École Doctorale IAEM Lorraine

ANNOTATIONS SEMANTIQUES POUR L'INTEROPERABILITE DES SYSTEMES DANS UN ENVIRONNEMENT PLM

(SEMANTIC ANNOTATIONS FOR SYSTEMS INTEROPERABILITY IN A PLM ENVIRONMENT)

Thèse

présentée et soutenue publiquement le 14-11-2013 pour l'obtention du

Doctorat de l'Université de Lorraine

(mentions: Automatique, Traitement du Signal et des Images, Génie Informatique, CNU 61ème section, et Informatique, CNU 27ème section)

par

Yongxin LIAO

Composition de jury :

<u>Président :</u>	
M. David CHEN	Professeur à l'Université de Bordeaux 1
<u>Rapporteurs</u> :	
M. John KROGSTIE	Professeur à Norwegian University of Science and Technology (Norvège)
M. Lionel ROUCOULES	Professeur à l'ENSAM Aix-en-Provence
<u>Examinateurs</u> :	
M. Osiris CANCIGLIERI JUNIOR	Professeur à l'Université Pontificale Catholique du Paraná (Brésil)
M. Hervé PANETTO	Professeur à l'Université de Lorraine (Directeur de thèse)
M. Nacer BOUDJLIDA	Professeur à l'Université de Lorraine (Directeur de thèse)
M. Mario LEZOCHE	Maître de Conférences à l'Université de Lorraine (Encadrant de thèse)

Centre de Recherche en Automatique de Nancy, UMR 7039, Université de Lorraine, CNRS
Laboratoire Lorrain de Recherche en Informatique et ses Applications, UMR 7503, Université de Lorraine, CNRS, INRIA
Fédération Charles Hermite, FR 3198, Université de Lorraine, CNRS

Acknowledgement

My deepest gratitude goes first and foremost to my supervisors: Prof. Hervé Panetto and Prof. Nacer Boudjlida. Prof. Hervé Panetto, with an enthusiastic smile on his face all the time, guided me through the development of all this doctoral research work and helped me a lot to clarify the research issues and gave me big instructions in the research. Prof. Nacer Boudjlida who gave me lots of useful advices and illuminating suggestions in each discussion we made. He enlightened me with wise ideas that enable me to overcome obstacles in my research. I would like to give my thanks to Dr. Mario Lezoche for his productive collaborations and inspiring encouragement during all these three years. Not only did he help me all the way through in the organization and modification in those papers but also he gave me in lots of helps in my life. Their professional working attitudes have greatly influenced me, from which I gain the benefits for my future study and also for my life.

I also would like to express my heartfelt thankfulness to the reporters of my thesis: Prof. Lionel Roucoules and Prof. John Krogstie. They gave me many valuable comments and effective advices that helped me to discover the existing issues I have been omitted during my research and possible directions that can be improved in my thesis.

In these three years PhD research, I have grown and gained a lot. I would like to give my sincere gratitude to the CRAN laboratory for providing me such a good environment and platform to implement all my work. I also owe my full thanks to the financial supports of my PhD: the “Charles Hermite Research Federation” in Nancy and the “Région Lorraine” local government.

I am so glad to have a group of knowledgeable and warm-hearted colleagues and friends who shared with me the whole or part of the PhD path. In alphabetical order by name: Alexis , Alexandre, Antonio, Benjamin, David, Eduardo, Esma, Fabien, Fengwei, Gabriela, Jérémy, Jingwen, Jun, Kun, Loana, Pierre, Pascale, Phuc, Thomas, William, Yong and so on. Particularly, I would like to thank Dr. Eduardo Loures, who is a constructive and patient person. We had lots of worthwhile discussions during the year that he passed in CRAN. I would like to thank Dr. Pierre Cocheteux for his guide in CATIA software and his generous help to my family and me. Many thanks go to Dr. Alexis Aubry and Dr. David Gouyon for their helps on the Sage X3 and Flexnet software. I am also greatly indebted to Prof. Arvid Perego, for the mathematical advices in the formalization part of my thesis.

Last my awesome gratitude goes to my family. I am thankful to my parents for their loving considerations and great confidence in me all through these years. Thanks are due to my beloved wife Xiaoyun You, from whom I benefited so much for her enormous love and support anytime and anywhere. Also, thanks to my lovely son Youqi Liao for bringing me happiness, surprises and energies, which really change the life of my wife and me.

Contents

List of Figures

List of Tables

Chapter 1 Introduction and Context

1.1 Introduction

This section provides an overview of the thesis, which begins with introducing the motivation of this research work (Section 1.1.1), specifying the research problems and posing five main research questions (Section 1.1.2). An overview of the solution that we proposed is presented in Section 1.1.3. Finally, Section 1.1.4 describes the contributions of this research work and Section 1.1.5 provides an outline of the thesis.

1.1.1 Motivation

In manufacturing enterprises, the Product Lifecycle Management (PLM) approach has been considered as an essential solution for improving the product competitive ability. It aims at providing a shared platform that brings together different enterprise systems at each stage of a Product Life Cycle (PLC) in or across enterprises [1]. Although the main software companies are making efforts to create tools for offering a complete and integrated set of systems, most of them have not implemented all of the systems. Finally, they do not provide a coherent integration of the entire information system. This results in a kind of "tower of Babel", where each application is considered as an island in the middle of the ocean of information, managed by many stakeholders in an enterprise, or even in a network of enterprises. The different peculiarities of those stakeholders are then over increasing the issue of interoperability. The objective of this thesis is to deal with the interoperability problems, mainly the issue of semantic interoperability, by proposing a formal semantic annotation method to support the mutual understanding of the semantics inside the shared and exchanged information in a PLM environment.

1.1.2 Research Problems and Questions

The concept of Product Life Cycle (PLC) has been revealed for more than sixty years [2]. It describes every stage of a product of interest (such as imagining, defining, realising, using/supporting and retiring/disposing of). In the meantime, along with the advent and the evolution of Computer Aided Design (CAD) systems, the problems of locating the required data and losing control of change process associated with these data have gradually appeared [2]. As a solution, Product Data Management (PDM) systems have been developed and introduced for supporting easy, quick and secured access to valid data during the product design phase [1]. However, as it is pointed out in [3], the data produced by CAD systems do not cover all the information that related to the whole product life cycle (from the requirement specification to dismantling information). The PDM systems are not able to give enough support for non-engineering data. In order to fill this gap, during the 1990s, the PLM solution is proposed to support the processes of capturing, representing, retrieving and reusing both engineering and non-engineering aspects of knowledge along the entire product life cycle. It intends to facilitate the knowledge management in or across enterprises [1]. Therefore, the knowledge concerning the product life cycle, which we named PLC-related knowledge, has become one of the critical concepts in a PLM solution.

Knowledge is an awareness of things that brings to its owner the capability of grasping the meaning from the information [4]. This definition is included in a more structured presentation of the DIKW Pyramid [5], a hierarchical model for representing the structural relationships between Data, Information, Knowledge and Wisdom (DIKW). In this research work, we consider knowledge as a kind of intangible thing that is only explicit to its owner but remains tacit to the external world [6]. One of the main purposes of knowledge management is to make knowledge accessible and reusable [7]. Knowledge Representation is the result of embodying the knowledge from its owner's mind into some explicit forms. It gives a possibility for external entities to perform some specific operations for achieving their particular needs. Knowledge representations act as the carriers of knowledge to assist collaboration activities.

Interoperability serves as a foundational role to support collaboration. It is the ability that diverse entities can exchange knowledge representations and make use of those knowledge representations that they have exchanged. Five possible levels of interoperability have been categorized by Euzenat [8]: encoding level, lexical level, syntactic level, semantic level and semiotic level. While encoding, lexical and syntactic issues are now can be formally solved by many technical standards, enabling a seamless semantic interoperability remains a huge challenge [9]. In order to cope with the semantic interoperability issue, two important obstacles still need to be overcame:

(1) The implicit semantics that is necessary for understanding a knowledge representation that is not made explicit;
(2) The lack of mechanisms to verify the correctness of explicit semantics in the exchanged knowledge representation.

A mutual understanding of the semantics inside the shared and exchanged knowledge representations is the cornerstone in the quest for semantic interoperability. Due to the essence of ontology, which is a kind of common agreement on the conceptualization of terms in a specific domain of interest, semantic annotations [10] are usually considered as a possible solution to deal with these two obstacles.

In this context, semantic enrichment is considered as a process that makes any implicit semantics more explicit through the use of semantic annotations. Some research questions are then emerging from the needs of semantic enrichment in a PLM environment:

(1) What are the semantic interoperability problems that exist during the cooperation in a PLM environment?

(2) What kinds of knowledge representation in a PLM environment need semantic enrichment?

(3) What kinds of ontology can be used to support the semantic enrichments of those knowledge representations?

(4) What are the essential elements of a semantic annotation and how to formally represent a semantic annotation in a suitable format?

(5) How to semantically enrich a knowledge representation and how can these enriched semantics contribute to the semantic interoperability in a PLM environment?

1.1.3 The Proposed Solution

Compared with the other types of annotations, a semantic annotation has two important features: (1) it can be read and processed by a machine [10]; (2) it contains a set of formal and shared terms in a specific context [11]. As a way to realize the semantic enrichment, semantic annotations use ontology to capture annotator's knowledge and then act as a knowledge carrier to enrich annotated object's semantics. It can then be widely used in many contexts for various purposes. In this work, there are two important aspects of the semantics that are made explicit by a semantic annotation:

(1) The domain semantics, which describes the context and the meaning of an annotated object in a specific domain;

(2) The structure semantics, which describes the interrelations between the annotated objects and the other objects related to them in a knowledge representation.

Based on the survey and exploration of current semantic annotation researches, three shortcomings have been identified:

(1) The formalization of semantic annotations is not the focus of most of the semantic annotation researches. They only considered the semantic annotation as a kind of "is a" association between one annotated object and one ontology

concept. Even if there are some specific formalizations, they are difficult to be reused in other researches but the studied ones;

(2) The domain semantics of the annotated objects is the only concern in most of the semantic annotation researches, where the structure semantics is ignored, or vice-versa;

(3) In most of the semantic annotation researches, there is lack of mechanism to support the inconsistency detection of the semantic annotations and the conflict identification between the annotated objects;

Therefore, we focus our research work on: (a) clearly identifying the essential elements of a semantic annotation by proposing a formalization that can be used to enrich different types of models; (b) proposing two mechanisms to detect the possible inconsistencies of semantic annotations and to identify possible conflicts between the annotated objects for facilitating and assisting the knowledge management in a PLM environment.

Based on this research focus, in this thesis, a semantic annotation is used as a way to employ one or several ontologies for making explicit both structure and domain semantics of an annotated target knowledge that needs to be made explicit. For this reason, a semantic annotation is considered as semantic relationships between the Target Knowledge Representations (TKRs) and the Ontology-based Knowledge Representations (OKRs). These relationships are formally defined in a Semantic Annotation Structure Model (SASM).

TKRs are the targets of semantic enrichment, namely the targets that semantic annotations are attached to. They contain implicit or possibly ambiguous explicit semantics, which is not easily intelligible. In a PLM environment, all the different types of models throughout the entire PLC are considered as embedding elements of TKRs (different kinds of modelling constructs represent different kinds of knowledge): such as data models, process models, state models, resource models, decision models.

OKRs are the ontologies that capture different aspects of knowledge and provide the common and shared conceptualizations for supporting the semantic enrichment of TKRs. In this research work, two types of ontologies are used: (1) The PLC-related ontologies that formalize the domain semantics of TKRs. They are normally collected by domain specialists and formalized by knowledge engineers. (2) The meta-model ontologies that formalize the structure semantics of TKRs. They are normally generated by modelling language experts.

The SASM contains a set of definitions that formalize the structure of a semantic annotation. It acts as a bridge to formally describe the semantic relationships between the TKRs and the OKRs. It identifies the essential elements of the semantic annotation, which can be used as the basis for developing the common semantic annotation schema for different kinds of models along the product life cycle. Differently from other semantic annotation proposals, we propose to annotate an object in a TKR with a set of selected ontology elements. These elements belong to one or more OKRs, which describe the domain and structure semantics of the annotated objects.

Moreover, in order to detect the inconsistencies between the semantic annotations and to identify the conflicts between the annotated objects, we proposed a semantic annotation suggestion mechanism and two verification mechanisms that rely on the SASM. The former verification is based on the domain semantics comparisons of two or more related semantic annotations of a common annotated object. The later verification is based on the results of the former verification. Both structure and domain semantics, which are made explicit by semantic annotations, contribute to these three mechanisms. The domain semantics acts as the data that are used for similarity comparison. The structure semantics supports the creation of reasoning rules that are used for the inference.

Finally, in order to apply the proposed solution into a PLM environment, a semantic annotation framework, which is composed of a general semantic annotation procedure and the architecture of the framework, are proposed.

1.1.4 The Contributions

This thesis presents a formal semantic annotation method that supports the semantic enrichment of models in a PLM environment for facilitating semantic interoperability issues. The contributions have been published in four papers ([12], [13], [14] and [15]), which are listed as follows:

(1) We surveyed the literature that relates to the Product Lifecycle Management, System Interoperability and Semantic Annotation. A comparison of some current semantic annotation researches is made for identifying the exiting drawbacks and potential challenges ([12], [13], [14] and [15]).

(2) We proposed a formalization of semantic annotations that can be used as a basis to create a semantic annotation schema for supporting the semantic enrichment of models ([12], [13], [14] and [15]).

(3) We proposed a semantic annotation suggestion mechanism and two verification mechanisms to support inconsistency detection between semantic annotations and conflict identification in model contents ([15]).

(4) We proposed a guideline that contains a number of procedures to guide an

engineer in applying the proposed solution and a semantic annotation framework for enriching semantics of models in a PLM environment([14] and [15]).

(5) We designed, implemented and validated a prototype annotation tool for the semantic annotation of process models based on the proposed formalization, annotation procedures and framework ([15]).

1.1.5 Thesis Outline

The research method is inspired by design science [16], and it is performed in five steps: context of the research (Chapter 1) and problem identification (Chapter 2), solution proposal (Chapter 3), prototype implementation (Chapter 4) and solution Validation (Chapter 5). Figure 1-1 shows the logical structure of the thesis and the interconnections among the chapters and the sections (besides the summary sections at the end of each chapter). In order to refine the contents of this figure, the concept that represents the main task or focus of each section is added to the corresponding name section in the figure. More specifically, the details of the thesis structure are illustrated as follows:

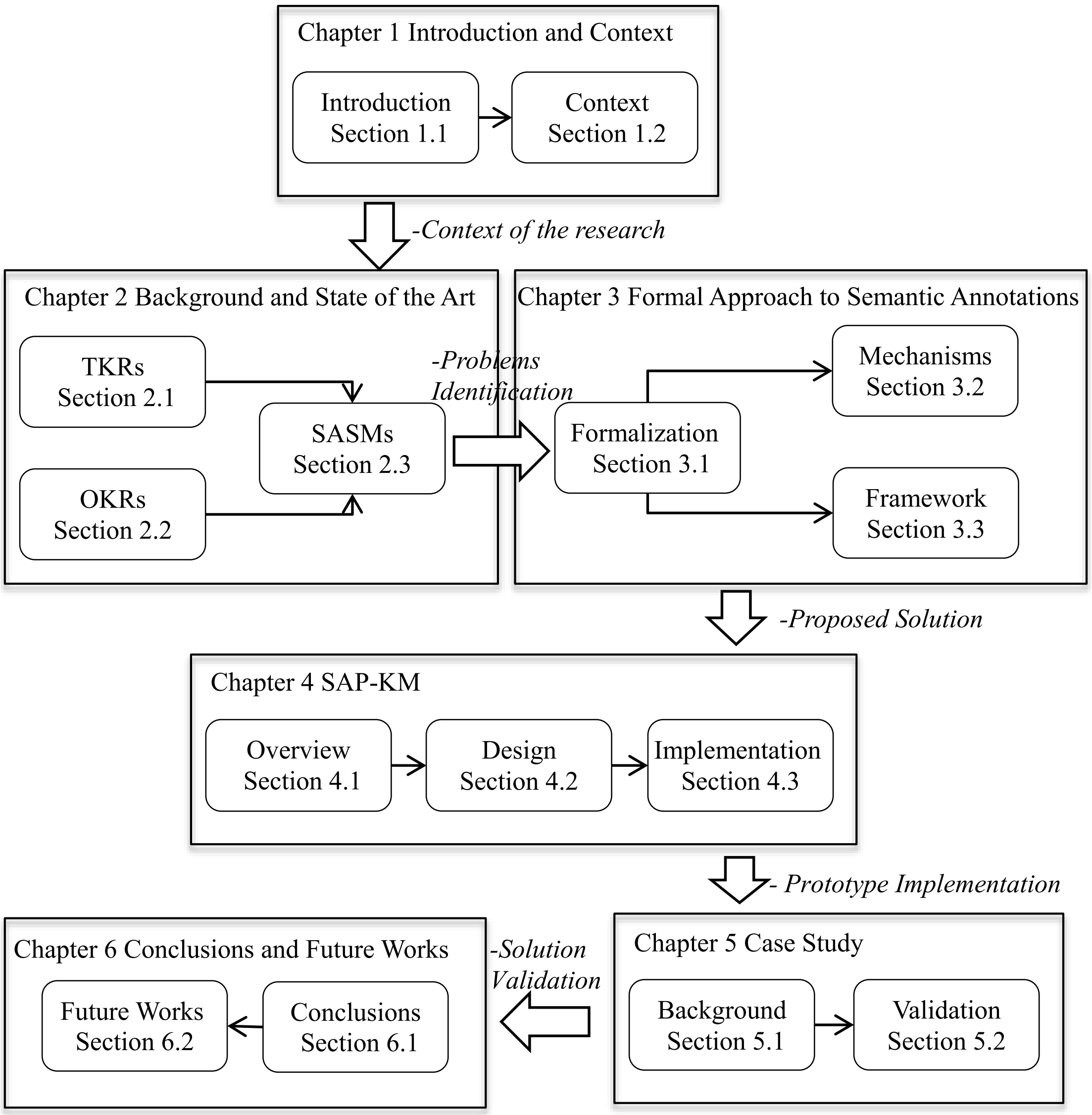


Figure 1-1 The Logical Structure of the Thesis

Chapter 1, *Introduction and Context*, first explains the motivation, states the problems and research questions, gives a brief overview of the proposed solution, and presents our main contributions in Section 1.1 (*Introduction*). The remaining of the current chapter (*Context* - Section 1.2) gives a literature review of the concepts about the Product Life Cycle, Product Lifecycle Management, Knowledge Management and System Interoperability to determine the domain of concern and proposes three postulates to support this research work.

Chapter 2, *Background and State of the Art*, is divided into three parts based on the three components of a semantic annotation. Firstly, this chapter surveys models in a PLM environment to investigate the TKRs that can be semantically enriched (*TKRs* - Section 2.1). Then it surveys ontology languages and existing ontologies to discover the OKRs that can be used to support the semantic enrichments (*OKRs* - Section 2.2).

Finally, it investigates the current semantic annotation researches from different domains and purposes to address those existing drawbacks and potential challenges (*SASM*- Section 2.3).

Chapter 3, *Formalization of semantic annotations*, presents the proposed solution to deal with the issues and drawbacks that are identified in the previous chapters. It first gives the details of our semantic annotation formalization proposals and identifies the essential elements of a semantic annotation (*Formalization* - section 3.1). Then the mechanism for the suggestion of semantic annotations, the detection of inconsistencies between semantic annotations and the identification of possible conflicts in a model are presented (*Mechanisms*- section 3.2). Finally, a semantic annotation framework that defines the semantic annotation procedure and the overall architecture are proposed (*Framework* - Section 3.3).

Chapter 4, *SAP-KM (Semantic Annotation Plug-in for Knowledge Management)*, instantiates the formalization of semantic annotations and demonstrates the applicability and usability for applying the proposed solution into real-life applications. This chapter begins with an overview of the requirement specifications and the prototype development environment (*overview* – section 4.1). Then the details of the design (section 4.2) and implementation (section 4.3) of the prototype annotation tool for the semantic enrichment of a process model are presented.

Chapter 5, *Case Study*, demonstrates how this proposal can be applied in order to contribute to the semantic interoperability in a PLM environment. Firstly, this chapter introduces the life cycle of a chosen product and presents the selected application scenario (*Background*- section 5.1). Then, in Section 5.2 (*Validation*), based on the semantic annotation procedure that is presented in Chapter 3, the proposed solution is applied in that particular application scenario.

Finally, Chapter 6, *Conclusions and Future Works*, gives a conclusion that summarizes this thesis, discusses the limitations and advantages of the work done, and lists a set of possible research directions that can be addressed in future works.

1.2 The Context

This section presents the context of the research. It starts with an introduction of the product life cycle (Section 1.2.1) and the product lifecycle management (Section 1.2.2) that helps us to determine the application domain. We then discuss the concepts about knowledge, knowledge representation and knowledge management in Section 1.2.3 to define an unambiguous semantics of these terms that are used along the remaining of the thesis. The issues of systems interoperability, especially the semantic interoperability, in a PLM environment is presented in Section 1.2.4. At the end, three postulates are proposed in Section 1.2.5 to support the proposed solution.

1.2.1 Product Life Cycle

The concept of the Product Life Cycle (PLC) has been introduced since the 1950s [17], and it is a biological metaphor that describes every phase a product goes through, from the first initial requirement until it is retired and disposed of [18]. However, based on different points of view and research concerns, a PLC model can be represented in the following two ways.

From the marketing point of view, a PLC is used to describe the unit sales curve for a product that extends from the time [17]. Birou et al. categorized a PLC model into five major stages [19], which are graphically described in the Figure 1-2 (a): (1) *Design stage*, which includes development and test marketing of a product; (2) *Introduction stage*, which begins with the full-scale lunch of a product into the market; (3) *Growth stage*, in which, unit sales grow rapidly and reach a relative peak; (4) *Maturity stage*, in which, unit sales may continue slowly increase until it decreases; (5) *Decline stage*, which commences with the long-run decline of unit sales. However, there are some researches that only make use of the last four stages and ignore the design stage. Meanwhile, a six-stage PLC is used by others through adding the *abandonment stage* at the end.

From the production point of view, as shown in the Figure 1-2(b), a PLC also can be classified into five main phases [18]: (1) *Imagination phase*, in which, a product only exists as an idea in human's mind; (2) *Definition phase*, in which, the idea of product is formulated by various kinds of description; (3) *Realisation phase*, in which, an actual product is manufactured following the description;(4) *Using and Supporting phase*, in which, a product is used by a customer and benefits the supports from the enterprise; (5) *Retiring and Disposing of phase*, in which, a product is no longer used by a customer and needs to be recycled or disposed of. In fact, this categorization is at a high abstraction level. Actual PLC models are always represented in a more complete way through extending more details in one or several of these phases.

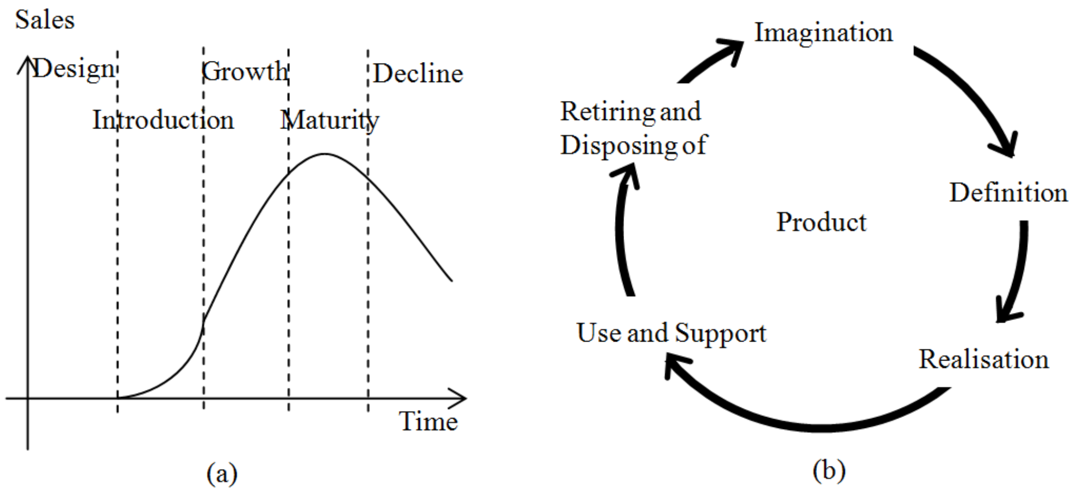


Figure 1-2 Product Life Cycle Models of two Research Concerns

Indeed, both of these two models can be used to represent the life cycle stages (phases) of a product. However, based on different research focuses, they should be distinguished from each other when they are being instantiated. Due to the fact that the scope of our research focuses more on the production knowledge management, the second PLC model is more appropriate to describe our context.

1.2.2 Product Life Management

The Computer Aided Design systems appears in the early 1980s, along with its evolution, the problems of locating the required data and losing control of change process associated with these data become increasingly intense [2]. The needs of easy, quick and secure access to valid data during the product design phase became the primary motivation to the development of a Product Data Management (PDM) solution [1]. However, due to the limited scope and the initial design of PDM solution, it is usually restricted to handling the product data in the engineering domain, but it remains inadequate with the non-engineering data, such as sales, planning, after sale services and so on. To be more specific, unlike the comprehensive supports to Computer Aided Design (CAD), Computer Aided Engineering (CAE), Computer Aided Process Planning (CAPP) and Computer Aided Manufacturing (CAM), PDM solutions cannot provide all the necessary supports to Enterprise Resource Planning (ERP), Supply Chain Management (SCM) and Customer Relationship Management (CRM).

In order to further extend the functionalities of a PDM solution and to fill the gap between the PDM proposal and the enterprise business activities, during the 1990s, the concept of Product Lifecycle Management (PLM) is proposed. Different from a PDM solution that only focuses on managing product data, a PLM solution focuses on managing all the PLC-related knowledge throughout the different phases of the PLC [1]. It aims at providing a shared platform for facilitating the process of capturing, representing, organising, retrieving and reusing the knowledge concerning the related product in or across enterprises, and to provide the integration strategies and technological supports to bring together all existing enterprise systems that dealt with the product [20].

More and more enterprises adopted the PLM solutions and discovered the benefits for their complex engineered products in the last decade. According to the market research in IT enterprises, PLM became one of the fastest growing markets and the total revenues of PLM in 2006 is projected to increase by $ 5.5 billion compared with the corresponding period in 2001 [1]. Presently, an increasing number of commercial PLM solutions have been developed, for example, to mention only a few, Agilie PLM solutions[1], Siemens PLM Software[2], Arena PLM solution[3], SAP PLM[4], PTC Windchill[5]. Based on their functions, the existing PLM solutions can be classified into three groups [21]:

(1) *Information management*, which provides methods to identify, structure, store, retrieve and share product, process and project-related data;

(2) *Process management*, which provides methods for modelling and operating formal and semi-formal processes;

(3) *Application integration*, which defines and manages the interfaces between the PLM platform and the variety of enterprise systems (such as CAD, CAM, CAE, ERP, MES, CRM, etc.).

Though, all existing PLM solutions try to propose an efficient and powerful collaboration environment for the variety of enterprise systems, they are still obstructed by various kinds of issues. From the collaboration point of view, due to multiplicity of formats, standards and versions, Ball et al. considered the information sharing and exchange as one of the main challenges in PLM [20]. From the implementation point of view, CIM data concluded that the cost, the quality, the time-to-market and the innovation are the four main challenges for a PLM solution [2]. Hewett indicated six main directions for improving the current PLM solutions: data exchange, design collaboration, enterprise-centric view, scale to reality, standard and technique for engineering processes, information and knowledge representation [22]. Among all

[1] Agilie PLM solutions: http://www.oracle.com/us/products/applications/agile/index.html
[2] Siemens PLM Software: http://www.plm.automation.siemens.com/
[3] Arena PLM solution: http://www.arenasolutions.com/
[4] SAP PLM: http://www.sap.com/france/solutions/business-suite/plm/index.epx
[5] PTC Windchill: http://www.ptc.com/product/windchill/

these issues, one of the main drawbacks of existing solutions draws our attention: they are mainly focusing on dealing with the syntax but rarely the semantics of the objects that are produced, transformed, exchanged during the PLC. The purpose of this research is to propose a way for assisting the mutual understanding of the semantics that embedded inside the shared and exchanged objects for further supporting the knowledge management processes in the context of PLC.

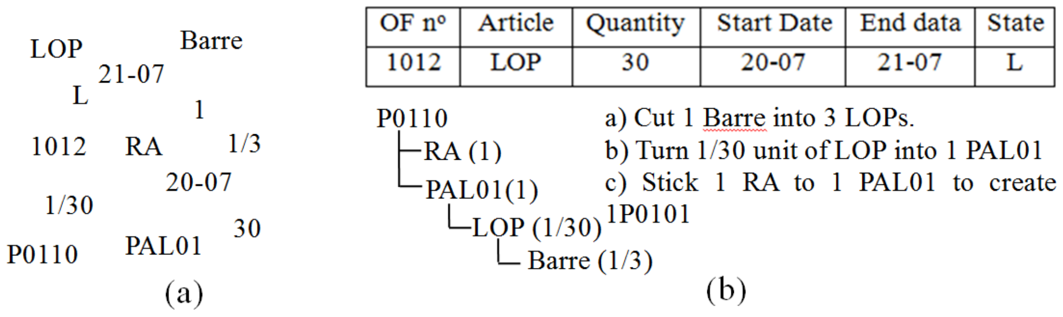
1.2.3 Knowledge Management

1.2.3.1 Data, Information, Knowledge and Wisdom

Making clear the definitions of data, information, knowledge and wisdom and the distinctions among these four basic concepts is an important step before proceeding to the introduction of knowledge management.

Data describe the unorganized and unprocessed facts or statistics [23], which is the essential foundation for information and knowledge. Data has no significance by itself without the related context. It can be gathered or invented by observers and described in any forms, for example, as shown in the Figure 1-3 (a), "LOP", "1/30", "PAL01", "Barre" and "3" are data.

Information represents a set of particular organized and processed data in a given context [24], which tries to express some meanings (semantics) in an appropriate way. Usually, it can only be understood by the one who has the relevant knowledge. Figure 1-3 (b) shows some simple instances of information, such as the manufacturing order of LOP, the bill of material of P0101 and a set of steps of a manufacturing process.



OF n°	Article	Quantity	Start Date	End data	State
1012	LOP	30	20-07	21-07	L

a) Cut 1 Barre into 3 LOPs.
b) Turn 1/30 unit of LOP into 1 PAL01
c) Stick 1 RA to 1 PAL01 to create 1P0101

Figure 1-3 Examples of Data and Information

Knowledge is an awareness of things that brings to its owner the capability of grasping the meanings (semantics) from the information [4]. It is obtained through the learning behaviours, in which, the external information from the real world is sublimated. In this work, knowledge is considered as a kind of intangible thing, which has to be made perceptible and afterward to be expressed under multifaceted forms of representations.

Wisdom is considered as a higher level of comprehension above knowledge [24][4]. It is defined as the quality to deal with (process, inference or reasoning) the existing knowledge and produce an appropriate new knowledge that can be regarded as the answers to some corresponding problems.

A so-called DIKW Pyramid (Hierarchy) is proposed by Zeleny and Ackoff [4] [5] (Figure 1-4), which refers to a hierarchical model for representing the structural relationships between Data, Information, Knowledge and Wisdom.

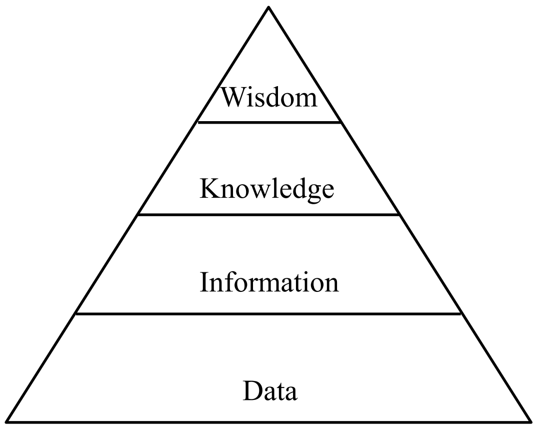


Figure 1-4 DIKW Pyramid

Nunamaker et al adopted these four concepts and used them to describe the gradual increase in levels of understandings [25]: (1) *Data level*, which signifies the meanings of symbols, in the context where they are collected, is completely understood; (2) *Information level*, which indicates the relationships between symbols, related to the context where they are presented, is being understood; (3) Knowledge level, which signifies patterns hidden in information is being understood; (4) *Wisdom level*, which indicates the principles embedded in knowledge is being understood.

1.2.3.2 Knowledge Dimension

Indeed, in the knowledge management area, there is an extensive literature devoted to debate the essential of knowledge. However, due to the research scope of this thesis, we are not going to give a comprehensive overview about it, but only to discuss the major issues that help us to clearly determine the important concepts.

As described by Polanyi in his book “The tacit dimension” [26], he began to reconsider personal knowledge from the fact that “we can know more than we can tell”. As a result, the part that is unable to be told is named as tacit knowing (knowledge), which acts as the core focus in his quest for the essential of personal knowledge.

Because of the different interpretations of Polanyi’s theory, the concepts of tacit knowledge and explicit knowledge have evolved and have been expressed with various definitions. For example, Nonaka [27] adapted Polanyi's theory to discover the essential elements in the organizational knowledge creation. In his research [27], two dimensions of knowledge are defined as follows:

- The Tacit Knowledge "*has personal quality, which makes it hard to formalize and communicate*".
- The Explicit Knowledge "*refers to knowledge that is transmittable informal, systematic language*".

Based on the assumption that knowledge is created through conversion between tacit and explicit knowledge, as shown in the Figure 1-5, he proposed a four modes of organizational knowledge conversion: (1) *Socialization* enables the conversion through the interactions between individuals, for example, shared experiences; (2) *Externalization* converts tacit knowledge into explicit knowledge through information creation; (3) *Combination* supports the conversion through social process, such as meetings; (4) *Internalization* converts explicit knowledge into tacit knowledge through organizational learning.

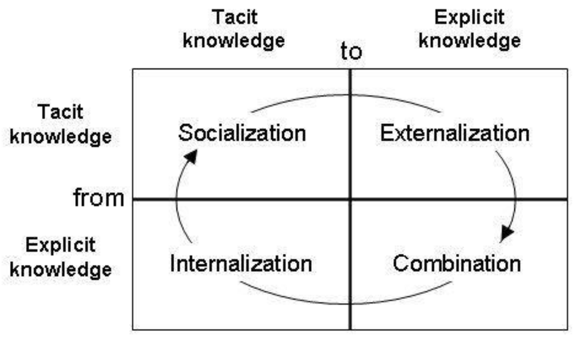


Figure 1-5 Four Modes of the Organizational Knowledge Creation [27]

Although Nonaka's organizational knowledge creation theory is the most widely accepted and referenced description about tacit and explicit knowledge, some researchers [28] [29] still do not want to accept his knowledge conversion proposition. With respect to the original definition that is proposed by Polanyi, they considered the tacit knowledge as knowledge that cannot be expressed externally. So as to overcome the gap between Nonaka and Polanyi, another concept so-call "implicit knowledge" is proposed as a third type of knowledge by Kim et al [28], in which, implicit knowledge is considered as a knowledge that can be externalized when it is needed but currently it is not. Even more, this concept is used as a substitute for the concept of tacit knowledge in some other researches [30].

According to our knowledge, the different research objects of Polanyi and Nonaka cannot be compared with each other directly. Because Polanyi's object is human knowledge and Nonaka's object is organizational knowledge. Furthermore, Polanyi does not state that the tacit knowledge could not be transferred, but he suggests no method on how this kind of knowledge could be transmitted [31]. Further, as a matter of fact, the boundary between tacit knowledge and explicit knowledge is ambiguous in practical [32]. Even Polanyi does not declare which things are known tacitly or explicitly. According to the main idea that Polanyi used to define the concept of knowledge, all knowledge is rooted in tacit knowledge [26]. Therefore, in order to avoid above misinterpretations, we argue that knowledge is an internal awareness that is only explicit to its owner but remains tacit to the external world.

One of the main purposes of knowledge management is to make knowledge accessible and reusable [7]. Knowledge representation is the result of embodying the knowledge from its owner's mind into some explicit forms. It gives a possibility for external entities to perform some specific operations for achieving their particular needs. More specifically, in order to clearly characterize the nature of a knowledge representation, Davis et al. [33] proposed in terms of five important and distinct roles that it plays: (1) It is a substitute of the knowledge itself; (2) It is a set of ontological commitments, which determines the terms to describe the world; (3) It is a partial theory of intelligent reasoning; (4) It is a medium for the efficient computation; (5) It is a medium of human expression and communication. Furthermore, we specify that each knowledge representation contains two kinds of semantics: (1) explicit semantics, which is directly expressed in the knowledge representation; (2) implicit semantics, in opposite, which is hidden.

Therefore, in a PLM environment, we consider that all the relevant resources produced by different stakeholders through the variety of enterprise systems are all knowledge representations. Several examples are shown in the Figure 1-6, such as requirement document, product design model, control interface design, process model, data model and observation video. They act as the carriers of the stakeholders' knowledge to assist the collaboration activities.

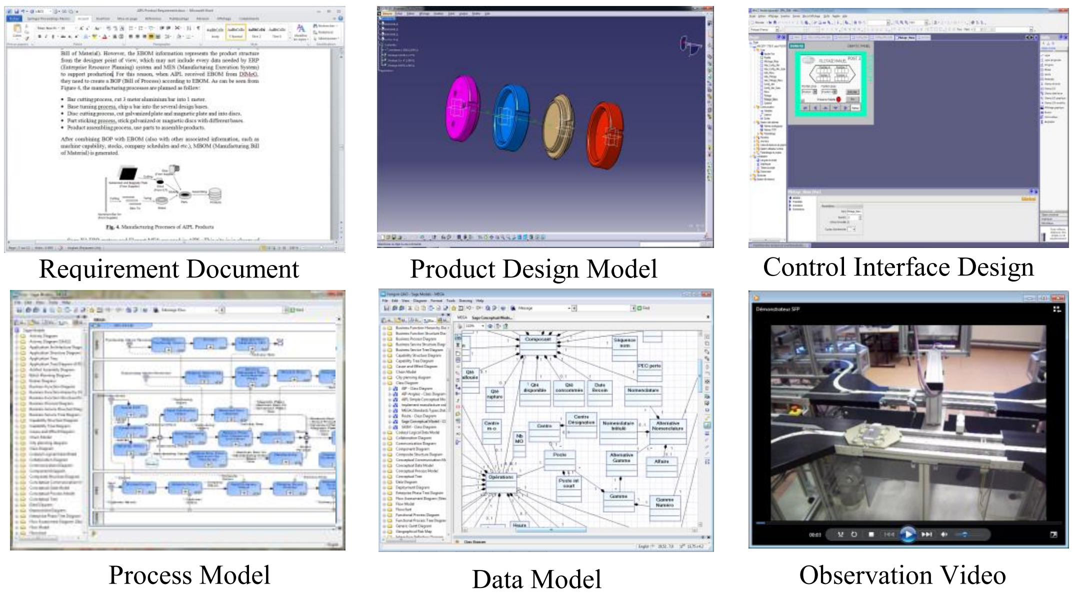


Figure 1-6 Examples of the Knowledge Representation in a PLC

1.2.3.3 Knowledge Management

Knowledge management has gained a lot of attentions and is widely adopted by a large number of organizations in the last decade [34]. However, the normative literature is not able to achieve an agreement on the definition of the knowledge management [35][36]. For example, Quintas et al. explained it as a "*process of continually managing knowledge of all kinds to meet existing and emerging needs, to identify and exploit existing and acquired knowledge assets and to develop new opportunities*" [37]; NASA knowledge management team defined it as an approach of "*getting the right information to the right people at the right time, and helping people create knowledge and share and act upon information in ways that will measurably improve the performance of NASA and its partners*" [38]; Schultze and Leidner suggested that "*knowledge management is the generation, representation, storage, transfer, transformation, application, embedding, and protecting of organizational knowledge*" [39]. Even more, in the research of Ameri and Dutta, they described the PLM solution itself as a knowledge management system that supports each phase in PLC [1].

From the above definitions, in a PLM environment, knowledge management can be considered as a set of processes to manage the knowledge that is involved in all the phases of the product life cycle and to support the efficient cooperation. To be more specific, these processes can be classified as follows:

(1) *Capturing knowledge*, which is the primary step that focuses on identifying the relevant critical knowledge from stakeholders' minds;

(2) *Representing knowledge*, which is the foundational task that converts the

captured knowledge into knowledge representations for the subsequent processes;

(3) *Organizing knowledge*, which arranges (for example, classification, modification, update, storage, etc.) the collected knowledge representations into the knowledge repositories;

(4) *Sharing and using knowledge*, which focuses on supporting dissemination mechanism that guarantees the right information in the right context is delivered to the right stakeholders at the right time.

Further, Nonaka and Takeuchi suggested two essential factors to achieve a successful knowledge management approach [40]: the process of *making explicit the internalized tacit knowledge* for better sharing and the process of *internalizing the knowledge* that is retrieved from the knowledge management system into personal knowledge. As the foundation to assist these two factors, the system interoperability in a PLM environment that supports the exchange and the use of those knowledge representations in the PLC is required.

1.2.4 System Interoperability

In the compilation of IEEE standard computer glossaries [41], the interoperability is defined as "*The ability of two or more systems or components to exchange information and to use the information that has been exchanged*". In other words, it assumes that at least two or more "actors" are able to exchange some kind of "object" between them and to operate on that "object". Therefore these "actors" need to unambiguously interpret the exchanged "object" [10]. In this work, the "actors" are systems in a PLM environment and the "objects" are various kinds of models or part of models that are produced by those systems.

As stated by Lemoigne [42], who studied human organization as systems that present special characteristics, a general system is an object doing something in a certain environment and providing a permanent structure that is able to evolve and generally generate some results. In a PLM environment, enterprise systems are usually considered as a number of software components with certain relationships that are used to manage all kinds of technical information related to products in or across enterprises with the purposes of producing some finalities (such as a product, a decision, a strategy).

When a system tries to access the understanding of the exchanged information from another system, five possible levels of interoperability can be categorized [8]:

(1) *Encoding level*, the receiver system is able to segment the information in characters;

(2) *Lexical level*, the receiver system is able to segment the information in words (or symbols);
(3) *Syntactic level*, the receiver system is able to structure the information into structured sentences (or formulas or assertions);
(4) *Semantic level*, the receiver system is able to construct the proposed meaning of the information;
(5) *Semiotic level*, the receiver system is able to construct the pragmatic meaning of information (or its meaning in context).

Although this layered presentation is still arguable, it represents the fact that each level of the interoperability cannot be achieved until the previous levels are completed. While the interoperability between two systems at the encoding, lexical and syntactic levels are now possible to be achieved through using existing technologies (such as XML[43]) and its related applications (such as WSDL[44]), enabling the semantic and semiotic levels of interoperability still remains a huge challenge [9]. Meanwhile, with the similar idea of ascending levels of interoperability, European Interoperability Framework [46] proposed to consider three aspects of interoperability: *Technical interoperability*, which covers the technical issues of information exchanged between systems; *Semantic interoperability*, which concerns in ensuring that the precise meaning of the exchanged information is understandable; and *Organizational interoperability*, which concerns with defining business goals, modelling business processes and supporting the collaboration of administration stuffs. Because the issues of organizational interoperability are out the scope of this thesis and we assume that the technical interoperability can be achieved through certain standards, our research focus is limited on the semantic interoperability, namely system interoperability from the semantic perspective.

Semantic interoperability is the ability to ensure that the exchanged information has got the same meaning considering the point of view of both the sender and the receiver [47]. In the context of PLM, stakeholders have to work together on the exchanged information and take decisions based on this information. They have different backgrounds, heterogeneous expertise, unique knowledge, particular needs and specific practices, which over increase the difficulty to achieve semantic interoperability [48]. This situation interferes in achieving a mutual understanding between all the stakeholders, and so does in the collaboration across the enterprise systems. In order to cope with this issue there are two important obstacles that need to be overcame: (1) the implicit semantics that is necessary for understanding a knowledge representation is not be made explicit; (2) the lack of semantics mechanism to verify the correctness of explicit semantics in the exchanged knowledge representation.

The mutual understanding of semantics that is embedded inside the exchanged knowledge representation is the cornerstone in the quest for semantic interoperability. The essential of ontology [49], which is a kind of common agreement of a conceptualization of terms in a specific domain, makes possible its utilization to semantically enrich the exchanged knowledge representation. This is usually considered as a possible solution to deal with these two obstacles [10]. Therefore, in this context, semantic enrichment is considered as the process of making the implicit semantics through ontologies. It not only provides the clear semantics for facilitating the communication, but also gives the possibility to perform the semantics verification for those knowledge representations that are not initially designed with this ability.

1.2.5 Postulates

According to the research context (a PLM environment) and the research questions (semantic interoperability issues), also taking into account our research focuses and domain expertise, three hypotheses need to be declared before we proceed to the identification of problems and the proposition of some solutions:

(H1) *All the knowledge that is needed for the semantic enrichment of models has already been captured, represented and formalized into ontologies.*

(H2) *The corresponding interconnections among all the used ontologies have already been prepared through certain methods.*

(H3) *The semantic similarity between two objects can be compared through certain mechanisms.*

The supports for these hypotheses can be provided by related researches in the corresponding domains. The research community, which working on knowledge discovery [26], conversion [27], and formalization[49], can give support to the hypothesis H1. Taking advantages from the researches about ontology matching [50], mapping [51], and merging [52], Hypothesis H2 is possible to be achieved. For the hypothesis H3, a number of researchers, such as [53], [54], and [104], have committed themselves in the evaluation of semantic similarities. Even if there is not any generic automatic solution for the comparison, at least, from our perspective, this process can be done with the participation of domain experts.

Based on these hypotheses, the research focus of this work is proposing a solution to formalize the semantic annotation for the semantic enrichment of models in a PLM environment. In the next chapter, surrounding the definition of semantic annotation, we first discuss the targets of semantic enrichments and the ontologies that can be used to support the semantic enrichment. And then, we make a survey on the semantic annotation researches from different domains. This investigation will give us the existing drawbacks and potential challenges, which are the starting points of the proposed solution.

.

Chapter 2 Background and State of The Art

Oxford dictionary defines the concept of annotation as "*a note by way of explanation or comment added to a text or diagram*". It can be represented in various kinds of forms, such as text, underlines, sequence numbers, highlights, images and links. In order to distinguish semantic annotation from other annotations, Bechhofer et al.[55] categorized annotation into three types: *Textual annotation*, which adds notes and comments to an object; *Link annotation*, which extends the previous annotation by linking the object to the annotation content; *Semantic annotation*, which contains human readable as well as machine readable and processable information. Similarly, Oren et al. [11] proposed to classify the annotation as: *Informal annotation*, which is expressed in an informal language and is not machine-readable; *Formal annotation*, which is machine-readable, but without ontological terms; *Ontological annotation*, which is only composed of ontological terms that are commonly accepted and understood. These classifications identify two important features of the semantic annotation: (1) It is machine readable and processable; (2) It contains a set of formal and shared terms in the specific context.

Because of the nature of ontology, it is usually considered as the most suitable candidate for describing the terms in the semantic annotation. Different researchers have suggested many definitions of the semantic annotation that related to ontologies. For example, Talantikite et al. [56] described it as "*A semantic annotation is referent to an ontology*"; Lin [57] considered it as "*an approach to link ontologies to the original information sources*"; Kiryakov [58] defined it as "*a specific metadata generation and usage schema, aiming to enable new information access methods and to extend the existing ones*". Based on the research context that we presented in Section 1.2, the semantic annotation is considered as a mean to perform the semantic enrichment of "something" by using one or several ontologies.

Taking Nunamaker's understanding levels [25] as a reference, one of the major intents of applying semantic annotation is to enable annotated objects to be "understood" by a machine, to augment the degree of "understanding" from data level to knowledge level and to perform certain intelligent behaviours for the semantic interoperability based on the "understanding" at wisdom level.

In a nutshell, so as to have a complete state of the art about the semantic annotation in our research context, this chapter is structured into three sections: Section 2.1 briefly surveys models and their corresponding meta-models in a PLM environment, which are the targets of semantic enrichments; Section 2.2 surveys ontology specifications and various existing ontologies that are proposed by different academic research literatures, which can be used to support the semantic enrichment; Section 2.3 first surveys the current semantic annotation researches dealing with the semantic interoperability issues in different domains, and then classifies, compares and discusses them to identify the existing drawbacks that lead us to the proposed approach.

2.1 Models in a PLM Environment

To define a model of a system of interest created by an enterprise system, we adopted the description from the OMG[6]'s Model Driven Architecture (MDA) approach [59], which states that a model is a description or a specification of a system and its environment for some certain purposes. A meta-model is defined as a model that specifies the concepts, relationships and rules to model a model, which usually comprises a formalized specification of the domain-specific notations [60]. An introduction is presented to give an overview of the models in an enterprise in Section 2.1.1. Then, Section 2.1.2 describes the selected kind of model that we used to demonstrate the proposed solution.

2.1.1 Enterprise Modelling

Enterprise modelling is a process that tries to capture and represent knowledge from different aspects of a system of interest and for activating the interoperations in or across enterprises. In this thesis, all different types of models along the product life cycle are considered as the targets of semantic enrichments. For example, to mention only a few, product design models, data models, process models, state models, resource models, and decision models. Let's take the data modelling, business process modelling, state modelling and graphical product design as examples:

- Data modelling is an activity for providing certain format and structure of data for different information systems. Data Modelling Profile in UML [61] is one of the data modelling languages that support the expression of the data models.
- State modelling is an activity of describing the possible states of certain behaviours in a system. State Diagram [62] is one type of diagrams that are able to describe certain behaviours and states of systems.
- Graphical Product Design is an activity for designing a product in a 3D dimension to facilitate the further automatic manufacture process. Computer

[6] OMG(Object Management Group): http://www.omg.org/

Aided Design is one way that can be used to describe product design model.

- Business process modelling is an activity for representing different business processes that are performed in or across enterprises. Business Process Modelling Notation (BPMN) [63] is one of the standard languages that can be used to describe process models.

These models are always created with particular perspectives and expressed in a given modelling notation (or description language). They are helpful to provide the best possible knowledge in order to validate a new product from the initial design to the final market. The interoperations among the systems along a product life cycle not only require that the models can be exchanged and operated on, but also demands the unambiguous understandings of the exchanged models. Therefore, the necessary implicit semantics in those models must be made explicit.

Considering the diversity of models in an enterprise, it is more suitable that we select and use one kind of model to demonstrate the proposed solution. Since process models often get involved with different teams in or across enterprise and each stakeholder may have different viewpoints of some processes in a PLM environment, process model is chosen as the target of semantic enrichment in the case study.

2.1.2 Process Model and its Meta-model

A process is a series of activities or steps that are taken to achieve some particular objectives. A process model is a collection of related process components, which use certain nodes or links to organize process fragments into some meaningful sub-networks for the process that is modelled [64]. Various processes modelling languages have been proposed to represent processes from different perspectives, for example, the Event-driven Process Chain (EPC) [65], the UML Activity Diagram [66], the Business Process Model and Notation (BPMN) [63] and so on. These modelling languages are made specific, because of their disparate initial design needs.

During the enterprise process modelling, engineers are able to use those process modelling languages, which are implemented by various kinds of modelling systems, to model the processes of interest. However, these implementations usually contain some specific contents that fall short of standards, even if the modelling systems claim that they are following the corresponding standards. The application specific meta-models are usually different from the standard one.

We took the Business Process Model and Notation (BPMN) as an example, which is proposed by the Business Process Management Initiative and currently maintained by OMG since 2005. Its intention is to provide a graphical notation that can be understood by all business users that participate in the business [63]. It defines a business process diagram that can be used to model the drafts of various kinds of processes. The BPMN specification provides a set of basic notations as follows [63]:

- Flow Elements: *Event*, which is something that "happens" during the course of a process; *Activity*, which is is performed within a process and can be atomic (*task*) or non-atomic (*sub-process*); *Gateway*, which is used to control the divergence and convergence of sequence flows within a process;
- Connecting Objects: *Sequence flow*, which is used to show the order of Flow Elements in a process; *Message flow*, which is used to show the flow of Messages between two Participants that are prepared to send and receive them; *Association,* which is used to associate information and Artifacts with Flow Elements;
- Swimlanes: *Pool*, which represents a Participant in a collaboration; *Lane*, which is a sub-partition with in a process and extends the length of the Pool;
- Artefacts: *Data Object*, which provides information about what Activities require to be performed and/or what they produce; *Group*, which is used to highlight certain sections of a diagram without adding any constraints; *Message*, which is used to describe the contents of a communication between two Participants; *Text Annotation*, which is additional text information added by a modeller.

Figure 2-1 shows part of the meta-model of the "*Activity*" in the BPMN specification.

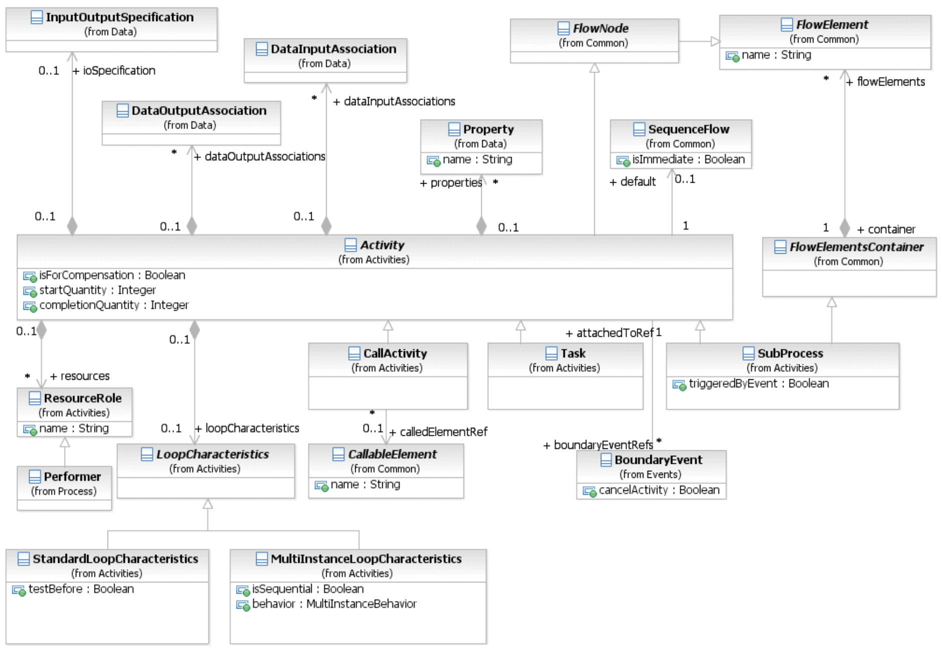

InputOutputSpecification (from Data)
DataInputAssociation (from Data)
FlowNode (from Common)
FlowElement (from Common)
name : String
0..1
+ ioSpecification
+ dataInputAssociations
+ flowElements
DataOutputAssociation (from Data)
Property (from Data)
name : String
SequenceFlow (from Common)
isImmediate : Boolean
+ dataOutputAssociations
+ properties
+ default
0..1
1
+ container
Activity (from Activities)
isForCompensation : Boolean
startQuantity : Integer
completionQuantity : Integer
FlowElementsContainer (from Common)
+ attachedToRef
CallActivity (from Activities)
Task (from Activities)
SubProcess (from Activities)
triggeredByEvent : Boolean
+ resources
ResourceRole (from Activities)
name : String
+ loopCharacteristics
+ calledElementRef
+ boundaryEventRefs
Performer (from Process)
LoopCharacteristics (from Activities)
CallableElement (from Common)
name : String
BoundaryEvent (from Events)
cancelActivity : Boolean
StandardLoopCharacteristics (from Activities)
testBefore : Boolean
MultiInstanceLoopCharacteristics (from Activities)
isSequential : Boolean
behavior : MultiInstanceBehavior

Figure 2-1 The Meta-model of the Activity in the BPMN Specification [63]

The MEGA modelling environment[7] supports the creation of various kinds of enterprise models. The "MEGA process BPMN" in MEGA supports the modelling of processes through the specific meta-model of BPMN notations that it implemented. Figure 2-2 shows part of its BPMN meta-model. The differences between the standard meta-model and the application specific meta-model can be identified. For example, to mention only a few, the connecting object *Association* between the *Data Object* and the *Sequence Flow* is omitted in this application; the *Process* is extended and divided into four kinds: *Organization processes*, *Business Process*, *Functional process* and *System Process*; the atomic *Activity* is extended and classified as *Operation, Functional Activity* and *Task* which correspond to the *Organization Process*, *Function Process* and *System Process* respectively.

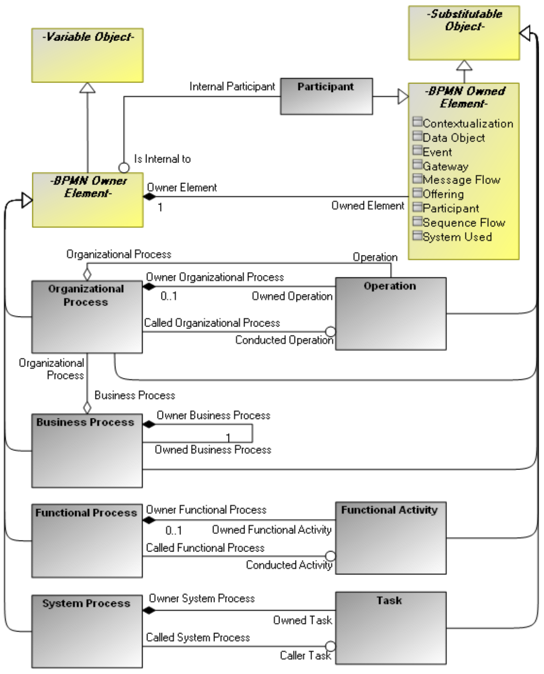


Figure 2-2 The Application Specific BPMN Meta-model in the MEGA

Both meta-models represent the knowledge about how a process model can be built. However, we can easily notice that, because of the differences between the two, the knowledge that is represented by the standard BPMN meta-model might not be sufficient enough to describe the structure semantics of a process model that is created by MEGA. For example, there are not corresponding concepts existing in the standard BPMN meta-model for the *Organizational Process* and its atomic activity *Operation*. This situation requires the supplementary knowledge from MEGA, which is represented by the application specific BPMN meta-model.

[7] MEGA: http://www.mega.com/ (we use the version 2009 SP5)

In order to demonstrate our proposed semantic enrichment solution, in the case study, the BPMN specification that is implemented by the MEGA Suite is chosen. It is used to create a manufacturing process in a particular application scenario. This process model is used as the target of semantic enrichment.

2.2 Ontologies

Based on the OMG's definition, from the representation point of view, a model "*is often presented as a combination of drawings and text. The text may be in a modeling language or in a natural language*" [59]. The mutual understanding of a model requires not only the understanding of the semantics of "combination of drawing" (structure semantics) but also the semantics of the "text" (domain semantics). Therefore, the ontologies employed by the semantic enrichment need to capture and represent both aspects of knowledge. In Section 2.2.1 we will briefly survey some major ontology specification languages, choose one of them, and use it in the remaining of the thesis. Section 2.2.2 shows the two kinds of existing ontologies that correspond to the domain semantics aspect and the structure semantic aspect respectively.

2.2.1 Ontology Specification Languages

Ontology research is one of the hottest topics that have attracted many attentions in the last decade. An ontology can be also considered as a formal and shared understanding of some domain of interests, which specifies the concepts and the relationships that can exist for an agent or a community of agents [49] [67].

A great effort has been made by different researchers in developing ontology specification languages. An ontology specification language is used to construct ontologies, which allows to encode certain knowledge in specific domains and to support a machine to perform reasoning, based on specific rules, on this knowledge. We are not going to give, a complete overview of ontology specification languages, but we will provide a brief introduction and discussion. As shown in Figure 2-3, an ontology language stack, which classifies the typical ontology languages, is presented by Song et al [68]:

(1) *Frame-based ontology specification language*, which is represented by frame languages. For example,
   a) *F-Logic* [69], which combines conceptual modelling with object-oriented, frame-based languages and offers a declarative, compact and simple syntax;
   b) *Ontolingua* [70], which is an extension of Knowledge Interchange Format[71] through adding frame-based representation and translation

functionalities to enable the specification in an object-oriented style;

c) *CML/OCML* [72]. Conceptual Modelling Language (CML) provides a structured textual and a diagrammatic notation to specify knowledge models (informal notation). Operational Conceptual Modelling Language (OCML) extends the CML with formal frame representations.

(2) *Logical-based ontology specification language*, which is represented by logical languages (such as first order logic and description logic) and has a long history in artificial intelligence domain. For example,

a) *LOOM* [73], which is a knowledge representation language and environment that focuses on supporting the reasoning of knowledge representations in the artificial intelligence domain;

b) *CycL* [74], which is a flexible knowledge representation language that extends the first-order predicate calculus by handling equality, default reasoning and some second order features.

(3) *Web-based ontology specification language*, which is based on HTML[8], XML and RDF/RDFS[75] technologies and is mainly used in semantic web. For example,

a) *SHOE* (Simple HTML Ontology Extension) [76], which extends HTML with a set of object-oriented tags to provide a structure for the knowledge acquisition.

b) *DAML* [77] + *OIL* [78], which is a semantic mark-up language for web resources that combines the features of both the DARPA (Defense Advanced Research Projects Agency) Agent Mark-up Language (DAML) and the Ontology Interchange Language (OIL).

c) *OWL(Web Ontology Language)* [79], which is a reversion of DAML+ OIL. It has more facilities for the expressing semantics than XML, RDF and RDFS by providing a machine interpretable content on the Web.

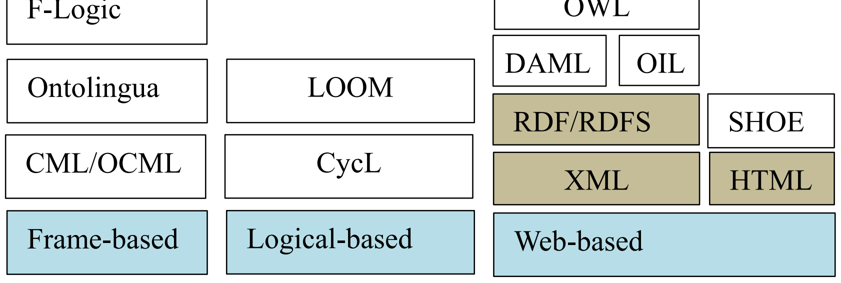


Figure 2-3 Ontology Languages Stacks [68]

[8] HTML http://www.w3.org/TR/REC-html40/

A comparison of most of above-mentioned ontology specification languages (besides OWL) has been made by Corcho et al. [80], which is based on eight main feature perspectives: Concepts, Attributes, Facets, Taxonomies, Relations, Functions, Axioms and Instances. As a comparison result, Ontolingua, LOOM and OCML are the three ontology specification languages that cover most of these evaluated features. However, because of this comparison is made in 2003, at that moment OWL was only a working draft, for that reason, it was not taken into account in that comparison. Currently, as a successor of DAML+OIL, OWL has attracted more and more attentions and became one of the W3C[9] recommendations in 2004.

After all, ontology not only defines the formal semantics that enables a reasoning machine to perform inference, but also represents a real-world semantics that enables human to use meaningful terminologies as machine processable contents. Based on the survey of the major ontology specification languages, and also taking into account the expressiveness and the related reasoning supports, we found that OWL is the most appropriate ontology specification language for supporting the implementation of the prototype annotation tool in this research work. In order to support the expression in Chapter 4 and 5, we briefly introduce the OWL specification together with its corresponding reasoning rules and reasoning engines in APPENDIX I.

2.2.2 PLC-related Ontologies and Meta-model Ontologies

Ontology engineering is one of the prominent solutions that is used to capture and represent knowledge and to provide precisely description of concepts and the relationships between them [50]. A six steps process, which supports the ontology creation activities during the ontology engineering, is proposed by Pulido et al [81]:

(1) *Gathering*, which is the collection of the relevant structured or unstructured resources from the domain of interest;
(2) *Extraction*, which extracts ontology concepts, relationships and instances from the collected resources;
(3) *Organization*, which uses these extracted contents to generate a formal ontological knowledge representation;
(4) *Merging*, which defines the mapping rules to merge other ontology from one context to another;
(5) *Refinement*, in which, domain experts are invited to improve the structure and contents of the ontology;

[9] W3C http://www.w3.org/

(6) *Retrieval*, which is the release of the ontology to support the final objective of semantic web.

Depending on the level of the knowledge that an ontology aims to represent, ontologies can be generally categorized into three levels as follows [82]:

(1) *Top level ontology*, which specifies only general concepts and relationships (such as time and space) and can be used in different domains;
(2) *Domain level ontology*, which captures the knowledge that is dedicated to a specific domain (such as production domain) and can be use and reused for different tasks in the same domain;
(3) *Application level ontology*, which represents the specific knowledge that is dedicated to a task in an application and normally is not reusable for other applications.

With the support of the fast growth of ontology technologies, more and more research interests focus on the realization of ontologies. In this research work, two aspects of ontologies are categorized and can be used to support the semantic enrichment of models in a PLM environment:

(1) *PLC-related ontologies*, which represent the PLC-related knowledge. They can be used to express the domain semantics of annotated objects that related to one or more stages of a product life cycle.
(2) *Meta-model ontologies*, which represent the model constructs knowledge. They can be used to express the structure semantics of annotated objects that are related to the interrelations between their counterpart components in its meta-model.

Because of the development of ontologies is not the focus of this research, we need to discover and employ some existing ontologies to support the semantic enrichment of models in a PLM environment. In this section, a survey is carried out to investigate a number of PLC-related ontologies and Meta-model ontologies.

PLC-related ontologies

Five ontologies that represent the knowledge related to one or more stages of a PLC are introduced as follows:

(1) ONTO-PDM [83] is proposed to formalize all technical data and concepts related to the definition of a product, for minimizing the loss of semantics. The authors postulated that an ontological model of a product may be considered as a facilitator for the interoperation of all application software, which share information during the physical product lifecycle.

(2) SCOR-Full ontology [84] is proposed to resolve the semantic inconsistencies and incompleteness of the SCOR (Supply Chain Operations Reference) models that are used for the knowledge management among the supply chain networks.

(3) OntoSTEP [85] is proposed to consolidate product information that are created by different languages for building a coherent knowledge base. Authors presents a way to transform the STEP [86] schema and its instances from EXPRESS [87] format to OWL one via mapping rules and result in a STEP ontology.

(4) CMMI ontology [88] is proposed to be used as a base for making fuzzy cost estimations of a project. Authors proposed to adopt the part of the Capability Maturity Model Integration (CMMI) [89] that is related to the development and maintenance of products and services covering the PLC and to formalize them as a cost estimation ontology.

(5) MSDL Ontology [90] is proposed to describe the capabilities of manufacturing services in different abstraction levels. It provides the formal semantics for enabling the machine agents to actively participate in the supplier discovery process.

Table 2-1 illustrated the comparative overview of these five ontologies through four aspects: *domains of application*, *notations of expression*, *levels of ontology* and foundation

Table 2-1 Comparison of five PLC-related Domain Ontologies

	ONTO-PDM [83]	SCOR-Full Ontology[84]	OntoSTEP [85]	CMMI Ontology [88]	MSDL Ontology[91]
Domains of Application	All product related data	Supply Chain	Product Design	Project Planning	Manufacture Capabilities
Notations of Expression	Class Diagram	OWL, Class Diagram	OWL-DL	Self-define Structure	OWL-DL
Levels of Ontology	Domain Level	Domain Level	Domain Level	Application Level	Domain Level
Foundation	IEC62264 [92] ISO10303 [86]	SCOR [93] Model	ISO 10303 [86]	CMMI[89] Standard	Author's Research

To our best knowledge, several conclusions can be made: (1) Although the domain of applications are different, there are still overlapping contents between each other; (2) According to the notations of expression, we found that OWL is the frequently used ontology specification language; (3) Most of the PLC-related ontologies are domain level ontology and most of them are created in relation to the corresponding standards.

Meta-model Ontologies

Five ontologies that represent the model constructs knowledge of different kinds of models are introduced as follows:

(1) Bunge-Wand-Weber [94] is proposed to provide the basis and fundamental concepts that are needed for the theoretical view of information systems. It is not only can be used to model a wide variety of information systems phenomena, but also can be used as essential elements for evaluating the “grammars” of a conceptual information system modelling method.

(2) General Process Ontology [95] is proposed to support the meta-model annotation by providing the common and core semantics of process modelling constructs. It acts as a mediator for the semantic conciliation between GPO concepts and different process modelling language constructs.

(3) Petri net Ontology [96] is proposed to support the annotations of different Petri net dialects and provide an infrastructure to enable the use and share of those Petri nets on the semantic web.

(4) Activity diagram ontology [97] is proposed to create the mutual understanding of terms that supports the communication between different software development teams.

(5) BPMN Ontology [98] is proposed to capture the structural components of the BPMN and provide a clear semantic formalisation for supporting the semantic enrichment of business process models .

Table 2-2 shows the comparative overview of these five meta-model ontologies, through four aspects: *domains of application*, *notations of expression*, *levels of ontology* and *foundation*

Table 2-2 Comparison of five Meta-model Ontologies

	BWW [94]	GPO [95]	Petri net ontology [96]	Activity Diagram Ontology [97]	BPMN ontology[98]
Domains of Application	Information system	Process model	Petri Net Model	Activity Diagram	BPMN Model
Notations of Expression	Theory	OWL-DL	OWL	Self-define Structure	OWL-DL
Levels of Ontology	Domain Ontology	Domain Ontology	Application Ontology	Application Ontology	Domain Ontology
Foundation	Mario Bunge’s ontology [99]	BWW [94]	PNML [100] APNN [101]	SE textbook [102], SWEBOK[103]	BPMN Specification [63]

We can assert several conclusions: (1) In scientific literature, there exists various studies that proposes ontology for representing the structural components of a modelling language; (2) According to the notations of expression, we discover that OWL is also the most frequently used ontology specification language in this type of ontologies; (3) most of the meta-model ontologies are domain level ontologies and most of them are created based on the corresponding language standards or specifications.

Some researchers argued that a meta-model is not an ontology. We cannot dissert about the specific philosophical and methodological problem. Through the comparison between the meta-models in the model specification languages and the meta-model ontologies that are created based on those meta-models, we discover that the contents (concepts and relationships) that we need from a meta-model can be represented in an ontology. We also found that the meta-model ontologies in the existing literatures are usually used as a mediator to annotate various kinds of meta-models for supporting the model exchange or transformation. However, there is lack of research that uses meta-model ontologies to describe the interrelation between annotated objects, and combine the PLC-related ontologies to support the annotation suggestion and verification. In the next section, we will survey some current semantic annotation researches from different aspects and give a detailed comparison.

2.3 Semantic Annotations

The objectives of semantic annotations can be categorized into three general groups:

(1) *Group 1*, to specify some embedded implicit semantics to improve the understanding of the annotated objects;
(2) *Group 2*, to identify the common semantics among those annotated objects from different sources to support further operations (such as transformation, mapping, exchanging);
(3) *Group 3*, to attach the machine processable semantics to those annotated objects and to obtain a set of semantic reasoning supports (such as querying, inferring and verification).

Usually, a semantic annotation research may involve one or more groups of the above-mentioned usages. In this section, based on the investigation performed in [12] and [13], an extended version of survey is presented. A number of current semantic annotation researches is classified and compared in Section 2.3.1. And then, the existing drawbacks and potential challenges among those researches are discussed in Section 2.3.2.

2.3.1 Comparisons of the Semantic Annotation Researches

Based on the supports of the ontologies, semantic annotations could be widely used in many contexts. Uren et al. [104] reviewed and classified the existing semantic annotation systems as four kinds: manual annotation (annotations are manually created by users), automatic annotation (annotation are created with the assistant of automation components), integrated annotation environments (standard tools, such as Microsoft Word, which has integrated with an annotation process), and On-demand annotation (tools that produce annotation-like service, such as highlighting text related to an ontology). While this survey serves to introduce and classify those annotation tools generally, deep analysis of the annotation formalization details is still lacking. In order to have a more complete and detailed overview of semantic annotations, the scope of the related semantic annotation researches are not only limited to the semantic annotation of models, but also extended to the semantic annotation for the Web services and texts.

2.3.1.1 Semantic Annotations for Web Services

The W3C defines a Web service as "*a software system designed to support interoperable machine-to-machine interaction over a network*" [105]. Adding semantic annotations to a Web service is mainly for supporting the automatic verification of certain tasks, which must be executed before or during invocation of corresponding services [106]. Lots of efforts have been made in the semantic enrichment of Web services.

Talantikite, et al. [56] proposed to use semantic annotations to annotate the Web service for assisting the creation of an inter-connected network (represented in OWL-S [107]), which is then processed by a composition algorithm to discover an appropriate composition services plan for answering the corresponding user requests. A semantic model, which can be considered as a kind of schema, is proposed to annotate a Web service. As shown in the Figure 2-4 (a), the *inputs* and *outputs* in this schema are used for the similarity measurement and *exec-time* and A*ll-Resources* are quality criteria for the evaluation of the best composition plan.

Patil, et al. [108] proposed MWSAF, a framework for semi-automatically annotating Web services with domain ontologies to help Web services discovery and composition. They first converted both WSDL and several ontologies into Schema Graphs (a set of nodes connected by edges) and then compare each concept from the former one versus the concepts from latter ones based on both linguistic similarity and structure similarity. After the comparison, the best-matched ontology is provided for user to verify the correctness of each mapping. At the end, semantic annotation is used as a simple ”is a” association to link annotated concepts and ontology concepts based on the accepted mappings.

In order to simplify and standardize the complex semantic annotation methods for Web services, the Semantic Annotation for WSDL and XML Schema (SAWSDL) [45] is proposed in 2007. It aims to add semantics to Web services by providing extension attributes that can be applied to the elements of both WSDL and XML Schema. As shown in the Figure 2-4 (b), the SAWSDL extensions can be distinguished in two kinds:

(1) Model References (*sawsdl:modelReference*), which describes the associations from a WSDL component or a XML Schema component to a concept in some semantic models;

(2) Schema Mappings (*sawsdl:liftingSchemaMapping* and *sawsdl:lowering SchemaMapping*), which specifies how an instance data in an XML Schema maps to a semantic data in a semantic model.

The former one lifts (transforms) an XML instance data from a Web service message into a semantic model; the latter one lowers (transforms) a semantic data from a semantic model into an XML message [106]. Due to the initial design of SAWSDL it assumes the semantic model can be identified through URIs, it is supposed to be able to cope with semantic models based on any ontology specification languages.

2.3.1.2 Semantic Annotations for Texts

The semantic enrichment of texts is designed mainly to fulfil the purpose of helping a machine to “understand” the annotated contents in the text and supporting automated processes (such as information navigation). These researches usually employed some information extraction technologies, such as natural language processing [109], to support the automatic extraction of structured information from the unstructured of semi-structured documents. Of course, not limited to this, a large number of researches have proposed.

Vargas-Vera, et al. [110] presented an ontology-based annotation tool, named MnM, which integrates web browser, ontology editor and open APIs to provide both automatic and semi-automatic supports to annotate web pages with semantic contents. The annotation (so-called mark-up) is performed through inserting a number of tags (based on the name of the selected ontology concept) into the selected segments of text on the web pages. Then a learning algorithm is applied on those corpora that are collected from the annotation phase to create new annotations. With the supports of the annotations, MnM is able to extract information from web pages and then fill them into a pre-defined template. Further, a simple type-based validation is proposed to verify the correctness of contents that are being filled in.

Popov et al. [111] developed a knowledge and information management (KIM) platform, which was based on a KIM ontology and a massive knowledge base to automatic annotate, index, and retrieve of documents. Based on the hypothesis that named entities (NE), such as people, location and others referred by name, constitute the essential semantics in a document. The automatic semantic annotation is considered as the process of NE recognition and annotation. KIM provides for each extracted NE two kinds of links: one link (URI) to the most specific class in KIM ontology to specify the named-entity type and another link to specific individual in knowledge base.

Ma et al. [112] proposed a framework to support the semantic reasoning on both domain and linguistic information that are contained in annotations of texts. That research uses two ontologies: (1) a domain ontology to provide semantic labels (domain knowledge) and (2) a language ontology to give text model (linguistic knowledge). For the former one, a semantic annotation assertion is defined as a triple <*tf,ot,at*>, as shown in the Figure 2-4 (c), where *tf* is text fragment; *at* is semantic labels; *ot* is relation between *tf* and *ot*. For the latter one, it is represented as a set of OWL axioms and SWRL rules, which contributes to bridge the inference constraints between domain and linguistic.

2.3.1.3 Semantic Annotations for Models

In a PLM environment, various kinds of models have been proposed to represent the PLC-related knowledge. As discussed in Section 1.2.4, the mutual understanding of the semantics inside the shared and exchanged knowledge representations is one of the important processes to achieve the semantic interoperability. We investigate several semantic annotation researches that focus on the semantic enrichment of enterprise models (from the general points of view), data models, product design models and process models.

Enterprise Models

Task Group 4 of the INTEROP project [55] committed themselves in investigating how annotations are able to contribute in making explicit the semantics and the structures of enterprise models to enable both semantic-based and model-based interoperability between collaborating actors. As shown in the Figure 2-4 (d), a general schema is proposed for the semantic annotation of all enterprise models. They assumed that any parts within an enterprise model may be annotated and can be annotated with multiple annotations [10]. With this hypothesis, they categorized annotations into three types for supporting the model exchange and transformation in an heterogeneous context: *Structural annotations*, which refer to a given meta-model that supports the mapping of model constructs; *Lexical/Terminological annotations*, which refer to a taxonomy or an ontology that supports the mapping of annotated object's names at semantics level; *Behaviour annotations*, which can be expressed in various forms to make explicit the business logic, the procedures, the rules, and the policies of the annotated object [113].

Data Models

Song et al. [68] investigated the issues of heterogeneous data systems and proposed a semantic information layer (SIL), which acts as a mediation media among these systems to overcome gaps of data and semantic heterogeneity. This research focuses on the development of an ontology-driven framework, which supports the extraction of ontologies from different databases and assists creation and management of the SIL. Semantic annotations are only used as links (paths) between the SIL and data schemas, which are generated automatically.

MOMIS project [114] proposed an annotation method to support the automatic and semi-automatic annotation on two or more data models that are extracted and converted from either structured or semi-structured data sources. Based on these annotations, MOMIS system first extracts four kinds of predefined semantic relationships (*Synonyms*, *Border Terms*, *Narrower Terms*, *and Related Terms*) from those annotated objects and then generates a global schema to support the data integration between different data sources. In the local source annotation phase, the generic WordNet lexical database and a domain glossary are employed to store and provide human readable meanings for annotators. However, semantic annotation is only considered as a kind of association between an element in data model and its WordNet text meanings.

Product Design Model

Attene et al. [115] developed a semantic annotation system, named ShapeAnnotator, which is able to decompose a shape into several interested features through a segmentation algorithm and to support the annotation of the selected features by connecting them to the corresponding individuals. These individuals are saved in a separated OWL file with the imports of domain ontology. Figure 2-4 (e) shows an annotation schema that we summarized from this research, which contains four main elements: *Class* that this individual asserts to; *ShannGeoContextURI* that contains the value of an URI that points to a multi-segmented mesh; *ShannSegmentID* that contains the value of an index that specifies a segment in that multi-segmented mesh; *Related Values* that is computed and added based on the feature descriptors (topological reactions between features and geometric aspects of a feature);

Li [116] proposed an ontology-driven semantic annotation framework for CAD systems (OntoCAD) to assist the product engineering with other multiple engineering viewpoints (such as cost estimation) in a product life cycle. An annotation data structure is proposed to formalize those annotations, as shown in the Figure 2-4 (f), which is composed of three parts: *Anchor* is filled with the geometric elements that are represented as OWL individuals; *OWL properties* are the objects property and the data property in OWL; *Content* can be OWL individuals or data values. A three layered ontology architecture knowledge base is proposed to capture, represent and manage multiple engineering viewpoints ontologies and to support the processing of querying and reasoning requests.

Process Models

Di Francescomarino [117] proposed some techniques to support the annotation of business process model with ontologies, which gives the possibility to perform some reasoning for assisting designers and analysts in the management of their business process models. On one hand, semantic annotation is represented in the "textual annotations" of a business process diagram by using a "@" symbol with the name of the selected ontology classes. On the other hand, annotated object in process model has a corresponding ontology individual element and an assertion to the selected class. Figure 2-4 (g) shows the annotation schema that we summarized from this research. Based on those three types of assertions, the semantic annotation is treated as a kind of assertion between individuals and classes.

Lin [57] proposed a semantic annotation framework to support the discovery and the sharing of process models in or between enterprises by reconciling the semantic heterogeneity between process modelling languages (meta-model) and model contents. The meta-model is annotated by a general process ontology (GPO) and model contents are annotated by a domain ontology. As shown in the Figure 2-4 (h), a process semantic annotation model (PSAM), which describes the process properties and annotation contents, is proposed to generate a common annotation schema for different process models. In order to better describe the semantic relationships between concepts in models and the concepts in ontologies, a set of refined relations is defined besides the simple *refers_to* relations.

2.3.1.4 Comparison

To be more specific, a comparison of above-mentioned semantic annotation researches is made in the Table 2-3 to give a more complete overview based on the following nine aspects:

(1) *Domains of Application*: this column describes the target of semantic annotations, which rely on the contexts of research, such as web services, texts, models and so on.
(2) *Usages of Annotation*: this column describes what semantic annotations are used in the corresponding research. It is classified into three groups at the beginning of Section 2.3.
(3) *Ways of Annotation*: this column describes how semantic annotations are added to the target. The contents of this column can be "manual annotation", "semi-automatic annotation" or "automatic annotation".
(4) *Semantic Browser*: this column describes what kind of browser is provided to annotators for browsing the semantic models. In the case of automatic annotation, this aspect can be omitted.
(5) *SA verification*: this column describes whether there is a mechanism to detect the inconsistencies between existing semantic annotations and a mechanism to give warnings for potential conflicts between those annotated objects.
(6) *SA Schema*: this column describes whether there is a semantic annotation schema in the corresponding research. In this aspect, the simple "is a" association is not considered as a schema.
(7) *Employed Ontologies*: this column describes what ontologies are used in the corresponding research.
(8) *SA Independency*: this column describes how semantic annotations attach to the annotated object. The contents of this column can be: embedded references in the target of annotations (such as URI, tag or ontology concept) or independent references from the target of the annotation (storing SA independently).
(9) *Structure Semantics*: this column describes whether the structure semantics of the target of annotations is taken account by the corresponding researches.

Table 2-3 The Comparison of Semantic Annotation Researches

Name of the Research/Authors	Domains of Application	Usages of Annotation	Ways of Annotation	Semantic Browser	SA Verification	SA Schema	Employed Ontologies	SA Independency	Structure Semantics
Talantikite, et al. [56]	Web Services	Group 3	No mention	No mention	No	Yes	Domain ontologies	Independent	No
MWSAF [108]	Web Services	Group 3	Manual Semi-automatic	Ontology tree view	No	No	Domain ontologies	Embedding ontology concepts	No
SAWSDL [45]	Web Services	No Specify	No mention	No mention	No mention	Yes	No Specify	Embedding URIs	No mention
MnM [110]	Text	Group 1 and 3	All three types	Ontology tree view	No	No	Domain ontologies	Embedding tags	No
KIM [111]	Text	Group 1 and 3	Automatic		No	No	KIM ontology, knowledge base	Embedding URIs	No
Yue Ma [112]	Text	Group 2 and 3	Automatic		Yes	Yes	Domain ontology Language ontology	Independent	Yes
Task Group 4 of INTEROP [55]	Enterprise Models	Group 1, 2 and 3	No mention	No mention	No	Yes	No mention	Embedding URIs	Yes
SIL [68]	Data Model	Group 2 and 3	Automatic		No	No	Extracted ontologies	Independent	No
MOMIS project [114]	Data Model	Group 1, 2 and 3	All three types	Natural Language view	No	No	WordNet Domain Glossary	No mention	No
ShapeAnnotator [115]	Product Design Model	Group 3	Manual	Ontology graph view	No	Yes	Domain ontology	Independent	Yes
OntoCAD [116]	Product Design Model	Group 1, 2 and 3	Manual	Ontology tree view	No	Yes	STEP ontology	Independent	No
Di Francesc-omarino [117]	Process Model	Group 2 and 3	Semi-automatic	No mention	Yes	Yes	BPMN ontology, BPO	Independent	Yes
Lin [57]	Process Model	Group 1, 2 and 3	Manual	Ontology tree view	No	Yes	GPO, goal and domain ontology	Independent	Yes

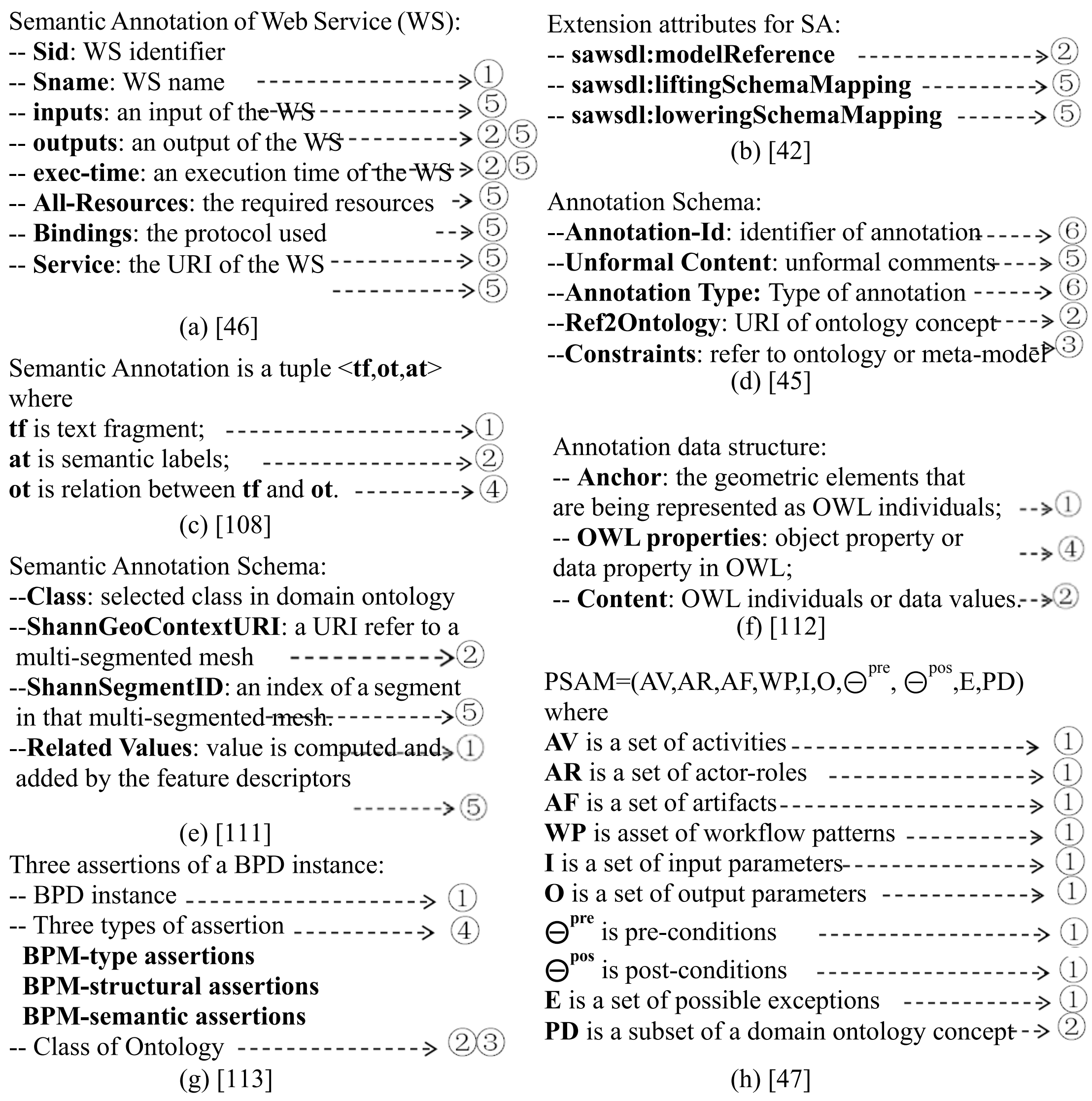
According to these comparison results, several conclusions can be emphasized as follows:

(1) Most of the researches focus on using semantic annotations to support the usage in group 3 (attaching the machine processable semantics to those annotated objects and obtaining a set of semantic reasoning supports);
(2) In the cases of manual annotation and semi-automatic annotation, ontology tree views are the most used browsers for annotators to browse semantic models;
(3) The verification mechanism is not considered by most of the researches;
(4) Various kinds of semantic annotation models have been proposed by different researches.
(5) In the cases of semantic annotation for Web services and texts, semantic annotations are always embedded in the annotation target. To the contrary, for models, semantic annotations are always independent;
(6) The domain semantics is usually made explicit through one or more domain ontologies and the structure semantics is usually made explicit through a meta-model ontology. Meanwhile, less than half of the researches take into account the structure semantics.

In order to make a more detailed comparison of the semantic annotation models that are proposed by above-mentioned researches, we classify six types of the elements that are contained in those schemas:

- Element type ①, which contains the identifier of annotated object.
- Element type ②, which contains the domain semantics.
- Element type ③, which contains the structure semantics.
- Element type ④, which contains the relations between annotated object (element type ①) and its domain or structure semantics (element type ② or ③).
- Element type ⑤, which contains some specific properties that are associated to the annotated object (the additional information that do not describe the semantics of annotated object).
- Element type ⑥, which contains some specific properties that are associated to the annotation itself (such as annotation id, annotation type and so on).

Figure 2-4 shows an overview of the eight semantic annotation schemas, in which, all elements in those schemas are categorized based on the proposed classification.


Semantic Annotation of Web Service (WS):
-- **Sid**: WS identifier
-- **Sname**: WS name ----------------> ①
-- **inputs**: an input of th~~e WS~~ -----------> ⑤
-- **outputs**: an output of the ~~WS~~ ---------> ②⑤
-- **exec-time**: an execution time of ~~the WS~~ > ②⑤
-- **All-Resources**: the required resources -> ⑤
-- **Bindings**: the protocol used --> ⑤
-- **Service**: the URI of the WS ----------> ⑤
----------> ⑤

(a) [46]

Extension attributes for SA:
-- **sawsdl:modelReference** --------------> ②
-- **sawsdl:liftingSchemaMapping** ---------> ⑤
-- **sawsdl:loweringSchemaMapping** ------> ⑤

(b) [42]

Semantic Annotation is a tuple <**tf,ot,at**> where
tf is text fragment; -------------------> ①
at is semantic labels; ----------------> ②
ot is relation between **tf** and **ot**. ---------> ④

(c) [108]

Annotation Schema:
--**Annotation-Id**: identifier of annotation -----> ⑥
--**Unformal Content**: unformal comments ----> ⑤
--**Annotation Type:** Type of annotation -----> ⑥
--**Ref2Ontology**: URI of ontology concept ---> ②
--**Constraints**: refer to ontology or meta-model > ③

(d) [45]

Semantic Annotation Schema:
--**Class**: selected class in domain ontology
--**ShannGeoContextURI**: a URI refer to a multi-segmented mesh ----------> ②
--**ShannSegmentID**: an index of a segment in that multi-segmented ~~mesh.~~ ---------> ⑤
--**Related Values**: value is comp~~uted and~~ > ① added by the feature descriptors
--------> ⑤

(e) [111]

Annotation data structure:
-- **Anchor**: the geometric elements that are being represented as OWL individuals; --> ①
-- **OWL properties**: object property or data property in OWL; --> ④
-- **Content**: OWL individuals or data values. --> ②

(f) [112]

Three assertions of a BPD instance:
-- BPD instance -------------------> ①
-- Three types of assertion ----------> ④
BPM-type assertions
BPM-structural assertions
BPM-semantic assertions
-- Class of Ontology --------------> ②③

(g) [113]

PSAM=(AV,AR,AF,WP,I,O,$\ominus^{pre}$, $\ominus^{pos}$,E,PD) where
AV is a set of activities -------------------> ①
AR is a set of actor-roles ----------------> ①
AF is a set of artifacts --------------------> ①
WP is asset of workflow patterns ----------> ①
I is a set of input parameters ---------------> ①
O is a set of output parameters -----------> ①
$\ominus^{\mathbf{pre}}$ is pre-conditions ------------------> ①
$\ominus^{\mathbf{pos}}$ is post-conditions ------------------> ①
E is a set of possible exceptions ----------> ①
PD is a subset of a domain ontology concept --> ②

(h) [47]

Figure 2-4 The Comparison of Semantic Annotation Models

Combining the element type ① with the contents of *SA independency* in the table 2-3, we can discover that this type of element only exists in (a), (c), (e), (f), (g) and (h), which belong to the research that stores semantic annotation independently. To the contrary, for those researches that embed references in the target of annotation, their schemas, such as (b) and (d), do not contain this element.

All the schemas contain the element type ②. Besides (b) and (d) that express this type of element as URIs, the rest of them use ontology concepts. Normally, this type of element is used to make explicit the domain semantics of annotated objects. However, in particular, the domain semantics in (a) is used to express the semantics of the inputs and outputs of the annotated object.

Based on the observation of those models, only (d) and (g) contain the element type ③. However, after the detailed analysis, we discover that the structure semantics is also taken into account by (c), (e) and (h):

- In (c) and (e), structure semantics is not directly appeared. Instead, for the former one, it is represented as a text model (language ontology) with pre-defined axioms and rules. For the latter one, it is expressed by the topological relations between two features (such as adjacency, overlap, disjointness and containment).
- In (h), besides element PD, the rest of the elements in the schema are generated based on a meta-model ontology, named GPO. After mapping a meta-model to GPO, the corresponding model element is converted into an individual of the mapped class in GPO. (e.g. when a "process" is annotated as an instance of *AV*, it automatically maps to activity in the GPO).

Outwardly, besides the (c), (f) and (g), the rest of them do not contain the element type ④, which defines the semantic relationship as a simple assertion or linking between annotated object and semantic content.

- In (c), relations are classified into four kinds: "sa:Concept" states *tf* is annotated by an class; "sa:Role" states *tf* is annotated by a property; "sa:Individual" states *tf* is annotated by an individual; "sa:Ind-Con" states *tf* is an individual, as well as the annotation content is the class that *tf* belongs to (*tf* is a special individual of this class).
- In (f), the definitions of OWL properties (*owl:ObjectProperty* and *owl:DatatypeProperty*) are reused for identifying the relations between annotated objects and annotation contents. The former one denotes the annotation content as an object and the latter one denotes the annotation content as a data value.
- In (g), relations are classified into three kinds of assertions: BPM-type assertions that assert an instance to a class of the BPMN ontology; BPM-semantic assertions that assert an instance to a class of a domain otology, BPM-structure assertion is used to describe the relations between two instances.

However, although element type ④ does not directly appear in (h), the semantic relationships are represented as OWL properties in that research. Seven kinds of refined relations are generated: Synonym (alternative_name, same_as), Polysemy (different_from), Hypernym (kind_of), Hyponym (superConceptof), Meronym (part_of, member_of, phase_of, partialEffect_of), Holonym (composition Concept_of), Instance (instance_of).

Element type ⑤ and ⑥ are specific elements in some contexts that usually are used to fulfil some particular requirements. For example, in (a), the "exec-time" is a property of the annotated object, which is used to record the execution time of a web service request. In (c) “Annotation-Id” is a property that is associated to the annotation itself, which is used to record the value of the identifier of that annotation.

In short, although various kinds of semantic annotation models have been proposed by different semantic annotation researches, they are defined differently and limited in their own studies. The essential elements of a semantic annotation are not clearly presented by above-mentioned researches. We will present the needs of the formalization of semantic annotations through the discussion of existing drawbacks and potential challenges in the next section.

2.3.2 Drawbacks and Challenges

Based on the investigations and comparisons, we found that despite lots of efforts have been made in the research of semantic enrichments, a number of existing drawbacks still needs to be noted.

The formalization of semantic annotations is not the focus in some of above-mentioned researches ([68], [108], [110], [111] and [114]), where it is only considered as a simple one to one association (a kind of “is a” association between an annotated object and an ontology concept). Meanwhile, some specific semantic annotation models are proposed by some of the rest ([56], [57], [112], [115] and [117]). However, these models are difficult to be reused in other researches but the studied ones. There exists a kind of general models in the research [45] and [55]. However, for the former one, although it provides the user a large degree of freedom, it does not contain any semantic relationships and without additional conventions, which results in a limited usage. As well as the latter one is only a general annotation model without detailed formal definitions.

Making explicit the domain semantics is the only concern in some of above-mentioned researches ([56], [68], [108], [110], [111], [114] and [116]), where the structure semantics is ignored. The advantages of making explicit the structure semantics have been acquired by the rest of them ([55], [57], [112], [115] and [117]). In [55], it is used to express modelling construct and support models transformations. In [57], it is used as a mediator for reconciliation of various process modelling language constructs. In [112], it is used to support the creation of text model and conserve linguistic knowledge. In [115], it is used to support the automatic computation of relations between features in the model. In [117], it is used to support the verification of modelling constraints. However, among all these usages, the structure semantics and domain semantics are defined separately. There is a lack of research that combines both structure and domain semantics together to contribute to the inference process.

In the cases of automatic or semi-automatic annotation, semantic annotations are usually suggested by some similarity measurements methods ([68], [108], [117] and [114]) or training corpus ([110], [111] and [112]). The verification of correctness of those semantic annotations still needs human involved. The mechanism for assisting this verification process is only taken into account by [112] and [117]. In [112], two SWRL rules are designed to report missing and erroneous annotations on a noun compound (three text fragments). In [117], four axioms are proposed to prevent erroneous annotations based on the types of concepts. However, they only focus on one annotated object and the possible inconsistencies between two or more semantic annotations are not taken into account.

In this research work, the process model is selected as the target of semantic enrichment. Therefore, among all the investigated researches, [57] and [117] are close related to the semantic annotation of process models. Besides the above-mentioned drawbacks, several more shortcomings still need to be noted:

- In [57], (1) process models and their meta-model descriptions are represented as tree views in the annotation tool. These tree views not only neglect the features of process models, but also increase the difficulty for annotators to perform the annotation; (2) During the ontology comparison, the assignment of weights are given to semantic relationships, but these weights are given without precise scientific evidence (for example, the weight of "same_as" is 1 and the weight of "kind_of" is 0.8, but where this 0.8 comes from?); (3) A general process ontology (GPO) is used to map different process meta-models. Some model constructs knowledge, which is represented by those meta-models, cannot find the corresponding concepts or relationships in the GPO (for example, the association in BPMN meta-model has not corresponding mapping in the GPO).
- In [117], (1) the semantic annotation is used without formal definitions; (2) the semantic relationships in the semantic annotation are only represented as three types of instance assertions. These semantic relationships require the employed ontologies to provide all the necessary classes that are needed by the assertion process; (3) they used description logics to ensure the process model fulfil the pre-defined constraints. However, they only used four simple axioms to support the type verification of semantic annotations (for example, a BPMN element of type "Activity" can be annotated only with a domain specific concept that is equivalent or more specific than "Activity").

Furthermore, we explore two main directions of the researches on semantic annotations: (1) researches that focus on developing an appropriate knowledge base, which has high-coverage of semantics; (2) researches that focus on discovering a suitable semantic annotation structure model and related mechanisms, which has high-adaptability of different knowledge bases. The challenges in the first direction are mainly the completeness and multiplicity of semantic models. The challenges in the second direction are mainly the applicability, tolerance and formalization of annotation model and related mechanisms. In this thesis, we are more biased toward the second direction.

2.4 Conclusion

In general, based on the investigation in this section, a number of requirements for our proposed solution are identified:

(1) It should provide a general semantic annotation structure model that is able be used to formalize semantic annotations for different kinds of models;

(2) It should discover the possibility of using both structure and domain semantics together in the inference process;

(3) It should provide some mechanisms to assist the detection of the inconsistencies between semantic annotations and the identification of the conflicts between annotated objects;

(4) It should provide a way to guide annotators in how to apply the formal semantic annotations and how to benefit from those semantic annotations;

(5) It should provide a framework to support the semantic enrichment of models along the product life cycle.

In the next chapter, we will propose a formalization of semantic annotations that follows these requirements.

Chapter 3 Formal Approach to the Semantic Annotations

In the previous chapters, we discussed the needs for the semantic enrichment of models in a PLM environment and highlighted that the mutual understanding of semantics in the shared and exchanged knowledge representations is the cornerstone in the quest for semantic interoperability. Semantic annotation is considered as one of the possible solutions for making explicit the implicit semantics that embedded in a knowledge representation and also for giving the possibility to perform semantic reasoning on the annotated objects. Based on the investigation of various current semantic annotation researches, we discovered a number of existing drawbacks and potential challenges and identified a number of requirements for our proposed solution. Three main drawbacks can be summarized as follows:

(1) Lack of a formalization of semantic annotations that is able to be used for the semantic enrichments of different kinds of models;

(2) Lack of a mechanism to combine both structure and domain semantics together to contribute in the inference process.

(3) Lack of a mechanism to assist the detection of inconsistencies between semantic annotations and the identification of conflicts between annotated objects.

In order to address these drawbacks, in this chapter, we propose a formal approach to assist the semantic enrichment of models in a PLM environment. Section 3.1 presents the details of the semantic annotation formalization proposals and identifies the essential elements of a semantic annotation. Then, in Section 3.2, taking advantages from the formalization, the reasoning mechanisms are proposed to support the detection of inconsistencies between semantic annotations and the identification of possible conflicts between model elements. Finally, Section 3.3 presents a semantic annotation framework for addressing the issue of semantic interoperability in a PLM environment.

3.1 Formalization of Semantic Annotations

In many research works, as we discussed in Chapter 2, the essential elements of a semantic annotation are not clearly defined. To better formalize semantic annotations, we first identify the major components of a semantic annotation (Section 3.1.1). Then, we propose two kinds of semantic blocks to support the definition of the semantic annotation and the creation of reasoning rules (Section 3.1.2). At the end, in Section

3.1.3, we propose the formal definitions that related to all the essential elements of a semantic annotation.

3.1.1 Meta Model of the Semantic Annotation

The comprehension of the knowledge that is represented by a model needs not only the domain semantics that is embedded in the contents of the model, but also the structure semantics that is embedded through the modelling constructs. Therefore, the relevant semantics is supposed to be contained in the employed ontologies. The ideal situation is that there exists equivalent semantics in ontologies for every annotated element in a model. However, because of the complexity of the reality, in most of the cases this situation rarely appears. Therefore, a more reasonable relation definition is required for describing the semantic relationships between an element of a target knowledge representation and its corresponding domain and structure semantics. To be more specific, in order to define the meta-model of the semantic annotation, based on the research context and the investigation of existing researches in the previous chapters, several important concepts that are used throughout this chapter need to be reviewed.

Target Knowledge Representation (TKR)

Models in a PLM environment act as an important role to enable the capturing and representation of the relevant knowledge from each product life cycle stage. These models are always expressed in some kinds of modelling languages or notations with designer’s specific peculiarities, such as, different backgrounds, unique knowledge, heterogeneous expertise, particular needs and specific practices. This results in the implicit, or possibly ambiguously explicit, semantics that is not easily intelligible by the humans or the machines. The interoperation process between enterprise systems and stakeholders not only requires that these models can be exchanged and operated on, but also demands the unambiguous understandings of the exchanged models. In this research work, all different kinds of models throughout the product life cycle are considered as Target Knowledge Representations (TKRs).

Ontology-based Knowledge Representation (OKR)

Ontology represents a real-world semantics that enables human to use meaningful terminologies as machine processable contents. It formalizes the common and shared understandings in a human and machine interpretable way [49], which is frequently chosen as a candidate procedure to formalize knowledge. According to the research context, as we discussed in Section 2.2.2, two kinds of ontologies (*PLC-related* *ontologies* and *Meta-model ontologies*) are used to support the semantic enrichment of models in a PLM environment. These ontologies are considered as Ontology-based Knowledge Representations (OKRs) in this research work.

Semantic Annotation Structure Model (SASM)

The Semantic Annotation is acting as a bridge to formally describe the semantic relationships between TKRs and OKRs. Two aspects of semantics are made explicit through a semantic annotation:

(1) The *domain semantics* that describes the context and the meaning of an annotated object in a certain domain;

(2) The *structure semantics* that describes the interrelations of the annotated object and the other related objects in the TKR.

The meta-model of the semantic annotation is presented in the Figure 3-1, which describes the main components of a semantic annotation and their relationships.

- A “Target Knowledge Representation” is the composition of one or more “Element of a TKR”.
- The “Ontology-based Knowledge Representation” is the generalization of the “Meta-model Ontology” and the “PLC-related Ontology”.
- A “Meta-model Ontology” is the composition of one or more “Element of a Meta-model Ontology”.
- A “PLC-related Ontology” is the composition of one or more “Element of a PLC-related Ontology”.
- An “Element of a TKR” can be annotated by zero or more “Semantic Annotation”.
- A “Semantic Annotation” contains one “Structure Semantics”.
- A “Semantic Annotation” contains zero or more “Domain Semantics”.
- A “Structure Semantics” is the aggregation of one “Element of a Meta-model Ontology”
- A “Domain Semantics” is the aggregation of one or more “Element of a PLC-related Ontology”.

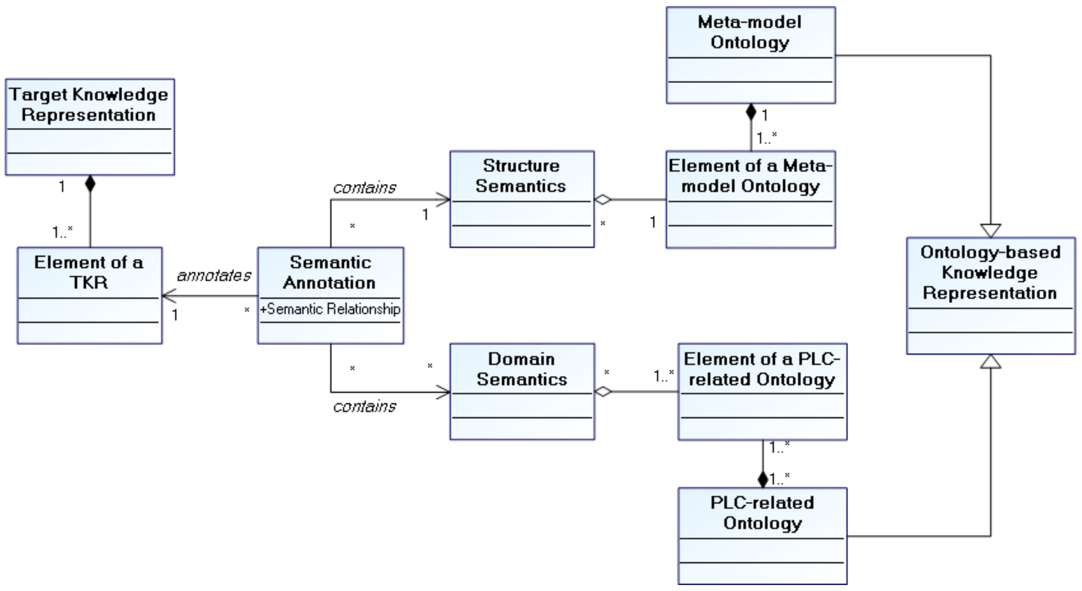


Figure 3-1 The Meta-model of the Semantic Annotation

Based on this meta-model, in the next section, we propose a semantic block delimitation method. This method will be used as a basis to support the proposition of formal definitions and the creation of reasoning rules.

3.1.2 Semantic Block Delimitation

In order to well support semantic annotations, we adapted the concept of "semantic block" proposed by Yahia et al. [118]. In their research, the semantic block concept represents a kind of aggregation of semantics. It is composed by a minimal number of mandatory concepts needed to express the full semantics of a concept. In our work, we propose a delimitation method to create semantic blocks. It, not only, extends the semantic block definition to cover the relations among those selected concepts, but also further categorizes the semantic blocks into two different kinds for facilitating annotation and reasoning processes.

Generally, both ontologies and enterprise models can be regarded as the composition of a set of entities (such as concepts, instances or model elements) and the corresponding explicit or implicit relations that are used to bind them together for some specific purposes. For example, Figure 3-2 (a) depicts a part of an ontology that contains explicit relations and Figure 3-3 (a) shows a part of a process model that contains implicit relations. A semantic block is considered as a shape (segment) of a model that contains a number of selected entities and corresponding relations among them. Each semantic block represents an aggregation of semantics.

According to the objects that the semantic block delimitation method applies to and the usages of those semantic blocks, two kinds can be categorized:

(1) *Semantic Blocks for Semantics Description*: the delimitation method supports the creation of a "Domain Semantics" by delimitating one or more "Element of a PLC-related Ontology" from one or more "PLC-related Ontology". The generated semantic block is used to describe the domain semantics of an "Element of a TKR" based on the semantics that it aggregates.

(2) *Semantic Blocks for Semantics Substitution*: the delimitation method supports the creation of a substitute by delimitating one or more "Element of a TKR" from one "Target Knowledge Representation" based on the "Structure Semantics" that they express. The produced semantic block is used as a substitute of those "Element of a TKR" it aggregates and acts as a new entity or a new relation in the "Target Knowledge Representation".

Let A be a set of entities in a model. Let $B \subseteq A \times A$ be a set of binary relations. Given $a_i, a_j \in A$, we say that a_i is relative to a_j through $b_{i,j} = \left(a_i, a_j\right) \in B$. We call a_i the domain of $b_{i,j}$ and a_j the range of $b_{i,j}$. The general definitions of above-mentioned two kinds of semantic blocks are presented in the following sections.

3.1.2.1 Semantic Blocks for Semantics Description

Since the relations among entities in ontologies are already explicit, the delimitation of semantic blocks is able to be applied directly.

Let $a_{i_0} \in A$ be the main entity and $A_{a_{i_0}} \subseteq A$ be a set that contains all the entities with selected relations, which are associated to a_{i_0}. Let $B_{a_{i_0}} \subseteq B$ be a set that contains those selected relations. Mathematically, $A_{a_{i_0}}$ is defined as:

$$A_{a_{i_0},0} = \{a_{i_0}\};$$
$$A_{a_{i_0},1} = \{a_{i_1} \in A | \exists b_{i_0,i_1} \in B_{a_{i_0}}, a_{i_0} \in A_{a_{i_0},0}, \left(a_{i_0}, a_{i_1}\right) = b_{i_0,i_1}\};$$
$$A_{a_{i_0},2} = \{a_{i_2} \in A | \exists b_{i_1,i_2} \in B_{a_{i_0}}, a_{i_1} \in A_{a_{i_0},1}, \left(a_{i_1}, a_{i_2}\right) = b_{i_1,i_2}\};$$
$$\ldots$$
$$A_{a_{i_0},n} = \{a_{i_n} \in A | \exists b_{i_{n-1},i_n} \in B_{a_{i_0}}, a_{i_{n-1}} \in A_{a_{i_0},n-1}, \left(a_{i_{n-1}}, a_{i_n}\right) = b_{i_{n-1},i_n}\};$$
$$A_{a_{i_0}} := \bigcup_n A_{a_{i_0},n}$$

According to user-defined reasoning rules (for example, only certain relations that fulfil the constraints in rules can be the relations in $B_{a_{i_0}}$) that are applied during the creation of a semantic block, an appropriate subset of the A can be determined. Then the semantic block of the entity a_{i_0} is defined as a pair:

$$SB_{a_{i_0}} := (A_{a_{i_0}}, B_{a_{i_0}}),$$

where every entity in $A_{a_{i_0}}$ can be attained by a_{i_0} through, at least, one path. All the relations in the paths are contained in $B_{a_{i_0}}$.

Figure 3-2 (b) shows an example of the semantic block for semantics description. SB_{a_7} is the semantic block of the main entity a_7, which can be used to describe the domain semantics of its annotated object based on the semantic it aggregates.

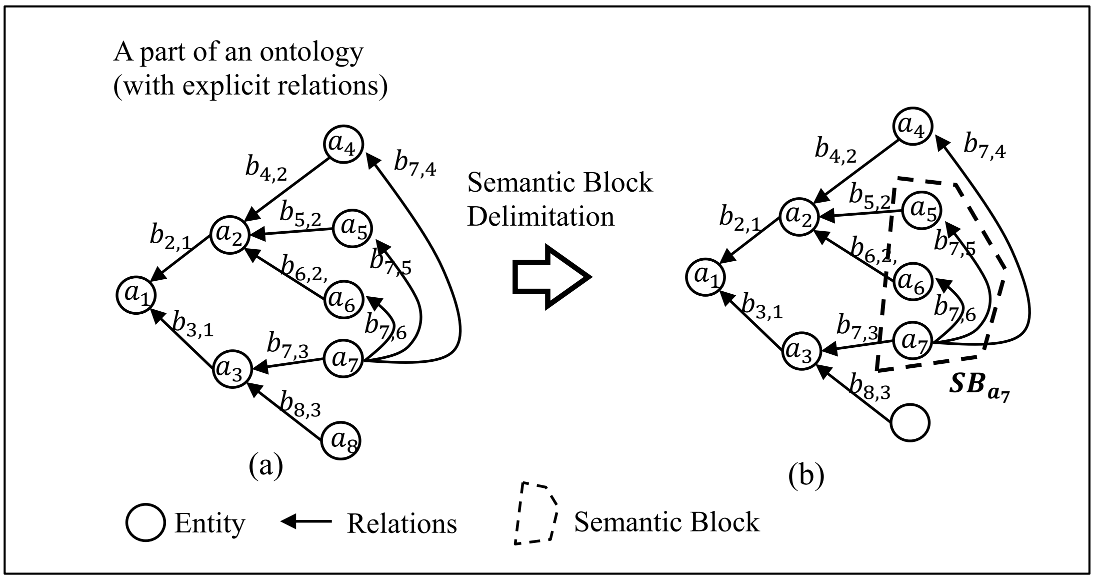


Figure 3-2 An Example of the Semantic Block for Semantics Description

3.1.2.2 Semantic Block for Semantics Substitution

Due to the fact that relations among the entities are implicit in enterprise models, the delimitation of semantic blocks cannot be applied directly. As shown in the Figure 3-3, we propose two procedures as follows:

(1) Relation Explicitation

The relation explicitation process focuses its interest on making explicit the implicit relations among elements in an enterprise model. Two general rules are proposed:

- Each model element is represented as an entity of the set A.
- A relation $b_{i,j} = (a_i, a_j) \in B$ is created between $a_i \in A$ and $a_j \in A$, if the model element that is represented by a_i is related to the model element that is represented by a_j.

(2) Semantic Block Delimitation

This kind of semantic blocks can be further divided into two categories depending on the role of a semantic block acting: as an entity or as a relation. Based on user-defined reasoning rules, a semantic block that is composed by a set of entities and the corresponding relations among them can be created. A number of restrictions to assist the creation of reasoning rules are presented as follows:

For a semantic block that acts as a new entity a_x, let $A_{a_x} \subseteq A$ be a set of selected entities and let $B_{a_x} \subseteq B$ be a set of relations among those entities in A_{a_x}. In order to substitute the semantics of its contents, the following two conditions are required:

- For every entity a_i in A_{a_x}, at least one entity a_j exists in A_{a_x} that has a relation $b_{i,j}$ in B_{a_x} to a_i. That is

$$\forall a_i \in A_{a_x},\ \exists a_j \in A_{a_x},\ \exists b_{i,j} \in B_{a_x},\ \ s.t.\ (a_i, a_j) = b_{i,j}\ .$$

- For every binary relation $b_{i,j}$ in B_{a_x}, the entities that appear in the domain and range of $b_{i,j}$ are the entities in the A_{a_x}. That is

$$\forall b_{i,j} \in B_{a_x},\ (a_i, a_j) = b_{i,j}\ \Rightarrow\ a_i, a_j \in A_{a_x}\ \ .$$

Then the semantic block SBE_{a_x} is defined as a pair:

$$SBE_{a_x} \coloneqq (A_{a_x}, B_{a_x}).$$

For a semantic block that acts as a new relation between $a_i \in A$ and $a_j \in A$. Let $A_{a_i,a_j} \subseteq A$ be a set of selected entities and let $B_{a_i,a_j} \subseteq B$ be a set of relations among a_i, a_j and the entities in A_{a_i,a_j}. In order to substitute the semantics of its contents, it needs to satisfy the following three conditions:

- A_{a_i,a_j} does not contain a_i and a_j. That is

$$a_i, a_j \notin A_{a_i,a_j}.$$

- For every entity a_k in the A_{a_i,a_j}, at least one entity a_l exists in A_{a_i,a_j} that has a relation $b_{k,l}$ in B_{a_i,a_j} to a_k. That is

$$\forall a_k \in A_{a_i,a_j},\ \exists a_l \in A_{a_i,a_j},\ \exists b_{k,l} \in\ B_{a_i,a_j}\ ,\ \ s.t.\ (a_k, a_l) = b_{k,l}.$$

- Besides a_i and a_j, for every binary relation $b_{k',l'}$ in the B_{a_i,a_j}, the entities that appear in the domain and range of $b_{k',l'}$ are the entities in the A_{a_i,a_j}. That is

$$\forall b_{k',l'} \in B_{a_i,a_j},\ \ (a_{k'}, a_{l'}) = b_{k',l'}\ \Rightarrow\ a_{k'}, a_{l'} \in A_{a_i,a_j} \cup \{a_i\} \cup \{a_j\}$$

Then the semantic block SBR_{a_i,a_j} is defined as a pair:

$$SBR_{a_i,a_j} := (A_{a_i,a_j}, B_{a_i,a_j})$$

Figure 3-3 (c) shows an example of the semantic block for semantics substitution. SBR_{a_1,a_4} is the semantic block that merges the semantics of its contents and acts as a new relation between a_1 and a_4 .

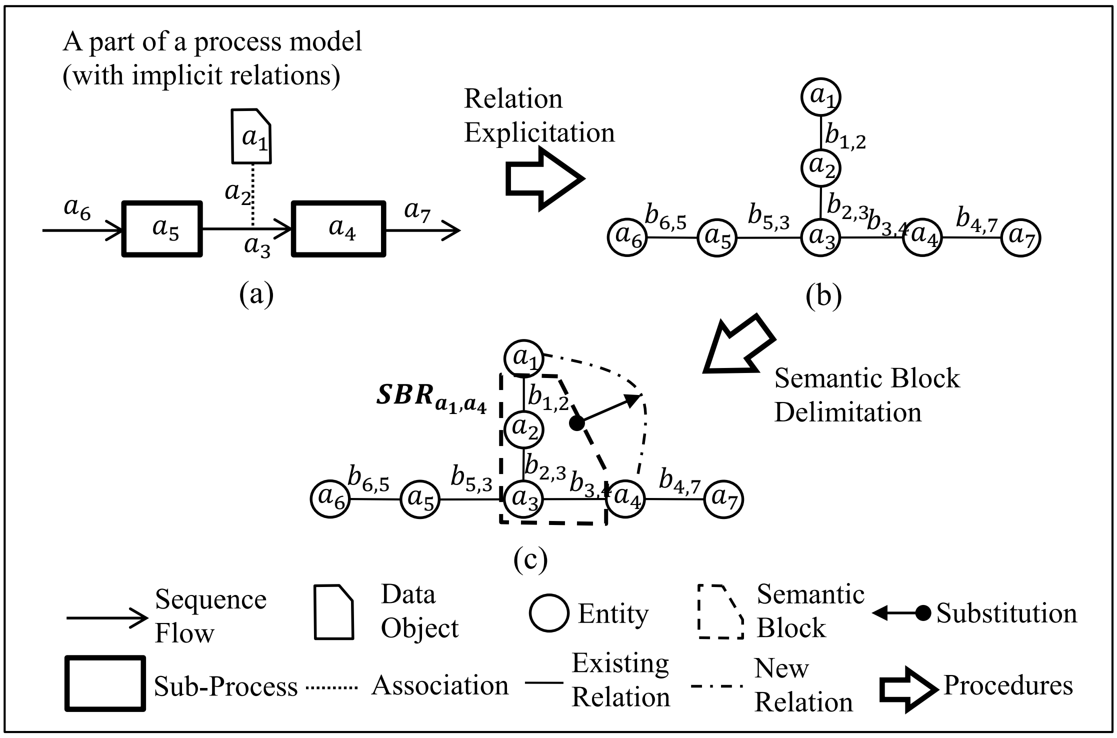


Figure 3-3 An Example of the Semantic Block for Semantics Substitution

The semantic blocks delimitation can be used to support the semantic annotation from the following two aspects:

(1) Taking advantage from the semantic block for semantics description, the annotators can, with a certain degree of freedom, delimitate an appropriate semantics that they need in the OKRs. Furthermore, it also brings the capability to adopt the diversity of different ontologies

(2) Taking advantage from the semantic block for semantics substitution, a combination of elements in the TKR can be delimitated. The produced semantic blocks act as new entities or new relations to assist the creation of reasoning rules. In this research work, we mainly uses the latter category, namely SBR_{a_i,a_j}.

3.1.3 Formal Definitions of Semantic Annotation

In this section, based on the definition proposed in [14] and [15], an improved version of the formal definitions is presented. Let E be the set of elements in a TKR and e_i be one of the elements in E (e_i is considered as an instance of the constructs of the meta-model that are used to design a model).

Definition 1. An ontology is a formal and shared understanding of some domains of interest, which specifies the concepts and the relationships that can exist for an agent or a community of agents [49] [67]. Let o_x represent an ontology, which is formalized by a triple:

$$o_x \coloneqq (\mathrm{C}_{o_x}, \mathrm{R}_{o_x}, \mathrm{A}_{o_x}),$$

where

- C_{o_x} is a set of concepts;
- R_{o_x} is a set of relationships;
- A_{o_x} is a set of axioms.

Let $oall_{o_x}$ be the set that contains all the elements from the set $C_{o_x}$and R_{o_x}. An ontology element $oe_{o_{x_y}}$ is represented as:

$$oall_{o_x} = \{oe_{o_{x_y}} | oe_{o_{x_y}} \in C_{o_x} \cup R_{o_x}\}.$$

Remark 1. The ontology elements that are used as part of the semantic annotation contents are the concepts and the relationships. The axioms only participate in the reasoning stage.

Definition 2. A meta-model is a model that specifies the concepts, relationships and rules to model a model. Let mm_x denote a meta-model, which is defined as a triple:

$$\mathrm{mm_x} \coloneqq (\mathrm{C}_{\mathrm{mm_x}}, \mathrm{R}_{\mathrm{mm_x}}, \mathrm{RU}_{\mathrm{mm_x}}),$$

where

- C_{mm_x} is a set of concepts;
- R_{mm_x} is a set of relationships;
- RU_{mm_x} is a set of rules.

Let mmo_x be an ontology that represents the meta-model $\mathrm{mm_x}$, which is defined as:

$$mmo_x \coloneqq (C_{mmo_x}, R_{mmo_x}, A_{mmo_x}).$$

Remark 2. In scientific literature there are at least two different visions about whether a meta-model is an ontology or not. In this work, we will take into account only those elements (the concepts and the relationships) that are needed for describing the interrelations among the annotated objects. Therefore we consider that a meta-model can be represented as an ontology. The literatures in Section 2.2.2.2 also prove this standpoint. However, because of the different representation rules and methods used by different researches, such as the research [96], [97] and [98], in this definition we will not discuss the matching between a meta-model and an ontology .

Definition 3. The domain semantics of a TKR is made explicit by one or more PLC-related ontologies. Let PO be the set of PLC-related ontologies and P be the set of selected ontology element sets from the powerset of all ontology elements of the PO, which is defined as:

$$\bigcup_{o_x \in PO} oall_{o_x} = \{oe_{o_{x_y}} | (\exists o_x)(o_x \in PO \land oe_{o_{x_y}} \in oall_{o_x})\},$$

$$P \subseteq \mathcal{P}\left(\bigcup_{o_x \in O} oall_{o_x}\right).$$

Remark 3. In a different way from other semantic annotation methods, each e_{i} from the TKR is annotated by a set of ontology elements that are delimitated by a semantic block (the semantic block for semantics description), which contains ontology elements from one or more PLC-related ontologies. Given $p_j \in P$, let us define p_j as a set of ontology elements that represents the appropriated semantics for describing the domain semantics of an e_{i}. This set is created by an annotator, or by a mechanism (such as the semantic block delimitation method introduced in section 3.1.2),

Definition 4. The structure semantics of a TKR is made explicit by a meta-model ontology mmo_x. Let MME be the set that contains all the elements from the set C_{mmo_x}. An ontology element mme_l is defined as:

$$MME := \{mme_l | mme_l \in C_{mmo_x}\}.$$

Remark 4. Each e_i from the TKR is annotated by one ontology element from the C_{mmo_x} of a meta-model ontology mmo_x. The relationships in R_{mmo_x} are not used for the annotation process, but they are used for defining the relationships between the annotated objects in the TKR.

Definition 5. Let A and B be two sets, any subset of $br \subseteq A \times B$ is a binary relation from A to B. Given $a \in A$ and $b \in B$, the br in the notation $a\ br\ b$ is defined as,

$$br := \{(a,b) | a\ is\ in\ the\ relation\ br\ with\ b\}.$$

Let $dom(br)$ represent the domain of the br and $ran(br)$ represent the range of the br, which are defined as

$$dom(br) := \{a \in A | \exists b \in B, (a,b) \in br\},$$

$$ran(br) := \{b \in B | \exists a \in A, (a,b) \in br\}.$$

Definition 6. $SR_{E,P}$ is a set of binary relations that describe the semantic relationships from E to P. Given, $e_i \in E$ and $p_j \in P$, and let $sem(e_i)$ represent the semantics of e_i and $sem(p_j)$ represent the semantics of p_j, we then define five specializations of the $SR_{E,P}$ as follows:

$$sr_{\sim} := \{(e_i, p_j) | sem(e_i)\ and\ sem(p_j)\ are\ equivalent\};$$

$$sr_{\sqsupset} := \{(e_i, p_j) | sem(e_i)\ is\ more\ general\ than\ sem(p_j)\};$$

$$sr_{\subset} := \{(e_i, p_j) | sem(e_i)\ is\ \ less\ general\ than\ sem(p_j)\};$$

$$sr_{\cap} := \{(e_i, p_j) |\ e_i\ and\ p_j\ have\ common\ semantics, (e_i, p_j) \notin sr_{\sim} \cup sr_{\supset} \cup sr_{\subset}\};$$

$$sr_{\perp} := \{(e_i, p_j) |\ e_i\ and\ p_j\ have\ not\ common\ semantics\}.$$

Remark 6. Five kinds of general semantic relationships that describe the relations from an "Element of a TKR" to a "Domain Semantics" are proposed as follows:

- The "*is equivalent to*" relation (marked as "$\sim$") denotes the fact that the semantics of two related elements from the domain and the range are equivalent. In this definition, it is $\mathrm{sr}_{\sim}$.
- The "*subsumes*" relation (marked as "$\supset$") denotes the fact that the semantics of an element from the domain is more general than the semantics of its related element from the range. In this definition, it is $\mathrm{sr}_{\supset}$.
- The "*is subsumed by*" relation (marked as "$\subset$") denotes the fact that the semantics of an element from the domain is less general than the semantics of its related element from the range. In this definition, it is $\mathrm{sr}_{\subset}$.
- The "*intersects*" relation (marked as "$\cap$") denotes the fact that the related two elements from the domain and the range have only a part of common semantics. In this definition, it is $\mathrm{sr}_{\cap}$.
- The "*is disjoint with*" relation (marked as "$\perp$") denotes the fact that the two related elements from the domain and the range have not common semantics. In this definition, it is $\mathrm{sr}_{\perp}$.

Definition 7. $MR_{E,MME}$ is a set of binary relations that describe the semantic relations from E to MME. Given $e_i \in E$ and $mme_l \in MME$, we then define one subset of MR as follow:

$$mr_{io} := \{(e_i, mme_l) |\ e_i\ is\ an\ instance\ of\ mme_l\}.$$

Remark 7. The semantic relationship that describes the relation from an "Element of a TKR" and a "Structure Semantics" is proposed as follows:

- The "*is instance of*" relation denotes that an element from the domain is the instance of its related element from the range. In this definition, it is mr_{io}.

Finally, with all above-mentioned definitions, we are now ready to formally define a semantic annotation.

Definition 8. Let TKR, PO and mmo_x be given, the semantic annotation SA that is associated to them is defined by a 5-tuple:

$$SA := (E, P, SR, MME, MR),$$

where

- E is a set of elements from a TKR;
- P is a set of selected ontology element sets from a set of PLC-related ontologies PO, which makes explicit the domain semantics aspect of the E;
- MME is a set of ontology elements from a meta-model ontology mmo_x, which makes explicit the structure semantics aspect of the E;
- $SR \coloneqq SR_{E,P}$;
- $MR \coloneqq MR_{E,MME}$.

We then associate SA with the following sets:

$$\mathcal{S}_{SA} \coloneqq E \times P \times SR \times MME \times MR,$$

which means explicitly:

$$\mathcal{S}_{SA} \coloneqq \{(e_i, p_j, sr_k, mme_l, mr_k) | e_i \in E , p_j \in P, sr_k \in SR, mme_l \in MME, mr_k \in MR\}.$$

An element $sa_x \in SA$ is then defined as:

$$sa_x \coloneqq (e_i, p_j, sr_k, mme_l, mr_k)$$

The semantic relationships between a TKR and one or more OKRs that are formalized by these formal definitions not only can be used to construct a semantic annotation model, but also can be used as the foundation for the creation of reasoning mechanisms.

3.2 Reasoning Mechanisms

In this work, the formalized semantic annotations mainly contribute to two main aspects: for assisting the creation of models and for supporting the identification of possible mistakes. Therefore, we set three main stages, with their corresponding mechanisms, for achieving these two purposes:

(1) The suggestion of semantic annotations;
(2) The inconsistency detection between semantic annotations;
(3) The conflict identification between annotated objects in a model.

In this section, the first stage is presented in Section 3.2.1 and the last two stages are presented in Section 3.2.2.

3.2.1 Suggestion of Semantic Annotations

In the oxford dictionary online, the term inconsistency is defined as "*the fact or* *state of being inconsistent*". The essence of an inconsistency is the contradictory among two or more facts that describe one common object. With the same principle, the inconsistency detection between semantic annotations is based on the comparison of two or more semantic annotations that describe the semantics of the same "Element of a TKR". Therefore, to cope with this premise, two types of semantic annotations are classified as follows:

- *Initial Semantic Annotations*, which are directly annotated on an "Element of a TKR" by an annotator;
- *Inferred Semantic Annotations*, which are suggested to annotate an "Element of a TKR" through the inference action that is based on its related element's semantic annotations and reasoning rules.

Both "Structure Semantics" and "Domain Semantics", which are made explicit by the semantic annotations, contribute in the annotation suggestion stage.

The "Structure Semantics" makes explicit the implicit relations between the annotated "Element of a TKR" and its related elements. Let E be a set of Elements in the TKR and mmo_x be the meta-model ontology that makes explicit the structure semantics of those elements. The following procedure is used to create a semantic block for semantics substitution, named SBR_{a_i,a_j}, as an example:

(1) Let the elements in E be annotated by the concepts in C_{mmo_x}(the set of concepts in mmo_x). Through the semantic relationship mr_{io}, these annotated elements are treated as instances of their corresponding concepts. The interrelations between two related instances are made explicit through the relationships in R_{mmo_x} (the set of relationships in mmo_x).

(2) Let select two concepts in mmo_x for the creation of the semantic block. Let $moc_i, moc_j \in C_{mmo_x}$ be these two selected concepts.

(3) Let select a set of concepts $A_{moc_i,moc_j} = \{moc_1 \dots moc_{n'}\} \subseteq C_{mmo_x}$ and a set of relationships $B_{moc_i,moc_j} = \{mor_1 \dots mor_{m'}\} \subseteq R_{mmo_x}$, which are the relationships among the concepts in A_{moc_i,moc_j}. The selection process need to satisfy the following three conditions:
   - $moc_i, moc_j \notin A_{moc_i,moc_j}$;
   - $\forall moc_k \in A_{moc_i,moc_j}, \exists moc_l \in A_{moc_i,moc_j}, \exists mor_z \in B_{moc_i,moc_j},$ $s.t.\ (moc_k, moc_l) = mor_z$;
   - $\forall mor_z \in B_{moc_i,moc_j},\ (moc_{k'}, moc_{l'}) = mor_z \Rightarrow moc_{k'}, moc_{l'} \in A_{moc_i,moc_j} \cup \{moc_i\} \cup \{moc_j\},$.

(4) Finally, the rule to delimitate the semantic block SBR_{moc_i,moc_j} can be created as follows:

$$mr_{io}(?a, moc_i), mr_{io}(?b, moc_1), \ldots, mr_{io}(?c, moc_n), mr_{io}(?d, moc_j),$$
$$mor_1(?a, ?b), \ldots, mor_m(?c, ?d) \rightarrow SBR_{moc_i, moc_j}(?a, ?d).$$

SBR_{moc_i, moc_j} acts as a new relationship from certain instances of moc_i to certain instances of moc_j, which fulfil all the conditions in the rule.

Because it is possible to have several different combinations of concepts and relationships that are selected to create the semantic block, moc_i and moc_j might have multiple SBR_{moc_i, moc_j}. In this case, these SBR_{moc_i, moc_j} should be named differently.

The "Domain Semantics" makes explicit the meaning of an annotated "Element of a TKR" in a domain of interest, which is used as the basis for the annotation suggestion. Let E be a set of elements in the TKR and PO be the set of PLC-related ontologies for making explicit the domain semantics of the TKR. The procedure to suggest a semantic annotation is listed in the following steps:

(1) Let $e_x, e_y \in E$ be two elements in the TKR. e_x is an instance of moc_i and e_y is an instance of moc_j. Let us assume that besides the interrelations that are made explicit by the corresponding relationships in R_{mmo_x}, e_x and e_y have a new relationship SBR_{moc_i, moc_j} between them.

(2) Let $poc_{i'}$ be a concept from the PO and acts as the main entity of the semantic block $SB_{poc_{i'}}$ (we named the "main entity" of a semantic block for semantics description as the "main concept" in the following of the thesis). Let $A_{poc_{i'}} = \{poc_1 \ldots poc_{n'}\}$ be the set of selected concepts from PO in $SB_{poc_{i'}}$. Let $B_{poc_{i'}} = \{por_1 \ldots por_{m'}\}$ be the set of corresponding relationships in $SB_{poc_{i'}}$.

(3) Let $e_x \in E$ be annotated by $SB_{poc_{i'}}$ through the semantic relationship $sr_{\sim}$ or $sr_{\subset}$.

(4) Let select a relationship $por_{z'} \in B_{poc_{i'}}$ and associate it with the new relationship SBR_{moc_i, moc_j}. If there is a concept $poc_{j'}$ that satisfies $(poc_{i'}, poc_{j'}) = por_{z'}$, a new semantic block $SB_{poc_{j'}}$ can be generated. It takes $poc_{j'}$ as starting point. The traverse that builds $A_{poc_{j'}}$ is based on $A_{poc_{i'}}$ and $B_{poc_{i'}}$ in $SB_{poc_{i'}}$ in the following steps:

$A_{poc_{j'},0} = \{poc_{j'}\};$

$A_{poc_{j'},1} = \{a_{i_1} \in A_{poc_{i'}} \mid \exists b_{i_0,i_1} \in B_{poc_{i'}}, a_{i_0} \in A_{poc_{j'},0}, (a_{i_0}, a_{i_1}) = b_{i_0,i_1}\};$

$A_{poc_{j'},2} = \{a_{i_2} \in A_{poc_{i'}} \mid \exists b_{i_1,i_2} \in B_{poc_{i'}}, a_{i_1} \in A_{poc_{j'},1}, (a_{i_1}, a_{i_2}) = b_{i_1,i_2}\};$

$\ldots$

$A_{poc_{j'},n} = \{a_{i_n} \in A_{poc_{i'}} \mid \exists b_{i_{n-1},i_n} \in B_{poc_{i'}}, a_{i_{n-1}} \in A_{poc_{j'},n-1}, (a_{i_{n-1}}, a_{i_n}) = b_{i_{n-1},i_n}\};$

$$A_{poc_{j'}} := \bigcup_n A_{poc_{j'},n}$$

Let $B_{poc_{j'}} \subseteq B_{poc_{i'}}$ be the set of relationships that appear during the creation of $A_{poc_{j'}}$, then $SB_{poc_{j'}}$ is created.

(5) Finally, the $SB_{poc_{j'}}$ is suggested to annotate e_y through the semantic relationship $sr_{\subset}$.

Two remarks need to be pointed out: (1) only the semantic relationship $sr_{\subset}$ and $sr_{=}$ can produce suggestions; (2) the semantic blocks that are nested within each other are not taken into account. Based on the initial semantic annotations and the suggestion rules, a number of inferred semantic annotations can be suggested by the annotation suggestion mechanism.

3.2.2 Inconsistency Detection and Conflict Identification

Once the same annotated object has two or more semantic annotations, the detection of inconsistencies can be performed. Using the case of inconsistency detection between two semantic annotations as the basis, let e_i be annotated by sa_x and sa_y, in which, p_x and p_y are used to make explicit the domain semantics of e_i. The semantic similarity comparison results between p_x and p_y can be categorized into five types (similar with what we defined in SR), so as to formally represent them, we employs the binary relation that we defined in the definition 5.

Definition 9. PR is a binary relation that describes the semantic relationships from P to P. Given $p_x, p_y \in P$, and let $sem(p_y)$ represent the semantics of p_x and $sem(p_y)$ represent the semantics of p_y, we then define five subsets of the PR as follows:

$$pr_{\sim} := \{(p_x, p_y) | sem(p_x) \ and \ sem(p_y) \ are \ equivalent\};$$

$$pr_{\supset} := \{(p_x, p_y) | sem(p_x) \ is \ more \ general \ than \ sem(p_y)\};$$

$$pr_{\subset} := \{(p_x, p_y) | sem(p_x) \ is \ less \ general \ than \ sem(p_y)\};$$

$$pr_{\cap} := \{(p_x, p_y) | \ p_x \ and \ p_y \ have \ common \ semantics, (p_x, p_y) \notin pr_{\sim} \cup pr_{\supset} \cup pr_{\subset}\};$$

$$pr_{\perp} := \{(p_x, p_y) | \ p_x \ and \ p_y \ have \ not \ common \ semantics\}.$$

As shown in the Table 3-1, according to the similarity comparison between two domain semantics of a common annotated object, three types of results can be identified as follows:

- result (a) expresses that sa_x and sa_y are consistent with each other;
- result (b) expresses that sa_x and sa_y are possible consistent with each other;
- result (c) expresses that there is an inconsistency between sa_x and sa_y

In order to make the contents in the table more succinct, we use the concept "Others" to replace the rest of the semantic relationships in the PR besides the one or several that are shown in a grid of the table.

Table 3-1 The Possible Results of the Inconsistency Detection between Semantic Annotations

	$e_i\ sr_{\sim}\ p_x$	$e_i\ sr_{\sqsupset}\ p_x$	$e_i\ sr_{\sqsubset}\ p_x$	$e_i\ sr_{\cap}\ p_x$	$e_i\ sr_{\perp}\ p_x$
$e_i\ sr_{\sim}\ p_y$	(a) $p_x\ pr_{\sim}\ p_y$ (c) Others	(a) $p_x\ pr_{\sqsubset}\ p_y$ (c) Others	(a) $p_x\ pr_{\sqsupset}\ p_y$ (c) Others	(a) $p_x\ pr_{\cap}\ p_y$ (c) Others	(a) $p_x\ pr_{\perp}\ p_y$ (c) Others
$e_i\ sr_{\sqsupset}\ p_y$	(a) $p_x\ pr_{\sqsupset}\ p_y$ (c) Others	(b) $p_x\ pr_{\sim}\ p_y$ $p_x\ pr_{\sqsupset}\ p_y$ $p_x\ pr_{\sqsubset}\ p_y$ $p_x\ pr_{\cap}\ p_y$ $p_x\ pr_{\perp}\ p_y$	(a) $p_x\ pr_{\sqsupset}\ p_y$ (c) Others	(b) $p_x\ pr_{\sqsupset}\ p_y$ $p_x\ pr_{\cap}\ p_y$ $p_x\ pr_{\perp}\ p_y$ (c) Others	(a) $p_x\ pr_{\perp}\ p_y$ (c) Others
$e_i\ sr_{\sqsubset}\ p_y$	(a)$p_x\ pr_{\sqsubset}\ p_y$ (c) Others	(a) $p_x\ pr_{\sqsubset}\ p_y$ (c) Others	(b) $p_x\ pr_{\sim}\ p_y$ $p_x\ pr_{\sqsubset}\ p_y$ $p_x\ pr_{\sqsupset}\ p_y$ $p_x\ pr_{\cap}\ p_y$ (c) Others	(b) $p_x\ pr_{\sqsubset}\ p_y$ $p_x\ pr_{\cap}\ p_y$ (c) Others	(b) $p_x\ pr_{\sqsubset}\ p_y$ $p_x\ pr_{\cap}\ p_y$ $p_x\ pr_{\perp}\ p_y$ (c) Others
$e_i\ sr_{\cap}\ p_y$	(a) $p_x\ pr_{\cap}\ p_y$ (c) Others	(b) $p_x\ pr_{\sqsubset}\ p_y$ $p_x\ pr_{\cap}\ p_y$ $p_x\ pr_{\perp}\ p_y$ (c) Others	(b) $p_x\ pr_{\sqsupset}\ p_y$ $p_x\ pr_{\cap}\ p_y$ (c) Others	(b) $p_x\ pr_{\sim}\ p_y$ $p_x\ pr_{\sqsupset}\ p_y$ $p_x\ pr_{\sqsubset}\ p_y$ $p_x\ pr_{\cap}\ p_y$ $p_x\ pr_{\perp}\ p_y$	(b) $p_x\ pr_{\sqsubset}\ p_y$ $p_x\ pr_{\cap}\ p_y$ $p_x\ pr_{\perp}\ p_y$ (c) Others
$e_i\ sr_{\perp}\ p_y$	(a) $p_x\ pr_{\perp}\ p_y$ (c) Others	(a) $p_x\ pr_{\perp}\ p_y$ (c) Others	(b) $p_x pr_{\sqsupset}\ p_y$ $p_x\ pr_{\cap}\ p_y$ $p_x\ pr_{\perp}\ p_y$ (c) Others	(b)$p_x\ pr_{\sqsupset}\ p_y$ $p_x\ pr_{\cap}\ p_y$ $p_x\ pr_{\perp}\ p_y$ (c) Others	(b) $p_x\ pr_{\sim}\ p_y$ $p_x\ pr_{\sqsupset}\ p_y$ $p_x\ pr_{\sqsubset}\ p_y$ $p_x\ pr_{\cap}\ p_y$ $p_x\ pr_{\perp}\ p_y$

The inconsistency detection results not only point out the inconsistencies (or possible inconsistencies) between two (or more) semantic annotations, but also can be used to identify the possible conflicts between those annotated elements in a TKR.

Using the case of conflict identification between two annotated elements in a TKR as the basis, we assume that there is an inconsistency between sa_x and sa_y that are both used to annotate e_i. Meanwhile, e_j and e_k are two other elements in the TKR. As shown in the Table 3-2, based on the types of sa_x and sa_y (initial semantic annotation or inferred semantic annotation, as we stated in Section 3.2.1), a possible conflict between two annotated elements in the TKR can be identified.

Table 3-2 The possible Results of Conflict Identification between two annotated Elements in a TKR

The sa_x on e_i is an / The sa_y on e_i is an	Initial Semantic Annotation	Inferred Semantic Annotation (inferred from the semantic annotation of e_j)
Initial Semantic Annotation		Between e_i and e_j, one of them is possibly wrong
Inferred Semantic Annotation (inferred from the semantic annotation of e_k)	Between e_i and e_k, one of them is possibly wrong	Between e_j and e_k (if $e_j \neq e_k$): one of them is possibly wrong

In order to apply the above-mentioned semantic annotation proposal in a PLM environment, a semantic annotation framework that contains a general semantic annotation procedure and an overall architecture is presented in next section.

3.3 The Semantic Annotation Framework

In this section, the semantic annotation framework for capturing, representing and managing the knowledge related to the system of interest through the semantic annotation is presented. Section 3.3.1 presents the main procedures for applying the semantic annotations. Section 3.3.2 gives an overall architecture of the framework together with the descriptions of its four main modules.

3.3.1 Semantic Annotation Procedure

Taking advantages from the formal definitions and the three main mechanisms presented in previous sections, as shown in the Figure 3-4, a general overview of the procedures for applying the semantic annotations is presented. This workflow is divided into three main phases: The Preparation Phase, The Annotation Phase and The Reasoning Phase.

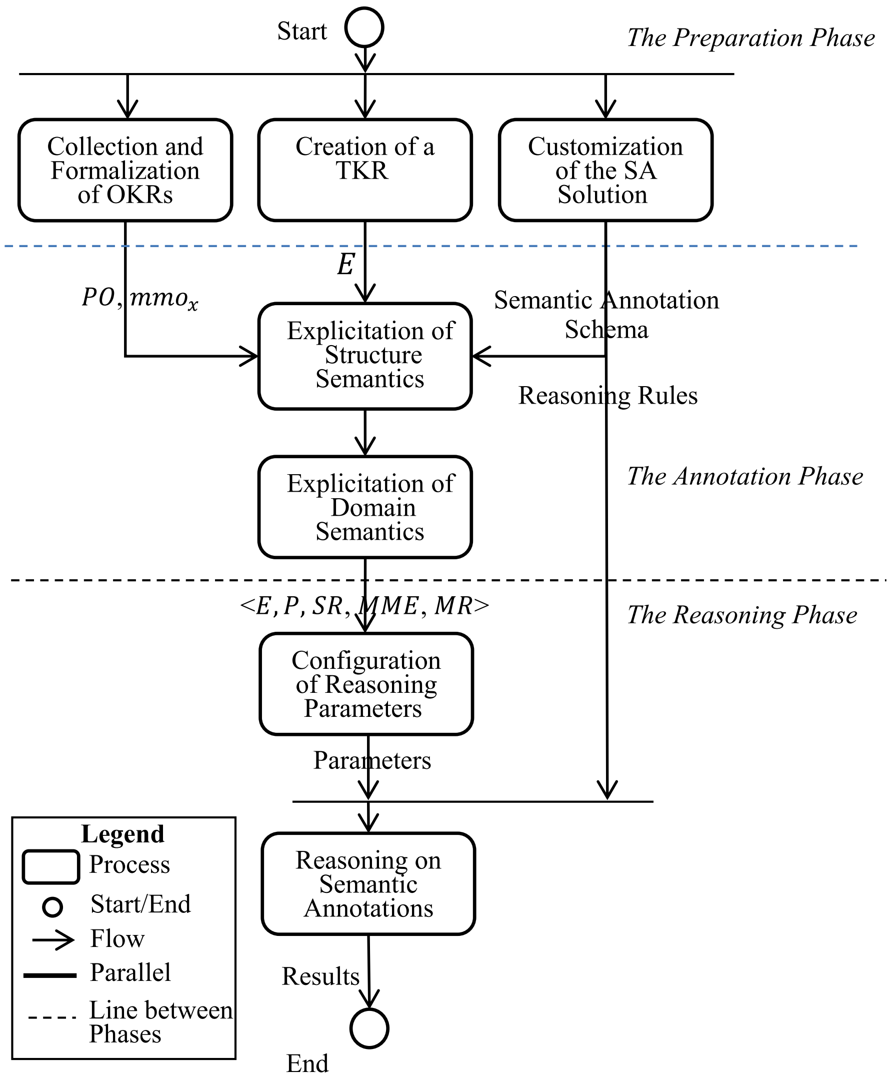

Start
The Preparation Phase
Collection and Formalization of OKRs
Creation of a TKR
Customization of the SA Solution
E
PO, mmo_x
Semantic Annotation Schema
Explicitation of Structure Semantics
Reasoning Rules
Explicitation of Domain Semantics
The Annotation Phase
$<E, P, SR, MME, MR>$
The Reasoning Phase
Configuration of Reasoning Parameters
Parameters
Legend
Process
Start/End
Flow
Parallel
Line between Phases
Reasoning on Semantic Annotations
Results
End

Figure 3-4 General Semantic Annotation Procedure.

The Preparation Phase: during this phase, all the elements that are needed by both the annotation phase and the reasoning phase are prepared.

(1) *Creation of a TKR*, in which, a model, namely a Target Knowledge Representation (TKR), is created by a modelling system. The set of elements E in this TKR are the output of this process.

(2) *Collection and Formalization of OKRs*, in which, the ontologies, namely Ontology-based Knowledge Representations (OKRs), for making explicit the domain semantics and structure semantics are captured and formalized. The output of this process is a number of PLC-related ontologies (PO) and a meta-model ontology (mmo_x). The selection of an ontology can be based on some ontology evaluation methods[119][120].

(3) *Customization of the SA Solution*, in which, the formal definitions of semantic annotations and the reasoning mechanisms are used as the basis for

customizing a semantic annotation model and the reasoning rules. The output of this process is divided into two parts: the *semantic annotation schema* that can be used as a repository to conserve the objects of semantic annotation (E), contents of semantic annotation (P and MME) and the semantic relationships (SR and MR) between them; the *Reasoning Rules* that can be used to support both delimitation of semantic block and the inference process in the reasoning phase, such as annotation suggestion rules, inconsistency detection rules, conflict identification rules and so on.

The Annotation Phase: during this phase, a number of semantic annotations are generated for supporting the reasoning phase.

(1) *Explicitation of Structure Semantics*, in which, the structure semantics of a TKR, namely the interrelations between the model elements, are made explicit. The customized E and MME are used to keep selected the elements in the model and in the meta-model ontology mmo_x respectively. The binary relations in MR are used to define the semantic relationships between elements in E and elements in MME.

(2) *Explicitation of Domain Semantics*, in which, the domain semantics of a TKR, namely the meaning of model contents in a domain of interest, are made explicit. The customized P is used to keep the selected ontology elements in the PLC-related ontologies PO. The binary relations in the SR are used to define the semantic relationships between the elements in the E and the elements in P.

Reasoning Phase: during this phase, the reasoning is performed based on the outputs of the above-mentioned two phases.

(1) *Configuration of Reasoning Parameters*, in which, based on the customization of semantic annotation schema and the practical situation for different TKRs, corresponding operations that support the configuration of reasoning parameters are performed by annotators or machines, such as the delimitation of semantic block for semantics substitution, the determination of the equivalents of two properties, the comparison of the semantic similarity between two domain semantics of common annotated objects, just to name three possible processes.

(2) *Reasoning on Semantic Annotations*, in which, the reasoning is performed based on the semantic annotations, the parameters and the reasoning rules to produce inference results. For example, the results of annotation suggestions,

annotation inconsistency detection, model conflict identification and so on.

This workflow describes the application of the semantic enrichment solution within one single TKR. In order to deal with the multiple TKRs in a PLM environment we propose a semantic annotation framework to address the issues of semantic interoperability.

3.3.2 The Framework Architecture

As shown in the Figure 3-5, on the left side, there is a series of processes to describe a product life cycle. They represent the *TKR Creation and Management* module. On the right side, there are four main modules of this framework: the *OKR Creation and Management* module, the *Knowledge Cloud* module the *Semantic Annotation and Processing Agent* (SAPA) module and the *Reasoning Engine* module.

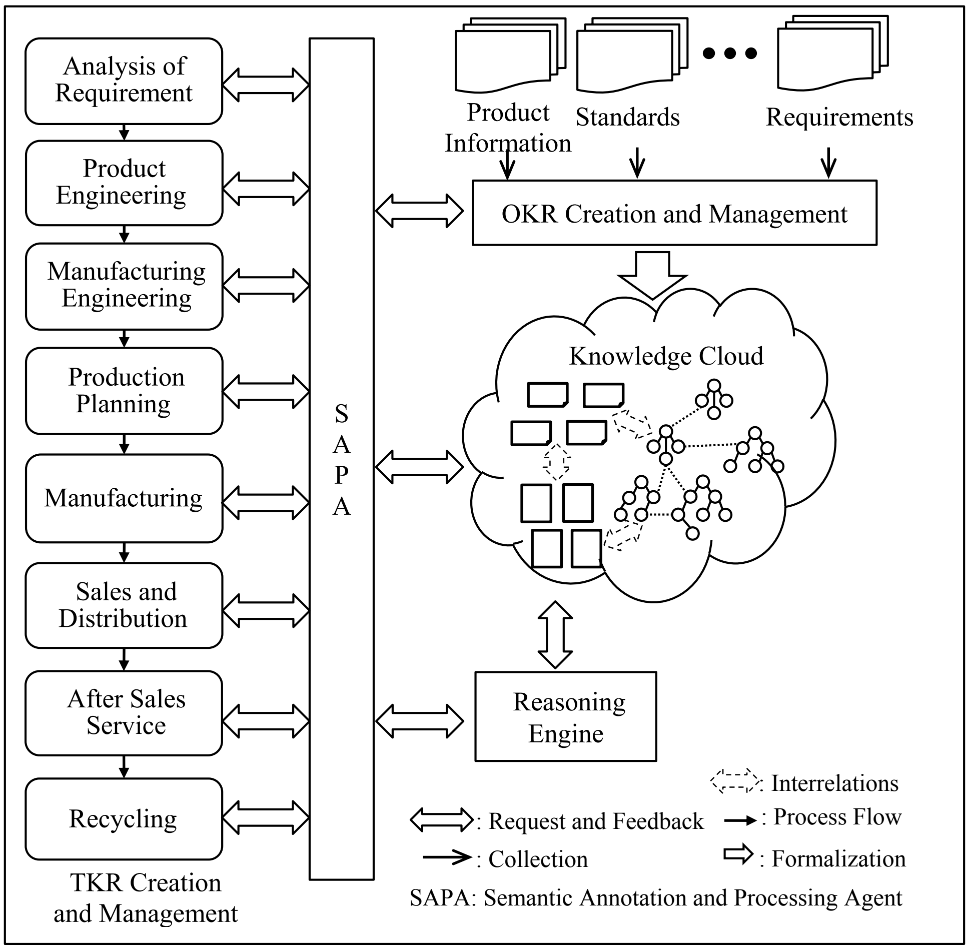


Figure 3-5 The Proposed Semantic Annotation Framework in a PLM Environment.

The *TKR Creation and Management* module is composed of a number of enterprise systems. Stakeholders in or across enterprises, during a product lifecycle use those systems to create TKRs following the modelling specifications and to manage those TKRs. Those systems need to provide sufficient APIs to enable the communications between themselves and the *Semantic Annotation and Processing Agent* module.

The *OKR Creation and Management* module is in charge of capturing, formalizing and managing PLC-related knowledge and model constructs knowledge into a knowledge base, namely, *Knowledge Cloud*. The OKRs are supposed to be in a platform-independent form, which ensures different kinds of ontologies that are collected from different sources, to be imported, mapped, merged and interrelated with each other.

The *Knowledge Cloud* module acts as a knowledge repository, which is in charge of storing different kinds of knowledge. As shown in the Figure 3-6, three main kinds of knowledge are stored in the knowledge cloud:

(1) All the OKRs produced by the *OKR Creation and Management* module. As shown in the Figure 3-6 (a), the OKRs are structured as the traditional three-levels structures.
   a) The *top level ontology*. It contains common terms and specifies the most common terminology that can be used in different domains.
   b) The *domain level ontology*. It is classified into two aspects:
      i. The *PLC-related ontologies* that represent the knowledge related to the product life cycle from different perspectives.
      ii. The *Meta-model ontologies* that represent the knowledge related to model constructs based on different specifications.
   c) The *application level ontology.* Corresponding to the two aspects of ontologies in the domain level, the related ontologies in this level are responsible for representing the specific terms that are defined and used in an enterprise and for representing the specific implementation of meta-model concepts in different modelling tools respectively.

(2) All the semantic annotations that are created by different stakeholders along the product lifecycle via the *Semantic Annotation and Processing Agent* module. As shown in the Figure 3-6 (b), these semantic annotations define the semantic relationships between TKRs and their corresponding OKRs. They also can be used as the bridges to make explicit the interrelations between the annotated elements in the disperse TKRs.

(3) All the reasoning rules, as shown in the Figure 3-6 (c), which are created based on the concepts and relationships in the OKRs and the customized semantic annotation schema. They are used for supporting the inference in the

Reasoning Engine module.

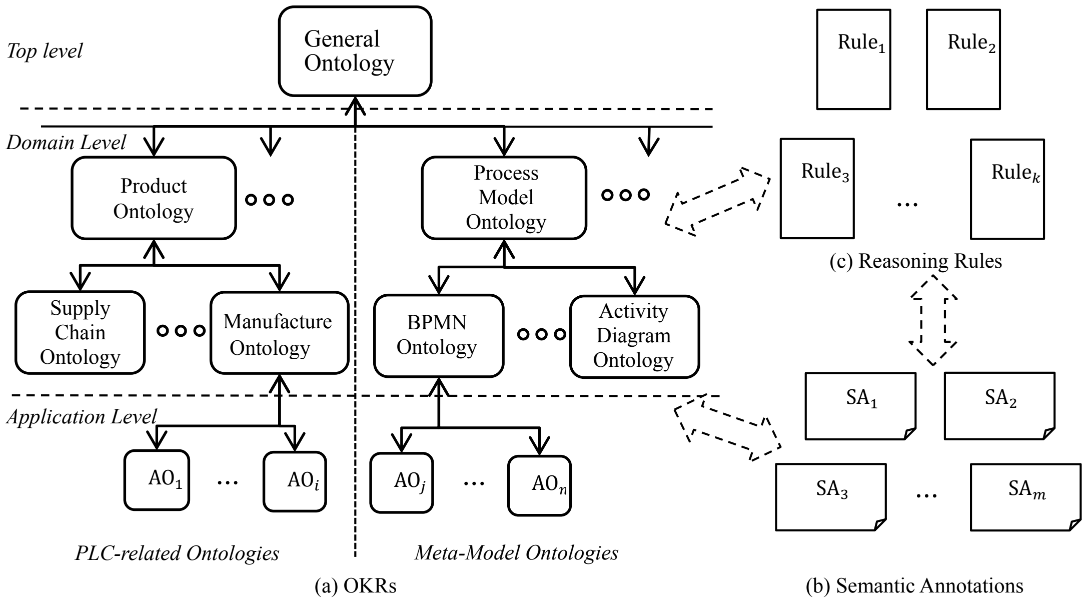


Figure 3-6 Three kinds of Knowledge in the Knowledge Cloud.

The *Reasoning Engine* module is an external call pattern-matching search engine, which uses different reasoning algorithms according to the stakeholders' requests. It performs the inferences on the knowledge that is stored in the semantic annotations, in the OKRs and in the reasoning rules.

The *Semantic Annotation and Processing Agent* (SAPA) model is mainly in charge of the semantic relationships definition process. It also acts as a mediator to support the communications (requests and feedbacks) between various kinds of modelling systems in different processes of the PLC (*TKR Creation and Management* module) and the three other modules (*Knowledge Cloud* module, *OKR Creation and Management* module and *Reasoning Engine* module) in the semantic annotation framework:

(1) Between the *Knowledge Cloud* module and the modelling systems: according to the particular semantic annotation requests from the stakeholders. The SAPA is in charge of querying (with the assistant of the *Reasoning Engine*) the *Knowledge Cloud* and provides appropriate OKRs as feedbacks. It also takes care of the management (such as creating, modifying, loading, deleting and so on) of existing semantic annotations and reasoning rules;

(2) Between the *OKR Creation and Management* module and the modelling system: based on the requests from stakeholders. SAPA is supposed to be able to communicate with OKR Creation and Management module for the manipulation of the OKRs;

(3) Between the *Reasoning Engine* and the modelling Systems: SAPA submits the inference requests from the stakeholders to the Reasoning Engine for performing the reasoning actions (such as the suggestion of semantic annotations, the detection of annotation inconsistencies, the identification of possible model conflicts and so on) and it sends back the corresponding results to the stakeholders.

This semantic annotation framework makes use of semantic annotations as a bridge between TKRs and their corresponding OKRs: 1) to make explicit the implicit semantics of TKRs; 2) to give a possibility to detect the inconsistencies between semantic annotations and identify the possible conflicts among the annotated elements in TKRs; 3) to make explicit the implicit relationships among all disperse TKRs.

3.4 Conclusion

In a nutshell, taking into account the five requirements that are identified at the end of Chapter 2, the corresponding contributions for each requirement are listed as follows:

(1) *It should provide a general semantic annotation structure model that is able be used to formalize semantic annotations for different kinds of models*;

Based on the major components of a semantic annotation (Section 3.1.1) and two semantic block delimitation methods (Section 3.1.2), we proposed a general semantic annotation structure model through eight formal definitions (Section 3.1.3). The structure semantics of an annotated object is represented as the annotation contents in its semantic annotation, but not a part of annotation schema. Therefore, this proposal can be apply in any models.

(2) *It should discover the possibility of using both structure and domain semantics together in the inference process*;

Section 3.2.1 presents a possible way to use these two aspects of semantics together for the suggestion of semantic annotations. The interrelations between annotated objects (structure semantics) are used as candidates for the property association. The meanings of an annotated object (domain semantics) are used as the basis and scope for the suggestion. Once the association between a relation in the structure semantics and a relation in domain semantics is created, a new semantic annotation can be suggested.

(3) *It should provide some mechanisms to assist the detection of the inconsistencies between semantic annotations and the identification of the conflicts between annotated objects*;

We presented two mechanisms to fulfil this requirement in Section 3.2.2. The annotation inconsistency detection mechanism takes the results of semantic similarity comparison between two domain semantics, which are used to annotate a common object, as input and produces possible annotation inconsistency detection results. The model conflict identification mechanism takes the results from the former mechanism as inputs and identifies the possible conflicts between annotated objects.

(4) *It should provide a way to guide annotators in how to apply the formal semantic annotations and how to benefit from those semantic annotations*;

In Section 3.3.1, we present a semantic annotation workflow, which is composed of three main phases: the preparation phase, the annotation phase and the reasoning phase. In the case study chapter, these phases are used as the section structure for explaining how we apply and use the semantic annotations in a particular application scenario.

(5) *It should provide a framework to support the semantic enrichment of models along the product life cycle.*

Based on the general semantic annotation structure model and the semantic annotation workflow, the overall architecture of the semantic annotation framework is proposed in Section 3.3.2. It shows the possibility of using formal semantic annotations to assist the modeller and the model user along the product lifecycle to “speak in a same language with the same semantics”.

Above all, according to the solution that we proposed, in the next chapter, we will present one of the possible ways to design and implement a prototype annotation tool.

Chapter 4 SAP-KM (Semantic Annotation Plug-in for Knowledge Management)

In order to apply the proposed solution into real-life applications, we designed and implemented a prototype annotation tool, named SAP-KM (Semantic Annotation Plug-in for Knowledge Management). It is used to support the semantic enrichment of models for making explicit their implicit semantics and to interface with a reasoning engine for performing reasoning on the semantic annotations.

In this chapter, Section 4.1 gives an overview of the SAP-KM through the presentation of requirement specifications and the prototype development environment. Section 4.2 first presents the data structure for the creation of a semantic annotation schema (Section 4.2.1). Based on this schema, we then approach also the design of procedures for making explicit the domain and structure semantics (Section 4.2.2) and the design of procedures for the preparation and the execution of reasoning (Section 4.2.3). In Section 4.3, the implementation of the SAP-KM is presented through the presentation of its seven main graphical user interfaces. Section 4.3.1 illustrates the functions for the explicitation of structure and domain semantics. Section 4.3.2 presents the functions for the annotation suggestions, the semantic similarity comparison and the inference. Two extended functions, the elements matching and the data query are introduced in Section 4.3.3. Finally, Section 4.4 concludes this chapter and discusses the possibility of applying the proposed solution into other modelling systems. In order to ease the understanding of the chapter's contents, a logical structure of this chapter is presented in the Figure 4-1, which is illustrated according to the main functions of the SAP-KM. This figure describes the responsible sections to different functions from the design and implementation point of view. The main dependent relations between these sections (sections from both Chapter 3 and Chapter 4) are also summarized in this figure. These dependencies describe the main connections between the proposed solution and its corresponding realisation.

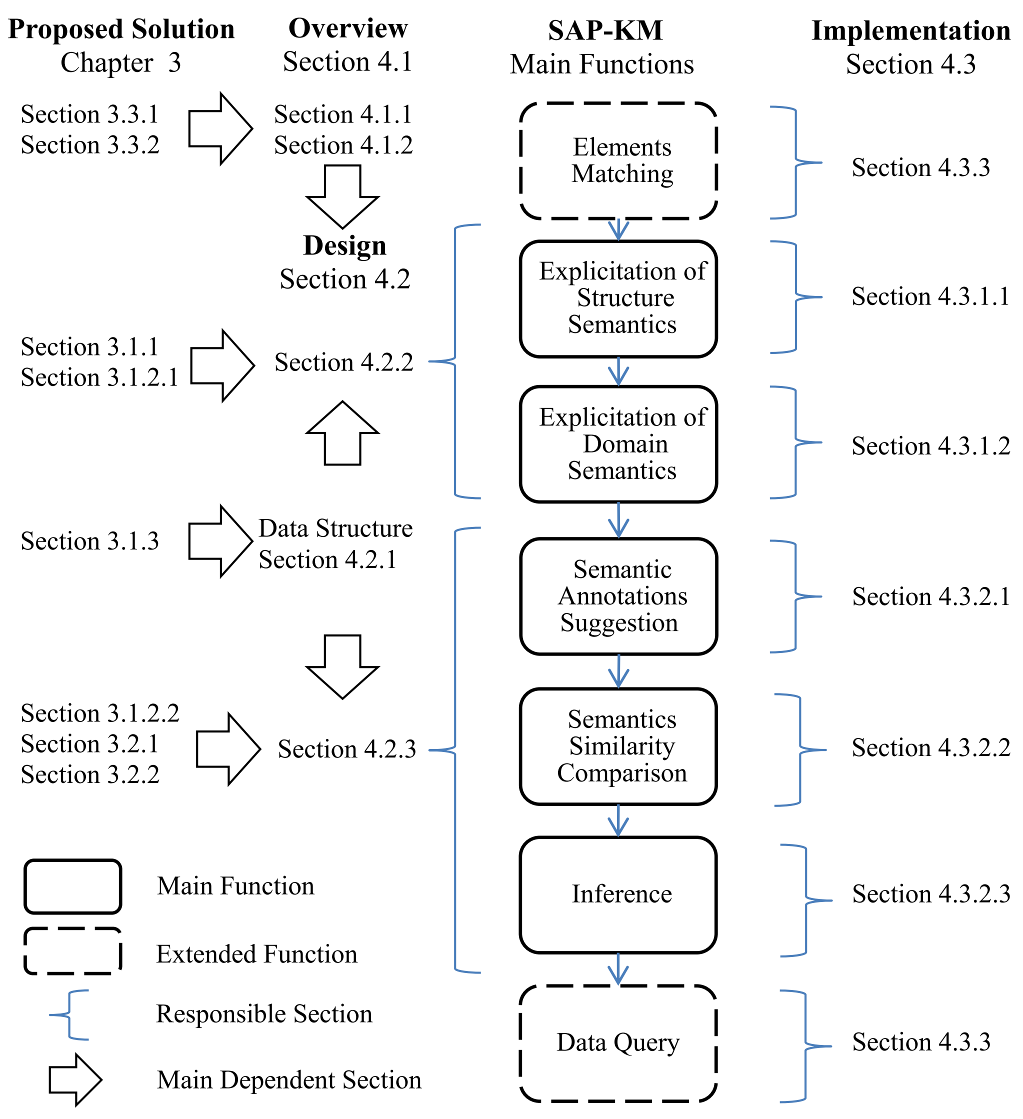


Figure 4-1 The Logical Structure of the Chapter

4.1 The Overview of the SAP-KM

In this work, the semantic annotation uses the concepts and relationships in ontologies to make explicit the domain and structure semantics of models. The explicit semantics is used to deal with the semantic interoperability issue in a PLM environment thanks to the provision of a common terminology between different stakeholders. The purpose of developing the SAP-KM is to cope with this objective and to demonstrate how the proposed semantic annotation formalization can be applied into real-life applications. The SAP-KM is designed as a plug-in of a general modelling platform, which gives a possibility to add semantic annotations to various kinds of models in a PLM environment. The model that we selected and used as an example is a process model.

Although the main direction is clearly defined at the beginning of this manuscript, the detailed requirements for this prototype need to be made clearer during the progress of the research. Therefore, the development of the SAP-KM follows an iterative development process [121], which lets engineers use the knowledge that they have learned from the previous cycles and apply it to update the system incrementally. As illustrated in the Figure 4-2, the initial requirements are the inputs of the iterative development cycle. The development process includes five main stages: (*i*) requirement analysis, (*ii*) design, (*iii*) implementation, (*iv*) testing and evaluation, and (*v*) generation of new requirements. The iteration will continue until all requirements have been achieved.

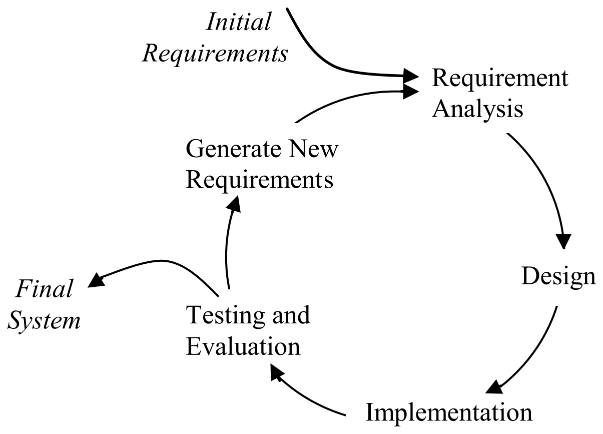


Figure 4-2 The Iterative Development Cycle [121]

To illustrate the prototype SAP-KM unambiguously, we summarize the contents inside each iterative development stage and present them as a whole from the requirements to the prototype validation. The main requirements of this prototype are listed in Section 4.1.1 and the development environment is presented in Section 4.1.2 to guide the design and implementation in the remaining sections of this chapter.

4.1.1 The Requirement Specifications

The requirement analysis is mainly considered as the task that determines the needs to achieve for a software system. They can also be extended to complete the evaluation factors for the final testing. Besides the initial general requirements, during the iterative development process, some new requirements may be generated. The requirements for the SAP-KM come from the components of the semantic annotation framework that we proposed in Section 3.3.2. The following modules are specifically concerned:

- The *TKR Creation and Management* module represents the knowledge in a domain of interest from different perspectives. The semantic annotations should be added directly to the model elements in the visual model diagram in a user-friendly way. After the annotation process, semantic annotations are stored without changing the annotated models. It should always be possible to load and to modify semantic annotations. The semantic conflicts between the annotated model elements should be identified in the inference results and they should be shown or informed to the annotator for assisting the model creation and to guarantee its correctness.
- The *OKR Creation and Management* module captures, formalizes and manages both PLC-related knowledge and model constructs knowledge. The ontologies that are produced by this module should be coded in a platform–independent way and should be accessed (loaded) by the prototype.
- The *Knowledge Cloud* module is a knowledge repository that stores all the ontologies, semantic annotations and reasoning rules. On the one hand, the prototype should be able to browse and select all knowledge (meta-model or PLC-related ontologies, existing semantic annotations and existing reasoning rules) in the knowledge cloud. On the other hand, the prototype should be able to store the created new semantic annotations and new reasoning rules into the knowledge cloud.
- The *Reasoning Engine* module is in charge of answering different kinds of query requests and performing certain inferences based on reasoning rules. It requires the prototype to provide sufficient inputs as the basis for reasoning. For example, the semantic annotations should be saved in a specific format, the rules should be written in a machine understandable syntax, etc. It also needs that the prototype has the ability to access and handle the feedbacks it produces.
- The *Semantic Annotation and Processing Agent* module is in charge of manipulating (for example, the creation, modification, deletion, etc.) the semantic annotations and communicating with other modules. This is the main module of the prototype. According to its responsibilities, the requirements are listed as follows:
  1. The semantic annotation schema should be implemented based on the formalization of the semantic annotations (Section 3.1.3).
  2. The elements in a TKR, the elements in OKRs and their semantic

relationships should be stored in the semantic annotation schema.

3. The inferred semantic annotations should be suggested according to the existing initial semantic annotations and the annotation suggestion mechanism (Section 3.2.1).
4. The inconsistencies between two semantic annotations of a common annotated object should be detected according to the results of the similarity comparison among their domain semantics and according to the inconsistency detection mechanism (Section 3.2.2).
5. The possible semantic mistakes in a model should be identified according to the inconsistency detection results and the mistake identification mechanism (Section 3.2.2).
6. The possibility of applying this prototype on other kinds of model should be proved.

Because of the existence and the maturity of tools for supporting models creation, ontology creation and ontology reasoning, we will not develop the corresponding software related to the previous four modules, we will just employ existing ones as part of the semantic annotation framework.

The Semantic Annotation and Processing Agent are considered as the central module of this framework that acts as an interface to support the communications among all the modules. In the next section we will introduce the employed tools and the technologies that are used to develop the prototype.

4.1.2 The Prototype Development Environment

In general, the SAP-KM is designed as a plug-in for an existing modelling system. It should support the semantic annotation processes and the reasoning processes. The semantic annotation processes take ontologies as inputs and generate semantic annotations as outputs. The reasoning process transfers the semantic annotations and the reasoning rules as inputs of an existing reasoning engine and retrieves its outputs as the reasoning results. During all these processes, the knowledge cloud is used as a repository of ontologies, annotations and rules. Therefore, for demonstrating the usability of the SAP-KM according to the semantic annotation framework, at least three kinds of existing tools need to be employed: a modelling system, an ontology editor and a reasoning engine.

As a modelling system, we chose the MEGA modelling environment[10] to implement the *TKR Creation and Management* module. It is used as the modelling environment for supporting the creation of enterprise models and providing visual model diagrams to add semantic annotations. A number of modelling tools and supports have been integrated into the MEGA, which fulfils different kinds of modelling requirements. For example, among many others, (*i*) the "MEGA process BPMN" (as we had introduced in Section 2.1.2) supports the modelling of processes through BPMN notation, (*ii*) the "MEGA IT Governance and Specifications" supports the modelling of IT architecture and (*iii*) the "MEGA Data Designer" supports the modelling of data and databases. The models that are created by MEGA can be exported in the form of XML that gives possibility to be easily exchanged. One of the most important features of MEGA, which is also one of the main reasons why we chose it, is that it provides APIs to support the development of plug-ins.

As an ontology editor, we chose the popular Protégé-OWL editor[11] to implement the *OKR Creation and Management* module. It responds for handling the PLC-related ontologies and meta-model ontologies in the knowledge cloud. Protégé supports the creation, visualization and manipulation of ontologies in various representation formats and it is also able to assist the operation of the ontology mapping, merging and versioning, which fulfils the needs of the knowledge cloud management.

The *Knowledge Cloud* module is supposed to be implemented as a server that allows all the annotation plug-ins of different modelling systems along the product lifecycle to load, modify and save ontologies, semantic annotations and rules. However, due to the limited resources, we use the local Microsoft windows folder system to manage all the knowledge. The folder "Knowledge Cloud" is created to act as the knowledge repository, which contains three sub folders named "OKRs", "Semantic Annotations" and "Reasoning Rules". The folder "OKRs" has two sub folders: the "Meta-model Ontologies" folder and the "PLC-related Ontology" folder. The ontologies, the semantic annotations and the reasoning rules are classified and stored in the corresponding folders.

For the *Reasoning Engine* module, in this work, we chose the built-in Jena[12] Reasoner for supporting the ontology query and inference processes. The Jena API gives the SAP-KM the possibility to load one or more ontologies into its embedded

[10] MEGA: http://www.mega.com/ (we use the version 2009 SP5)

[11] Protégé -OWL editor: http://protege.stanford.edu/

[12] Jena http://jena.apache.org/

ontology model. In the following thesis, we will use the expression "ontology model" to represent this embedded ontology model in Jena. SAP-KM can employ an instance of Jena reasoners to manipulate all these loaded ontologies. There are four available default reasoners [120] that are provided by the Jena API: the transitive reasoner, the RDFS rule reasoner, the OWL reasoner and the generic rule reasoner. We mainly use the default OWL reasoner to support the retrieval of ontologies and the generic rule reasoner to execute the reasoning rules.

Finally, for the *Semantic Annotation and Processing Agent* module, during the development of the SAP-KM, we mainly use the NetBeans[13] programming platform, based on the Java[14] programming language.

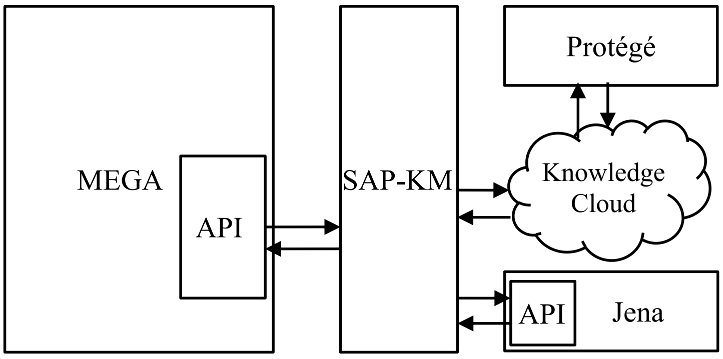


Figure 4-3 The Collaboration between the SAP-KM and the other four Modules

Figure 4-3 illustrates the collaboration between the SAP-KM and the four modules. Through the MEGA APIs, the model elements in models can be retrieved. The PLC-related ontologies and meta-model ontologies can be edited and transformed into the appropriate format by the Protégé OWL Editor. The ontologies, the semantic annotations and the reasoning rules are stored in the Knowledge Cloud. Through the APIs that are provided by Jena, the SAP-KM is able to: (1) parse ontologies that are created and managed by the Protégé-OWL editor; (2) support the function of loading ontology concepts and relationships into a ontology browser; (3) execute the reasoning rules (such as the rules for the semantic block delimitation, annotation inconsistency detection and model conflict identification); (4) generate all the semantic annotation results into the designed semantic annotation schema. We will present the details of its design and implementation in the following two sections.

[13] Netbeans: https://netbeans.org/

[14] Java: http://www.java.com/en/

4.2 The Design of the SAP-KM

The SAP-KM has two main tasks: (1) Define the semantic relationships between a selected model element and its structure and domain semantics; (2) Perform the reasoning on the existing semantic annotations according to the reasoning rules for obtaining certain inference results. In Section 4.2.1, we will show one of the possible designs of the data structure that follows the formalization of semantic annotations. Taking advantages from this data structure, Section 4.2.2 illustrates the design of the procedure of how the SAP-KM assists an annotator in using the meta-model ontology and the PLC-related ontologies for making explicit the structure semantics and the domain semantics of the elements in a model. Section 4.2.3 presents the design of the procedure of how the SAP-KM supports an annotator to perform the reasoning on the existing semantic annotations based on the corresponding reasoning rules.

4.2.1 The Design of the Data Structure

Based on the formal definitions of semantic annotations presented in Section 3.1.3, a semantic annotation schema is designed to store the annotation results. In order to use the existing reasoning engines to assist the annotation and reasoning processes, this schema is structured as an ontology, named Semantic Annotation Schema. It uses appropriate Classes, Properties and Individuals to represent the five main elements of the Semantic Annotation SA[15](Section 4.2.1.1 and Section 4.2.1.2) and some additional Properties to assist the creation of reasoning rules (Section 4.2.1.3).

4.2.1.1 The Data Structure for E, P and MME

E, P and MME, as shown in the Table 4-1, are represented as three disjoint Classes, named “E”, “P” and “MME” respectively, in the Semantic Annotation Schema.

Table 4-1 The E, P and MME in the Semantic Annotation Schema

Definitions		Descriptions
E	e_i	e_i is represented as an Individual
P	PO	PO is a number of PLC-related ontologies
	p_j	p_j is supposed to be represented by a sub ontology (ideally). In reality, it is represented as an Individual, together with three Object Properties “*hasMainConcept*”, “*hasSBEntity*” and “*hasSBRelation*”,

[15] According to definition 8 in Section 3.1.3, the semantic annotation SA is defined by 5-tuples: $SA \coloneqq (E, P, SR, MME, MR)$, where E is a set of elements from a Target Knowledge Representation; P is a set of selected ontology element sets from a set of PLC-related ontologies PO (definition 3); MME is a set of ontology elements from a meta-model ontology mmo_x (definition 4); SR is a set of binary relations that describes the semantic relationships between E and P (definition 6); MR is a set of binary relations that describe the semantic relationships between E and MME (definition 7).

		one Datatype Property "*hasLongNS*", and two Classes "*SBRelations*" and "*NSstore*".
MME	mmo_x	mmo_x is a meta-model ontology
	mme_l	mme_l is a Class in the mmo_x

More specifically, the data structures for each element in the Table 4-1 are defined as follows:

e_i is represented as an Individual of the Class "E". The local name (the name of an ontology element without its namespace) of each e_i is constructed according to the name syntax in the Figure 4-4. An example of Class E and its Individuals are shown in the Figure 4-5 (a).

PO is a number of PLC-related ontologies from the knowledge cloud, which are imported in the Semantic Annotation Schema. p_j is supposed to be represented as a subset of one or more ontologies, which contains a number of selected ontology elements (a semantic block for semantics description). However, due to the fact that the OWL DL specification does not support the expression of p_j, we need to define an appropriate way to represent it:

- p_j is represented as an Individual of the class P. The local name of each p_j is constructed based on the name syntax in the Figure 4-4.
- The Object Property "$hasMainConcept$" defines the relationship between a p_j and a selected Class or Individual in PO.
- The Object Property "$hasSBEntity$" defines the relationship between a p_j and a selected Class or Individual in POthrough the semantic block delimitation.
- The Class "SBRelations" is used to store all the semantic block relations that describe the relationship between two selected concepts in a semantic block. Each semantic block relation, named SBRelaion, represents an Individual of this Class. The local names of these Individuals are constructed according to the name syntax in the Figure 4-4.
- The Object Property "$hasSBRelation$" defines the relationship between a p_j and an Individual of the Class "SBRelations".
- The Class "NSstore" is used to store all the namespace abbreviations that are used for supporting the local name construction of SBRelations. Each namespace abbreviation is represented as an Individual of this Class.
- The Datatype Property "$hasLongNS$" is used to describe the relationship between a namespace abbreviation and its full ontology namespace. Each real ontology namespace is stored as a data with the datatype "xsd:string" and

connected to the corresponding individual in the class "*NSstore*" through this property.

Additionally, because the data values contained in every "*SBEntity*" are difficult to be represented, this data structure does not directly represent them in p_j. However, during the utilization of p_j (such as annotating, comparing, reasoning and so on), each *SBEntity*'s data values is queried from the ontologies where they originally belong to and used as a part of p_j. Figure 4-5 (b) shows an example of the class P and its individuals.

mmo_x is a meta-model ontology from the knowledge cloud, which is imported in the Semantic Annotation Schema. The mme_l is a class in the mmo_x. Each mme_l has its model instances, which are the individuals in the Class "*E*". Once the relationship between an e_i and a mme_l is defined, the mme_l will be set as a sub-class of the class "*MME*". An example of the class "*MME*" and its sub-classes are shown in the Figure 4-5 (c).

$\boldsymbol{e_i}$	::= terms1 *node1* term2	
	//e.g. *Manufacture_Prod3::P0110-31FB3A3052052113*	
$\boldsymbol{p_j}$	::= term3 term4 *node1* term5 *node1* e_i	
	//e.g. *P2-Of-Manufacture_Prod3::P0110-31FB3A3052052113*	
SBRelation	::= term6 *node1* term10 *node2* term11 *node2* term10	
	//e.g. *309824-xzpfva-P0110_____ xzpfva-hasShape_____rzesed-Cylinder*	
	or term6 *node1* term10 *node2* term11 *node2* term15	
	//e.g. *341211-rzesed-Turing_____rzesed-isPerformedOn_____allValuesFrom _____ rzesed-Lathe*	
term1	::= the name of a model element	//e.g. *Manufacture_Prod3::Bases_Turning*
term2	::= the unique identification of a model element	//e.g. *31FB3A3052052113*
term3	::= the string “P”	
term4	::= a number	// e.g. 2, 17
term5	::= the string “Of”	
term6	::= the unique identification of a SBRelation	// e.g. *309824*
term7	::= the abbreviation of the namespace of an ontology	// e.g. *xzpfva, rzesed*
term8	::= the local name of an ontology concept	// e.g. *P0110, Cylinder*
term9	::= the local name of an ontology relationship	// e.g. *hasShape*
term10	::= term7 *node1* term8	// e.g. *xzpfva-P0110*
term11	::= term7 *node1* term9	// e.g. *xzpfva-hasShape*
term12	::= the string “allValuesFrom” or “someValueFrom”	
term13	::= the string “maxCardinality” or “minCardinality” or “Cardinality”	
term14	::= the string “intersectionOf” or “unionOf” or “complementOf”	
term15	::= term12 *node2* term10	
	or term13 term4 *node3* term10	
	or term12 *node2* term14 *node3* term10 … term10 *node4*	
	or term13 term4 *node2* term14 *node3* term10 … term10 *node4*	
node1	::= the symbol “-”	
node2	::= the symbol “_____”	
node3	::= the symbol “(”	

Figure 4-4 The Name Syntax of the e_i , p_j and SBRelation

4.2.1.2 The Data Structure for SR and MR

SR and MR, as can be seen in the Table 4-2, are represented as two Object Properties, named “SR” and “MR” respectively, in the Semantic Annotation Schema.

Table 4-2 SR and MR in the Semantic Annotation Schema

Definitions		Property Names	Descriptions
SR	$sr_{\sim}$	*SR_isEquivalentTo*	Sub property of: *SR* Domain: *E* Range: *P*
	$sr_{\supset}$	*SR_subsumes*	Sub property of: *SR* Domain: *E* Range: *P*
	$sr_{\subset}$	*SR_isSubsumedBy*	Sub property of: *SR* Domain: *E* Range: *P*
	$sr_{\cap}$	*SR_intersects*	Sub property of: *SR* Domain: *E* Range: *P*
	$sr_{\perp}$	*SR_isDisjointWith*	Sub property of: *SR* Domain: *E* Range: *P*
MR	mr_{io}	*rdf:type*	*rdf:type* is used to state that a resource is an instance of a class

To be more specific, the data structures for each element in the Table 4-2 are shown as follows:

The class "*E*" and the class "*P*" are defined as the domain and the range of the Object Property "SR". The Object Properties "*SR_isEquivalentTo*", "*SR_subsumes*", "*SR_isSubsumedBy*", "*SR_intersects*" and "*SR_isDisjointWith*" are used to represent $sr_{\sim}$, $sr_{\supset}$, $sr_{\subset}$, $sr_{\cap}$ and $sr_{\perp}$ respectively. They are defined as the sub properties of Object Property "*SR*" and inherit their super property's domain and range. Because the domain and the range of these properties are disjointed from each other, these sub properties have not special property characteristics. One example of $sr_{\subset}$ is shown in the Figure 4-5 (d).

mr_{io} is supposed to be represented as the sub property of the Object Property "MR" in the Semantic Annotation Schema. However, during the implementation, for efficiency reason, we employed the "rdf:type" to represent mr_{io}. It is used to describe the semantic relationship between an individual of the Class "*E*" and a sub class of the Class "*MME*". One example of mr_{io} is shown in the Figure 4-5 (e).

Figure 4-5 shows an example of a semantic annotation that is represented by the Semantic Annotation Schema. The Class "*E*" (the (a) in the figure) contains two individuals. The individual "*Manufacture_Prod3::P0110-31FB3A3052052113*" is annotated by an individual of the Class "*P*" (the (b) in the figure) and a subclass of the Class "*MME*" (the (c) in the figure). The Object Property "*SR_isSubsumedBy*" (the (d) in the figure) denotes that the domain semantics of the individual "*Manufacture_Prod3::P0110-31FB3A3052052113*" is less general than the domain semantics of the individual "*P1-Of-Manufacture_Prod3::P0110-31FB3A3052052113*". The employed property "*rdf:type*" (the (e) in the figure) denotes that the individual "*Manufacture_Prod3::P0110-31FB3A3052052113*" is an instance of the Class "DataObject". The individual "*P1-Of-Manufacture_Prod3::P0110-31FB3A3052052113*" represents a semantic block. It uses the Object Property "*hasMainConcept*" to define its main concept (the individual "*P0110*"). It uses the Object Property "*hasSBEntity*" to define the other concepts (the individual "*Cylinder*") in the semantic block. It uses the Object Property "*hasSBRelation*" to define its semantic block relations. The Class "SBRelations" has an individual "*309824-xzpfva-P0110_____xzpfva-hasShape_____rzesed-Cylinder*", which represents the relationship (the Object Property "*hasShape*") between two selected concepts (the individual "*P0110*" and the individual "*Cylinder*"). The Class "NSstore" has two individuals (the individual "*xzpfva*" and the individual "*rzesed*"), which keep the full namespaces through the Datatype Property "*hasLongNS*".

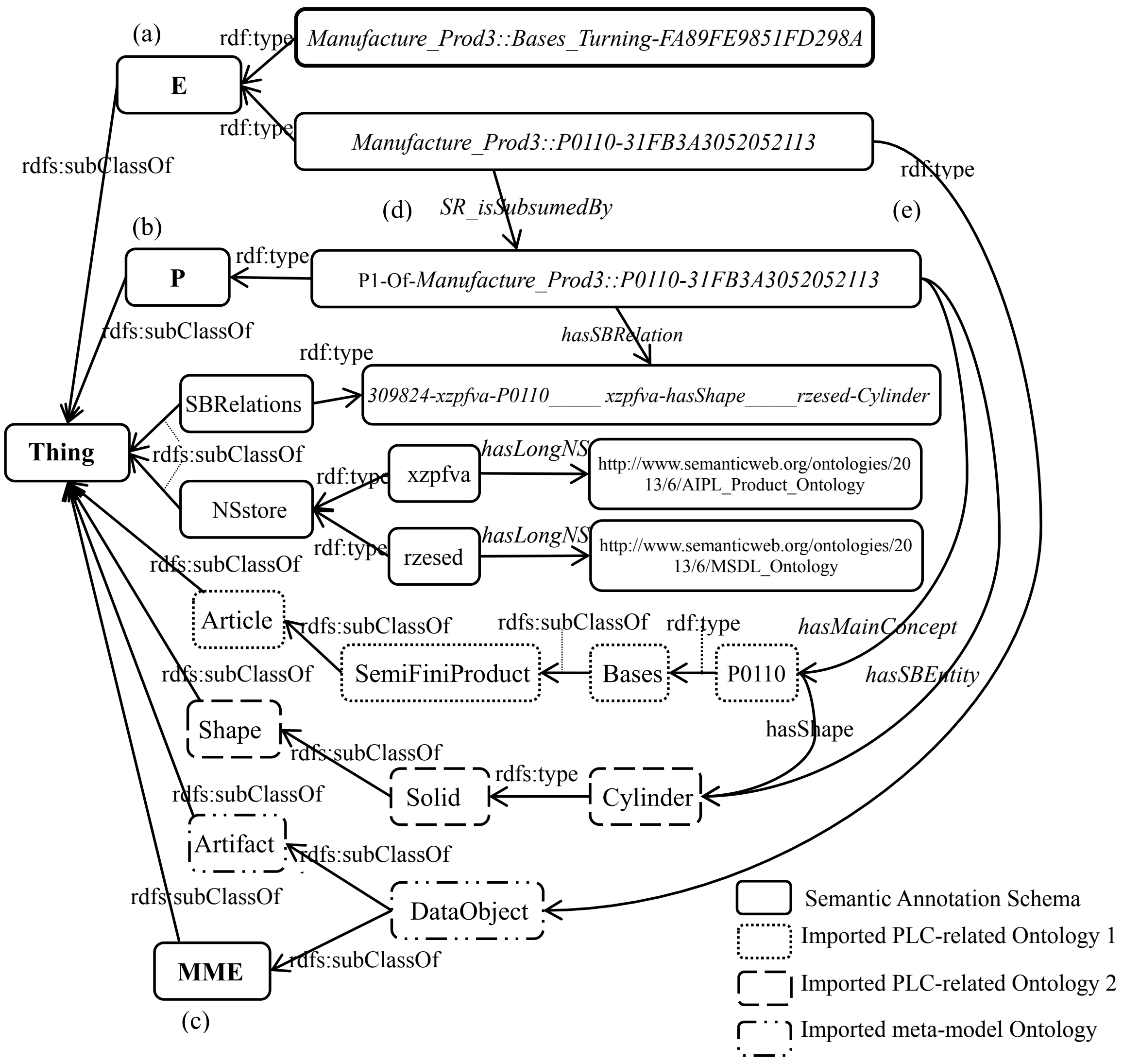


Figure 4-5 An Example of Instantiation of E (a), P (b), MME (c), SR (d) and MR (e) within

the Semantic Annotation Schema

4.2.1.3 The Data Structure for Several Additional Properties

Furthermore, besides the above-mentioned designs in order to assist the reasoning process and simplify the expression of reasoning rules, several additional properties are added into the Semantic Annotation Schema.

The semantic relationship PR[16] is used to represent the semantic similarity comparison results between two domain semantics of a common annotated object. As pictured in the Table 4-3, it is represented as an Object Property, named "*PR*", in the Semantic Annotation Schema. The class "*P*" is defined as the domain and as well as the range of the Object Property "*PR*". The Object Properties "*PR_isEquivalentTo*", "*PR_subsumes*", "*PR_isSubsumedBy*", "*PR_intersects*" and "*PR_isDisjointWith*" are used to represent $pr_{\sim}$, $pr_{\sqsupset}$, $pr_{\sqsubset}$, $pr_{\sqcap}$ and $pr_{\perp}$ respectively, which are defined as the sub properties of the Object Property "*PR*" and inherit their super property's domain and range. In this table, the corresponding property characteristics for each sub property are defined.

Table 4-3 PR in the Semantic Annotation Schema

Definitions		Property Names	Descriptions	
	$pr_{\sim}$	*PR_isEquivalentTo*	Sub property of: *PR* Characteristic: Transitive, Symmetric	Domain: *P* Range: *P*
	$pr_{\sqsupset}$	*PR_subsumes*	Sub property of: *PR* Characteristic: Transitive	Domain: *P* Range: *P*
PR	$pr_{\sqsubset}$	*PR_isSubsumedBy*	Sub property of: *PR* Characteristic: Transitive	Domain: *P* Range: *P*
	$pr_{\sqcap}$	*PR_intersects*	Sub property of: *PR* Characteristic: Symmetric	Domain: *P* Range: *P*
	$pr_{\perp}$	*PR_isDisjointWith*	Sub property of: *PR* Characteristic: Symmetric	Domain: *P* Range: *P*

As shown in the first three rows of Table 4-4, the three types of results about the detection of inconsistencies between semantic annotations are represented as three Object Properties:

- The Object Property "*isConsistentWith*" denotes that the domain semantics that is expressed by an individual in the domain and domain semantics that is expressed by an individual in the range are consistent[17] with each other.
- The Object Property "*isPosConsistentWith*" denotes that the domain semantics that is expressed by an individual in the domain and an individual in the range

[16] The definition 9 in Section 3.2.2

[17] Consistency is defined as the two domain semantics do not contain any logical contradictions.

are possibly consistent with each other.

- The Object Property "*isNotConsistentWith*" denotes that the domain semantics that is expressed by an individual in the domain and an individual in the range is not consistent with each other.

The Class "*P*" is defined as the domain and as well as the range of these three properties.

As shown in the fourth row of Table 4-4, the result about the identification of possible conflicts in a model is represented as an Object Property:

- The Object Property "isConflictWith" denotes that an individual in the domain and an individual in the range are conflicting[18].

The Class "*E*" is defined as the domain and as well as the range of this property.

As shown in the last two rows of the Table 4-4, two Object Properties are added into the Semantic Annotation Schema for simplifying the expression of reasoning rules:

- The Object Property "*isAnnotatedBy*" denotes that an individual in its domain is annotated by an individual in its range. The Class "*E*" is defined as its domain and the Class "*P*" is defined as its range.
- The Object Property "*isInferredFrom*" denotes whether an individual in its domain is an initial semantic annotation or an inferred semantic annotation. The Class "*E*" is defined as its domain and as well as its range. For example, if p_j is inferred from the semantic annotation of e_i, this Object Property will be added from p_j to e_i. Conversely, if p_j is initially added by an annotator, this object property does not appear.

Table 4-4 The Additional Properties in the Semantic Annotation Schema

Property Names	Descriptions	
isConsistentWith	Sub property of: *topObjectProperty* Characteristic: *Symmetric*	Domain: *P* Range: *P*
isPosNotConsistentWith	Sub property of: *topObjectProperty* Characteristic: *Symmetric*	Domain: *P* Range: *P*
isNotConsistentWith	Sub property of: *topObjectProperty* Characteristic: *Symmetric*	Domain: *P* Range: *P*
isConflictWith	Sub property of: *topObjectProperty* Characteristic: *Symmetric*	Domain: *E* Range: *E*
isAnnotatedBy	Sub property of: *topObjectProperty*	Domain: *E* Range: *P*
isInferredFrom	Sub property of: *topObjectProperty*	Domain: *P* Range: *E*

[18] Conflict is defined as the two annotated objects are incompatible or at variance.

A complete and empty Semantic Annotation Schema is shown in the Appendix II. Of course, not limited to this designed data structure, based on the formal definitions of semantic annotations, the Semantic Annotation Schema can be designed differently for adopting different kinds of requirements.

4.2.2 The Design of the Annotation Phase

The flowchart in the Figure 4-6 illustrates how the SAP-KM assists an annotator in using meta-model ontologies and PLC-related ontologies to make explicit the structure semantics and the domain semantics of the elements in a model. This procedure can be considered as the manipulation (such as adding, modifying, removing, etc.) of the classes, properties and individuals in the Semantic Annotation Schema. In the flowchart, the processes and judgements drawn with the thick line mean that the SAP-KM needs an annotator's participation, as well as the processes and judgements drawn with the thin line means that it is automatically performed by the SAP-KM. To be more specific, based on the different kinds of semantics that the SAP-KM makes explicit, this procedure is divided into two stages: (a) the explicitation of the structure semantics and (b) the explicitation of the domain semantics.

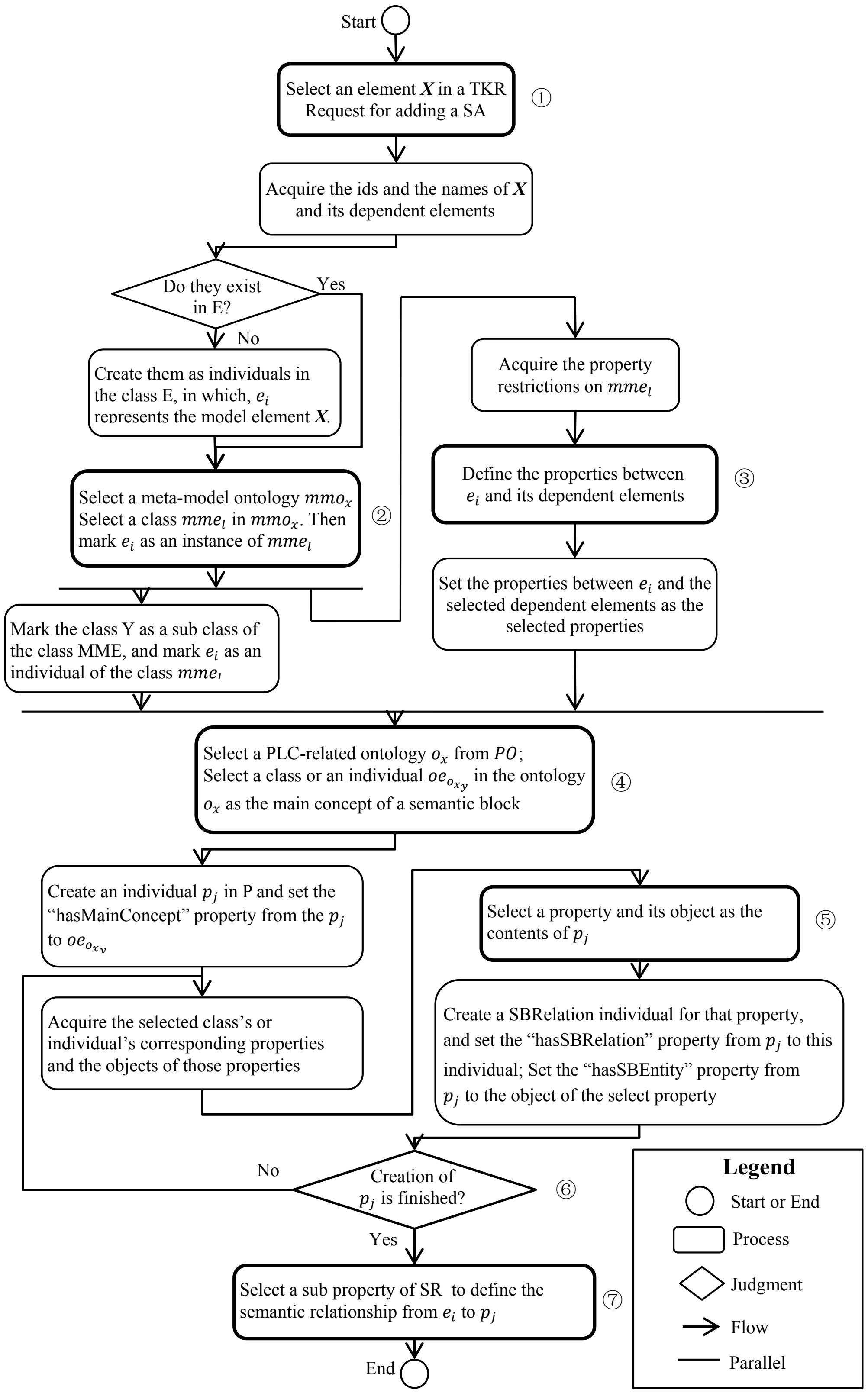

Start
Select an element X in a TKR Request for adding a SA
①
Acquire the ids and the names of X and its dependent elements
Do they exist in E?
Yes
No
Create them as individuals in the class E, in which, e_i represents the model element X.
Select a meta-model ontology mmo_x Select a class mme_l in mmo_x. Then mark e_i as an instance of mme_l
②
Mark the class Y as a sub class of the class MME, and mark e_i as an individual of the class mme_l
Acquire the property restrictions on mme_l
Define the properties between e_i and its dependent elements
③
Set the properties between e_i and the selected dependent elements as the selected properties
Select a PLC-related ontology o_x from PO; Select a class or an individual $oe_{o_{x_y}}$ in the ontology o_x as the main concept of a semantic block
④
Create an individual p_j in P and set the "hasMainConcept" property from the p_j to $oe_{o_{x_y}}$
Acquire the selected class's or individual's corresponding properties and the objects of those properties
Select a property and its object as the contents of p_j
⑤
Create a SBRelation individual for that property, and set the "hasSBRelation" property from p_j to this individual; Set the "hasSBEntity" property from p_j to the object of the select property
No
Creation of p_j is finished?
⑥
Yes
Select a sub property of SR to define the semantic relationship from e_i to p_j
⑦
End
Legend
Start or End
Process
Judgment
Flow
Parallel

Figure 4-6 The Procedure to make explicit the Domain Semantics and Structure Semantics

The path of a semantic annotation starts when an annotator selects an element X in a model and decides to annotate it. Based on the data structure that we designed in

Section 4.2.1, each annotated element $\boldsymbol{X}$ is represented as an individual e_i of the Class "*E*" in the Semantic Annotation Schema. Once the annotation request is made, the information of $\boldsymbol{X}$ and its dependent elements (those elements that are connected to $\boldsymbol{X}$ directly, for example, if $\boldsymbol{X}$ is an operation, the sequence flow that is connected to it will be its dependent element.) are acquired and transferred to the SAP-KM. The verification is made by the SAP-KM to check whether their corresponding individuals already existed in the class "*E*" or not. For the model elements that will get the "not" answer in the verification step, the SAP-KM will create their corresponding individuals based on the name syntax in the Figure 4-4. After this preparation, the explicitation of the structure semantics of $\boldsymbol{X}$ can be performed.

As shown in the Figure 4-6, process ②, a class mme_l in a meta-model ontology mmo_x is selected. According to the design of the data structure, a class mme_l is marked as a sub-class of the class "*MME*". The e_i is marked as the individual of the selected class mme_l. Meanwhile, the property constraints on the class mme_l are acquired and listed as candidate properties for the process ③.

In the process ③, the relationships from the individual e_i to its dependent elements are defined according to the listed candidate properties. These annotation results are saved into the Semantic Annotation Schema.

The explicitation of the domain semantics of $\boldsymbol{X}$ starts from the process ④ in the Figure 4-6. In this process, a class or an individual $oe_{o_{xy}}$ that is in a PLC-related ontology o_x is selected as the main concept of a semantic block (the semantic blocks for semantics description in Section 3.1.2). According to the data structure, this semantic block is represented as an individual p_j with its three kinds of object properties. The property "*hasMainConcept*" is set as the relationship from the p_j to the $oe_{o_{xy}}$. A query process, started by the selection action, takes this selected class or individual as its input and produces a list of corresponding properties (that are the class's property constraints or the individual's properties) together with all the objects that are related to these properties.

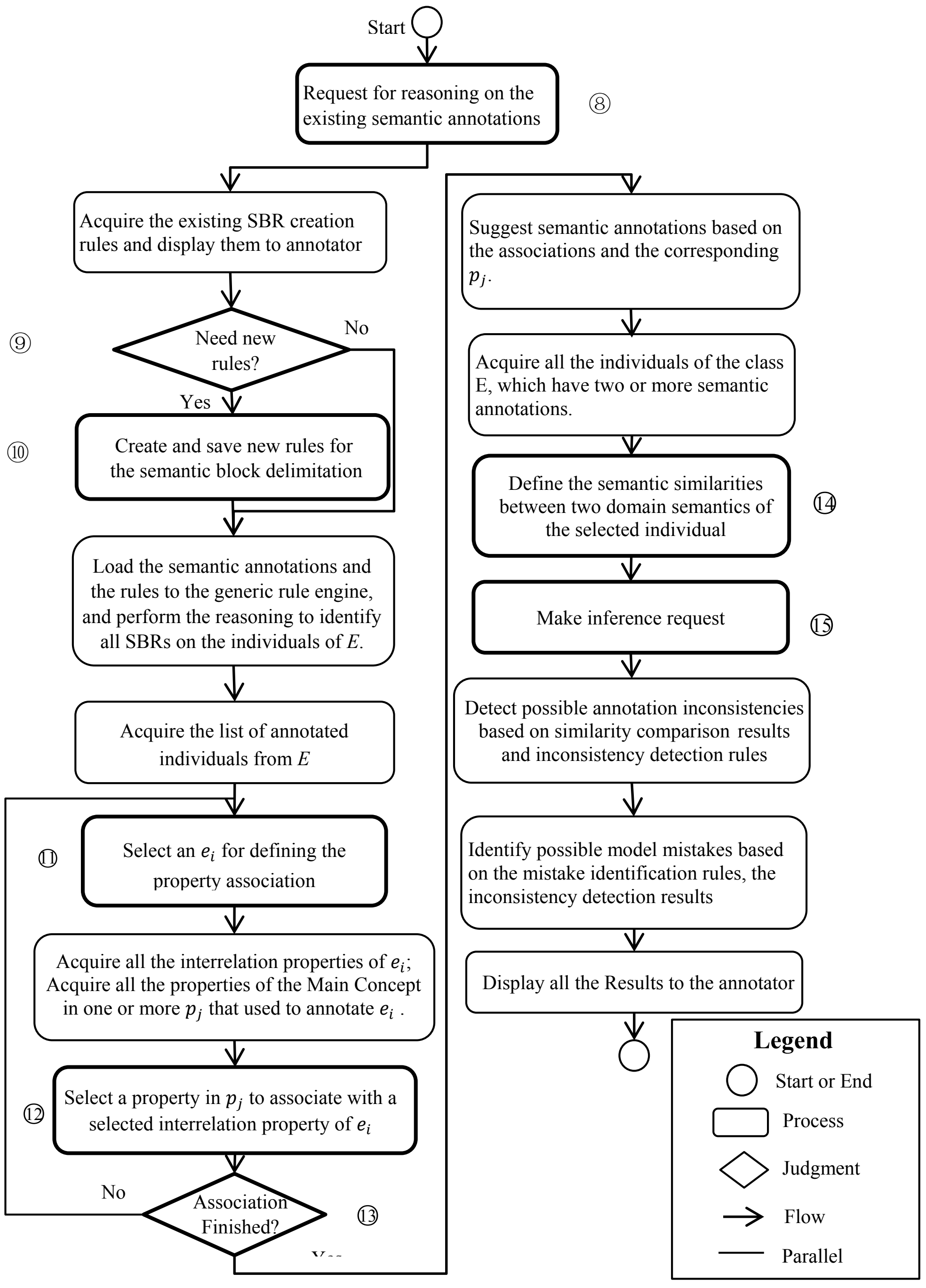
In the process ⑤, these query results are listed as candidates for the semantic block creation. Once a property is selected, an individual, which has got its name following the name syntax in the Figure 4-3, will be created in the Class "*SBRelations*". Then, the "*hasSBRelation*" property is used to define the relationship from the p_j to this individual. Meanwhile, the "*hasSBEntity*" property will be used to define the relationship from the p_j to the object of that selected property.

These query, selection and creation processes will continue until the semantic block delimitation is finished. Once the semantic block is created, one of the sub-properties of the Object Property "*SR*" can be used to define the semantic relationship between the e_i and the p_j. At this point, a semantic annotation for the model element ***X*** is created. And through this semantic annotation, the structure semantics and the domain semantics of the model element ***X*** can be made explicit.

There are some remarks need to be noted. In order to ease the complexity of Figure 4-6, the procedure presented in the figure does not contain the modification or deletion processes. It is possible to perform the process ② and the process ③ automatically via pre-defined mappings between the meta-models of the modelling environment and the selected meta-model ontology. It is also possible to perform semantic block delimitation in the process ⑤ automatically (for example, the method that is provide by Yahia et al. [118]). However, these are not implemented in the SAP-KM, because the main purpose of this research is not to design a number of automatic algorithms to assist the annotation, but to show all the steps and details of how the structure and domain semantics can be made explicit.

4.2.3 The Design of the Reasoning Phase

Once the semantic annotations of a TKR are created, they can be used together with the corresponding reasoning rules to produce the results of annotation suggestions, the annotation inconsistency detection and the model conflict identification through the reasoning process. The flowchart in the Figure 4-7 illustrates how SAP-KM supports an annotator to perform the reasoning on the semantic annotations and the corresponding reasoning rules. In the flowchart, the processes and judgements drawn with the thick line mean that the SAP-KM needs an annotator's participation, as well as the processes and judgements drawn with the thin line mean that it is automatically performed by the SAP-KM. To be more specific, based on the different stages of the preparation, this procedure is divided into two parts: (*i*) the property association to assist the suggestions of semantic annotations and (*ii*) the semantics similarity comparison to support the detection of annotation inconsistency and the identification of possible model content conflicts.


Start
Request for reasoning on the existing semantic annotations
⑧
Acquire the existing SBR creation rules and display them to annotator
⑨
Need new rules?
No
Yes
⑩
Create and save new rules for the semantic block delimitation
Load the semantic annotations and the rules to the generic rule engine, and perform the reasoning to identify all SBRs on the individuals of E.
Acquire the list of annotated individuals from E
⑪
Select an e_i for defining the property association
Acquire all the interrelation properties of e_i; Acquire all the properties of the Main Concept in one or more p_j that used to annotate e_i .
⑫
Select a property in p_j to associate with a selected interrelation property of e_i
No
Association Finished?
⑬
Yes
Suggest semantic annotations based on the associations and the corresponding p_j.
Acquire all the individuals of the class E, which have two or more semantic annotations.
Define the semantic similarities between two domain semantics of the selected individual
⑭
Make inference request
⑮
Detect possible annotation inconsistencies based on similarity comparison results and inconsistency detection rules
Identify possible model mistakes based on the mistake identification rules, the inconsistency detection results
Display all the Results to the annotator
Legend
Start or End
Process
Judgment
Flow
Parallel
Figure 4-7 The Procedure to perform the Inference

The path of those semantic annotations continues when the annotator requests to perform the reasoning (the process ⑧). As shown in the Figure 4-7, a number of existing SBR delimitation rules is loaded and displayed to the annotator through the SAP-KM. If these rules can not satisfy the needs of an annotator (the process ⑨), new rules can be created and saved through the SAP-KM (the process ⑩). The SBR delimitation rules and the existing semantic annotations will be loaded as inputs of the generic rule engine to perform the reasoning. The individuals of the Class “E”, which satisfy the rules, are identified. The corresponding SBRs that describe the implicit relations among these identified individuals will be added between them as the outputs of this reasoning process.

Once the above processes are finished, the property association can be stated. A list of annotated individuals in the class E is acquired by the SAP-KM. In the process ⑪, once an individual e_i from this list is selected, two query processes will start: (1) the first process will query all e_i's interrelation properties, which are composed of the generated SBRs that are related to the e_i and the properties that have been made explicit in the process ⑪; (2) the second process will query all the properties of the main concept of one or more p_j, which are used to annotate the e_i. Both the lists of properties are shown to the annotator for determining the associations (the process ⑫). Once the property association process is finished, the process ⑬ will lead the annotator to the suggestion stage. The SAP-KM takes the existing semantic annotations and the property associations' results as its inputs and generates the inferred semantic annotations.

After the suggestion of the inferred semantic annotations, the semantics similarity comparison can be performed. The SAP-KM acquires all the individuals in the Class "*E*", which has two or more semantic annotations and shows them to the annotator. In the process ⑭, the similarities between two domain semantics of the same annotated element (the selected individuals in the Class "*E*") will be defined. Once the comparison is finished, the inference request can be made (the process ⑮).

The SAP-KM loads the inconsistency detection rules, the existing semantic annotations and the comparison results as the inputs of the Jena generic rule reasoner. A list of possible inconsistencies among the elaborated semantic annotations will be produced. Then, the SAP-KM loads the model conflict identification rules, the existing semantic annotations and the inconsistency detection results as the inputs of the Jena generic rule reasoner again. The possible conflicts between those annotated model elements will be produced. At the end, all the results are presented to the annotator. At this point, the path of the semantic annotations in one TKR ends up.

Figure 4-7 shows the procedure to perform the inference, from the initial preparation until the final execution. In the process ⑫, it is possible to design a consultation mechanism, which processes the existing property association to support the creation of new property associations. It is also possible to perform the similarity comparison between two domain semantics in process ⑭ automatically. Since the similarity comparison is out of the scope of this research, we are not going to design the corresponding algorithms to support the automatic comparison in the SAP-KM.

4.3 The Implementation of the SAP-KM

As stated in Section 4.1.2, the SAP-KM is implemented as a plug-in of the MEGA modelling environment. The architecture of a MEGA plug-in written in Java is illustrated in the Figure 4-8. On the one hand, the plug-in is called by using a Macro MEGA. On the other hand, it uses the MEGA API to call MEGA application.

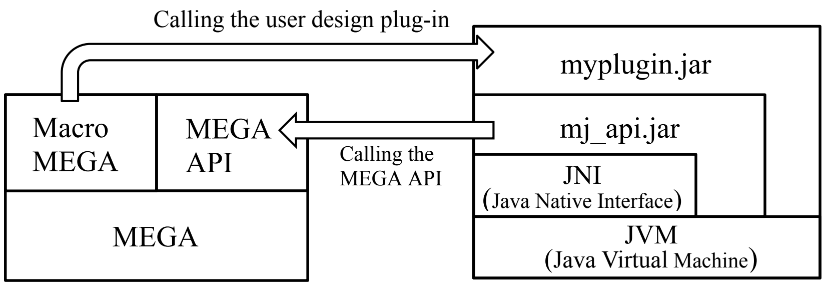
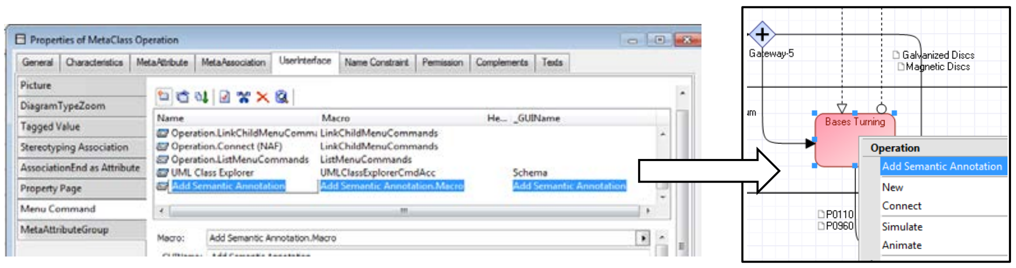

Calling the user design plug-in
myplugin.jar
Macro MEGA
MEGA API
mj_api.jar
Calling the MEGA API
JNI (Java Native Interface)
MEGA
JVM (Java Virtual Machine)

Figure 4-8 The Architecture of a MEGA Plug-in written in Java [122]

In order to enable the SAP-KM to assist the annotator to perform the annotation directly on the model diagram, as shown in the Figure 4-9, we configured the "Menu Command" property of several meta-model elements of the process model in the MEGA repository (such as operation, sequence flow, data object, etc.) and to create the corresponding macro references that enable the calling of the SAP-KM. Furthermore, with the assistance of the MEGA API, the SAP-KM is able to retrieve corresponding information of model elements that are necessary for semantic annotations from the MEGA. The objects and interfaces that are provided by this API can be applied by importing the corresponding packages into the project library.


Properties of MetaClass Operation

Name	Macro	He...	_GUIName
Operation.LinkChildMenuComm	LinkChildMenuCommands		
Operation.Connect (NAF)	LinkChildMenuCommands		
Operation.ListMenuCommands	ListMenuCommands		
UML Class Explorer	UMLClassExplorerCmdAcc		Schema
Add Semantic Annotation	Add Semantic Annotation.Macro		Add Semantic Annotation

Macro: Add Semantic Annotation.Macro

Configure the Menu Command and Macro Reference

The Right Click Menu of the "Operation"

Figure 4-9 The Configuration of the Menu Command and the Reference of the Macro

Through this implementation, an annotator can directly perform the annotation on a model diagram and the SAP-KM can retrieve the necessary information about a model it annotates. This interaction supports the annotator to perform the process ① in the Figure 4-6 and the process ⑧ in the Figure 4-7.

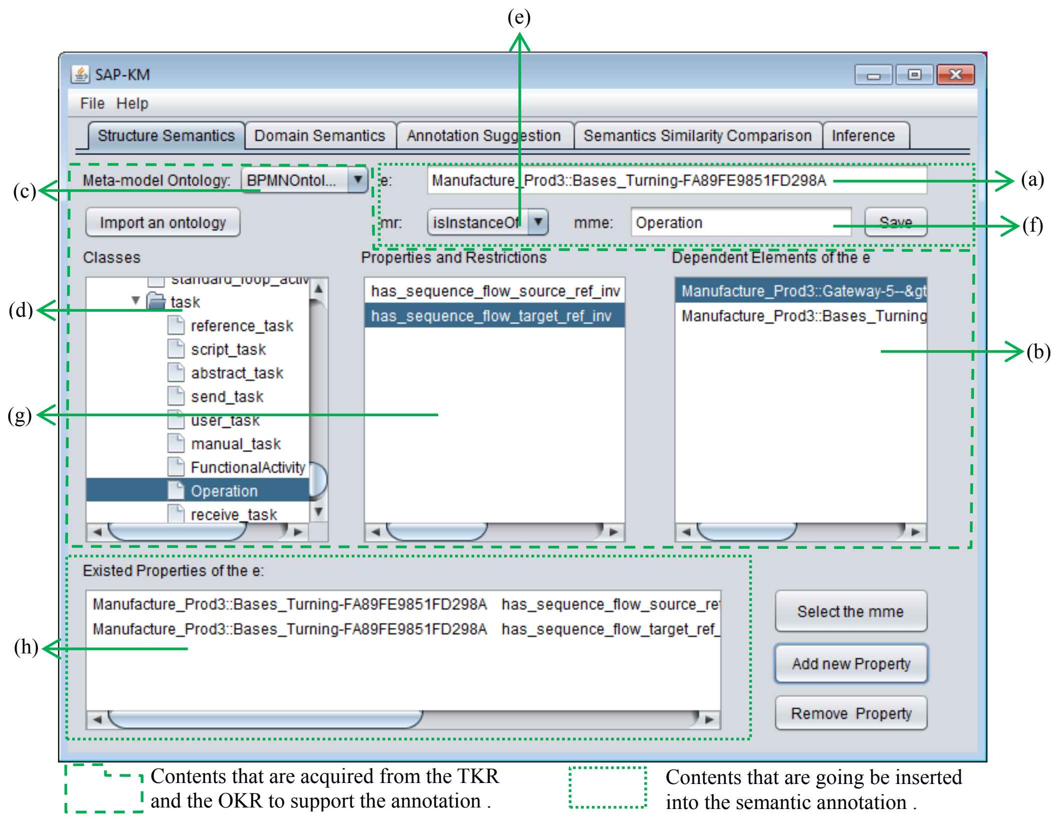
The SAP-KM also takes advantage from the Java libraries that are provided by the Jena API, which give a powerful support for the management of ontologies. In order to perform the reasoning on semantic annotations, the Semantic Annotation Schema is represented as an ontology that is serialized as an RDF/XML file. As we stated in Section 4.1.2, during the annotation process, the SAP-KM will load the Semantic Annotation Schema into an ontology model for all the operations.

Based on the design of the data structure, the design of the annotation procedures and the design of the reasoning, we implemented three specifics modules in the SAP-KM: (1) The explicitation of structure and domain semantics (Section 4.3.1); (2) The preparation and the execution of reasoning (Section 4.3.2); (3) The elements matching and data querying (Section 4.3.3). The previous two modules are used to apply semantic annotations in one TKR. The major graphical user interfaces in these two modules are "Structure Semantics", "Domain Semantics", "Annotation Suggestion", "Semantic Similarity Comparison" and "Inference". The last module is used to extend the usages of semantic annotation to other stages of a product life cycle. The major graphical user interfaces in this module are "Elements Matching" and "Data Querying".

4.3.1 The Explicitation of Structure and Domain Semantics

4.3.1.1 The Explicitation of Structure Semantics

The graphical user interface under the "Structure Semantic" tabbed pane is in charge of making explicit the structure semantics. This user interface enables an annotator to perform the process ② and the process ③ in the Figure 4-6. As shown in the Figure 4-10, the tasks of the SAP-KM are divided into two parts: (1) acquiring the information of the selected model element and its dependent elements from the model through the MEGA API and (2) loading a selected meta-model ontology from the knowledge cloud through the Jena API.


(e)
SAP-KM
File Help
Structure Semantics
Domain Semantics
Annotation Suggestion
Semantics Similarity Comparison
Inference
Import an ontology
Classes
Properties and Restrictions
Dependent Elements of the e
Existed Properties of the e:
Select the mme
Add new Property
Remove Property
Save
(a)
(b)
(c)
(d)
(f)
(g)
(h)
Contents that are acquired from the TKR and the OKR to support the annotation .
Contents that are going be inserted into the semantic annotation .

Meta-model Ontology: BPMNOntol...
e: Manufacture_Prod3::Bases_Turning-FA89FE9851FD298A
mr: isInstanceOf
mme: Operation

Classes
- standard_loop_activ
- task
  - reference_task
  - script_task
  - abstract_task
  - send_task
  - user_task
  - manual_task
  - FunctionalActivity
  - Operation
  - receive_task

Properties and Restrictions
- has_sequence_flow_source_ref_inv
- has_sequence_flow_target_ref_inv

Dependent Elements of the e
- Manufacture_Prod3::Gateway-5--&gt
- Manufacture_Prod3::Bases_Turning

Existed Properties of the e:

```
Manufacture_Prod3::Bases_Turning-FA89FE9851FD298A   has_sequence_flow_source_re
Manufacture_Prod3::Bases_Turning-FA89FE9851FD298A   has_sequence_flow_target_ref_
```

Figure 4-10 The Graphical User Interface of the Explicitation of Structure Semantics

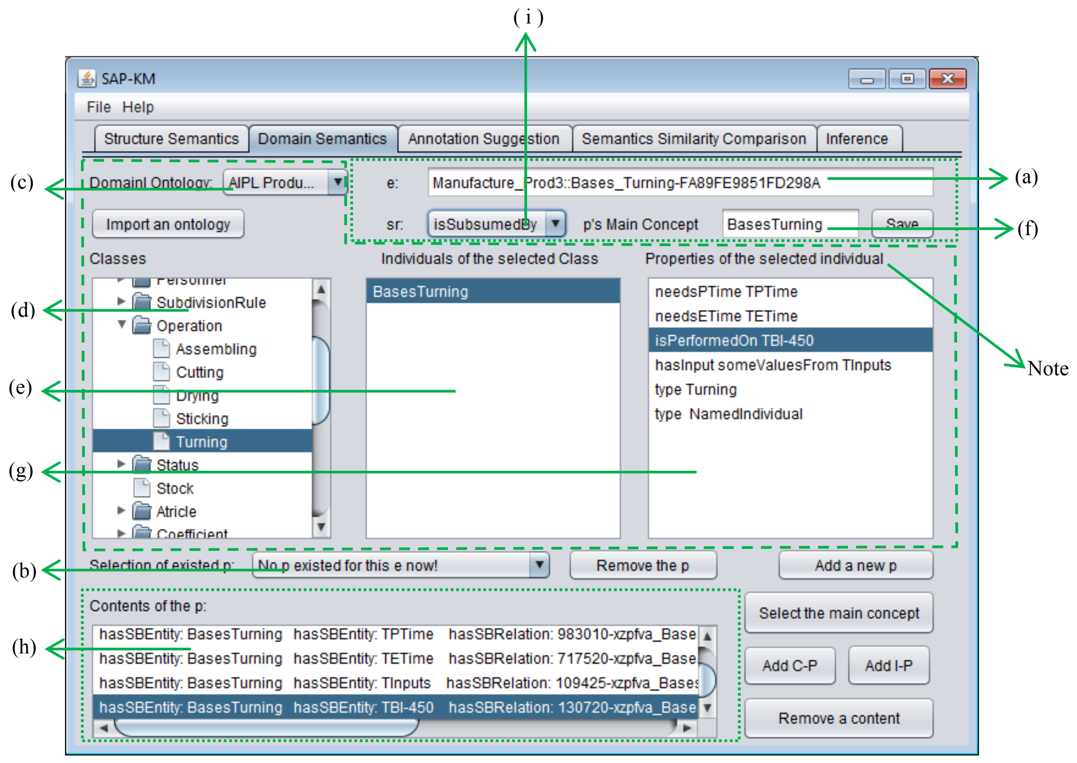
For the first task, the SAP-KM acquires the name and the identifier of the model element, we called it as "***e***" in the interface, which requested a semantic annotation (the (a) in the Figure 4-10). The names and the identifiers of its dependent elements are also acquired (the (b) in the Figure 4-10). These acquired information are used as inputs of the name construct algorithm, which is based on the name syntax in the Figure 4-4. An existence checking takes place to verify whether the same individuals exist in the Class "*E*". If some/all of them do not exist, the SAP-KM will use their constructed names to create the corresponding new individuals into the Class "*E*".

For the second task, SAP-KM acquires the list of classes in the imported meta-model ontology (the (c) in the Figure 4-10) and shows them as a tree view (the (d) in the Figure 4-10). In this way, the annotator can browse all the candidate ontology elements and then select a class. Through the action of selecting the semantic relationships (the (e) in the Figure 4-10) and clicking the button '*select the mme*', ***e*** will be defined as an instance of the selected class (the (f) in the Figure 4-10). Meanwhile, the corresponding property restrictions of the selected class are listed (the (g) in the Figure 4-10) as candidates for highlighting the coherent properties between ***e*** and one of ***e***'s dependent elements. All the existing properties of ***e*** are shown at the bottom of this interface (the (h) in the Figure 4-10).

Once the explicitation of the structure semantics is finished, the annotator can save the annotation results into the ontology model that keeps the semantic annotations. In the current version of the SAP-KM, each model element in a model can only be annotated by one class from the meta-model ontology. When the annotator wants to add a new semantic annotation on the same model element, the SAP-KM will acquire the existing structure semantics of this model element and shows it in this tabbed pane for the modification.

4.3.1.2 The Explicitation of Domain Semantics

The graphical user interface under the "Domain Semantic" tabbed pane is in charge of the explicitation of domain semantics. This user interface enables an annotator to perform the processes ④, ⑤, ⑥ and ⑦ in the Figure 4-6. As shown in the Figure 4-11, the tasks of the SAP-KM are divided into two parts: (1) acquiring the value of ***e*** from the previous annotation process and (2) loading PLC-related ontologies from the knowledge cloud trough the Jena API.



DomainI Ontology: AIPL Produ...

e: Manufacture_Prod3::Bases_Turning-FA89FE9851FD298A

sr: isSubsumedBy p's Main Concept BasesTurning

Classes: Personnel, SubdivisionRule, Operation (Assembling, Cutting, Drying, Sticking, Turning), Status, Stock, Atricle, Coefficient

Individuals of the selected Class: BasesTurning

Properties of the selected individual:

needsPTime TPTime
needsETime TETime
isPerformedOn TBI-450
hasInput someValuesFrom TInputs
type Turning
type NamedIndividual

Selection of existed p: No p existed for this e now!

Contents of the p:

```
hasSBEntity: BasesTurning  hasSBEntity: TPTime   hasSBRelation: 983010-xzpfva_Base
hasSBEntity: BasesTurning  hasSBEntity: TETime   hasSBRelation: 717520-xzpfva_Base
hasSBEntity: BasesTurning  hasSBEntity: TInputs  hasSBRelation: 109425-xzpfva_Bases
hasSBEntity: BasesTurning  hasSBEntity: TBI-450  hasSBRelation: 130720-xzpfva_Base
```

Contents that are acquired from the OKR to support the annotation .

Contents that are going be inserted into the semantic annotation .

Note: The text in the label on top of the list (the one on the right hand side) is changed based on the annotator 's selection (for example, when annotator click a class in the tree view, the text of this label will be "Property restrictions of the selected class")

Figure 4-11 The Graphical User Interface of the Explicitation of Domain Semantics

For the first task, the value of ***e*** (the (a) in the Figure 4-11) is acquired and used as the basis to query whether there is a sub-property of the Object Property "*SR*" that relates from this $\boldsymbol{e}$ to any individuals in the Class "*P*" (we call it $\boldsymbol{p}$ in the interface). If there exists one or more $\boldsymbol{p}$, the name of $\boldsymbol{p}$ will be added into the combo box (the (b) in the Figure 4-11). On the one hand, the "remove the p" button will erase the property between the $\boldsymbol{e}$ and the selected $\boldsymbol{p}$ and on the other hand, the "add a new p" button will create a new $\boldsymbol{p}$ as an individual of the Class "*P*" and will wait for the determination of semantic relationship. The value of the $\boldsymbol{e}$ and the value of the combo box item are used as input of the name construct algorithm to produce the name of the new $\boldsymbol{p}$.

For the second task, the SAP-KM acquires the list of classes in the imported PLC-related ontology (the (c) in the Figure 4-11). These classes are showed as a tree view (the (d) in the Figure 4-11) for an annotator to browse and select. After selecting a class, the SAP-KM retrieves its individuals and shows them in the list view in the middle of the interface (the (e) in the Figure 4-11). In this way, the annotator can locate all the concepts that are contained in this ontology. According to the data structure design in Section 4.2.1, a $\boldsymbol{p}$ has three mandatory Object Properties, namely "*hasMainConcept*", "*hasSBEntity*" and "*hasSBRelation*". In the following paragraphs we will present the corresponding implementation.

(1) For the "*hasMainConcept*" property: both classes and individuals are candidates for the selection of the main concept of $\boldsymbol{p}$ (the (f) in the Figure 4-11). After selecting a class or an individual and clicking the button "select the main concept", this Object Property will be added from $\boldsymbol{p}$ to the selected class or from $\boldsymbol{p}$ to the selected individual.

(2) For the "*hasSBEntity*" property and the "*hasSBRelation*" property: the list view on the right hand side (the (g) in the Figure 4-11) is in charge of listing the corresponding properties (or property restrictions). Let us take the case that an individual and one of its properties are selected as an example. After clicking on the button "add I-P" (add Individual and its Property), the object property "*hasSBEntity*" will be added from $\boldsymbol{p}$ to the selected individual and from $\boldsymbol{p}$ to the object that the selected property points to. Meanwhile, a new individual that describes the selected property will be created in the Class "*SBRelations*", through name construct algorithm. The object property "*hasSBRelation*" will be added from $\boldsymbol{p}$ to this new individual. The Pseudo Code 1 in the APPENDIX III shows this procedure. A similar process is performed on a class and its property restrictions, through clicking the "add C-P" (add Class and its Property restriction) button.

All the established selection results will be shown in a list view at the bottom of this interface (the (h) in the Figure 4-11). Once this semantic block delimitation process (the selection of the contents of the ***p***) is finished, a sub property of Object Property "SR" (the (i) in the Figure 4-11) can be used to describe the semantic relationship between the ***e*** and the ***p***. Finally, the annotation results will be saved into the ontology model that keeps the semantic annotations. Meanwhile, the Object Property "*isAnnotatedBy*" is also added from the individual ***e*** to the individual ***p*** when the annotation result is saved.

In the current version of the SAP-KM, each model element can be annotated by multiple ***p***. When an annotator wants to modify an existing ***p***, through selecting the ***p*** in the combo box (the (b) in the Figure 4-11), the SAP-KM will acquire the existing domain semantics and will show its contents in a list view at the bottom of the interface (the (h) in the Figure 4-11). One more issue needs to be mentioned is that the current version of the SAP-KM cannot deal with the complex property restriction of a class, but only with the normal expression, such as *Turing* (Class) <u>*isPerformedOn*</u> (Property) *allValuesFrom Lathe* (Class).

4.3.2 The Preparation and Execution of Reasoning

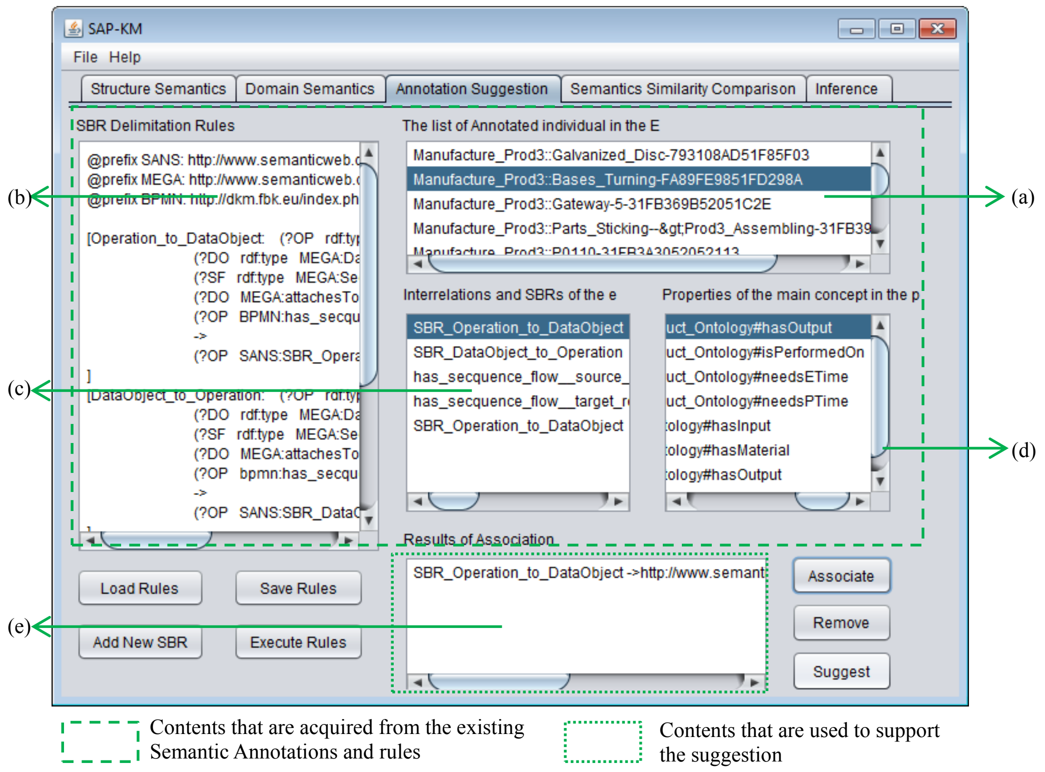
As we stated in Section 3.3.1, the reasoning process is based on the reasoning rules and the corresponding reasoning parameters.

Concerning the reasoning rules, we designed three kinds of rules (the annotation inconsistency detection rules, the model content conflict identification rules and several SBR delimitation rules) and an annotation suggestion algorithm. The creation of these three kinds of rules follow the syntax of Jena Rules [123], which is presented in APPEDIX I .

Concerning the reasoning parameters, the SAP-KM needs the results of the determination of the association between two properties (for the annotation suggestions), the results of the similarity comparison between two domain semantics of a common annotated object (for the detection of annotation inconsistencies) and results of the detection of annotation inconsistencies (for the identification of possible conflicts between annotated model elements).

4.3.2.1 The Annotation Suggestion

The graphical user interface under the "Annotation Suggestion" tabbed pane is in charge of the delimitation of SBRs and the suggestion of inferred semantic annotations. This user interface enables an annotator to perform the processes 9, 10, 11, 12 and 13 in the Figure 4-7.


SAP-KM
File Help
Structure Semantics
Domain Semantics
Annotation Suggestion
Semantics Similarity Comparison
Inference
SBR Delimitation Rules
The list of Annotated individual in the E
Interrelations and SBRs of the e
Properties of the main concept in the p
Results of Association
Load Rules
Save Rules
Add New SBR
Execute Rules
Associate
Remove
Suggest
(a)
(b)
(c)
(d)
(e)
Contents that are acquired from the existing Semantic Annotations and rules
Contents that are used to support the suggestion

```
@prefix SANS: http://www.semanticweb.c
@prefix MEGA: http://www.semanticweb.c
@prefix BPMN: http://dkm.fbk.eu/index.ph

[Operation_to_DataObject:  (?OP  rdf:typ
                  (?DO  rdf:type  MEGA:Da
                  (?SF  rdf:type  MEGA:Se
                  (?DO  MEGA:attachesTo
                  (?OP  BPMN:has_secqu
                  ->
                  (?OP  SANS:SBR_Opera
]
[DataObject_to_Operation:  (?OP  rdf:typ
                  (?DO  rdf:type  MEGA:Da
                  (?SF  rdf:type  MEGA:Se
                  (?DO  MEGA:attachesTo
                  (?OP  bpmn:has_secqu
                  ->
                  (?OP  SANS:SBR_DataC
]
```

```
Manufacture_Prod3::Galvanized_Disc-793108AD51F85F03
Manufacture_Prod3::Bases_Turning-FA89FE9851FD298A
Manufacture_Prod3::Gateway-5-31FB369B52051C2E
Manufacture_Prod3::Parts_Sticking--&gt;Prod3_Assembling-31FB39
Manufacture_Prod3::P0110-31FB3A3052052113
```

```
SBR_Operation_to_DataObject
SBR_DataObject_to_Operation
has_secquence_flow__source_
has_secquence_flow__target_r
SBR_Operation_to_DataObject
```

```
uct_Ontology#hasOutput
uct_Ontology#isPerformedOn
uct_Ontology#needsETime
uct_Ontology#needsPTime
tology#hasInput
tology#hasMaterial
tology#hasOutput
```

```
SBR_Operation_to_DataObject->http://www.semant
```

Figure 4-12 The Graphical User Interface of the Annotation Suggestion

As shown in the Figure 4-12 (a), the SAP-KM makes the queries to the individuals in the Class "*E*", which has already been annotated, and shows them in the list view. The SBR delimitation rules are loaded and displayed to the annotator (the (b) in the Figure 4-12), so that the annotator can modify the existing rules or create some new rules. In the case of SBR rule creation, the annotator needs above all to click the "add New SBR" button to insert a new object property into the Semantic Annotation Schema, and then begins to edit the new SBR delimitation rules. For example, in the Figure 4-13, a rule to define a SBR that makes explicit one of the possible the relations between an instance of the operation and an instance of the data object is shown.

After the rule execution, these SBR delimitation rules participate in the parameterization of the generic rule reasoner. After that an instance of the reasoner, according to this parameterization, is created to perform the reasoning on the ontology model that keeps all the semantic annotations.

An inference model (the model keeps both the semantic annotations and the results of reasoning) is created after the reasoning step. In this ontology model, the SBRs are already set between two individuals in the Class "*E*", which fulfil all the conditions in the delimitation rules. When the annotator selects an individual from the listed annotated individuals (the (a) in the Figure 4-12), all its interrelations (the structure semantics that have been made explicit in the "Structure Semantic" tabbed pane) and the SBRs that related to it will be shown in one list view (the (c) in the Figure 4-12). Meanwhile, the SAP-KM makes the query to the existing domain semantics (***p***) of the selected individual, and then acquires all the properties that are related to the ***p***'s main concept and shows them in another list view (the (d) in the Figure 4-12). The pseudo code 2 and pseudo code 3 in the APPENDIX III show how the SAP-KM decomposes of a SBRelation and lists the properties of a ***p***'s main concept. The annotator can perform the association between two properties from these two list views. The results of the association are enumerated at the bottom of the interface (the (e) in the Figure 4-12). When the annotator clicks the button "suggest", the suggestion algorithm takes these results as input, and suggests the inferred semantic annotations. The pseudo code 4 in the APPENDIX III shows the annotation suggestion algorithm.

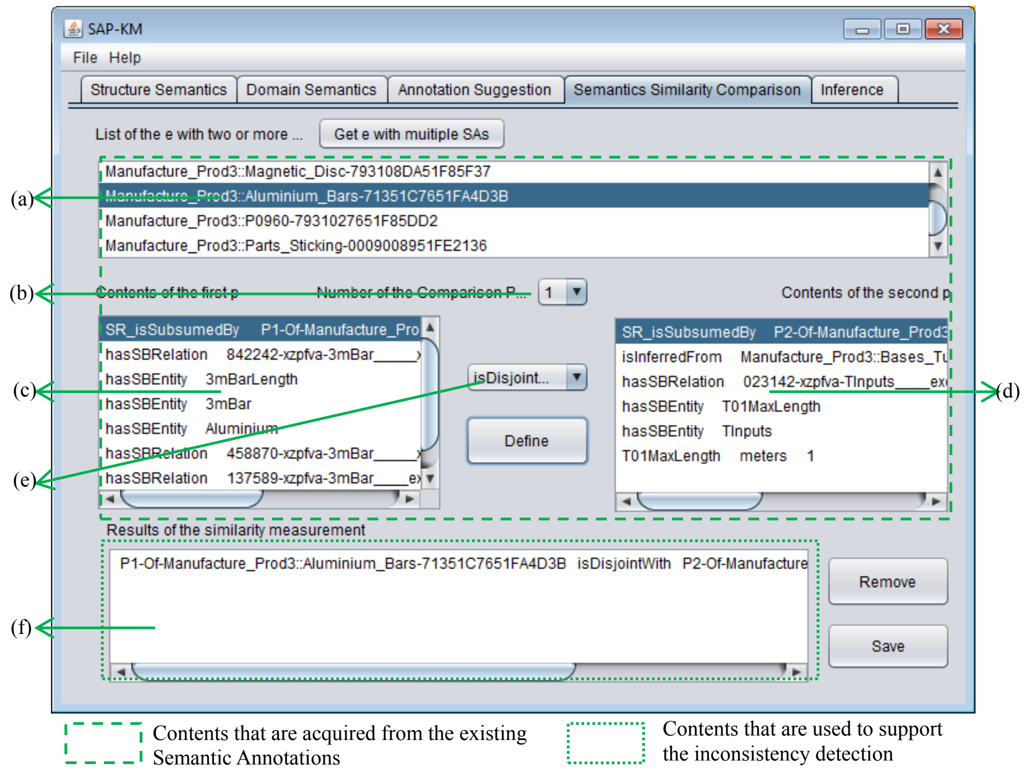
```
@prefix SANS: http://www.semanticweb.org/ontologies/2013/6/SemanticAnnotations#
@prefix MEGA: http://www.semanticweb.org/ontologies/2013/6/MEGA_BPMN#
@prefix BPMN: http://dkm.fbk.eu/index.php/BPMN_Ontology#

[Operation_to_DataObject:   (?OP   rdf:type   MEGA:Operation)
                            (?DO  rdf:type   MEGA:DataObject)
                            (?SF   rdf:type   MEGA:SequenceFlow)
                            (?DO  MEGA:attachesTo   ?SF)
                            (?OP   BPMN:has_secquence_flow_source_ref_inv   ?SF)
                            ->
                            (?OP   SANS:SBR_Operation _to_DataObject   ?DO)
]
```

Figure 4-13 A Forward Rule to define a SBR for making explicit a Relation

4.3.2.2 The Semantic Similarity Comparison

The graphical user interface under the "Semantic Similarity Comparison" tabbed pane is in charge of the similarity determination between two domain semantics of the same annotated ***e***. This user interface enables an annotator to perform the process ⑭ in the Figure 4-7.

As shown in the Figure 4-14 (a), the SAP-KM acquires all the individuals of the Class "*E*", which have two or more semantic annotations (both initial semantic annotations and inferred semantic annotations). In order to make the interface simpler, for each ***e***, if there exists more than two semantic annotations, the SAP-KM will create combination couples from those semantic annotations. The corresponding couple numbers are inserted into the combo box (the (b) in the Figure 4-14) for the selection. Once an ***e*** that has multiple semantic annotations is chosen, two domain semantics of this item will be shown in the list view (c) and list view (d) in the Figure 4-14 for the semantic similarity comparison. Five kinds of semantic relationships can be selected from the combo box in the middle (the (e) in the Figure 4-14). All the similarly comparison results are shown in at the bottom of the interface (the (f) in the Figure 4-14). These results will be inserted into the ontology model when the comparison is finished.


SAP-KM
File Help
Structure Semantics
Domain Semantics
Annotation Suggestion
Semantics Similarity Comparison
Inference
List of the e with two or more ...
Get e with muitiple SAs
Contents of the first p
Number of the Comparison P...
1
Contents of the second p
isDisjoint...
Define
Results of the similarity measurement
Remove
Save
(a)
(b)
(c)
(d)
(e)
(f)
Contents that are acquired from the existing Semantic Annotations
Contents that are used to support the inconsistency detection

```
Manufacture_Prod3::Magnetic_Disc-793108DA51F85F37
Manufacture_Prod3::Aluminium_Bars-71351C7651FA4D3B
Manufacture_Prod3::P0960-7931027651F85DD2
Manufacture_Prod3::Parts_Sticking-0009008951FE2136
```

```
SR_isSubsumedBy   P1-Of-Manufacture_Pro
hasSBRelation   842242-xzpfva-3mBar_____x
hasSBEntity   3mBarLength
hasSBEntity   3mBar
hasSBEntity   Aluminium
hasSBRelation   458870-xzpfva-3mBar_____x
hasSBRelation   137589-xzpfva-3mBar____ex
```

```
SR_isSubsumedBy   P2-Of-Manufacture_Prod3
isInferredFrom   Manufacture_Prod3::Bases_Tu
hasSBRelation   023142-xzpfva-TInputs____ex
hasSBEntity   T01MaxLength
hasSBEntity   TInputs
T01MaxLength   meters   1
```

```
P1-Of-Manufacture_Prod3::Aluminium_Bars-71351C7651FA4D3B  isDisjointWith  P2-Of-Manufacture
```

Figure 4-14 The Graphical User Interface of the Semantic Similarity Comparison

4.3.2.3 The Inference

After the above-mentioned processes, the final inference can be performed. The graphical user interface under the "Inference" tabbed pane is in charge of starting the inference and displaying the inference results to the annotator. This user interface enables an annotator to perform the process ⑮ in the Figure 4-7.

The inference process is based on the existing semantic annotations, the semantic similarity comparison results, and two kinds of reasoning rules. These rules follow the annotation inconsistency detection mechanism and the model conflict identification mechanism, which are presented in the Table 3-1 and Table 3-2 in Section 3.2.2. Each column and each row of the two tables are created as reasoning rules.

Two examples of these two kinds of rules are shown in the Figure 4-15. According to the rule (a), ?Ei is an individual of the Class "*E*". ?Px and ?Py are the individuals of Class "*P*". Given the flowing three conditions are all satisfied:

- the semantic relationship between ?Ei and ?Px is "*SR_isSubsumedBy*";
- the semantic relationship between ?Ei and ?Py is "*SR_isSubsumedBy*";

- the semantic similarity between ?Px and ?Py is "*PR_isDisjointWith*".

The Object Property "*isNotConsistentwith*" will be added from ?Px to ?Py for denoting the inconsistency between them.

According to the rule (b), ?Ei and ?Ek are individuals of the Class "*E*". ?Px and ?Py are the individuals of Class "*P*". Given the flowing four conditions are all satisfied:

- ?Ei is annotated by ?Px;
- ?Ei is annotated by ?Py;
- ?Py is inferred from the semantic annotation of ?Ek ;
- ?Px and ?Py are marked as inconsistent to each other.

The Object Property "isConflictWith" will be added from ?Ei to ?Ek for denoting there is a possible conflict between them.

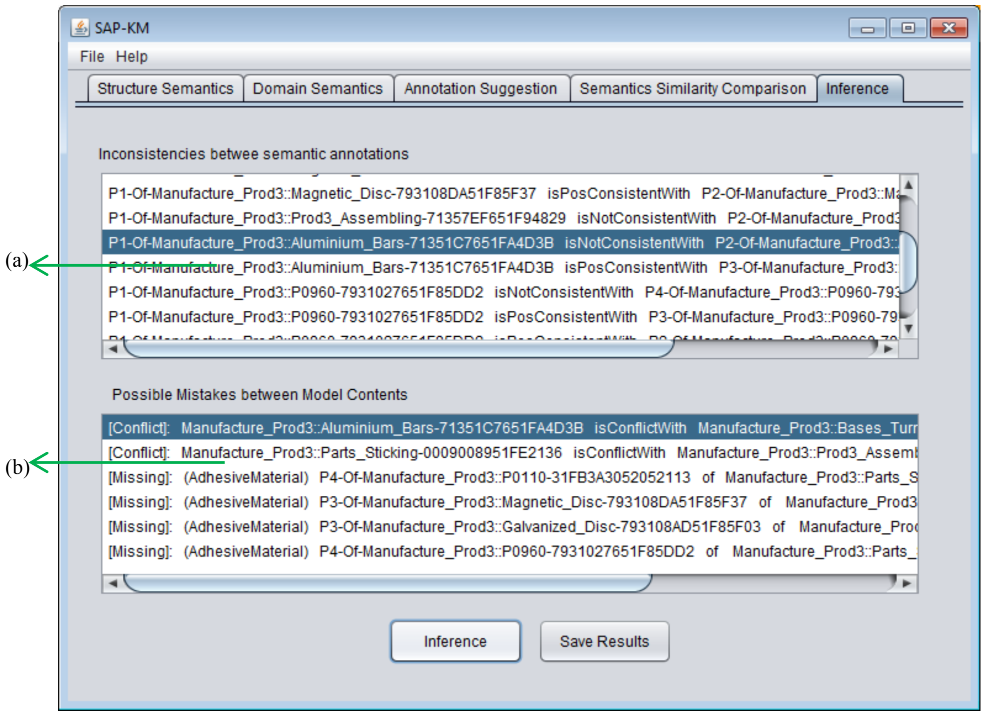
```
@prefix SANS: http://www.semanticweb.org/ontologies/2013/6/SemanticAnnotations#

[possible_inconsistance_table_row3_column3_5: (?Px  rdf:type  SANS:P)
                             (?Py  rdf:type  SANS:P)
                             (?Ei  rdf:type  SANS:E)
                             (?Ei  SANS:SR_isSubsumedBy  ?Px)
                             (?Ei  SANS:SR_isSubsumedBy  ?Py)
                             (?Px  SANS:PR_isDisjointWith  ?Py)
                             ->
                             (?Px  SANS:isNotConsistentWith  ?Py)
]
```
(a)

```
@prefix SANS: http://www.semanticweb.org/ontologies/2013/6/SemanticAnnotations#

[possible_wrong_table_row2_column1: (?Ei   rdf:type   SANS:E)
                         (?Ek   rdf:type   SANS:E)
                         (?Px  rdf:type   SANS:P)
                         (?Py  rdf:type   SANS:P)
                         (?Ei   SANS:isAnnotatedBy   ?Px)
                         (?Ei   SANS:isAnnotatedBy   ?Py)
                         (?Py  SANS:isInferredFrom   ?Ek)
                         (?Px  SANS:isNotConsistentWith  ?Py)
                         ->
                         (?Ei   SANS:isConflictWith ?Ek)
]
```
(b)

Figure 4-15 Two Examples of the Rules for Annotation Inconsistency Detection and Model Conflict Identification

However, because of the rule syntax limitation, these rules can only deal with the simple case (each inferred semantic annotation of an annotated object is suggested by one of the other model elements), but not the complex case (multiple inferred semantic annotation of an annotated object are suggested by one of the other model elements). Besides these rules, an algorithm is carried out in the SAP-KM to discover whether all the required conditions that are suggested in the inferred semantic annotation are satisfied. The Pseudo Code 5 in APPENDIX III shows this algorithm, which is one possible way to deal with the complex case.

Finally, the inconsistent annotations (the (a) in the Figure 4-16) and the possible model content mistakes (the (b) in the Figure 4-16), are generated and shown to the annotator. These results are used to assist annotators in the detection of the inconsistencies between existing semantic annotations and the identification of the possible conflicts between the annotated model elements. Different from the SBR delimitation rules, which are limited to their own specific context, these two kinds of reasoning rules can be applied on any kind of models. Although these rules are not shown in the graphical user interface, the annotators are able to view and edit them in the Knowledge Cloud.



Inconsistencies betwee semantic annotations

```
P1-Of-Manufacture_Prod3::Magnetic_Disc-793108DA51F85F37  isPosConsistentWith  P2-Of-Manufacture_Prod3::M
P1-Of-Manufacture_Prod3::Prod3_Assembling-71357EF651F94829  isNotConsistentWith  P2-Of-Manufacture_Prod3
P1-Of-Manufacture_Prod3::Aluminium_Bars-71351C7651FA4D3B  isNotConsistentWith  P2-Of-Manufacture_Prod3::
P1-Of-Manufacture_Prod3::Aluminium_Bars-71351C7651FA4D3B  isPosConsistentWith  P3-Of-Manufacture_Prod3:
P1-Of-Manufacture_Prod3::P0960-7931027651F85DD2  isNotConsistentWith  P4-Of-Manufacture_Prod3::P0960-79
P1-Of-Manufacture_Prod3::P0960-7931027651F85DD2  isPosConsistentWith  P3-Of-Manufacture_Prod3::P0960-79
```

Possible Mistakes between Model Contents

```
[Conflict]:  Manufacture_Prod3::Aluminium_Bars-71351C7651FA4D3B  isConflictWith  Manufacture_Prod3::Bases_Tur
[Conflict]:  Manufacture_Prod3::Parts_Sticking-0009008951FE2136  isConflictWith  Manufacture_Prod3::Prod3_Assem
[Missing]:  (AdhesiveMaterial)  P4-Of-Manufacture_Prod3::P0110-31FB3A3052052113  of  Manufacture_Prod3::Parts_S
[Missing]:  (AdhesiveMaterial)  P3-Of-Manufacture_Prod3::Magnetic_Disc-793108DA51F85F37  of  Manufacture_Prod3
[Missing]:  (AdhesiveMaterial)  P3-Of-Manufacture_Prod3::Galvanized_Disc-793108AD51F85F03  of  Manufacture_Pro
[Missing]:  (AdhesiveMaterial)  P4-Of-Manufacture_Prod3::P0960-7931027651F85DD2  of  Manufacture_Prod3::Parts_
```

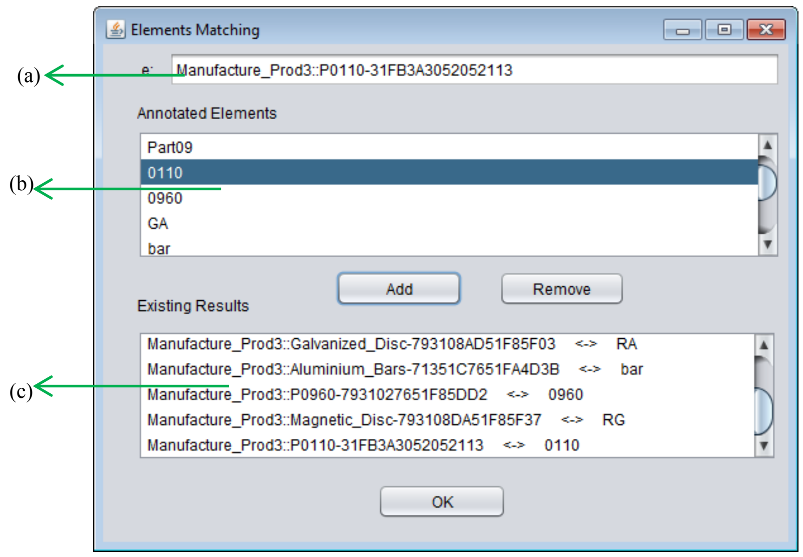
Figure 4-16 The Graphical User Interface for Inference Reasoning

At this point, the path of semantic annotations in one TKR is ended. In order to apply the proposed solution in a PLM environment, the SAP-KM needs to be able to exchange and share the semantic annotations with the plug-ins of the other systems.

4.3.3 The Elements Matching and Data Querying

The SAP-KM allows semantic annotation to participate in other stages of a product life cycle in which the semantic annotations are passed from one system to another together with the information that they exchange. It implements two extended functions: (1) Elements Matching Function, which enables the SAP-KM to reuse the semantic annotations that are created by other systems and (2) Data Querying Function, which enables the SAP-KM to show the data that are kept in the semantic annotations.

The graphical user interface "Elements Matching" (Figure 4-17) is in charge of the first function. When the SAP-KM receives semantic annotations from another system, the annotator is able to perform the matching between the selected element in the current model (the (a) in the Figure 4-17) and the annotated elements in the model that is created by that system. The matching action result declares that the two selected individuals represent the same domain semantics. It enables the selected element in the current model to inherit the domain semantics of the matched element. The Figure 4-17 (b) shows a list of annotated elements from one or more model systems. They are used as candidates for the elements matching. The Figure 4-17 (c) shows all the elements matching results.



e: Manufacture_Prod3::P0110-31FB3A3052052113

Annotated Elements

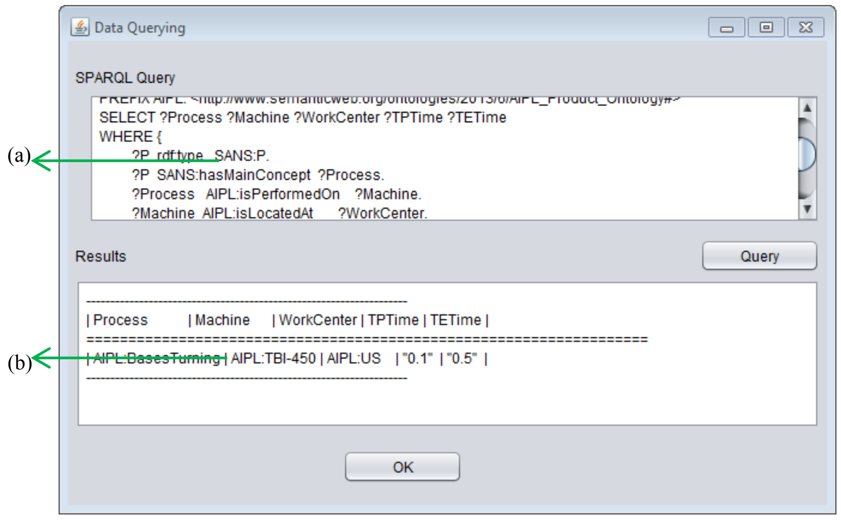
Part09
0110
0960
GA
bar

Existing Results

```
Manufacture_Prod3::Galvanized_Disc-793108AD51F85F03   <->   RA
Manufacture_Prod3::Aluminium_Bars-71351C7651FA4D3B   <->   bar
Manufacture_Prod3::P0960-7931027651F85DD2   <->   0960
Manufacture_Prod3::Magnetic_Disc-793108DA51F85F37   <->   RG
Manufacture_Prod3::P0110-31FB3A3052052113   <->   0110
```

Figure 4-17 The Graphical User Interface for Elements Matching

The graphical user interface "Data Querying" is in charge of the second function. Taking advantages from the Jena query package that is provided by Jena API, the SAP-KM can use the SPARQL queries to acquire the data that are kept in the existing semantic annotations. Figure 4-18 (a) shows an example of the SPARQL queries. Figure 4-18 (b) shows the query results.



```
PREFIX AIPL: <http://www.semanticweb.org/ontologies/2013/6/AIPL_Product_Ontology#>
SELECT ?Process ?Machine ?WorkCenter ?TPTime ?TETime
WHERE {
      ?P  rdf:type   SANS:P.
      ?P  SANS:hasMainConcept  ?Process.
      ?Process   AIPL:isPerformedOn    ?Machine.
      ?Machine  AIPL:isLocatedAt      ?WorkCenter.
```

```
-------------------------------------------------------------------
| Process         | Machine     | WorkCenter | TPTime | TETime |
===================================================================
| AIPL:BasesTurning | AIPL:TBI-450 | AIPL:US    | "0.1"  | "0.5"  |
-------------------------------------------------------------------
```

Figure 4-18 The Graphical User Interface for Data Querying

In short, these two extended functions enable the SAP-KM to be used in a cooperative situation: (1) to manipulate the semantic annotations that it received and (2) to display the data that is kept in the existing semantic annotations.

4.4 Conclusion

The development of the SAP-KM demonstrates the possibility for applying the proposed solution into the real-life applications. Although the presented design and the implementation of the SAP-KM is only one of the possible realizations of the formal approach, it can be easily extended to be applied on others types of models. Because (1) it has not any meta-model constraints in the semantic annotation model; (2) MEGA is an integration modelling environment that provides possible assistance to different kinds of model. The SAP-KM can be extended to other kinds of models in MGEA through adding macros into the related meta-classes. The issue left behind is to create the SBR delimitation rules that correspond to the different model specifications.

Applying this prototype to other modelling environments require the selected system has the ability to provide sufficient APIs that enable the plug-in to be launched and allow it to retrieve necessary model information. In the next chapter, an information flow in a particular application scenario is chosen as the background of the case study. An example of managing semantic annotation in the current system, exchanging and sharing semantic annotations with upstream and downstream systems along the product lifecycle is presented.

Chapter 5 Case Study

In order to apply our proposed solution with the SAP-KM in a PLM environment, a case study is developed in this chapter. Section 5.1 presents the context of the case study. It introduces the background of an AIPL[19] product lifecycle and a selected application scenario with different models related to the proposed product. Then Section 5.2 presents three main phases of the semantic annotation procedure: Section 5.2.1 introduces the preparation of TKRs and OKRs; Section 5.2.2 illustrates the annotation processes in the current system (a system in use in a specific point of an information flow), which takes into account the semantic annotations from the upstream system (according to the information flow, it is the system that is placed before the current system) and for the downstream system (according to the information flow, it is the system that is placed after the current system); Section 5.2.3 presents the reasoning processes based on the existing semantic annotations and the reasoning rules. Finally, a conclusion of the case study demonstrates the use of our proposed approach and tools for assisting the model creation during the product engineering life cycle.

5.1 The Context of Case study

5.1.1 A Product Lifecycle at the AIPL

In order to show how semantic annotations are able to contribute to the systems interoperability in a PLM environment, a small scale facility for manufacturing products, named AIPL, has been chosen as the context of this case study. In this facility, as shown in the Figure 5-1, various kinds of enterprise systems are participating and interoperating together at the engineering side and at the execution one. The complex information flows go through the entire life cycle of its products. The used systems can be divided into two groups: the one referring to the engineering stage and the other referring to the manufacturing stage. The former group of systems are in charge of designing models of the products, the processes, the operation interfaces and all the others product components and services that will be used by the production units. The latter group of systems will use those models to perform the parameterization of some application software and to apply them into the reality of the manufacturing system.

[19] AIPL (Atelier Inter-Etablissements de Lorraine): http://www.aip-primeca.net/

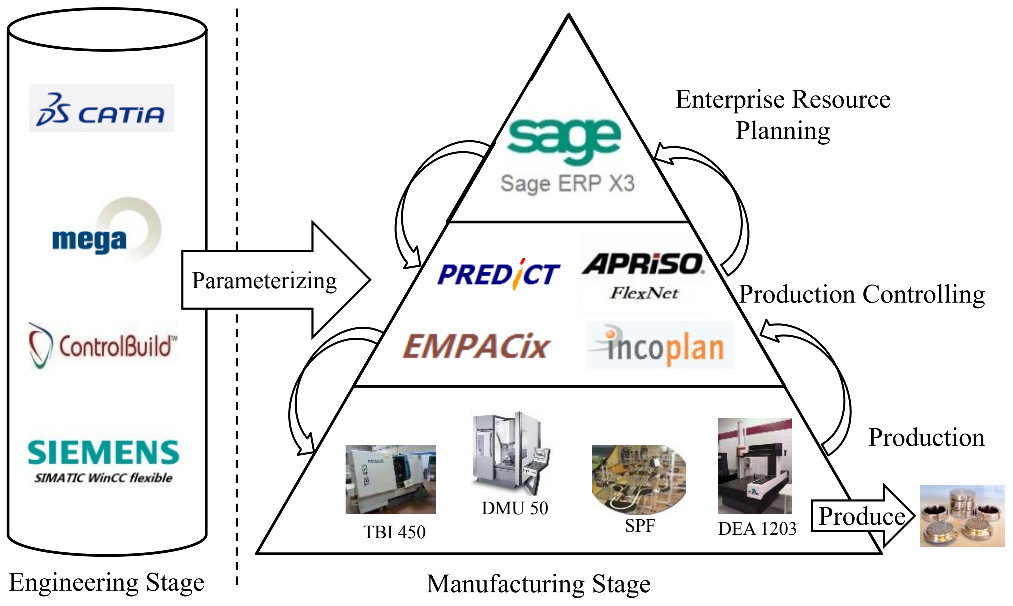


Figure 5-1 The System Stack in the AIPL

Among the diverse products that the AIPL produces we selected the educational combination product [124] for demonstrating how semantic annotations can be applied to improve the engineering phase and to facilitate the parameterization of the application software. Before presenting the life cycle of this product, we would like to give a clear vision of the product itself. As shown in the Figure 5-2, this educational combination product is composed of six types of Prods (components to be assembled), which are designed to be assembled and disassembled easily.

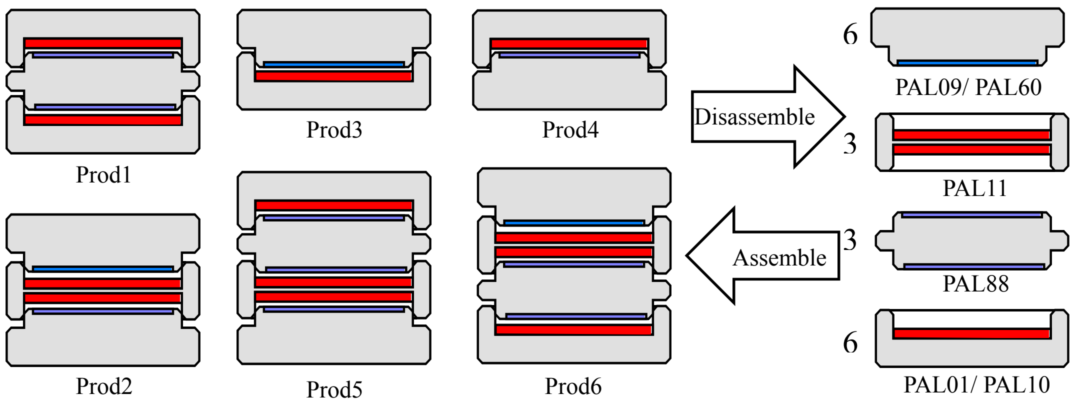


Figure 5-2 The Overview of the Educational Combination Product in the AIPL

The requirements of this product come from the needs of reusability of the educational materials. Some engineers at AIPL are experts in Mechanical Engineering and they conceived and designed the educational combination product using the

CATIA[20] Computer-Aided Design software (we name it as CATIA in the remaining chapter), which generates, among many other models, the product technical information into a so-called Engineering Bill of Material (EBOM). However, the information in the EBOM represents the product structure from the designer's point of view, which does not include all the information that is needed by the systems at the manufacturing stage to support the production. For this reason, a Bill of Process (BOP) needs to be combined together with EBOM. These processes are defined and modelled using the MEGA modelling environment (we name it as MEGA in the remaining chapter). The Figure 5-3 gives a brief overview of the manufacturing processes of this product:

- *Bases turning process*, which is in charge of chipping an aluminium bar into a number of designed bases.
- *Discs cutting process*, which is in charge of cutting galvanized plates and magnetic plates into a number of designed discs.
- *Parts sticking process*, which is in charge of using glues to stick the galvanized discs or the magnetic discs to the corresponding bases for producing different kinds of designed parts (the four kinds of components on the right hand side of the Figure 5-3).
- *Products assembling process*, which is in charge of assembling different kinds of parts into the designed Products (the six types of combinations on the left hand side of the Figure 5-3).

Most of these processes are designed and performed within the AIPL manufacturing workshops. However, because of the lack of a high force cutting equipment, the disc cutting process is outsourced to the IUT[21] (Institut Universitaire de Technologie Nancy-Brabois), a partner of the AIPL that owns the required equipment for this operation.

[20] CATIA http://www.3ds.com/products-services/catia/
[21] IUT http://www.iutnb.uhp-nancy.fr/

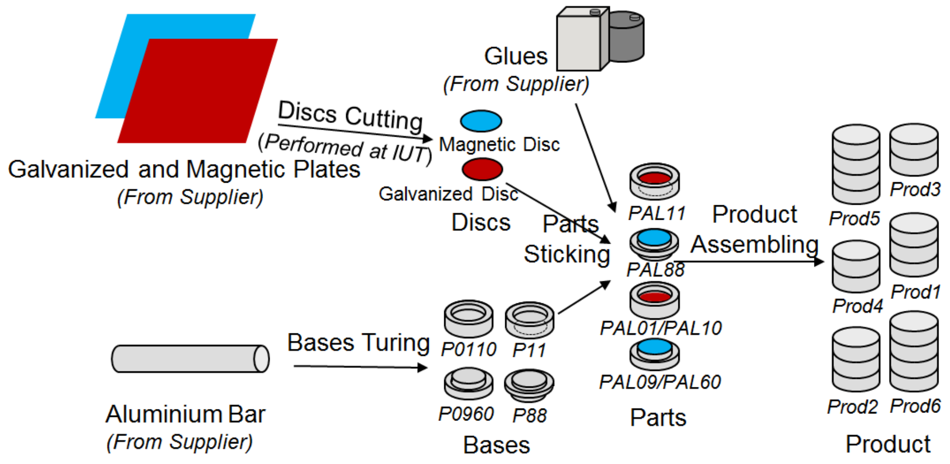


Figure 5-3 The Main Manufacturing Processes of the Educational Combination Product

Once the engineers, at the engineering stage, finished the design (for example, the product model and the process model), the EBOM and the BOP are used for the parameterization of the enterprise systems during the manufacturing stage.

For example, the Sage X3 ERP system[22] (we name it as Sage X3 in the remaining chapter), after being parameterized will take into account incoming customer orders for generating the different work orders for supporting the purchase of raw materials, the outsourcing of some processes and the manufacturing of the components and the related products:

- *For the purchasing part*, it will generate the purchasing orders to order raw materials (aluminium bars, galvanized plates, magnetic plates and glues) from different suppliers;
- *For the outsourcing part*, it will generate the outsourcing orders to send the galvanized plates to the IUT and to retrieve the components in the form of galvanized disc-shaped for the production;
- *For the manufacturing part*, it will generate the work orders to be compiled and scheduled by the Flexnet MES application[23], which is in charge of executing and controlling the production, and to retrieve the production states for updating the stocks information and various other performance indicators.

At the end, after some quality examination, all the qualified products are packed in boxes and dispatched to the production engineering teaching group. These educational combination products are used in an automatic assembly line, which allows students to participate to some practical lectures in production engineering and

[22] Sage X3 http://www.sage.com/
[23] Apriso Flexnet http://www.apriso.com/

programming.

For our case study, we selected three of these systems, two at the engineering stage and one in the manufacturing stage. We will apply our semantic annotation framework on these systems for (a) improving the knowledge of the engineers when they are designing the products with CATIA and MEGA and, (b) helping other engineers when they are parameterizing Sage X3.

5.1.2 The Application Scenario

During the engineering stage, the engineers need a lot of information exchange between each other for acquiring the exact semantics that is expressed in the received models. They spend a lot of time in reading the corresponding documents to guarantee the semantic consistency between the received models and the under development models. The precise semantics of the model elements, specified by the design engineers, is difficult to be directly passed or used by the following processes engineers through the exchanged information. Therefore, although all the products can be produced using the current mechanism, a great effort is required through the manual verification of the semantic correctness and semantic consistency of model elements.

We propose to use the formal semantic annotations to assist the expression, the storage and the sharing of the engineers' knowledge along all the product life cycle. In order to make explicit the implicit semantics in a TKR, according to the definition of semantic annotations given in Chapter 2, the knowledge needs to be firstly formalized and represented in OKRs. With the assistance of semantic annotations, the system engineers can:

- acquire the initial semantics that the stakeholders , who manipulate the upstream system, wanted to express;
- verify, semi-automatically, the semantic consistency between the contents in the received models and in the developing models;
- guarantee that the embedded semantics in the under development models is made explicit for the stakeholders, who manipulate the downstream system.

Of course, the information in the product life cycle is not just simply passed from one system to another in a linear unique direction. The information created by a system in a later stage might go back to a system in a former stage to improve the previous step.

Therefore, in order to determine a clear-cut information flow, to differentiate the systems that are used in this information flow, and to show the interoperation between those systems, we define three kinds of systems as follows:

- *Current system*, a system is in use in a specific point of the selected information flow.
- *Upstream system*, according to the information flow, it is the system that is placed before the current system.
- *Downstream system*, according to the selected information flow, it is the system that is placed after the current system

We choose MEGA as the current system, together with its upstream system (CATIA) and downstream system (Sage) as the application scenario. As shown in the Figure 5-4, the process model at the bottom shows all the different processes that are implied in the production of the educational combination product. There exist one or more systems that participate in each of these processes. CATIA is used for the "product design" process and MEGA is used for the "process design" process. They represent the systems in the engineering stage. Sage X3 is used for the "production plan" process, which represents one of the systems in the manufacturing stage. These three systems represent the TKR creation and management module in the Semantic Annotation Framework. For the other part of the framework, as stated in Section 4.1.2, the Protégé is used as the OKR Creation and Management module, the Microsoft windows folder system capability is used as the Knowledge Cloud module, the SAP-KM is the Semantic Annotation and Processing Agency, and the Jena Reasoner is used as the Reasoning Engine Module.

Besides the duty of supporting the definition and inference of the semantic annotations, the SAP-KM also serves as a mediator to assist the interoperation between the different modules. In the current version of the SAP-KM, there are two developed interfaces. One is between MEGA and the SAP-KM, and the other one is between the Jena Reasoner and the SAP-KM. The SAP-KM can also communicate with the Knowledge Cloud module to perform some operations (such as importing, querying, modifying and so on) on the OKRs, on the rules and on the semantic annotations. Because this research does not focus on the creation of the OKRs, the interface between the SAP-KM and the Protégé is not considered as part of the implementation. We used Protégé directly to manipulate the OKRs. Furthermore, the current implementation of SAP-KM already shows the possibility to handle the requests and feedbacks between itself and the TKR Creation and Management Module (MEGA in this case study). Several research literatures [115], [116], [113] and [68] show the possibility of developing a semantic annotation plug-in for the product model and the data model. In order to avoid the unnecessary repetition with those works, the interface between

CATIA/Sage X3 and the SAP-KM is not developed. In the case study, we assume that the corresponding plug-ins are already developed, following the solution we proposed, for CATIA and Sage.

In this application scenario, the information flow starts from the upstream system (CATIA) and ends at the downstream system (Sage X3). As we stated in Section 4.1.1, MEGA is in charge of creating process models, which formalized the BOP for the production. The upstream system, CATIA, provides the EBOM to assist the creations of the BOP. The downstream system, Sage X3, takes the EBOM and BOP to assist its parameterization.

The remainder of this chapter is divided into three phases, according to the semantic annotation procedure that we stated in Section 3.3.1: namely the Preparation Phase, the Annotation Phase and the Reasoning Phase and these phases are successively presented hereafter.

Application Scenario:

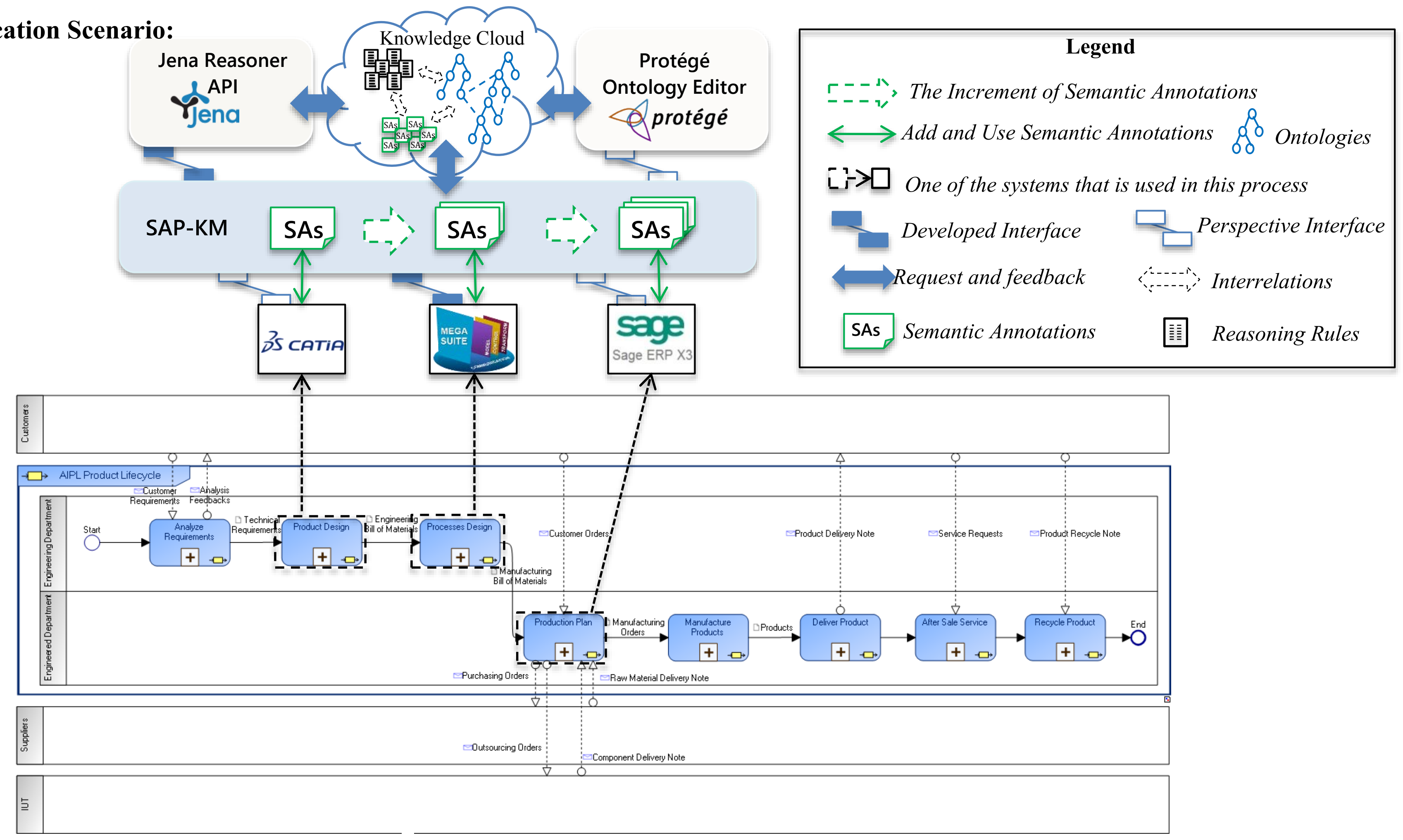


Figure 5-4 The Application Scenario of the Case Study

5.2 Preparation, Annotation and Reasoning Phases

In this section, we will show the activities in the case study following the three main phases described in the semantic annotation procedure:

(1) the preparation phase: we present the preparation of the TKRs and the OKRs which are needed in the two following phases;

(2) the annotation phase: the loading of the semantic annotations from the upstream system and the creation of the semantic annotations for the downstream system;

(3) the reasoning phase: the detection of inconsistencies between semantic annotations and the identification of possible model content mistakes according to the existing semantic annotations and reasoning rules.

5.2.1 The Preparation Phases

Concerning the TKRs, we consider two models for the application scenario: the product model created by CATIA (the combination of four kinds of metallic bases is shown in the Figure 5-5) and the process model created by MEGA (a segment of the manufacturing process of the component Prod3 is shown in the Figure 5-6).

As we stated in the previous section, MEGA is considered as the current system, and the process model created by this system is treated as the model that needs semantic enrichment. The product model from CATIA is considered as the model from upstream system, which has already been annotated. Sage X3 does not participate in the preparation and annotation phases. It will use the data that are contained in the semantic annotations.

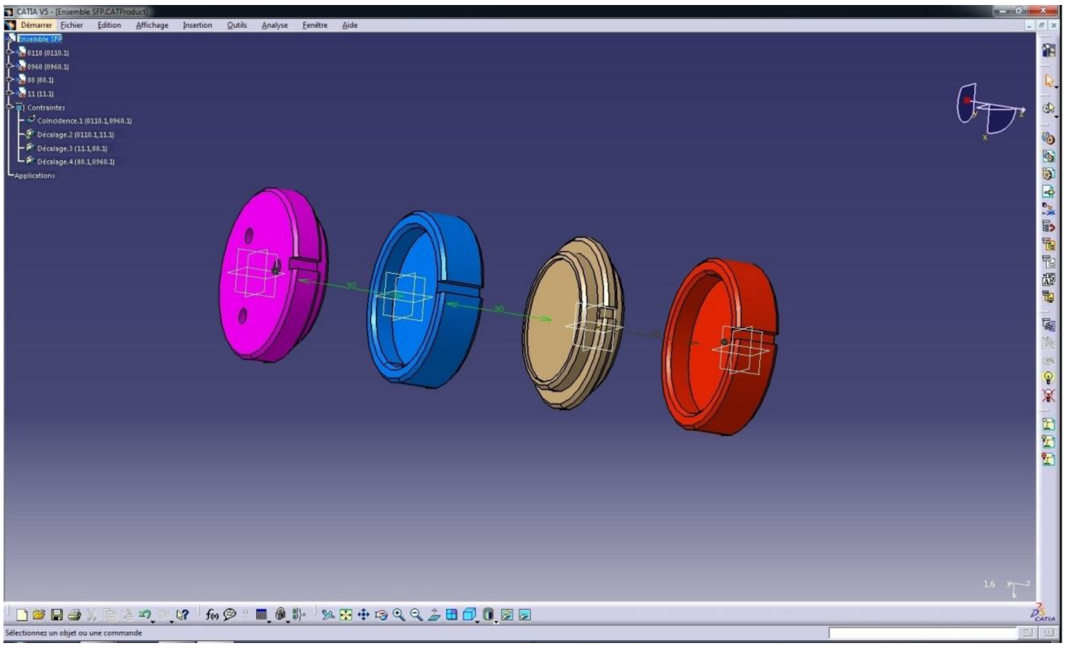


Figure 5-5 The four kinds of Bases that contain in the Product Model

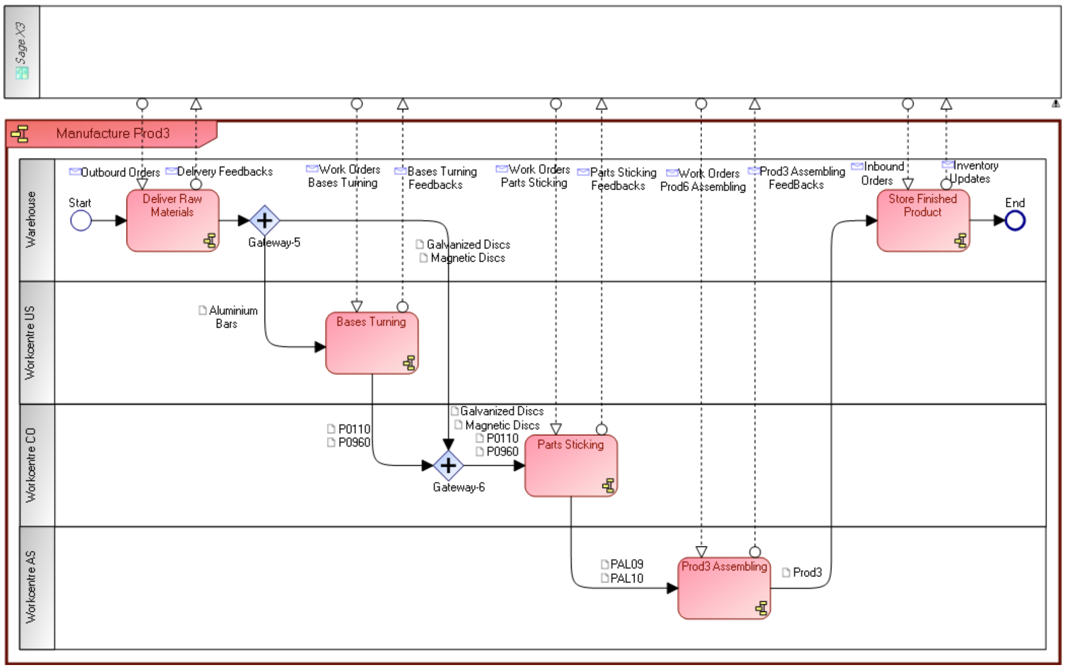


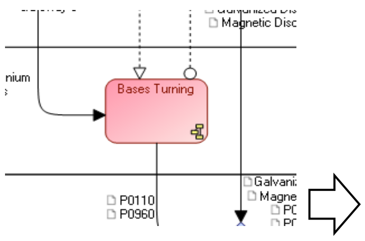
Figure 5-6 The Process Model from the MEGA

To be more specific, the process model in the Figure 5-6 contains five main participants:

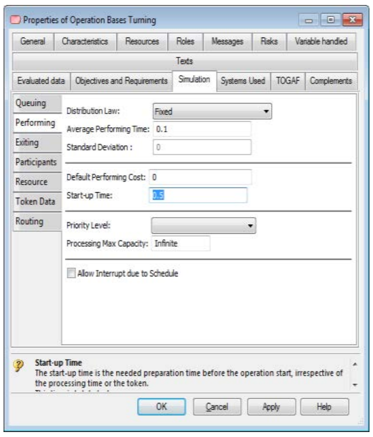
- the application participant, Sage X3, which produces different kinds of orders for other participants and collects the corresponding feedbacks;
- the warehouse, which is in charge of delivering raw materials to the work centre US (the Aluminium Bars) and to the work centre CO (the Galvanized Discs and the Magnetic Discs). It also stores the finished component (Prod3);

- the work centre US, which is in charge of the bases turning operation. It takes the aluminium bars as inputs and produces two kinds of bases (the P0110 and the P0960);
- the work centre CO, which is in charge of the parts sticking operation. It takes the outputs (the P0110 and the P0960) of the previous operation and the raw materials (the Galvanized Discs and the Magnetic Discs) from the warehouse to produce two kinds of parts (the PAL01 and the PAL60);
- the work centre AS, which is in charge of prod assembling operation. It takes the outputs (the PAL01 and the PAL 60) from the sticking operation to produce the component (Prod3). At the end, this component is sent to the warehouse.

The corresponding data is stored within the simulation properties of these operations in MEGA. As shown in the Figure 5-7, the start-up time and the performing time of the “bases turning” operation are stored in the properties of the referring model objects. That information can be further used to support the operation simulations.



The Bases Turning Process
Start-up Time: *0.5* hours
Performing Time: *0.1* hours


Properties of Operation Bases Turning

Distribution Law: Fixed
Average Performing Time: 0.1
Standard Deviation : 0
Default Performing Cost: 0
Start-up Time: 0.5
Priority Level:
Processing Max Capacity: Infinite
☐ Allow Interrupt due to Schedule

Start-up Time
The start-up time is the needed preparation time before the operation start, irrespective of the processing time or the token.

Figure 5-7 The Properties of an Operation in the MEGA

Concerning the OKR part, two domain level ontologies (the MSDL ontology [90] and the BPMN ontology [98]) are employed. Although the focus of the research is not the development of ontologies, we also created one top ontology (the general ontology) and two application ontologies (the AIPL product ontology and the MEGA BPMN ontology) to construct the three-level structures in the knowledge cloud. We generated only some parts of these three related ontologies, which contain the necessary concepts and relationships that can be used for the annotations in the case study.

The general ontology contains the definitions and restrictions of several general concepts at the top level, which is used as super classes of both PLC-related ontologies and the meta-model ontologies. The MSDL Ontology is a domain level PLC-related ontology that describes the manufacturing capability. The AIPL product ontology is an application level PLC-related ontology, which formalizes the specific product knowledge in the AIPL. The BPMN Ontology is a domain level meta-model ontology, which represents the structure components and their relationships based on the OMG's business process modelling notation specification. The MEGA BPMN ontology is an application level meta-model ontology that extends the standardised BPMN ontology with some specific concepts and relationships. As shown in the Figure 5-9, the contents in black colour are the extracted parts of these five ontologies. These five ontologies have their own namespaces, which are different from each other. To ease the reading, the namespaces are omitted in the figure.

A number of pre-processes is carried out on these five ontologies. As shown in the Figure 5-9, the contents in green colour show some results of the pre-processes. To be more specific, these pre-processes are used to:

(1) Add additional relationships: a set of additional relationships is added between the concepts in different ontologies. For example, the Object Property "hasShape" is added from the Individual "P0110" to the Individual "Cylinder".
(2) Complete the top-level hierarchy: a set of "subClassOf" is added from the top-level classes to the Class "Thing" (The Protégé is still able to show the top-level classes as the subclass of Thing without this pre-process action). This action is used to support the ontology loading in the Jena Reasoner.
(3) Enrich the semantics of existing ontologies: two aspects of the semantics are formalized and added into the PLC-related ontology (in both domain level and application level): (1) the semantics of a concept that is embedded in a general context; (2) the semantics of a concept that is embedded in a specific context.
(4) Store the ontologies: these five ontologies are stored in RDF/XML format to facilitate the ontology loading in the Jena Reasoner.

In the enriching step (3), the general concept definitions are selected from the WordNet[24] service. As shown in the Figure 5-8, (a) and (b) show the general definitions of the concept "Turning" and "Sticking". The context specific semantics of concepts are acquired from the special environment in the AIPL. As shown in the Figure 5-8, (c) and (d) show specific semantics that is embedded for the concept "BasesTurning" and the "Sticking".

[24] WordNet http://wordnet.princeton.edu/

turning: the ***activity*** of ***shaping something*** on a ***lathe***
lathe: the machine tool for shaping ***metal or wood***
shaping: the act of fabricating ***something*** in ***a particular shape***

(a)

sticking: fasten with ***an adhesive material*** like ***glue***
adhesive material : a substance that unites or bonds ***surfaces*** together
substance: ***material*** of a particular kind or constitution
surface: the outer boundary of ***an artifact***

(b)

The **bases turning** operation is performed on the lathe ***TBI-450***, which has a ***input length limited*** (max ***1 meter***).

(c)

After the **sticking** operation, a period of **drying** operation is necessary.

(d)

Figure 5-8 The Semantics of Concepts that embedded in their General/Specific Context

The Table 5-1 shows how the semantics of the concept "Turning" and the concept "Sticking" in a general context are represented in the MSDL ontology and the General ontology, based on their definitions (the (a) and (b) in the Figure 5-8).

Table 5-1 The Formal Representation of General Semantics of Concepts (in the MSDL Ontology and the General Ontology)

Concept	Relationship	Definition
Turning	*subClassOf* Activity	the *activity* of shaping something on a lathe[25]
	hasInput some Tinputs	the activity of shaping *something* on a lathe
	hasOutpt some Toutputs	the act of fabricating *something* in a particular shape
	isPerformedOn some Lathe	the activity of shaping something on a *lathe*
Lathe	*subClassOf* MachineTool	the machine tool for shaping metal or wood
Tinputs	*hasMaterial* only (Metal or Wood)	the machine tool for shaping metal or wood
Toutputs	*hasShape* some Shape	the act of fabricating something in *a particular shape*
Sticking	*hasInput* some AdhesiveMaterial	fasten with *an adhesive material* like glue
	hasInput some Artifact	a substance that unites or bonds *surfaces* together surface: the outer boundary of an *artefact*
AdhesiveMaterial	*subClassOf* Material	a substance that unites or bonds surfaces together substance: *material* of a particular kind or constitution
Glues	*subClassOf* AdhesiveMaterial	fasten with an adhesive material like *glue*
Artifact	*hasSurface* some Surface	surface: the outer boundary of an *artefact*

[25] Lathe is a machine for shaping wood, metal, or other material by means of a rotating drive which turns the piece being worked on against changeable cutting tools.

The Table 5-2 shows how the semantics of the concept "BasesTurning" and the concept "Sticking" in a specific context are represented in the AIPL Product ontology , based on their definitions (the (c) and the (d) in the Figure 5-8).

Table 5-2 The Formal Representation of Specific Semantics of Concepts in the AIPL Product Ontology

Concept	Relationship	Definion
BasesTurning	*isPerformedOn* TBI-450	The *BasesTurning is performed on* the lathe TBI-450,
	hasInput some Tinputs	has a *input length limited* (*max 1 meter*).
Tinputs	*hasMaxLength* value T01MaxLength	
T01MaxLength	*type* Artifact	
	meters 1	
TBI-450	*IsIndividualOf Lathe*	The BasesTurning is performed on the *lathe* TBI-450
Sticking	*hasNextProcess* some Drying	After the *sticking* operation, a period of *drying* operation is necessary.

The Figure 5-9 shows a fragment of the five ontologies in the knowledge cloud together with the pre-processing results. The figure contents in black colour are the initial ontology elements and the figure contents in the green colour are the results of pre-processes. Most of the ontology elements that are shown in this figure are used in the follow two phases to support the annotation and the reasoning.

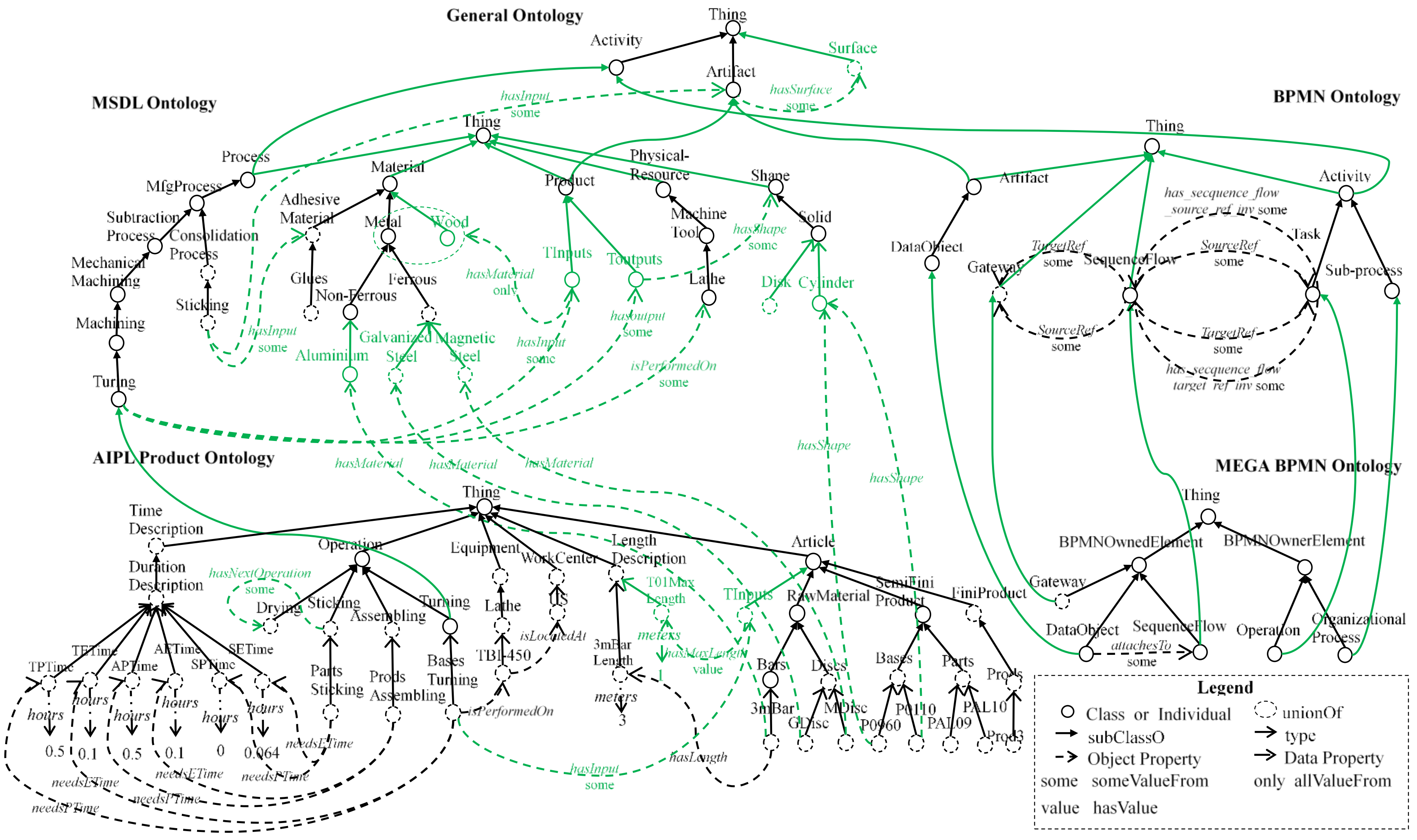


Figure 5-9 A Part of the five Ontologies in the Knowledge Cloud together with the pre-processing Results

These ontologies are stored in the Knowledge Cloud and imported into the Semantic Annotation Schema. In this way the knowledge contained in the semantic annotation can be shared with all the systems along the product life cycle. The above-specified manipulation of ontologies is performed through the Protégé OWL Editor.

The customized Semantic Annotation Schema that is used in this case study has been already presented in Chapter 4. The presented solution only shows one of the multiple possibilities to implement the formal semantic annotations. In the next section, we will present the use of the received semantic annotations from the upstream system and the creation of the semantic annotations in the current system and for the downstream system.

5.2.2 The Annotation Phases with the SAP-KM

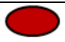
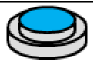
The semantic annotations that participate in the case study are divided into two parts: (1) the received semantic annotations from the upstream system, which are imported in the current system; (2) the created semantic annotations in the current system, which will be used by the downstream system.

Because of the limited time and resource, we did not develop the plug-ins in CATIA and Sage X3. We assume the existence of the corresponding plug-ins that follow the proposed solution and can store the annotation results into the Semantic Annotation Schema. In order to avoid the massive semantic annotation details for each annotation (such as the semantic annotation example that is shown in the Figure 4-5) and also to ease the explanation and reading, we omit the details of data structure. We represent the semantic annotations in the syntax of the "namespace; ontology element".

The "namespace" represents the namespace of the ontology element. The abbreviation namespace for the General Ontology, MSDL Ontology, BPMN Ontology, AIPL Product Ontology, MEGA BPMN Ontology and the Semantic Annotation Schema are respectively &GO, &MSDL, &BPMN, &AIPL, &MEGA and &SANS. The "ontology element" can be a class, an individual or a property.

Those semantic annotations from the upstream system are imported by the SAP-KM to assist the model creation and the inconsistency detection in MEGA. The Table 5-3 shows a list of model elements from the product model with their corresponding domain semantics. The structure semantics of a model is highly specific for the corresponding model. In our work, it is mainly used to describe the internal relationships between the model elements. However, the domain semantics of a model element is possible to be inherited by other model elements.

Table 5-3 The Domain Semantics of the Annotated Elements from the Upstream System

Model Elements	Domain Semantics	SR
e_1= 'bar'	p_1= &AIPL;**3mBar** &AIPL;hasLength &AIPL;3mBarLength &AIPL;**3mBar** rdfs:subClassOf &AIPL;Bars &AIPL;**3mBar** &AIPL;hasMaterial &MSDL;Aluminium &AIPL;3mBarLength &AIPL; meters 3 &AIPL;Bars rdfs:subClassOf &AIPL;RawMaterial	e_1 $\boldsymbol{sr}_{\subset}$ p_1
e_2= '0110'	p_2= &AIPL;**P0110** &AIPL;hasShape &MSDL;Cylinder &AIPL;**P0110** rdfs:subClassOf &AIPL;Bases &AIPL;Bases rdfs:subClassOf &AIPL;SemiFiniProduct	e_2 $\boldsymbol{sr}_{\subset}$ p_2
e_3= '0960'	p_3= &AIPL;**P0960** &AIPL;hasShape &MSDL;Cylinder &AIPL;**P0960** rdfs:subClassOf &AIPL;Bases &AIPL;Bases rdfs:subClassOf &AIPL;SemiFiniProduct	e_3 $\boldsymbol{sr}_{\subset}$ p_3
e_4= 'RA'	p_4= &AIPL;**MDisc** &AIPL;hasShape &MSDL;Disk &AIPL;**MDisc** rdfs:subClassOf &AIPL;Discs &AIPL;**MDisc** &AIPL;hasMaterial &MSDL; MagneticSteel &AIPL;Discs rdfs:subClassOf &AIPL;RawMaterial	e_4 $\boldsymbol{sr}_{\subset}$ p_4
 e_5= 'RG'	p_5= &AIPL;**GDsic** &AIPL;hasShape &MSDL;Disk &AIPL;**GDsic** rdfs:subClassOf &AIPL;Discs &AIPL;**GDsic** &AIPL;hasMaterial &MSDL;GalvanizedSteel &AIPL;Discs rdfs:subClassOf &AIPL;RawMaterial	e_5 $\boldsymbol{sr}_{\subset}$ p_5
 e_6= 'Part09'	p_6= &AIPL;**PAL09** &AIPL;hasDiscSide &AIPL;Downward &AIPL;**PAL09** rdfs:subClassOf &AIPL;Parts &AIPL;Parts rdfs:subClassOf &AIPL;SemiFiniProduct	e_6 $\boldsymbol{sr}_{\subset}$ p_6
e_7= 'Part10'	p_7= &AIPL;**PAL10** &AIPL;hasDiscSide &AIPL;Upward &AIPL;**PAL10** rdfs:subClassOf &AIPL;Parts &AIPL;Parts rdfs:subClassOf &AIPL;SemiFiniProduct	e_7 $\boldsymbol{sr}_{\subset}$ p_7
e_8= 'Prod3'	p_8= &AIPL;**Prod3** rdfs:subClassOf &AIPL;Prods &AIPL;Prods rdfs:subClassOf &AIPL;FiniProduct	e_8 $\boldsymbol{sr}_{\subset}$ p_8

In the current system, as shown in the Figure 5-10, a number of model elements of the process model is selected. In order to demonstrate the applicability of our proposal, we only chose three main operations in the manufacturing of Prod3 (the "Bases Turning" operation, the "Parts Sticking" operation and the "Prods Assembling" operation) together with their inputs and outputs as the candidates for semantic enrichment.

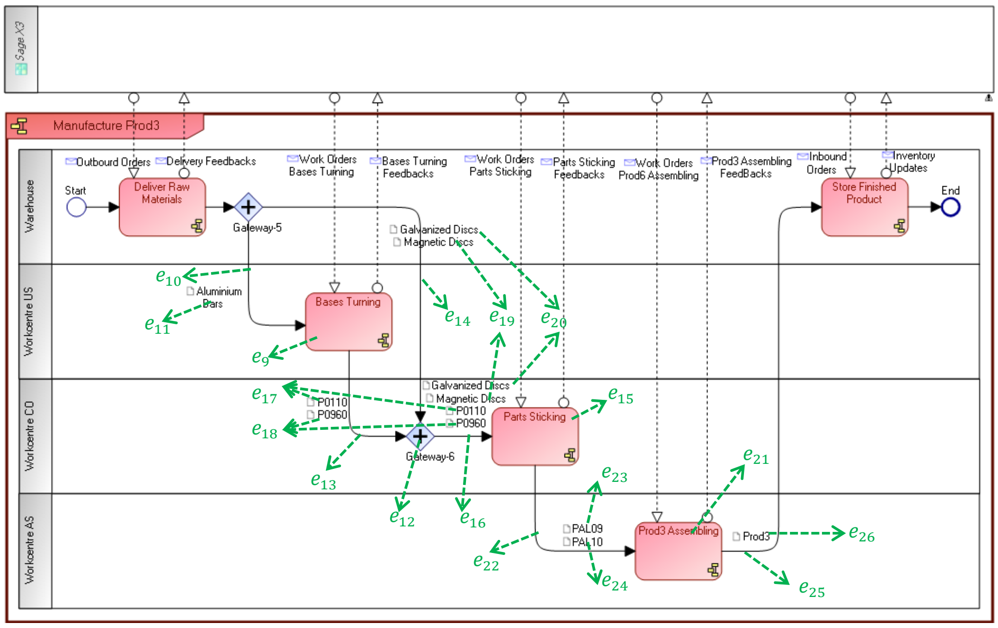


Figure 5-10 The Selected Model Elements in the Process Model

For the explicitation of the structure semantics part (represented by the meta-model ontology), the SAP-KM provides a graphical user interface under the tabbed pane of "Structure Semantics" (the Figure 4-10). The internal relationships between the model elements are made explicit through the use of the BPMN Ontology and the MEGA BPMN Ontology. The Table 5-4 shows the structure semantics of the annotated elements in the process model.

Table 5-4 The Structure Semantics of the Annotated Model Elements in the Process Model

Model Elements	Structure Semantics	MR
e_{9}= 'Bases Turning'	mme_{9} = &MEGA;Operation e_{9} &BPMN;has_secquence_flow_target_ref_inv e_{10} e_{9} &BPMN;has_secquence_flow_source_ref_inv e_{13}	e_{9} $mr_{io}mme_{9}$
e_{10}= 'Gateway-5-> Bases Turning'	mme_{10} = &MEGA;SequenceFlow e_{10} &BPMN; TargetRef e_{9}	e_{10} $mr_{io}mme_{10}$
e_{11}= 'Aluminium Bars'	mme_{11} = &MEGA;DataObject e_{11} &MEGA;attachesTo e_{10}	e_{11} $mr_{io}mme_{11}$
e_{12}= 'Gateway-6'	mme_{12} = &MEGA;Gateway e_{12} &BPMN;has_secquence_flow_target_ref_inv e_{13} e_{12} &BPMN;has_secquence_flow_source_ref_inv e_{16}	e_{12} $mr_{io}mme_{12}$
e_{13}= 'Bases Turning -> Gateway-6'	mme_{13} = &MEGA; SequenceFlow e_{13} &BPMN;TargetRef e_{12} e_{13} &BPMN;SourceRef e_{9}	e_{13} $mr_{io}mme_{13}$
e_{14}= 'Gateway-5-> Gateway-6'	mme_{14} = &MEGA;SequenceFlow e_{14} &BPMN; TargetRef e_{12}	e_{14} $mr_{io}mme_{14}$
e_{15}= 'Parts Sticking'	mme_{15} = &MEGA;Operation e_{15} &BPMN;has_secquence_flow_target_ref_inv e_{16} e_{15} &BPMN;has_secquence_flow_source_ref_inv e_{22}	e_{15} $mr_{io}mme_{15}$
e_{16}= 'Gateway-6 -> Parts Sticking'	mme_{16} = &MEGA;SequenceFlow e_{16} &BPMN;TargetRef e_{15}	e_{16} $mr_{io}mme_{16}$

	e_{16} &BPMN;SourceRef e_{12}	
e_{17}= ‘P0110’	mme_{17} = &MEGA;DataObject e_{17} &MEGA;attachesTo e_{13} e_{17} &MEGA;attachesTo e_{16}	$\mathrm{e}_{17}\ \mathrm{mr}_{\mathrm{io}}\mathrm{mme}_{17}$
e_{18}= ‘P0960’	mme_{18} = &MEGA;DataObject e_{18} &MEGA;attachesTo e_{13} e_{18} &MEGA;attachesTo e_{16}	$\mathrm{e}_{18}\ \mathrm{mr}_{\mathrm{io}}\mathrm{mme}_{18}$
e_{19}= ‘Galvanized Discs’	mme_{19} = &MEGA;DataObject e_{19} &MEGA;attachesTo e_{14} e_{19} &MEGA;attachesTo e_{16}	$\mathrm{e}_{19}\ \mathrm{mr}_{\mathrm{io}}\mathrm{mme}_{19}$
e_{20}= ‘Magnetic Discs’	mme_{20} = &MEGA;DataObject e_{20} &MEGA;attachesTo e_{14} e_{20} &MEGA;attachesTo e_{16}	$\mathrm{e}_{20}\ \mathrm{mr}_{\mathrm{io}}\mathrm{mme}_{20}$
e_{21}= ‘Prods Assembling’	mme_{21} = &MEGA;Operation e_{21} &BPMN;has_secquence_flow_target_ref_inv e_{22} e_{21} &BPMN;has_secquence_flow_source_ref_inv e_{25}	$\mathrm{e}_{21}\ \mathrm{mr}_{\mathrm{io}}\mathrm{mme}_{21}$
e_{22}= ‘Parts Sticking -> Prods Assembling’	mme_{22} = &MEGA;SequenceFlow e_{22} &BPMN; TargetRef e_{21} e_{22} &BPMN; SourceRef e_{15}	$\mathrm{e}_{22}\ \mathrm{mr}_{\mathrm{io}}\mathrm{mme}_{22}$
e_{23}= ‘PAL09’	mme_{23} = &MEGA;DataObject e_{23} &MEGA;attachesTo e_{22}	$\mathrm{e}_{23}\ \mathrm{mr}_{\mathrm{io}}\mathrm{mme}_{23}$
e_{24}= ‘PAL10’	mme_{23} = &MEGA;DataObject e_{23} &MEGA;attachesTo e_{22}	$\mathrm{e}_{24}\ \mathrm{mr}_{\mathrm{io}}\mathrm{mme}_{24}$
e_{25}= ‘Prods Assembling -> Store Finished Product’	mme_{25} = &MEGA;SequenceFlow e_{25} &BPMN;SourceRef e_{21}	$\mathrm{e}_{25}\ \mathrm{mr}_{\mathrm{io}}\mathrm{mme}_{25}$
e_{26}= ‘Prod3’	mme_{26} = &MEGA;DataObject e_{26} &MEGA;attachesTo e_{25}	$\mathrm{e}_{26}\ \mathrm{mr}_{\mathrm{io}}\mathrm{mme}_{26}$

For the explicitation of the domain semantics part, the SAP-KM provides two possibilities: (1) to reuse the domain semantics in the imported semantic annotations through its Elements Matching function (Section 4.3.3) and (2) to create new domain semantics for the selected model elements (Section 4.3.1.2).

In order to reuse the semantic annotations from the upstream system, the matching between the annotated elements in the former model and the elements current model need to be implemented. The SAP-KM provides a graphical user interface (the Figure 4-17) to support this matching process. Table 5-5 shows the matching results. In the current version of SAP-KM, it is only able to deal with the one to one matching.

Table 5-5 The Elements Matching between Product Model and Process Model

Model elements (Product Model)	Model elements (Process Model)
e_1= ‘bar’	e_{11}= ‘Aluminum Bars’
e_2= ‘0110’	e_{17}= ‘P0110’
e_3= ‘0960’	e_{18}= ‘P0960’
e_4= ‘RA’	e_{19}= ‘Galvanized Discs’
e_5= ‘RG’	e_{20}= ‘Magnetic Discs’

e_6= ‘Part09’	e_{23}= ‘PAL09’
e_7= ‘Part10’	e_{24}= ‘PAL10’
e_8= ‘Prod3’	e_{26}= ‘Prod3’

After the matching process, the matched concepts in the product model have their domain semantics related to their corresponding matched elements in the process model.

Besides reusing semantic annotations, the SAP-KM also supports the creation of new semantic annotations by providing a graphical user interface under the tabbed pane of “Domain Semantics” (the Figure 4-11). The domain semantics of the annotated model elements are made explicit through using the General Ontology, the MSDL Ontology and the AIPL Product Ontology. The Table 5-6 shows the domain semantics of the annotated elements in the process model.

Table 5-6 The Domain Semantics of the Annotated Elements in the Process Model

Model Elements	Domain Semantics	SR
e_9= ‘Bases Turning’	p_9= &MSDL;**Turning** &MSDL;isPerformedOn some &MSDL;Lathe &MSDL;**Turning** &MSDL;hasInput some &MSDL;TInputs &MSDL;**Turning** &MSDL;hasOutput some &MSDL;TOutputs &MSDL;**TInputs** &MSDL:hasMaterial only unionOf (&MSDL;Wood or &MSDL;Metal) &MSDL;TOutputs &MSDL:hasShape some &MSDL;Shape	$e_9\ \boldsymbol{sr}_{\subset}\ p_9$
e_9= ‘Bases Turning’	p_{10}= &AIPL;**BasesTurning** &AIPL;isPerformedOn &AIPL;TBI-450 &AIPL;**BasesTurning** &AIPL;hasInput some &AIPL;TInputs &AIPL;**BasesTurning** &AIPL;needsPTime &AIPL;TPTime &AIPL;**BasesTurning** &AIPL;needsETime &AIPL;TETime &AIPL;TInputs &AIPL;hasMaxLength value &AIPL;T01MaxLength &AIPL; TBI-450 &AIPL;isLocatedAt &AIPL;US &AIPL; T01MaxLength &AIPL;meters 1 &AIPL; TETtime &AIPL. hours 0.5 &AIPL; TPTtime &AIPL. hours 0.1	$e_9\ sr_{\subset}\ p_{10}$
e_{11}= ‘Aluminium Bars’	p_1= &AIPL;**3mBar** &AIPL;hasLength &AIPL;3mBarLength &AIPL;**3mBar** rdfs:subClassOf &AIPL;Bars &AIPL;**3mBar** &AIPL;hasMaterial &MSDL;Aluminum &AIPL;3mBarLength &AIPL; meters 3 &AIPL;Bars rdfs:subClassOf &AIPL;RawMaterial	$e_{11}\ sr_{\subset}\ p_1$
e_{15}= ‘Parts Sticking’	p_{11}= &MSDL;**Sticking** &MSDL:hasInput some &MSDL;AdhesiveMaterial &MSDL; **Sticking** & MSDL:hasInput some &Artifact &GO; Artifact & GO:hasSurface some &GO;Surface	$e_{15}\ \boldsymbol{sr}_{\subset}\ p_{11}$
e_{15}= ‘Parts Sticking’	p_{12}= &AIPL; **PartsSticking** &AIPL;isPerformedOn &AIPL;CO &AIPL; **PartsSticking** rdf;type &AIPL;Sticking &AIPL;**PartsSticking** &AIPL;needsPTime &AIPL;SPTime &AIPL; **PartsSticking** &AIPL;needsETime &AIPL;SETime &AIPL;Sticking &AIPL;hasNextOperation some &AIPL;Drying &AIPL; SPTime &AIPL;hours 0 &AIPL; SETime &AIPL;hours 0.064	$e_{15}\ \boldsymbol{sr}_{\subset}\ p_{12}$
e_{17}= ‘P0110’	p_2= &AIPL;**P0110** &AIPL;hasShape &MSDL;Cylinder &AIPL;**P0110** rdfs:subClassOf &AIPL;Bases &AIPL;Bases rdfs:subClassOf &AIPL;SemiFiniProduct	$e_{17}\ sr_{\subset}\ p_2$
e_{18}= ‘P0960’	p_3= &AIPL;**P0960** &AIPL;hasShape &MSDL;Cylinder &AIPL;**P0960** rdfs:subClassOf &AIPL;Bases	$e_{18}\ sr_{\subset}\ p_3$

	&AIPL;Bases rdfs:subClassOf &AIPL;SemiFiniProduct	
e_{19}= 'Galvanized Discs'	p_4= &AIPL;**MDisc** &AIPL;hasShape &MSDL;Disk &AIPL;**MDisc** rdfs:subClassOf &AIPL;Discs &AIPL;**MDisc** &AIPL;hasMaterial &MSDL; GalvanizedSteel &AIPL;Discs rdfs:subClassOf &AIPL;RawMaterial	e_{19} $sr_{\subset}$ p_4
e_{20}= 'Magnetic Discs'	p_5= &AIPL;**GDsic** &AIPL;hasShape &MSDL;Disk &AIPL;**GDsic** rdfs:subClassOf &AIPL;Discs &AIPL;**GDsic** &AIPL;hasMaterial &MSDL; MagneticSteel &AIPL;Discs rdfs:subClassOf &AIPL;RawMaterial	e_{20} $sr_{\subset}$ p_5
e_{21}= 'Prods Assembling'	p_{14}= &AIPL; **ProdsAssembling** &AIPL;isPerformedOn &AIPL;SPF &AIPL;**ProdsAssembling** &AIPL;needsPTime &AIPL;APTime &AIPL;**ProdsAssembling** &AIPL;needsETime &AIPL;AETime &AIPL; SPF &AIPL;isLocatedAt &AIPL;AS &AIPL; AETtime &AIPL;hours 0.5 &AIPL; APTtime &AIPL;hours 0.1	e_{21} $\boldsymbol{sr}_{\subset}$ p_1
e_{23}= 'PAL09'	p_6= &AIPL;**PAL09** &AIPL;hasDiscSide &AIPL;Downward &AIPL;**PAL09** rdfs:subClassOf &AIPL;Parts &AIPL;Parts rdfs:subClassOf &AIPL;SemiFiniProduct	e_{23} $sr_{\subset}$ p_6
e_{24}= 'PAL10'	p_8= &AIPL; **PAL10** &AIPL:hasDiscSide &AIPL; Upward &AIPL; **PAL10** rdfs;subclass some &AIPL; Parts &AIPL;Prods rdfs:subClassOf &AIPL;FiniProduct	e_{26} $\boldsymbol{sr}_{\subset}$ p_8
e_{26}= 'Prod3'	p_8= &AIPL;**Prod3** rdfs:subClassOf &AIPL;Prods &AIPL;Prods rdfs:subClassOf &AIPL;FiniProduct	e_{26} $\boldsymbol{sr}_{\subset}$ p_8

All the semantic annotations are stored in the Semantic Annotation Schema, which can be used as one of the basis for the reasoning phase.

5.2.3 The Reasoning Phases with the SAP-KM

In the case study, the reasoning phase is mainly in charge of (1) suggesting inferred semantic annotations for the corresponding model elements; (2) detecting some inconsistencies between several semantic annotations of an annotated model element; and (3) identifying the possible mistakes, namely conflicts, among annotated model elements.

The SAP-KM supports the semantic block delimitation and the semantic annotation suggestion providing a graphical user interface under the tabbed pane "Annotation Suggestion" (as shown in the Figure 4-12). Three semantic block delimitation rules are used in the case study to make explicit the internal relationships among the annotated elements in the process model. As shown in the Figure 5-11, the rule "Operation_to_DataObject" and the rule "DataObject_to_Operation" are used to create the semantic blocks that supersede the semantics between an operation and the data objects that are related to it (two possible situations). The former one adds a new relationship between an operation and the data objects that are attached to its outgoing sequence flows. The latter one adds a new relationship between an operation and the data objects that are attached to its incoming sequence flows. The rule

"Operation1_to_Operation2" is used to create the semantic blocks that substitutes the semantics between two operations, which are connected by a sequence flow. These three rules only show three possible situations between two appointed types of model elements. However they are enough for supporting the SBR delimitation in the case study.

```
@prefix SANS: http://www.semanticweb.org/ontologies/2013/6/SemanticAnnotations#
@prefix MEGA: http://www.semanticweb.org/ontologies/2013/6/MEGA_BPMN#
@prefix BPMN: http://dkm.fbk.eu/index.php/BPMN_Ontology#

[Operation_to_DataObject:   (?OP  rdf:type  MEGA:Operation)
                            (?DO  rdf:type  MEGA:DataObject)
                            (?SF  rdf:type  MEGA:SequenceFlow)
                            (?DO  MEGA:attachesTo  ?SF)
                            (?OP  BPMN:has_secquence_flow_source_ref_inv  ?SF)
                            ->
                            (?OP  SANS:SBR_Operation_to_DataObject  ?DO)
]
[DataObject_to_Operation:   (?OP  rdf:type  MEGA:Operation)
                            (?DO  rdf:type  MEGA:DataObject)
                            (?SF  rdf:type  MEGA:SequenceFlow)
                            (?DO  MEGA:attachesTo  ?SF)
                            (?OP  bpmn: has_secquence_flow_target_ref_inv ?SF)
                            ->
                            (?OP  SANS:SBR_DataObject_to_Operation ?DO)
]
[Operation1 to Operation2:   (?OP1  rdf:type  MEGA:Operation)
                             (?SF   rdf:type  MEGA:SequenceFlow)
                             (?OP2  rdf:type  MEGA: Operation)
                             (?SF   BPMN:SourceRef  ?OP1)
                             (?SF   BPMN:TargetRef  ?OP2)
                             ->
                             (?OP1  SANS:SBR_Operation1_to_ Operation2 ?OP2)
]
```

Figure 5-11 Three Rules to define a SBR for making explicit the Relations

Let us take the rule "DataObject_to_Operation" as an example. Based on this rule, the reasoner:

- first collects all the individuals of the class "&MEGA;Operation" and puts them in the variable ?OP (e_9, e_{15} and e_{21}),
- collects all the individuals of the class "&MEGA;DataObject" and puts them in the variable ?DO (e_{11}, e_{17}, e_{18}, e_{19}, e_{20}, e_{23}, e_{24} and e_{26}),
- collects all the individuals of the class "&MEGA;SequenceFlow" and puts them in the variable ?SF (e_{10}, e_{13}, e_{14} and e_{16}).
- Then, for each individual in the ?DO (let's take e_{11} as example) find the corresponding individual in the ?SF (e.g. e_{10}), where the individual of the class "&MEGA;DataObject" has the property "&MEGA;attachesTo" to the

individuals of the class “&MEGA;SequenceFlow”.

- Then based on the found sequence flow (e.g. e_{10}), find the corresponding individual in the ?OP (e.g. e_9), which has the property “&BPMN;has_has_secquence_flow_target_ref_inv” to the appointed sequence flow (e.g. e_{10}).
- At the end, the “&SANS;SBR_DataObject_to_Operation” property is added from the corresponding individuals in the ?OP (e.g. the e_9) and the corresponding individuals in the ?DO (e.g. the e_{11}).

After the semantic block delimitation, these three propeties are added between the corresponding ontology elements in the Semantic Annotation Schema, and then the property association process is then able to be performed. As shown in the Table 5-7, the first column shows properties that are made explicit based on the SBR delimitation rules and the meta-model ontologies. The second column shows the properties in the PLC-related ontologies.

Table 5-7 The Associations between two Properties

First Properties	Second Properties
&SANS;SBR_DataObject_to_Operation	&MSDL;hasInput
&SANS;SBR_DataObject_to_Operation	&AIPL;hasInput
&SANS;SBR_Operation_to_DataObject	&MSDL;hasOutput
&SANS;SBR_Operation_to_DataObject	&AIPL;hasOutput
&SANS;SBR_Operation1_to_Operation2	&AIPL;hasNextOperation

Based on the inferred semantic annotations suggestion algorithm (the Pseudo Code 4 in APPENDIX III), corresponding inferred semantic annotations are suggested. Table 5-8 shows all the inferred semantic annotations .

Let us take the association between the “&SANS;SBR_DataObject_to_Operation” and “&AIPL;hasInput” as an example. The SAP-KM:

- first queries all the individuals (from e_1 to e_{26}) in the Class “&SANS;E” and selects the individuals that has the property “&SANS;SBR_DataObject_to_Operation” (e_9, e_{15} and e_{21}).
- Then, for each of the selected individuals (let’s take e_9 as example), get the list of objects that the property “&SANS;SBR_DataObject_to_Operation” points to (e.g. e_{11}).
- Then, it verifies all the semantic annotations of the selected individual (e.g. e_9) and lists all its domain semantics (e.g. p_9 and p_{10}) that has the semantic relation “&SANS;SR_isEquivalentTo” or “&SANS;SR_isSubsumedBy” to it.
- Then, it queries the main concept of the collected domain semantics (e.g.

"&MSDL;Turning" (in p_9) and "&AIPL;BasesTurning" (in p_{10})). For each main concept, it queries if it has got a property "&AIPL;hasInput" and if it gets the object that the property points to (e.g. "&MSDL;TInputs" (in p_9) and "&AIPL;TInputs" (in p_{10})).

- Then it marks them ("&MSDL;TInputs" and "&AIPL;TInputs") as the main concepts of new domain semantics (e.g. p_{17} and p_{18} respectively). The concepts of the new domain semantics are created on the base of the paths that are related to its main concepts. The scope is limited by the original domain semantics (e.g. p_9 and p_{10}).
- At the end, the new domain semantics will be suggested to the selected individual (e.g. e_9) and it is then marked with the property "&SANS;SR_isSubsumedBy".

Table 5-8 The Domain Semantics of the Inferred Semantic Annotations

Model Elements	Domain Semantics	SR
e_{11}= 'Aluminium Bars'	p_{17}=&AIPL;**TInputs** &AIPL;hasMaxLength value &AIPL;T01MaxLength &AIPL;T01MaxLength &AIPL;meters 1 p_{17} &SANS;isInferredFrom e_9	e_{11} $\boldsymbol{sr}_{\subset}$ p_{17}
	p_{18}= &MSDL;**TInputs** &MSDL:hasMaterial only unionOf (&MSDL;Wood or &MSDL;Metal) p_{18} &SANS;isInferredFrom e_9	e_{11} $\boldsymbol{sr}_{\subset}$ p_{18}
e_{17}= 'P0110'	p_{19}= &MSDL;**TOutputs** &MSDL:hasShape some &MSDL;Shape p_{19} &SANS;isInferredFrom e_9	e_{17} $\boldsymbol{sr}_{\subset}$ p_{19}
	p_{21}= &GO; **Artifact** & GO:hasSurface some &GO;Surface p_{21} &SANS;isInferredFrom e_{15}	e_{17} $\boldsymbol{sr}_{\subset}$ p_{21}
	p_{22}= &MSDL; **AdhesiveMaterial** p_{22} &SANS;isInferredFrom e_{15}	e_{17} $\boldsymbol{sr}_{\subset}$ p_{22}
e_{18}= 'P0960'	p_{20}= &MSDL;**TOutputs** &MSDL:hasShape some &MSDL;Shape p_{20} &SANS;isInferredFrom e_9	e_{18} $\boldsymbol{sr}_{\subset}$ p_{20}
	p_{32}= &GO; **Artifact** &GO:hasSurface some &GO;Surface p_{23} &SANS;isInferredFrom e_{15}	e_{18} $\boldsymbol{sr}_{\subset}$ p_{23}
	p_{24}= &MSDL; **AdhesiveMaterial** p_{24} &SANS;isInferredFrom e_{15}	e_{18} $\boldsymbol{sr}_{\subset}$ p_{24}
e_{19}= 'Galvanized Discs'	p_{25}= &MSDL; **Artifact** & MSDL:hasSurface some &Surface p_{25} &SANS;isInferredFrom e_{15}	e_{19} $\boldsymbol{sr}_{\subset}$ p_{25}
	p_{26}= &MSDL; **AdhesiveMaterial** p_{26} &SANS;isInferredFrom e_{15}	e_{19} $\boldsymbol{sr}_{\subset}$ p_{26}
e_{20}= 'Magnetic Discs'	p_{27}= &MSDL; **Artifact** & MSDL:hasSurface some &Surface p_{27} &SANS;isInferredFrom e_{15}	e_{20} $\boldsymbol{sr}_{\subset}$ p_{27}
	p_{28}= &MSDL; **AdhesiveMaterial** p_{28} &SANS;isInferredFrom e_{15}	e_{20} $\boldsymbol{sr}_{\subset}$ p_{28}
e_{21}= 'Prods Assembling'	p_{29}= &MSDL; **Drying** p_{29} &SANS;isInferredFrom e_{15}	e_{21} $\boldsymbol{sr}_{\subset}$ p_{29}

After the suggestion of semantic annotations, the comparison of the similarity between two domain semantics of a same annotated model element can be performed.

SAP-KM provides the graphical user interface under the tabbed pane of “Semantic Similarity Comparison” (the Figure 4-14). It queries all the individuals that have both initial and inferred semantic annotations in the Class “&SANS;E” (the first column of Table 5-9) and it generates all the possible comparison pairs between the initial one to the inferred one (the second column of the Table 5-9). The semantic relationships between two domain semantics are defined in the third column of the Table 5-9. The similarity comparison results and the inconsistency detection rules are used as inputs of the reasoning engine to produce the inconsistency detection results. Finally, these results are listed in the last column of the Table 5-9.

Let us take the model element e_{11} in the Table 5-9 as an example. For the first pair (p_1 and p_{17}), $\mathrm{e}_{11}\ \mathrm{sr}_{\subset}\ \mathrm{p}_1$ and $e_{11}\ \boldsymbol{sr}_{\subset}\ p_{17}$ indicate the facts that e_{11} is supposed to inherit all the conditions that are described in p_1 and p_{17}.

- The domain semantics p_1 means that e_{11} is a kind of “&AIPL;3mBar”, which has the length 3 meter and is made from the material “&MSDL;aluminium”.
- The domain semantics p_{17} means that $\mathrm{e}_{11}$is a kind of “&AIPL;TInputs” that has a maximum length of one meter.

Because e_{11} is impossible to be an individual that fulfils the condition “has the length 3 meter” and the condition “has a maximum length of 1 meter” at the same time. Therefore, p_1 has no common semantics with p_{17}. The result is noted as $p_1\ \ pr_{\perp}\ \ p_{17}$.

For the second pair (p_1 and p_{18}), $e_{11}\ \boldsymbol{sr}_{\subset}\ p_1$ and $e_{11}\ \boldsymbol{sr}_{\subset}\ p_{18}$ indicate the facts that e_{11} is supposed to inherit all the conditions that are described in p_1 and p_{18}.

- The domain semantics p_1 means that e_{11} is a kind of “&AIPL;3mBar”, which has the length 3 meter and is made from the material “&MSDL;aluminium”.
- The domain semantics p_{18} means that e_{11} is a kind of “&MSDL;TInputs”, which is made from the material either a kind of “&MSDL;Wood” or a kind of “&MSDL;Metal”.

Because the “&MSDL;aluminium” is an individual of the Class “&MSDL;Non-Ferrous”, which is the subclass of the Class “&MSDL;Metal”. Therefore, the semantics of p_1 is less general than the semantics of p_{18}. The result is noted as $p_1\ \ pr_{\subset}\ \ p_{18}$.

Finally, according to the rules that are created based on the contents in the third column and third row of the possible inconsistency table (the Table 3-1), the property “&SANS;isNotConsistentWith” is added from p_1 to p_{17} and the property “&SANS; isPosConsistentWith” is added from p_1 to p_{18}.

Table 5-9 The Results of Inconsistency Detection between two Domain Semantics

Model Elements	Pairs	PR	Consistency Detection Results
e_{11}= 'Aluminium Bars'	1	$p_1\ pr_{\perp}\ p_{17}$	p_1 &SANS;isNotConsistentWith p_{17}
	2	$p_1\ pr_{\subset}\ p_{18}$	p_1 &SANS; isPosConsistentWith p_{18}
e_{17}= 'P0110'	1	$p_2\ pr_{\subset}\ p_{19}$	p_2 &SANS; isPosConsistentWith p_{19}
	2	$p_2\ pr_{\subset}\ p_{21}$	p_2 &SANS; isPosConsistentWith p_{21}
	3	$p_2\ pr_{\perp}\ p_{22}$	p_2 &SANS;isNotConsistentWith p_{22}
e_{18}= 'P0960'	1	$p_3\ pr_{\subset}\ p_{20}$	p_3 &SANS; isPosConsistentWith p_{20}
	2	$p_3\ pr_{\subset}\ p_{23}$	p_3 &SANS; isPosConsistentWith p_{23}
	3	$p_3\ pr_{\perp}p_{24}$	p_3 &SANS; isNotConsistentWith p_{24}
e_{19}= 'Galvanized Discs'	1	$p_4\ pr_{\subset}\ p_{25}$	p_4 &SANS; isPosConsistentWith p_{25}
	2	$p_4\ pr_{\perp}\ p_{26}$	p_4 &SANS;isNotConsistentWith p_{26}
e_{20}= 'Magnetic Discs'	1	$p_5\ pr_{\subset}\ p_{27}$	p_5 &SANS; isPosConsistentWith p_{27}
	2	$p_5\ pr_{\perp}\ p_{28}$	p_5 &SANS;isNotConsistentWith p_{28}
e_{21}= 'Prods Assembling'	1	$p_{13}\ pr_{\perp}\ p_{29}$	p_{13} &SANS;isNotConsistentWith p_{29}

The inconsistency detection results in the Table 5-9 are used to assist the identification of possible conflicts between annotated model elements during the modelling phase. Based on the model conflict identification rules and algorithms, the results are shown in the Table 5-10. Those results are used to draw attentions to modellers for examining the correctness of two annotated elements in the process model.

Table 5-10 The Possible Mistakes

No	The Possible Mistakes	The Inconsistency Detection Results
1	e_{11} &SANS;isPosConflictWith e_9	p_1 &SANS;isNotConsistentWith p_{17}
2	The domain semantics described in p_{22} , p_{24}, p_{26} and p_{28} ,which is required by e_{15}, is not being satisfied.	p_2 &SANS;isNotConsistentWith p_{22} p_3 &SANS;isNotConsistentWith p_{22} p_4 &SANS;isNotConsistentWith p_{22} p_5 &SANS;isNotConsistentWith p_{22}
3	e_{21} &SANS;isPosConflictWith e_{15}	p_{13} &SANS;isNotConsistentWith p_{29}

Ideally, the model content conflicts identification results are supposed to contain the reason why two model elements have conflicts and how to solve these possible mistakes. For example, these results could be used to provide suggestions as follows:

(1) The input "Aluminium Bars" is 3 meters, which is out of the range (≤1 meter) of the "bases turning" operation. You can change the input or change the operation.

(2) The domain semantics "&MSDL; AdhesiveMaterial" is not stratified by any current inputs of the "Part Sticking" operation. You can add a new input related

to this domain semantics.

(3) The "Prods Assembling" operation is not an individual of the Class "&MSDL;Drying". You can change the "Prods Assembling" operation or change the "Parts Sticking" operation.

However, these kinds of suggestions highly rely on the power of the reasoning engine. The current reasoning engines are only able to deal with some simple reasoning, such as classification (class subsumption and individual memberships) and class consistency (whether a class can have individuals or not), but they cannot deal with sub-ontologies. Therefore, we cannot use the current reasoning engine to provide these kinds of suggestions. Once there is a reasoning engine that is able to perform the reasoning on sub-ontologies, these kinds of suggestions might be implemented. The current version of the SAP-KM is only able to tell if two model elements have conflicts or if there is a domain semantics of a model element that is missing.

The process model and the created semantic annotations are sent to Sage X3 to assist the parameterization. Let us take the table of "process planning" in Sage X3 as an example. The "process", "WorkCentre", "preparation time" and "execution time" are the four of its main elements in the parameterization. Table 5-11 shows the data that are contained in the semantic annotations of e_9, e_{15} and e_{21}.

Table 5-11 The Model Elements in Process Model with Domain Semantics.

Model Elements	Domain Semantics
e_9= 'Bases Turning'	p_{10}= &AIPL;**BasesTurning** &AIPL;isPerformedOn &AIPL;TBI-450 &AIPL;**BasesTurning** &AIPL;hasInput some &AIPL;TInputs &AIPL;**BasesTurning** &AIPL;needsPTime &AIPL;TPTime &AIPL;**BasesTurning** &AIPL;needsETime &AIPL;TETime &AIPL;TInputs &AIPL;hasMaxLength value &AIPL;T01MaxLength &AIPL; TBI-450 &AIPL;isLocatedAt &AIPL;US &AIPL; T01MaxLength &AIPL;meters 1 &AIPL; TETime &AIPL. hours 0.5 &AIPL; TPTime &AIPL. hours 0.1
e_{15}= 'Parts Sticking'	p_{12}= &AIPL; **PartsSticking** &AIPL;isPerformedOn &AIPL;CO &AIPL; **PartsSticking** rdf;type &AIPL;Sticking &AIPL;**PartsSticking** &AIPL;needsPTime &AIPL;SPTime &AIPL; **PartsSticking** &AIPL;needsETime &AIPL;SETime &AIPL;Sticking &AIPL;hasNextOperation some &AIPL;Drying &AIPL; SPTime &AIPL;hours 0 &AIPL; SETime &AIPL;hours 0.064
e_{21}= 'Prods Assembling'	p_{14}= &AIPL; **ProdsAssembling** &AIPL;isPerformedOn &AIPL;SPF &AIPL;**ProdsAssembling** &AIPL;needsPTime &AIPL;APTime &AIPL;**ProdsAssembling** &AIPL;needsETime &AIPL;AETime &AIPL; SPF &AIPL;isLocatedAt &AIPL;AS &AIPL; AETime &AIPL;hours 0.5 &AIPL; APTime &AIPL;hours 0.1

Once the semantic annotations are created in the Sage X3 data model, the corresponding elements matching interface in the Sage X3 plug-in is able to assist the stakeholder to fill the right data into the right fields of the “process planning” table.

5.3 Conclusion

This case study shows the use of the formal semantic annotation proposal in a particular application scenario. It also shows one of the possible ways of using semantic annotations for assisting the model creation and conflict identification in the current system and for assisting the semantic interoperability with the upstream and downstream systems. Though the SAP-KM is only developed for MEGA modelling environment, it also shows the possibility of using the same method to apply the formal semantic annotation into other systems. Although the case study allow us to see the applicability of the proposed solution, in the future work, it should be better to be validated in a real and larger scale industrial facility with more applications and more complex information flows.

Chapter 6 Conclusions and Future Works

This chapter provides a conclusion of this research work and a perspective for the future work. It first gives an overall summary by answering the five research questions that we listed in Section 6.1. Then, the main contributions are elaborated in Section 6. 2. Finally, in Section 6.3, we identify the limitations of the current work and the possible future research directions.

6.1 Research Questions and Answers

In Chapter 1, we listed five major research questions that this work attempts to answer:

(1) What are the semantic interoperability problems that exist during the cooperation in a PLM environment?

A Product Lifecycle Management (PLM) approach aims at providing a shared platform, which enables the collaboration of all different enterprise systems that participate in each stage of the Product Life Cycle (PLC) in or across enterprises. Interoperability serves as a foundational role to support collaboration, which, in this context, is considered as the ability of diverse systems to be able to exchange and make use of the knowledge representations between each other. Currently, the open issues of a seamless semantic interoperability have attracted many research attentions. The different backgrounds, heterogeneous expertise, unique knowledge, particular needs and specific practices of stakeholders, also over increase this issues, not only in the collaboration across the enterprise systems but also in the achieving a mutual understanding between the stakeholders. Chapter 1 discusses the open issues of interoperability, especially the semantic interoperability, within the context of a PLM environment and it identifies that semantic enrichment is one of the solutions that could deal with these issues. A survey is carried out in Chapter 2 to investigate: (*i*) the models and their meta-models in a PLM environment that need semantic enrichments, (*ii*) the ontology specification languages and existing ontologies that can be used to support the semantic enrichment and (*iii*) the current semantic annotation researches that deal with the semantic interoperability issues in different domains. Based on this investigation, the research scope is narrowed down to a study of proposing a formalization of semantic annotations that deal with the existing drawbacks and potential challenges (Chapter 3). Our research work prototypes an annotation tool that shows one of the possible implementations of the proposed solution (Chapter 4). It validates the prototype in an application scenario to show how the proposed solution is able to semantically enrich the models and assist the semantic interoperability (Chapter 5).

(2) *What kinds of knowledge representation in a PLM environment need semantic enrichment?*

Based on the introduction of the product life cycle (Section 1.2.1), the product lifecycle management (Section 1.2.2) and the discussion of the knowledge management (Section 1.2.3), we discovered that the knowledge about the system of interest can be represented in various kinds of knowledge representations in a product life cycle. While the enterprise modelling (Section 2.1.1) is considered as a process that tries to capture and represent knowledge for activating the interoperations in or across enterprises, in this thesis, we consider all different types of models along the product life cycle as the targets of semantic enrichments. Considering the diversity of models, we select and use the process model (Section 2.1.2) to act as the target of semantic enrichment. In reality, every knowledge representation in a PLM environment can be enriched. We perhaps need to think at this question in the opposite way: Whether are there semantic models that could provide sufficient semantics to enrich these knowledge representations?

(3) *What kinds of ontology can be used to support the semantic enrichments of those knowledge representations?*

At least two aspects of semantics need to be made explicit in order to obtain a mutual understanding of models through the semantic enrichment: the structure semantics, which describes how a model is modelled, and the domain semantics, which explains the context and the meaning of the model elements in a specific domain. Therefore, ontologies that are used to support the semantic enrichment are supposed to capture and represent both aspects of the knowledge. In this research work, we classified them as the PLC-related ontology and the meta-model ontology respectively. The major ontology specification languages that could be used to create ontologies (Section 2.2.1) and several existing ontologies that contain these two aspects of semantics (Section 2.2.2) are surveyed in Section 2.2. The former investigation is to assist the selection of an appropriate ontology specification language that could be used in this work, as well as, the latter one is to discover whether some ontologies could be reused for the semantic enrichments of models that exist in a PLM environment.

(4) *What are the essential elements of a semantic annotation and how to formally represent a semantic annotation in a suitable format?*

A detailed survey is carried out to classify and compare a number of current semantic annotation researches (Section 2.3.1) and then it discusses the existing drawbacks and potential challenges among these researches (Section 2.3.2). Based on the above literature analysis, in the context of the PLC, we focus our work on clearly identifying the essential elements of a semantic annotation by proposing a formalization that can be used to enrich both domain and structure semantics of different types of models. The meta-model of the semantic annotation (Section 3.1.1) is proposed to describe the major components of a semantic annotation with their interrelations. Taking advantages from the definitions of semantic blocks (Section 3.1.2), a formal definition of the semantic annotation is proposed. This definition identifies and depicts all the essential elements of a semantic annotation (Section 3.1.3). This formalization provides a blueprint for generating a semantic annotation schema. In Section 4.2.1 we present one of the possible designs of the data structure that formally represents the semantic annotations.

(5) *How to semantically enrich a knowledge representation and how can these enriched semantics contribute to the semantic interoperability in a PLM environment?*

Section 3.3.1 presents the main procedures of how to perform the semantic enrichments and how the enriched semantics can be used in a single knowledge representation. Section 3.3.2 gives an overall architecture of the framework, which supports annotator to perform semantic enrichment and to use the semantic annotations in a PLM environment. A prototype annotation tool is designed and implemented to instantiate the formal semantic annotations. It also demonstrates its applicability and usability of our semantic enrichment solution. Taking advantages from the design of the data structure and the semantic annotation workflow, we design two steps of semantics enrichment (Section 4.2.2 and 4.2.3) and implemented them (Section 4.3.1) in the prototype tool. It assists an annotator in using the meta-model ontologies and the PLC-related ontologies to make explicit the structure semantics and the domain semantics in a model.

In this work, the enriched semantics is mainly used to contribute in two main aspects: assisting the creation of models and supporting the detection of possible mistakes. Three main steps are proposed for achieving these two purposes: the suggestion of semantic annotations (Section 3.2.1), the detection of inconsistencies between semantic annotations and the identification of possible conflicts in a model (Section 3.2.3). Based on these three steps, we designed the corresponding functions in the prototype annotation tool (Section 4.2.3) and we implemented them (Section 4.3.2 and 4.3.3) for supporting an annotator to perform the reasoning on the existing semantic annotations.

6.2 Summary of the Contributions

In conclusion, the proposed solution in this thesis provides some fundamental contributions, which are remarked as follows:

- We presented an in-depth survey on the current semantic annotation researches, which provides a detailed classification and a comprehensive comparison. Based on this investigation, we identify the existing drawbacks and potential challenges.
- The semantic annotation meta-model that unambiguously describes the major components of a semantic annotation and their interrelations. The formal definitions of a semantic annotation, which can be used as a basis to create semantic annotation schemas for realizing the semantic enrichment of models.
- We proposed two kinds of semantic blocks: the semantic blocks for the semantics description and the semantic blocks for semantics substitution. This proposition can be adapted to other researches that need the aggregation of semantics.
- The three reasoning mechanisms that show and validate the usages of semantic enrichments.
- The semantic annotation procedure that provides a guideline to show how to apply formal semantic annotations and how to benefit from them.
- The semantic annotation framework that shows the four main modules and their inter actions that are needed to perform the semantic enrichment along a PLM environment.
- The semantic annotation prototype, which shows as much as possible the details of the life cycle of a semantic annotation from the initial creation step to the final inference step.

6.3 Limitations and Perspectives

From a general point of view, the discussion of the limitations and the perspectives can start from the three hypotheses that are presented in the Section 1.2.5. These hypotheses, highlighted in this research work, can be considered as three important factors that affect the semantic enrichment.

(1) *All the knowledge that is needed for the semantic enrichment of models has already been captured, represented and formalized into ontologies*.

We understand that the cost of the creation of a new ontology and the management of ontology need lots of resources and might decrease the benefits of related approach. However in order to achieve the semantic interoperability, one way or another, a common and shared knowledge base needs to be created. Ontology that captures stakeholder's knowledge can be widely used in many ways. For example, it can be used for improving the semantic information retrieval [125], supporting meaningful semantic verification [126] and assisting the semantic integration of information[127]. Semantic annotation is a way to support the semantic interoperability, which is also benefiting from those formalized concepts in ontology. Though this hypothesis is strong assumptions that limit the research problem area, the researches to support them are out of the objective of this P.hD research.

(2) *The corresponding interconnections among all the used ontologies have already been prepared through certain methods*.

The interconnections among ontologies are the fundamental of using multiple ontologies together to perform semantic enrichment and perform inference on semantic annotations. The absence of these interconnections results in unsafe inference results. Reasoning engines are not able to perform reasoning on concepts coming from different ontologies which do not have relationships (directly or indirectly) between each other.

(3) *The semantic similarity between two objects can be compared through certain mechanisms*.

In this work, the semantic relationships between two objects are used as the basis to support the reasoning. The more precise semantic similarities are, the more precise results can be produced. These semantic similarity comparison results might be enhanced with the assistance of the Nature Language Processing techniques and the Artificial Intelligence techniques.

From the practical point of view, the prototype implementation and validation shows the possibility of using the formalization of semantic annotations for system interoperability in a PLM environment. However, the limitations of this implementation need to be pointed out. The SAP-KM is not a commercial software, but a tool developed through previous and on-going research. This positioning of the prototype results in complex graphical user interfaces. Several possible improvements are listed as perspectives:

(1) Creating mappings between the meta-model of the model of interest and the selected meta-model ontology for assisting the automatic explicitation of structure semantics.
(2) Creating automatic semantic block delimitation mechanisms to assist the *explicitation* of domain semantics (see for example, the method proposed by

Yahia et al. [118]).

(3) Creating a more complete set of SBR delimitation rules to decrease the annotators' workload. This is for taking in to account all the possible meta-model element combinations that can be found in a modelling language specification.

(4) Improving the reasoning engine (as discussed at the end of the Section 5.2.3) to enable the semi-automatic or automatic similarity comparison between two domain semantics of an annotated object.

(5) Applying the proposed solution in a real and larger scale industrial facility with more applications and more complex information flows. In this way, the applicability of the proposed solution can be evaluated more completely.

Furthermore, in the context of a PLM, three interesting directions can also be considered as future works.

(1) *For enabling the traceability of requirements*. With the assistance of semantics annotation, it is possible to trace the validation of each requirement in every stage of the product lifecycle, from the initial design until the final deposit of.

(2) *For the explicitation the relationships among TKRs*. Semantic annotations not only make explicit the "implicit semantics" and guarantee the correctness of "explicit semantics" of a TKR, but they can also be used to make explicit the hidden relationships among all the disperse TKRs along the product lifecycle.

(3) *For addressing the versioning of models*. The issue about the versioning of models in a PLC is difficult to be avoided. Semantically enriching models gives the possibility to ensure that the modified model contents do not semantically in conflict with existing ones.

In a nutshell, the purpose of this work is to deal with the issue of semantic interoperability. Despite of some limitations, as discussed in this section, we are convinced that the proposed formalization of semantic annotations is able to support and guarantee the mutual understanding of the semantics inside the shared and exchanged information in a PLM environment.

APPENDICES

APPENDIX Number	Content
APPENDIX I	Web Ontology Language (OWL)
APPENDIX II	Semantic Annotation Schema
APPENDIX III	Pseudo Codes

APPENDIX I Web Ontology Language (OWL)

Web Ontology Language (OWL) is a language for defining and instantiating web ontologies, which provides three increasingly expressive sub-languages: OWL Lite, OWL DL and OWL Full [128]. OWL Lite is considered as a simplest version of OWL with a lower expressiveness, which only provides a classification hierarchy and simple constraint features. On the contrary, OWL Full provides the maximum expressiveness and the syntactic freedom of Resource Description Framework (RDF), but without computational guarantees. Compared with OWL Lite and OWL Full, OWL DL not only supports the maximum expressiveness without losing computational completeness but also provides the decidability for inference. Due to the needs of expressive restriction constructs and the supports degrees of reasoning, OWL DL is frequently adopted by many different researches.

The basic elements of an OWL ontology are classes (OWL class), properties (OWL property) and instances of classes (OWL individual). An OWL class is used to define a concept in an ontology, while an OWL individual is used to define one member of an OWL class. An OWL property is a binary relation, which can be classified into two kinds:

(1) Datatype property (*owl:DatatypeProperty*) that signifies the relation between OWL individuals and a RDF literal (e.g. *refs:Literal*) or a XML Schema datatype (e.g. *xsd:srting*);

(2) Object property (*owl:ObjectProperty*) that represents the relations between two OWL individuals. This relation could have a domain (*rdfs:domain*) and a range (*rdfs:range*).

In order to provide a more powerful mechanism for enhanced reasoning capability, OWL specifies five property characteristics, as shown in the Table I-1.

Table I-1 Five types of OWL Property Characteristics [128]

Property Characteristic	**Assumed Conditions**	**Conclusion**
TransitiveProperty	P is a transitive property ; x, y and z are OWL individuals.	$P(x, y) \land P(y, z) \rightarrow P(x, z)$
SymmetricProperty	P is a symmetric property; x and y are OWL individuals.	$P(x, y) \leftrightarrow P(y, x)$
FunctionalProperty	P is a functional property; x, y and z are OWL individuals.	$P(x, y) \land P(x, z) \rightarrow y=z$
InverseOf	P is inverse property of Q; x and y are OWL individuals.	$P(x, y) \leftrightarrow Q(y, x)$
InverseFunctionalProperty	P is an inverse functional property;	$P(y, x) \land P(z, x) \rightarrow y=z$

	x, y and z are OWL individuals.	

Besides designating the property characteristics, OWL distinguishes two kinds of property restrictions to support the class description, as shown in the Table I-2: (1) value constraint, which puts constraints on the range of a property; (2) cardinality constraint, which puts constraints on the number of values a property can contain.

Table I-2 Two kinds of OWL Property Restrictions [79]

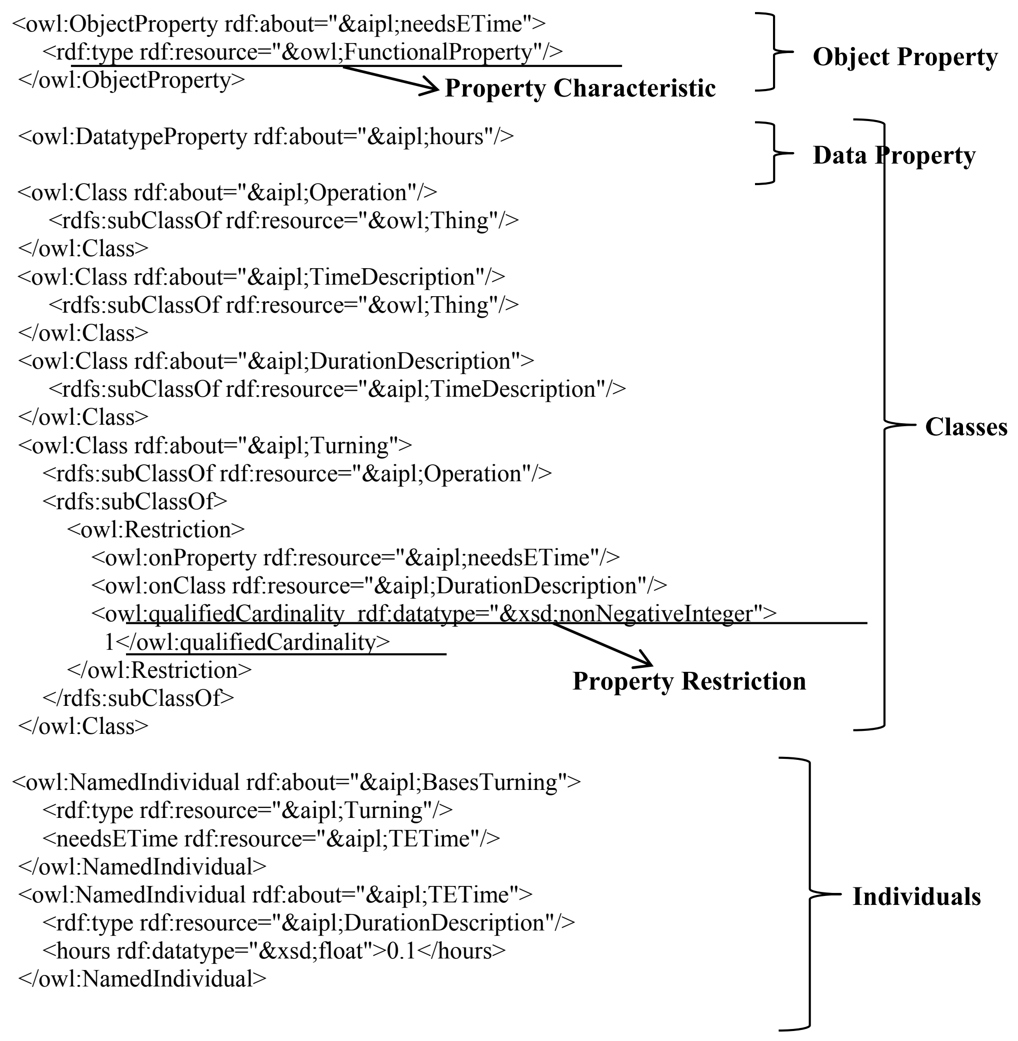
Types of Restriction	**Property Restriction**	**Conditions and Conclusions**
Value Constraint[26]	allValuesFrom	P has an allValuesFrom constraint on class D and links to R; R can be a class description or data range. $\forall x \in D \land P(x, y) \rightarrow \forall y \in R$
	someValuesFrom	P has a someValuesFrom constraint on class D and links to R; R can be a class description or data range. $\forall x \in D \land P(x, y) \rightarrow \exists y \in R$
	hasValue	P has a hasValue constraint on class D and links to a values v; v can be an individual or a data value. $\forall x \in D \land P(x, y) \rightarrow \exists y = v$
Cardinality Constraint[27]	maxCardinality	P has a maxCardinality constraint on class D and links to n; n is a nonnegative integer data value. $\forall x \in D \land P(x, y) \rightarrow$ for each x, there is at most n semantically distinct y.
	miniCardinality	P has a minCardinality constraint on class D and links to n; n is a nonnegative integer data value $\forall x \in D \land P(x, y) \rightarrow$ for each x, there is at least n semantically distinct y.
	cardinality	P has a cardinality constraint on class D and links to n; n is a nonnegative integer data value. $\forall x \in D \land P(x, y) \rightarrow$ for each x, there is exactly n semantically distinct y.

Further, OWL also provides a number of basic set operators (such as union, intersection and complement) to support the creation of the class expressions. Figure I-1 shows an example of a part of an OWL ontology, in which, “Operation”, “TimeDescription”, “DurationDescription” and “Turning” are defined as OWL classes. The “BasesTurning” is an individual of the class “Operation”. The “TETime” is an individual of the class “DurationDescription”. The OWL object property “needsETime” is characterised as a functional property. Together with a cardinality constraint, the

[26] In hasValue constraint, “ = ” is semantically equal: when v is a data value, it means same value; when v is an individual, it means with the same URI reference or defined as the same individual (owl:sameAs) [79].

[27] Semantically distinct: For datatypes, it means different values; for individuals, it means the those individuals are defined as the different individuals from each other (owl: differentFrom or owl:AllDifferent)

"needsETime" used to define the relationship between the class "Operation" and the class "DurationDescription"; the "hours" is an OWL data property, which defines the relationship between "TETime" and a data value "0.1".



```
<owl:ObjectProperty rdf:about="&aipl;needsETime">
   <rdf:type rdf:resource="&owl;FunctionalProperty"/>
 </owl:ObjectProperty>

 <owl:DatatypeProperty rdf:about="&aipl;hours"/>

 <owl:Class rdf:about="&aipl;Operation"/>
     <rdfs:subClassOf rdf:resource="&owl;Thing"/>
 </owl:Class>
 <owl:Class rdf:about="&aipl;TimeDescription"/>
     <rdfs:subClassOf rdf:resource="&owl;Thing"/>
 </owl:Class>
 <owl:Class rdf:about="&aipl;DurationDescription">
     <rdfs:subClassOf rdf:resource="&aipl;TimeDescription"/>
 </owl:Class>
 <owl:Class rdf:about="&aipl;Turning">
    <rdfs:subClassOf rdf:resource="&aipl;Operation"/>
    <rdfs:subClassOf>
       <owl:Restriction>
          <owl:onProperty rdf:resource="&aipl;needsETime"/>
          <owl:onClass rdf:resource="&aipl;DurationDescription"/>
          <owl:qualifiedCardinality  rdf:datatype="&xsd;nonNegativeInteger">
           1</owl:qualifiedCardinality>
       </owl:Restriction>
    </rdfs:subClassOf>
 </owl:Class>

<owl:NamedIndividual rdf:about="&aipl;BasesTurning">
    <rdf:type rdf:resource="&aipl;Turning"/>
    <needsETime rdf:resource="&aipl;TETime"/>
 </owl:NamedIndividual>
 <owl:NamedIndividual rdf:about="&aipl;TETime">
    <rdf:type rdf:resource="&aipl;DurationDescription"/>
    <hours rdf:datatype="&xsd;float">0.1</hours>
 </owl:NamedIndividual>
```

Figure I-1 The Example of a Segment of an OWL ontology

Although OWL has a series of language constructs to support the representation of knowledge, as indicated in research [129], it still lacks power of expression capabilities; for example, the expression of relation chain (e.g. child of married parents). Because of this reason, reasoning rules are proposed to address this issue trough expressing and adding user-defined rules as a complement to the OWL.

A reasoning rule can be considered as an implication between an antecedent (body) and a consequent (head), which intends to state that if the conditions specified in the antecedent part are satisfied, then the conditions specified in the consequent part must also be satisfied. Various kinds of reasoning rules have been proposed, for example, Jena Rules [123], SWRL [130], Jess Rules[131] and so on. Different reasoning rules have their own rule format and corresponding reasoning engines. Figure I-2 shows the abstract syntax of Jena Rules.

```
Rule      ::=  bare-rule .
          or   [ bare-rule ]
          or   [ ruleName : bare-rule ]
bare-rule  :=  term, ... term -> hterm, ... hterm    // forward rule
             or  term, ... term <- term, ... term    // backward rule
hterm     ::=  term
          or   [ bare-rule ]
term      ::=  (node, node, node)           // triple pattern
          or   (node, node, functor)        // extended triple pattern
          or   builtin(node, ... node)      // invoke procedural primitive
functor   ::=  functorName(node, ... node)  // structured literal
node      ::=  uri-ref                      // e.g. http://foo.com/eg
          or   prefix:localname             // e.g. rdf:type
          or   <uri-ref>                    // e.g. <myscheme:myuri>
          or   ?varname                     // variable
          or   'a literal'                  // a plain string literal
          or   'lex'^^typeURI               // a typed literal, xs:* type names supported
          or   number                       // e.g. 42 or 25.5
```

Figure I-2 The Abstract Syntax of Jena Rules[123]

With supports from existing reasoning engines, such as, among others, Jena Reasoners [132], Pellet [133] and Jess Rule Engine [131], ontology users are able to discover the advantages of reasoning based on the input ontologies and corresponding reasoning rules. For example, to mention only a few, the consistency checking (guarantees an ontology has not contradictory facts), the concept satisfiability (verifies whether a class can have any instances or not), the classification (determines the subclass relations between every named class), the realization (finds the most specific classes that an instance belongs to).

In this work, the Jena rules syntax is chosen as the format to formalize our reasoning rules. Consequently the Jena Reasoners are used as the reasoning engines to support the development of prototype annotation tool.

APPENDIX II Semantic Annotation Schema

The empty semantic annotation schema is show as follows:

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY j.0 "http://jena.hpl.hp.com/2003/RuleReasoner#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
<!ENTITY SANS "http://www.semanticweb.org/ontologies/2013/6/SemanticAnnotations#" >

]>

<rdf:RDF xmlns="http://www.semanticweb.org/ontologies/2013/6/SemanticAnnotations#"
     xml:base="http://www.semanticweb.org/ontologies/2013/6/SemanticAnnotations"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:j.0="http://jena.hpl.hp.com/2003/RuleReasoner#"
     xmlns:SANS="http://www.semanticweb.org/ontologies/2013/6/SemanticAnnotations#">
   <owl:Ontology rdf:about="http://www.semanticweb.org/ontologies/2013/6/SemanticAnnotations"/>

   <owl:ObjectProperty rdf:about="&SANS;PR"/>

   <owl:ObjectProperty rdf:about="&SANS;PR_intersects">
      <rdf:type rdf:resource="&owl;SymmetricProperty"/>
      <rdfs:range rdf:resource="&SANS;P"/>
      <rdfs:domain rdf:resource="&SANS;P"/>
      <rdfs:subPropertyOf rdf:resource="&SANS;PR"/>
   </owl:ObjectProperty>

   <owl:ObjectProperty rdf:about="&SANS;PR_isDisjointWith">
      <rdf:type rdf:resource="&owl;SymmetricProperty"/>
      <rdfs:range rdf:resource="&SANS;P"/>
      <rdfs:domain rdf:resource="&SANS;P"/>
      <rdfs:subPropertyOf rdf:resource="&SANS;PR"/>
   </owl:ObjectProperty>

   <owl:ObjectProperty rdf:about="&SANS;PR_isEquivalentTo">
      <rdf:type rdf:resource="&owl;SymmetricProperty"/>
      <rdf:type rdf:resource="&owl;TransitiveProperty"/>
      <rdfs:domain rdf:resource="&SANS;P"/>
      <rdfs:range rdf:resource="&SANS;P"/>
      <rdfs:subPropertyOf rdf:resource="&SANS;PR"/>
   </owl:ObjectProperty>

   <owl:ObjectProperty rdf:about="&SANS;PR_isSubsumedBy">
      <rdf:type rdf:resource="&owl;TransitiveProperty"/>
      <rdfs:range rdf:resource="&SANS;P"/>
      <rdfs:domain rdf:resource="&SANS;P"/>
      <rdfs:subPropertyOf rdf:resource="&SANS;PR"/>
   </owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:about="&SANS;PR_subsumes">
   <rdf:type rdf:resource="&owl;TransitiveProperty"/>
   <rdfs:domain rdf:resource="&SANS;P"/>
   <rdfs:range rdf:resource="&SANS;P"/>
   <rdfs:subPropertyOf rdf:resource="&SANS;PR"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&SANS;SR"/>

<owl:ObjectProperty rdf:about="&SANS;SR_intersects">
   <rdfs:domain rdf:resource="&SANS;E"/>
   <rdfs:range rdf:resource="&SANS;P"/>
   <rdfs:subPropertyOf rdf:resource="&SANS;SR"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&SANS;SR_isDisjointWith">
   <rdfs:domain rdf:resource="&SANS;E"/>
   <rdfs:range rdf:resource="&SANS;P"/>
   <rdfs:subPropertyOf rdf:resource="&SANS;SR"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&SANS;SR_isEquivalentTo">
   <rdfs:domain rdf:resource="&SANS;E"/>
   <rdfs:range rdf:resource="&SANS;P"/>
   <rdfs:subPropertyOf rdf:resource="&SANS;SR"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&SANS;SR_isSubsumedBy">
   <rdfs:domain rdf:resource="&SANS;E"/>
   <rdfs:range rdf:resource="&SANS;P"/>
   <rdfs:subPropertyOf rdf:resource="&SANS;SR"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&SANS;SR_subsumes">
   <rdfs:domain rdf:resource="&SANS;E"/>
   <rdfs:range rdf:resource="&SANS;P"/>
   <rdfs:subPropertyOf rdf:resource="&SANS;SR"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&SANS;hasMainConcept">
   <rdfs:domain rdf:resource="&SANS;P"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&SANS;hasSBEntity">
   <rdfs:domain rdf:resource="&SANS;P"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&SANS;hasSBRelation">
   <rdfs:domain rdf:resource="&SANS;P"/>
   <rdfs:range rdf:resource="&SANS;SBRelations"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&SANS;isAnnotatedBy">
   <rdfs:domain rdf:resource="&SANS;E"/>
   <rdfs:range rdf:resource="&SANS;P"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&SANS;isConflictWith">
   <rdf:type rdf:resource="&owl;SymmetricProperty"/>
```

```
        <rdfs:domain rdf:resource="&SANS;E"/>
        <rdfs:range rdf:resource="&SANS;E"/>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="&SANS;isConsistentWith">
        <rdf:type rdf:resource="&owl;SymmetricProperty"/>
        <rdfs:range rdf:resource="&SANS;P"/>
        <rdfs:domain rdf:resource="&SANS;P"/>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="&SANS;isInferredFrom">
        <rdfs:range rdf:resource="&SANS;E"/>
        <rdfs:domain rdf:resource="&SANS;P"/>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="&SANS;isNotConsistentWith">
        <rdf:type rdf:resource="&owl;SymmetricProperty"/>
        <rdfs:range rdf:resource="&SANS;P"/>
        <rdfs:domain rdf:resource="&SANS;P"/>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="&SANS;isPosConsistentWith">
        <rdf:type rdf:resource="&owl;SymmetricProperty"/>
        <rdfs:domain rdf:resource="&SANS;P"/>
        <rdfs:range rdf:resource="&SANS;P"/>
    </owl:ObjectProperty>

    <owl:DatatypeProperty rdf:about="&SANS;hasLongNS"/>

    <owl:Class rdf:about="&SANS;E">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>

    <owl:Class rdf:about="&SANS;MME">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>

    <owl:Class rdf:about="&SANS;NSstore">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>

    <owl:Class rdf:about="&SANS;P">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>

    <owl:Class rdf:about="&SANS;SBRelations">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>
</rdf:RDF>
```

APPENDIX III Pseudo Codes

Pseudo Code 1. This pseudo code is in charge of adding a selected individual, its properties and the objects of the properties into a semantic block, namely the contents of a ***p*** (an individual of the Class "*P*")

```
Get p, the selected individual, the selected property and the object of the propoerty
// the function to add  a selected individual, its properties and the objects of the properties into a semantic   block,
namely the contents of a p (an individual of the Class "P")
{   Get the namespaces of the selected individual(ni), property(np), and the object(no);
    Get the local names of the selected individual (li), property(lp), and the object(lo);
    Get the list of objects that the "hasSBEntity" (p's) points to;
    If the selected individual is not existed in this list
    {   Add "hasSBEntity" from p to the selected individual;
    }
    If the selected object is not existed in this list
    {   Add "hasSBEntity" from p to the object;
    }
    Get the list of individuals of the class "SBRelations";
    Get the first six characters of the local names of these individuals, and put them into a
    string array LN; Produce a random six numbers (to string) rn, until it do not equal to any
    string in the LN;

    Get the list of individuals of the class "NSstore";
    Get the numbers of individuals  n of the class "NSstore";
    Create a string array L;
    Create a variable nis to represent the abbreviation of li;
    Create a variable i to compare with the numbers of individuals;
    For each the individual of the class "NSstore"
    {   Get the object of the property "hasLongNS" (this individual's) points to;
        Put the local name of the individual into L;
        i++;
        if the object (to string) equals to ni;
        {     Set nis as the local name of this individual;
        }
    }
    If the i < n
    {   Set nis as a random six letters string, until it do not equal to any strings in the  L;
       Create an individual X to the Class "NSstore", named it as the value of nis;
        Add "hasLongNS" from X to a data value equals to the ni;
    }
    Same processes for np and no, to acquire their abbreviation nps and nos

    Format a string S as rn+"-"+nis+"- "+li+"_____"+nps+"-"+lp+"_____"+nos+"-"+lo;
    Create an individual K of  class "SBRelations", named it as the string S;
    Add "hasSBRelation" from p to K;
}
```

Pseudo Code 2. This pseudo code is in charge of decomposing a SBRelation

```
Get the SBRelation   //The function to decompose a SBRelation and return real values
{    Decompose SBRelation based on its name syntax;
     Get the three namespace abbreviations na1, na2 and na3;
     Get the three local names n1, n2 and n3;
     Create three variable v1, v2 and v3;
     Get the list of individuals Z of the class "NSstore";
     For each individual i1 in the list Z
     {       If the local name of the i2 equals to the na1
             {  Get the real name space rns1 through the property "hasLongNS";
                Set the value of v1 as the value of rns1;
             }
             Same processes for the na2 to get the real name space and put into v2;
             Same processes for the na3 to get the real name space and put into v3;
         }
     }
     Format a string s1 as v1+n1;
     Format a string s2 as v2+n2;
     Format a string s3 as v3+n3;
     Return the  s1, s2, s3
}
```

Pseudo Code 3. This pseudo code is in charge of getting the list of Properties of a ***p***'s main concept.

```
Get the InfModel;
Get the selected individual e;
Get the list of individuals X in the class "P", which are used to annotate the e;
For each individual p in the list X
{   Get the main concept mc of the  p through the property "hasMainConcept";
    Get the list of SBRelations Y of the p through the property "hasSBRelation"';
    For each individual i1 in the list Y
    {   Use the SBRelation decomposition function on the i1 and get the s1, s2 and s3;
        //( The Pseudo Codes 2)
         If the s1equals to mc
        {   Add the s2 to the list view;
         }
    }
```

Pseudo code 4. This pseudo code is in charge of the suggestion of semantic annotations.

```
Get the InfModel;            // of course all the property needs their namespaces
Get the list of association;
For each association in the that list;
{   Get the first property x and the second property y;
    Get the list of individuals L1of the class "E";
    For each individual i1 in the list L1
    {   Get the list of objects L2 of the i1 through the first property x;
        For each object o1 in the list L2
        {Get the list of objects L3 of the i1 through the property "SR_subsumes"
          For each object p1 in the list L3
          { Get the main concept mc of the  p1 through the property "hasMainConcept";
            Get the list of objects L4 of the mc through the second property y
            For each object e1 in the list L4
            {Create an individual p2 in the class "P" based on the name syntax of the $p_j$;
               Add "hasMainConcept" from the p2 to the e1;
               Create a string array AR;
               Get the list of objects L5 of the p1 through the property "hasSBRelation"
               For each object SBRelation1 in the list L5
               {    Use the SBRelation decomposition function on the SBRelation1
                    and get the s1, s2 and s3; //( The Pseudo Code 2)
                    Save the s1, s2 and s3 into AR;
               }
               Traverse the array AR, discover all the paths related to the e1
               Add traverse results to p2;  //(The Pseudo Code 1)
               Add "SR_subsumes" property from the e1 to the p2;
            }
          }
        }
    }
}
```

Pseudo Code 5. This pseudo code is in charge of one of the mistake identification algorithms.

```
Get the InfModel;            // of course all the property needs their namespaces
Get the list of association;
For each association in the that list;
{  Get the first property x;
   Get the list of individuals L1of the class "E";
   For each individual i1 in the list L1
   {  Get the list of objects L2 of the i1 through the first property x;
      Create array AR1, AR2 and AR3;
      For each object o1 in the list L2
      {  Get the list of object L3 of the o1 through the property "isAnnotatedBy"
         For each individual i2 in the list L3
         {  If i2 has the property "isInferredFrom"
            {  If the object of this property is i1
               {  Save the i2 into AR2;}
            }
            else
            {  Save the i2 into AR1;}
         }
         For each individual i3 in AR1
        {  Create a variable v1=0;

            For each individual i4 in AR2
            { If i3 has the property "isConsistentWith" or "isPosConsistentWith" to i4
              {  Save the i4 into AR3;
                 v1=1;
              }
            }
            If v1=0    // i3 is conflict with all inferred semantic annotations
            {  Add property "isConflictWith" between i1 and o1;}
         }
      }
     For each individual i5 in the AR2, if i5 is not existed in the AR3
     {
          Get the main concept mc of i5,
          Add a warning to specify the mc (i5) of i1 is not be satisfied ;j
     }
   }
}
```

References

1. Ameri, F., Dutta, D.: Product lifecycle management: closing the knowledge loops. Comput.-Aided Des. Appl. 2, 577–590 (2005).
2. CIMdata: PDM to PLM: Growth of An Industry. (2003).
3. Elgueder, J., Cochennec, F., Roucoules, L., Rouhaud, E.: Product–process interface for manufacturing data management as a support for DFM and virtual manufacturing. Int. J. Interact. Des. Manuf. IJIDeM. 4, 251–258 (2010).
4. Ackoff, R.L.: From Data to Wisdom. J. Applies Syst. Anal. 16, 3–9 (1989).
5. Zeleny, M.: Management support systems: towards integrated knowledge management. Management. 1, 7 (1980).
6. Leibniz, G.W.: The Monadology. In: Loemker, L.E. (ed.) Philosophical Papers and Letters. pp. 643–653. Springer Netherlands, Dordrecht (1989).
7. O'Leary, D.E.: Enterprise knowledge management. Computer. 31, 54–61 (1998).
8. Euzenat, J.: Towards a principled approach to semantic interoperability. Presented at the IJCAI 2001, workshop on ontology and information sharing , Seattle, USA August (2001).
9. Panetto, H.: Towards a classification framework for interoperability of enterprise applications. Int. J. Comput. Integr. Manuf. 20, 727–740 (2007).
10. Boudjlida, N., Panetto, H.: Annotation of enterprise models for interoperability purposes. In: IEEE (ed.) IEEE International Workshop on Advanced Information Systems for Enterprises, IWAISE'2008. pp. 11–17. , Constantine, Algeria (2008).
11. Oren, E., Hinnerk Möller, K., Scerri, S., Handschuh, S., Sintek, M.: What are Semantic Annotations? (2006).
12. Liao, Y., Lezoche, M., Panetto, H., Boudjlida, N.: Semantic Annotation Model Definition for Systems Interoperability. In: Meersman, R., Dillon, T., and Herrero, P. (eds.) On the Move to Meaningful Internet Systems: OTM 2011 Workshops. pp. 61–70. Springer Berlin Heidelberg (2011).
13. Liao, Y., Lezoche, M., Panetto, H., Boudjlida, N.: Why, Where and How to use Semantic Annotation for Systems Interoperability. 1st UNITE Doctoral Symposium. pp. 71–78. , Bucarest, Roumanie (2011).
14. Liao, Y., Lezoche, M., Rocha Loures, E., Panetto, H., Boudjlida, N.: Formalization of Semantic Annotation for Systems Interoperability in a PLM environment. OTM 2012 Workshops 2nd INBAST. pp. 207–218. Rome, Italy (2012).
15. Liao, Y., Lezoche, M., Rocha Loures, E., Panetto, H., Boudjlida, N.: Semantic Enrichment of Models to Assist Knowledge Management in a PLM environment. 21 st International Conference on Cooperative Information Systems (CoopIS 2013). , Graz, Austria (2013).
16. Cross, N.: Designerly ways of knowing: design discipline versus design science. Des. Issues. 17, 49–55 (2001).
17. Rink, D.R., Swan, J.E.: Product life cycle research: A literature review. J. Bus. Res. 7, 219–242 (1979).
18. Stark, J.: Product Lifecycle Management. Product Lifecycle Management. pp. 1–16. Springer London (2011).
19. Birou, L., Fawcett, S.E., Magnan, G.M.: Integrating Product Life Cycle and Purchasing Strategies. J. Supply Chain Manag. 33, 23–31 (1997).
20. Ball, A., Ding, L., Patel, M.: An approach to accessing product data across system

and software revisions. Adv. Eng. Informatics. 22, 222–235 (2008).
21. Abramovici, M.: Future trends in product lifecycle management (PLM). The future of product development. pp. 665–674. Springer (2007).
22. Hewett, A.: Product Lifecycle Management (PLM): Critical Issues and Challenges in Implementation. Information Technology and Product Development. pp. 81–105. Springer (2009).
23. Bellinger, G., Castro, D., Mills, A.: Data, information, knowledge, and wisdom. (2004).
24. Stafford, S.P.: Data, information, knowledge, and wisdom. Knowl. Manag. Organ. Intell. Learn. Complex. Encycl. Life Support Syst. EOLSS Www Eolss Net Accessed. 22, (2006).
25. Nunamaker, J.J.F., Romano Jr, N.C., Briggs, R.O.: Increasing intellectual bandwidth: generating value from intellectual capital with information technology. Group Decis. Negot. 11, 69–86 (2002).
26. Polanyi, M.: The tacit dimension. Peter Smith Gloucester, MA (1966).
27. Nonaka, I.: A dynamic theory of organizational knowledge creation. Organ. Sci. 5, 14–37 (1994).
28. Kim, Y.-G., Yu, S.-H., Lee, J.-H.: Knowledge strategy planning: methodology and case. Expert Syst. Appl. 24, 295–307 (2003).
29. Tsoukas, H., Vladimirou, E.: What is organizational knowledge? J. Manag. Stud. 38, 973–993 (2001).
30. Yim, N.-H., Kim, S.-H., Kim, H.-W., Kwahk, K.-Y.: Knowledge based decision making on higher level strategic concerns: system dynamics approach. Expert Syst. Appl. 27, 143–158 (2004).
31. Polanyi, M.: Personal knowledge: Towards a post-critical philosophy. Psychology Press (1958).
32. Virtanen, I.: Epistemological Problems Concerning Explication Of Tacit Knowledge. J. Knowl. Manag. Pr. 11, (2010).
33. Davis, R., Shrobe, H., Szolovits, P.: What is a knowledge representation? AI Mag. 14, 17 (1993).
34. Prusak, L.: Where did knowledge management come from? IBM Syst. J. 40, 1002–1007 (2001).
35. Hlupic, V., Pouloudi, A., Rzevski, G.: Towards an integrated approach to knowledge management:"hard","soft"and "abstract"issues. Knowl. Process Manag. 9, 90–102 (2002).
36. Singh, S., Chan, Y.E., McKeen, J.D.: Knowledge Management Capability and Organizational Performance: A Theoretical Foundation. Conference at the University of Warwick, Coventry. pp. 1–54 (2006).
37. Quintas, P., Lefrere, P., Jones, G.: Knowledge management: A strategic agenda. Long Range Plann. 30, 385–391 (1997).
38. Olla, P., Holm, J.: The role of knowledge management in the space industry: important or superfluous? J. Knowl. Manag. 10, 3–7 (2006).
39. Schultze, U., Leidner, D.: Studying knowledge management in information systems research: discourses and theoretical assumptions. Manag. Inf. Syst. Q. 26, 213–242 (2002).
40. Nonaka, I., Takeuchi, H.: The knowledge-creating company: How Japanese companies create the dynamics of innovation. Oxford University Press, USA (1995).
41. IEEE Standard Computer Dictionary. A Compilation of IEEE Standard Computer

Glossaries. IEEE Std 610. 1– (1991).

42. Le Moigne, J.-L.: La théorie du système général: théorie de la modélisation. Presses Universitaires de France-PUF (1994).
43. W3C: Extensible Markup Language (XML) 1.1 (Second Edition), http://www.w3.org/TR/xml11/.
44. W3C: Web services description language (WSDL) 1.1, http://www.w3.org/TR/wsdl.
45. W3C: Semantic Annotations for WSDL and XML Schema, http://www.w3.org/TR/2007/REC-sawsdl-20070828/.
46. Commission, E.: European interoperability framework for pan-european egovernment services. IDA Work. Doc. Version. 2, (2004).
47. Pokraev, S., Quartel, D., Steen, M.W.A., Reichert, M.: Semantic Service Modeling: Enabling System Interoperability. In: Doumeingts, P.G., Müller, P.J., Morel, P.G., and Vallespir, P.B. (eds.) Enterprise Interoperability. pp. 221–230. Springer London (2007).
48. Etienne, A., Guyot, E., Wijk, D.V., Roucoules, L.: Specifications and development of interoperability solution dedicated to multiple expertise collaboration in a design framework. Int. J. Prod. Lifecycle Manag. 5, 272–294 (2011).
49. Gruber, T.R.: A translation approach to portable ontology specifications. Knowl. Acquis. 5, 199–220 (1993).
50. Maedche, A., Staab, S.: Measuring Similarity between Ontologies. Presented at the 13th International Conference on Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web , Sigüenza, Spain (2002).
51. Doan, A., Madhavan, J., Dhamankar, R., Domingos, P., Halevy, A.: Learning to match ontologies on the Semantic Web. VLDB J. 12, 303–319 (2003).
52. Stumme, G., Maedche, A.: Ontology merging for federated ontologies on the semantic web. Presented at the FMII-2001, International Workshop for Foundations of Models for Information Integration , Viterbo, Italy September (2001).
53. Čeh, M., Podobnikar, T., Smole, D.: Semantic Similarity Measures within the Semantic Framework of the Universal Ontology of Geographical Space. In: Riedl, D.A., Kainz, P.W., and Elmes, P.G.A. (eds.) Progress in Spatial Data Handling. pp. 417–434. Springer Berlin Heidelberg (2006).
54. Pesaranghader, A., Muthaiyah, S.: Definition-based Information Content Vectors for Semantic Similarity Measurement. In: Noah, S.A., Abdullah, A., Arshad, H., Bakar, A.A., Othman, Z.A., Sahran, S., Omar, N., and Othman, Z. (eds.) Soft Computing Applications and Intelligent Systems. pp. 268–282. Springer Berlin Heidelberg (2013).
55. Boudjlida, N., Dong, C., Baïna, S.: A practical experiment on semantic enrichment of enterprise models in a homogeneous environment. (2006).
56. Talantikite, H.N., Aissani, D., Boudjlida, N.: Semantic annotations for web services discovery and composition. Comput. Stand. Interfaces. 31, 1108–1117 (2009).
57. Lin, Y.: Semantic annotation for process models: Facilitating process knowledge management via semantic interoperability, PhD Thesis(2008).
58. Kiryakov, A., Popov, B., Terziev, I., Manov, D., Ognyanoff, D.: Semantic annotation, indexing, and retrieval. Web Semant. Sci. Serv. Agents World Wide Web. 2, 49 – 79 (2004).

59. Miller, J., Mukerji, J.: MDA Guide Version 1.0. 1. Object Manag. Group. 234, 51 (2003).
60. OMG: Meta Object Facility (MOF) Core Specification (2.4)., http://www.omg.org/spec/MOF/2.4.1/.
61. OMG: Unified Modeling Language (UML) Version 2.3, http://www.omg.org/spec/UML/2.3/.
62. Harel, D.: Statecharts: A visual formalism for complex systems. Sci. Comput. Program. 8, 231–274 (1987).
63. OMG: Business Process Model and Notation (BPMN) Version 2.0, http://www.omg.org/spec/BPMN/2.0/.
64. Krasner, H., Terrel, J., Linehan, A., Arnold, P., Ett, W.H.: Lessons learned from a software process modeling system. Commun. ACM. 35, 91–100 (1992).
65. Van der Aalst, W.M.: Formalization and verification of event-driven process chains. Inf. Softw. Technol. 41, 639–650 (1999).
66. OMG: Unified Modeling Language (UML) Version 2.0, http://www.omg.org/spec/UML/2.0/.
67. Uschold, M., Gruninger, M.: Ontologies: Principles, methods and applications. Knowl. Eng. Rev. 11, 93–136 (1996).
68. Song, F., Zacharewicz, G., Chen, D.: An ontology-driven framework towards building enterprise semantic information layer. Adv. Eng. Informatics. 27, 38–50 (2012).
69. Kifer, M., Lausen, G., Wu, J.: Logical Foundations of Object-Oriented and Frame-Based Languages. Presented at the (1995).
70. Fikes, R., Farquhar, A., Rice, J.: Tools for Assembling Modular Ontologies in Ontolingua. Presented at the Proceedings of the 14th national conference on artificial intelligence and 9th conference on Innovative applications of artificial intelligence (1997).
71. Genesereth, M.R., Fikes, R.E.: Knowledge interchange format-version 3.0: reference manual. (1992).
72. Motta, E.: An overview of the OCML modelling language. the 8th Workshop on Methods and Languages (1998).
73. MacGregor, R.M.: Inside the LOOM description classifier. ACM Sigart Bull. 2, 88–92 (1991).
74. Lenat, D.B., Guha, R.V.: The evolution of CycL, the Cyc representation language. ACM SIGART Bull. 2, 84–87 (1991).
75. W3C: Resource Description Framework (RDF), http://www.w3.org/RDF/.
76. Heflin, J., Hendler, J., Luke, S.: SHOE: A knowledge representation language for internet applications. (1999).
77. Hendler, J., McGuinness, D.L.: The DARPA agent markup language. IEEE Intell. Syst. 15, 67–73 (2000).
78. Fensel, D., Van Harmelen, F., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F.: OIL: An ontology infrastructure for the semantic web. Intell. Syst. IEEE. 16, 38–45 (2001).
79. W3C: OWL 2 Web Ontology Language, http://www.w3.org/TR/owl2-overview/.
80. Corcho, O., Fernández-López, M., Gómez-Pérez, A.: Methodologies, tools and languages for building ontologies. Where is their meeting point? Data Knowl. Eng. 46, 41–64 (2003).
81. Pulido, J.R.., Ruiz, M.A.., Herrera, R., Cabello, E., Legrand, S., Elliman, D.: Ontology languages for the semantic web: A never completely updated review.

Knowl.-Based Syst. 19, 489–497 (2006).
82. Roche, C.: ONTOLOGY: A SURVEY. Presented at the 8th Symposium on Automated Systems Based on Human Skill and Knowledge IFAC (2003).
83. Panetto, H., Dassisti, M., Tursi, A.: ONTO-PDM: Product-driven ONTOlogy for Product Data Management interoperability within manufacturing process environment. Adv. Eng. Informatics. 26, 334–348 (2012).
84. Zdravković, M., Panetto, H., Trajanović, M., Aubry, A.: An approach for formalising the supply chain operations. Enterp. Inf. Syst. 5, 401–421 (2011).
85. Barbau, R., Krima, S., Rachuri, S., Narayanan, A., Fiorentini, X., Foufou, S., Sriram, R.D.: OntoSTEP: Enriching product model data using ontologies. Comput.-Aided Des. 44, 575–590 (2012).
86. ISO: ISO/TS 10303 STEP modules related to Product Data Management. Industrial automation systems and integration - Product data representation and exchange. Geneva, Switzerland, (2004).
87. Spiby, P.: ISO 10303 industrial automation systems–product data representation and exchange–part 11: Description methods: The express language reference manual. ISO DIS. 10303–11 (1992).
88. Lee, C.-S., Wang, M.-H., Yan, Z.-R., Lo, C.-F., Chuang, H.-H., Lin, Y.-C.: Intelligent estimation agent based on CMMI ontology for project planning. IEEE International Conference on Systems, Man and Cybernetics, 2008. SMC 2008. pp. 228–233 (2008).
89. Chrissis, M.B., Konrad, M., Shrum, S.: CMMI: Guidelines for Process Integration and Product Improvement. 2008. Upper Saddle River, NJ: Addison-Wesley (2002).
90. Ameri, F., McArthur, C., Asiabanpour, B., Hayasi, M.: A web-based framework for semantic supplier discovery for discrete part manufacturing. SME/NAMRC. 39, (2011).
91. Ameri, F., Dutta, D.: A matchmaking methodology for supply chain deployment in distributed manufacturing environments. J. Comput. Inf. Sci. Eng. 8, 010301–1 (2008).
92. IEC: Enterprise-control system integration. Part 1. Models and terminology. Part 2: Model objectattributes: ISO/IEC FDIS Standard. Geneva, Switzerland, (2002).
93. Stewart, G.: Supply-chain operations reference model (SCOR): the first cross-industry framework for integrated supply-chain management. Logist. Inf. Manag. 10, 62–67 (1997).
94. Wand, Y., Weber, R.: An ontological model of an information system. Softw. Eng. IEEE Trans. 16, 1282–1292 (1990).
95. Lin, Y., Krogstie, J.: Semantic annotation of process models for facilitating process knowledge management. Int. J. Inf. Syst. Model. Des. IJISMD. 1, 45–67 (2010).
96. Gašević, D., Devedžić, V.: Petri net ontology. Knowl.-Based Syst. 19, 220–234 (2006).
97. Wongthongtham, P., Chang, E., Dillon, T., Sommerville, I.: Development of a Software Engineering Ontology for Multisite Software Development. Knowl. Data Eng. IEEE Trans. 21, 1205–1217 (2009).
98. Ghidini, C., Rospocher, M., Serafini, L.: A formalisation of BPMN in description logics. Technical report TR 2008-06-004, FBK-irst (2008).
99. Bunge, M.: Treatise on basic philosophy: Vol. 3: Ontology I: The furniture of the world. Reidel Boston. 22, (1977).

100. Billington, J., Christensen, S., Hee, K., Kindler, E., Kummer, O., Petrucci, L., Post, R., Stehno, C., Weber, M.: The Petri Net Markup Language: Concepts, Technology, and Tools. In: Aalst, W.P. and Best, E. (eds.) Applications and Theory of Petri Nets 2003. pp. 483–505. Springer Berlin Heidelberg (2003).
101. Bause, F., Kemper, P., Kritznger, P.: Abstract Petri Net Notation. Petri Nate Newsl. 9–27 (1995).
102. Sommerville, I.: Software Engineering. International computer science series. Addison Wesley, May (2004).
103. Dupuis, R.: Software Engineering Body of Knowledge. (2004).
104. Uren, V., Cimiano, P., Iria, J., Handschuh, S., Vargas-Vera, M., Motta, E., Ciravegna, F.: Semantic annotation for knowledge management: Requirements and a survey of the state of the art. Web Semant. Sci. Serv. Agents World Wide Web. 4, 14–28 (2006).
105. Haas, H., Brown, A.: Web services glossary. W3C Work. Group Note 11 Febr. 2004. (2004).
106. Kopecky, J., Vitvar, T., Bournez, C., Farrell, J.: Sawsdl: Semantic annotations for wsdl and xml schema. Internet Comput. IEEE. 11, 60–67 (2007).
107. W3C: OWL-S: Semantic Markup for Web Services, http://www.w3.org/Submission/OWL-S/.
108. Patil, A.A., Oundhakar, S.A., Sheth, A.P., Verma, K.: Meteor-s web service annotation framework. Proceedings of the 13th international conference on World Wide Web. pp. 553–562 (2004).
109. Manning, C.D., Schütze, H.: Foundations of statistical natural language processing. MIT press (1999).
110. Vargas-Vera, M., Motta, E., Domingue, J., Lanzoni, M., Stutt, A., Ciravegna, F.: MnM: Ontology driven semi-automatic and automatic support for semantic markup. Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web. pp. 379–391. Springer (2002).
111. Popov, B., Kiryakov, A., Kirilov, A., Manov, D., Ognyanoff, D., Goranov, M.: KIM–semantic annotation platform. The Semantic Web-ISWC 2003. pp. 834–849. Springer (2003).
112. Ma, Y., Lévy, F., Ghimire, S.: Reasoning with annotations of texts. 24th International FLAIRS Conference (FLAIRS11): Track AI, Cognitive Semantics, Computational Linguistics and Logics (2011, to appear) (2011).
113. Boudjlida, N., Panetto, H., Baïna, S., Diamantini, C., Krogstie, J., Lin, Y., Sarraipa, J., Zouggar, N., Hahn, A., Delgado, M.: DTG4. 2: Experimental Semantic Enrichment of Enterprise Models for Interoperability and its Practical Impact. (2007).
114. Bergamaschi, S., Beneventano, D., Corni, A., Kazazi, E., Orsini, M., Po, L., Sorrentino, S.: The Open Source release of the MOMIS Data Integration System. Proc. of the Nineteenth Italian Symposium on Advanced Database Systems, SEBD. pp. 26–29 (2011).
115. Attene, M., Robbiano, F., Spagnuolo, M., Falcidieno, B.: Characterization of 3D shape parts for semantic annotation. Comput.-Aided Des. 41, 756–763 (2009).
116. Li, C.: Ontology-Driven Semantic Annotations for Multiple Engineering Viewpoints in Computer Aided Design, PhD Thesis (2012).
117. Di Francescomarino, C.: Semantic annotation of business process models, PhD Thesis (2011).
118. Yahia, E., Lezoche, M., Aubry, A., Panetto, H.: Semantics enactment for

interoperability assessment in Enterprise Information Systems. Annu. Rev. Control. 36, 101–117 (2012).

119. Sabou, M., Lopez, V., Motta, E., Uren, V.: Ontology selection: Ontology evaluation on the real semantic web. (2006).
120. Vrande\vcić, D.: Ontology evaluation. Springer (2009).
121. Basil, V.R., Turner, A.J.: Iterative enhancement: A practical technique for software development. Softw. Eng. IEEE Trans. 390–396 (1975).
122. MEGA: MEGA Java Creating a MEGA plug-in in Java. (2009).
123. Jena: The Syntax of Jena Rules, http://jena.apache.org/documentation/inference/.
124. Gouyon, D.: Contrôle par le produit des systèmes d'exécution de la production: apport des techniques de synthèse, http://tel.archives-ouvertes.fr/tel-00081851/, (2004).
125. Bouramoul, A., Kholladi, M.-K., Doan, B.-L.: How ontology can be used to improve semantic information retrieval: the AnimSe finder tool. Int. J. Comput. Appl. IJCA–ISSN. 0975–8887 (2011).
126. Narayanan, S., McIlraith, S.A.: Simulation, verification and automated composition of web services. Proceedings of the 11th international conference on World Wide Web. pp. 77–88 (2002).
127. Noy, N.F.: Semantic integration: a survey of ontology-based approaches. ACM Sigmod Rec. 33, 65–70 (2004).
128. McGuinness, D.L., Van Harmelen, F.: OWL web ontology language overview. W3C Recomm. 10, 10 (2004).
129. Kuba, M.: Automated trust negotiation in identity federations using OWL-based abduction of missing credentials. Internet Technology and Secured Transactions (ICITST), 2011 International Conference for. pp. 164–169 (2011).
130. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A semantic web rule language combining OWL and RuleML. W3C Memb. Submiss. 21, 79 (2004).
131. Friedman-Hill, E.: JESS in Action. Manning Greenwich, CT (2003).
132. Jena: Apache Jena, http://jena.apache.org/.
133. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical owl-dl reasoner. Web Semant. Sci. Serv. Agents World Wide Web. 5, 51–53 (2007).