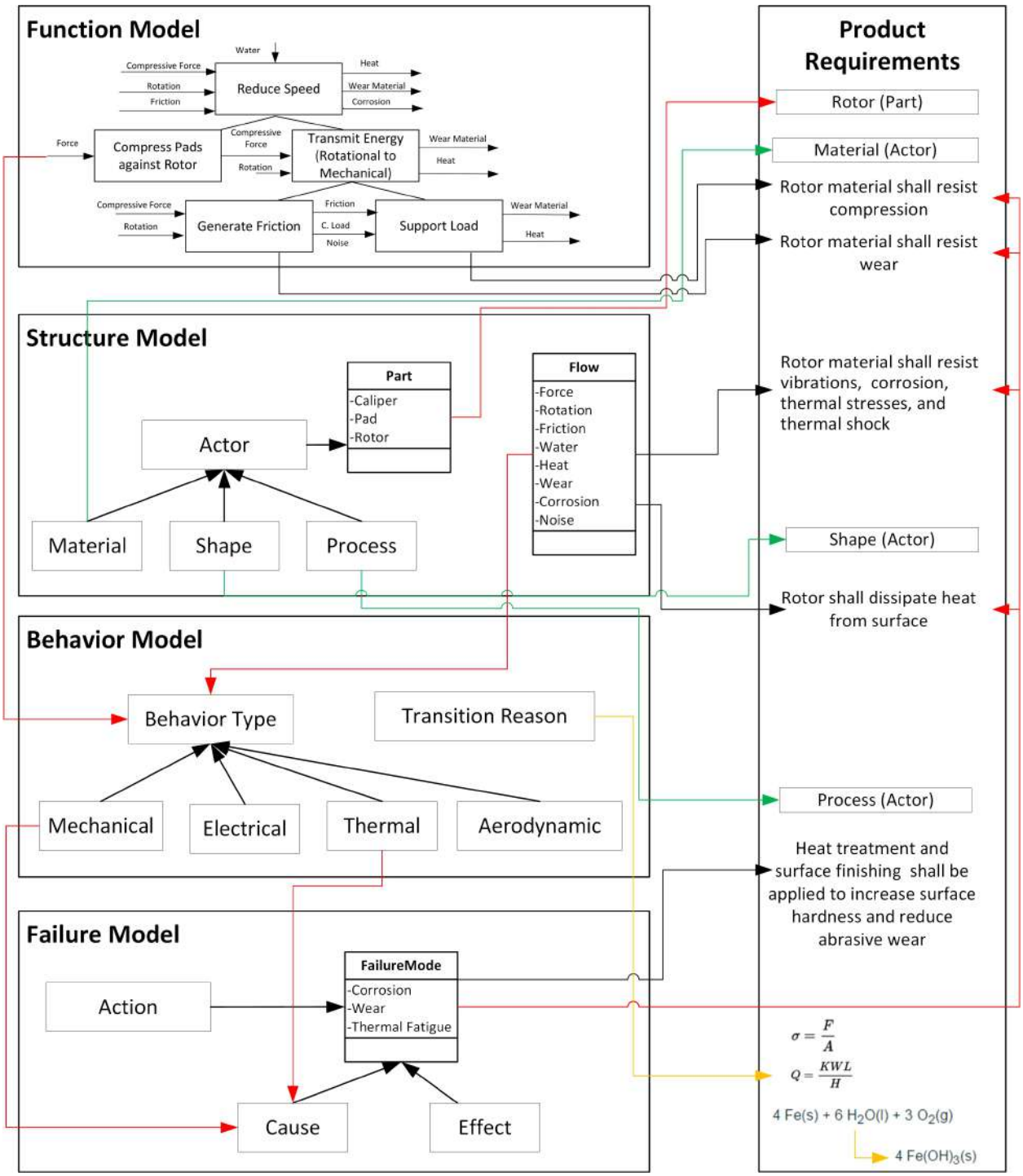


Figure 5.7: Requirement Rationale Model Utilization and Product Requirement Generation for Automobile Brake System

### 5.2.2 Instantiation of the PDROM Model and Completeness Check

Using the proposed PDROM, product requirements can be represented in a structured way. In Table 5.1, product requirements like R1, R2, R3, etc., are represented as textual descriptions that emphasize the critical factors of the brake system design. These requirements are elaborated and instantiated according to the PDROM model as shown in Table 5.2. It shows that every product requirement has a type, state, actor, and measurement. Figure 5.8 shows how the detailed instance mapping corresponds to requirements 12 and 13. Requirement R12 (Caliper shall keep distance between the pad surface and rotor surface bigger than 0.3 cm during the non-braking time to avoid unnecessary friction and wear) and R7 (Rotor material shall be recyclable) for the brake system are instantiated in figure 5.8. The selected requirements imply that part shape and material should be studied carefully. A key issue with this instance mapping is to instantiate the **Actor**, **State**, and **RequirementMeasurement** classes which are described above.

In the example, R12 has two **Shape** actors (rotor surface and caliper surface) and three **Part** actors (caliper, pad, and rotor), and it is considered in the **MOL** (usage) stage. Besides that, **RequirementMeasurement** should be identified to see whether the requirement is **Quantitative** or **Qualitative**. R12 is a **Quantitative** requirement, which measures the distance between the rotor and caliper as represented in the table 5.2 below. Similarly, R13 has one **Part** actor (rotor) and one **Material** actor (rotor material), and the **RequirementMeasurement** is qualitatively represented because the recycle percentage of the part material is not defined in the requirement list. Moreover, R13 associates to the **EOL** (disposal) and **BOL** (material selection) stages. It means that R13 must be considered during activities of these stages.

Representing the R12 with the developed OWL schema make it more complete by providing

Table 5.2: Categorization of the Automobile Brake System Requirements in PDROM

| ID | Type | Stage | Actor | RequirementMeasurement |
|---|---|---|---|---|
| R1 | Functional | MOL | Part | Qualitative |
| R2 | Functional | MOL | Part and Shape | Qualitative |
| R3 | Functional | MOL | Part | Qualitative |
| R4 | Functional | MOL | Part | Qualitative |
| R5 | Functional | MOL | Part | Qualitative |
| R6 | Qualitative | BOL and MOL | Part, Shape and Material | Qualitative |
| R7 | Functional | BOL and MOL | Part and Material | Qualitative |
| R8 | Functional | BOL and MOL | Part and Material | Qualitative |
| R9 | Functional | BOL and MOL | Part and Material | Qualitative |
| R10 | Functional | BOL and MOL | Part and Material | Quantitative |
| R11 | Functional | BOL and MOL | Part and Material | Qualitative |
| R12 | Functional | MOL | Part and Shape | Quantitative |
| R13 | Non-Functional | BOL and EOL | Part and Material | Qualitative |
| R14 | Regulatory | BOL | Process and Environment | Qualitative |
| R15 | Regulatory | BOL | Process and Environment | Quantitative |
| R16 | Non-Functional | BOL | Part, Process, Shape and Material | Qualitative |
| R17 | Non-Functional | BOL and MOL | Part, Process, Shape and Material | Qualitative |
| R18 | Functional | BOL and MOL | Part and Material | Qualitative |
| R19 | Functional | BOL and MOL | Part, Shape and Material | Qualitative |
| R20 | Non-Functional | BOL and MOL | Part, Process, Shape and Material | Qualitative |

additional information. This representation also helps to improve the completeness of the whole requirement set by taking the advantage of inference and reasoning mechanisms. Figure 5.9 shows how the Protégé reasoning mechanism is used to infer additional requirement information, which is required for the complete class definitions of **Requirement** and **Specification**.

The SWRL rules developed for inferring the instances of PDROM are discussed above. In figure 5.9, the left column shows the manually entered information of the R12 and the right column shows the inferred information of the R12 after reasoning. The inference mechanism infers the following information for the R12:

- Requirement Actor

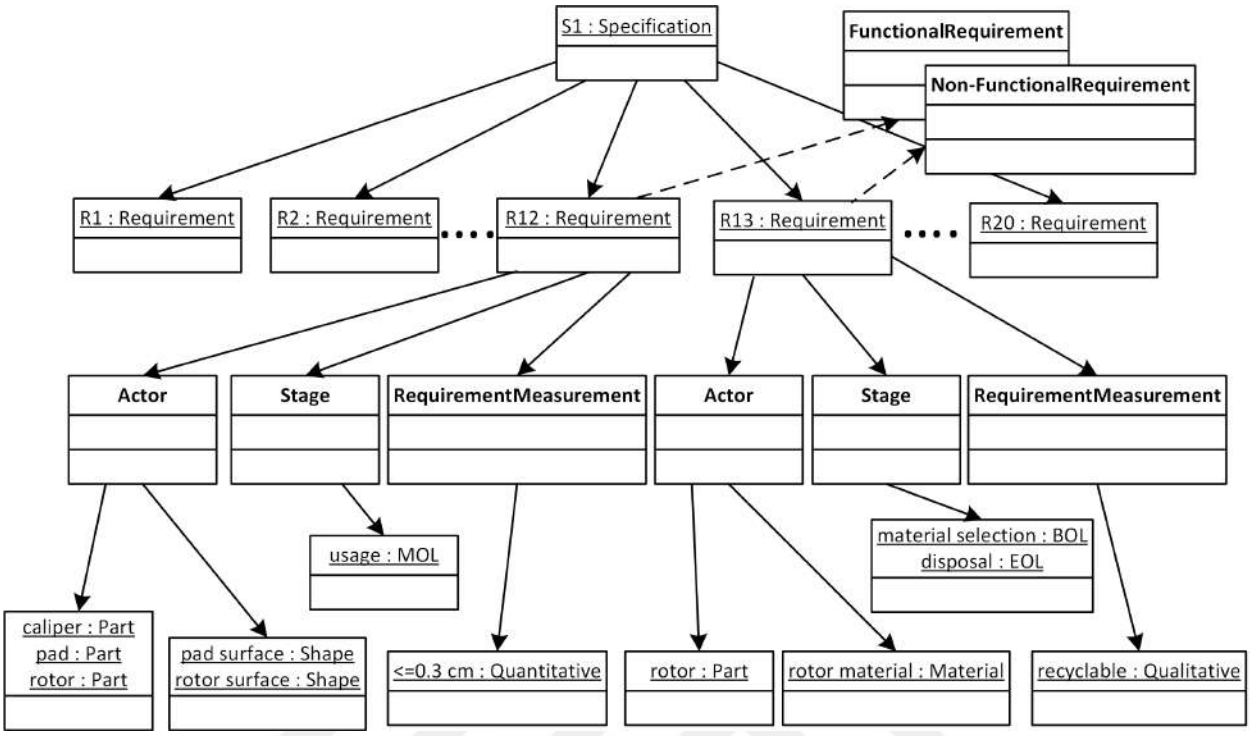


Figure 5.8: Instantiation of the PDROM Model-An Example

- Requirement Stage
- Requirement Type
- Requirement Measurement
- Structure, Behavior, Function and Failure Models

The R12 is derived from R6, R8, R9, and R11 and contributes to the completeness of the automobile brake requirement specification. Also, inferred information contributes to the completeness of the R12 itself.

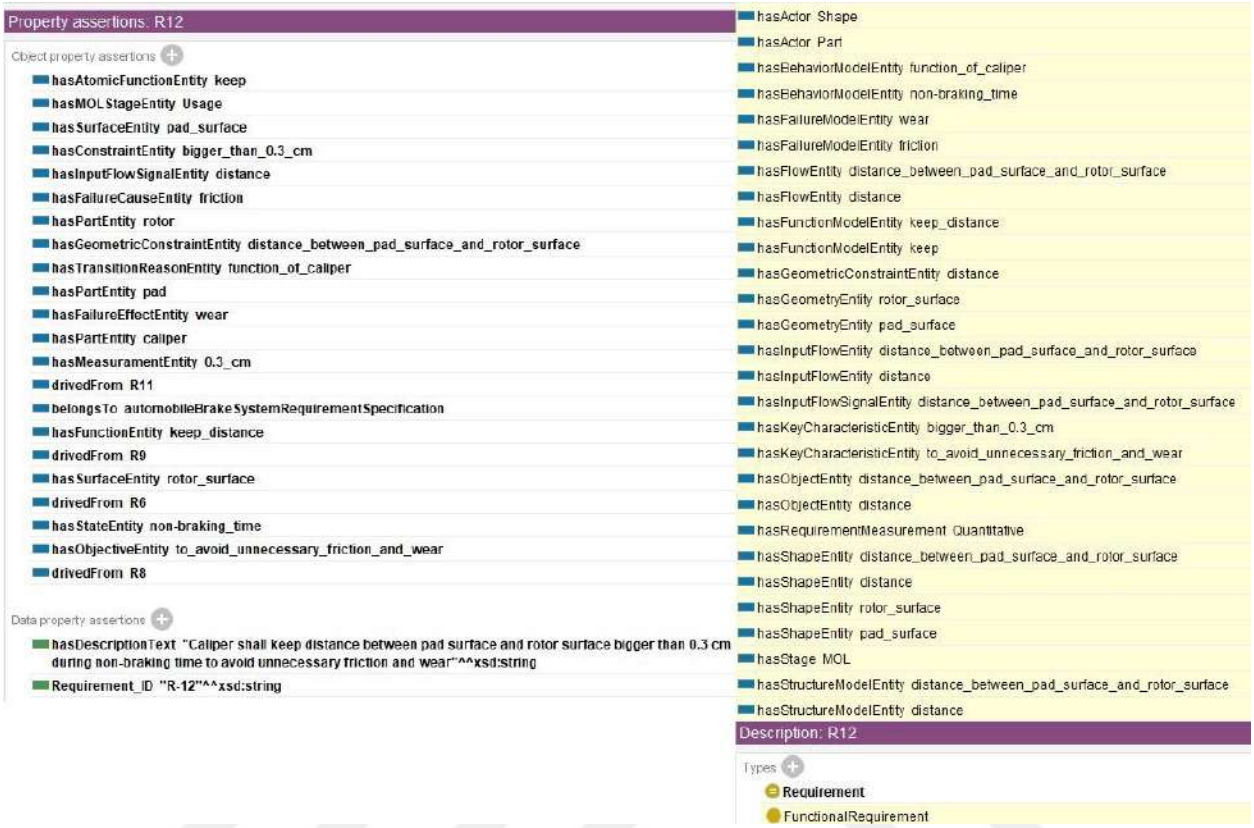


Figure 5.9: Inference and Reasoning Mechanisms for R12

### 5.2.3 Consistency Check

As mentioned before, the semantics embedded into the PDROM can be used to detect any inconsistency in the instantiated information structure. The left portion of figure 5.10 shows the original information storage and inferred knowledge in Protégé corresponding to requirement 13 (R13). Without entering any **Stage** information, PRDOM infers that this requirement relates to the *material selection* stage (**BOL**) and *disposal* stage (**EOL**) and completes the requirement definition. If one more information which is inconsistent with the remaining information such as ”R13 is related to the *usage* stage (**MOL**)” is intentionally added, such inconsistency can be captured automatically, as shown in figure 5.11. As it is evident in figure 5.11, inconsistency occurs

since ”*usage*” stage is an instance of **MOL** and it contradicts the definition of rule represented in figure 5.11.

Property assertions: R13

Object property assertions

- hasEnvironmentalPropertyEntity recyclability
- hasRequirementMeasurement Qualitative
- hasMaterialEntity rotor_material
- hasPartEntity rotor
- belongsTo automobileBrakeSystemRequirementSpecification
- hasActor Material
- hasActor Part
- hasBOLStageEntity material_selection
- hasEOLStageEntity recycling
- hasMaterialEntity recyclability
- hasMaterialPropertyEntity recyclability
- hasStage EOL
- hasStage BOL
- hasStageEntity material_selection
- hasStageEntity recycling

Data property assertions

- hasDescriptionText "Rotor material shall be recyclable"^^xsd:string
- Requirement_ID "R-13"^^xsd:string

**Original R13 Storage and Inferencing**

Property assertions: R13

Object property assertions

- hasEnvironmentalPropertyEntity recyclability
- hasStageEntity Usage
- hasRequirementMeasurement Qualitative
- hasMaterialEntity rotor_material
- hasPartEntity rotor
- belongsTo automobileBrakeSystemRequirementSpecification

Data property assertions

- hasDescriptionText "Rotor material shall be recyclable"^^xsd:string
- Requirement_ID "R-13"^^xsd:string

**Inconsistent R13 Storage**

Figure 5.10: R7 Storage in Information Structure

Explanation 8 Display laconic explanation

Explanation for: owl:Thing SubClassOf owl:Nothing

1) MOL SubClassOf Stage
2) hasStageEntity Domain Requirement
3) DisjointClasses: BOL, EOL, MOL
4) recyclability Type EnvironmentalProperties
5) R13 hasEnvironmentalPropertyEntity recyclability
6) R13 hasStageEntity Usage
7) Requirement(?x), hasEnvironmentalPropertyEntity(?x, recyclability), EnvironmentalProperties(recyclability), Stage(?y), hasStageEntity(?x, ?y) -> BOL(?y), EOL(?y)
8) Usage Type MOL

Figure 5.11: Inconsistency Checking

## 5.3 Query Processing

After one creates the requirement instances (individuals), runs the reasoning, and infers the instances, the OWL file can be queried to retrieve information based on how semantic queries are developed in Protégé. The semantic queries are developed to extract information from OWL ontologies using an SWRL-based query language, which is SQWRL. The retrieved information might be from the information that is manually entered or information that is inferred by SWRL rules. In Protégé, SQWRL queries are created and executed through the SQWRL Query Tab. Using this tab, the user can select any query rules from the query table and execute it by using the ”Run” button and can review the displayed results in tabular form. Examples below show how information is successfully retrieved from PDROM OWL by using SQWRL query rules and how beneficial it is to use this query processing.

**Query 43** lists all requirement individuals with requirement I.D., requirement description, **Actor**, **Stage** and **Measurement** information. Figure 5.12 displays the information, which is queried by the query rule 43. This query creates the same information, listed in table 5.2 that we created above.

**Query 48** lists all material property individuals with their corresponding requirement individuals for the rotor part as shown in figure 5.13. These kinds of queries are very convenient and beneficial for engineers and designers involved in product development to reduce human errors, reduce time while creating and searching for a certain type of information.

Figure 5.12: Requirement Individuals with Necessary Information

Figure 5.13: Material Property and Requirement Description Instances for Rotor Part

## 5.4 Summary of PDROM OWL Implementation

This chapter discusses the PDROM implementation in Protégé and the instantiation of the Model by translating UML models into the Web Ontology Language (OWL). Web Rule Language (SWRL) rules for formal machine reasoning and interpretation of the PDROM for requirement analysis and SQWRL semantic query rules for searching requirement information in the classified ontology are represented. PDROM OWL implementation is summarized in figure 5.14 according to axiom, class, object property, data property, named individual, and rules and queries count. All defined names for these five schemas and the complete PDROM OWL implementation can be accessed

from http://product-requirement-ontology.sourceforge.net.

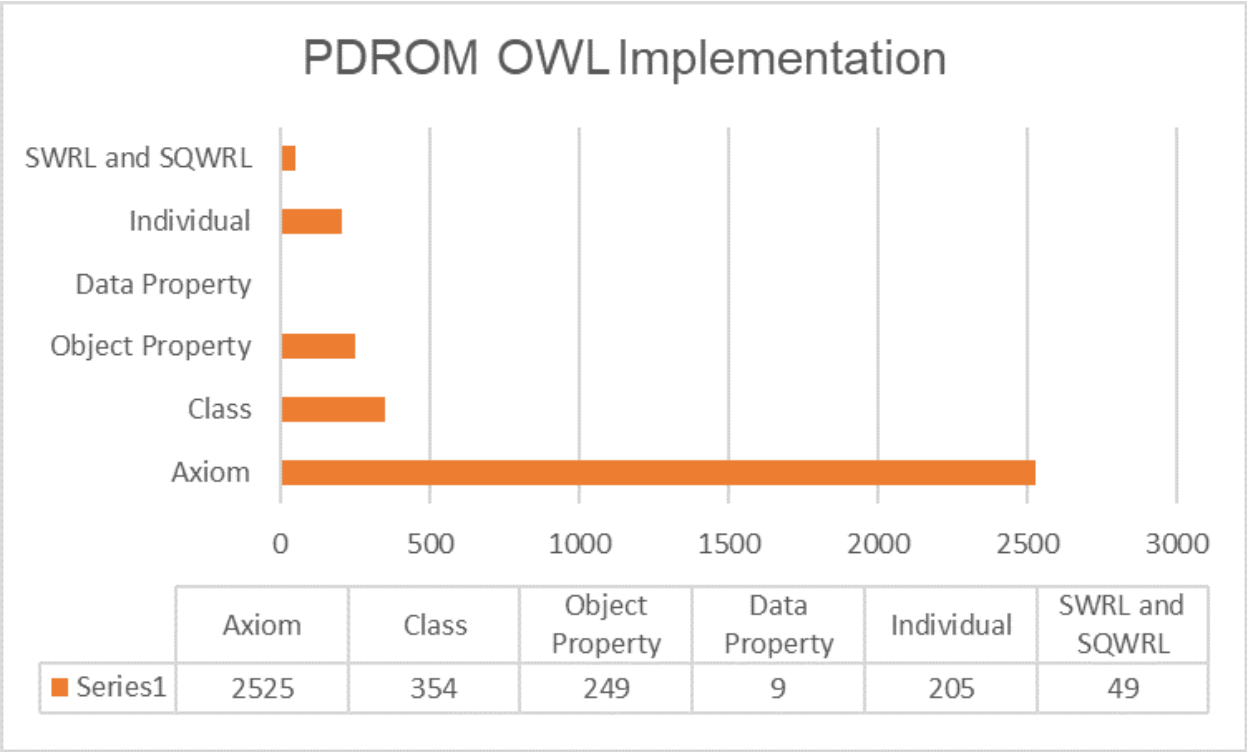


Figure 5.14: PDROM OWL Implementation Summary

# CHAPTER 6

# PLM SYSTEM AND COMPUTATIONAL PLATFORM IN THE REKMA

In this chapter, the development of the PLM system, computational platform, and communication mechanism in requirement knowledge management architecture is presented. This chapter starts with the implementation of the information model's structure in a PLM tool (Aras Innovator). Then, this chapter explains how the PLM system is used to gather, manage, and represent product requirements and related information. Then, it discusses the development and use of the computational platform which is integrated into the PLM system. Finally, this chapter presents a communication mechanism among a product, the PLM system, and computational platform to transport data and file and to convert relational data from the PLM system into the Resource Description Framework (RDF).

## 6.1 PLM System Development

As it is discussed in chapter 3, the first task for the development of the REKMA is ontological model development for the product requirements and related domains. Ontology development for the product requirements and related information is presented in chapter 4. The second task is to develop a PLM system as a data repository, a knowledge source that is composed of a relational database, file storage, and a user interface. It enables the users to input and access the product requirements with related data and documents. For this purpose, the ontological model is first simplified to convert the semantic model structure to the data model. Then this

data model is implemented in a PLM tool by customization. Aras Innovator PLM [130], which is an open-source PLM software for engineering design is customized by using requirement classes and relationships described in the PDROM. The PLM system provides interfaces to stakeholders, include end-users for the product requirement representation and generation while it consists of a data management platform and data repository. A virtual engineering product can be represented by storing and organizing product-related data in the repository. Aras Innovator uses the concepts Item and Relationship for the objects and their connection. In Aras Innovator everything is an Item. Items may have relationships, which are also Items that have a source and related Item Properties forming an Item configuration. Aras Innovator stores all the data and information with the necessary life-cycle information.

Requirement and requirement-related data of the product are entered into the PLM repository manually by the user or autonomously through an API (Application Programming Interface). Requirement knowledge generated by using manually and autonomously entered data through data analytic applications is also represented in the PLM system. It makes requirement data and knowledge available for the engineers, users, as well as to people involved in the product development process. It also possesses data fusion and works with the computational platform for decision-making to support the requirement specification. The customized Aras PLM supports a new level of product requirement storage, traceability, and integration between product life-cycle stages by addressing different facets of requirement specifications.

### 6.1.1 Customization of the PLM System based on PDROM Structure

This part focuses on the development of a PLM system for the product requirement knowledge management based-on proposed requirement information model architecture. This system pro-

vides the full capability to access, use, and maintain product requirement information, as well as it supports the requirement generation and representation processes related to all activities across the product life-cycle phases.

As for a simple or a complex product system, product requirements as for whole and each individual requirement have their life-cycles. The PLM system for product requirements needs to accommodate all the data and activities of the product development and operation with their life-cycles. In order to manage this huge volume of data and information content of a PLM system for the product requirement management, concepts of the traditional PLM system are extended. Customized PLM system within the REKMA addresses the problems and issues related to the collection, storage, manipulation, and mining requirement knowledge from historical as well as real-time streaming data using varieties of data analytics tools, and maintains compatibilities and integration between different product requirements. As stated above, the data structures of the PDROM Part I, PDROM Part II, and domain ontologies; MIM, PIM, and SIM are implemented in Aras Innovator as shown in figure 6.1. The PLM system is supported by a computational platform to reveal the autonomous capabilities of the ontology-based requirement information model such as consistency analysis, requirement information inferencing, requirement text generation, and validation, etc.

The left menu on the ARAS PML is called the Table of Contents (TOC) where it displays different "categories" in a tree-view. Figure 6.1 shows some important components (class, relationship) of PDROM. TOC displays requirement actors: **Part**, **Process**, **Shape**, **Material** and **Environment** under Design category, **Structure**, **Behavior**, **Function** and **Failure** models under Engineering System category, **Requirement**, **Key Characteristics** and **Requirement Specification** under the Requirement category, **Dataset**, **Data Analytics Model** and **Applications** under the Data Analytic

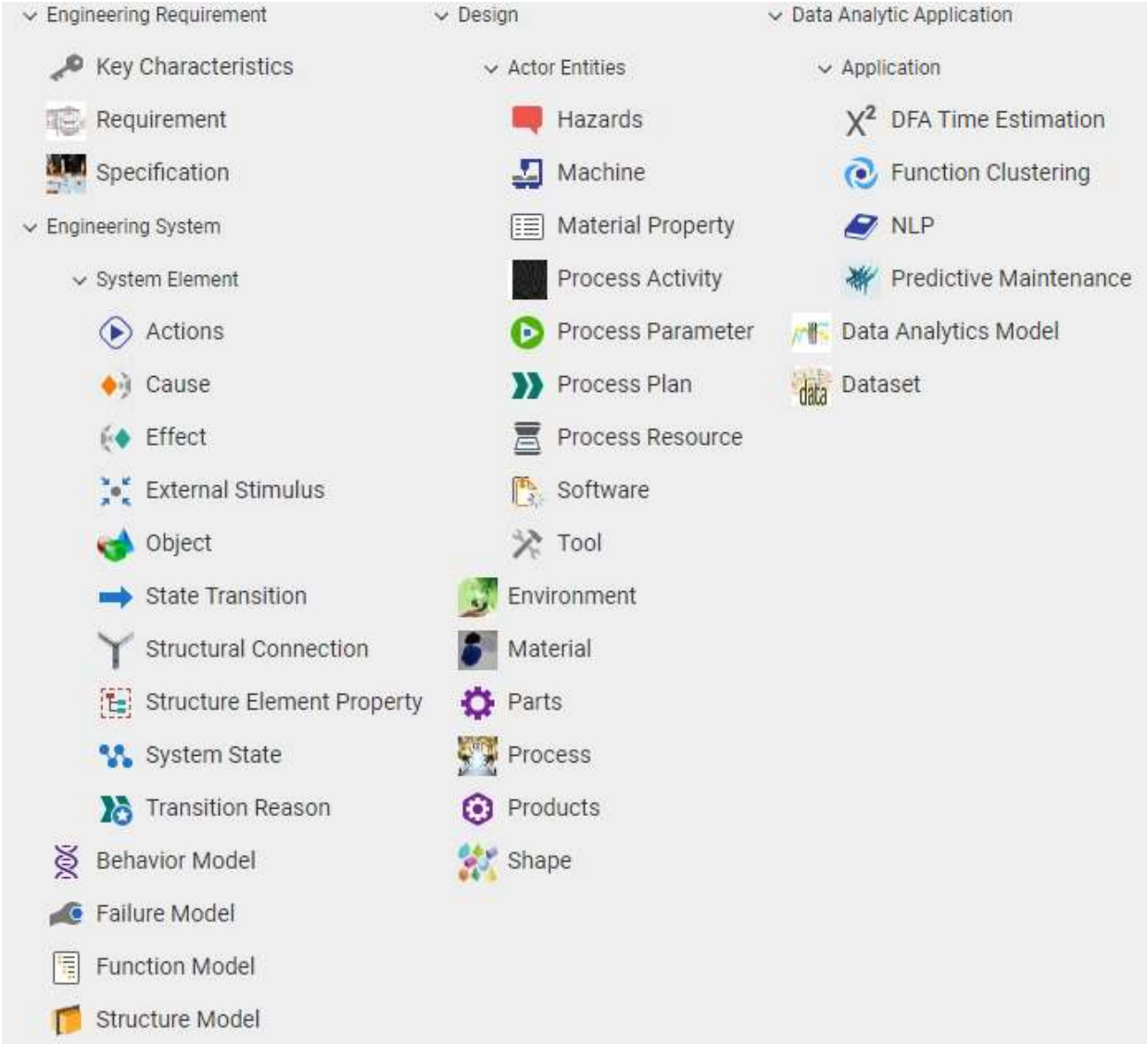


Figure 6.1: PDROM Contents in the PLM System

Application category and **Requirement Specification Documents** under the Documents category.

Relationships are displayed with tabs on the bottom of the window that bind ItemTypes. As seen in figure 6.3 and 6.4, Requirement item is related to requirement actors and SBFF models using Relationship ItemTypes. Whenever a requirement is created, the user can add requirement related information and requirement rationale using these relationship ItemTypes. As an example, the requirement of a brake system represented as an 'Automobile brake system shall reduce speed

safely ' in textual format. After entities are created, the function name 'reduce speed' and function verb 'reduce' entity types and entity names are created. This information can be represented in Function Model ItemType as seen in figure 6.2 and relates to the requirement through the Function Model relationship ItemType.

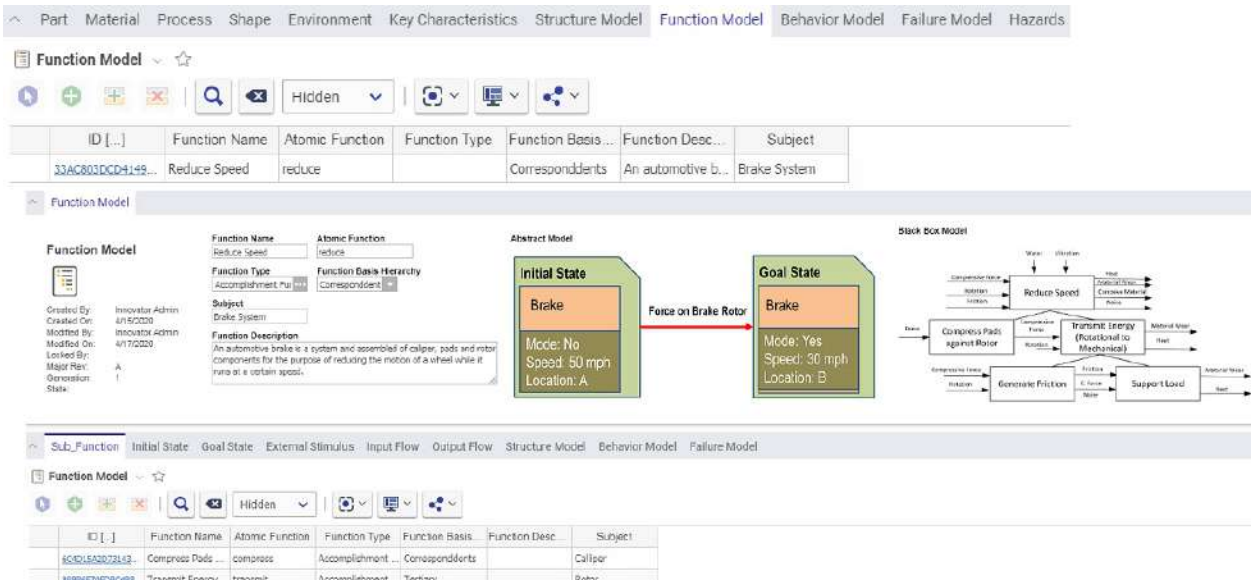


Figure 6.2: Function Model in PLM for Brake System

### 6.1.2 Representation of the Product Requirements In The PLM System

The Aras PLM displays each item of these categories on the window with the item's Form and Item's Relationships. The requirement item is represented with two Forms of the user interface. The first Form displays some of the custom properties of the product requirement data model that is created based on the PDROM structure, including Requirement Number, Requirement Type, Requirement Measurement, Requirement Text, Actor, and Stage as seen in figure 6.3. This is the interface that the user is presented with when he/she is creating a new requirement item. These custom properties are the requirement information that only displays the categories for the user

inputs. This interface does not show all custom properties created for Requirement ItemType.

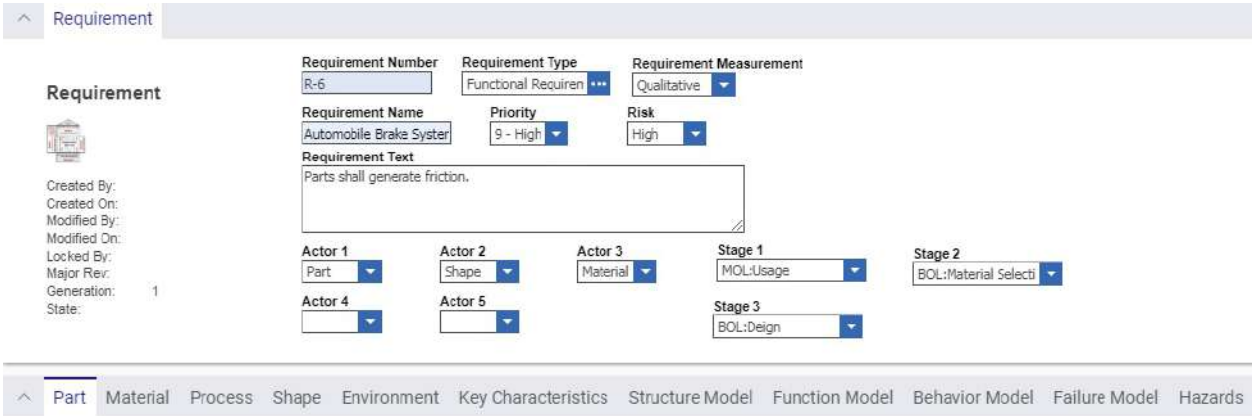


Figure 6.3: Requirement Form When User Inputs Requirement Information to Create a New Requirement Item

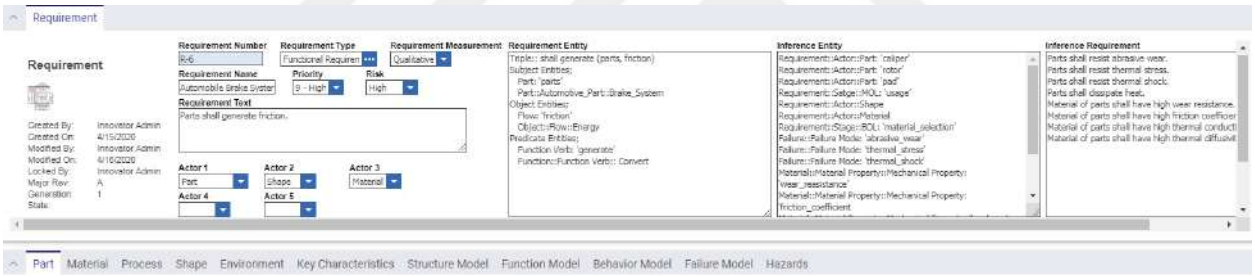


Figure 6.4: Requirement Form When the PLM Gets the Information from the Computational Platform

Two-way communication between Aras Innovator and a third-party computational platform exposes data through an API. A brake system case study which is discussed in the previous chapter is represented in the PLM system as a proof-of-concept. Once the requirement data is added to the first interface (Form), the computational platform executes data analytics applications which use this data and create three types of requirement information; (i) requirement entities for requirement text, (ii) inference entity created by using entities of requirement text and entities given by the user, and (iii) inference requirement in the textual format as seen in figure 6.4. The PLM system allows

the users to edit the information that is created by the computational platform. They can keep the requirement information as it is, edit, or completely remove them.

### 6.1.3 Software Installation and Data Collection

Aras Innovator is an object-oriented, web-based PLM platform as part of a service-oriented architecture (SOA) that offers a huge advantage to choose it as the PLM software to implement PDROM. Similar to Ontology, Aras Innovator uses the concepts **Item** and **Relationship** to abstract arbitrary objects and connections between objects. It has the following advantages for our project:

- It is an open-source PLM software for product design
- It can be easily customized for a specific project
- Can be integrated with third-party software systems
- Runs on the web in a hosted environment or the cloud
- Provides control on user permission and workflow
- Several users can access it through the internet at the same time

For this project, a virtual server is requested for our research lab from Syracuse University's Information Technology Services (ITS) and Aras Innovator is downloaded and installed on one of the University's servers by IT staff. It requires a server running MS Windows Server 2008 or higher with Internet Information Server (IIS), NET 4.5.2 or higher, and MS SQL server. Two databases are created on this server: (i) InnnovatorSolutions which is the main database to implement PDROM and (ii) EAGER that is used to study Structure-Behavior-Function (SBF) ontology. InnovatorSolutions is contributed by me with the administrative privilege and Design for Manufacturing (DFM-692 ) class students as a user through the years 2015, 2016, and 2017. It is accessible by defined users. All users have an option to access Aras Innovator on the university server

from on-campus and off-campus through the internet using a web address with the defined user's username and password. It is unnecessary to download and install the innovator software by users because it works as a client-server.

Users access and login the system through the server web address: http://128.230.60.59/InnovatorServer as seen on the left hand side of the figure 6.5. it can also be accessed with Remote Desktop using the computer name which is lcs-vc-plm.ad.syr.edu as seen on the right hand side of the figure 6.5.

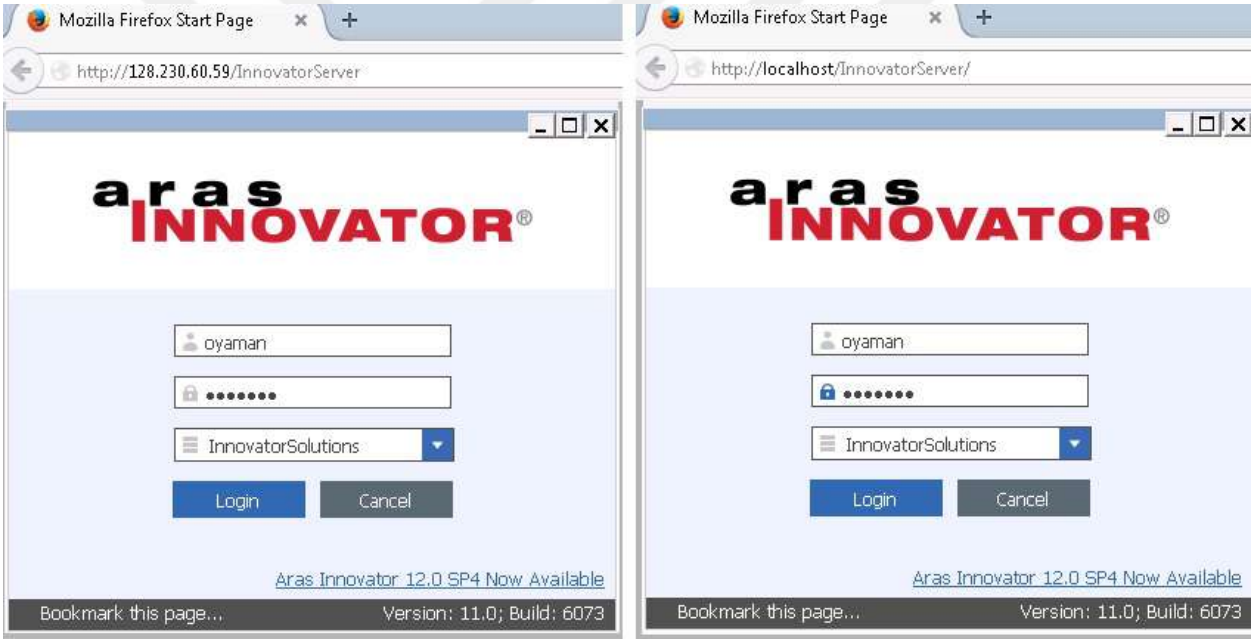


Figure 6.5: Aras Innovator login window

The first step after configuring the Innovator database based on the PDROM is to create users and define their permission. Every year around thirty three students are registered for the DFM-692 course. All these students are added to the system as users and a username and default password are created for each user. If they want they have an option to change their password later. Three students come together for a group project and group identities are created for the DFM-692 every

year in the Innovator database. For this database, thirty group identities are created and data for thirty products such as kitchen appliances, drones, etc. are created in the database based on customized concepts. Each user and group identity are given permission to create, update, and delete the PDROM items while students of one group could not access information created by other groups. After the information is created, the admin checks the correctness of data, promotes and releases it using the Life Cycle item, or asks the user to edit data if anything wrong.

To support the requirement representation, information extraction, and retrieval, and requirement text generation a computational platform is developed. Also, a communication mechanism to support the data exchange and data conversion is created. They are discussed in the following section.

## 6.2 Computational Platform and Integration of the Systems

The third task to build the REKMA is to develop a computational platform that includes requirement information extraction and retrieval applications, a knowledge-based system for the requirement information inferencing and text generation, and a communication system for the file and data transfer. The main purpose to develop a computational platform is to use analytical applications and enable requirement representation within the proposed format and autonomous requirement information generation. These applications work on requirement data which are inputted to the PLM system or streamed from the product life-cycle phases directly. Therefore, an integrated architecture and communication mechanism are necessary among the product, PLM system, computational platform, and RDF knowledge-base. The communication mechanism supports the two-way data and file exchange between the computational platform and the PLM system. It also translates the product requirement data and metadata from the PLM into the RDF

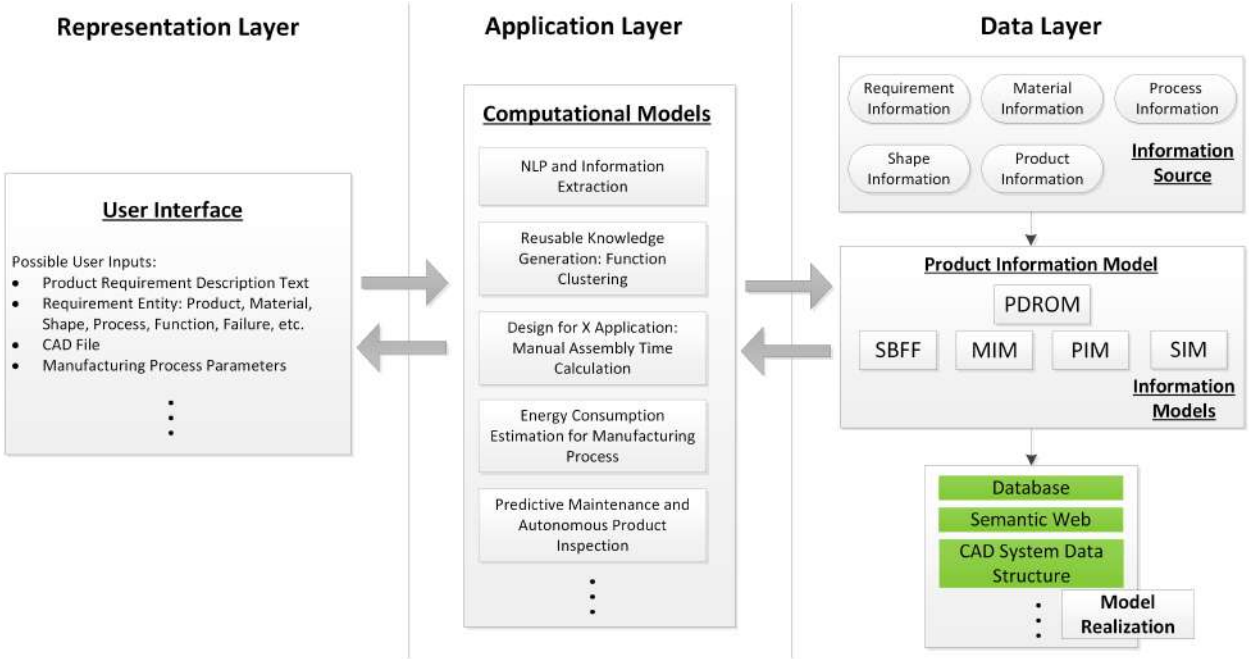


Figure 6.6: The Proposed Three-Layered Framework for Information Model Utilization

knowledge-base.

The applications in the computational platform extract and retrieve the product requirement semantics by using syntax analysis, semantic analysis, domain ontologies, and many AI tools and techniques. A three-layered framework [39] is proposed for the PDROM utilization in the PLM system and integration of the computational platform as shown in Figure 6.6. This framework is organized hierarchically with three layers: Data Layer, Application Layer, and Representation Layer.

Data Layer: In this layer, the proposed PDROM is implemented in the PLM relational database. A system data structure that integrates product requirements with product design information can be stored according to the PDROM data structure. The stored data in a relational database can be also converted into RDF format for the semantic web and especially for the distributed working environment applications.

Application Layer: The application layer contains computational applications to aid information extraction, decision making, and requirement information generation through the product life-cycle processes. The computational platform can have many computational applications, including natural language processing, energy consumption estimation, manual assembly time calculation, etc. to support product requirement representation, validation, execution, and generation.

Representation Layer: The representation layer has a user interface that handles the user's inputs and displays the requirements in the proposed format. User inputs are product requirement texts and related PLM data which are defined during product life-cycle phases, especially at the design phase. While for most of the applications, ARAS PLM is used for a user interface, sometimes computational platform applications might have their own user interface.

This three-layered framework is proposed with consideration of the closed-loop PLM. The requirement knowledge management in the CL2M also requires a communication and integration among a product, the PLM system, and computational platform to support the autonomy for the use and development of the product requirement information by the computational applications. It allows the computational platform a full access, use, and maintenance of requirement and related product information. It also supports the PLM system by accommodating the data and activities involving the computational model development to facilitate interactions between product engineers, software engineers and data scientists. General framework to manage product requirement and requirement related data through CL2M as shown in figure 6.7. Detailed representation for components of this system overview is discussed below.

In this system architecture, requirement and requirement-related data of the product can be entered into the PLM repository manually or through API (Application Programming Interface) autonomously. Computational applications can generate knowledge from manually and autonomously

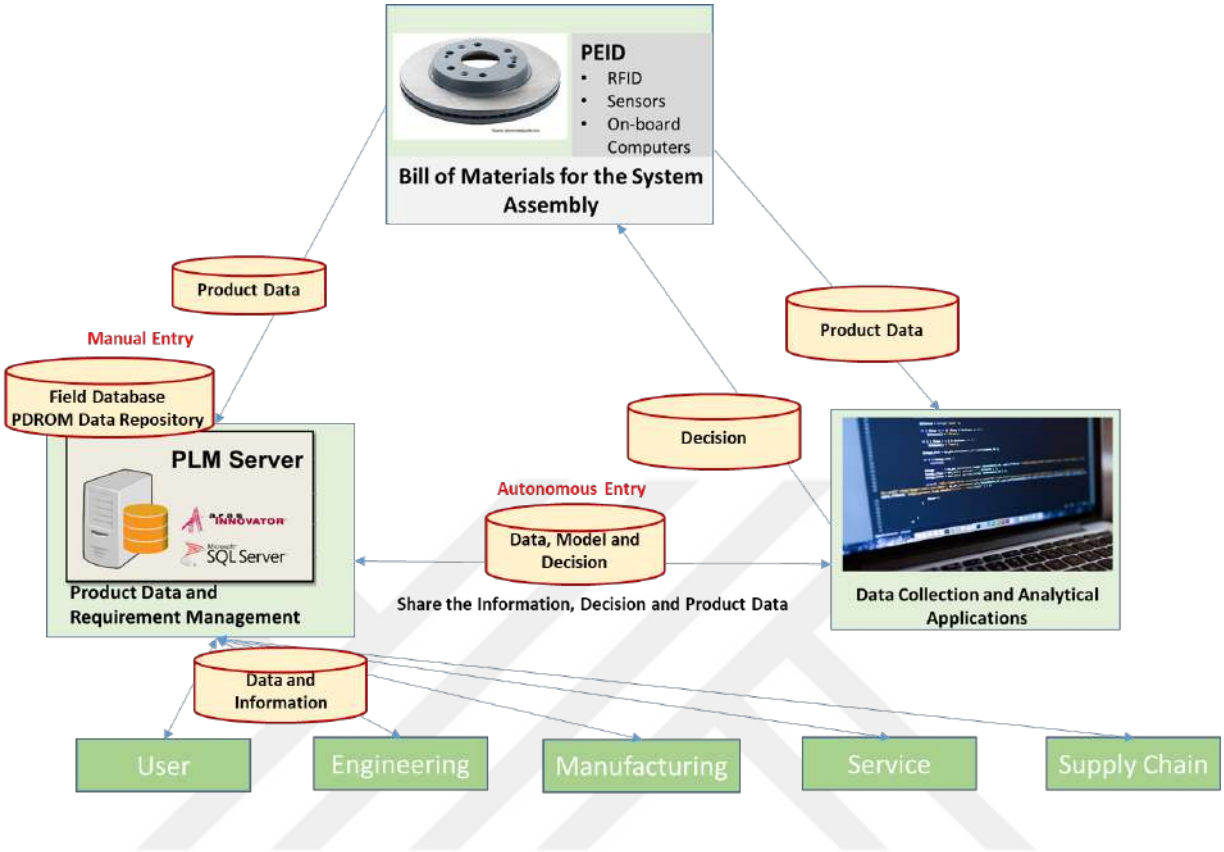


Figure 6.7: General CL2M Framework for the Requirement Data Management

entered data through data analytic applications that would increase the reliability of product requirement specification and provide information about product requirements to engineers, users, as well as to people involved in the product development process. The PLM platform with the PDROM data structure possesses data fusion and has an integrated computational platform for decision-making to support the requirement specification.

The REKMA with this system integration supports a new level of product requirement storage and management across the product life-cycle phases by addressing different facets of requirement specifications. As discussed above, a communication mechanism for the integration of the systems to support product requirement representation and generation through the product life-cycle phases. Below, it is introduced how to establish a communication mechanism among product,

PLM system, and computational platform, before the main computational application which is the product requirement information extracting and retrieval is discussed.

### 6.2.1 Data and Information Exchange

A communication framework needs to be built to support the data and file sharing among the PLM system, the computational platform, and the product. While the computational platform is mainly for information extraction and retrieval, it communicates with the PLM system through C# REST (Representational State Transfer) Web Server that is created using open source NHttp which is a simple asynchronous HTTP server written in C# for the .NET framework. The REST web service is composed of a REST web service server and a client. In the REST architecture, clients send HTTP (Hypertext Transfer Protocol) request messages to the server to retrieve or modify resources, and servers send responses to these requests. A request consists of HTTP methods, four of which are commonly used in REST based architecture: “POST”, “GET”, “PUT”, and “DELETE” that define the kind of operation to perform. In this communication mechanism, two clients and two servers are created.

Aras Innovator is used as a PLM system to store data relating to the product requirement. It provides a REST client to send the requirement data to the computational platform. A Linux laptop as a computational platform holds a REST server to call the “POST” method in order to get the requirement data sending from the Aras. In the Linux laptop, the REST web service server and client are created for the python environment. The computational platform executes the requirement data and sends the results to the NHTTP-Rest server through the REST client. NHttp server uses Aras.IOM library which is C# API for Aras to access the PLM database and update the requirement data. Aras.IOM library has classes that allow connecting to a specified Innovator

server and interacts with it, sending to it AML (Adaptive Markup Language) requests and getting back AML responses. By using this Aras's C# API, methods like "create", "read", "update", and "delete" are created to allow the REST requests to access Aras's data repository.

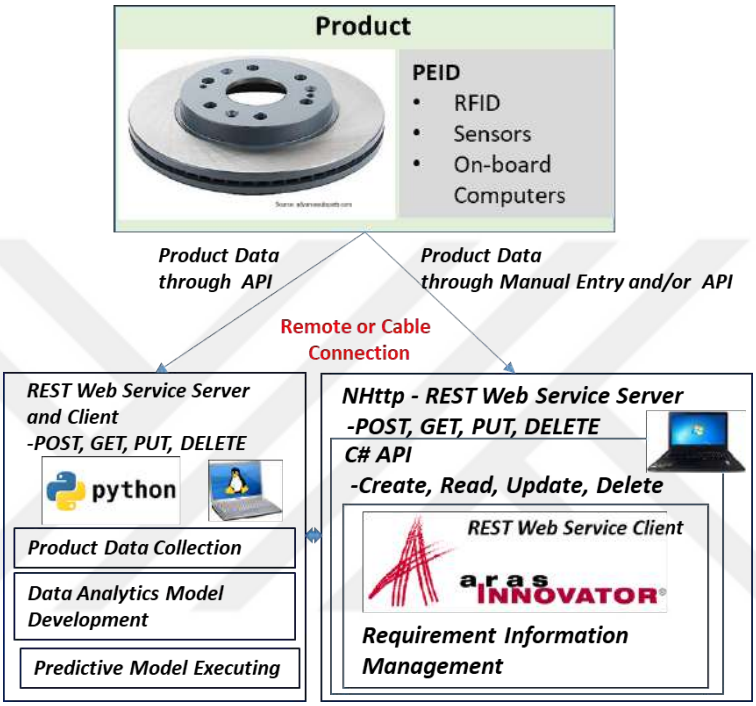


Figure 6.8: Implementation of the Communication System

The implementation architecture is presented in Figure 6.8 above. Requirement and requirement-related data can be created through manual entry by users and/or through an API. In certain applications, if the user does not need to store data in the PLM system, the computational platform can collect the data from a product through an API and process the data by using analytical models, and send the results to the PLM system as requirement information through the REST web service. Product users, engineers, or data analysts can access, update and create product requirement data stored in Aras Innovator using four methods "POST", "GET", "PUT", and "DELETE" from the computational platform.

## 6.3 Requirement Data in RDF Format

The communication mechanism is also necessary to convert the requirement data in relational format into RDF format, complying with the W3C standard. In this part, it is illustrated a very simple way of converting requirement data created in Aras Innovator into RDF Turtle format that includes subject URIs, along with predicates and objects that describe a resource's properties and their values. There are many different methods and tools available for converting relational data into RDF graph data format. OpenRefine, formerly called Google Refine, is used to create RDF form requirement data from a relational database. OpenRefine [131] is an open-source desktop application for data cleanup and transformation to other formats. OpenRefine extends data with web services and links it to external databases. It has an RDF extension written by DERI that allows users to convert flat spreadsheet data to RDF form. An illustrative example of converting requirement data from relational to RDF turtle format using OpenRefine is shown in figure 6.9.

The RDF Extension for OpenRefine provides a graphical user interface. It allows users to export relational data, set up ontology structure mapping between a column's value (object) and a predicate, select RDF format, and export RDF data. The RDF Extension for OpenRefine also uses URIs so that each resource can be linked with other resources both within this dataset and across the World Wide Web. Since OpenRefine doesn't support importing from the Aras RDB database directly, CVS data is created by using AML studio. As shown in figure 6.9, the CVS file that includes function model data is created and imported to OpenRefine. RDF Skeleton window is selected o set up the ontology by clicking the add property link and then clicking on the property itself. After the semantic data is produced, simply select one of the "RDF as" options from the Export menu to export the RDF data. An autonomous method that converts PLM system SQL data

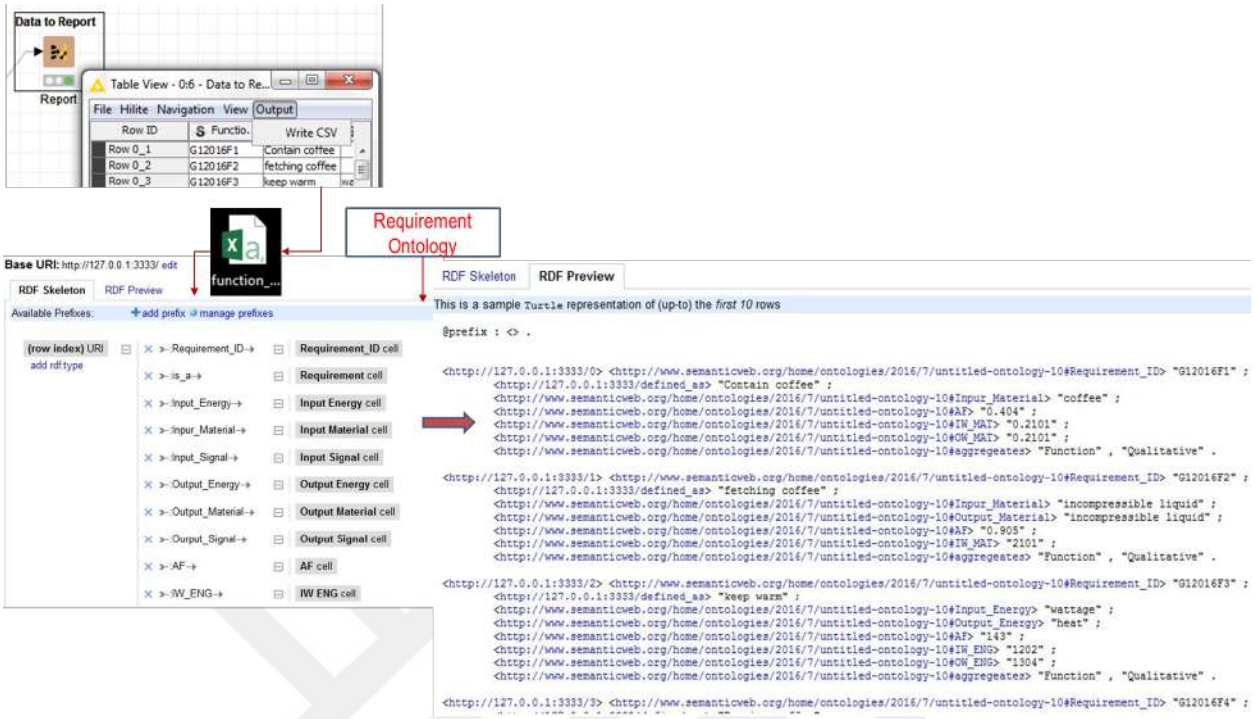


Figure 6.9: Requirement Data Generation in RDF Format

into RDF format is under study for future applications.

## 6.4 Summary

In this chapter, the role of the PLM system and computational platform in REKMA is discussed. First, how the PLM system represents the product requirement and requirement related information is presented. Then, data structure of the ontology-based requirement information model is implemented in the Aras PLM. Then, the development and use of the computational platform which is integrated into the PLM system is introduced. Finally, the communication mechanism among a product, the PLM system, and computational platform to transport data and file and to convert relational data from the PLM system into the Resource Description Framework (RDF) is implemented.

In the PLM system, two user interfaces are created for the requirement representation; (i) which represents user inputs mainly the requirement sentence, and (ii) which represents the outcomes of the computational applications in the computational platform for the requirement representation, requirement entity inference, and requirement text generation. The following chapter discusses how to convert a requirement sentence into the proposed representation format, infer requirement information, and generate requirement texts.

# CHAPTER 7

# REQUIREMENT INFORMATION EXTRACTION AND TEXT GENERATION IN THE REKMA

This chapter mainly discusses the development of the computational applications for the requirement information extraction and retrieval to convert requirement sentence into the proposed representation format, requirement information inference, and requirement text generation. These applications aid designers/engineers with representation, generation, execution, and verification of product requirements and requirement-related information flow across the product life-cycle. First, this chapter introduces how to retrieve product requirements and requirement-related data and generate reusable knowledge for the requirement specification. Next, it discusses the development of the requirement NER which is the main model for the requirement information extraction and retrieval. Finally, this chapter presents the syntax analysis that describes the syntactic structure of requirement text and identifies controlled requirement syntaxes to automatically generate textual requirements.

## 7.1 Product Requirement Information Extraction and Retrieval

In current PLM practice, product requirements, and related data are described most commonly with textual description, textual document, string, and numerical data type. If the requirement and related data are stored in the PLM system as a string and numerical type, not as a textual format, it can be transferred to the computational platform and processed for information extraction and

retrieval without applying any NLP pre-processing. For example, a function verb (create), product name (rotor), or a value (120), can be directly transferred to the computational platform and used in an analytical application. An application for the requirement information generation is developed to aid designers/engineers during the functional requirements generation of a product. Whenever the user defines the functional requirement, which includes the product name and its main function in the PLM environment, the application in the computational platform is able to generate functional requirements for sub-functions of the product. The development of the application requires building a functional architecture library in the computational platform to execute the requirement generation of products. The product data, stored in Aras PLM can be accessed through AML (Adaptive Markup Language) from the computational platform and analyzed with AI techniques to establish a functional requirements architecture as shown in figure 7.1.

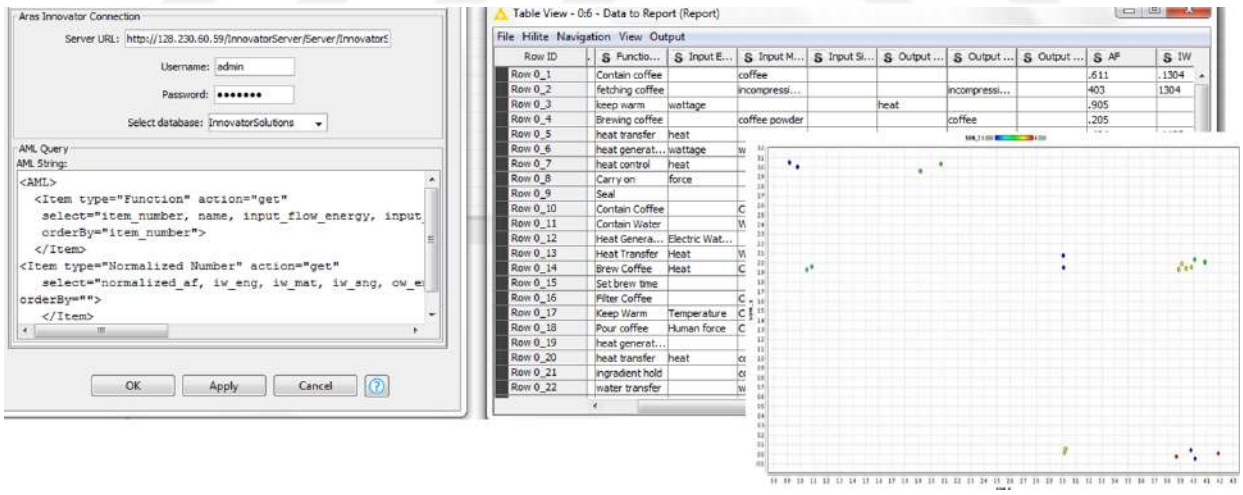


Figure 7.1: Data Exchange and Initial clustering of Product Functions for a Coffee Maker in Computational Platform

Functional architecture for any product can be created by processing the product's functional information through data analysis methods. In this study, an application is developed to establish

a functional requirement architecture for coffee makers, building upon function-based product architecture (FPA) work done by Ong et al. [114]. The functional architecture of the coffee maker is created in the computational platform using neural networks by processing existing product data in the PLM system. The goal of this functional knowledge extraction is to reorganize and formalize functional requirement knowledge to support requirement information generation. In the appendix C.1, how functional information is collected, functional architecture is built, and functional requirements are generated for a coffee maker are discussed.

As seen in figure 7.1, the functional information of the coffee maker is retrieved from Aras PLM for an analytical application without the need for NLP pre-processes. However, if the requirement data entered to PLM as a textual description such as 'Rotor shall dissipate heat' which is the most common preference, then using an application developed by natural language processing and text mining techniques becomes crucial to represent product requirements in the proposed format and infer requirement information. The application must extract terms from the requirement sentence and classify these terms based on PDROM classification. Below, product requirement information extraction and retrieval from a requirement sentence and requirement sentence analysis processes are discussed.

The activities of the requirement sentence analysis process to convert a requirement sentence into the proposed format to infer requirement information, and generate requirement text are presented in figure 3.3. This series of activities first converts a requirement sentence into the proposed representation format as shown in table 3.3. Then it generates inference entities as shown in table 3.5. Finally, it creates textual requirements by using inference entities and formalized syntaxes as seen in table 3.6. Automatically extracting requirement semantics from the textual description requires mainly two types of study: (i) syntax analysis that describes the syntactic

structure of requirement text and identifies entity types of requirement, and (ii) semantic analysis that reveals meanings of the text based on the PDROM using NLP techniques. Requirement information extraction steps and NLP techniques that are tokenizing, Part of Speech (POS) tagging, lemmatizing, NER, etc. are illustrated in figure 3.4. Semantic and syntactic analysis and the development of these activities are discussed below.

### 7.1.1 Syntax Analysis of the Product Requirements

Product requirement sentences are studied from a lexical viewpoint to identify common Part of Speech (POS) tags and entity types defined within the PDROM in a requirement sentence and to define how to generate a requirement sentence by using inference entities. Textual requirements are organized in the syntactic structures of the formalized requirement syntaxes for functional and non-functional requirements that are shown in table 3.6. Then, these requirement statements are broken up into their entities such as part, function, flow, unit, constraints, failure, etc. Below some examples are given to illustrate subject, predicate, and object categories of product requirements, their part of speech tags, and entity types.

Table 7.1: Syntax Analysis of the General Requirement Form

| **Triple:** | ***Subject*** | ***Predicate*** | | ***Object*** |
|---|---|---|---|---|
| **Requirement Text:** | Automobile brake system | shall | generate | friction. |
| **POS:** | <noun> | <modal verb> | <verb> | <noun> |
| **Entity Type:** | PRODUCT | | FUNCTION | FLOW |

Table 7.1 illustrates the syntax analysis of the general requirement form that uses a transitive verb and a direct object. This is the most basic functional requirement syntax. Additional information can be added to the object of the sentence to modify the subject and verb phrase if needed.

Table 7.2: Syntax Analysis of a Requirement with Verb Phrase Modification

| **Triple:** | ***Subject*** | | ***Predicate*** | | ***Object*** | |
|---|---|---|---|---|---|---|
| **Requirement Text:** | Automobile | brake | shall | reduce | speed | safely. |
| **POS:** | <noun> | <noun> | <modal verb> | <verb> | <noun> | <adverb> |
| **Entity Type:** | PRODUCT | | | FUNCTION | FLOW | CONSTRAINT |

Table 7.2 illustrates a constraint entity that modify the verb phrase (function verb) in the requirement sentence that presented in table 7.1. As shown in this example, the functional requirement consists of product ‘automobile brake’ function verb ‘reduce’ and flow ‘speed’ entities. Even though these entities are enough to generate a consistent functional requirement, the completeness of the requirement is increased by adding a constraint entity ‘safely’ which modifies the function verb entity ‘reduce’.

Lexical study of the product requirements shows that some words in a requirement sentence might be tagged with many entity types that are called overlapping entities. It means that one entity type can consist of many entity types. The example in table 7.3 demonstrates overlapping entities in a functional requirement. The object of the sentence uses adjunct ‘up to 600 degree Celsius’ that adds additional information about the function verb entity ‘resist’ and property entity ‘temperature’ which form the function name entity ‘ resist temperature’. In this example, while ‘up to’ is a constraint entity, ‘600’ is a value entity, and ‘degree Celsius’ is a unit entity, as a whole ‘up to 600 degree Celsius’ is a requirement measurement entity. As discussed in chapter 4, requirement measurement might be qualitative if the value entity is not numeric and might be quantitative if the value entity is numeric. The measurement entity might consist of a value entity, value and unit entities, constraint, value, and unit entities. Similarly, the function name entity is described in the PDROM as a combination of the function and flow entities. For example, ’reduce’

is a function and 'speed' is a flow entity. Both entities construct the function name entity 'reduce speed'. Overlapping entities are discussed in the following section.

Table 7.3: Syntax Analysis of a Requirement with Overlapping Entities

| **Triple:** | ***Subject*** | ***Predicate*** | | ***Object*** | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Requirement Text:** | Automobile brake system | shall | resist | temperature | up | to | 600 | degree | Celsius. |
| **POS:** | <noun> | <modal verb> | <verb> | <noun> | <adposition> | <adposition> | <number> | <noun> | <noun> |
| **Entity Type:** | PRODUCT | FUNCTION | | PROPERTY | CONSTRAINT | | VALUE | UNIT | |
| | | | | | MEASUREMENT | | | | |

Non-functional requirements are also lexically studied. The example in table 7.4 demonstrates syntax analysis for a non-functional requirement. In a non-functional requirement, the difference is the use of an auxiliary verb instead of an action verb. In this example, the verb phrase 'have' tagged with a non-function entity. The object of the sentence 'high surface hardness' complements the subject 'rotor' by providing information about the subject which is a part entity in a non-functional requirement. The object of the sentence includes two types of entities; 'high' is a value entity and 'surface hardness' is a property entity.

Table 7.4: Syntax Analysis of a Non-Functional Requirement

| **Triple:** | ***Subject*** | ***Predicate*** | | ***Object*** | | |
|---|---|---|---|---|---|---|
| **Requirement Text:** | Rotor | shall | have | high | surface | hardness. |
| **POS:** | <noun> | <modal verb> | <auxiliary verb> | <adjective> | <noun> | <noun> |
| **Entity Type:** | PRODUCT | NON-FUNCTION | | VALUE | PROPERTY | |

So far, requirements are demonstrated with the subject that includes only one entity. Sometimes the subject might include more than one entity type as shown in table 7.5. In this non-functional requirement, the subject includes two entity types; 'CO2 emission' which is an environmental hazard and tagged as a hazard entity, and 'part production' which is a stage in product life-cycle, is tagged as stage entity.

Table 7.5: Syntax Analysis of a Requirement with Two Entities in Subject Part

| **Triple:** | ***Subject*** | | | | | ***Predicate*** | | ***Object*** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Requirement Text:** | CO2 | emission | in | part | production | shall | be | less | than | 120 | g. |
| **POS:** | <noun> | <noun> | <adposition> | <noun> | <noun> | <modal verb> | <auxiliary verb> | <adjective> | <conjunction> | <number> | <noun> |
| **Entity Type:** | HAZARD | | STAGE | | | NON-FUNCTION | | CONSTRAINT | | VALUE | UNIT |
| | | | | | | | | MEASUREMENT | | | |

When the verb phrase is an auxiliary verb in a requirement sentence, the object of the requirement sentence requires a subject complement which is necessary for the completeness of the requirement. The example in table 7.6 demonstrates the use of a complement and its entity in a non-functional requirement. The material property entity 'recyclable' is a subject complement that provides necessary information about the subject 'rotor's material'. The material property entity defines the characteristic of the material entity located in the subject.

Table 7.6: Syntax Analysis of a Non-Functional Requirement with Subject Complement

| **Triple:** | ***Subject*** | | ***Predicate*** | | ***Object*** |
|---|---|---|---|---|---|
| **Requirement Text:** | Rotor's | material | shall | be | recyclable. |
| **POS:** | <noun> | <noun> | <modal verb> | <auxiliary verb> | <noun> |
| **Entity Type:** | PRODUCT | MATERIAL | NON-FUNCTION | | MATERIA_PROPERTY |

Some product requirement sentences are studied and discussed from the lexical viewpoint to identify common Part-of-Speech tags and requirement entity types defined within the PDROM. The above examples show the separation of the linguistic structure of product requirements, part of speech tags of these separated categories, and entity types that are defined for the phrases of these categories. In a requirement sentence, the subject is presented first, then the predicate is defined, and lastly, the object is defined till to end of the requirement sentence. The subject mostly includes requirement actor classes as an entity type. Function and non-function entities are located in the predicate. Since the object can include direct object, indirect object, and additional information,

it can consist of many entity types defined in the PDROM. Lexical study of the requirement text examples shows that mostly flow, property, failure, constraint, material property, stage, and measurement entities are located in the object. Twenty requirement entities which are the classes of the PDROM are defined as the most common requirement related information that contracts the requirement text. These entity types and how to extract them from the requirement sentences are discussed in the next section.

### 7.1.2 Requirement Semantics Extraction

In this dissertation, to extract the semantic meaning of the requirement text, firstly the PDROM is developed and syntactic structures of requirement texts are studied. Automatic entity recognition of a requirement text and requirement information extraction require studying natural language processing, ontology, and taxonomies. UML representation of product requirement and taxonomies are discussed in chapter 3. In order to recognize and identify UML model structure, class hierarchy, relationships, and attributes in a requirement text, the first and most important task towards extracting requirement information is to classify requirement named entities into pre-defined categories as discussed before. The NLP technique, Named-entity recognition (NER) which is a sub-task of information extraction is used for this task.

#### 7.1.2.1 Development of the Requirement Named Entity Recognition Model

An NLP system that processes requirement text to identify types of requirement named entities (product, material, function, flow, constraint, etc.) are developed using spaCy which is a free and open-source Python library written in Python and Cython for advanced Natural Language Processing (NLP) [132]. SpaCy uses residual convolutional neural networks (CNN) and incremental

parsing with Bloom embedding for NER. The spaCy library is compatible with 64-bit CPython 2.7 / 3.5+ which we run on Linux. NER, which is the core component of our NLP system, is one of the NLP tasks, along with tokenizing, lemmatizing, part of speech (POS) tagging, phrase chunking, etc. The NER system is not restricted to taxonomy-based vocabularies. It depends on statistical models such as machine learning and linguistic grammar-based techniques. A requirement Named Entity Recognition model using spaCy's Named Entity Recognition library is created in order to capture key actors (entities) in a requirement text. SpaCy offers various statistical neural network models trained for different languages such as; English, German, Spanish, Portuguese, French, Italian, Dutch, and multi-language NER. Once spaCy is installed, the pre-trained model for the English language is downloaded. The NER model can help to answer the following questions about the product requirements:

- What is the actor mentioned in the requirement text?
- What is the type of requirement?
- If it's a functional requirement, what is the function verb and flow that are the main elements of the function model?
- Which failure information mentioned in requirement text?
- What is the measurement type of requirement?
- Does requirement indicate a specific stage of product life cycle?

SpaCy's NER is trained for eighteen named entities on the OntoNotes 5 corpus. However, this model does not support the requirement named entities that are described in the PDROM.

SpaCy pre-trained NER model is not trained to identify any function, flow, property, failure, material, etc. entities presented in the requirement text as shown in figure 7.2. Clearly, the spaCy pre-trained NER model does not fit the requirement domain, so a requirement NER model

```
Text ['Apple is looking at buying U.K. startup for $1 billion']
Entities [('Apple', 'ORG'), ('U.K.', 'GPE'), ('$1 billion', 'MONEY')]
Tokens [('Apple', 'ORG', 3), ('is', '', 2), ('looking', '', 2), ('at', '', 2), ('buying', '', 2), ('U.K.', 'GPE', 3), ('startup', '', 2),
('for', '', 2), ('$', 'MONEY', 3), ('1', 'MONEY', 1), ('billion', 'MONEY', 1)]

Text ['The drone shall avoid obstacles during flight']
Entities []
Tokens [('The', '', 2), ('drone', '', 2), ('shall', '', 2), ('avoid', '', 2), ('obstacles', '', 2), ('during', '', 2), ('flight', '', 2)]

Text ['An electrical motor shall converts electricity to rotational energy']
Entities []
Tokens [('An', '', 2), ('electrical', '', 2), ('motor', '', 2), ('shall', '', 2), ('converts', '', 2), ('electricity', '', 2), ('to', '', 2),
('rotational', '', 2), ('energy', '', 2)]
```

Figure 7.2: SpaCy NER Results of Sample Requirement Texts

is trained. This new model is trained on the data set which includes product requirement and requirement related sentences. The model supports the requirement named entity types, which are defined as requirement classes in PDROM. These entity types and their descriptions are listed in table 7.7.

Table 7.7: Requirement Named Entity Types and Descriptions

| Requirement Entity Type | Description |
|---|---|
| PRODUCT | Object, part, component, system, etc. |
| MATERIAL | Engineering material classes and their members |
| MATERIAL_PROPERTY | Mechanical, physical, chemical, manufacturing and environmental property of materials |
| PROCESS | Product life cycle activities like material production, manufacturing, maintenance, etc. |
| SHAPE | Geometric terms and relationships like form types, features, topology, etc. |
| ENVIRONMENT | Environmental information for Life cycle activities like process, product use, etc. |
| FUNCTION | Function verb (Action Verb) |
| FUNCTION_NAME | Objective of system |
| NON_FUNCTION | Non-Function verb (Linking Verb) |
| FLOW | Inputs and outputs for system |
| PROPERTY | Attributes of flow |
| FAILURE | Situations that a system does not meet the objective particularly or completely |
| CONSTRAINT | Limitation and restriction on subject or object |
| OBJECTIVE | Things to achieve with/for system |
| HAZARD | Environmental, physical and health hazards |
| STAGE | Product life cycle stages |
| USER | People who involves its life cycle activities |
| VALUE | Quantitative and qualitative data |
| UNIT | Units of Measurement |
| MEASUREMENT | Size, weight, capacity, amount, or other aspect of something |

Among entity types, FUNCTION_NAME and MEASUREMENT are not able to be trained and located in the text through the requirement NER because these entities overlap with other entities.

For example, FUNCTION_NAME overlaps with FUNCTION and FLOW entities. 'reduce' is a FUNCTION entity, 'speed' is FLOW entity, and 'reduce speed' is FUNCTION_NAME entity. The training model does not train overlapping entities therefore structural rules are created to identify these entity types. These rules are represented in table 7.8.

Table 7.8: Overlapping Entities and Rules to Define Entity Type

| **Entity Type** | **Rule** | **Example** |
|---|---|---|
| FUNCTION NAME | FUNCTION + CONSTRAINT + FLOW | 'take quality image' |
| FUNCTION NAME | FUNCTION + FLOW | 'reduce speed' |
| FUNCTION NAME | FUNCTION + PROPERTY | 'resist temperature' |
| FUNCTION NAME | FUNCTION + FAILURE | 'resist abrasive wear' |
| MEASUREMENT | VALUE | 'hot' |
| MEASUREMENT | VALUE + UNIT | '120 g' |
| MEASUREMENT | CONSTRAINT + VALUE + UNIT | 'up to 600 degree Celsius ' |

The first task to create the requirement NER is to annotate training data manually to train the model since NER is a statistical supervised learning system. Entities that are described above in table 7.7 are labeled in requirement text data using rasa-nlu-trainer [133] to create training data. This trainer creates data in JSON format which is shown in figure 7.3 to train Rasa NLU [134] models. However, SpaCy requires training data to be in the format of TRAIN_DATA = [(Sentence, {entities: [(start, end, label)]}, ...] as shown in figure 7.4 with the same requirement text represented in figure 7.3.

Since the spaCy NER model is trained to create requirement NER, the script shown in Figure 7.5 is created to convert the data format of NLU training data into spaCy's training format.

As indicated before, first it is wanted to incorporate requirement NER with spaCy NER because spaCy NER already supports the PRODUCT, PERSON, etc. entities that are part of the PDROM entities. However as shown in figure 7.2, spaCy trained NER even fails to find product entity in the

```
"text": "An automobile brake system shall reduce speed safely.",
"intent": "",
"entities": [
  {
    "start": 3,
    "end": 26,
    "value": "automobile brake system",
    "entity": "PRODUCT"
  },
  {
    "start": 33,
    "end": 39,
    "value": "reduce",
    "entity": "FUNCTION"
  },
  {
    "start": 40,
    "end": 45,
    "value": "speed",
    "entity": "FLOW"
  },
  {
    "start": 46,
    "end": 52,
    "value": "safely",
    "entity": "CONSRAINT"
  }
]
```


Figure 7.3: Sample of the Training Data in Json Format


```
[["An automobile brake system shall reduce speed safely.",
{"entities": [[3, 26, "PRODUCT"], [33, 39, "FUNCTION"], [40, 45, "FLOW"], [46, 52, "CONSTRAINT"]]}]]
```


Figure 7.4: Sample of the Training Data in spaCy Format

requirement domain. Therefore, a blank NER model is created. SpaCy pre-trained NER and how to train a blank model can be found on SpaCy's website at [135]. After training the requirement NER model using python's spaCy module, it is tested on test data that includes requirement text written in different formats. The model is trained with 650 and it is tested with 45 requirement text data. In order to make the model's predictions more accurate, more training data can be added to the training set.

```python
import json

def convert_data(path):

    with open(path, 'r', encoding='utf-8-sig') as f:
        content = json.load(f)

    common_examples = content['rasa_nlu_data']['common_examples']

    converted = []

    for ex in common_examples:

        entities = []
        for entity in ex['entities']:
            entities.append((entity['start'], entity['end'], entity['entity']))

        entities_dict = {'entities': entities}
        entry = (ex['text'], entities_dict)

        converted.append(entry)

    return converted
```

Figure 7.5: Convert Json File to spaCy's Trainig Format

#### 7.1.2.2 Results and Evaluation of the Requirement NER Model

The requirement NER model is tested using 45 requirement texts. The NER model defines requirement entities of a requirement text and these entities are represented in the proposed structured format that is discussed in chapter 3.

Table 7.9: Requirement NER Output of a Product Requirement Sentence

| |
|---|
| **Requirement Text:** Rotor shall resist abrasive wear. |
| **Prediction:** |
| Entity: PRODUCT, Value: Rotor |
| Entity: FUNCTION, Value: resist |
| Entity: FAILURE, Value: abrasive wear |
| Entity: FUNCTION_NAME, Value: resist abrasive wear |
| **Ground truth:** |
| Entity: PRODUCT, Value: Rotor |
| Entity: FUNCTION, Value: resist |
| Entity: FAILURE, Value: abrasive wear |
| Entity: FUNCTION_NAME, Value: resist abrasive wear |

Table 7.10: Accuracy Score of Requirement NER

| Recognized Requirement Entity | Precision | Recall |
|---|---|---|
| PRODUCT | 0.9623 | 0.9808 |
| MATERIAL | 1.0000 | 0.9167 |
| MATERIAL_PROPERTY | 1.0000 | 1.0000 |
| PROCESS | 1.0000 | 1.0000 |
| SHAPE | 1.0000 | 1.0000 |
| ENVIRONMENT | 1.0000 | 1.0000 |
| FUNCTION | 0.9688 | 0.9688 |
| FUNCTION_NAME | 1.0000 | 0.9524 |
| NON_FUNCTION | 1.0000 | 1.0000 |
| FLOW | 1.0000 | 0.9474 |
| PROPERTY | 1.0000 | 0.6667 |
| FAILURE | 0.8571 | 1.0000 |
| CONSTRAINT | 0.8750 | 0.9130 |
| OBJECTIVE | 1.0000 | 1.0000 |
| HAZARD | 1.0000 | 1.0000 |
| STAGE: | 1.0000 | 0.8571 |
| USER | 1.0000 | 1.0000 |
| VALUE | 1.0000 | 1.0000 |
| UNIT | 1.0000 | 1.0000 |
| MEASUREMENT | 1.0000 | 1.0000 |
| **OVERALL** | 0.9722 | 0.9646 |

As shown in table 7.9, the requirement NER tags the requirement entities; 'rotor' as a Part entity, 'resist' as a Function entity, 'abrasive wear' as a Failure entity, and 'resist abrasive wear' as Function_Name entity in the requirement text 'Rotor shall resist abrasive wear'. It identifies all the entity types for the values correctly. It is impressive how the requirement NER model is capable of finding requirement entities. The precision and recall are calculated for each entity type that the model recognizes to measure the capability of the requirement NER. Table 7.10 represents the values of these metrics for each entity and an overall score for requirement NER to evaluate the model on the test data consisting of 45 requirement texts. These requirement texts and 5 out of 45

test results, which are used for performance analysis by comparing prediction entities and ground truth entities are presented in the appendix D.

### 7.1.3 Determination for Super-entity and Sub-entity Hierarchy

Entities that are trained for requirement NER are not all entities defined in PDROM ontology. When requirement entities are tagged, the words are classified based on their superclass under one entity type. For example, yield strength is a strength property, defined under mechanical property and thermal conductivity is a thermal property, defined under mechanical property according to the material property taxonomy. Requirement NER recognizes and tags for these properties as material property not as specific material property. In order to define and represent the super-entity and sub-entity hierarchy of requirement entities in the formal requirement specification, tagged words are searched in the taxonomy of the entity type. For example, compaction is an activity of the powder metallurgy process. Both compaction and powder metallurgy processes are tagged as a process entity in a requirement sentence. However, compaction is further searched in the process taxonomy and it is defined as the sub-class of the powder metallurgy. If the tagged word does not exist in the taxonomy, its semantic similarity with the words in the taxonomy is determined by running the similarity method on the spyCy model. The highest similarity value determines the super-class and sub-class hierarchy of the entity.

This method compares the word vectors generated using an algorithm like word2vec and supports two methods to find word similarity: i) small model context-sensitive tensors, and (ii) word vectors. World similarity is the value ranges from 0 to 1 which tells us how close two words are, semantically. 1 means both words are the same and 0 shows no similarity between both words. For example, Figure 7.6 shows the semantic similarity analysis of Flow entity ‘coffee

beans' to define a subclass of flow such as material, energy, and signal to identify the super-entity and sub-entity hierarchy. In this example, 'coffee beans' is first searched in the flow taxonomy to determine the entity hierarchy. Because 'coffee beans' is not found under any type of flow: signal, energy, and material in the taxonomy, its semantic similarity is checked with each of the flow instances, defined under the flow entity. This example shows that 'coffee beans' has the highest semantic similarity with 'water', which is a material flow. Therefore 'coffee beans' is also tagged as material flow.

```
similarities to coffee beans
signalflow::status: 9.67%
signalflow::auditory: 2.98%
signalflow::word: 23.71%
signalflow::pressure: 20.85%
energyflow::human force: 17.39%
energyflow::acoustic pressure: 19.35%
energyflow::mass flow: 23.89%
energyflow::radioactive intensity: 20.10%
energyflow::temperature: 29.26%
materialflow::gas: 26.93%
materialflow::liquid: 36.04%
materialflow::solid: 26.13%
materialflow::object: 18.72%
materialflow::particulate: 6.20%
materialflow::composite: 6.59%
materialflow::plasma: 16.46%
materialflow::mixture: 39.75%
materialflow::water: 42.55%
```

Figure 7.6: Semantic Similarity of 'coffee beans'

So far, how a requirement sentence is converted into the proposed representation format is discussed. As shown in table 7.11, first, requirement text is demonstrated as a triple format; Subject, Object, and Predicate. Then, entities of the requirement text are identified by the requirement NER. Finally, super-class and sub-class hierarchy is defined by searching the entity word or checking semantic similarity of the entity word in a taxonomy such as product, shape, function, flow, material, material property, manufacturing process, etc where each taxonomy is constructed hierarchically.

Table 7.11: Requirement NER Output of a Product Requirement Sentence

| |
|---|
| **Requirement Text::** Rotor shall resist abrasive wear. |
| **Triple::** shall resist (rotor, abrasive wear) |
| **Subject Entities;** |
| Part:'rotor' |
| Part::Automotive Part::Brake System |
| **Object Entities;** |
| Failure:'abrasive wear' |
| Failure::Failure Mode |
| **Predicate Entities;** |
| Function:'resist' |
| Function::Function Verb::Control |

Applications that are developed for the computational platform can also infer the requirement entities by using pre-defined semantic rules which are discussed before. Requirement entity inference is introduced in the next section.

### 7.1.4 Entity Inference

In this study, "inference" is used as a meaning of discovering new requirement entities based on the created entities and some additional information, which comes from expert knowledge in the form of a vocabulary. Expert knowledge is represented as a set of rules. These rules are created by using SWRL for the PDROM OWL implementation in Protégé and discussed in chapter 4. For the requirement knowledge management, a requirement entity inference engine is created by using a set of entity inference rules. The goal of the development of the inference rules is to check whether a requirement entity can be associated with any input entity types and values. The computational platform stores IF-THEN rules, provided by experts. These rules work on the requirement entities, extracted from the given requirement sentence. They contain the input and output entities, which are modeled as a linguistic variable. A couple of inference rule examples are listed in table 7.12.

Table 7.12: Examples of the Requirement Information Inference Rules

| |
|---|
| **Entity:** *FUNCTION*<br>→ Requirement_Type: Functional_Requirement<br><br>**Entity:** *NON_FUNCTION*<br>→ **Requirement_Type:** *Non_Functional_Requirement*<br><br>**Entity:** *FAILURE* and **Value:**, *abrasive wear*<br>→ **Entity:** *MATERIAL_PROPERTY* and **Value:** *wear resistance*<br><br>**Entity:** *MATERIAL_PROPERTY*<br>→ Actor: Material and Stage: Material_Selection<br><br>**Entity:** *OBJECTIVE*, **Value:** *minimize* **Entity:** *PROPERTY* and **Value:** *weight*<br>→ **Entity:** *MATERIAL_PROPERTY* and **Value:** *density* |

The first rule indicates that if there is a 'Function' entity in a requirement text, the type of requirement is a functional requirement. If the entity type is 'Non-Function', then the requirement type is a non-functional requirement. In addition, entity type, 'Failure' and entity value, 'abrasive wear' are coded to infer entity type, 'Material Property' and its value, 'wear resistance'. Any material property entity requires to define 'Requirement Actor' as 'Material' and 'Stage' as 'Material Selection'. The last rule shows that the objective to minimize the weight of a product is mainly about the density property of the product material. Whenever a requirement sentence is created in the PLM system, firstly, requirement NER tags the words in the requirement sentence and defines requirement entities. Then, these entities are used as inputs of the inference rules. Finally, return values of these rules are explicitly added to the PLM system as inference entities as shown in figure 7.7 and they are represented in a format as **Entity Type::Sub-Entity Type:'Entity Name'**.

Figure 7.7 illustrates a requirement sentence, proposed representation of this sentence, and inference entities. These are generated by the applications in the computational platform and

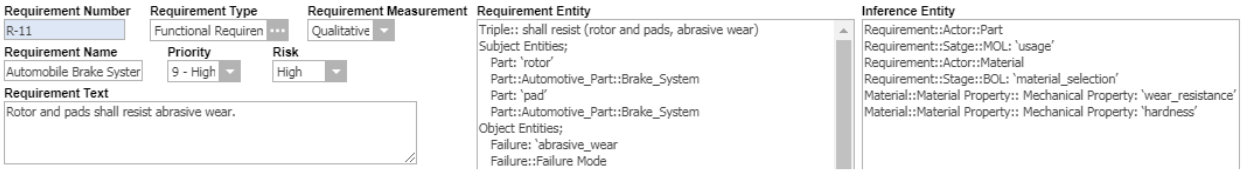


Figure 7.7: Inference Entities in PLM System

added to the PLM system. Inference entities are the return values of inference rules, which have failure entity input, ‘abrasive wear’, part entity input, ‘rotor’, and ‘pad’ and function entity input, ‘resist’. These inputs are first generated by NLP application from the requirement text, and then they are used to infer requirement entities by the inference engine.

## 7.2 Requirement Text Generation

As discussed in chapter 2, model based languages like SysML allow users to generate requirement text manually through the captured information in diagrams. When a requirement text is created, the system must automatically capture the rationale by decomposing high level need statements into necessary requirement entities and also inference related entities and generate requirement text using inference information. Such a system allows designers/engineers to quickly construct requirement specification documents, to ensure completeness of requirement sets, trace the impact of requirement changes and overall reduces requirement management costs.

Formalized requirement syntaxes for both functional and non-functional product requirements are represented in table 3.6 in chapter 3 to generate requirement text systematically starting with the subject components, followed by modal and main verbs followed by objects components. The subject component is mostly specified by requirement actor: product, material, shape, environment, and process. The modal verb ‘shall’, function verb for the functional requirement, and linking verb

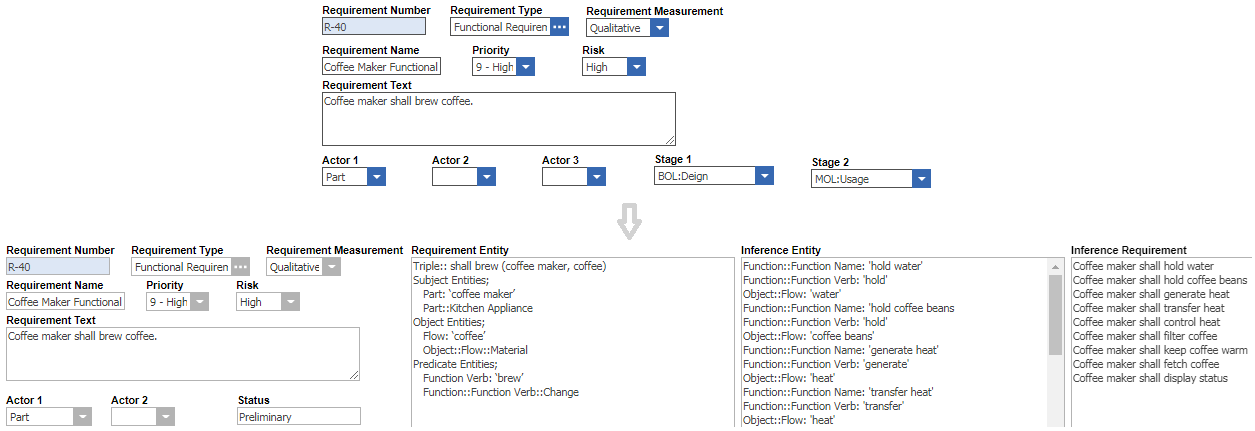


Figure 7.8: Function Requirement Generation and Representation in the PLM System

for non-functional requirement follows the subject. For a functional requirement, the verb phrase (function verb) is followed by a flow entity and additional information (adjunct) if it's available. For non-functional requirements, the verb phrase such as 'be', 'have', etc. is followed by a subject complement that might include various requirement entities.

The functional requirements for the coffee maker as shown in figure 7.8 are generated by the computational platform using inferred function entities and inference rules. When users enter the functional requirement text ' Coffee maker shall brew coffee' for a coffee maker, which includes the product name entity 'coffee maker' and its main function entity 'brew' into the PLM system, the computational platform first identifies requirement text entities by using the requirement NER model. Then it infers the function name, function verb, and flow entities by using the rule-based system. In this example, the reusable function information for coffee makers which is identified in the functional architecture is embedded into rule-based systems in the computational platform in order to infer the sub-functions of a coffee maker. Finally, it defines the functional requirements in a textual format by using pre-defined requirement syntaxes and corresponding requirement generation rules.

Figure 7.8 illustrates two screenshots from the PLM system. The upper part of the figure represents the requirement text input, 'Coffee maker shall brew coffee' in the PLM system. The bottom part of the figure represents the computational platform outputs: requirement entities, which are generated by proposed NLP; inference entities; and requirement texts, which are generated by the rule-based system.

Table 7.13: Requirement Text Generation for Coffee Maker and Rules

| | |
|---|---|
| **Inference Rule**:<br>*If { Product= "coffee maker", Function Name= "brew coffee"} THEN {Function = "hold", Flow = "water" and Function Name = "hold water"}*<br><br>**Requirement Syntax**:<br><subject><modal><transitive verb><direct object>{<adjunct>}<br><br>**Tex Generation Rule**:<br>*If { Part = "X", Function = "Y", Flow = "Z"} THEN*<br>*{Subject = "X"}*<br>*{Modal = "shall"}*<br>*{Transitive verb = "Y"}*<br>*{Direct object = "Z"}* | **Requirement Text:** *Coffee maker shall hold water*<br>**Triple::** *shall hold* (*coffee maker*, *water*)<br>**Subject Entities;**<br>Part:'*coffee maker*'<br>Part::Kitchen Appliances<br>**Object Entities;**<br>Flow:'*water*'<br>Flow::Material<br>**Predicate Entities;**<br>Function::'*hold*'<br>Function::Function Verb |

Table 7.13 represents a coffee maker requirement, which is generated by the computational platform based on the information extracted from the requirement text 'Coffee maker shall brew coffee' and corresponding rules. Knowledge base stores IF-THEN rules provided by experts for requirement information inference and requirement text generation. The computational platform first infers the requirement information and then generates requirement text by using IF-THEN rules. Return values of inference rules represent requirement entities that are used to generate product requirement text. Return values of requirement generation rules are created based on the

pre-defined requirement syntaxes. Whenever outputs of inference rules are matched with the inputs of the requirement generation rules, a requirement text is generated and represented in the proposed format.

This application can also generate product requirements by using the streaming data of a product. A case study for a quad-copter is discussed to show how the proposed architecture supports product requirement management and generate product requirements across the life-cycle phases. Real-time vibration data is streamed from a quad-copter, inspection requirements for the users are automatically executed, and maintenance requirements are generated for the users by the developed applications. The Computational platform receives vibration data from the quad-copter when users turn it on during the usage stage. This data is used to execute inspection requirements, which users have to execute before every flight. Inspection requirements are first created by designers in the PLM system as textual format during the design stage. Then they are converted into the proposed representation format by NLP application. The intention to convert them to the proposed format is to make them machine understandable and processable. The Computational platform can then read this information for the inspection requirements and execute the requirements by using the developed inspection model. First, it checks components for structural damage then checks the installation of propellers and motors. If a structural fault is detected, this information is transferred to the PLM system to generate maintenance requirements about the structural fault. Figure C.3 illustrates the inspection requirement in the PLM system, which is created by designers during the design stage and generated maintenance requirements by the computational platform during the usage stage. After the computational platform executed the inspection requirement (R-1), a maintenance requirement is created and the status of the inspection requirement is promoted to 'executed' in the PLM system. The generated maintenance requirement is then represented as a

separate requirement instance in the PLM system as seen in figure C.3. The detailed description of this case study can be found at C.1

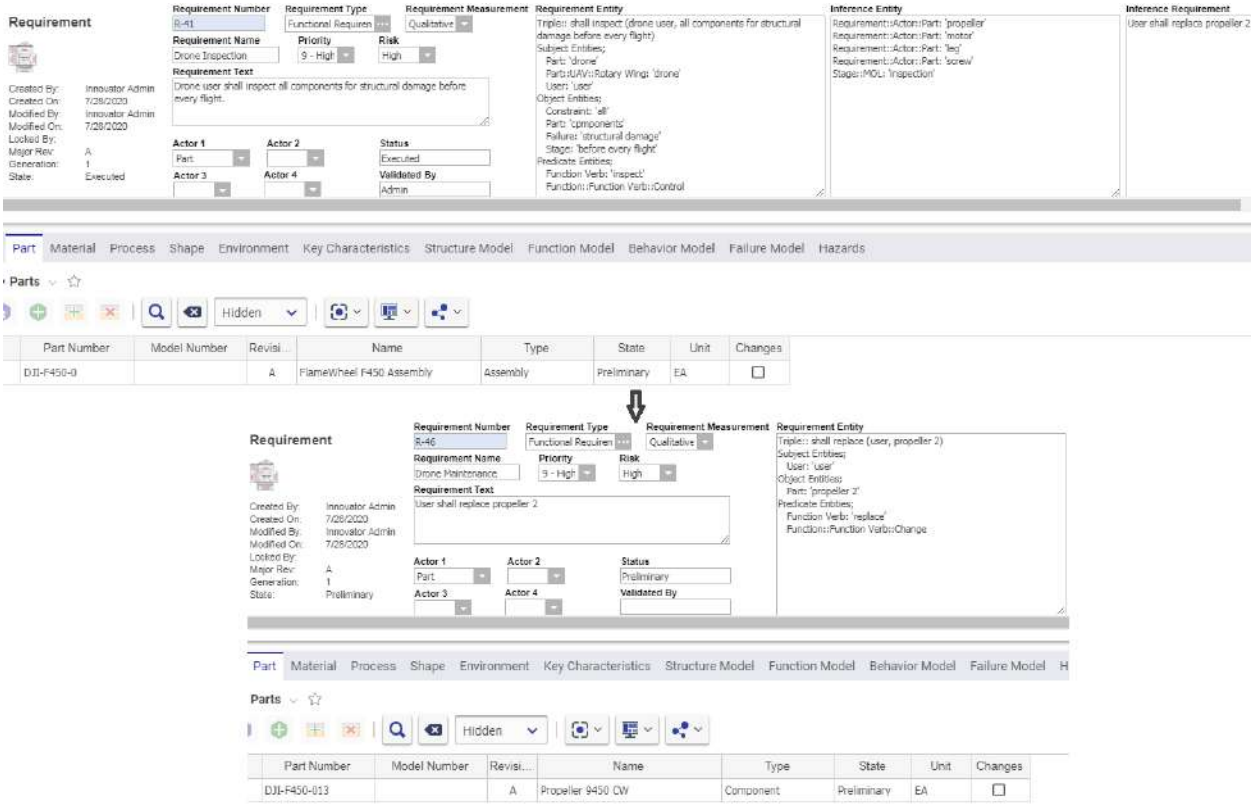


Figure 7.9: Maintenance Requirement Generation

## 7.3 Summary

An analytical application is proposed in this chapter, which automatically constructs a structured product requirement data and semantic-based representation of requirements from unstructured textual representation using natural language processing and PDROM ontology structure. This application is mainly developed for the product requirement information extraction and retrieval and requirement text generation in the use of the REKMA. The requirement NER model is developed

and validated for requirement information extraction. Also, requirement information inference and requirement text generation by using a rule-based system and controlled requirement syntaxes are presented.

# CHAPTER 8

# CONCLUSION

In this chapter, the conclusion of this dissertation and the contributions of this research are discussed. First, the proposed studies for the formal product requirement representation and generation and related works are summarized. Then, research contributions are addressed. Finally, the limitation of the research and possible future directions for extending the works presented in this dissertation are discussed.

## 8.1 Dissertation Summary

The product requirement generation and representation for any product have always been a challenging job during the product design phase and further life-cycle phases. Determination of appropriate criteria for the product design and further life-cycle activities depends on the requirement specifications, which usually has manually written document-based representation. This natural language representation may lead to inconsistencies and incompleteness in product requirements and it is hard to process by data analytics applications. The use of the technological advantages and analytical applications in product design and further life-cycle phases requires requirement management studies evolve into model-based approaches, which can formally represent and help to generate product requirement and requirement related information. A knowledge-based approach is used in this dissertation for the formulation of requirement specification to support the product requirement representation and generation. Information models and an integrated requirement

knowledge management architecture including both PLM and SW technologies are developed to define product requirement data, metadata, and relationships and represent them in individual requirements statements. Three major studies of this dissertation are divided into the following aspects:

The first major study is the development of an efficient and effective requirement knowledge management architecture (REKMA) that covers both the PLM and SW technologies. It captures product requirements and related information within the data structures of the information model(s), manages them within the PLM system and computational platform, and stores them as both graph data model and relational format. To build this architecture, first ontology-based information models are developed to present product requirement classes and relationships and provide a huge set of predefined requirements metadata for the requirement specification. The PLM system is used as a data source and user interface for the requirement knowledge management. It enables users to access and manage product requirement data and documents. It is also used as a management tool to control requirement information flow across the product life-cycle phases. In this dissertation, a PLM tool (Aras Innovator) is enhanced using the proposed requirement information model (PDROM) and integrated with a computational platform for the requirement representation and generation activities. Representation formats for the product requirement sentences and inference requirement information are proposed. Requirement syntaxes for requirement text generation are discussed. The computational platform is created to convert any product requirement text into the proposed requirement representation formats. It generates requirement information from the requirement and related data stored in the PLM. It also generates requirement texts in the proposed syntax structure by using the generated requirement information. It supports the two-way data exchange between the computational platform and the PLM system and it trans-

lates the product requirement data and metadata from the PLM into the RDF knowledge-base. To support the data exchange among the computational platform, the PLM system, and the RDF knowledge-base, a communication mechanism is developed.

The second major study is the development of the ontology-based product requirement information model for the formal product requirement representation. A set of sub-models is also developed and discussed in the product domain models: Material Information Model (MIM), Process Information Model (PIM), and Shape Information Model (SIM). PDROM and domain models are discussed in the three main sections: (1) a formal PDROM description using the UML class diagram, (2) a formal Web Ontology Language (OWL) based PDROM implementation and (3) implementation of PDROM data structure in the PLM relational database. The proposed PDROM and domain models have the capabilities to specify and further analyze the product requirements information by explicitly capturing the requirement semantics. As a proof of concept, PDROM and domain models are implemented in Protégé, and description rules are established to offer solutions to problems, such as inconsistency and completeness check in product requirements management. How PDROM can store requirements and detect incompleteness or inconsistency in the requirement specification are discussed. The rule-based reasoning and Web Rule Language (SWRL) rules for formal machine reasoning and interpretation of PDROM for requirement analysis are represented.

The third major study is the development of the analytical applications for the computational platform to support requirement representation and generation. These applications can generate requirement information and knowledge from product requirement data. Requirement information and knowledge generation increases the reliability of product requirement specification and provides a better insight into the product requirements to all the stakeholders. The main analytical ap-

plication developed for the computational platform is the product requirement information extraction and retrieval. It is utilized to extract and retrieve requirement semantics and information from the textual product requirements by combining the ontology models and Artificial Intelligence (AI). AI technologies; natural language processing (NLP) and machine learning techniques are used to develop a Named Entity Recognition (NER) model to support this application by capturing product requirement concepts automatically in a requirement sentence. The development of this application combines two main studies; syntax analysis of the product requirements and semantic analysis of product requirements through the ontology models.

The requirements of an automobile brake system are utilized in PDROM and REKMA to demonstrate the models' usage and their validity. Both SW and relational database successfully accomplish two main works of product requirement management process: (1) the formal representation of product requirement specification and (2) the product requirement generation with requirement rationale. Models are validated through different applications. Both PDROM and REKMA are open to continuous update and modification in order to increase the capability and usability.

## 8.2 Research Contribution

This work contributes to the formal representation of the product requirements and requirement-related information. It also supports autonomous requirement information and text generation across the product life-cycle phases and provides an understanding of the requirements by representing their rationale. It is a first attempt for the development & utilization of a comprehensive model-based product requirement specification supported by ontology-based information models, artificial intelligence techniques, semantic web, and product life-cycle management technologies.

Two major contributions of this study are listed as follows:

The first major contribution is the semantics-based representation and generation of the product requirements and related information through the proposed ontology-based information models. The PDROM represents a conceptual information model in the domain of design requirement specification, and it is created by using the UML class diagram. It is implemented into the OWL for formal machine reasoning, interpretation, and web search queries. Formal implementation of PDROM is discussed, and this contribution is divided into four sub-contributions:

1. Development of the product requirement and domain ontology models to represent the knowledge structure of product requirements using UML schema.

2. Instantiation of the PDROM and OWL implementation to specify product requirement information that can be published and accessed through the web.

3. Development of the SWRL rules for inference and reasoning mechanisms

4. Development of the SWQRL rules for searching classified ontology and necessary information retrieval.

The second major contribution is the development of the REKMA that covers both the PLM and SW technologies for the formal representation and generation of the product requirements and related information across the product life-cycle phases. This architecture fills the requirement specification gaps by enabling the use of analytical applications in requirement management. It supports the extraction and retrieval of the requirement information and semantics from requirement sentences to represent them in the proposed formats. It also supports the generation of the requirement information and text. These applications are divided into two aspects:

1. An NLP application that automatically extracts requirement information and semantics from the textual requirement description. It includes the requirement NER model developed by

using python's spaCy module. This application is built on two types of studies (i) syntax analysis that describes the syntactic structure of requirement text and identifies entity types of requirement, and (ii) semantic analysis that reveals the semantic meanings of the text based on the PDROM.

2. Analytical applications that support the generation of the requirement and requirement-related information across the product life-cycle and requirement text generation. For these purposes, three applications are generated:

I. Functional requirements architecture for a coffee maker that represents requirement knowledge extraction and functional requirement generation.

II. Knowledge-based requirement information inference and requirement text generation.

III. Product inspection and maintenance requirement generation.

In general, this dissertation contributes to product requirement management by proposing a model-based approach for product requirement generation and representation. The proposed approach guides the designers/engineers through the requirements definition to requirements specification by checking, testing, and formally representing requirement specifications during the requirement generation syntactically and semantically. It obtains semantic information gathered directly from the requirement texts by converting them into proposed representation formats. It also generates requirement information and analyzes the requirements. It offers immediate feedback to the user before the design, process, or any life-cycle activities start. It increases the users' understanding of the problem by supporting them with expert knowledge and eliminates mistakes and misunderstandings. This process is repeatable until there is a consensus between the designers/engineers and end-users. The distinctive features of this model-based approach for the users are:

- To automatically convert a requirement sentence into the proposed representation format, infer requirement information, and generate requirement text
- To provide useful support to designers/engineers to adhere to the requirements specification activities across the product life-cycle phases. It allows users to generate requirement in a textual format as required
- To provide more detailed descriptions of the high-level system definition to increase the users' understanding of the product requirements and related knowledge by using requirement-specific ontologies
- To represent dependencies in requirement and product information across the product life-cycle phases, and in identifying the downstream effects of the requirement information
- To generate the requirement text in a standardized syntax so that requirements specification documents would be easier to read and understand by the human users
- To create an automatic graph-based representation of the requirements from the requirement text so that requirements specification documents can be machine-interpretable; (i) it provides completeness and consistency checking with respect to the proposed ontology model, and (ii) it supports a powerful query system for the specific requirement information requirements.

## 8.3 Limitation and Future Work

This dissertation addresses the benefits of model-based design methodologies and considerations of lifecycle activities in the development of the product requirement specification. While it expresses

the utilization of the model-based requirement representation and generation in the domain of requirement specification with both OWL and SQL implementations and many potential applications, there remain many opportunities to extend the scope of this dissertation. The proposed models and applications can be improved further by extending works as listed below:

In the future, the main goal is to continuously extend the proposed PDROM and implement it in a web ontology environment to make it a reference (standard) model by doing further modifications and validation with more requirement specification applications and populations of more instances.

In OWL implementation of PDROM, this work uses pre-defined mechanisms of Protégé plug-ins, which are not specifically developed for requirement specification. For richer semantics of product requirements, one of the long-range goals is to develop Protégé plug-ins that support requirement management through the product life-cycle activities.

The REKMA can be supported with more comprehensive semantic and syntactic studies of the product requirements. An elaborative syntax analysis for the different kinds of product requirements and a more powerful requirement semantic extraction model are still needed. The requirement NER model can be continuously trained using the requirement data entered into the PLM system. Requirement text generation can be supported with more requirement syntaxes to increase the options while generating different kinds of product requirements. Also, the number, usability, and reusability of the requirement inference rules can be increased by working with the experts of every single product domain and product life-cycle phase.

Finally, a possible direction for future work is to increase the autonomy of the REKMA. A more comprehensive communication mechanism that autonomously converts product requirement data in the PLM SQL database into the RDF format is required for future applications. As the reader might notice, the application that extracts and retrieves requirement information from the

product streaming data needs user interaction. The long-range goal is to develop requirement specification applications that can generate requirement information and text from the product data that is streamed at each of the product life-cycle activities and integrate them into the REKMA without the need for user interaction.

# Appendices

# APPENDIX A

# PRODUCT REQUIREMENT INFORMATION MODEL

## A.1 UML Description

The UML (Unified Modeling Language) class diagram is an ideal object-oriented tool to represent the proposed ontology-based information models since it provides classes, objects, attributes, and functions to represent the relationships between domain concepts. In a UML class diagram, a rectangular box represents classes or concepts while a diamond arrow, hollowed triangular arrow, and line identify the relationships between classes. The Composition relationship is denoted as a filled diamond arrow, Inheritance relationship is denoted as a hollowed triangular arrow, and Association relationship is denoted as a straight line as shown in figure A.1.

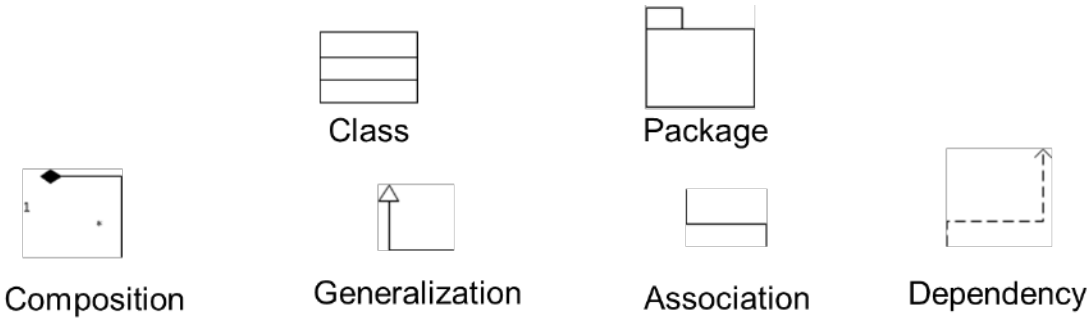


Figure A.1: UML Diagrams

## A.2 Material Properties in the Material Information Model

**Mechanical Properties**: The mechanical properties are considered as the most important properties compared to others (physical, chemical, thermal, electrical, etc.). They usually relate to the

elastic or plastic behavior of material and are defined as the characteristics of the material under force. Examples of mechanical properties are; strength, hardness, elasticity, plasticity, and fatigue properties.

**Physical Properties**: The physical properties are characteristics of materials that can be perceived or measured without changing the chemical nature of matter. Examples of physical properties are; thermal properties, electrical properties, viscosity, smell, density, the condition of the outer surface of the product such as color, texture, surface finish, and other surface attributes. etc.

**Chemical Properties**: The chemical properties are characteristics of materials that can be perceived or measured by changing the chemical nature of matter by performing a chemical change or chemical reaction. Examples of chemical properties are; corrosion resistance, chemical resistance, porosity, flammability, reactivity with water, PH, etc.

**Environmental Properties**: The environmental properties are not listed in property handbooks, but they are very important properties for selecting a material. These properties are embodied energy, carbon footprint, embodied water, end-of-life options, and RoHS compliance.

**Manufacturing Properties**: The manufacturing properties are the ability of materials when they are converted into the product. They are called based on the manufacturing processes such as machinability, formability, weldability, etc.

### A.3 Process Information Model

Figure A.2 shows the expanded PIM for the powder metallurgy process. The expanded PIM keeps most entities of the generic PIM but expands the ***Material*** class, the ***ManufacturingProcess*** class and the ***Activity*** class to serve for powder metallurgy manufacturing process and a specific material. The differences between this expanded model and the generic PIM is mainly on the

composition of the material and process activities.

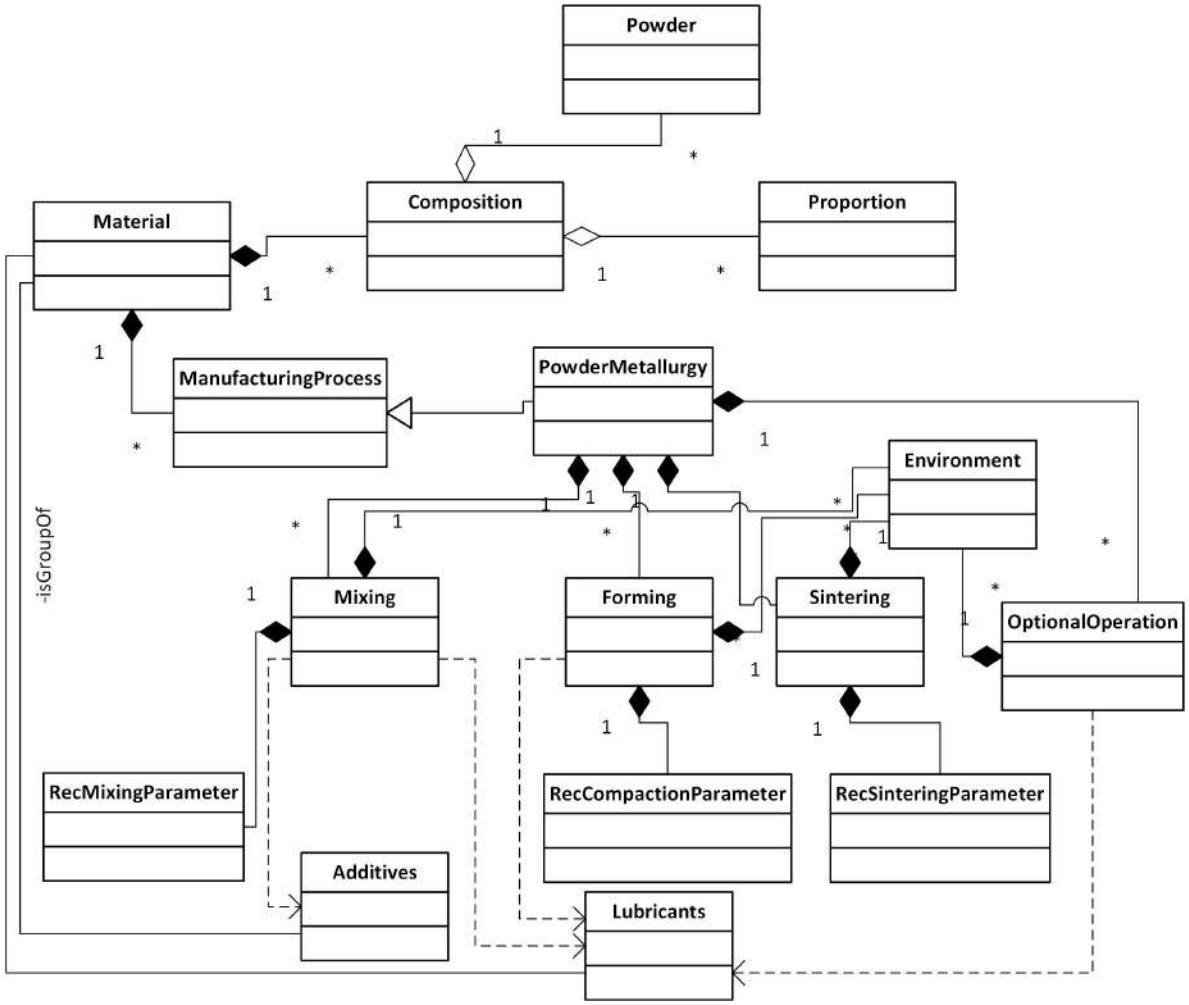


Figure A.2: Expanded PIM for the Powder Metallurgy Process

To write a consistent and complete set of process requirements for the powder metallurgy process activities for a certain product, a deep understanding of not only part geometry, size, material type, and material properties, but also effects of these factors on powder metallurgy process and production are needed. Critical factors such as sintering time, temperature, etc. are identified and further used to bridge manufacturing activities with part/process characteristics as illustrated in Table A.1.

Table A.1: Factors Affecting Powder Metallurgy Part Quality

| | | Part/Process Characteristics | | | | |
|---|---|---|---|---|---|---|
| **Factors Affecting Powder Metallurgy** | | Part Geometry | Part Size | Material Processability | Material Properties | Process Type |
| **Characteristics of Powder** | Particle Shape | X | | | X | X |
| | Particle Size | X | | | X | X |
| | Particle Size Distribution | X | | | X | X |
| | Flow Rate | X | | X | X | X |
| | Purity | | | | X | X |
| **Lubricants and Additives** | Type and amount of lubricants | | X | X | X | X |
| | Type and amount of additives | | | X | X | X |
| | Thermal properties of lubricants and additives | | | | | X |
| **Blending and Mixing** | Type, volume and volume ratio of mixers or blenders | | | | X | X |
| | Type of metal powders and their characteristics (Surface properties, sizes and shape) | X | X | X | X | X |
| | Mixing or blending time | | | X | X | X |
| | Mixing or blending temperature | | | X | X | X |
| | Type of mixing ( wet or dry) | | | X | X | X |
| | Type and amount of additives and lubricants | | | | X | X |
| **Compaction** | Methods and volume of compaction | X | X | X | X | X |
| | Type of compaction (Hot, Warm or Cold) | X | X | X | X | X |
| | Compaction tools' characteristics (structure and surface) | X | X | X | X | X |
| | Type and amount of die wall lubricant | | X | X | X | X |
| | Type of metal powders and their characteristics | X | X | X | X | X |
| | Compaction time | X | X | X | X | X |
| | Compaction temperature | X | X | X | X | X |
| | Compaction pressure | X | X | X | X | X |
| | Type of die materials and their properties | | X | X | | X |
| **Sintering** | Sintering methods | X | X | X | X | |
| | Type and amount of used lubricants and additives) | X | X | X | X | X |
| | Type of metal powders and their characteristics | X | X | X | X | X |
| | Sintering time | | X | X | X | X |
| | Sintering temperature | X | X | X | X | X |
| | Green compact density | | | | X | X |
| | Green compact surface quality | | | | X | X |
| | Sintering atmosphere | | | X | X | X |
| | Rate of heating and cooling | X | X | X | X | X |
| | Furnace capability | | X | | X | X |

Table A.1 shows that each activity of powder metallurgy part production has critical factors that must be considered during a product's conceptual design stage to generate complete design requirement specification. These factors should further be defined with consideration of part/process characteristics such as part geometry constraints, part size, material processability, material properties, and process constraints. Failing to precisely control these factors will yield undesirable properties. As an example, if the lubricant amount is not defined in design requirement specification and not arranged properly, it will affect flow rate, apparent density, and of course finished part's properties.

# APPENDIX B

# PDROM IMPLEMENTATION IN PROTEGE

## B.1 SWRL and SQWRL Rules

Semantic rules which prescribe many restrictions for the requirement classes, properties, and instances are developed and represented in rule languages (SWRL and SQWRL) to reveal the meaning of requirements and requirements' rationale. These rules are empirical and subjective, and they can be extended if needed. Some of the rules are explained below:

**Requirement**(?x) ∧ **AtomicFunction**(?y) ∧ ***hasAtomicFunctionEntity***(?x, ?y)
→ **FunctionalRequirement**(?x)

This rule defines the meaning of **FunctionalRequirement** and states that a **FunctionalRequirement** is a **Requirement** which has **AtomicFunction**. The reason behind such a definition is the fact that functional requirement has to be represented with functional entities which are mainly atomic functional verbs.

**Requirement**(?x) ∧ ***hasEnvironmentalPropertyEntity***(?x, *recyclability*) ∧
**EnvironmentalProperties**(*recyclability*) ∧ **Stage**(?y) ∧ ***hasStageEntity***(?x, ?y)
→ **BOL**(?y)∧EOL(?y)

This rule defines specific stages of **Requirement** that have **EnvironmentalProperties** that are *recyclability*. Such a definition is because recycle activity is mainly about material selection,

which happens in the design stage and material retrieval from the discarded product at the end of the product life cycle.

**Requirement**(?x) ∧ ***hasObjectiveEntity***(?x, *minimize_weight*) ∧

**Objective**(*minimize_weight*) → ***hasPhysicalPropertyEntity***(?x, *density*) ∧

***hasBOLStageEntity***(?y, *material_selection*)

This rules indicates that a **Requirement** has an **Objective** which is *minimize_weight*. This objective is related to a **PhysicalProperty**, which is *density*, and a **BOL** stage, which is *material_selection*. This rule is constructed based on the information that minimizing weight is mainly about part material density and is considered in the material selection stage.

**Requirement**(?x) ∧ **Flow**(*friction*) ∧ ***hasFlowEntity***(?x, *friction*) →

**Flow**(*heat*) ∧ ***hasOutputFlowEntity***(?x, *heat*) ∧ **FailureMode**(*abrasive_wear*)

∧ ***hasFailureModeEntity***(?x, *abrasive_wear*)

This rule describes **Requirement** inference entities when a requirement instance is *friction*, which is a **Flow** instance. It defines *abrasive_wear* as a **FailureMode** entity and *heat* as a **Flow** entity which are results of *friction*. Such a definition is because input and output flows can cause failures.

More rules are also defined in Protégé related to **MaterialProperties** and material selection **Stage**. For example, the following rules define expert knowledge to reveal material properties and avoid material related failures.

**Requirement**(?x) $\wedge$ ***hasFailureModeEntity***(?x, *abrasive_wear*) $\wedge$ **FailureMode**(*abrasive_wear*) $\rightarrow$ ***hasMaterialPropertyEntity***(?x, *wear_resistance*)

**Requirement**(?x) $\wedge$ ***hasFlowEntity***(?x, *friction*) $\wedge$ **Flow**(*friction*) $\rightarrow$ ***hasMaterialPropertyEntity***(?x, *friction_coefficient*)

**Requirement**(?x) $\wedge$ **FailureMode**(*abrasive_wear*) $\wedge$ ***hasFailureModeEntity***(?x, *abrasive_wear*) $\rightarrow$ ***hasMaterialPropertyEntity***(?x, *surface_hardness*))

**Requirement**(?x) $\wedge$ ***hasMaterialPropertyEntity***(?x, ?y) $\wedge$ **MaterialProperty**(?y) $\rightarrow$ ***hasBOLStageEntity***(?x, *material_selection*) $\wedge$ BOL(*material_selection*)

These kinds of rules help the user generate complete and consistent sets of requirements for a specific product during the design stage. Possible relationships between the **Flow**, **Failure Mode**, and **Material** actor must be well defined with semantic rules in order to generate the necessary requirements and complete the requirement set for requirement specification. For example, *friction* is a mechanical substance (flow entity) and causes *abrasive wear* and *heat*. *Heat* is directly related with the *thermal fatigue* and *thermal shock* failure modes. These failure modes are also related to the *thermal diffusivity* that helps to remove generated *heat* and avoid *thermal shock* and *thermal fatigue*. Furthermore, increasing *surface hardness* will decrease *abrasive wear* that occurs on *surfaces*. This knowledge shows that *friction* flow has a relationship with **Material** attributes (**MaterialProperty**) and that all this information must be specified in other **Requirement** indi-

viduals to complete the PDS. We also developed query rules in Protégé for the PDROM OWL ontology in order to extract requirement information.

**Requirement**(?x) ∧ Requirement_ID(?x, ?*id*) ∧ hasDescriptionText(?x, ?*description*) ∧ ***hasActor***(?x, ?*actor*) ∧ ***hasStage***(?x, ?*stage*) ∧ ***hasRequirementMeasurement***(?x, ?*measurement*)

→ sqwrl:select(?*id*, ?d*escription*, ?*actor*, ?*stage*, ?*measurement*)

This query rule searches and extracts the **Requirement** information from the PDROM OWL ontology. Mainly, requirement individuals represented with requirement I.D., description, **Actor**, **Stage**, and **Measurement** information.

**Requirement**(?x) ∧ ***hasPartEntity***(?x, *rotor*) ∧ Requirement_ID(?x, ?*id*)∧ hasDescriptionText(?x, ?*description*) ∧ ***hasMaterialPropertyEntity***(?x, ?*materialProperties*)

→ sqwrl:select(?*id*, ?*description*, ?*materialProperties*)

This query rule extracts the **Material Property** information with the related requirement I.D. and description from **Requirement** individuals for the *rotor* component. We need such a query because design engineers in the material selection stage can more easily identify material properties to select the most appropriate material for the components.

# APPENDIX C

# DEVELOPMENT OF THE COMPUTATIONAL APPLICATIONS

## C.1 Building of the Product Function Architecture

### C.1.1 Collection of Functional Data for Coffee Makers

There are many activities in brewing coffee with a coffee maker. They can be listed in order like; water in the reservoir flows through the hole into plastic and aluminum tubes. The heating element starts heating the aluminum tube and heats the water in the tube when the switch is on. Heated water rises up in the plastic tube and reaches the pouring sprayer, and the water flows through the ground coffee beans in the filter basket. Coffee grounds stay in the filter, and brewed coffee passes through the filter and drips into the carafe by gravity. Once the coffee is made, the heating element keeps the coffee warm. Figure C.1 shows the general system component structure and function model with black box representation. The coffee maker consists of many components. The functions of these components are sub-functions of the coffee maker's main function, which is 'brew coffee'.

In order to create the functional requirement architecture for the coffee maker, functional data of seventeen different coffee makers, related to the above activities are entered into the PLM system. Students who take the Design for Manufacturing (MFE 692) course through three semesters generate data for coffee makers in the PLM system. The function structure of each coffee maker is established through functional decomposition and functional data are entered into the PLM system by students. They use function and flow taxonomy tables, which are defined in

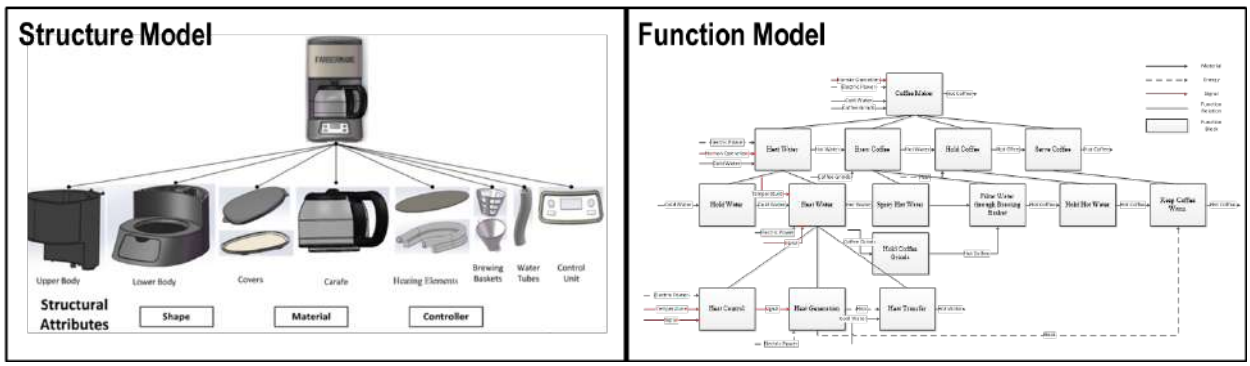


Figure C.1: Structure and Function Representation for a Coffee Maker

[114] to create functional data for coffee makers. The coffee maker consists of many components.

By following the coding schemes provided in [114], users generate the function model table with defining product name (P#); function name; atomic function verb; flow and codes for atomic function (Af); flow energy code (Iw-Eng); input flow material (Iw-Mat); input flow signal (Iw-Sng); output flow energy (Ow-Eng), output flow material(Ow-Mat), and output flow signal (Ow-Sng). The function model data for two out of seventeen coffee makers are shown in table C.1.

Table C.1: Function Model Data of Coffee Makers

| P# | Function Name | Atomic Function Verb | Flow | Af | Iw-Eng | Iw-Mat | In-Sng | Ow-Eng | Ow-Mat | Ow-Sng |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | heat generation | generate | heat | 0.302 | 0.1202 | 0 | 0.3006 | 0.1304 | 0 | 0 |
| 1 | heat transfer | transfer | heat | 0.403 | 0.1304 | 0.2102 | 0 | 0 | 0.2102 | 0 |
| 1 | hold water | hold | water | 0.905 | 0 | 0.2102 | 0 | 0 | 0 | 0 |
| 1 | hold coffee grinds | hold | coffee grinds | 0.905 | 0 | 0.2006 | 0 | 0 | 0 | 0 |
| 1 | spray hot water | hold | hot water | 0.413 | 0 | 0.2102 | 0 | 0 | 0.2101 | 0 |
| 1 | filter coffee | filter | coffee | 0.142 | 0 | 0.2101 | 0.3108 | 0 | 0 | 0 |
| 1 | hold hot coffee | hold | hot coffee | 0.905 | 0 | 0.2101 | 0 | 0 | 0 | 0 |
| 1 | keep coffee warm | keep warm | coffee | 0.905 | 0 | 0.2101 | 0 | 0 | 0.2101 | 0 |
| 1 | control heat | control | heat | 0.503 | 0 | 0 | 0.3102 | 0 | 0 | 0.3006 |
| 2 | heat generation | generate | heat | 0.302 | 0.1202 | 0 | 0.3006 | 0.1304 | 0 | 0 |
| 2 | heat transfer | transfer | heat | 0.403 | 0.1304 | 0.2102 | 0 | 0 | 0.2102 | 0 |
| 2 | hold coffee grinds | hold | coffee grinds | 0.905 | 0 | 0.2006 | 0 | 0 | 0 | 0 |
| 2 | hold water | hold | water | 0.905 | 0 | 0.2102 | 0 | 0 | 0 | 0 |
| 2 | filter coffee | filter | coffee | 0.142 | 0 | 0.2101 | 0.3108 | 0 | 0 | 0 |
| 2 | hold hot coffee | hold | hot coffee | 0.905 | 0 | 0.2101 | 0 | 0 | 0 | 0 |
| 2 | keep coffee warm | keep warm | coffee | 0.905 | 0 | 0.2101 | 0 | 0 | 0.2101 | 0 |
| 2 | heat control | control | heat | 0.503 | 0 | 0 | 0.3102 | 0 | 0 | 0.3006 |
| 2 | auto brew | brew | signal | 0.603 | 0 | 0 | 0.3005 | 0 | 0 | 0.3005 |
| 2 | auto shut off | shut off | signal | 0.625 | 0 | 0 | 0.3005 | 0 | 0 | 0.3005 |

### C.1.2 Building Functional Architecture for Coffee Maker

Functional data of coffee makes are first created in the PLM system. To create the functional architecture of a coffee maker, these data are retrieved using AML studio and analyzed using neural networks by the computational platform.

Figure 7.1 represents that how the functional data of seventeen coffee makers are first transferred from the PLM system into the computational platform using an AML studio. After running data processing, functions of seventeen coffee makes are clustered by neural networks according to their similarity. Cluster analysis clusters similar functions of a coffee maker in the same group. Besides the general function of a coffee maker, ***brew coffee***, nine sub-function clusters are identified for the coffee maker product by clustering analysis. The clusters represent the common functional information set for a coffee maker, which is denoted as follows: ***hold water***, ***hold coffee beans***, ***generate heat***, ***transfer heat***, ***control heat***, ***filter coffee***, ***keep coffee warm***, ***fetch coffee***, and ***display status***.

## C.2 Product Inspection and Maintenance Requirement Generation

This application is developed to aid product users to do execute inspection requirements about the product structure health and generate maintenance requirements during the usage stage of the product. A machine learning model is developed for this application by analyzing vibration data of Unmanned Aerial Vehicles (UAVs) to perform fault detection on the component of the vehicle. This model is called vibration-based structural health monitoring and discussed in [150]. After the model is developed, an Unmanned Aerial System (UAS) repository is created in the PLM system to manage UAV inspection and maintenance requirements and related information.

Table C.2: Representation of the Drone Inspection Requirements

| | |
|---|---|
| **R-1**: *Drone user shall inspect all components for structural damage before every flight*<br>**Triple::** *shall inspect* (*drone user*, *all components for structural damage before every flight*)<br>**Subject Entities;**<br>Part: '*drone*'<br>Part::UAV::Rotary Wing:'*drone*'<br>User:'*user*'<br>**Object Entities;**<br>Constraint: '*all*'<br>Part: '*components*'<br>Failure: '*structural damage*'<br>Stage: '*before every flight*'<br>**Predicate Entities;**<br>Function: '*inspect*'<br>**Inference Entity;**<br>Requirement::Actor::Part: '*propeller*'<br>Requirement::Actor::Part: '*motor*'<br>Requirement::Actor::Part: '*leg*'<br>Requirement::Actor::Part: '*screw*'<br>Stage::MOL: '*inspection*' | **R-2:** *Drone user shall check for the proper installation of components*<br>**Triple::** *shall check* (*drone user*, *for the proper installation of components*)<br>**Subject Entities;**<br>Part: '*drone*'<br>Part::UAV::Rotary Wing:'*drone*'<br>User:'*user*'<br>**Object Entities;**<br>Constraint: '*proper*'<br>Property: '*installation of components*'<br>Part: '*component*'<br>**Predicate Entities;**<br>Function: '*check*'<br>**Inference Entity;**<br>Requirement::Actor::Part: '*propeller*'<br>Requirement::Actor::Part: '*motor*'<br>Requirement::Actor::Part: '*leg*'<br>Requirement::Actor::Part: '*screw*'<br>Stage::MOL: '*inspection*' |
| **R-3:** *Drone user shall identify which components need repair*<br>**Triple::** *shall identify* (*drone user*, *which components need repair*)<br>**Subject Entities;**<br>Part: '*drone*'<br>Part::UAV::Rotary Wing:'*drone*'<br>User:'*user*'<br>**Object Entities;**<br>Part: '*components*'<br>Non-Function: '*need*'<br>Failure: '*repair*'<br>**Predicate Entities;**<br>Function: '*identify*'<br>**Inference Entity;**<br>Requirement::Actor::Part: '*propeller*'<br>Requirement::Actor::Part: '*motor*'<br>Requirement::Actor::Part: '*leg*'<br>Requirement::Actor::Part: '*screw*'<br>Stage::MOL: '*inspection*'<br>Failure::Action: '*repair*' | **R-4:** *Drone user shall forward component repairs to maintenance and repairs* department<br>**Triple::** *shall forward* (*drone user*, *component repairs to maintenance and repairs department*)<br>**Subject Entities**;<br>Part: '*drone*'<br>Part::UAV::Rotary Wing:'*drone*'<br>User:'*user*'<br>**Object Entities**;<br>Part: '*component*'<br>Failure: '*repair*'<br>Constraint: '*maintenance and repairs department*'<br>**Predicate Entities**;<br>Function: '*forward*'<br>**Inference Entity**;<br>Requirement::Actor::Part: '*propeller*'<br>Requirement::Actor::Part: '*motor*'<br>Requirement::Actor::Part: '*leg*'<br>Requirement::Actor::Part: '*screw*'<br>Stage::MOL: '*inspection*' |

The Computational platform receives vibration data from the air vehicle when the user turned it on during the usage stage. This data is used to execute inspection requirements, which the user has to do before every flight. Inspection requirements of an air vehicle are created in the PLM system and represented in table C.2. It demonstrates the four inspection requirements of a drone that are created during the design stage. These requirements are first created by the designer in the PLM system as textual format then they are converted to the proposed structured format by Requirement NLP application. Our intention to convert them to the structured format is that computer applications can read and execute the orders of the requirements. The Computational platform receives this structured inspection requirement information and executes requirements through the developed model. First, it checks components for structural damage then checks the installation of propellers and motors. If a structural fault is detected, this information is transferred to the PLM system to generate maintenance requirements about the structural fault. In the end, the PLM system provides information about the structural fault to the product user, as well as to people involved in the product development process.

### C.2.1 Model Development for the Application

In this application, a quadcopter (drone) is used to develop the machine learning model through the collected vibration data in the preflight stage when the drone is armed. Vibration data is collected by using DroneKit-Python API (Application Programming Interface), while drone has no structural faults and it has seven different structural faults as listed below:

- No propeller on motor one
- Propeller 1 is damaged
- Propeller 2 is damaged

o Propeller 3 is damaged

o Propeller 4 is damaged

o Leg 1 has loose screws

o Motor 1 has loose screws

Various time series classification schemes are used to classify the dataset: Gated Recurrent Units (GRUs); Long Short Term Memory networks (LSTMs); Convolutional Neural Network (CNN). After 487 train and 45 test samples of vibration data are processed, we gained an overall test accuracy of 100% with GRU, 99.3% with CNN, and 98.9% with LSTM. Because the GRU performs better than other methods and train faster, this method is used in the computational platform.

### C.2.2 Overall System Architecture Of UAS Predictive Maintenance Application

The PLM platform for UAS is proposed in this application [136] that possess data fusion and has an integrated computational platform for real-time decision-making on the UAV operations by executing UAV requirements. The PLM system is used as a knowledge source for UAS, a files storage and user interface in which the designer can input inspection requirements and the user can access the maintenance requirements, product-related data, and documents.

Figure C.2 introduces the system overview of the implemented UAS system for the vibration-based structural health-monitoring scenario that executes inspection requirements and generates maintenance requirements of the UAS. Some important components of each UAV are shown in figure C.2: the flight controller, accelerometer, and other physical components are virtually represented in the PLM database. The communication mechanisms, which supports the data exchange between the Computational Platform and the PLM system is discussed in chapter 5.

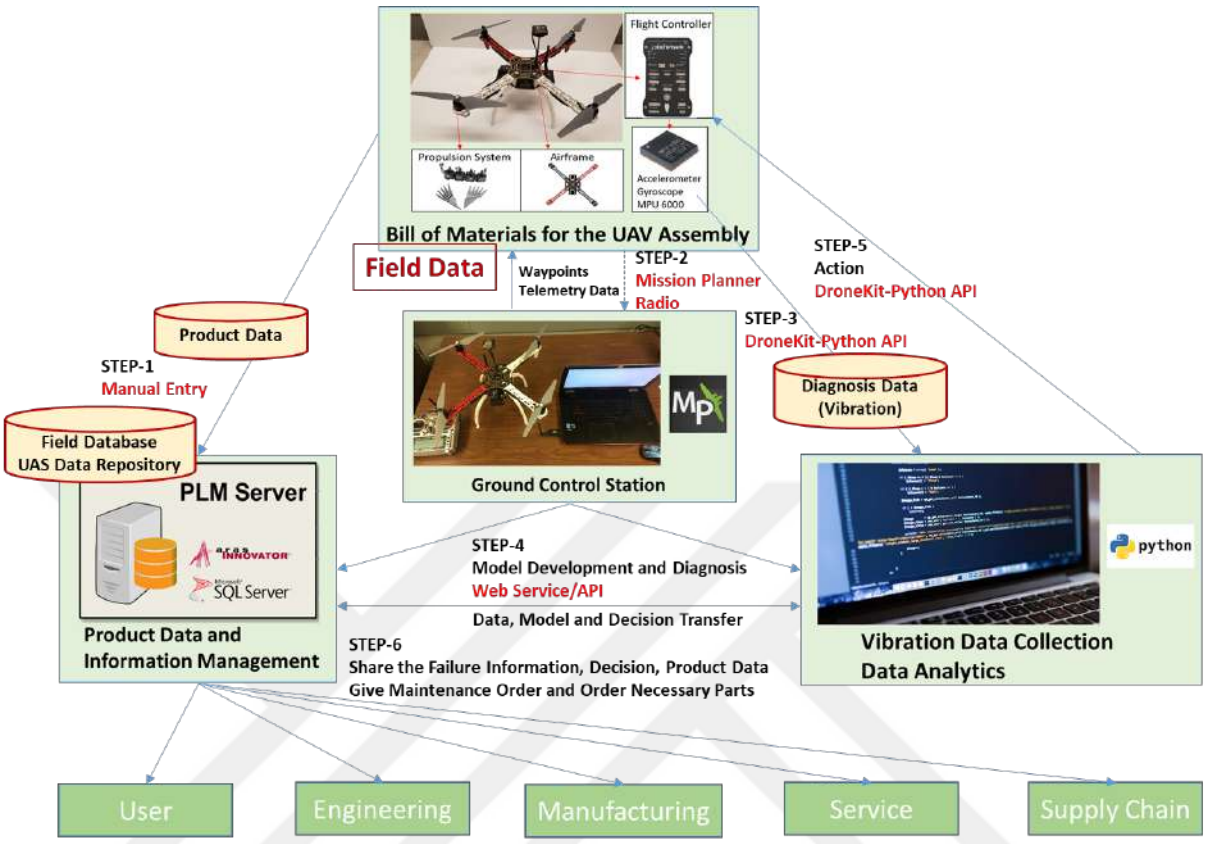


Figure C.2: Overall System Architecture for Inspection and Maintenance Requirement Management

While the computational platform uses DroneKit-Python API to communicate with the UAV, the requirement management system in the PLM and the computational platform communicate through a C# REST (Representational State Transfer) Web Server that is created using open source NHttp, which is a simple asynchronous HTTP server written in C# for the .NET framework.

The Computational platform analyzes streaming data using a developed data analytics method. It executes the order of inspection requirements, which are represented in table C.2 It inspects the components of the drone (R-1) then checks the proper installation of components (R-2), then identifies the components that need repair (R-3), and finally generates maintenance and repair orders for faulty components (R-4). When the model detects the failure, for example, “propeller 2

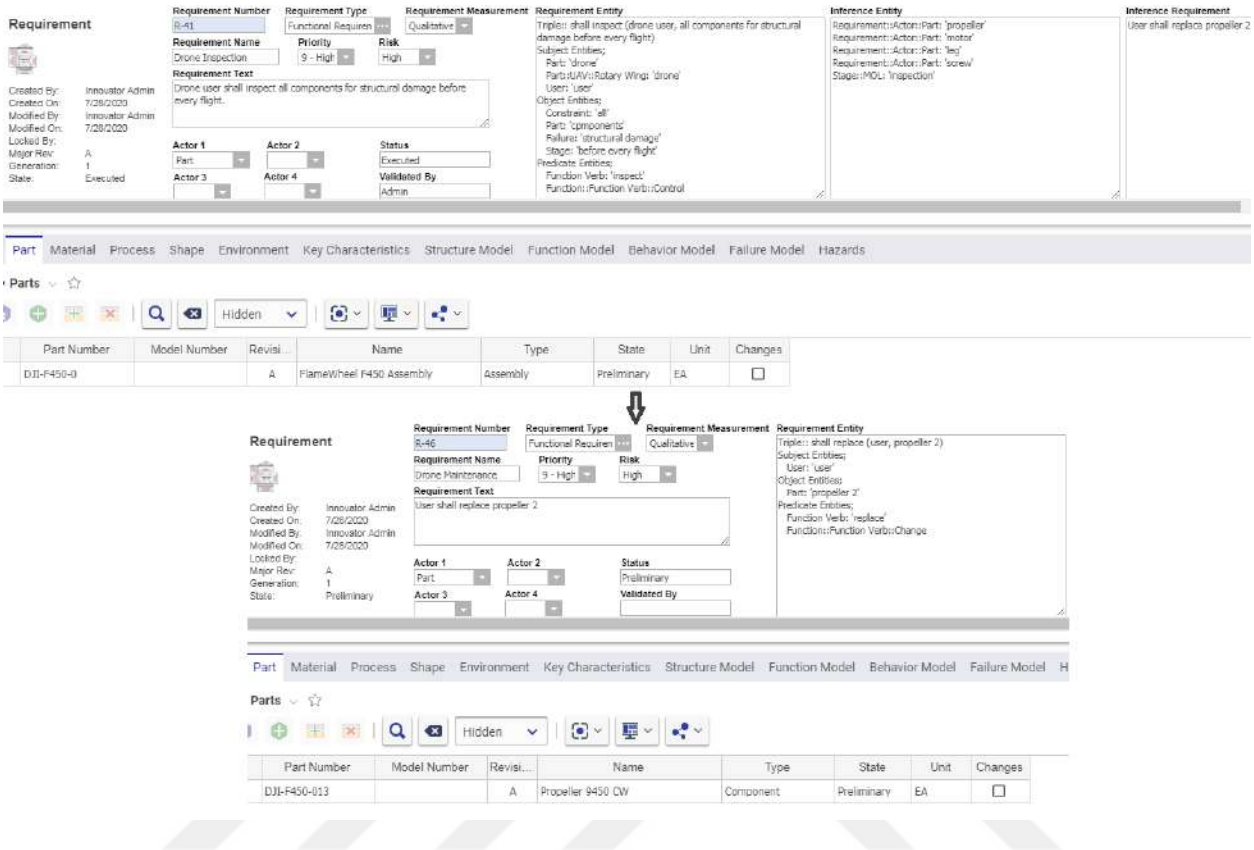


Figure C.3: Maintenance Requirement Generation

is damaged", it generates maintenance and service requirements that 'User shall replace propeller 2'. The prediction result and maintenance requirements are posted in the PLM system by using the REST web service. Figure C.3 illustrates the inspection requirement in the PLM system, which is created by the designer during the design stage and generated maintenance requirements by the computational platform during the usage stage. After the computational platform executed the inspection requirement (R-1), a maintenance requirement is created and the status of the inspection requirement is promoted to 'executed'. The generated maintenance requirement is then represented as a separate requirement instance in the PLM system as seen in figure 14.

Whenever a maintenance requirement is generated and represented in the PLM system, PLM shares this requirement with users by adding supporting documents like the physical specifications

of the propeller, its cost, and supplier, etc. as shown in figure C.4. It illustrates the implementation of this application in the PLM system and the general framework to manage the UAV inspection and maintenance requirements. The PLM system also stores data, which shows how many times this maintenance requirement is generated and shares it with the design department. Briefly, knowledge from field data helps the user to execute inspection requirements, the internal design, and the manufacturing department to improve product requirement traceability, product design, and performance, and the service department has a better knowledge of the product life-cycle.

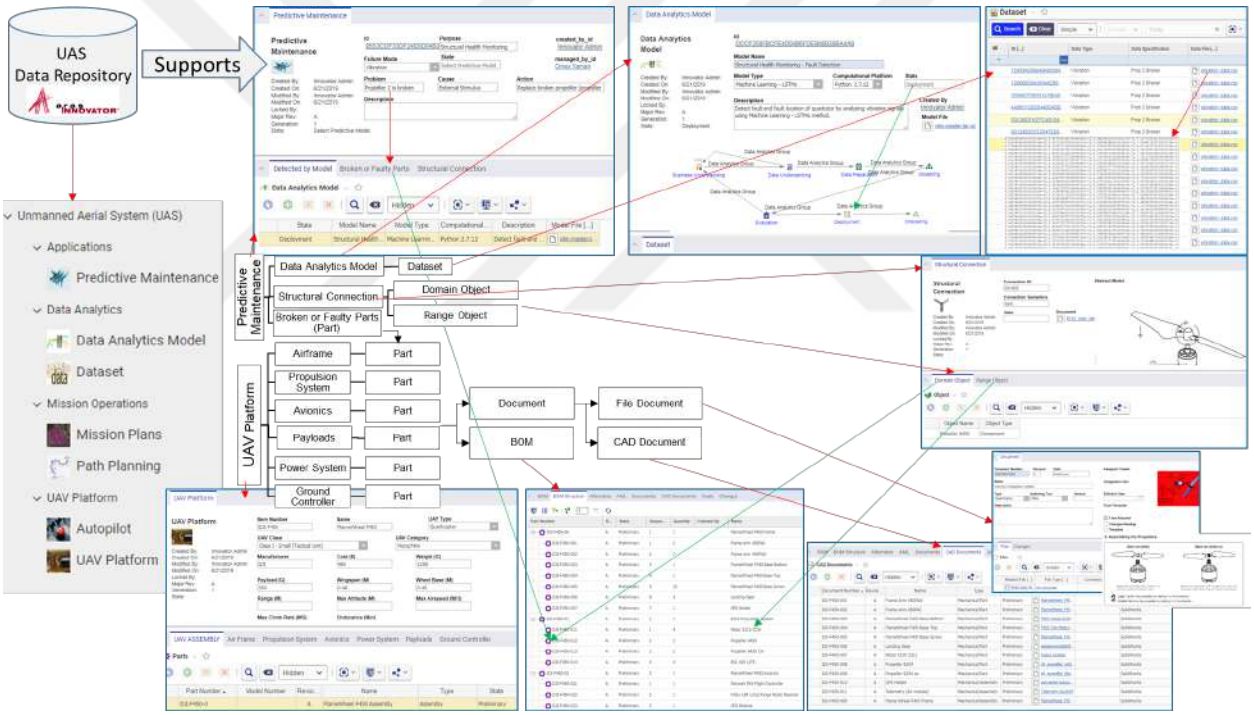


Figure C.4: UAV Data Repository Structure and PLM Implementation

# APPENDIX D
# CUSTOM REQUIREMENT NER

## D.1 Requirement Texts for NER Testing

1. The UAV shall deliver package up to 1 kg.
2. Rotor material shall have high compressive strength.
3. Rotor material shall have low cost.
4. Rotor material shall be suitable for die casting and machining.
5. Rotor material shall have high hardness.
6. Rotor material shall have high wear resistance.
7. Part material shall have high friction coefficient.
8. Part material shall have high thermal conductivity.
9. Part material shall have high thermal diffusivity.
10. Rotor material shall have high corrosion resistance.
11. Rotor material shall have high specific heat capacity.
12. Parts shall dissipate heat.
13. Parts shall resist abrasive wear.
14. Parts shall resist thermal stress.
15. Parts shall resist thermal shock.
16. The drone users shall keep drone within the visual line of sight during flight.
17. The drone users shall check drone for any loose parts.
18. The drone users shall check proper installation of propellers, legs, motors and screws.
19. The drone users shall inspect all components for visible damage before flight.
20. A mechanic shall inspect all components of car.
21. Driver shall check car for fluid leaks.
22. Seat belts shall be easily accessible.

23. The system shall receive, process and store the information from all sensors mounted on the drones during the flights.
24. The drone shall detect the best landing location.
25. The drone shall change path when the drone encounter object or obstacle.
26. The drone shall update location.
27. The drone shall take clear image.
28. The system shall detect errors during flight.
29. The ground station shall receive and transmit flight data in encrypted form.
30. The drone shall keep radio contact with ground station.
31. The drone shall avoid obstacles during flight.
32. Drone operators shall learn aeronautical knowledge.
33. Production shall take less than 2 days.
34. The system shall process every payment transaction in 0.1 second.
35. The drone shall able to fly at rainy weather.
36. The system shall detect user face within 1 second.
37. The drone shall able to carry camera with up to 200 g.
38. Rotor surface shall have holes to dissipate heat.
39. Water pollution during material production shall be low.
40. CO2 emission during disposal shall be low.
41. The drone shall be designed to maximize flight time.
42. The ground station shall provide view of flight.
43. The brake rotor shall be made of carbon steel.
44. The brake rotor shall be made of ceramic for race cars.
45. The brake rotor shall convert energy of motion into heat.

### D.2 NER Test Result for Performance Analysis

**Requirement Text:** The UAV shall deliver package up to 1 kg.
**Prediction:**
Entity: PRODUCT, Value: UAV
Entity: FUNCTION, Value: deliver
Entity: FLOW, Value: package
Entity: CONSTRAINT, Value: up to
Entity: VALUE, Value: 1
Entity: UNIT, Value: kg
Entity: FUNCTION_NAME, Value: deliver package
Entity: MEASUREMENT, Value: up to 1 kg
**Ground truth:**
Entity: PRODUCT, Value: UAV
Entity: FUNCTION, Value: deliver
Entity: FLOW, Value: package
Entity: CONSTRAINT, Value: up to
Entity: VALUE, Value: 1
Entity: UNIT, Value: kg
Entity: FUNCTION_NAME, Value: deliver package
Entity: MEASUREMENT, Value: up to 1 kg

**Requirement Text:** Rotor material shall have high compressive strength.
**Prediction:**
Entity: PRODUCT, Value: Rotor
Entity: MATERIAL, Value: material
Entity: NON_FUNCTION, Value: have
Entity: VALUE, Value: high
Entity: PROPERTY, Value: compressive strength
Entity: MEASUREMENT, Value: high
**Ground truth:**
Entity: PRODUCT, Value: Rotor
Entity: MATERIAL, Value: material
Entity: NON_FUNCTION, Value: have
Entity: VALUE, Value: high
Entity: MATERIAL_PROPERTY, Value: compressive strength
Entity: MEASUREMENT, Value: high

**Requirement Text:** Rotor material shall be suitable for die casting and machining.
**Prediction:**
Entity: PRODUCT, Value: Rotor
Entity: MATERIAL, Value: material
Entity: NON_FUNCTION, Value: be
Entity: CONSTRAINT, Value: suitable
Entity: PROCESS, Value: die casting

Entity: PROCESS, Value: machining

**Ground truth:**

Entity: PRODUCT, Value: Rotor
Entity: MATERIAL, Value: material
Entity: NON_FUNCTION, Value: be
Entity: CONSTRAINT, Value: suitable
Entity: PROCESS, Value: die casting
Entity: PROCESS, Value: machining

**Requirement Text:** The drone users shall inspect all components for visible damage before flight.

**Prediction:**

Entity: PRODUCT, Value: drone
Entity: USER, Value: users
Entity: FUNCTION, Value: inspect
Entity: CONSTRAINT, Value: all
Entity: PRODUCT, Value: components
Entity: CONSTRAINT, Value: visible
Entity: SHAPE, Value: damage
Entity: STAGE, Value: before flight

**Ground truth:**

Entity: PRODUCT, Value: drone
Entity: USER, Value: users
Entity: FUNCTION, Value: inspect
Entity: FLOW, Value: all components
Entity: CONSTRAINT, Value: visible
Entity: FAILURE, Value: damage
Entity: STAGE, Value: before flight
Entity: FUNCTION_NAME, Value: inspect all components

**Requirement Text:** The ground station shall receive and transmit flight data in encrypted form.

**Prediction:**

Entity: PRODUCT, Value: ground station
Entity: FUNCTION, Value: receive
Entity: FUNCTION, Value: transmit
Entity: FLOW, Value: flight data
Entity: CONSTRAINT, Value: encrypted form
Entity: FUNCTION_NAME, Value: transmit flight data

**Ground truth:**

Entity: PRODUCT, Value: ground station
Entity: FUNCTION, Value: receive
Entity: FUNCTION, Value: transmit
Entity: FLOW, Value: flight data
Entity: CONSTRAINT, Value: encrypted form
Entity: FUNCTION_NAME, Value: transmit flight data

## REFERENCES


[1] M. Ashby, *Materials Selection in Mechanical Design (3rd ed.)* Burlington, Massachusetts: Butterworth-Heinemann. ISBN 0-7506-4357-9, 1999.

[2] U. Roy, N. Pramanik, S. Rachuri, R. D. Sriram, and K. W. Lyons, "Function-to-form mapping: Model, representation and applications in design synthesis," *Computer-Aided Design, 33(10), 699-719*, 2001.

[3] J. Jiao and C.-H. Chen, "Customer requirement management in product development: A review of research issues," in *Concurrent Engineering: Research and Applications, 14(3), 173-185*, 2006.

[4] D. Firesmith, "Are your requirements complete?" *Journal of Object Technology, 4(1), 27-44*, 2005.

[5] T. B. Sexton, M. P. Brundage, M. L. Hoffman, and K. C. Morris, "Hybrid datafication of maintenance logs from ai-assisted human tags," in *Proceedings of the 2017 IEEE International Conference on Big Data (BIGDATA)*, 2017.

[6] M. Ashby, *Materials Selection in Mechanical Design (4th ed.)* Burlington, Massachusetts: Butterworth-Heinemann. ISBN 978-1-85617-663-7, 2011.

[7] D. Kiritsis, A. Rolstadas, and B. Moseng, "Promise final activity report," 2008.

[8] M.-J. Yoo, C. Grozel, and D. Kiritsis, "Closed-loop lifecycle management of service and product in the internet of things: Semantic framework for knowledge integration," *Sensors. doi:10.3390/s16071053*, 2016.

[9] D. V. Wijk, A. Etienne, E. Guyot, B. Eynard, and L. Roucoules, "Enabled virtual and collaborative engineering coupling plm system to a product data kernel," in *5th International Conference on Digital Enterprise Technology*, 2008.

[10] M. Sampson and S. Friedenthal, "Model-based systems engineering (mbse) initiative opening plenary," in *Presented at 2011 INCOSE International Workshop*, 2011.

[11] B. London and P. Miotto, "Model-based requirement generation," in *IEEE Aerospace Conference. DOI: 10.1109/AERO.2014.6836450*, 2014.

[12] O. Yaman, B. Zhu, and U. Roy, "Towards the development of an ontology-based product requirement model," in *ASME 2014 International Mechanical Engineering Congress and Exposition*, 2014.

[13] ReQtest, *Reqtest requirements management*, https://reqtest.com/, May 2019.

[14] J. Dick, E. Hull, and K. Jackson, *Requirements Engineering (4th ed.)* Springer, Cham. ISBN 978-3-319-61072-6, 2017.

[15] R. Veryard, "Information modelling: Practical guidance," *Prentice Hall, ISBN: 9780134541822*, 1992.

[16] T. A. Halpin, "Information modeling and relational databases: From conceptual analysis to logical design," *Morgan Kaufmann, ISBN-13: 978-1558606722*, 2001.

[17] S. Fenves, S. Foufou, C. Bock, N. Bouillon, and R. Sriram, "A revised core product model for representing design information," *Tech. Rep. NISTIR 7185, National Institute of Standards and Technology, Gaithersburg, MD 20899, USA*, 2004.

[18] M. M. Baysal, U. Roy, R. Sudarsan, R. D. Sriram, and K. W. Lyons, "The open assembly model for the exchange of assembly and tolerance information: Overview and example," in *Proceedings of the DETC 2004 ASME Design Engineering Technical Conferences, pp. 1-19n*, 2004.

[19] S. C. Feng, H. Lee, C. Joung, T. Kramer, P. Ghodous, and R. Sriram, "Information model for disassembly for reuse, recycle and remanufacturing," *National Institute of Standards and Technology, 2011, NISTIR 7772, Gaithersburg, MD 20899, USA*, 2011.

[20] K.-S. Chin, Y. Zhao, and C. Mok, "Step-based multiview integrated product modelling for concurrent engineering," *Int. J. Adv. Manuf. Technol.*, vol. 20, no. 12, pp. 896–906, 2002.

[21] T. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," *International Journal of Human Computer, 43(5), November, pp. 907- 928*, 1995.

[22] W3C, *Owl 2 web ontology language primer (second edition)*, https://www.w3.org/OWL/, 2012.

[23] W3C, *Rdf resource description framework primer*, https://www.w3.org/RDF/, 2014.

[24] K.-Y. Kim, S. Chin, O. Kwo, and D. Ellis, "Ontology-based modeling and integration of morphological characteristics of assembly joints for network-based collaborative assembly design," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing. 23, pp. 71-88*, 2009.

[25] W3C. (May 2004). Swrl: A semantic web rule language combining owl and ruleml.

[26] S. Lemaignan, A. Siadat, J. Dantan, and A. Semenenko, "Mason: A proposal for an ontology of manufacturing domain," in *IEEE Workshop On Distributed Intelligent Systems: Collective Intelligence and its Applications, 195-200*, 2006.

[27] Z. Usman, R. Young, N. Chungoora, C. Palmer, K. Case, and J. A. Harding, "Towards a formal manufacturing reference ontology," *International Journal of Production research, 51(22), pp. 6553-6572*, 2013.

[28] F. Ameri and D. Dutta, "An upper ontology for manufacturing service description," in *ASME Conference Proceeding 2006. 10.1115/DETC2006-99600*, 2006.

[29] F. Ameri and C. Mcarthur, "An experimental evaluation of a rule-based approach to manufacturing supplier discovery in distributed environments," in *Proceedings of the ASME Design Engineering Technical Conference. 2. 10.1115/DETC2011-47768*, 2011.

[30] F. Ameri, C. Urbanovsky, and C. Mcarthur, "A systematic approach to developing ontologies for manufacturing service modeling," *CEUR Workshop Proceesings. 886*, 2012.

[31] A. Giovannini, A. Aubry, H. Panetto, M. Dassisti, and H. El Haouzi, "Ontology-based system for supporting manufacturing sustainability," *Annual Reviews in Control. 36. 10.1016/j.arcontrol.2012* 2012.

[32] H. Graves, *Ontology Engineering for Product Development*. OWLED, 2007.

[33] K. Wang and S. Tong, "An ontology of manufacturing knowledge for design decision support," *International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM 2008. 1 - 5. 10.1109/WiCom.2008.2510*, 2008.

[34] M. T. George and J. F. Steven, "Knowledge-based assistance for finite-element modeling," *IEEE Intelligent Systems, 11 (3) 23-32*, 1996.

[35] Q. Ma, Y. Wang, Y. Lv, X. Jin, and M. Zhou, "Ontology-based exchange of product data semantics between cad and cae," *International Journal of Multimedia and Ubiquitous Engineering Vol. 9, No. 12, pp. 293-306*, 2014.

[36] W. Sun, Q. Ma, and S. Chen, "A framework for automated finite element analysis with an ontology-based approach," *Journal of Mechanical Science and Technology 23, 3209-3220*, 2009.

[37] P. Benjamin, M. Patki, and R. Mayer, "Using ontologies for simulation modeling," in *Proceedings of the 2006 IEEE Winter Simulation Conference*, 2012.

[38] T. Ashiso and M. Fujita, "Definition of a web ontology for design-oriented material selection," *Data Science Journal, vol. 5, pp. 52-63*, 2009.

[39] Y. Zhang, X. Luo, Y. Zhao, and H.-C. Zhang, "An ontology-based knowledge framework for engineering material selection," *Advanced Engineering Informatics. 29. 10.1016/j.aei.2015.09.002*, 2015.

[40] D. WIJK, A. Etienne, E. GUYOT, B. Eynard, and R. Lionel, "Enabled virtual and collaborative engineering coupling plm system to a product data kernel," in *5th International Conference on Digital Enterprise Technology, Nantes, France, 22-24 October*, 2008.

[41] F. Karakoyun, "A methodology for holistic lifecycle approach as decision support system for closed-loop lifecycle management," *Dissertation*, 2015.

[42] D. Kiritsis, "Closed-loop plm for intelligent products in the era of the internet of things," *Computer-Aided Design, 43(5), 479–501*, 2011.

[43] M.-J. Yoo, C. Grozel, and D. Kiritsis, "Closed-loop lifecycle management of service and product in the internet of things: Semantic framework for knowledge integration," *Sensors (Basel), 16, 1053; doi: 10.3390/s1607105*, 2016.

[44] H.-B. Jun, and D. Kiritsis, "Several aspects of information flows in plm," *In L. Rivest, A. Bouras, and B. Louhichi (Eds.), Product Lifecycle Management. Towards Knowledge-Rich Enterprises SE - 2 (Vol. 388, pp. 14–24). Springer Berlin Heidelberg*, 2012.

[45] P. Zhan, "An ontology-based approach for semantic level information exchange and integration in applications for product lifecycle management," *Dissertation DOCTOR OF PHILOSOPHY, WASHINGTON STATE UNIVERSITY*, 2007.

[46] H.-B. Jun, D. Kiritsis, and P. Xirouchakis, "A primitive ontology model for product lifecycle meta data in the closed-loop plm," *729-740. 10.1007/978-1-84628-858-6-80*, 2007.

[47] A. Matsokis and D. Kiritsis, "An ontology-based approach for product lifecycle management," *Computers in Industry. 61. 787-797. 10.1016/j.compind.2010.05.007*, 2010.

[48] A. Matsokis and D. Kiritsis, "Ontology applications in plm," *Int. J. of Product Lifecycle Management, 5. 84 - 97. 10.1504/IJPLM.2011.038104*, 2011.

[49] P. Angelo and R. Subramanian, *Powder Metallurgy: Science, Technology and Applications, Second Editions*. PHI Learning Pvt. Ltd., New Delhi, 2009.

[50] *Powder Metal Technologies and Applications, Vol. 7, Second Edition*. ASM international, 1998.

[51] M. Delavaril, A. Salarvand, A. Rahi, and F. Shahri, "The effect of powder metallurgy process parameters on mechanical properties of micro and nano-iron powder," *International Journal of Engineering, Science and Technology, Vol. 3, No. 9, pp. 86-94*, 2011.

[52] R. Pareek and J. Bhamniya, "Optimization of injection moulding process using taguchi and anova," *Journal of Scientific and Engineering Research,4(1)*, 2013.

[53] M. Rathi and M. Salunke, "Analysis of injection moulding process parameters," *International Journal of Engineering Research and Technology (Vol. 1, No. 8)*, 2012.

[54] J. Zhu, J. Chen, and E. Kirby, "Tensile strength and optimization of injection molding processing parameters using the taguchi method," *The International Journal of Modern Engineering, 4 (2)*, 2004.

[55] R. Pal, S. Mukhopadhyay, and D. Das, "Optimization of microinjection molding process with respect to tensile properties of polypropylene," *Indian Journal of Fibre and Textile Research, 37(1), 11*, 2012.

[56] G. Pötsch and W. Michaeli, *Injection moulding: An introduction*. Munich: Hanser Publishers, 1995.

[57] J. Brydson, *Plastic materials*. Butterworth-Heinmann: Oxford, 1995.

[58] D. Rosato and M. Rosato, *Injection moulding handbook*. Massachusetts: Kluwer Academic Publishers, 2000.

[59] M. Huang and C. Tai, "The effective factors in the warpage problem of an injection-moulded part with a thin shell feature," *J. Mat. Proc. Tech., vol. 110, 2001, pp. 1–9*, 2001.

[60] M. Altan, "Reducing shrinkage in injection mouldings via the taguchi,anova and neural network methods," *Materials and Design. 31. 599-604. 10.1016/j.matdes.2009.06.049*, 2010.

[61] H. Li, Z. Guo, and D. Li, "Reducing the effects of weldlines on appearance of plastic products by taguchi experimental method," *The International Journal of Advanced Manufacturing Technology, 32(9-10), 927-931*, 2007.

[62] V. Bouchereau and H. Rowlands, "Methods and techniques to help quality function deployment (qfd)," *Benchmarking: An International Journal*, vol. 1, no. 7, pp. 8–20, 2000.

[63] L. Vanegas and A. Labib, "A fuzzy quality function deployment (fqfd) model for driving optimum targets," *International Journal of Production Research*, vol. 39, no. 1, pp. 99–120, 2001.

[64] L. Morris and L. Stauffer, "Design taxonomy for eliciting customer requirements," *Computers and Industrial Engineering*, vol. 27, no. 1, pp. 557–560, 1994.

[65] K. Rounds and J. Cooper, "Development of product design requirements using taxonomies of environmental issues," *Research in Engineering Design*, vol. 13, no. 2, pp. 94–108, 2002.

[66] A. Mousavi, P. Adl, R. Rakowski, A. Gunasekaran, and N. Mirnezami, "Customer optimization route and evaluation (core) for product design," *International Journal of Computer Integrated Manufacturing*, vol. 14, no. 2, pp. 236–243, 2001.

[67] D. McAdams, R. Stone, and K. Wood, "Functional interdependence and product similarity based on customer needs," *Research in Engineering Design*, vol. 11, no. 1, pp. 1–19, 1999.

[68] J. Spivey, *The Z Notation. A Reference Manual*. Prentice-Hall, Inc., 1989.

[69] R. Halligan, "Requirements analysis and specification writing," *Project Performance International*, 2012.

[70] W. Johnson, M. Feather, and D. Harris, "Representation and presentation of requirements knowledge," *IEEE Transactions on Software Engineering*, pp. 853–869, 1992.

[71] IBM, *Ibm engineering requirements management doors family*, `https://www.ibm.com/us-en/marketplace/requirements-management`, May 2019.

[72] P. Kroha, R. Janetzko, and J. Labra Gayo, "Ontologies in checking for inconsistency of requirements specification," *3rd International Conference on Advances in Semantic Processing, SEMAPRO*,

[73] P. Kroha and M. Rink, "Text generation for requirements validation," *Lecture Notes in Business Information Processing*,

[74] V. Castañeda, L. Ballejos, M. Caliusco, and M. Galli, "The use of ontologies in requirements engineering," *Glob J Res Eng. 10*, 2010.

[75] S. Lee and R. Gandhi, *Ontology-based Active Requirements Engineering*. 2005.

[76] S. Farfeleder, T. Moser, A. Krall, T. Stalhane, I. Omoronyia, and H. Zojer, "Ontology-driven guidance for requirements elicitation," *ESWC 2001, Part II, LNCS 6644*,

[77] I. Sommerville, *Software engineering (10th ed.)* Pearson Education, Inc.. ISBN: 978-0133943030, 2016.

[78] C. Kücherer, *Use of Domain Ontologies to Improve Requirements Quality*. REFSQ Workshops, 2017.

[79] R. Guizzardi, F.-L. Li, A. Borgida, G. Guizzardi, J. Horkoff, and J. Mylopoulos, "An ontological interpretation of non-functional requirements," *Frontiers in Artificial Intelligence and Applications. 267. 10.3233/978-1-61499-438-1-344*, 2014.

[80] G. Guizzardi, G. Wagner, R. de Almeida Falbo, R. Guizzardi, and J. Almeida, "Towards ontological foundations for the conceptual modeling of events," *in Conceptual Modeling, Springer*,

[81] H. Kaiya and M. Saeki, "Using domain ontology as domain knowledge for requirements elicitation," *186-195. 10.1109/RE.2006.72*, 2006.

[82] T. Avdeenko and N. Pustovalova, "The ontology-based approach to support the completeness and consistency of the requirements specification," *International Siberian Conference on Control and Communications, SIBCON 2015 - Proceedings. 10.1109/SIBCON.2015.71471842*, 2015.

[83] K. Siegemund, "Contributions to ontology-driven requirements engineering," *Dissertation, Technische Universität Dresden*, 2013.

[84] J. Lin, M. Fox, and T. Bilgiç, "A requirement ontology for engineering design," *Concurrent Engineering. 4. 10.1177/1063293X9600400307*, 1996.

[85] A. Weissman, S. Gupta, X. Fiorentini, S. Rachuri, and R. Sriram, "Formal representation of product design specifications for validating product designs," *National Institute of Standards and Technology(NIST), Gaithersburg, MD, NIST Interagency/Internal Rep. (NISTIR)7626*, 2009.

[86] T. Riechert, K. Lauenroth, J. Lehmann, and S. Auer, "Towards semantic based requirements engineering," *In: Proceedings of the 7th International Conference on Knowledge Management (I-KNOW)*, 2007.

[87] H. Kaiya and M. Saeki, "Ontology-based requirements analysis: Lightweight semantic processing approach," *223-230. 10.1109/QSIC.2005.46*, 2005.

[88] B. Glasgow, A. Mandell, D. Binney, L. Ghemri, and D. Fisher, "Mita, an information-extraction approach to the analysis of free-form text in life insurance applications," *AI Magazine 19, 59–71*, 1998.

[89] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach, 2nd Edition*. Prentice-Hall, Englewood Cliffs, NJ, 848–850, 2003.

[90] D. Embley, "Toward semantic understanding: An approach based on information extraction ontologies," *In: Proceedings of the 15th Australasian Database Conference (Australian Computer Society, Darlinghurst, Australia)*, 2004.

[91] B. Yildiz and S. Miksch, "Ontox-a method for ontology-driven information extraction," *In: Proceedings of the 2007 International Conference on Computational Science and its Applications, Springer, Berlin*, 2007.

[92] E. Oro and M. Ruffolo, "Towards a system for ontology-based information extraction from pdf documents," in *In: R. Meersman and Z. Tari (eds), Proceedings of the OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008. Part II on On the Move to Meaningful Internet Systems, Springer-Verlag, Berlin*, 2008.

[93] Z. li, M. C. Yang, and K. Ramani, "A methodology for engineering ontology acquisition and validation," *AI EDAM. 23. 37-51. 10.1017/S0890060409000092*, 2009.

[94] K. Verma and A. Kass, "Requirements analysis tool: A tool for automatically analyzing software requirements documents," in *In: Sheth, A. et al. (Eds.): Proceedings of ISWC 2008, LNCS 5318, pp. 751-763, Springer*, 2008.

[95] H. Saggion, A. Funk, D. Maynard, and K. Bontcheva, "Ontology-based information extraction for business intelligence," in *In Proceedings of the 6th international the semantic web and 2nd Asian conference on Asian semantic web conference, ISWC'07/ASWC'07, 843–856. Berlin, Heidelberg: Springer-Verlag*, 2007.

[96] U. Yasavur, R. Amini, C. Lisetti, and N. Rishe, "Ontology-based named entity recognizer for behavioral health," in *In Proceedings of the Twenty-Sixth FLAIRS Conference, AAAI Press*, 2013.

[97] S. Friedenthal, A. Moore, and R. Steiner, *A Practical Guide to SysML: The Systems Modeling Language (Third Edition)*. Morgan Kaufmann, 2015.

[98] Model Based Systems Engineering, *What's new in sysml 1.5 – requirements modeling*, `https://mbse4u.com/2017/05/09/whats-new-in-sysml-1-5-requirements-modeling/`, Jan. 2019.

[99] G. Lami, "Quars: A tool for analyzing requirements," in *TECHNICAL REPORT CMU/SEI-2005-TR-014 ESC-TR-2005-014*, 2005.

[100] K. Verma and A. Kass, "Requirements analysis tool: A tool for automatically analyzing software requirements documents," in *A. Sheth et al. (Eds.): ISWC 2008, LNCS 5318, pp. 751–763, Springer-Verlag Berlin Heidelberg*, 2008.

[101] V. Werneck, A. d. P. Oliveira, and J. Leite, "Comparing gore frameworks: I-star and kaos," in *In Ibero-American Workshop of Engineeringof Requirements, Val Paraiso, Chile*, 2005.

[102] T. Bures, P. Hnetynka, P. Kroha, and V. Simko, "Requirement specifications using natural languages," in *Technical Report D3S-TR-2012-05, Charles University*, 2012.

[103] M. F. Sriti, B. Eynard, P. Boutinaud, N. Matta, and M. Zacklad, "Towards a semantic-based platform to improve knowledge management in collaborative product development," 2006.

[104] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web, scientific american," 2001.

[105] E. Westkämper, "Life cycle management and assessment: Approaches and visions towards sustainable manufacturing (keynote paper)," *CIRP Annals-Manufacturing Technology, 49(2), 501-526.*, 2000.

[106] A. Weissman, S. K. Gupta, X. Fiorentini, R. Sudarsan, and R. D. Sriram, "Formal representation of product design specifications for validating product designs," *National Institute of Standards and Technology, NISTIR 7626*, 2009.

[107] A. Mavin, P. Wilkinson, A. Harwood, and M. Novak, "Easy approach to requirements syntax (ears)," *317 - 322. 10.1109/RE.2009.9*, 2009.

[108] G. Lami, "Quars: A tool for analyzing requirements," *Carnegie Mellon, Pittsburgh, Technical Report 2005-TR-014*, 2005.

[109] C. Lamar, *Linguistic Analysis of Natural Language Engineering Requirements*. 2009.

[110] W. Marquardt, J. Morbach, A. Wiesner, and A. Yang, *OntoCAPE: A Re-Usable Ontology for Chemical Process Engineering*. Springer Publishing Company, Incorporated, 2012.

[111] B. Zhu, "An information model in the domain of disassembly planning for sustainable manufacturing," *Dissertations - Dissertations - ALL. 536. https://surface.syr.edu/etd/536*, 2016.

[112] A. Goel, S. Rugaber, and S. Vattam, "Structure, behaviour, and function of complex systems: The structure, behaviour, and function modelling language," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 23(1), 23–35*, 2009.

[113] I. Tumer, R. Stone, and D. Bell, "Requirements for a failure mode taxonomy for use in conceptual design," in *Proceedings of the 2003 International Conference on Engineering Design, Stockholm, Sweden*, 2003.

[114] S. Ong, Q. Xu, Nee, and A. Y.C., "Design reuse in product development modeling, analysis and optimization," *Singapore [u.a.]: World Scientific, ZDB-ID 2185842-1. -Vol. 4*, 2008.

[115] Aerospace Standard AS9103, *Critical characteristics and key product characteristics (kc), published by sae, warrendale, pa usa.*

[116] ISO-10303-45, *Product data representation and exchange - part 45: Integrated generic resources: Materials, international organization for standardization (iso), geneva, switzerland.*

[117] IPC-1751A, *Generic requirements for declaration process management*.

[118] IPC-1752A, *Materials declaration management*.

[119] ISO-10303-21, *Product data representation and exchange - part 21: Implementation methods: Clear text encoding of the exchange structure, international organization for standardization (iso), geneva, switzerland*.

[120] R. Gupta and B. Gurumoorthy, "Unified taxonomy for reference ontology of shape features in product model," *IFIP Advances in Information and Communication Technology. 409. 295-307. 10.1007/978-3-642-41501-2-30*, 2013.

[121] A. Perzylo, N. Somani, M. Rickert, and A. Knoll, "An ontology for cad data and geometric constraints as a link between product models and semantic robot task descriptions," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) At: Hamburg, Germany*, 2015.

[122] ISO-10303-42, *Product data representation and exchange - part 42: Integrated generic resources: Geometric and topological representation, international organization for standardization (iso), geneva, switzerland*.

[123] M. Baysal, "Functional and behavioral product information representation and consistency validation for collaboration in product lifecycle activities," in *Ph. D. Dissertation, Syracuse University, Syracuse, NY.*, 2012.

[124] M. Baysal, M. Sarigecili, and U. Roy, "Functional and behavioral representation of product information for collaboration in product lifecycle," in *ASME International Mechanical Engineering Congress and Exposition, Proceedings, v 3, p 133-141, Proceedings of the ASME International Mechanical Engineering Congress and Exposition, Seattle, WA.*, 2007.

[125] U. Roy and O. Yaman, "Development of material information model to support the powder metallurgical product and processes," *Proceedings of NAMRI/SME, Vol. 42, 2014*, 2014.

[126] I. Stroud, *Boundary Representation Modeling Techniques (1st ed.)* London: Springer-Verlag, 2006.

[127] SQWRL, `https://github.com/protegeproject/swrlapi/wiki/SQWRL`, Accessed: 2020-02-05.

[128] W3C, *Owl 2 web ontology language document overview (second edition)*, `http://www.w3.org/TR/owl-overview`, Dec. 2012.

[129] Stanford Center for Biomedical Informatics Research (BMIR), *Protégé: A free, open-source ontology editor and framework for building intelligent systems*, `https://protege.stanford.edu/`, Jan. 2018.

[130] ARAS, *Aras product lifecycle management*, http://www.aras.com/, Accessed: 2019-09-20.

[131] OpenRefine, https://openrefine.org/, Accessed: 2020-05-10.

[132] spaCy, https://spacy.io/, Accessed: 2020-01-15.

[133] RASA NLU Trainer, https://rasahq.github.io/rasa-nlu-trainer/, Accessed: 2020-03-20.

[134] RASA NLU, https://rasa.com/docs/rasa/nlu/training-data-format/, Accessed: 2020-03-20.

[135] spaCy, https://spacy.io/usage/training#ner/, Accessed: 2020-01-01.

[136] O. Yaman and U. Roy, "Vibration-based structural health monitoring for safe and secure unmanned systems in closed-loop plm," 2020.

[137] I. Wright, "A review of research into engineering change management: Implication for product design," *Design Studies. Vol. 18 (1999), p. 33-42*, 1999.

[138] B. Hamraz, N. Caldwell, and P. Clarkson, "A holistic categorisation framework for literature on engineering change management," *System Eng. Vol. 16 (2013) No. 4, p. 473-505*, 2013.

[139] D.-B. Luh, Y.-T. Ko, and C.-H. Ma., "A structural matrix-based modelling for designing product variety," *Journal of Engineering Design. Vol. 22 (2011) No. 1, p. 1-29*, 2011.

# VITA

**Name of Author**: Omer YAMAN

**Place of Birth**: Alanya, Turkey

**Date of Birth**:04/25/1986

**Email**:oyaman@syr.edu

**Address**:Link 277 Syracuse University, Syracuse, NY 13244

## EDUCATION

01/2014-12/2020

**Doctor of Philosophy in Mechanical and Aerospace Engineering**

College of Engineering & Computer Science, Syracuse University

08/2012-12/2013

**Master of Science in Mechanical and Aerospace Engineering**

College of Engineering & Computer Science, Syracuse University

09/2007-02/2011

**Bachelor of Science in Manufacturing Engineering**

Faculty of Mechanical Engineering, Istanbul Technical University

09/2005-06/2009

**Bachelor of Science in Aeronautical Engineering**

Faculty of Aeronautics and Astronautics, Istanbul Technical University

## PROFESSIONAL EXPERIENCE

Teaching Assistant at Department of Physics, Syracuse University, 2019

Intern at UsPLM, Inc. (Syracuse, NY), 2018

## AWARDS & HONORS

Scholarship to study Master's and PhD in U.S.A (Ministry of National Education of Turkey), awarded to 250 students out of 20,000 applicants, 2011.

Honor Certificates, Istanbul Technical University, Aeronautical Engineering Department, 2006-2009.