Yun Lin

Semantic Annotation for Process Models:

Facilitating Process Knowledge Management via Semantic Interoperability

Department of Computer and Information Science
Norwegian University of Science and Technology
N-7491 Trondheim, Norway

NTNU
Norwegian University of Science and Technology

Thesis for the degree of Ph.D.

Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Computer and Information Science

ISBN 978-82-471-5157-0 (printed version)
ISBN 978-82-471-5160-0 (electronic version)
ISSN 1503-8181

Doctoral theses at NTNU, 2008:03

Printed by NTNU-trykk

To MY BELOVED HUSBAND HAO DING
AND OUR DAUGHTER SABRINA DING

Abstract

Business process models representing process knowledge about doing business are necessary for designing Information Systems (IS) solutions in enterprises. Interoperability of business process knowledge in legacy systems is crucial for enterprise systems interoperation and integration due to increased enterprise cooperation and business exchange. Many modern technologies and approaches are deployed to support business process interoperability either at the instance level or the protocol level, such as BPML, WSDL and SOAP. However, we argue that a holistic approach is necessary for semantic interoperability of business process models at the conceptual level when considering the process models as reusable process knowledge for other (new or integrated) IS solutions. This brings requirements to manage semantic heterogeneity of process knowledge in process models which are distributed across different enterprise systems. Semantic annotation is an approach to achieve semantic interoperability of heterogeneous resources. However, such an approach has usually been applied to enhance the semantics of unstructured and structured artifacts (e.g. textual resources [72] [49], and Web services [166] [201]).

The aim of the research is to introduce an ontology-based semantic annotation approach to enrich and reconcile semantics of process models — a kind of semi-structured artifact, for managing process knowledge. The approach brings together techniques in process modeling, ontology building, semantic matching, and Description Logic inference in order to provide a comprehensive semantic annotation framework. Furthermore, a prototype system that supports the process of ontology-based semantic annotation of heterogeneous process models is described. The applicational goal of our approach is to facilitate process knowledge management activities (e.g. discovery, reuse, and integration of process knowledge/models) by enhanced semantic interoperability.

A survey has been performed through identifying semantic heterogeneity in process modeling and investigating semantic technology from theoretical and practical views. Based on the results from the survey, a comprehensive semantic annotation framework has been developed, which provides a method to manage semantic heterogeneity of process models from the following perspectives. First, basic descriptions of process models (*profile annotation*); second, process modeling languages (*meta-model annotation*); third, contents of process models (*model annotation*) and finally intentions of process model owners (*goal annotation*). Applying the semantic annotation framework, an ontology-based annotation method has been elaborated, which results in two categories of research activity — ontology building and semantic mapping. In ontology building, we use Web Ontology Language (OWL), a Semantic Web technology, which can be used to model ontologies. GPO (General Process Ontology) comprising core concepts in most process modeling languages is proposed; domain concepts are classified in the corresponding categories of GPO as a domain ontology; design principles for building a goal ontology are introduced in order to serve the annotation of process models pragmatically. In semantic mapping, a set of mapping strategies are developed to conduct the annotation by considering the semantic relationships between model artifacts and ontology references and as well the semantic inference mechanism supported by OWL DL (Description Logic). The annotation method is finally formalized into a process semantic annotation model - PSAM.

The proposed approach has been implemented in a prototype annotation tool —

ProSEAT to facilitate the annotation process. Procedures of applying the semantic annotation approach with the tool are described through exemplar study. The annotation approach and the prototype tool are evaluated using a quality framework. Furthermore, the applicability of the annotation results is validated by going through a process knowledge management application. The Semantic Web Rule Language (SWRL) is applied in the application demonstration. We argue that the ontology-based annotation approach combined with the Semantic Web technology is a feasible approach to reconcile semantic heterogeneity in the process knowledge management. Limitations and future work are discussed after concluding this research work.

The contributions of this thesis are summarized as follows. First, a general process ontology is proposed for unifying process representations at a high level of abstraction. Second, a semantic annotation framework is introduced to describe process knowledge systematically. Third, ontology-based annotation methods are elaborated and formalized. Fourth, an annotation system, utilizing the developed formal methods, is designed and implemented. Fifth, a process knowledge management system is outlined as the platform for manipulating the annotation results. Moreover, applying results of the approach is demonstrated through a process model integration example.

Contents

List of Figures

List of Tables

Preface

This thesis is submitted to the Norwegian University of Science and Technology (NTNU) for the doctoral degree – Doctor of Philosophy (Ph.D.). The work has been carried out at the Information System Group (IS-group), Department of Computer and Information Science (IDI), under supervision of Professor Arne Sølvberg.

Acknowledgments

First and foremost, I would like to acknowledge my principle supervisor — Professor Arne Sølvberg. I am grateful that Arne became my supervisor and gave the guidance in conducting my doctoral research work. I thank Arne for his constant encouragement and precise advice all the way through. He helped me a lot to clarify the research issues, especially during the thesis edition phase. His professional working attitudes have also influenced me, from which I gain the benefits for my future carrier.

I am thankful to my second supervisor — Professor John Krogstie, especially for advising me on the research topics on process modeling and quality evaluation framework. He also provided me access to the resources such as process modeling languages and tools. I also thank Professor Mihhail Matskin, my third supervisor, who enlightened me with some ideas from the semantic Web services. I would like to thank Professor Arne Jørgen Berre for hosting my visit at SINTEF in Oslo and providing the working environment where I finished the implementation of the prototype.

I am lucky to have knowledgeable, friendly and helpful colleagues around during my work in IS-group. I especially acknowledge Darijus Strašunskas for the productive collaboration, inspiring and constructive discussions and the generous and sincere help in reviewing my papers as well as proof reading of the thesis. Thank Jennifer Sampson for exchanging study and life experiences with me when we shared the office for four years. I also thank her very much for the proof reading of the final version of the thesis. I thank Sari Hakkarainen for her constructive advice on the scientific writing. Many thanks go to Xiaomeng Su for the personal communication and encouragement. Thanks to Csaba Veres, Peep Küngas, Raimundas Matulevičius, Lillian Hella, Rune Molden for all the support, talks and discussions.

Thanks to the members of the INTEROP project for providing a platform for the development and exchange of research issues on ontology and semantic interoperability. The exemplars in this work were adapted from the INTEROP project.

I appreciate all the people, past and present, at IDI for providing the needed working environment and atmosphere. Warm thanks to my friends in Norway and abroad for the joy I shared with them and the help received from them.

My tremendous gratitude goes to my family. I am greatly indebted to my parents for their love, support and help anytime anywhere. I thank my sister and brother-in-law for sharing cheerful time together in Norway. I also thank my parents-in-low for helping look after Sabrina during the final stage of this work. I am grateful to my husband Hao Ding for his enormous love, support, understanding as well as thoughtful discussions. Thanks to my lovely daughter Sabrina Ding for bringing me happiness and enriching my life.

Yun Lin
February 24, 2008

Part I

Background and Context

Chapter 1

Introduction

Business process models depict process knowledge of enterprises and they are reusable knowledge assets in building Information System (IS) solutions for the business. When different enterprises collaborate in business by integrating their process for a new IS solution, there is a need for methods and technologies to manage reusable process knowledge in the distributed process models. Semantic interoperability is an important and difficult issue caused by the heterogeneity of various models. This thesis addresses semantic annotation for improving semantic interoperability of heterogeneous process models to manage process knowledge from different enterprises.

In this chapter we explain our initial interests of the topic and state the research problem. Then we specify research questions and objectives of the work and describe our approach under a set of research methods. Finally, we present our contributions and outline the structure of the thesis.

1.1 Motivation

IS solutions applied in enterprises are usually expressed in certain documents addressing different features on many levels of abstraction. For each feature and level there is a choice of competing languages employing different system specification models (i.e., different modeling concepts). In addition, every IS solution is developed in its own context, and has to satisfy the intentions of the "system's owner". Yet, similar IS problems may have different solutions, because the contexts differ as do the intentions and goals of the systems' owners.

On the other hand, different solutions for similar systems may have many similarities, so that components of one system solution may be reused (with possible modifications) for another system. In such a case, the reusable solutions can be considered as knowledge addressing the problem of the system. The question is to recognize the candidates for reuse, in spite of the different languages employed, the different contexts and the different goals of the system owners.

As one type of IS solution models play essential roles in the modern system development method, such as MDD (Model Driven Development) and MDA (Model Driven Architecture) [123]. With the method of MDD, business process models representing organizational knowledge consist of potentially reusable process model fragments for

other similar business system development; the logical business process models on the conceptual level can be transformed into physical models for implementation (e.g. services). Hence, facilitating reuse and management of business process models becomes critical.

Business process models built as solutions for different enterprises are as well various in process modeling languages, process context and intentions of the systems' owners. Different levels of interoperability can be identified based on different types of information heterogeneity and system heterogeneity. Our research objects are business process models at conceptual level, mainly concerning the representation of process semantics. We therefore focus on *semantic interoperability*. Semantic interoperability is about how to achieve the mutual understanding of interchanged data [170] in spite of semantic heterogeneity.

Ontology-based semantic annotation is usually considered as a technique to achieve semantic interoperability by introducing common understanding and standardization. Semantic annotation has been developed and applied on both unstructured and structured artifacts to improve semantic interoperability (e.g. textual resources and Web services). The main applications based on semantic annotation are semantics-based information retrieval and automatic semantics-based discovery of services. According to our knowledge, few efforts on annotating semi-structured artifact (e.g. enterprise/business process models) were found when we started this work. The importance of semantic interoperability of enterprise process models becomes more and more obvious as pervasive development of applications for business process integration under a SOA (Service Oriented Architecture), where process models are vital assets of system specifications and interoperating objects in integration systems. On the other hand, semantic issues are leveraged and the usage of Semantic Web technology is gradually gaining attention in the enterprise modeling domain due to the advances of machine intelligence on semantics and inference. We follow such a trend and contribute our efforts on the research activities of building a descriptive methodology and implement a tool for the semantic annotation of heterogeneous business process models in order to facilitate managing process knowledge from different organizations.

1.2 Problem Statement and Research Questions

The following scenario exemplifies the problem to be solved in this thesis. A travel agency wants to build a new travel booking service. In the travel domain, there are legacy models about business of travel agency made by different organizations, e.g., models for a flight ticket booking system and models for a hotel reservation system. For the new system it is not necessary to build models for the travel booking service from the scratch, but try to reuse available models. In this case, we assume that the business process models from legacy systems are at conceptual level and can be used as business process knowledge. In order to reuse the knowledge, it requires a centralized management system to manipulate those heterogeneous models which were built by different organizations. The following semantic interoperability problems may arise:

- Since models are created in different modeling languages, a same business phenomenon is represented variously in different models. For example, a process

model of flight tickets booking is built in ActionWorkflow [103] and one hotel reservation process is modeled in CPR (Core Plan Representation) [135]. There would be interoperability problem when exchanging information about `Agent` or `Actor` between two models because they are using different modeling languages. In this case, `Agent` in ActionWorkflow models is semantically equal to `Actor` in CPR models. However, it is difficult for the user and the machine to identify they are the same without the support of the semantic mappings between those two modeling elements.

- Terminology is used differently in different models. For instance, "Client" is used in the flight ticket booking model and "Customer" is used in the hotel reservation model, although the two terms represent the same concept in this case. If only keyword based search is applied, the concept Client represented by Agents in ActionWorkflow model and the concept Customer represented by Actors in CPR model can not be recognized as the same concept by machine.

- Conceptualization mismatches include different classifications, aggregations, attribute assignments and value types. For example in this case, City is modeled in a class containing city name, graphic location and tourism information. It is used as the resource of a hotel reservation model. City is also one of attributes of Class Airport as input of a flight ticket booking process. Though the same term is used, the ways of representing the semantics are different. When reusing or integrating the models by the third party, incomplete or redundant semantics might be applied without knowing the differences.

In order to cope with semantic interoperability problems, agreed semantics should be referenced to annotate the heterogeneous representations in process modeling languages and business process model contents in a human and machine understandable manner. An ontology is considered a kind of agreement on a domain representation. Hence, ontologies of process modeling languages and modeling domains must be developed. The Semantic Web technologies as the emerging technological advances in the manipulation of ontologies, have made several new possibilities and challenges apparent.

Hence the main research question that the thesis attempts to answer: *How can semantic interoperability of process models be improved by using semantic technologies such as ontologies in the process knowledge management applications?* More specifically,

- **RQ1.** What kind of semantic interoperability problems exist in process knowledge management?

- **RQ2.** What kind of ontologies are required for process knowledge management and how to represent them?

- **RQ3.** What metadata are essential for process model interoperability and how are they defined concerning reference ontologies for the reconciliation of the heterogeneous semantics of process models?

- **RQ4.** How can Semantic Web technology to be incorporated in a tool using the proposed approach?

- **RQ5.** How can we use the proposed approach to facilitate process knowledge management?

1.3 Objectives

Corresponding to the main research question, the overall objective of this thesis is to propose a semantic annotation approach for dealing with semantic heterogeneity of process models in order to facilitate process knowledge management activities (such as recognizing and reusing IS solutions of process models) via semantic interoperability. It is decomposed into the following four sub-objectives which are to be achieved step by step during the development of the work.

1. To investigate semantic heterogeneity issues in business process modeling.

2. To explore a comprehensive annotation approach to deal with heterogeneous semantics of process knowledge with referenced ontology.

3. To develop an annotation tool to implement the approach by applying Semantic Web technologies.

4. To evaluate quality and use feasibility of the proposed approach and tool in supporting process knowledge management activities.

1.4 Approach and Scope

We focus on business process models at a conceptual level. They are available process knowledge resources which need to be managed for further reuse. Therefore, heterogeneous process models should be reconciled and process knowledge conveyed by the models should be explicitly expressed.

The approach taken is (1) to express the process properties of each business process model in a common annotation system, (2) to express model context in ontologies that may be compared to each other, one ontology for each context, and (3) to map the intentions of the systems' owners to goal structures that may be compared to each other. These three elements constitute the means to profile a business process model, and provides the basis for building a library of modeling assets, and for recognizing similar model fragments to make them easier available for reuse.

Ontologies aid to share knowledge on the basis of the assumption that there is a single reality and the sharing is a matter of aligning the way different people or systems think about it [70]. Annotating models with the agreed ontologies provides a way of reconciling heterogeneity and reaching consensus on the semantics of terms and concepts. To facilitate management and reuse of the reconciled models, the semantics of the process models should be interpretable and inferable by machines as modeling assets. The approach is hereby deployed having the two concerns: 1) reconciling process knowledge representation of heterogeneous models, and 2) explicating process knowledge in a machine-interpretable manner.

1.4.1 Semantic reconciliation of business process models

The central topic of the research is within information system modeling area. Thus, the working basis is modeling theory. We distinguish semantic heterogeneity into two levels: *meta-model* and *model*, which correspond to the M2 and M1 layers defined in MOF [124]. Although there is semantic heterogeneity on the M0 layer (instance level models), we do not take instance level models as reusable knowledge in this research. Therefore, ontologies related to meta-model and model levels are required for the annotation in this approach. For *meta-model level*, an ontology should supply common terminology and conceptualization of process modeling. There is no mature process ontology as we know, though there are many ongoing activities relevant to this topic [12] [139] [140]. Based on the investigation and analysis of numbers of existing process ontologies and process meta-models, we propose a process ontology which consists of most essential concepts for process modeling languages. For *model level*, an ontology should contain standardized terms and definitions of concepts and relationships about a certain domain. Domain-specific ontologies should be created by domain experts. Such ontologies can be built based on certain domain standards or reference models. Besides, to facilitate process knowledge management, a set of profile metadata is required to describe process models as products. Furthermore, process models can be pragmatically discovered based on intentional knowledge by goals of process. To explicate such knowledge, intentional concepts are represented in a goal ontology. Goal concepts are used to annotate intentional usage of process models. In our approach, all those ontology references and profile information are annotated to the original models through a set of metadata. Metadata can aid in the identification, discovery, assessment, and management of the described information-bearing entities [28]. In this way, annotation does not intervene the original semantic representation and it is stored separately from the original models. One original model can have several versions of annotation serving different purposes. As an outcome of the research work, a semantic annotation framework is developed to systematically organize the above four perspectives, i.e. profile annotation, meta-model annotation, model annotation and goal annotation.

Annotating models with ontologies is a procedure to establish certain mapping between objects in models and in ontologies. Two objects from the different sets — 'annotater' and 'annotatee' have not only a simply one-to-one correspondence but more comprehensive semantic relationships. We define a set of mapping strategies and rules to guide users to conduct the annotation. An annotation model — PSAM (Process Semantic Annotation Model) is formalized following the proposed framework and the mapping method. The PSAM model provides a basis for the implementation. An annotation system is proposed and implemented as a prototype in order to prove the feasibility of the proposal.

1.4.2 Machine-interpretable process knowledge

From a technical aspect, semantics of ontologies and PSAM models should be machine-interpretable. Emerging Semantic Web technology can encode semantics of Web resources in a machine-readable form. The OWL Web Ontology Language is a language for defining and instantiating Web ontologies. An OWL ontology may include descriptions of classes, properties and their instances. Given such an ontology, the OWL formal

semantics specifies how to derive its logical consequences, i.e. facts not literally present in the ontology, but entailed by the semantics [195]. As a result of this research, the management of distributed process models can be operated through the Web within or across applications and systems. It is reasonable and feasible in that Internet and Intranet are so pervasive in enterprises and the Web provides a good platform to share distributed information. Moreover, the Semantic Web technology is already applied in cooperative business and industries for the interoperability as a technical standard. OWL is consequently chosen in this thesis to define ontologies and annotation models in order to make use of the Semantic Web technology.

Semantics of different models are reconciled through annotation. The annotation information is represented in a machine-interpretable way for a common platform – the Semantic Web. Process knowledge is exposed by annotation and it is organized in a semantic annotation model. Then the process knowledge can be inferred based on the definitions of ontologies. Such knowledge will enable applications to better understand process models and to use them more intelligently, which is also the goal of the Semantic Web research [52]. The applicability of the proposal is validated through a walkthrough scenario of a process management application.

1.5 Research Method

The research methods applied in this work consist of a descriptive analysis phase, a normative development, an implementation phase and an evaluation phase. All together the phases include the following steps.

1. The *survey of the state-of-the-art* step includes investigation and analysis of the semantic representations of process models, semantic interoperability and process ontologies, semantic annotation methods, and knowledge management.

2. The *analysis of requirements* step includes an inventory of the problems about the semantic heterogeneity of process models with regard to process knowledge management and an analysis of raised requirements on semantic annotation tool.

3. The *development of the approach* step includes the specification of the semantic annotation framework, the design of the annotation approach, the efforts to generalize the ontological specifications for process modeling and goal modeling, and the definitions of mapping rules involved in the annotation process.

4. The *prototype application* step includes development and implementation of the prototypical environment for the process models and knowledge management based on the results of the previous steps.

5. The *quality evaluation and applicability analysis* step includes the evaluation of the proposed framework and approach using a quality framework based on the observations from a walkthrough scenario of a process knowledge management application.

1.6 Major Contributions

Following the above research method and steps, the research has been deployed and it has resulted in the following major contributions of this thesis:

1. A General Process Ontology (GPO) is the result of our investigation and analysis of process ontologies and business process modeling languages, which is an effort towards the unified process ontology for the interoperability of process models.

2. A basic semantic annotation framework is proposed to enrich semantics of process models for process knowledge management.

3. An ontology-based annotation method including a semantic annotation model and rules are elaborated to guide users to apply the framework for the annotation.

4. The extended semantic annotation framework is a goal annotation method with a goal ontology representation and semi-automatic goal annotation algorithms.

5. A prototype of the annotation tool applying the Semantic Web technology is implemented following the proposed framework and method.

6. A process knowledge management system integrating the semantic annotation approach is outlined to validate the applicability of the framework and the method.

1.7 Thesis Outline

In this introduction chapter, we has explained the motivation of the work, described the problems, listed the objectives of this work, provided an overview of the approach, and presented the research method and our major contributions. The rest of the thesis is organized following the research phases described in the research method in section 1.5. Chapter 2 and 3 provide a background, context and relevant work survey for this research. Chapter 4 and 5 present our theoretical and methodology contributions, while chapter 6 describes the implementational contributions of the work. Exemplar studies are deployed to demonstrate the proposed approach in chapter 7. Chapter 8 evaluates the quality of the method, and Chapter 9 validates the applicability of the work. Finally, conclusions and further work are discussed in chapter 10.

Chapter 2 - Problem Settings provides basic concepts, theory and technology about information modeling, business process modeling, semantic interoperability, Semantic Web and process knowledge management. Those issues compose the context of this research work.

Chapter 3 - State of the Art discusses a number of existing business process modeling languages and process ontologies to investigate modeling constructs and ontological concepts corresponding to process modeling perspectives. The survey of semantic annotation methods and tools shows most of work has been done on the semantic annotations of unstructural and structural information, e.g. Web pages or Web services. Comprehensive approaches for the semantic annotation of semi-structured information, e.g. business process models, from meta-model and model levels are also demanded and some efforts have been launched.

Chapter 4 - Semantic Annotation Framework for Process Models introduces our basic semantic annotation framework consisting of profile annotation, meta-model annotation and model annotation. Metadata are defined for profile annotation. A general process ontology is presented as ontological basis for meta-model annotation to cope with various modeling languages applied in different IS solutions. In model annotation, domain ontologies are referenced by model contents to represent systems' context. A semantic annotation model formalizes the framework with a set of semantic mapping rules.

Chapter 5 - Extension Semantic Annotation — Goal Annotation elaborates a goal ontology representation and goal annotation algorithms. Such extension can be used to compare the IS solutions brought by the process models of systems through specifying the intentions of the systems' owners.

Chapter 6 - Pro-SEAT (Process SEmantic Annotation Tool) presents the implementation of the prototype of a semantic annotation tool. System modules provide an overview of the architecture of the prototype. Data structures depict the logical design of the proposed approach. Goal algorithms are implemented to automate the goal annotation. Functionality and graphical user interfaces are illustrated to elucidate the interactions between users and the tool prototype.

Chapter 7 - Exemplar Studies and Application System exemplifies the procedure of the approach and the usage of the prototype through two exemplar cases from different business organizations. A process knowledge management system is also outlined by integrating the semantic annotation approach.

Chapter 8 - Quality Evaluation of the Method evaluates the quality of the work using a quality framework. The quality evaluation of our annotation approach is discussed according to the quality categories defined in a quality framework.

Chapter 9 - Validation of Applicability validates the applicability of the semantic annotation approach in a walkthrough scenario of an integration application of process models. The integration is deployed based on the semantic annotation results of models in the exemplar studies.

Chapter 10 - Conclusions and Future Work summarizes the work, and outlines the future work to point out the possible improvements and the interesting directions of further research on semantic interoperability of business process management.

Chapter 2

Problem Setting

In this chapter, we establish basic concepts, theoretical and technical background and the context of this research work. Since we work in the information modeling area, we start the study with theories and methodologies in information system engineering discipline. Concepts, standards and relevant issues about semantic interoperability and semantic annotation are then discussed in this chapter. The research target – *business process model* and its application – *business process knowledge management* are also discussed concerning the context of the work.

2.1 Models in Information System Engineering

A *model* of a system is defined by OMG[1] as a description or specification of that system and its environment for certain purpose. In the conventional system development lifecycle, models are mainly used in system analysis and design phases and they are not reusable resources after a system is developed. In the modern methods of information system development, models play an important role as reusable IS solutions.

MDA[2] is an approach to system development, which increases the power of models in that work. MDA provides a means for using models to direct the course of understanding, design, construction, deployment, operation, maintenance and modification [117]. MDA is initiated with the idea of separating specifications of the operation of a system from details of the way that system uses the capabilities of its platform. The three primary goals of MDA are portability, interoperability and reusability. The objective of MDA is to provide an open, vendor-neutral approach to the challenge of business and technology change [123] by separating different concerns. Three different viewpoint models are distinguished as Computation Independent Model (CIM), Platform Independent Model (PIM) and Platform Specific Model (PSM) [117].

- CIM — A *computation independent model* is a view of a system from the computation independent viewpoint. A CIM is sometimes called a domain model and a vocabulary that is familiar to the practitioners of the domain in question is used in its specification.

[1] Object Management Group, http://www.omg.org/
[2] Model Driven Architecture, http://www.omg.org/mda/

- PIM — A *platform independent model* is a view of a system from the platform independent viewpoint. A PIM exhibits a specified degree of platform independence so as to be suitable for use with a number of different platforms of similar type.

- PSM — A *platform specific model* is a view of a system from the platform specific viewpoint. A PSM combines the specifications in the PIM with the details that specify how that system uses a particular type of platform.

In our research, we adopt the vision of MDA. Model plays an essential role in the information system development lifecycle and it should achieve system products portable, reusable and interoperable. Model itself can be treated as product and representation of knowledge. Therefore, model should also be portable, reusable and interoperable.

Model is also found central and important in the Zachman framework [171], where models are classified based on different roles involved in the system development and enterprise's functionings (see Figure 2.1). In this architecture framework, there are three model categories — enterprise model, system model and technology model, distinguished from different perspectives of the different participants [214]. Each model category contains six basic models regarding different types of descriptions oriented to different aspects of the object being described. The six types of descriptions are entities, functions, locations, people, times and motivations. In the matrix representation of the framework, enterprise model is at the conceptual level and it includes semantic model, business process model, business logistics system, workflow model, master schedule and business plan.

In our work, we mainly focus on the business process model, which indicates that our research focus and scope is the functional descriptions of enterprise business at the conceptual level. While, corresponding to the MDA definitions, our research subject is CIM.

2.2 Modeling Basis

2.2.1 Semiotic triangle

Modeling at the conceptual level is an activity of representing phenomena of the real world in a model. It complies with the famous semiotic triangle adapted from Ogden and Richards' triangle of meaning [121], i.e. the relationships between a *concept*, a *referent* and a *sign*. Concepts are mental things, words of mind. A referent is a thing in reality to which a concept refers. Signs are expressions, symbols or labels to signify concepts in some language. Modeling aims to conceptualize referents by employing modeling signs in certain languages. Obviously models could be quite different from each other due to what (*referent*) to model, how to model (*concept*) and with what (*sign*) to model. The three elements interact on each other from a semantic representation perspective. The capability of signs' expressiveness can determine what semantics of referents can be represented, also how semantics are conceptualized.

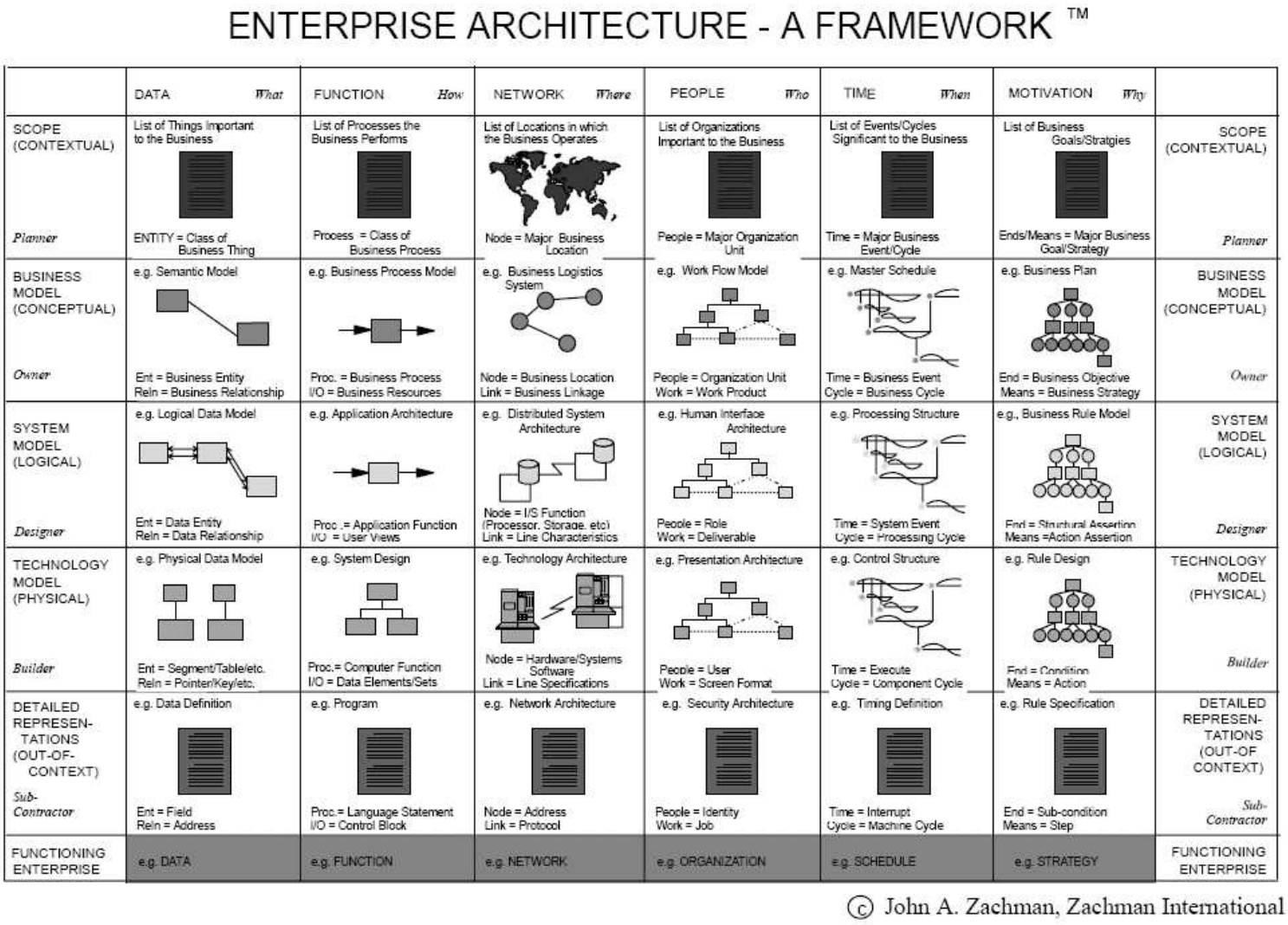


Figure 2.1: Zachman Enterprise Architecture Framework [189] [171] [214]

2.2.2 Modeling language, meta-model and model semantics

Any model must be built in a certain modeling language. A modeling language is any artificial language that can be used to express information or knowledge or systems in a structure that is defined by a consistent set of rules. A modeling language is used to represents concepts and relationships and other phenomena with a set of graphical notations or symbols. Such a modeling language is often called graphical modeling language. ER [15], UML [125], BPMN [12], EPC [160] are examples of graphical modeling languages. A modeling language can also be a set of standardized textual keywords or markup language accompanied by parameters, for instance, OWL [195], BPML [11], BPEL4WS [116], and EPML [105]. Graphical modeling languages usually facilitate readability of models whilst textual modeling languages enable models machine-interpretable and tool-interchangeable.

Interpretations of the meaning of modeling components in a modeling language are generally defined in a *meta-model*. A modeling components such as a graphical notation or a symbol and a textual keyword or a markup label is called a *meta-model element* or a *modeling construct* in our work. Meta-model elements defined in a meta-model are the building bricks of a model. A meta-model is also a model of a domain of interest and it is an instance of a meta-meta-model. A model in a certain modeling language is the instance of the meta-model of this modeling language. A model having all the instances of user objects is the instance of model. Instantiated model, model, meta-model and meta-meta-model respectively correspond to the M0, M1, M2 and M3 layers of OMG's four layer meta-data architectures [124].

In this thesis, meta-meta-model is beyond our research focus. While, instantiated model at the M0 layer is too specific to be reused. We therefore mainly focus on M1 and M2 layers, i.e. model and meta-model.

Common uses of meta-models are as follows [207]:

- As a schema for semantic data that needs to be exchanged or stored.
- As a language that supports a particular method or process.
- As a language to express additional semantics of existing information.

The three usages of meta-models are all employed in the thesis. The meta-model of a business process model supports process modeling. It is also stored as a schema associated with a model file. Facilitating the exchange of different process models is one of our research goals so that meta-model is one of our research objects. Establishing a semantic annotation method itself is a process of creating a meta-model to specify the additional semantics of an existing model.

The term *semantics* in linguistics means the study or science of meaning in language, or the study of relationships between signs and symbols and what they represent. It also indicates the meaning or the interpretation of a word, sentence, or other language form [3]. Model semantics is hereby the meaning or the interpretation of a model. Model semantics are represented by *model elements*. Model elements are the components of a model. Interpretation of model elements usually depends on the context of the system which the models represent the solutions for. Besides, model elements are composed of modeling constructs, and the semantics of modeling constructs are defined in a

meta-model of the modeling language. Thus, we conclude that model semantics are interpreted according to the system's context and the modeling language applied in the solution.

2.2.3 Ontology-driven and domain-specific

Modeling at the conceptual level trends to be seemingly independent of the computer world in the development of modeling methodologies. Researchers turn to philosophy and linguistics to try to find the inherent principles in the concepts of real world domains. And the boundary between the conceptual modeling and knowledge modeling becomes gradually fuzzy.

For example, Ontology-Driven Conceptual Modeling draws fundamental notions from formal ontology and establishes a minimal top-level ontology to drive conceptual modeling [47]. Thesaurus Conceptual Model uses the Knowledge/Data Model (KDM) [137] that incorporates an object-oriented view of data, together with knowledge regarding its usage [68]. Ontologies are used for the development of conceptual schemas of information systems by pruning irrelevant concepts in the ontologies [18].

On the other hand, domain model framework and meta-model make reuse and flexibility of models possible. A software system can never be finished, new and changed domain concepts will always be appearing, forcing continuous rebuilding, testing and redeployment of systems [8]. The main idea of the approach proposed in [8] is the removal of domain concepts from concrete software and database models, into standardized vocabularies and libraries of domain concept models. Re-engineering software and database is done using a generic reference object model (ROM) system architecture in [8]. Reference models are used in [160] to combine "formal driven" and "content driven" approaches in a new way to develop information systems. The formal driven approach aims to develop and implementing a technical correct running system. The goal of the content driven approach is developing and implementing an organizational correct running system. Goals of content and technology are concerned in reference models, which are regarded as "blue prints" for business engineering and can be used to model and optimize business processes. Domain-specific methods implemented with metaCASE technology [67] use meta-modeling languages to develop a domain meta-model mapped to combinations of components in order to generate a product.

All these approaches address the process of modeling, and the ideas can be analogically applied on the results of modeling — models. In an ontology-based semantic annotation method, the utility of an ontology and domain knowledge is applied after modeling. The additional semantic treatment is still necessary for the exchange or reuse of models across different organizations, because a modeling process usually is not centralized and local ontologies and local domain/reference models applied in the modeling are still various from different organizations. In this work, consensual ontologies and domain reference models are therefore needed to be determined in an annotation phase based on the requirements on interoperability and applications.

2.3 Information Systems and Semantic Web

As the development of the Web, more and more Information Systems start to be implemented using Web technology. Some Web technologies, e.g. Semantic Web techologies, are introduced into Information Systems development. On the other hand, the Web applications also benefit from traditional Information Systems theories and practices. COEUR-SW (Concepts On Enhancing, Understanding and Representing the Semantics on the Web) triangle [170] discloses how an Information Systems on the Semantic Web comply with the semiotic triangle [121]. COEUR-SW triangle is developed based on the semiotic triangle in the IS group at IDI[3]. In the COEUR-SW triangle (see Figure 2.2), a concept in a Universe of Thought (UoT) is related to an uttered symbol in a Universe of Language (UoL), the symbol is related to a referent in a Universe of Structure (UoS), and the referent is related back to the concept. The concept, the symbol and the referents are related to a context in the Universe of Discourse (UoD). The COEUR-SW program approaches the UoD from three interrelated angles and thus seeks to capture the heart of the Semantic Web. Domain modeling is used in order to capture the knowledge within a UoT into a man/machine understandable theory in a UoS. Ontology Mapping is used in order to capture the utterances in a UoL into man/machine-retrievable knowledge. Metadata analysis is used in order to capture the theories in a UoS into enriched man/machine generative utterances in a UoL.

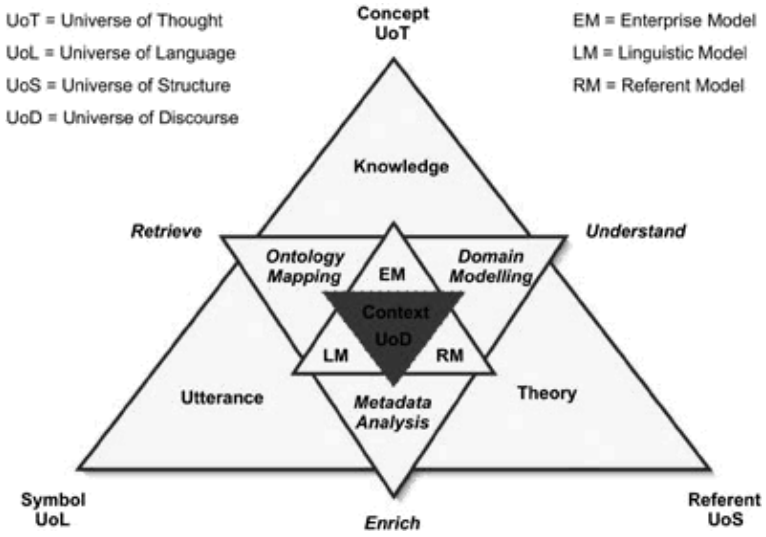


Figure 2.2: COEUR-SW triangle

COEUR-SW triangle defines three roles of the Web in Information Systems [170]:

- the role of the dominant medium for information dissemination;
- the role of the tool for information compilation;
- the role of the evolving information repository.

[3]Dept. of Computer and Information Science at Norwegian University of Science and Technology

In the context of our research, process models as process knowledge resources can be disseminated through the Web. The consequent question is how the Web compiles process models. This issue relates to the knowledge categorization and semantic interpretation issues. The Semantic Web supplies some standards such as RDF [199], OWL [195] to support the semantic interpretation. The knowledge representation of process models needs to be transformed into those Semantic Web standards. The Web can be viewed as a large distributed repository for the process models. However, distributed models are originally from different autonomous systems and stored in various schemas. Technologies facilitating interoperability of heterogeneous models such as ontology and semantic annotation, are required when organizing the knowledge in such a repository.

2.3.1 Ontology in information systems

Ontology

Ontology is "an explicit representation of conceptualization" [42]. According to the investigation in [13], ontology locates two disjoint literatures with virtually no personnel in common: the world of formal ontology specification e.g. [36] and the world of ontologies for language-related AI (Artificial Intelligence) tasks e.g. [113]. It is consequently found that people use the word "ontology" to mean different things, e.g. glossaries & data dictionaries, thesaurus & taxonomies, schemas & data models, and formal ontologies & inference. Ontologies applied in this work are defined as formal ontologies. A formal ontology defines the basic terms and their relationships comprising the vocabulary of an application domain and the axioms for constraining the relationships among terms [194]. A formal ontology is usually expressed in an ontology representation language, e.g. OWL.

A formal ontology consists of a set of specifications of representational vocabularies (*concepts*) for a shared universe of discourse, which may include definitions of classes, relations, functions, and other objects [43]. In AI applications and research, ontology is often used as a means of achieving consistent communication between agents in multi-agent systems [134]. Hence, ontology contains agreed-upon definitions in the form of human readable text and machine-enforceable, declarative constraints (agent readable format) on their well-formed use [41]. Besides, ontologies consist of a set of inference rules from which machines can make logical conclusions [1].

In [45], different kinds of ontologies are classified based on the level of generality as follows:

- *Top-level ontologies* describe very general concepts like space, time, matter, object, event and action. They are independent of a particular problem or domain. Top-level ontologies in some literature are also called upper-level ontologies.
- *Domain ontologies* describe the vocabulary related to a generic domain (like medicine, or automobiles).
- *Task ontologies* describe generic tasks or activities (like diagnosis or selling).
- *Application ontologies* describe concepts depending both on a particular domain and task.

The concepts in domain ontologies and task ontologies are specialized from the ones in the top-level ontology. Application ontologies are often specializations of both domain ontologies and task ontologies. Such classification can be reflected into the four layer meta-data architecture mentioned previously, i.e. top-level ontologies are at M2 and domain and task ontologies are at M1 and application ontologies are at M0. Domain, task and application ontologies about a certain domain usually construct the general context of the systems in that domain.

Three main uses of ontology are identified in [45]: 1) For communication between implemented computational systems, between humans, between humans and implemented computational systems; 2) For computational inference, e.g. for internally representing and manipulating plans and planning information, and for analyzing the internal structures, algorithms, inputs and outputs of implemented systems in theoretical and conceptual terms; 3) For reuse (and organization) of knowledge e.g. structuring or organizing libraries or repositories of plans and planning and domain information. The beneficial applications of ontology could locate in the following areas [178]:

- **Semantic Web.** The Semantic Web relies heavily on formal ontologies that structure underlying data for the purpose of comprehensive and transportable machine understanding. Ontologies are used to define the proper meaning of data and metadata for the Semantic Web [173].

- **Knowledge Management.** The technology of the Semantic Web brings out the knowledge pieces oriented view of knowledge management. Ontologies enable intelligent push service, the integration of knowledge management and business process to support the vision of ubiquitous knowledge. For the applications of knowledge management, ontologies are employed to annotate unstructured information with semantic information, to integrate information and to generate user specific views that make knowledge access easier [180] [25].

- **Interoperability.** Interoperability is an important issue in the applications of integration and reusing existing systems. As inter-lingua, ontology provides a common and machine-interpretable format for data interchange [177] [188].

- **Information Retrieval.** Conventional information retrieval approaches suffer from problems of the inconsistency between the query and the vocabulary of the documents, which reduces recall of search. Ontologies help to decouple description and query vocabulary and increase retrieval performance [46].

- **Service Retrieval** As the development of the research activities and applications of Web services, locating online services is increasingly critical in many domains. Online services include software applications, software components, process models, or service organizations. The approaches of using ontologies in querying those services are exploited and evaluated to improve the retrieval precision [10] [182] [131].

Those areas are usually overlapping in an application. For example, knowledge management is conducted under a Semantic Web environment; information retrieval or services retrieval may be one of required services in knowledge management; the proper knowledge management facilitates interoperability.

Conceptual model vs. ontology

The term "conceptual model" appears contrastively to "logical model" and "physical model" to differentiate the levels of model abstraction. The enterprise models on the second row of the Zachman framework are conceptual models. A conceptual model is often taken as an abstract model and defined as a theoretical construct that represents phenomena in a certain problem domain, with a set of variables and a set of logical and quantitative relationships between them. Conceptual model, especially conceptual data schema – representing the structure perspective, is comparable with ontology because they share some modeling principles.

A comparison of conceptual data schema and ontology is made in [63]. They are similar because both consist of concepts, relations and rules[4]. There are two main disparities discussed in [63]:

1. Conceptual data schema is being preserved in off-time model diagrams; while, ontology typically is *sharable and exchangeable at run-time*, i.e. machine-processable semantics.

2. Unlike conceptual data schema that capture semantics for a *given application domain*, ontologies are supposed to capture semantics about real-world domains, independent from specific application needs, i.e. *"relatively" generic knowledge*. Therefore, *the generality (application-independency) of knowledge is a fundamental asset in ontology modeling, and that mostly distinguishes ontology from conceptual data schema*.

The two disparities just characterize the two essential aspects of ontologies as described in [35]: ontologies define *formal* semantics for information, consequently allowing information processing by a computer; ontologies define *real world* semantics, which makes it possible to link machine processable content with meaning for humans based on consensual terminologies.

Conventional conceptual models are still widely used in information systems engineering and play vital roles in enterprise modeling as we have discussed previously. Meanwhile, ontologies become key enabling technology for the Semantic Web. Moreover, ontology modeling can benefit from the existing conceptual modeling methodologies and tools [5] [19] [38] [104]. Legacy conceptual schema can be also mined and/or "ontologized" [63]. On the other hand, ontology and the Semantic Web technology are also found to be applied in enterprise modeling and applications [139] [140].

We are aware of common grounds and differences between the two disciplines in this work and devote our efforts to build the links between conceptual process models and process ontologies. The work results in a combination of the usage of ontologies and conceptual models. Business process models at the conceptual level created for a given application domain still serve information systems development as *Representation of systems and requirements*, *Vehicle for communication*, *Basis of design and implementation* and *Documentation and sense-making* [76]. Meanwhile the extensive exchange, integration and reuse of business process models between different organization can benefit from the Semantic Web technology through annotating models using ontologies.

[4]Rules in conceptual modeling are called "constraints", and they are axioms in ontology.

2.3.2 RML and OWL

Two modeling languages – RML and OWL – related to ontology modeling in this thesis, are introduced in this sub-section. RML (Referent Model Language) [168] [169] is a concept modeling language representing structural perspective [76]. OWL (Web Ontology Language) is a language for defining and instantiating Web ontologies for creating Semantic Web applications [195].

RML

RML is concentrated on describing the static structure of a model. It is targeted towards applications in areas of information management and heterogeneous organization of data. The formal basis of RML is the set theory. The phenomena in RML are modeled as collections (set) or atoms (an individual phenomenon). A set is represented by a list of elements, or by stating property that all elements must have to qualify as a member of the set. Four different types of concepts – individual concepts, class concepts, relation concepts and quantitive concepts, are identified in RML. From the four types, the modeling constructs of RML are derived: class concepts, individual concepts, attributes, binary relationships, cardinality of a relationship, hierarchical abstraction of classification (*instance of*), aggregation(*part of*), generalization (*is-a*), and association (*member of*). Refined semantics can also be specified in RML: generalization is specified into disjoint generalization (e.g. A person is either a man or a woman.) and overlapping generalization (e.g. A person may both be an artist and an athlete.); relation concepts are distinguished into individual relation concepts, class relation concepts, order concepts and operation concepts; binary relation concepts are depicted as many-to-many, many-to-one or one-to-one mappings; "any" ($\forall$) and "some" ($\exists$) value restrictions are specified as the coverage of a relation concept; properties can be elaborated into reflexivity, symmetry, transitivity and connexity. Part of notations of RML are displayed in Figure 2.3. Detailed specifications of RML are provided in [169].

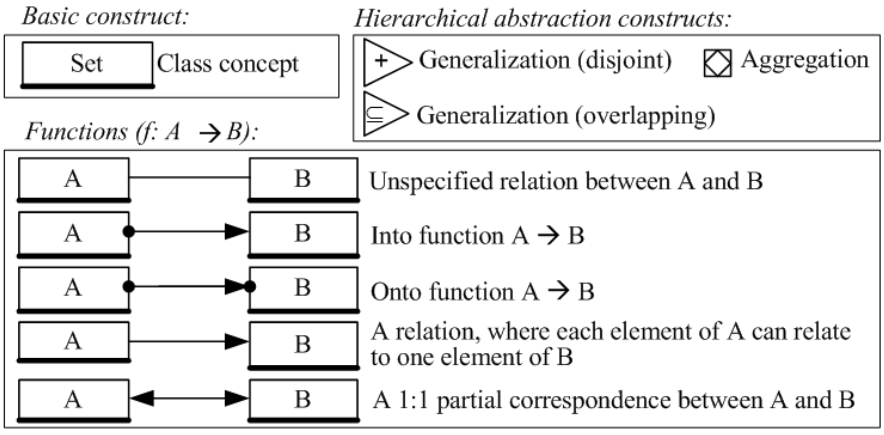


Figure 2.3: Graphical Notations of RML (limited to the constructs used in this thesis)

OWL

OWL is designed for use by applications that need to process the content of information instead of just presenting information to humans [197]. Building upon RDF and RDFS, OWL provides more machine-interpretable semantics by defining additional vocabulary along with formal semantics. OWL builds on Description Logics which is a restriction of First Order Logic. OWL provides three increasingly expressive sublanguages: OWL Lite, OWL DL (Description Logics), and OWL Full. Each of these sublanguages is an extension of its simpler predecessor. Compared to the other two sublanguages, OWL DL is often chosen as the ontology modeling language because of its capacity of fair semantics expressiveness and inference. Most available OWL reasoners support OWL DL, such as Pellet [96], RACER [61] and FaCT++ [162].

An OWL ontology usually consists of classes, properties, instances of classes, and relationships between these instances. Instances of classes in OWL are called individuals. OWL classes are described through "class descriptions", which can be combined into "class axioms" [196]. With class axioms, OWL Lite can represent generalization (`rdfs:subClassOf`), equality (`owl:equivalentClass`). Besides, OWL DL can specify classes as logical combinations of other classes (`owl:intersectionOf`, `owl:unionOf`, `owl:complementOf`), or as enumerations of specified objects (`owl:oneOf`) or as distinction of two classes (`owl:disjointWith`).

OWL distinguishes between two main categories of properties — object properties (`owl:ObjectProperty`) to link individuals to individuals and datatype properties (`owl:DatatypeProperty`) to link individuals to data values. Properties can be specified through domains (`rdfs:domain`) and ranges (`rdfs:range`). More property axioms are supported by OWL are sub-property (`rdfs:subPropertyOf`), equivalent property (`owl:equivalentProperty`), inverse property (`owl:inverseOf`), functional property (`owl:FunctionalProperty`), transitive property (`owl:TransitiveProperty`), symmetric property (`owl:SymmetricProperty`) and etc. An arbitrary number (zero or more) of values for a property is represented by cardinality constraints (`owl:maxCardinality`, `owl:minCardinality`, and `owl:cardinality`). Value constraints (`owl:allValuesFrom`, `owl:someValueFrom` and `owl:hasValue`) specify the quantifier restriction of a property. OWL individuals are specified through the class axiom `rdf:subClassOf`. The identity of individuals can be stated by referring to the same individual (`owl:sameAs`), or referring to different individuals (`owl:differentFrom`), or listing all different individuals (`owl:AllDifferent`).

Use of RML and OWL

In our work, RML is used during the descriptive analysis and the normative development phases to analyze concepts and relationships at meta-model level in our research. While, OWL DL is applied during the implementation and the experimental evaluation phases for serving as the three roles of: ontology modeling language, machine-interpretable semantics builder and first order logic inference basis. Meanwhile, RML is also used as a graphical modeling language to model ontologies in this work. Although Protégé-OWL provides a tool to model an ontology in a graph, the graphical notations are not as rich as ones defined in RML. Some axioms and constraints can not be visually displayed in the graphic models. When comparing the semantic expressiv-

ity of RML and OWL, it is not surprising to find a big overlapping of the modeling constructs between the two languages since both of them share a similar logic basis[5]. We do not intend to build one-to-one mapping between RML and OWL, but take the advantages of both as our research tools. We mainly use RML to visualize the concepts and relations of ontologies for human comprehension, and employ OWL to formalize ontology models for machine interpretation and reasoning.

2.4 Semantic Interoperability

Interoperability is the ability of two or more systems or components to exchange information and to use the information that has been exchanged [119]. Interoperability is a broadly used term, encompassing many of the issues impinging upon the effectiveness with which diverse information resources might fruitfully co-exists. The issues can be defined for different purpose, such as, semantics.

Semantic interoperability is the ability of two or more computer systems to exchange information and have the meaning of that information accurately and automatically interpreted by the receiving system. The main obstacle of semantic interoperability is semantic heterogeneity of the information to be exchanged. Common understanding of semantics and standardization of semantic representation are usually concerned as the solutions tackling the semantic heterogeneity to achieve semantic interoperability.

2.4.1 Semantic heterogeneity

Semantic heterogeneity is usually distinguished from syntactic heterogeneity and structural heterogeneity in the database community [37] [39] [86] [88] [89]. Syntactic heterogeneity is concerned with the heterogeneity of data formats. Standardizing data formats is taken as an approach to solve syntactic heterogeneity problems. For example, XML is used as a standard format for all forms of Web-accessible data. Structural heterogeneity is associated with different data models, data structures or schemas, e.g. relational and object-oriented database models. An example of the solutions for structural heterogeneity is that RDF based on XML syntax provides a unified way to structure information sources or object models for Web-based information exchange [100]. When two information sources are modeled in a same format by applying a same modeling methodology, there still might be semantic heterogeneity problem. Semantic heterogeneity can be identified according to the different types of conflicts [172]:

1. *Semantic conflicts.* Different modelers do not perceive exactly the same set of real world objects, but instead they visualize overlapping sets (included or intersecting sets). For example, a "Student" object class may appear in one schema, while a more restrictive "CS-Student" object class (grouping students majoring in computer science) is in another schema. The "CS-Student" class will be integrated as a subclass of the "Student" class in the integration of two schemas.

2. *Descriptive conflicts.* Descriptive conflicts include naming conflicts due to homonyms and synonyms [7] [111], attribute domain, scale, constraints, operations, etc. [84].

[5]FOL (Fist Order Logic) can formalize the set theory.

3. *Structural conflicts.* Such structural conflicts are different from structural heterogeneity. Even if two modelers use the same data model, they can choose different constructs to represent common real-world objects. For instance, in object-oriented models when a modeler describes a component of an object type O, he has the modeling choices between creating a new object type or adding an attribute to O.

2.4.2 Semantic annotation

The goal of empowering computer systems with semantic interoperability rests on the desirability of computer systems being able to find information and to use it for purposes that the original creator of the information did not anticipate. This goal of flexible information reuse requires some degree of understanding of the information, which in turn requires that the information be encoded in some standard fashion that is interpreted identically by all systems using that information. As a shared model of what the information represent, ontologies are usually used to achieve the level of understanding. *Semantic annotation* is an approach to link ontologies to the original information sources.

Annotation is the extra information associated with a particular point in a document or other piece of information. For semantic annotation, the extra information is meaning definitions of the concepts used in a document. The meaning definitions are in most cases represented in ontologies. Annotation can be in the form of comments, or in the form of metadata. Metadata is data about data and it is used to facilitate the understanding, use and management of data. Machine-manipulable annotations are often organized as metadata, which is also the format of semantic annotations. There are a number of alternative approaches regarding the organization, structuring, and preservation of annotations. For instance, all the markup languages (HTML, SGML, XML, etc.) can be considered schemas for embedded or in-line annotation. On the contrary, open hypermedia systems use stand-off annotation models where annotations are kept detached, i.e. non-embedded in the content. Both annotation approaches can be document-level (annotating the document as a whole) or character-level (referring just a specific part of the text) [81] (see Figure 2.4). Embedded annotation seems easier to maintain. However, non-embedded annotations allow dynamic, user-specific semantic annotations because they can change corresponding to the interest of the user or the context of usage. The embedded annotations might also have negative impact on the volume of the content and complicate the maintenance [70].

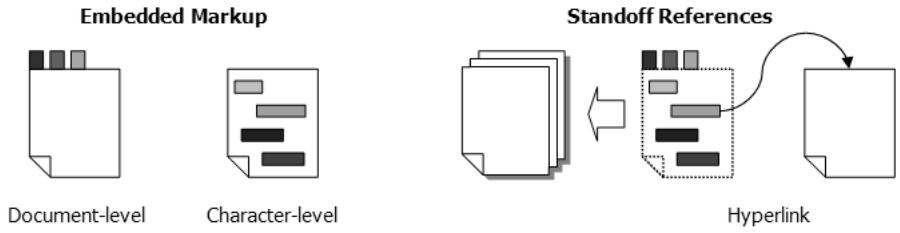


Figure 2.4: Embedded annotation and stand-off annotation [81]

In [70], semantic annotation is used to establish links from the entities in the text to their semantic descriptions so that a number of basic prerequisite for representation of semantic annotations are identified:

- Ontology (or at least taxonomy) defining the entity classes. It should be possible to refer to those classes;
- Entity identifiers which allow those to be distinguished and linked to their semantic descriptions;
- Knowledge base with entity descriptions.

Semantic annotation of Web services has emerged under the hypothesis that semantics can improve software reuse and discovery, significantly facilitate composition of Web services and enable integrating legacy applications as part of business process integration [203]. Semantic annotation of Web services is also called semantic markup of Web services, for which numbers of semantic markup languages and approaches are proposed such as WSMO [209], METEOR-S [187], OWL-S [198], SWSA/SWSL [181], WSDL-S [203]. They can be categorized into: a) annotating information in WSDL with ontologies (METEOR-S, WSDL-S); b) formalizing ontologies of Web service as a Semantic Web services representation language (WSMO, OWL-S and SWSA/SWSL).

In [206], *semantic annotation of process models* is concerned as a prerequisite of the vision of Semantic Business Process Management, which is very close to our proposal. It will enable (or enhance) additional functionalities, namely the discovery and autocompletion of process fragments, which lead to more effective modeling with respect to the reuse of existing process artifacts at the conceptual level. The executable process models can be partly generated from the conceptual business process models, which indicates there are underlying links between business process models and executable Web services. Semantic annotation of business process models could therefore enable more automation in the implementation phase because the corresponding Semantic Web services can be discovered automatically [206]. Although the work has just initiated and it is still an ongoing project, it shares the same vision with ours, i.e. semantic annotation can be also concerned as an alternative approach to achieve the semantic interoperability of semi-structured sources such as business process models, in spite of semantic annotations of unstructured sources (e.g. textual documents) and structured sources (e.g. WSDL described Web services). Efforts on the semantic enrichment of enterprise models by semantic annotations are also put by TG4 (Task Group 4: Semantic Enrichment of Enterprise Modeling, Architectures and Platforms) in EU project INTEROP (Interopreability Research for Networked Enterprise Applications and Software, FP6 508011) [62], in which the main achievable targets are the semantic interoperability for model exchange, model transformation and model traceability. As the contemporary work, our research shares some similar objectives and available technologies. Since we participated in the INTEROP project, our contributions are also devoted as part of the results in the project.

2.5 Business Process Model

A *business process* is defined in [23] as a set of coordinated tasks and activities, conducted by both people and equipments, that will lead to accomplishing a specific organizational goal. A process model describes a way of working at the type level according to the abstraction levels for processes in [155], which is corresponding to OMG's M1 layer. A process is an instantiation of the process model, corresponding to the M0 layer. The same process model is used repeatedly for the development of many applications and thus, has many instantiations. A process meta-model is at the meta-type level with respect to a process (see Figure 2.5).

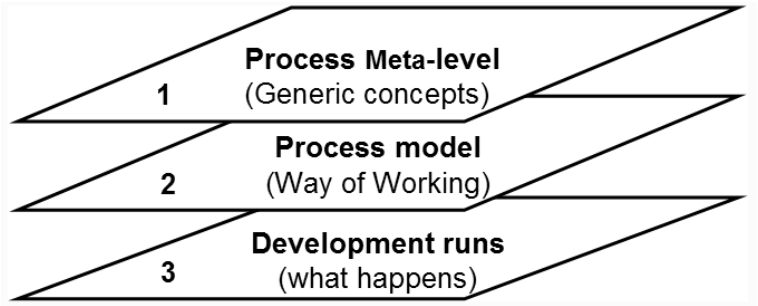


Figure 2.5: Abstraction levels of processes [155]

There are four types of coverage where the term process model has been defined differently in [27]:

- Activity-oriented: related set of activities conducted for the specific purpose of product definition; a set of partially ordered steps intended to reach a goal [34].
- Product-oriented: series of activities that cause successive product transformations to reach the desired product.
- Decision-oriented: set of related decisions conducted for the specific purpose of product definition.
- Context-oriented: sequence of contexts causing successive product transformations under the influence of a decision taken in a context.

Therefore a business process model defines the details of a business process flow and modeling all the data, resources, and other elements that the flow uses. As described in [9], a business process is usually composed of process steps that are normally connected through control flows, and these control flows connect activities with decision nodes. A decision node holds the business rules (transition conditions) that are evaluated to decide what path in the process should be followed. Modeling includes decomposition of a business process into sub-processes and adding required process elements to the model.

A business process model is initially constructed based on certain business requirements. The usage of a process model could be for the analysis of the process (i.e. descriptive, prescriptive and explanatory) or the creation of new process models by

modifying existing models rather than recreating it from scratch [9]. A *process model fragment* is a part of a business process model which is designed and managed to be reusable. Generally, reuse of pre-existing model fragments can facilitate and speed-up the construction of a new model.

As reusable knowledge, process models can be discovered and reused in a goal-driven approach through matching process models to the desired goals. Goals lead to the exploration and consideration of alternatives, decision spaces, tradeoffs and decisions. Goals have been used as an important mechanism for connecting requirements to design and supporting reuse [213]. Goal-driven search of design components [54] and goal-driven discovery of services [90] use such kind of mechanism, where the components or services are selected based on capability to fulfill the desired goals (requirements).

A process model represents how to do things not why to do things. Although a process must achieve certain goals, the relationships between goals and processes are not explicitly represented in many process models because few process modeling languages and modeling tools offer such mechanism. With few process modeling languages, goals can be modeled and linked to elements of process models, e.g. EEML process and goal models [77]. Some goal modeling methods for requirement engineering such as KAOS [21], i*/GRL [55], GBRAM [56] need to be adapted for process modeling. However, the representations of the goals and processes are various in different models.

2.6 Process Knowledge Management

2.6.1 Knowledge and process knowledge

Knowledge is information that comes laden with experience, judgment, intention and values, in short, knowledge is information that is digested and internalized by human beings [169]. However, a simply collection of information is not knowledge. In [185], information and knowledge is distinguished as follows. Information only tells what it is, with great dependence on context for its meaning and with little implication for the future. Pattern [6] has the potential to represent knowledge when one is able to realize and understand the patterns and their implications. For example, the information — interest rate 5% — is the factor used by the bank to compute interest on the principal. Based on the information, there is a pattern that for $100 in a savings account the bank pays 5% interest yearly and then at the end of one year the principal in the account will be $105. Such pattern represents knowledge of the amount of the interest will be earned depends on the interest rate, the amount of the principle in the account and the period of the saving. Therefore, a pattern tends to create its own context rather than being context dependent to the same extent that information is. Pattern also serves as an Archetype [165] with both an implied repeatability and predictability. Understanding the pattern knows one what is knowledge.

Process knowledge is also known as process-related knowledge. Taking the pattern view of knowledge, process knowledge is the pattern about processes which are understandable, repeatable and predictable. Another way to specify process knowledge is to ask the question "Where is the knowledge in processes?". The answers are provided in [112] such as:

- **Inputs and Outputs.** One place knowledge comes into play in a process is as a production input, which is to be operated upon and transformed into an output.
- **Controllers.** Another place where knowledge comes into play in a process is in the form of controls. In this case, knowledge can be regarded to be "embedded" in the process.
- **Processors.** The "Processor" performs the actions needed to produce a result from the process. This kind of knowledge is "encoded" in the process.
- **Design.** A lot of knowledge is "embedded" in the processes in the form of specifications for the outputs, the inputs, the routines and the requirements. The knowledge in the process makes itself felt through what is done, when, how, by whom and to what standards.

The first three answers concern the process automation and execution. The last one is associated with process models in the analysis and design phases. Our research objects are business process models at the conceptual level. In [122], Olivé visions a goal of automating IS building by a conceptual schema-centric development (CSCD). A conceptual schema provides general knowledge about the IS domain and about the functions the IS has to perform. With such a vision, the main purpose of the activity of business process modeling is to elicit the process knowledge of the corresponding IS. Hence, we take the fourth answer to specify that the process knowledge in our research context is embedded in business process models. This statement is also reflected in [79] with "process models are carriers of process knowledge, knowledge of how to do things".

2.6.2 Knowledge representation

Nonaka and Takeuchi [114] argued that a successful knowledge management program needs to, on the one hand, convert internalized tacit knowledge into explicit codified knowledge in order to share it, but also on the other hand for individuals and groups to internalize and make personally meaningful codified knowledge once it is retrieved from the knowledge management system. A crucial fact is knowledge representation.

Although process knowledge is embedded in process models, it does not mean all process models are process knowledge, that is, not every process modeling representation is knowledge representation. Knowledge representation can be various in terms of the distinct roles it plays, each crucial to the task at hand [22]:

- A knowledge representation (KR) is most fundamentally a *surrogate*, a substitute for the thing itself, used to enable an entity to determine consequences by thinking rather than acting, i.e., by reasoning about the world rather than taking action in it.
- It is a set of ontological commitments, i.e., an answer to the question: In what terms should we think about the world?
- It is a fragmentary theory of intelligent reasoning, expressed in terms of three components: (i) the representation's fundamental conception of intelligent reasoning; (ii) the set of inferences the representation sanctions; and (iii) the set of inferences it recommends.

- It is a medium for pragmatically efficient computation, i.e., the computational environment in which thinking is accomplished.

- It is a medium of human expression, i.e., a language in which we say things about the world.

The conclusions of the research in [22] are drawn all five roles matter; representation and reasoning are interwinded; different representations are often combined; they have formal equivalence. Various artificial languages and notations have been proposed for representing knowledge, such as CycL [20], IKL [50], KIF [29], Loom [97], OWL [195] and KM [71]. They are typically based on logic and mathematics, and have easily parsable grammars to ease machine processing.

2.6.3 Knowledge management activities associated with process models

Knowledge Management (KM) is an approach to discovering, capturing, and reusing both tacit (in people's heads) and explicit (digital or paper based) knowledge as well as the cultural and technological means of enabling the knowledge management process to be successful. Usually, the knowledge management covers both instance level and type level knowledge — learning from instances and abstracting them to reusable knowledge. In a KM system, knowledge processes essentially revolve around the following steps [174]:

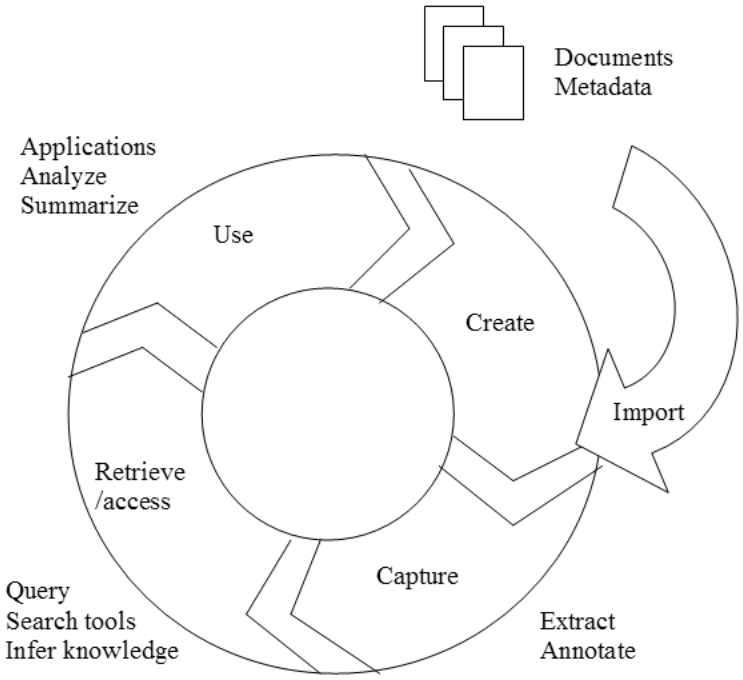


Figure 2.6: Knowlege Process [174]

- *Creation or import.* The contents need to be created or converted so that they fit the conventions of the company.
- *Capture.* Knowledge items have to be captured in order to determine their importance and how they mesh with the company's vocabulary conventions.
- *Retrieval and access.* This step satisfies the searches and queries for knowledge by the knowledge worker.
- *Use.* The knowledge worker will not only recall knowledge items, but will process them for further use.

The steps are illustrated in Figure 2.6. In this thesis, the scope of the knowledge management is only limited on type level process knowledge – enterprise/business process models. Therefore, in the context of our process knowledge management, process modeling in knowledge representation language is concerned as knowledge creation. Importing knowledge can be the transformation of conventional process models in the knowledge representations. Knowledge capture is to capture the essential contents in process models through annotation techniques by creating metadata conforming to ontologies. Process knowledge retrieval and access can be conducted through a conventional query or search tools, or by applying the ontology and the inference mechanism to derive further views of process knowledge. The possible uses of process knowledge include analysis of existing process models, reusing the legacy models to create new process models, integrating systems based on the process descriptions, etc.

2.7 Summary

This chapter has outlined the context of this research work. We have introduced the theoretical and technical setting relevant to our research topic, such as modeling theories and methodologies, the Semantic Web, ontology, semantic annotation, business process model and knowledge management. Main points are summarized as follows.

- From the discussion of modeling theory and methodology, we has scoped that our research area is at conceptual modeling level.
- Ontology as the key enabling technology for the Semantic Web provides an explicit representation of a shared conceptualization.
- Ontology and conceptual model share certain common grounds, which indicates the potential links between them, e.g. usage combination and technology benefits from each other.
- In this work, a concept modeling language – RML is used to analyze meta-models and ontologies with graphical notations; while, an ontology modeling language – OWL is applied to enable ontology machine-interpretable.
- Semantic annotation is an approach to achieve semantic interoperability of heterogeneous information resources making use of ontology.

- Process knowledge is embedded in process models whose semantics might be represented heterogeneously.
- Semantic annotation can be applied in capturing and representing process knowledge to facilitate process knowledge management such as the retrieval and reuse of heterogeneous process models.

After having established the key areas of our research in this chapter, we will continue to detail survey of state of the art in the these areas, namely, process modeling languages, process ontologies, goal modeling and annotation approaches.

Chapter 3

State of the Art

This chapter investigates a number of existing process modeling languages and process ontologies which are often used in industrial business and academic research projects. Such investigation is intended to seize the most essential concepts and relationships describing the semantics of process, which construct the basis of the process ontology. The process ontology is used as the intermediary for the semantic annotation to build the semantic mapping between different process representations. Semantic annotation approaches and tools for different types of information are surveyed to identify requirements of semantic annotation methods and tools.

3.1 Process Modeling Languages

Business Process Modeling (BPM) plays a vital role in the lifecycle of enterprise systems development. BPM is sometimes known as Business Process Management. Over the years, the scope of business processes and BPM has broadened. It ranges from process definition, workflow management system, and integration techniques. For the specification aspect, process definition is conducted by modeling activities with certain process modeling language. For the execution aspect, workflow management system is necessary for the automation and control of process execution based on process models. For the system aspect, BPM today is an enterprise integration technology complementing Service-Oriented Architecture (SOA) and other integration APIs.

In the history of business process modeling, numbers of process modeling languages were created, evolved and new proposals continue to emerge. Some of them cease to be used because of the evolution of the modeling methodologies, but many of them are still lively applied. Those process modeling languages exist contemporaneously due to their special expressive powers to emphasize different aspects for various applicational purposes. Process modeling languages might be informal, semi-formal or formal. Informal modeling language is just natural language, which is seldom used in modeling discipline. Most visual modeling languages are semi-formal modeling languages, in which there are a set of notations with semantic definitions by meta-models. Formal modeling languages are those whose semantics are rigidly defined by logic or mathematics. A semi-formal modeling language is generally easier to understand than a formal language, while the latter can enable the computation of model semantics.

Process modeling languages are distinct from each other because they have different meta-model elements/modeling constructs to represent process properties. Process properties are also generally called process perspectives in this thesis. Although the representations of process properties are various from the different modeling languages, the following perspectives of business process are often presented in most process models: structural, operational/functional, control, resources, organizational, data transaction. Structural perspective is a view of the structure of activities or processes, such as the structure of parts (sub-activity, sub-process). Operational/Functional perspective is normally represented through the descriptions about activities or functions with their inputs and outputs. Control is associated with operational/functional perspective and it emphasizes the ordering and constraints of processes. Resources of processes are those consumed or produced along the processes, and they also include tools or other mechanisms required to aid processes. Organizational perspective display the process flows between different organizations and participants involved in processes. Data transaction includes information or message transit and state change, in which data value is often involved.

Based on the discussions above, we make a paradigm of BPM (Business Process Management) systems with the focuses on process definitions (Figure 3.1), which is adapted from the Business Process Management Mind Map [14].

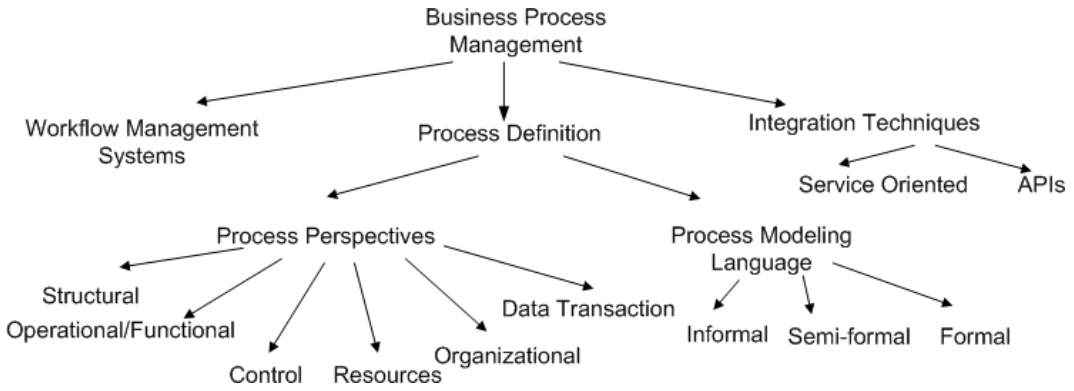


Figure 3.1: The paradigm of business process management systems

We in this chapter introduce several business process modeling languages. The features of each modeling language are investigated from the process definition perspective according to the paradigm of business process management systems. Being consistent with our research subject and scope, those modeling languages should be able to represent business processes on the conceptual level. That is, the process modeling languages could be used for CIM (Computation Independent Model), but not restricted only to CIM.

3.1.1 Petri Nets

Petri net is a formal, graphical, executable language for specifying dynamic behavior. Petri nets were introduced by C.A.Petri in the early 1960s as a mathematical tool for modeling distributed systems. It is widely used in software design, workflow manage-

ment, data analysis, concurrent programming, reliability engineering and diagnosis of programming. As a graphical modeling language, Petri nets can visualize dynamic behavior with the nets consisting of place nodes (circles), transition nodes (bars) and arcs connecting places with transitions. Places can contain tokens. Tokens are used in these nets to simulate the dynamic and concurrent activities of systems. In a mathematical form, the state of a Petri nets can be represented as a M vector, algebraic equations, and other mathematical models governing the behavior of systems.

Due to such a small number of modeling constructs, Petri nets have the limited expressivity of the structural, resources, organizational, functional or operational perspectives. However, it performs very well on the data transaction and control perspectives. Many execution, simulation and analysis techniques and applications have been developed based on Petri nets. Numbers of extensions have also been made to deal with the drawbacks of Petri nets, such as lacking data and hierarchy concepts, not user-oriented and scope limited to process aspects. The development of high-level Petri nets and hierarchical Petri nets targets the data structuring and hierarchical decomposition issues, e.g. coloured Petri nets [64]. User-oriented visualizations and complementary modeling perspectives, e.g. for resources [30], structural [82] [83] [108], and time, have been proposed.

3.1.2 EPC (Event-driven Process Chain)

EPC is a semi-formal graphical modeling language for business process workflows. It was developed within the framework of ARIS [160] in the early 1990s. EPC has been applied in ERP system describing workflow, e.g. SAP R/3, and now more widely. Basic EPC notations only include events (hexagon), functions (rounded rectangle) and connectors which can be transformed into Petri nets, but it introduces AND, OR and XOR connectors. An example of the notations in an EPC model is illustrated in Figure 3.2. EPC emphases more on the operational/functional and control perspectives than data transaction perspective as Petri nets do. An extended EPC includes organizational units to represent roles or persons, information or resource object that can be seen as input or output to functions, and process path to describe hierarchies of EPC processes.

A major strength of EPC is claimed to be its simplicity and easy-to-understand notation. This makes EPC a widely acceptable technique to denote business processes. Unfortunately, neither the syntax nor the semantics of EPC are well-defined [190]. EPC requires non-local semantics [69], so that the meaning of any portion of the diagram may depend on other portions arbitrarily far away. Recently, an XML-based EPC — EPML (Event-driven Process Chain Markup Language) has been proposed by Jan Mendling [105]. The motivation of EPML is to support data and model interchange in the face of heterogeneous Business Process Modeling tools. Tool orientation deals with the graphical representation of EPCs and EPML is able to store various layout and position information for EPC elements [129].

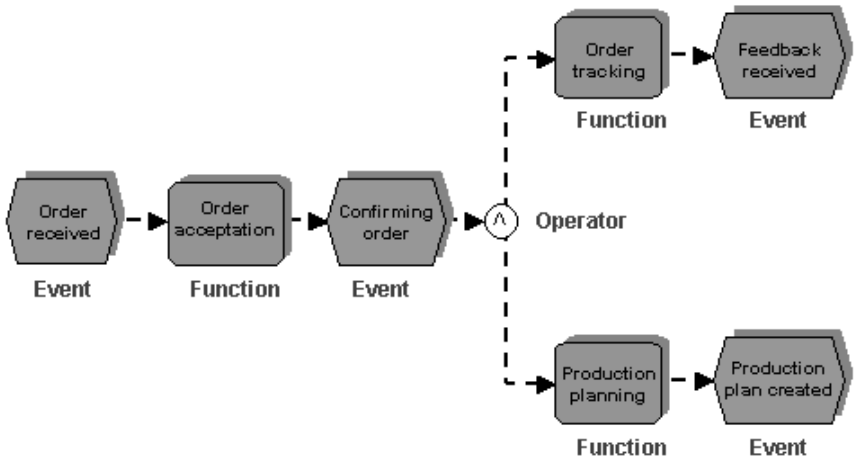


Figure 3.2: An example of EPC model [60]

3.1.3 EEML (Extended Enterprise Modeling Language)

EEML is originally developed in EXTERNAL project [59] to support development and use of interactive models[1]. The technological approach in EXTERNAL involves integration of several business process technologies, such as process-centric enterprise modeling, computational simulation models, hypermedia collaboration tools and emergent workflow tools. EEML is the research result of process-centric enterprise modeling. EEML is created to support process modeling across the four layers of process models:

- **Generic Task Type.** Identifying the constituent tasks of generic, repetitive processes and the logical dependencies between these tasks (so-called *reference processes*).
- **Specific Task Type.** Expanding and elaborating process models to facilitate business solutions. Elaboration includes concretization, decomposition, and specialization.
- **Manage Task Instances.** Detail planning, co-ordination and preparation for resource allocation concerning actual work environment and process instance.
- **Perform Task Instances.** The actual execution of tasks according to the determined granularity of work breakdown, which in practice is coupled to issues of empowerment and decentralization. At this layer resources are utilized or consumed, in an exclusive or shared manner.

EEML can be used for process and enterprise modeling on different levels (both type and instance level). Type level related to generic task type and specific task type layers are our interests. The language vocabulary of EEML is grouped into several domains. In this work, we only concern two domains: EEML process domain and EEML

[1]Interactive model: prescribed aspects of the model are automatically interpreted and ambiguous parts are left to the users to resolve, with tool support [65].

resource domain. EEML is a semi-formal language and has a set of graphical modeling notations. An illustrated example of EEML process model provides an overview of EEML notations for the process and resource domains in Figure 3.3. In process domain, EEML models the operational/functional perspective by task which is notated as a round rectangle with input and output (two diamonds on the left and right side within the rectangle). A task can be decomposed so that the hierarchy of process can be structured. The control perspective is represented through the flows (lines with arrow) linking tasks, milestones (diamonds along the flow with "start", "milestone" and "finish") and decision point (diamonds along the flow with logic connector like AND, OR, XOR). Organizational and resources perspectives are well supported by specifying different resource types (e.g. person, organization, information object, ...) associated with resource roles (circles in a task). It is not obvious to model the data transaction perspective in EEML although milestone could be used as "state" sometimes. Details of the EEML modeling language are described in **Appendix B**.

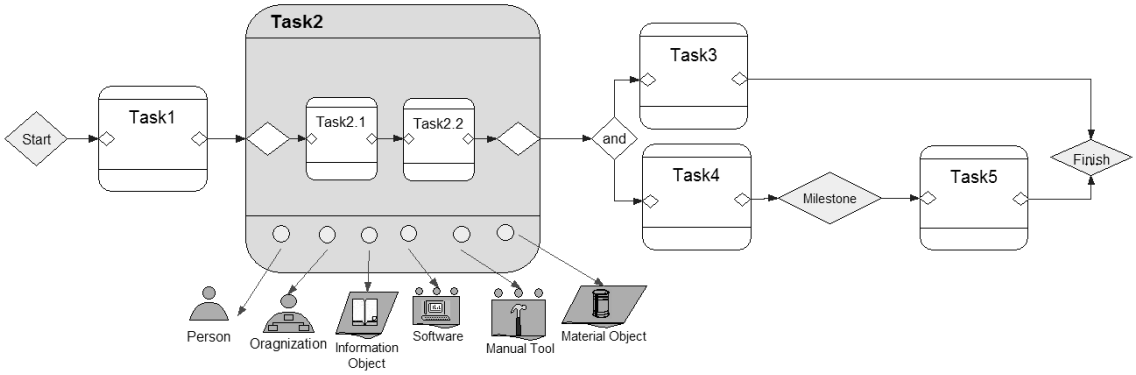


Figure 3.3: Overview of EEML modeling constructs and relationships for process and resource domains

3.1.4 UML Activity Diagram

As one of standard proposals in UML [125], UML Activity Diagrams in the behavior category are typically used for business process modeling. An Activity Diagram in UML represents the business and operational step-by-step workflows of components in a system, the overall flow of control. In UML 1.x, an Activity Diagram is a variation of the UML State Diagram[2] where the "states" represent actions, and the transitions represent the activities that happen when the operation is complete. The UML 2.0 Activity Diagram, while similar looking to the UML 1.x Activity Diagram, now has semantics based on Petri nets. In UML 2.0, Activity Diagram can represent the interaction overview of e-business.

UML Activity Diagram is a semi-formal language with the following basic graphical notations [2]: initial node and activity final node (filled circle with or without border), activity (round rectangle), flow/edge (arrow), fork and join (black bar), decision and

[2]UML State Diagrams depict the dynamic behavior of an entity based on its response to events, showing how the entity reacts to various events depending on the current state that it is in [2].

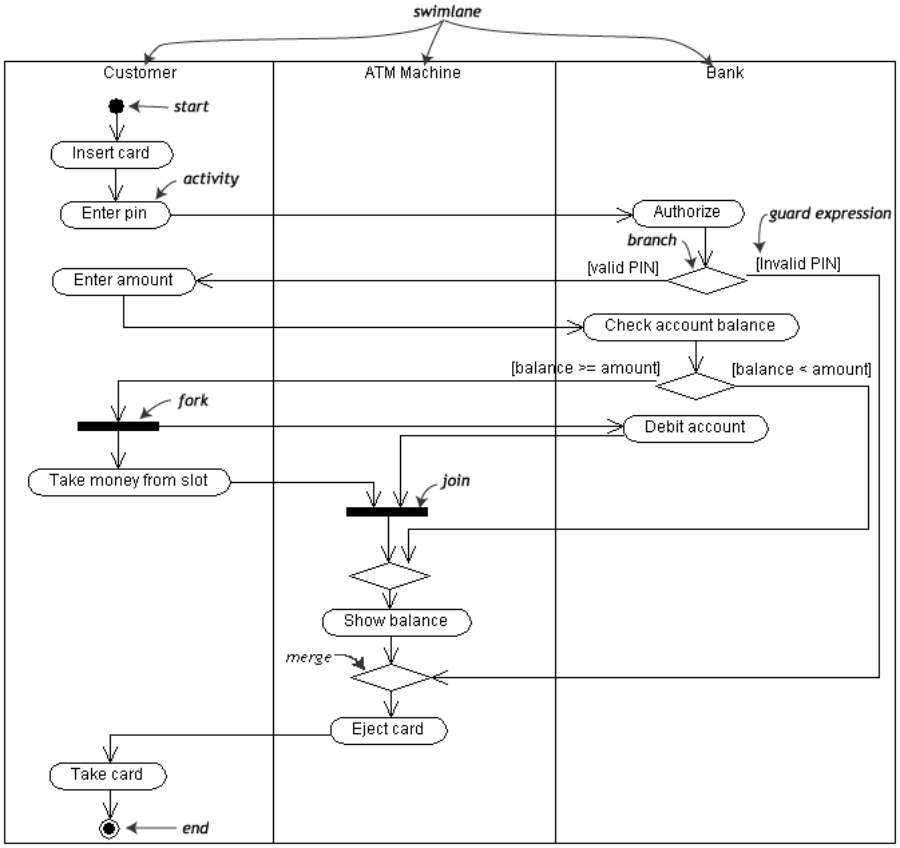


Figure 3.4: An example of UML activity model [107]

merge (diamond), partition/swimlane (line without arrow like a border). The notations are exemplified in Figure 3.4. Partition/swimlane is normally used to indicate who/what is performing the activities, which can specify the organizational perspective. The rest notations can represent the operational/functional and control perspectives. No particular notations are provided for the phenomena associated with data in UML Activity Diagram. However, it allows to use texts linked to the flows and activities to indicate condition, object or message passed in the process. UML Activity Diagram is often asked to be combined with other UML diagrams to model multiple process perspectives. For example, UML Use Case Diagram can capture loose collections of related activities at the high level, which might provide a view of the structural perspective of processes. After all, the lack of first class process modeling primitives is a major limitation of UML [26].

3.1.5 BPMN (Business Process Modeling Notation)

BPMN is a new standard of modeling business processes and Web service processes put forth by the Business Process Management Initiative [126], which is merged with OMG organization now. BPMN is a semi-formal modeling language providing notations understandable by business users. Nevertheless, it is also created as visual

Table 3.1: Modeling constructs of different business process modeling languages

	Petri Net	**EPC**	**EEML**	**UML Activity**	**BPMN**
Structural	-	process path	process/task decomposition	UML use case	collapsed /expanded sub-processes
Operational /Functional	-	functions	process/task (with input and output)	activity	task, process
Control	transition node, arc	connector, flow	flow, milestone, decision point	flow/edge, fork, join, decision and merge	sequence flow, fork, join, decision, merging, looping
Resource	-	*extension with information, resource object*	resource role, resource type	-	data object
Organizational	-	*extension with role, person*	resource role, resource type (organization, person)	partition, swimlane	pool, (swim)lane
Data Transaction	token	event	milestone, flow with resource	UML state diagram	message flow with data object

modeling language for execution languages, such as BPEL4WS [58] and BPML [11]. The BPMN specification provides a mapping between the graphics of the notation and the constructs of those formal languages. BPMN intends to provide businesses with the capability of understanding their internal business procedures in graphical notations and give organizations the ability to communicate these procedures in a standard manner.

BPMN is initiated as a standard process modeling language for conventional business, B2B and services process modeling. Hence BPMN has the capabilities of handling B2B business process concepts, such as public and private processes and choreograhies, as well as advanced modeling concepts, such as exception handling and transaction compensation in addition to the traditional business process notations [12]. Private business processes are those internal processes to a organization and they are traditional business processes or workflows. Public processes are also called abstract processes or interface processes according to the BPMI terminology. They represents the interactions between a private business process and another process or participant. Choreographies represents collaboration processes. A collaboration process depicts a sequence of activities that represent the messages being sent between the entries involved.

According to the BPMI terminology, the modeling constructs are the elements with corresponding notations. The main elements of BPMN are event, task, process/subprocess, sequence flow, message flow, pool, lanes, data object, fork (AND-split), join (AND-join), decision/branching point (OR-split), merging (OR-join), looping, etc. We refer those elements to the perspectives in the BPM systems paradigm. A sub-process is a compound activity included within a process. The collapsed/expanded sub-process can hide and show the details of the sub-process. Hereby, the structural perspective can be represented by BPMN through such a mechanism. Respectively, task, process/subprocess are elements for the operational/functional perspective. Tasks and processes are controlled through event, sequence flow, fork, join, decision, merging and looping. Resource can be represented by data object. Organizations are usually represented by lanes in the pool. The data transaction is specified through message flow together with data object.

The details of the BPMN specifications from BPMI can refer **Appendix A**. The names and the notations of the BPMN elements used in this section are from the standard proposal. When a modeling tool implements a certain modeling language (e.g. BPMN), the names and the notations might be slightly changed to adjust the tool implementation. For example, task, process/subprocess are implemented as "Logical Process" in an enterprise modeling tool Metis [183].

3.1.6 Categorizing the modeling constructs of process modeling languages

The investigation of existing process modeling languages has shown the diversity of modeling constructs. The modeling constructs of the process modeling languages is categorized in Table 3.1 according to the six process perspectives defined in the paradigm of BPM systems.

3.2 Semantic Interoperability and Process Ontologies

Semantic heterogeneity baffles the effective management of distributed knowledge, which relates to semantic interoperability under an extensive system exchange and integration situation. Semantic interoperability issue has been studied in different domains and applications such as schema and data integration of databases, meta-data interoperation among distributed digital libraries , agent-based Web services discovery and composition, enterprise integrations and so on. Research on semantic interoperability generally is categorized as: *mapping* and *intermediary* approach. The *mapping approach* can be sub-classified into point-to-point mapping and global mapping. Point-to-point mapping is the particular mapping built for two participating systems. Such a mapping can maximally preserve the original semantics of each system, but it is costly when a large number of systems need to interchange with each other or new unexpected system needs to participate. Global mapping attempts to construct mappings between participant systems and a global schema [24] [17] [66]. This requires to build a global schema which is designed to be dependent of particular schemas or applications [152]. The problem of this solution is not portable and does not adapt well to the addition of new systems. The *intermediary approach* is to make use of intermediary mechanisms to coordinate heterogeneous semantics. Domain-specific knowledge, mapping knowledge, or rules are modeled by the intermediary mechanisms such as mediators, agents, ontologies, etc. [132]. Generally, this approach uses a machine understandable definition of concepts and relationships between concepts so that there is a shared common understanding within a community. The knowledge or rules are domain specific, but independent of particular schemas and applications. However, one needs additional tools to actually capture and represent the knowledge (i.e., specifications and mappings) needed in order to resolve semantic conflicts.

Semantic Web technology and some research results of ontology have initiated larger amount of research interests on the intermediary approach, especially applied in the applications of collaborative business (such as interoperation of enterprise processes, Web services, etc.). We apply the intermediary approach in this research work. Hence, the top-level ontology — *process ontology* for business process representation, and *domain ontology* for a given business domain are used as the intermediaries in our approach.

The process ontology is used to reconcile the heterogeneous semantics of process modeling constructs (i.e. meta-model semantics) existing in different process modeling languages. It indicates that a process ontology should include a set of meta-concepts that are able to describe the semantics of process models. In this section we survey a number of process ontologies having different motivations: BWW ontology [204] for building a modeling fundamental of information systems, MIT process handbook [99] for organizing process knowledge, TOVE ontologies [31] for creating a set of ontologies to support enterpise modeling, PSL [120] for serving as an interlingua of all types of manufacturing processes, PIF [85] for exchanging business process models across different formats and schemas, OWL-S [198] for establishing a descriptive language of Web services, WSMO [209] for describing various aspects related to Semantic Web services, POP* [139] for providing a mapping mechanism between enterprise models and modeling tools, and UEML2 [140] for creating an intermediate language of various existing modeling languages. The process perspectives of the paradigm of BPM systems

(Figure 3.1) are also applied to study the semantic representations of those process ontologies.

3.2.1 BWW (Bunge-Wand-Weber) ontology

BWW (Bunge-Wand-Weber) ontological constructs provide a semantic basis of meta models of conceptual models. The BWW ontology can be used as the top level ontology because the BWW ontology is initially built as a set of core constructs that underlie the computer science and information systems fields, especially for modeling tasks. Although BWW ontology is not specially created as process ontology, it covers the concepts which can be used for process modeling. BWW ontology consists of the following major concepts, namely *thing*, *property*, *state*, *law*, *event*, *transformation* and *system*. Those concepts are too abstract to specify semantics regarding the process perspectives. Therefore, the BWW ontology is often used as a meta-meta-model and specified further according to the modeling perspectives, e.g. the process perspectives.

The BWW ontology has been modeled in different languages, e.g. originally set-theoretic definition [204] (a formal representation), a list of textual descriptions [128] (an informal representation), entity relationship model [158] (a semi-formal representation).

3.2.2 MIT process handbook

The MIT process handbook project has developed rich online libraries for sharing and managing many kinds of knowledge about business. The essential activity of the MIT process handbook is to organize process knowledge in libraries, which is related to the process representation issue. Two sources of intellectual leverage are exploited for the representation: (1) notions of *specialization of processes* based on the ideas about inheritance from object-oriented programming, and (2) concepts about *managing dependencies* from coordination theory [98]. Specialization includes part specialization and type specialization. Dependencies are distinguished as flow dependencies, sharing dependencies and fit dependencies.

The representation of a process in the MIT process handbook is modeled as different types of *entries* of a given activity in the repository [99]:

- Description. Descriptions include any useful or interesting information to users such as definitions, comments, figures, sources for further information, links to other entries, or links to other Web pages.
- Parts. The point of view embodied in this entry is that these activities constitute one possible representation of the 'deep structure'.
- Properties. Any other kind of information related to a given activity: the last modification, time required to do the activity, cost of doing the activity, location of the activity, and so on.
- Related Processes. An extensive network of relationships among different entries.
- Specializations. Specialization hierarchies listing a number of levels (e.g. 18 levels in some cases) of increasingly specialized activities of the given activity.

- Bundles. A group of related specializations. In general, they are grouped based on questions of an activity: how? what? who? when? where? and why? For most activities in the handbook, some subset of these questions provides a systematic and logical way of grouping the different specializations that appear.
- Uses. A list of activities where the given activity is used as a part.
- Generalizations. The set of activities are type processes of the given activity.

Other kinds of entries are also used in the process handbook, such as Dependencies (flow, sharing and fit), Resources (inputs, outputs, actors and locations), and Exceptions (exception types and ways of exception detection, anticipation, avoidance and solution).

For the process perspectives, the operational/functional perspective of a process is elaborated through the entries of an activity. Parts and Uses provide a structural view of activities. Resources and Bundles can specify resources and organizational perspectives. The control perspective is represented through Dependencies. The data transaction perspective is not specified explicitly through those entries.

The MIT process handbook contains generic models of typical business activities (e.g. buying, making, and selling) and specific case examples of business practices. They are classified and represented through the entries. In such a way, process knowledge in the repository looks like business process taxonomy and patterns. The semantics of entries are not formally defined for the repository but they can be represented in PIF [85] to share the process descriptions.

3.2.3 TOVE (Toronto Virtual Enterprise) ontologies

TOVE ontologies are stated as a *common sense enterprise model* covering a set of integrated ontologies for the modeling of both commercial and public enterprises. The TOVE consists of a set of generic core ontologies, including an activity ontology that spans activity, state, time and causality, a generic resource ontology describing the properties of the enterprise resources such as machines, electricity or raw materials, capital, human skill and information, an organization ontology spanning structure, roles, and communication, and a product ontology which includes features, parameters, assemblies, versions, and revisions. It also includes a set of extensions to these generic ontologies to cover concepts such as cost and quality. However, the primary component of the ontology is the terminology for classes of processes and relations for processes and resources, along with definitions of the classes and relations [44]. The KIF (Knowledge Interchange Format) [29] is the language providing the logical lexicon for the TOVE ontologies. The non-logical part of the lexicon consists of expressions (constants, functions, relations) that refer to concepts in some domain.

At the heart of the TOVE enterprise model lies the representation of an *activity* and its corresponding enabling and caused *states*. Activities and states specify transformations by fluent and actions. Axioms of temporal projection represent how the actions change the status of a state. A set of constants are defined for status (of state and of activities), and a set of actions are defined by functions with parameters of the state and the associated activity [101]. Evidently, the data transaction perspective can be supported by activity ontology. The states related to classes of resources

in resource ontology can represent the resources perspective. Organization ontology in TOVE includes organization-role, goal, skill, constraints, organization-membership, communication link, authority link and empowered, which covers the organizational perspective. Logic connections of process are not explicitly specified in TOVE, but the temporal axioms describe the control of the activities at the time dimension. TOVE includes the aggregation of activities by the predicate *subactivity(a,a')* to present the structure perspective.

3.2.4 PSL (Process Specification Language)

The PSL project at the National Institute of Standards and Technology (NIST) is addressing the interoperability issue by creating a neutral, standard language for process specification to serve as an interlingua to integrate multiple process related applications throughout the manufacturing life cycle [161]. PSL is a formally defined process specification language. The underlying language of PSL is KIF. The model theory of PSL provides a rigorous mathematical characterization of semantics of the terminology of PSL. The proof theory of PSL provides axioms for the interpretation of terms in the ontology.

PSL-Core specifies the semantics of the primitives in the PSL ontology corresponding to the fundamental intuitions about activities. In PSL-Core, there are four basic classes: *Object*, *Activity*, *Activity Occurrence*, and *Timepoint* and four basic relations: *Participates-in*, *Before*, *BeginOf*, and *EndOf*. Extensions can be made to PSL-Core. Figure 3.5 illustrates the extended modules required to define the terminology for generic classes of activities and their ordering relations. Other extensions are made to PSL-Core, such as Resource Roles, Processor Actions, and Resource Paths.

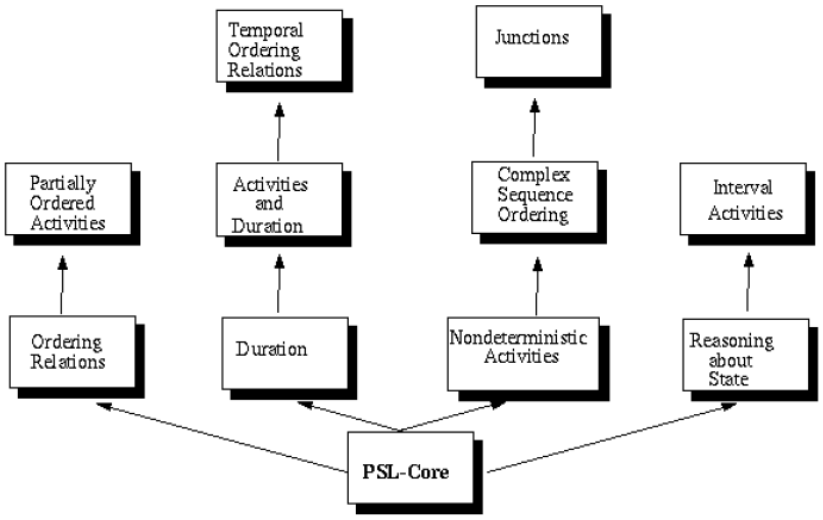


Figure 3.5: PSL modules for generic classes of activities and their ordering relations [161]

PSL-Core provides basic expressivity on the operational/functional, data transaction, control, and resources perspectives. The extensions can enrich the semantics of those perspectives.

3.2.5 PIF (Process Interchange Format)

PIF [85] aimed to develop an interchange format to help automatically exchange process descriptions among a wide variety of business process modeling and support systems such as workflow software, flow charting tools, process simulation systems, and process repositories. PIF is a formal process modeling language, whose syntax adopts that of KIF. The main classes and their relationships are displayed in Figure 3.6. The classes and the relationships are notated with rectangle box in the figure, and they are all the modeling constructs of PIF.

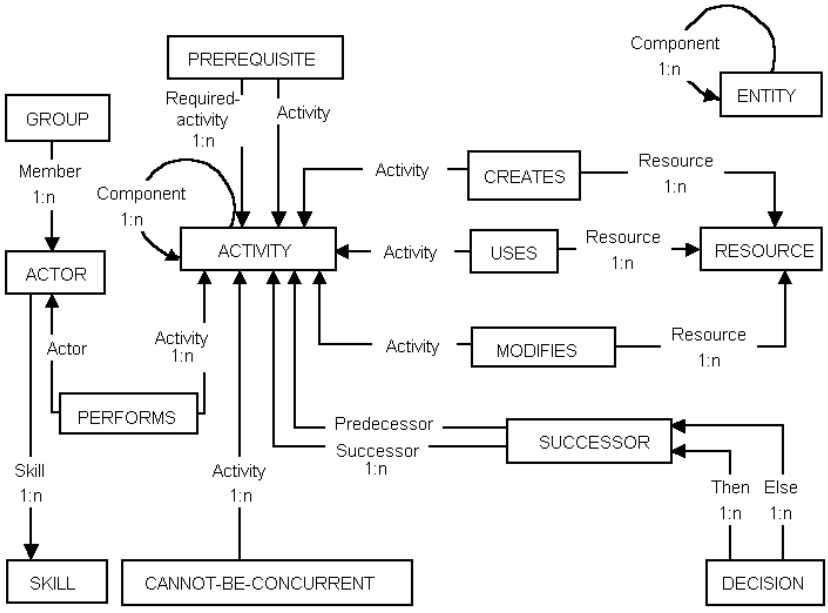


Figure 3.6: The PIF classes and relationships [85]

There are three core classes — *Activity*, *Actor* and *Resource*. Other classes related with the three classes through the relationships are attributes of them. It is obvious that PIF supports the structural perspective by the decomposition relationship of Activity. The operational/functional perspective can not be properly represented by lacking input/output specifications. Relationships among the class Resource provide the expressivity of the resources perspective. The control perspective can be represented through the classes such as Prerequisite, Cannot-be-concurrent, Successor, Decision and the relationships among them and Activity. Actor and their attributes can depict the organizational perspective. No data transaction is supported by PIF.

PIF provides formally-defined declarative semantics, the expressive power to represent knowledge required for a typical application knowledge base, and a structure that enables semi-automatic translation into and out of typical representation languages. PIF also adopts the frame syntax, which is an extension of the KIF syntax for representing object-based knowledge [85]. The PIF project has been merged with the PSL project at NIST, so that the PIF has been incorporated into the PSL Core and its extensions.

3.2.6 OWL-S

OWL-S is a language for specifying the Web service ontology, based on OWL, which augments current Web services architecture with semantic metadata [186]. The motivations of the applications of OWL-S are automatic Web services discovery, automatic Web services invocation and automatic Web service composition and interoperation [198].

Three essential types of knowledge about a service can be described with OWL-S: advertising information for prospective clients by ServiceProfile, process model by ServiceModel, and transport protocols by ServiceGrounding. A process represented by the ServiceModel is a specification of the ways that a client may interact with a service. The process ontology is a set of concepts and relationships which are used to represent a ServiceModel. The process ontology modeled in OWL classes, properties and axioms is shown in Figure 3.7.

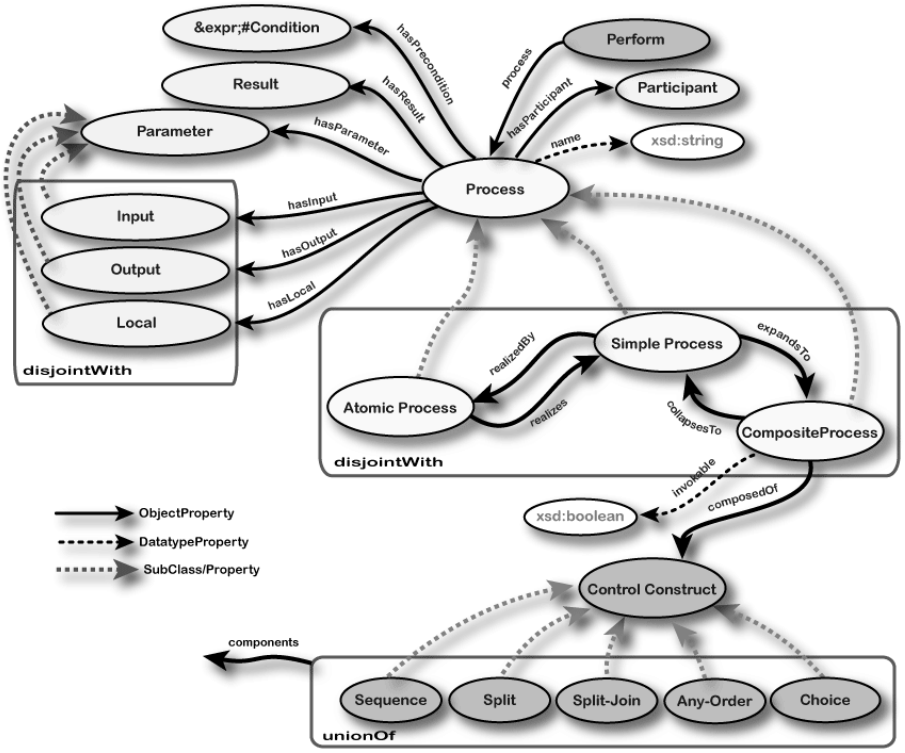


Figure 3.7: The process ontology of OWL-S [198]

In the process ontology of OWL-S, the operational/functional perspective is represented through process classes, parameter classes, their subclasses, and their relations. Distinguished subclasses of process — atomic process, simple process and composite process depict the structural perspective. A set of control constructs connecting processes support the control perspective. The organizational perspective is included by specifying the class participant in a process. The data transaction perspective is implicitly represented through the effect (the class result) of a process. The resources perspective is not specified in the process ontology although it might be inferred by

linking the parameters to a resource class which is defined separately from the process ontology.

The Intelligent Software Agents Lab at Carnegie Mellon University has played a critical role in the development of OWL-S from its very conception. In addition, the Softagent Group has also developed the most complete and integrated set of OWL-S development tools, the OWL-S IDE. However, OWL-S is criticized to suffer conceptual ambiguity, lack concise axiomatization, be designed too loosely and offer an overly narrow view on Web services [106].

3.2.7 WSMO (Web Service Modeling Ontology)

WSMO is a conceptual model for describing various aspects related to Semantic Web services. The objective of WSMO and its accompanying efforts is to solve the application integration problem for Web services by defining a coherent technology for Semantic Web services [210].

WSMO provides ontological specifications for the core elements of Semantic Web services. The representation of WSMO follows the MOF (Meta-Object Facility) [124] specification, and the MOF constructs **Class**, **sub-Class**, **Attributes** and **type** are used in the definition of WSMO. Every construct of WSMO is called a WSMO element. There are five top-level elements: annotations, ontologies, Web services, goals and mediators. WSMO does not contain the constructs to represent any of the process perspectives, but only provide interface to describe the functionality of the Web service. A twofold view on the operational competence of the Web service is provided: *choreography* decomposes a capability in terms of interaction within the Web service; *orchestration* decomposes a capability in terms of functionality required form other Web services. The process of Web services composition is excluded from WSMO. Another important feature of WSMO is that it introduces the elements **Goal** and **Mediator**. Goal can represent an objective of the execution of a Web service. Mediator is exploited to overcome interoperability problems between different WSMO elements. For example, a wgMediator (Web service to goal Mediator) links Web services to goals, indicating that the Web service (totally or partially) fulfills the goal to which it is linked. Such a mediator may explicitly state the difference between the two entities and map different vocabularies through the use of ooMediators (ontology to ontology Mediators) [209].

3.2.8 POP* (Process, Organization, Product and others)

POP* methodology is one of the research contributions of the EU project ATHENA [139]. The overall objective of ATHENA is to enable enterprise to seamlessly interoperate with others. The POP* methodology aims to develop a set of core modeling methodology elements for capturing collaborative enterprises design and management. It offers a model exchange device by providing a common format along with a mapping methodology to define the mappings from the various enterprise modeling languages to the common format. The POP* methodology includes a common format as the exchange format — the POP* meta-model containing a set of basic modeling constructs. There are five dimensions included in POP*, namely the Process, Organization, Product, Decision and Infrastructure dimensions [138].

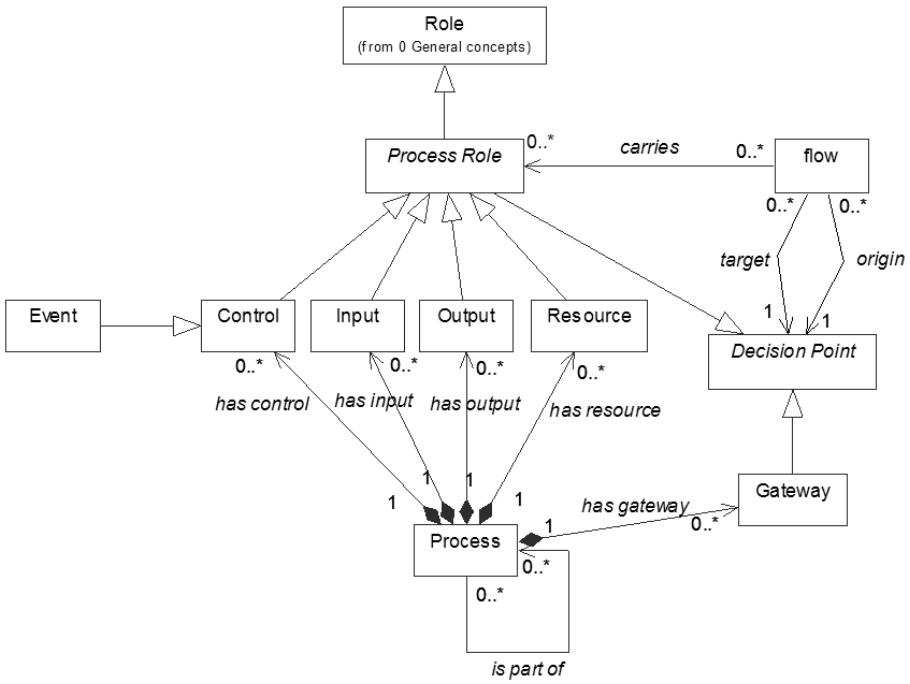


Figure 3.8: The POP* meta model for Process dimension [138]

The POP* meta-model is represented in UML class diagram to specify the concepts and their interrelations (including the cardinality). The properties of each concept and relationship are defined by property names and value types. Therefore, POP* can be described in the UML 2.0 Profile as a basis for further implementation of POP* as an enterprise modeling language [138]. Figure 3.8 provides an overview of the modeling constructs in the Process dimension.

Process is the central construct and the other constructs associated with it can well support the representations of the operational/functional, structural, control and resources perspectives. Transactions of state and data are implicitly represented through the constructs *event* and *process role* carried by *flow*. Th organizational perspective is not included in the Process dimension in this version of POP*. Some organizational concepts are introduced in the Organizational dimension, but no specific relationships are defined to associate them to process.

3.2.9 UEML2 (Unified Enterprise Modeling Language version 2)

UEML was initiated by the UEML project [130] to provide industry with a unified and expandable modeling language. UEML 2.0 and UEML 2.1 are further developed in the INTEROP-NoE project [140]. The work on UEML2 in this project is aimed on characterising **correspondences** between enterprise modeling languages to enable model interchange. A "UEML template approach" is applied in the development of UEML2.1. This approach requires a detailed (ontological) analysis of the constructs found in enterprise modeling languages and allows to formally define correspondences between constructs in distinct languages and thereby a UEML-based core enterprise modeling

Table 3.2: Ontological representations of different process ontologies

		BWW ontology	**MIT process handbook**	**TOVE**	**PSL**	**PIF**	**OWL-S**	**POP***	**UEML**
Representation primitives		concept set (set theory)	entry	sets of core ontology, axiom	class, relationship	class, attribute, relationship	class, property, sub-class	dimension, class, property (name & value type)	class, property
Process Perspectives	Structural	*extension of thing, property*	parts, uses	subactivity	-	component of activity	composite process, simple process, atomic process	process decomposition	composite, component
	Operational /Functional	*extension of thing, property*	activity	activity, action	activity	activity	process (with input, output, local)	process (with input, output)	activeThing, inputThing, outputThing
	Control	law	dependency	temporal axiom	ordering relationship	prerequisite, cannot-be, concurrent, successor, decision	control construct (sequence, split, split-join, any-Order, choice)	event, control, flow, decision point, gateway	*extension*
	Resource	*extension of thing, property*	resource	resource ontology	object	resource, entity	-	resource	resource
	Organizational	*extension of thing, property*	who (bundle)	organizational ontology	-	actor, skill, group	participant	organizational dimension	*extension of resource*
	Data Transaction	state, transformation	-	status	state	-	result	-	*extension of changingAttributedThing*

language [141]. The meta-model level of UEML is based on the BWW ontology but the UEML approach encourages the addition of new ontological classes, properties, states and transformations when describing modeling constructs. The growth of the BWW ontology is referred to as the **UEML ontology** [142]. Figure 3.9 displays the main UEML ontology classes. Due to the extensibility of the core ontology, the UEML ontology is able to cover all the process perspectives through the proper extensions.

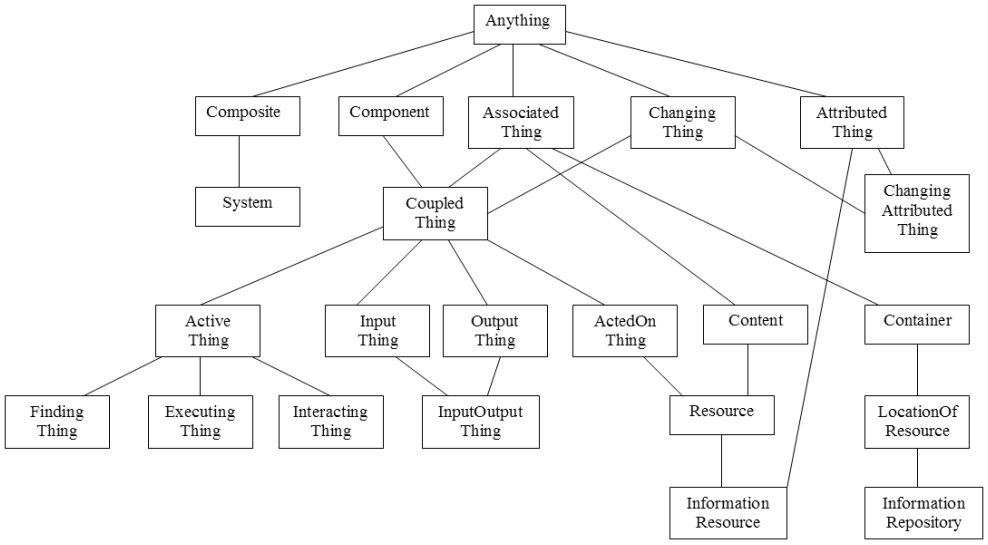


Figure 3.9: Generalization hierarchy of UEML ontology classes [144]

The UEML ontology are represented using OWL Full with the Protégé as the UEMLBase tool. The UEMLBase tool has been used for experimentally describing the following languages: (Class and Activity diagrams of) UML 2.0, CPN (Coloured Petri Net), GRL (Goal-oriented Requirements Language), KAOS (Knowledge Acquisition in autOmated Specification), IDEF3 (Integrated DEFinition methods – Process Description Capture), and, partially, ISO 19440; the result is the UEMLBase content that can be retrieved, visualized and navigated by using the UEMLBase tool functionality [142].

3.2.10 Comparison of process ontology representations

Table 3.2 displays the comparison of ontology representation primitives, and main concepts and representation mechanisms for the process perspectives. The comparison presents how the semantics of process, i.e. the process perspectives are specified in different ontological representation forms. Some of them are specified like a meta-model of process modeling languages, such as PIF, OWL-S, POP*. Some are very general and need to be extended to represent more specific process semantics, such as BWW and UEML. The MIT process handbook and TOVE provide process templates and standardized taxonomy. WSMO is unique from the other ontologies by the use of interfaces to describe the functionality of Web services, so that there are no particular ontological representations for the process perspectives but mainly for the mediation of services.

From the survey, we also found that the formal representation of the ontologies are usually specified through the primitives such as class (sub-class), property (attribute, relationship), and axiom. Standard taxonomy and process templates are also often included in the process ontologies.

3.3 Goal Modeling

Goal is considered as an important concept to represent business strategy and support decision making. Goal models are used to represent business objectives for stakeholders and also to drive the elaboration of business requirements for business analysis and executions. The research on goal-driven methodologies brought out some goal modeling approaches and languages. From the literature, goal modeling is mostly designed as a requirements engineering method, such as KAOS [21], i*/GRL [55], GBRAM [56]. With the development of Web services, goal modeling is associated with Web services modeling for goal-driven services discovery, such as WSMO [210].

3.3.1 KAOS (Knowledge Acquisition in autOmated Specification)

KAOS [21] is a goal-driven methodology which is based on a rich framework for requirements elicitation, analysis and management. Goal modeling in KAOS is based on a goal-oriented process. The process starts with goals which are easy to understand and communicate. They describe the problems instead of the solutions, and can be refined to different levels of abstraction for incremental analysis process. With KAOS, functional and non-functional requirements are formally modeled in terms of goals, constraints, objects, operations, agents, etc. Goals are from available sources and asking why and how questions; objects, relationships and attributes are derived from the goal specifications; agents are identified and are assigned alternative responsibility based on goals; operations and domain pre-/postconditions are also identified from the goal specifications [192].

3.3.2 i*/GRL (Goal-oriented Requirement Language)

GRL is an extended version of i*, a language for supporting actor/role oriented modeling, goal-oriented modeling and reasoning of requirements, especifally for dealing with non-functional requirements [40]. Main concepts associated with goals are represented by intentional elements, intentional relationships and actors. The intentional elements in GRL are goal, task, softgoal, belief and resource. The intentional relationships include means-ends, decompositions, contribution, correlation and dependency. An actor is an active entity that carries out actions to achieve goals. Goals and requirements are analyzed through modeling strategic actor relationships. The relationships are modeled through Strategic Dependency (SD) models and Strategic Rational (SR) models. In a SD model, one actor (the depender) depends on another actor (the dependee) for a dependum. According to different types of dependum, several types of dependencies are distinguished in the Strategic Dependency model, namely goal dependency, task dependency, resource dependency, and softgoal dependency. The SR model provides a more detailed level of modeling by looking "inside" actors to model internal intentional

relationships. Intentional elements appear in SR models not only as external dependencies, but also as internal elements arranged into (mostly hierarchical) structures of *means-ends*, *task-decompositions* and *contribution* relationships [212]. The *means-ends links* provide understanding about *why* an actor would engage in some tasks, pursue a goal, need a resource, or want a soft goal; the *task-decomposition links* provide a hierarchical description of intentional elements that make up a routine; the *contribution links* provide elaboration of the effects from intentional elements to softgoals.

3.3.3 GBRAM (Goal-Based Requirements Analysis Method)

GBRAM [56] addresses the critical nature of the discovery process in goal analysis. GBRAM can be used in goal analysis and goal refinement/evolution. It defines a top-down analysis method refining goals and attributing them to agents starting from inputs such as corporate mission statements, policy statements, interview transcripts etc. Therefore operationalization process is provided for defining a goal with enough detail so that its subgoals have an operational definition. GBRAM distinguishes between achievement goals and maintenance goals. Achievement goals are objectives of functional processes. Maintenance goals are those goals that are satisfied while their target condition remains true. They tend to be operationalized as actions or constraints that prevent certain states from being reached. Achievement goals usually map to actions that occur in a system, while maintenance goals map to nonfunctional requirements.

In GBRAM, agents are the entities or processes that seek to achieve goals within an organization or system based on the implicit responsibility that must assume for the achievement of certain goals; constraints place conditions on the achievement of a goal; goal obstacles are behaviors or other goals that prevent or block the achievement of a given goal. GBRAM supports goal decomposition to subdividing a set of goals into a logical subgrouping.

3.3.4 Goal modeling in EEML

Goals in an EEML model can be decomposed into sub-goals and goal connectors such as "and", "or", or "xor". Those connectors can be used to specify logical relationships between the goal and its sub-goals. Besides, EEML provides more connecting relation types to associate goals with the elements in an EEML process model. The connecting relationships in the EEML goal modeling are listed in Table 3.3.

3.3.5 Goal specification in WSMO

Goal models in WSMO [210] are descriptions of Web services that would potentially satisfy the user desires. They provide the user view in the Web service usage process. Goal is represented through the capability of the Web services the user would like to have, and the interface of the Web service the user would like to have and interact with. A set of properties strictly belonging to a goal are defined as non-functional properties of a WSMO goal. A goal may be defined by reusing one or several already-existing goals by using goal mediators.

Table 3.3: EEML goal modeling relations

Relation type	Related type	Explanation
Goal connector	Goal	Indicate relationships between goals. A goal connector is used when needing to link several goals
Is source of	person|organization (origin)	The person or organization that is the source of the goal
Applies to	task|resource role (target)	The task or resource role which the goal is relevant for
Is precondition for	input port (target)	A rule to describe more precisely the condition for starting a task
Is postcondition for	output port (target)	A rule to describe more precisely the condition for ending a task
Is decision rule of	decision point (target)	A rule to guide the performance of a decision
Is action rule of	task (target)	A rule for automation of task

3.3.6 Goal modeling and process modeling

Although most goal modeling methods do not directly address business process issues, the modeling primitives in those modeling languages are close to process modeling elements, such as actor, task, resource. However, those goal modeling methods do not provide mechanisms to associate goal models to legacy process models. The i*/GRL approach analyzes requirements with strategic dependencies among goals, actors, tasks and resources. In KAOS requirements specification, goals are refined into constraints, objects and operations which are assigned to agents. Actor or agent, resource or object, task or operation and constraint are usually modeling constructs in most process models. That discloses the underlying relationships between goal models and process models. EEML [59] includes goal modeling as one of its four modeling domains. Different types of connecting relationships between goals and process models are defined in EEML goal modeling. WSMO [210] has goals to represent an objective for which fulfillment is sought through the execution of a Web service. The association of a goal and Web services is established through the capability of Web services and a set of ontology mediators. For a business driven SOA, intentional service modeling [157] describes a service in business terms, i.e. goals and to organize their publication, search and composition of the basis of these descriptions. A goal in the intentional service model is described as a state to be reached or maintained by the service. Operation of a business goal is executed through a composition of derived intentional services from MAP [156], a process model expressed in a goal driven perspective.

3.4 Semantic Annotation Methods and Tools

Semantic annotation is currently regarded as an approach to enable the Web content machine-understandable in order to achieve the Semantic Web applications. Such a kind of approach usually requires an annotation schema or reference ontologies with explicitly-defined, consensually-agreed, and machine-understandable semantics to annotate the content.

Semantic annotation methods have been proposed to be used in giving semantics to unstructured digital resources, such as digital documents, Web pages and images by

providing unified structured metadata. Emerging semantic annotation of Web services is relating concepts in Web services to domain specific ontologies, such as MWSAF (METEOR-S Web Service Annotation Framework) [133].

Manual annotation is a tedious work. Semi-automatic and automatic annotation can facilitate this task. Some efforts on automatic and semi-automatic semantic annotation of textual documents have already done in [32] and [70]. Usually, the text extraction using some lexical analysis and grammatical functions is applied to achieve the semi-automatic annotation. AI technologies and semantic mapping are often used in Web services for the semi-automatic semantic annotation.

3.4.1 MnM

MnM [184] is an annotation tool to support both automated and semi-automated semantic markup of Web pages. MnM integrates a Web browser with an ontology editor and provides open APIs to link to ontology servers and for integrating information extraction tools [193]. With the browser, users can have an overview of the knowledge models which are stored on the ontology server. A set of tags defined in the ontology are selected to annotate segments of text on web pages. MnM uses SGML/XML format tags and inserts them into the document.

A machine learning technique is applied for information extraction in MnM. The manual tagged documents are treated as training corpus over which a learning algorithm is run to learn the extraction rules. Learning can result in a library of induced rules which can be used to extract information from texts. The extracted information is sent to the ontology server and fills predefined slots associated with an extraction template. The slots are the properties of a particular class defined in the ontology. MnM can handle multiple ontologies at the same time and allows inherited definitions to be displayed for ontology editing and browsing. Moreover, MnM can access ontology servers through APIs, and also access ontologies specified in a markup format, such as RDF and DAML+OIL [193].

3.4.2 KIM (Knowledge & Information Management)

KIM [127] is a knowledge and information management platform for automatic semantic annotation, indexing and retrieval of Web documents. The automatic semantic annotation in KIM can be seen as a classical named-entity recognition (NER) and annotation process. The NE (Named-Entity) type is specified through a reference to the KIMO (KIM Ontology) with the focus on named entity classes. KIMO is a light-weight top-level ontology including some basic distinctions between entity types (such as locations, agents, events, situations), real-world entity types of general importance (e.g. mettings, employment positions, commercial, government), and characteristic attributes and relations for featured entity types. The KIM KB (Knowledge Base) aims to provide quasi exhaustive coverage of the most important entities in the world. The entities stored in KB are instances with their proper classes and aliases. The entity descriptions in KB and the KIM ontology are stored in the same RDF(S) repository. Thus, KIM provides for each entity reference in the document (i) a link (URI) to the specific class in the ontology, and (ii) a link to the specific instance in the KB [136].

The dual mapping allows the information extraction process to be improved by providing disambiguation clues based on attributes and relations [153]. KIM uses GATE's document management functionality and other generic NLP components such as Tokenizer, Part-of-Speech Tagger, and Sentence Splitter [136] in the information extraction processing.

3.4.3 AeroDAML

AeroDAML [72] is a knowledge markup tool applying NLP (Natural Language Processing) techniques to automatically generate DAML annotations from Web pages. The natural language information extraction techniques are used to assign words and relationships to ontologies as instances of DAML classes and properties. A commercial information extraction product called AeroText is used in AeroDAML as the NLP component. There is a common knowledge base in AeroDAML which contains domain independent rules for extracting proper nouns and relations. DAML generation components are used to translate the extraction results into a RDF triple model by referencing an ontology. As a result, the RDF model is serialized to a DAML annotation. Two levels of default ontologies can be referenced: the lower level ontology from the common knowledge base and the top level ontology from the WordNet noun synset hierarchy. In addition, a custom ontology can also be input as the reference ontology in the annotation.

3.4.4 OntoMat-Annotizer

OntoMat-Annotizer (OntoMat for short) is a component-based, ontology-driven annotation tool for Web documents. It is the implementation of CREAM (CREAting Metadata for the Semantic Web), a framework for an annotation environment. In the CREAM method, an ontology is represented by statements expressing definitions of DAML+OIL classes and properties. Based on those definitions, annotations are a set of instantiations of classes, attributes and relationships that attached to an HTML document. Annotations are described through the metadata which are derived from ontology definitions. Such metadata is stated as relational metadata because it contains relationship instances [49].

OntoMat supports both manual and semi-automatic annotation. The semi-automatic annotation approach is developed in S-CREAM (Semi-automatic CREAM) The semi-automatic metadata creation takes advantage of information extration techniques to propose annotations to metadata creators. A learnable information extraction component – Amilcare [16] is integrated in S-CREAM.

3.4.5 MWSAF (METEOR-S Web Service Annotation Framework)

MWSAF [133] is a framework for semi-automatic annotation of Web service descriptions with relevant ontologies. MWSAF is developed under the METEOR-S project [187]. METEOR-S provides a mechanism to add data, functional, execution, and QoS (Quality of Service) semantics to WSDL files and a comprehensive framework for Semantic Web composition is provided in MWSCF [166].

Ontologies referenced in the annotation are modeled in DAML, RDF-S, or OWL. WSDL descriptions use XML Schema to provide the basic structure of the data to be exchanged by Web services. Hence, a SchemaGraph is used as a common representation format to facilitate the match between an ontology model and a XML Schema. Through the SchemaGraph, pairs of comparable concepts from the ontology model and WSDL descriptions are found. The semi-automatic annotation of MWSAF is accomplished through *Element Level Match* and *Schema Level Match* between the comparable ontology concepts and WSDL concepts. The element level match is the measure of the linguistic similarity between two concepts based on their names. Some NLP techniques are applied in the element level match, such as NGram, synonym matching, abbreviation expansion, stemming, tokenization, etc. [159]. The schema level match is the measure of structural similarity between two concepts. The structural similarity is calculated by the geometric mean of sub-concept similiary and sub-concept match. Sub-concepts of an ontology concept are the property concepts of the ontology class, while sub-concepts of WSDL descriptions are the elements of a "complexType" entity in the XML Schema. A ranking mechanism to find the best matching is employed after a WSDL concept has been compared against all the concepts from the ontology.

3.4.6 SAWSDL (Semantic Annotations for WSDL and XML Schema)

SAWSDL [201] as a W3C recommendation is aimed to provide an annotation mechanism to add semantics to WSDL descriptions and XML Schemas. The annotation mechanism is accomplished through the extension attributes of WSDL and XML Schema elements to reference concepts from semantic models. A semantic model could be any machine-interpretable representations about an area of knowledge or some parts of the world, e.g. ontologies or mapping documents. The extension attributes fit within the WSDL 2.0, WSDL 1.1 and XML Schema, so that the annotations are embedded in WSDL files and XML Schema. The semantic annotation results are supposed to be used in the Web service publishing, discovery and composition. Moreover, the data mapping of XML Schema types to and from an ontology by the annotation can be used for invocation.

SAWSDL defines the extension attributes **modelReference**, **liftingSchemaMapping** and **loweringSchemaMapping**. The first one is used to annotate a concept in some semantic model to a WSDL or XML Schema component. The component include XML Schema type definitions, element declarations, attribute declarations as well as WSDL interfaces, operations, and faults. The latter two are added to XML Schema element declarations and type definitions for specifying the URIs that reference mapping definitions. **liftingSchemaMapping** defines how an XML instance in a schema is transformed to data in a semantic model. Reversely, **loweringSchemaMapping** denotes how data in a semantic model is transformed to XML instance data [201]. The annotation attributes have prefix "sawsdl", e.g. `sawsdl:modelReferece`, `sawsdl:liftingSchemaMapping`.

In the SAWSDL specification, multiple semantic annotations are allowed to be associated with WSDL elements. No restriction is made on the ontology expression language and mapping languages.

3.4.7 Semantic annotation schema in the INTEROP project

Semantic annotation of enterprise models is the research topic of the Task Group 4 (*Semantic Enrichment of Enterprise Modeling, Architectures and Platforms*) of the INTEROP project. The purpose of the annotation is to enhance the interoperability of enterpise models in the applications of model exchange, model transformation and model traceability.

As the research results, a semantic annotation schema is proposed based on a set of semantic annotation concerns. The following elements are defined in the annotation schema: identification of the annotation, annotation types, textual description of the annotation content, identification/location of the annotation target, formal definition of the annotation content (e.g. expression of complementary information) [143]. The annotation schema is displayed in Figure 3.10.

<*Annotation*

Annotation-Id: Annotation identifier

Unformal Content = Natural language comments explaining the intent of the annotation;

Annotation Type = for example, it could be one of:

- *Decoration*: annotations are comments associated with the annotated object;
- *Linking*: annotations are links;
- *Instance Identification*: the annotated object is an instance of a given class and the annotation content (*Ref2Ontology*) is a link to that class (*uri*);
- *Aboutness*: no assertion is made about the existence of an instance of the concept, but there is a loose association with the concept;
- *Pertinence*: the target of the annotation may be of interest for the annotated object.

Ref2Ontology = (reference (*uri*) to the ontology concept(s) related to the current model concept)

Constraints = (might be written using OCL, with references to the ontology or to the meta-model, when these ones are represented as UML class diagrams).

/>

Figure 3.10: Schema for semantic annotation of enterprise models [143]

The schema is extendible. It is extended in [145] by three complementary annotations, i.e. structural annotation, lexical/terminological annotation, and behavior annotation. Structural annotation concerns the semantic mapping of modeling constructs at the meta-model level; lexical/terminological annotation refers the names associated with the constructed artifacts to a commonly agreed definition of terms; behavior annotation is used to explicit specify the business logic, the procedures, the rules and the policies of the annotated object.

No specific tool is developed to implement this approach. The automation of the annotation process is neither concerned in this project.

3.4.8 ASTAR

A* is the ATHENA semantic annotation system, which is partially supported by the ATHENA project [57]. A* is a Web application connected with ATHOS tool and with THEMIS tool — two tools are used to model the resources in RDF repositories. Ontology-based semantic annotation in this system is aimed at reconciling semantic heterogeneity among business and technical resources. The following semantic mismatch cases are identified in Table 3.4 [145].

A* allows four increasing levels of annotation, namely terminological, path, simple and full level annotations.

- Terminological Annotation. By such annotation, terms are associated with a set of terms from the ontology. Usually naming mismatch, content mismatch and abstraction mismatch can be reconciled by terminology annotation.

- Path Annotation. The structure of the schema and the ontology are compared. Complete paths, from the root element to the leaves, of the annotated schema are associated with a set of paths of the ontology. Such annotation can solve the semantic problems such as structural mismatch, subClass-Attribute mismatch, schema instance mismatch and content mismatch.

- Simple annotation. At this level it is possible to specify the type of mismatch that each annotation intends to cover. Such information will be re-used for Attribute Granularity mismatch; Encoding mismatch; Precision mismatch, among others.

- Full Semantic Annotation. It is a complex annotation including OWL expressions.

3.4.9 Discussion on the survey results of annotation tools and methods

Most of annotation tools and methods in the survey are for unstructured Web textual resources, such as MnM, KIM, AeroDAML and OntoMat-Annotizer, because there have been many research activities and development in this area. For those unstructured information, the annotation is to structure and supply semantic descriptions for information. Information extraction is one of the techniques often applied for the automation of the semantic annotation of the textual resources. We have also surveyed the semantic annotation of structured information — WSDL information for the Web services discovery and composition, such as MWSAF and SAWSDL. Since WSDL is already structured, the annotation is to build the mapping between the structured information and the ontological concepts and relations. Some NLP techniques can also be employed to automate the mapping. Very few work has been completed on the semantic annotation of semi-structured information. Having the both characters of the unstructured and the structured, the semantic annotation of semi-structured information should be able to formalize the structure of information and also build the links between information and ontology. We has included the related work in the INTEROP project, which presents some initial ideas and surface results about the semantic annotation of enterprise models. Another related work — A* provides a tool to annotate models in order to develop future model transformations. The limitation of A* is that

Table 3.4: Lossless mismatches and examples

Name	Description	Document Schema	Reference Ontology
Naming	different labels for the same concept	request for quotation indicated as: RFQuote	request for quotation indicated as: RFQ
Attribute Granularity	the same information is decomposed into a different number of attributes (or sub-attributes)	Telephone is represented as a single string	Telephone is a composition of hasPartCountryCode, hasPartAreaCode, hasPartLocalPhoneNumber
Structuring	different design structures in organizing the information	Buyer is directly linked to PurchaseOrder	Buyer is linked to the Parties concept and Parties is linked to PurchaseOrder
SubClass-Attribute	an attribute, with a predefined value set, is represented by a set of subclasses, one for each value	the type RawMaterial can be represented as an enumeration ('iron', 'copper')	¡ or as two subclasses: iron subClassOf RawMaterial, copper subClassOf RawMaterial
Schema-Instance	data holding schema information. In the DS example, the semantics of the second field (Name) depends on the value of the first field (Role). In the RO, the semantics is fully captured in the attribute names.	ContacInfo are represented as a pair: Role and Name. While Role is an enumeration = 'Director', 'Secretary', Name is a string	ContacInfo has one field for each Role, which is indicated in the attribute name: DirectorName, SecretaryName, both strings
Encoding	different formats of data or units of measure (a Weight expressed in different unit of measure)	Weight expressed in ounces	Weight expressed in kilograms
Content	different content denoted by the same concept - typically expressed by enumeration (the concept described by different enumeration items)	PaymentTerms=(30days, 60days, 90days)	PaymentTerms=(30days, 45days)
Coverage	presence/absence of information (a given information is considered in the RO, but not in DS)	preferredDeliveryDate is not considered in the DS	preferred delivery date is present in the RO
Precision	the accuracy of information	Size of a pallet expressed by three integer values: height, length, width	Size of a pallet expressed by a constant conventional value: (small, medium, large)
Abstraction	level of specialisation/refinement of the information: generic vs detailed representation	DeliveryTerms	DeliveryTerms gets specialized into: NationalDeliveryTerms and InternationalDeliveryTerms

models must be expressed in RDF format and based on ATHOS metamodel. Moreover, the full semantic annotation in OWL expressions are not supported yet.

3.5 Requirements for Semantic Annotation Systems

Based on the above survey, we have identified a list of requirements for a semantic annotation system, some of which we are targeting in our work.

1. The system should be able to present and parse annotation source models which are originally in certain formats and representations.
2. The system should provide an ontology browser for the overview and manipulation of ontological knowledge.
3. Semantic annotation schema or metadata should be supplied (pre-defined) or generated (ontology-based) by the system.
4. The annotation procedure should be easily manipulated by users with the system, i.e. easy to locate "annotatee" (e.g. entity in source model) and "annotater" (e.g. concept in ontology) during the annotation.
5. The system should support the maintenance of annotation results (e.g. embedded vs. stand-off annotation).
6. Multiple-ontology references (e.g. different levels of ontologies) might be supported in the system.
7. Different types of annotations (e.g. instance identification, URI links, and other semantic relationships) might be supported.
8. Semi-automation or automation of annotation might be considered in the system.
9. The system might be able to serialize annotation for reuse of annotation results in different systems.
10. It might be possible to conduct semantic inference among ontology-based semantic annotations.

3.6 Summary

In this chapter, we has investigated the diversity of the modeling constructs defined in a number of existing business process modeling languages. The modeling constructs have been categorized according to the process perspectives. The categories illuminates the possibility to map the modeling constructs between different modeling languages in each process perspective. We have also surveyed a number of different process ontologies. Through the survey, we have compared the ontological representations of process perspectives in different process ontologies. The comparison results provide some principles of ontological representations of a process ontology which we can apply in our process ontology proposal for the semantic annotation purpose.

In the goal modeling survey, we have found some overlapping modeling constructs in goal modeling and process modeling, such as actor or agent, resource or object, task or operation and constraint. That discloses the underlying relationships between goal models and process models. Connecting relationships in EEML goal modeling provide more explicit view of such links between goals and processes.

A list of requirements have been identified based on the survey of the annotation tools and methods. Although they are not tailored to the semantic annotation of process models, they provide a good baseline — what a semantic annotation tool should look like and how to develop an ontology-based semantic annotation approach.

Part II

Design and Application

Chapter 4

Semantic Annotation Framework for Process Models

As we have discussed previously, process properties of a business process model are represented by meta-model elements/modeling constructs of a modeling language, and model solutions are represented by model elements composed of the modeling constructs. Common expressions of process properties and context references make the heterogeneous process models reconciled and comparable. The enhanced semantic interoperability can facilitate the management and reuse of those process models. In this chapter, a basic semantic annotation framework [94] [92] [93] for process models is presented. The framework provides a common semantic schema to express process properties and reference the heterogeneous solutions of business process models to a consensual representation of systems' context.

Before we present our framework, we discuss the theoretical basis of this work, and then we elaborate the structure and components of the framework. Under such a framework, a general process ontology is introduced and used as the reference ontology to annotate process properties. A set of semantic mapping rules are defined as the annotation method, and a semantic annotation model is introduced as the annotation schema.

4.1 Theoretical Basis

The theoretical basis underlying our proposal is Ogden & Richard's semiotic triangle [121], which has been mentioned in section 2.2.1. We use the semiotic triangle to analyze semantic heterogeneity of process models. Ontology plays a major role in our semantic annotation approach, which is also fitted into the semiotic triangle. Thus, in this section we discuss about the relationship between the process models and ontology in the context of semantic interoperability based on the semiotic triangle.

4.1.1 Semiotic triangle

The semiotic triangle is applied in both information modeling and philosophy disciplines. The theory describes relationships between reference, sign and concept (refer to

section 2.2.1). Based on the semiotic triangle we can build the relationships between model, meta-model, modeling language and ontology as shown in Figure 4.1. In the triangle, a model is a conceptualization of referents and it is represented as a set of model denotations in a certain modeling language. Model denotations are signs which signify concepts in the model. The model is an instance of a meta-model, and the meta-model defines a modeling language. A concept is a mental perception which is in a human's mind. One concept referring to a referent can be represented differently. On the other hand, representations of different concepts may look similar in two models. The differences are results of the way of conceptualization of modeling or representations of model denotations defined in a modeling language. The semantic heterogeneity existing in different models can therefore be distinguished at the model and the modeling language level (we also call it the meta-model level).

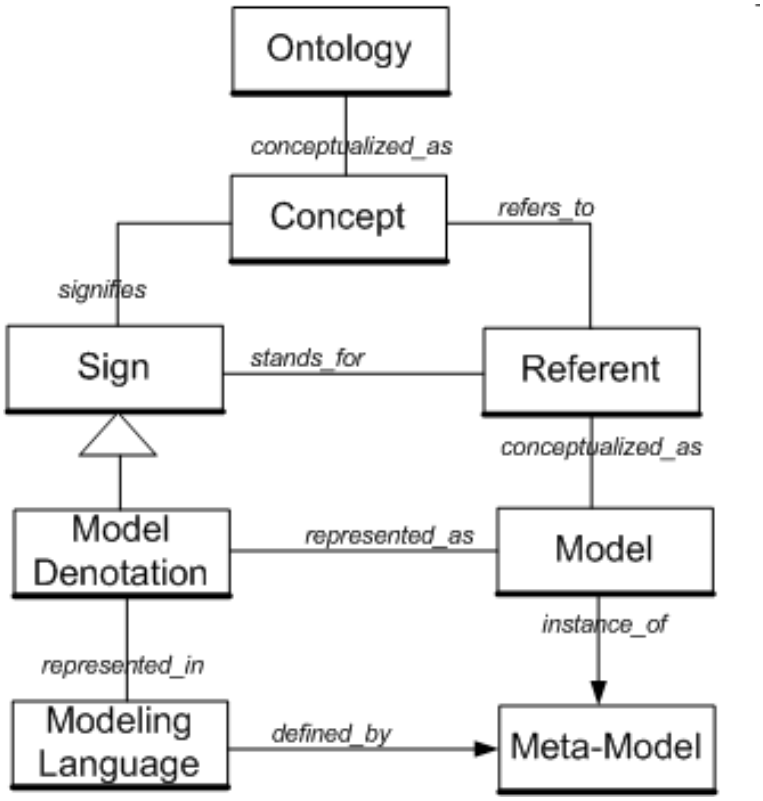


Figure 4.1: Relationships between ontology, model, meta-model and modeling language in the semiotic triangle

In order for a machine to understand the heterogeneous semantics in the models (e.g. various signs of referents referring to the same concepts or synonym signs of referents referring to different concepts), a common understanding of concepts has to be formalized in a machine-interpretable way. An ontology is created for this purpose. In the semiotic triangle, the semantics of concepts are formally in a standard i.e. they are represented as an ontology. Ontologies aid the sharing of knowledge on the basis of the assumption that there is a single reality and the sharing is a matter of aligning the way different people or systems think about it [70]. Therefore, in the semiotic triangle

concepts can be conceptualized as an ontology in a consensual way.

A meta-model is also a model – a model of the modeling language. For a meta-model, a modeling referent is a meta-model element or modeling construct, and a modeling sign is the notation of a meta-model element. A modeling sign standing for a meta-model element is used to represent a modeling concept in a certain modeling domain. Semantic heterogeneity problems will still occur on the meta-model level provided that a meta-model is a model. We assume that a modeling ontology can provide the common conceptualization of modeling concepts referring to the same modeling referents. According to Leppänen's OntoFrame [87], a meta-model can be adapted from a modeling ontology. The semantic heterogeneity of modeling languages can therefore be reconciled through the modeling ontology. Based on this theory, we annotate the meta-model of a modeling language with the modeling ontology in this research. The relationship between modeling language, meta-model and modeling ontology is displayed in Figure 4.2.

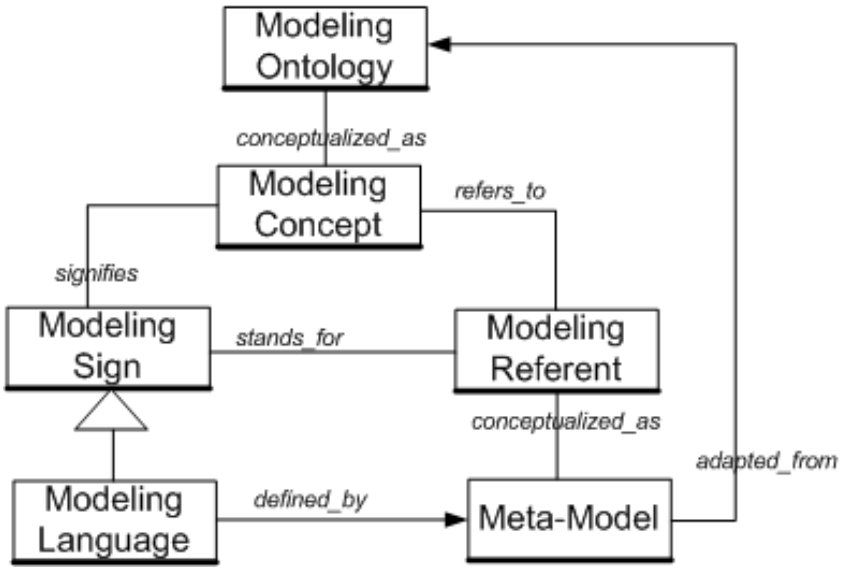


Figure 4.2: Relationships between modeling ontology, meta-model and modeling language in the semiotic triangle

4.1.2 Semiotic triangle for process modeling

In the context of our research, process meta-model, process modeling language, process model, process model denotation and process ontology are specified in the semiotic triangle (Figure 4.3). We also include model levels in the figure to explicate the positions of those modeling concepts. The model levels are adapted from the model level ontology in [87] to show the different levels of process models. Process meta-model and process model are both models. Process meta-models are categorized at the meta level, and process models are at the type level. In this research, we focus on process models at the type level including their meta models at the meta level. Process models at the type level are the resources of process knowledge in our context.

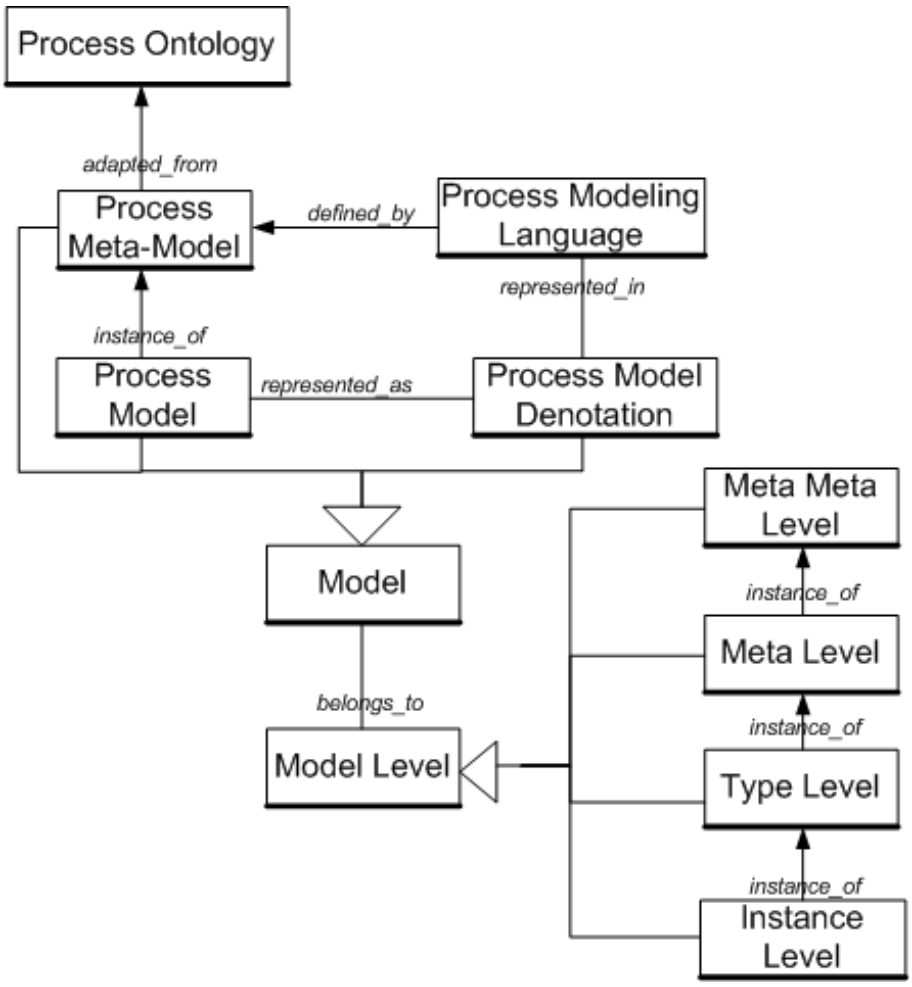


Figure 4.3: Relationships between model level, process ontology, process model, process meta-model and process modeling language (adapted from model level ontology in [87])

4.2 Overview of the Framework

We use ontology-based semantic annotation to deal with the interoperability of heterogeneous process models. The semantic annotation approach is developed and refined in a semantic annotation framework, which contains profile annotation, meta-model annotation and model annotation.

4.2.1 Ontology-based annotation

In this approach, ontology provides a standard representation of terminology and conceptualization to reconcile the semantic heterogeneity of different models. Corresponding to the two-level semantic heterogeneity — meta-model and model level, we need two kinds of ontology: an ontology to relate constructs across different process modeling languages, as well as an ontology to reconcile and align domain specific terminology used in process models. A general and global process ontology can be used to annotate

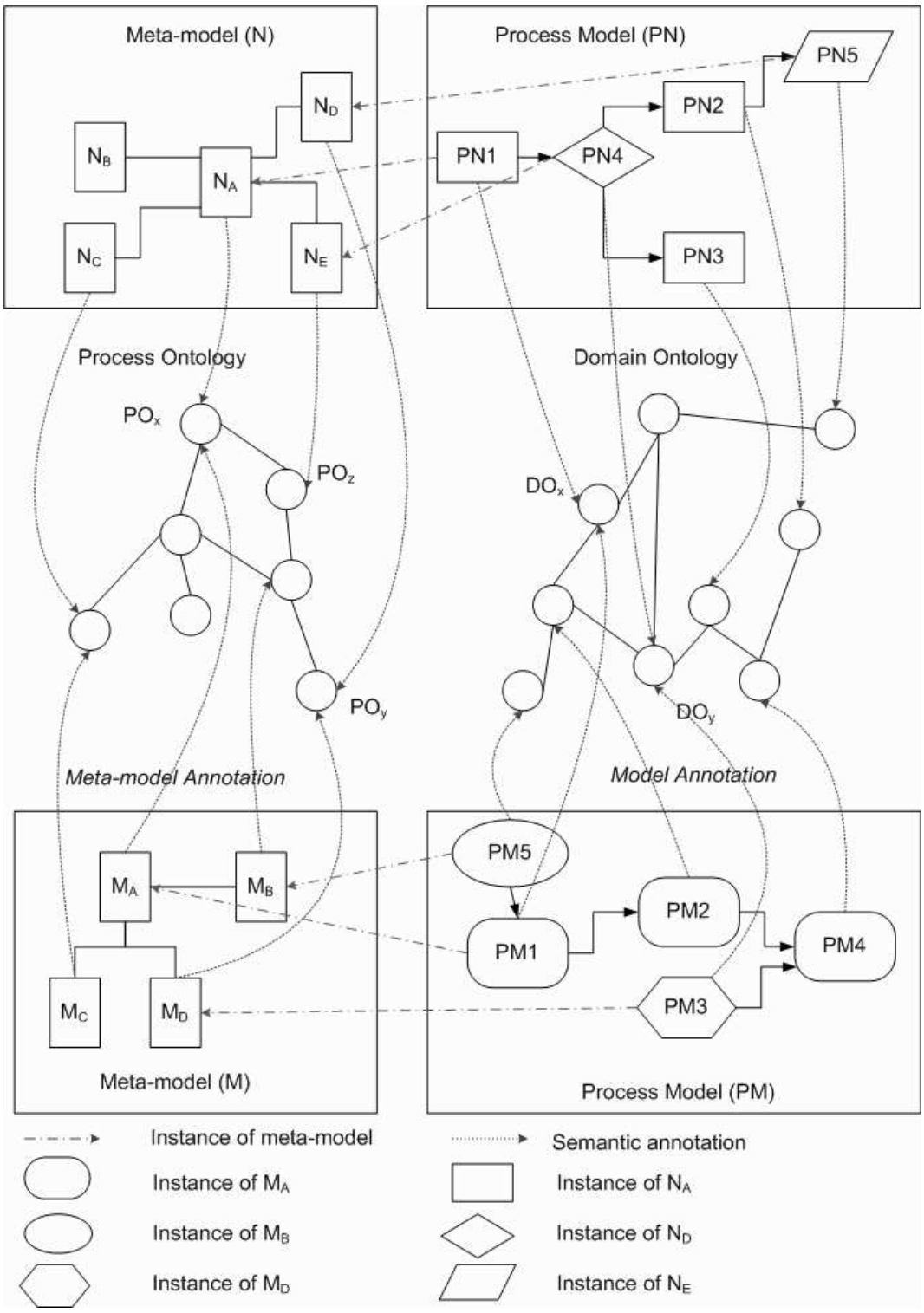


Figure 4.4: Semantics reconciliation of process models through ontology-based annotation

the specific and local process meta-model element. The concepts in a local process modeling language usually have the `is_a` relationship with the concepts from a global process ontology. If the global process ontology can cover all the semantics of the local modeling language (i.e., the local modeling language is a subset of the global process meta model ontology), the annotation work can keep the complete semantics of the local modeling language. If the local process modeling language is more expressive than the global process ontology (i.e., the set of notations of the local modeling language is larger than the global one), the model will lose some of the semantics which are not annotated by the global process ontology. For those lost semantics, referencing the meta-model of the modeling language is necessary.

A domain ontology provides the standardized terminology and conceptualization of a particular domain. We suppose that a certain domain ontology is agreed and used to reconcile the semantics of model contents. The ontological concepts are referenced by model contents through simple URIs or semantic relationships.

Figure 4.4 provides a depiction of the basic idea of ontology-based semantic annotation. Process models PM and PN are respectively represented in process modeling languages M and N. The notations used in process models are instances of modeling concepts defined in meta-models. For example, the notation for $PM1$, $PM2$ and $PM4$ is an instance of M_A, and $PN1$, $PN2$ and $PN3$ are instances of N_A. The semantics of the notations are defined in meta-models. In order to reconcile the semantics of the notations, a process ontology is used to mediate the mapping between meta-models M and N. For instance, M_A and N_A are annotated with a same ontological concept PO_x in the process ontology, which means notations of M_A in PM ($PM1$, $PM2$ and $PM4$) stand for the same meta-model element as notations of N_A in PN ($PN1$, $PN2$ and $PN3$). However, model contents in those notations should also be semantically reconciled to understand both models, i.e. what domain phenomenon the notations represent. A domain ontology is referenced to annotate model content. Usually the model contents represented by the same meta-model element are comparable. For example, the contents of $PM1$, $PM2$ and $PM4$ should be reconciled with $PN1$, $PN2$ and $PN3$. Since $PM1$ and $PN1$ are annotated with a same domain ontological concept DO_x, they are interpreted as semantically equal concepts when comparing the two models. If the contents of two model constructs with different meta-model elements reference to a same domain ontological concept, they are not considered as semantically equal. Although the contents of $PM3$ and $PN4$ have a same ontology reference DO_y, they are not same because they do not share a same modeling ontological concept at the meta level (PO_y for $PM3$ and PO_z for $PN4$). In this way, various representations of process models are aligned and reconciled through the reference ontologies, i.e. the process ontology and domain ontology.

4.2.2 Annotation aspects

Annotation in this work is intended to expose the process knowledge carried by heterogeneous process models. In general, the annotation should be able to provide both an overview and a sophisticated comprehension of the process knowledge. We therefore include a profile annotation to profile process models as products. In the profile annotation, a set of metadata is defined to describe the significant characteristics of a

process model from a general view. Comprehending a process model depends on the understanding of the semantic representations of process modeling languages and model contents. We use meta-model and model annotation to catch and represent the process knowledge in the models through building reference relationships between models and ontologies.

Technically, process models to be annotated are serialized into XML representations. During the annotation, the meta-model schema of process models are mapped to a common process ontology in OWL. Model contents in the XML file are transformed into the OWL annotation model. Domain ontologies are referenced by the model contents in the OWL annotation model.

4.3 Profile Annotation

Basic and characteristic features of a process model are described by a set of metadata in a profile annotation. A profile contains information about a process model such as the problem domain of the model, the name of the model and the location of the model etc. We categorize metadata elements for the profile annotation according to four types of metadata — administrative, descriptive, preservation, technical and use [4] (see Table 4.1).

- **Administrative**: Metadata used in managing and administering information resources.
- **Descriptive**: Metadata used to describe or identify information resources.
- **Preservation**: Metadata related to the preservation management of information resources.
- **Technical**: Metadata related to how a system functions or Metadata behave.
- **Use**: Metadata related to the level and type of use of information resources.

Usually, the profile information is manually input by annotators. In order to prevent the semantic heterogeneity in the input values, a set of categories and taxonomy are chosen to be referenced for profile annotation. For example, a list of standard abbreviations for the natural languages (e.g. "EN" for English) are predefined for the value of `dc:language`.

4.4 Meta-model Annotation

Semantic heterogeneity problems of diverse process modeling languages can be solved through mapping two modeling languages to each other, or mapping languages to one common process modeling language. Meta-model annotation in our framework deals with this problem by mapping different meta-mode elements to ontological concepts in a process ontology. A process ontology is not supposed to be a new process modeling language but provide a way to represent process knowledge. Therefore, compared with certain process modeling languages, the process ontology provides only general semantics for process modeling but essential semantics for process knowledge.

Table 4.1: Metadata for profile annotation

Type	Element Label	Cardinality	Description
Administrative	profileAnno:model_version	{0,1}	Version of the process model
	profileAnno:version_date	{0,1}	Date when this version of the process model was created
	profileAnno:URI	{1,1}	A Uniform Resource Identifier (URI) is a compact string of characters for identifying the process model
	profileAnno:URL	{1,1}	A Uniform Resource Locator (URL) where the process model can be accessed
	dc:creator	{1,N}	Persons or organizations primarily responsible for making the structure and content of the process model
	dc:publisher	{1,1}	A person or an organization responsible for making the process model available
	dc:date	{1,1}	Date of availability of a process model
	dc:language	{1,1}	Natural language of contents of a process model
Descriptive	dc:title	{1,1}	An name given to the process model
	dc:description	{1,1}	An account of contents of the process model
	profileAnno:category	{1,1}	Genre of contents of the process model
	profileAnno:domain	{1,1}	Domain of contents of the process model
	profileAnno:domainOntology_URI	{1,1}	A Uniform Resource Identifier (URI) of the domain ontology used in the model annotation
	profileAnno:domainOntology_URL	{1,N}	A Uniform Resource Locator (URL) where the domain ontology can be accessed
Preservation	profileAnno:documation	{0,N}	Different formats of description of the content of the process model (e.g. an abstract, an overview of how the process model has been developed, a graphical representation of the process model, etc.)
Technical	profileAnno:modeling_language	{1,1}	A modeling language the process model was created in
	profileAnno:language_version	{0,1}	A version of the modeling language
	profileAnno:modeling_tool	{1,N}	Modeling tools (including the version of the tools) which support the process model
Use	profileAnno:used_in	{0,N}	Projects and/or applications in which the process model is used
	profileAnno:examplemodel_URL	{0,N}	Uniform Resource Locator (URL) where examples of the process model can be accessed

4.4.1 GPO (General Process Ontology)

For meta-model annotation, a process ontology is an explicit and formal specification of concepts which are used to model processes in general. It is supposed to provide common and core semantics of process modeling constructs. We have investigated a number of process modeling languages and process ontologies in Chapter 3. Most of them were not initiated for annotation purposes and are not represented in OWL, except OWL-S. OWL-S is designed to describe processes of Web services in OWL. Although a process model in our approach may be regarded as a service, the difference between a Web service and a process model is that a Web service is executable and uses programming-like control constructs as their basic building blocks which are inadequate for all the modeling issues, especially for the type level of modeling. Therefore, we propose a new ontology for meta-model annotation purpose in our framework.

It is difficult to build an ontology to cover all the semantics of various modeling constructs, which would be very complicated. To facilitate the annotation, representations of the ontology should be easily understandable and able to preserve enough semantics of process knowledge. As Musen stated [109]: "Although no simple predicate tells us unambiguously whether a particular specification is an ontology, we can still agree on certain things. We can agree that ontologies enumerate the salient concepts in an application area." A process ontology therefore consists of core concepts which can present the process perspectives specified in Chapter 3.

We set a **design principle** of the process ontology as *simple but comprehensive for modeling process knowledge*. Based on such principle, we create an ontology for process modeling, namely General Process Ontology (GPO).

Main concepts defined in GPO

The main concepts in GPO are ***Activity, Artifact, Actor-role, Input, Output, Precondition, Postcondition, Exception*** and ***WorkflowPattern***. GPO consisting of those concepts and their relationships are represented using RML [168] in Figure 4.5.

- *Activity* is a central concept which composes a process. *Activity* is a synonym of *process*. However, we also found *Process* is seldom a construct or just a package construct in most process modeling languages because it is obvious that a process model describes processes. *Event* is used to trigger *Activities* but it is sometimes used like an *Activity*. It also sometimes mixed with state. In order to avoid any ambiguity of those concepts, *event* and *state* are not included in GPO. We still think *event* and *state* are crucial for process models at the execution level but less at the conceptual business process level. We therefore use the concept *Activity* in our general process ontology. An activity may be an atomic activity or a composed activity represented by the aggregation relation between activities, i.e. one activity is a *subActivity* of another activity.

- *Artifact* represents something involved in an activity such as product, information, tool and software.

- *Input* and *Output* of an *Activity* specify what are consumed and produced by this activity respectively. The way an *Artifact* is involved in an activity can be

specified through its relations with inputs or outputs.

- *Actor-role* is the one who interacts with an activity. Although actor and role are two different concepts, we combine these into one to represent that a role is acted by an actor and an actor is a person, or an organization or a system that operates the activity. The role is not acted by any artifact which is distinguished from the definitions of role in some modeling languages, e.g. ResourceRole in UEML.

- *Precondition* and *Postcondition* represent respectively constraints before an activity starts and after an activity ends. Pre- or Post- conditions may also directly constrain inputs and outputs of activities.

- *Exception* provides additional information about failures of a process or any exceptional cases in a process. The exception can be handled by activities.

- *WorkflowPatterns* represent orderings of different activities. WorkflowPattern can be refined into several specific patterns according to the workflow patterns defined in [191], such as *Choice* (*Exclusive Choice*, *MultipleChoice*, *ParallelSpit*), *Merge* (*SimpleMerge*, *MultipleMerge*, *Synchronization*) and *Sequence*, which are basic control workflow patterns supported by most process modeling languages with logical symbols like `AND`, `OR` and `XOR`. The aim of including workflow patterns in GPO is for the user to navigate preceding, succeeding, synchronizing and exclusive activities, because those workflow patterns denote semantics of process orders.

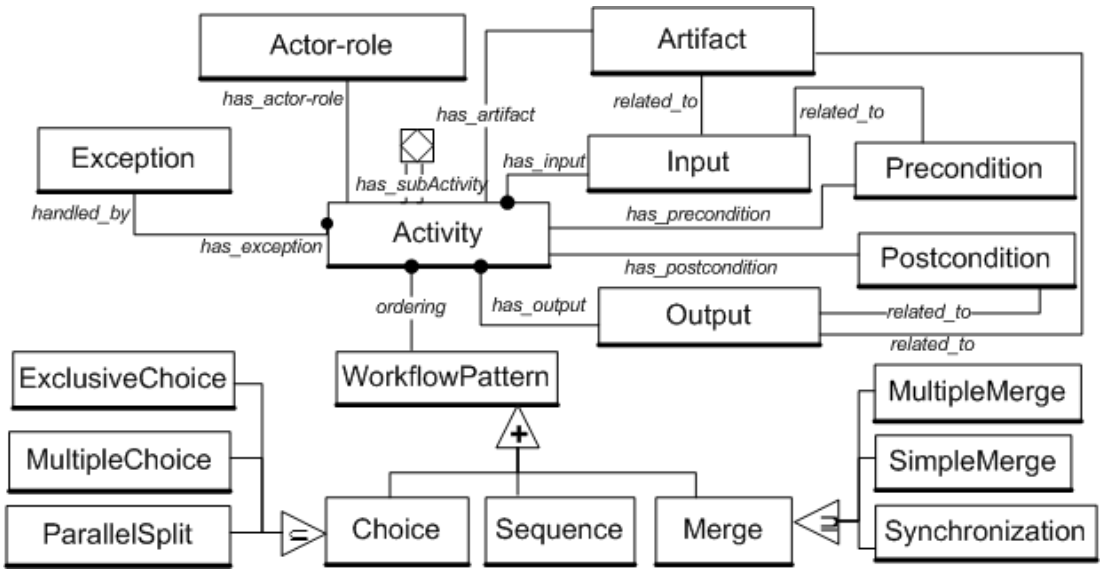


Figure 4.5: General Process Ontology

Process perspectives in GPO

With regard to the process perspectives from the paradigm of BPM systems (defined in Chapter 3), GPO has following correspondences: GPO uses *Activity* together with

Input and *Output* to represent the operational/function perspective. Decomposable *Activities* and their *subAcitivities* can be applied to specify the structural perspective. The resource and organizational perspectives can be presented through the specifications of *Artifact* and *Actor-role* respectively. The control perspective of a process is represented by *Precondition*, *Postcondition* of *Activities* and a set of *WorkflowPatterns* and *Exception* to link *Activities*. No particular concept is identified to represent the data transaction perspective but the links between *Artifact* and *Input* or *Output* can be used to specify changes of data values associated with a certain *Artifact*.

4.4.2 Mapping rules in Meta-model annotation

GPO is a mediator for semantic reconciliation of process modeling concepts and it should not be seen as a new process modeling language, but a means to annotate the process modeling constructs. In a meta-model annotation a process modeling language is annotated manually by annotators. The procedure of meta-model annotation is in fact to set mapping rules between the GPO concepts and process modeling language constructs or meta-model elements. The mapping rules consist of both one-to-one and one-to-many correspondences between the GPO concepts and modeling language constructs or meta-model elements. More complicated cases might occur, such as a correspondence between a GPO concept and a combination of several modeling constructs or meta-model elements. To define mapping rules for different cases, we categorize three types of modeling constructs — *AtomicConstruct*, *EnumeratedConstruct* and *ComposedConstruct*. Each single modeling construct is an *AtomicConstruct*, an *EnumeratedConstuct* is a list of several *AtomicConstructs*, and a *ComposedConstruct* is composed of several *AtomicConstructs*.

Mapping Rules:

- One-to-one mapping: a GPO concept is referred by an *AtomicConstruct* (e.g. `GPO:Activity`) is mapped to `EEML:Task`);
- One-to-many mapping: a GPO concept can be referred by several modeling language constructs respectively which are enumerated in an *EnumeratedConstruct* (e.g. `GPO:Artifact`) can be mapped to `EEML:Information Object` or `EEML:Material Object`);
- One-to-combination mapping: a GPO concept is referred by a combination of those modeling language constructs in a *ComposedConstruct* (e.g. `GPO:WorkflowPattern` is mapped to a combination of `EEML:Flow` and `EEML:Decision Point`).

Different technologies can be employed to represent the mapping rules, such as XML, RDF/RDFS or OWL. An OWL representation is exemplified below. A namespace `metaAnno` is used to encode meta-model annotation in these three mapping cases:

```
<metaAnno:AtomicConstruct rdf:ID=CONSTRUCT_ID>
  <metaAnno:refers_to
  rdf:resource=&GPO_ONTOLOGY#MODELING_ONTOLOGY_CONCEPT/>
  <metaAnno:modeling_language_construct
     rdf:resource=&MODELNG_LANGUAGE#LANGUAGE_CONSTRUCT/>
```

```
</metaAnno:AtomicConstruct>
<metaAnno:EnumeratedConstruct rdf:ID=CONSTRUCT_ID>
          <metaAnno:refers_to
          rdf:resource=&GPO_ONTOLOGY#MODELING_ONTOLOGY_CONCEPT/>
          <metaAnno:has>
            <metaAnno:AtomicConstruct rdf:resource=#CONSTRUCT_ID/>
            . . .
          </metaAnno:has>
</metaAnno:EnumeratedConstruct>
<metaAnno:ComposedConstruct rdf:ID=CONSTRUCT_ID>
  <metaAnno:refers_to
  rdf:resource=&GPO_ONTOLOGY#MODELING_ONTOLOGY_CONCEPT/>
  <metaAnno:composed_of>
    <metaAnno:AtomicConstruct rdf:resource=#CONSTRUCT_ID/>
    . . .
  </metaAnno:composed_of>
</metaAnno:ComposedConstruct>
```

Once the mapping rules are defined for a certain process modeling language, process models in that process modeling language can be described by the GPO concepts, i.e. the GPO concepts are used as metadata to annotate process semantics. We call the process models described by the GPO metadata as GPO-annotated process models. A GPO-annotated process model will be formalized in a process semantic annotation model (**PSAM**) in section 4.6.

4.5 Model Annotation

The purpose of model annotation is to annotate *model (contents)* with *domain ontologies*. Models (contents) are instances of the meta-model and those instances usually describe certain domains. The representations of domains are often various due to diverse uses of terminology and conceptualization, resulting in semantic heterogeneity of model contents. Domain ontologies are agreed as standard representations and semantic definitions of domain concepts by annotation users. Semantic heterogeneity of model (contents) can be reconciled by referencing ontological concepts represented in domain ontologies. In model annotation, the annotation method is building semantic mappings or relationships between model contents and domain ontologies.

The mapping method for model annotation

Different mapping strategies can be used between concepts in the model and the domain specific ontology. They can be simple rules applied in meta-model annotation – by referring specific model contents in modeling constructs to corresponding domain concepts. More complicated mappings can be defined through refined semantic relationships between concepts used in models and concepts defined in a domain ontology.

Simple reference. If a simple mapping by reference is applied, it assumes that almost all concepts in the model have equal or approximately equal concepts in the

ontology. The semantic relationship of mapping can be defined as one type – *refers_to*. We have adopted such a mapping strategy in our method for meta-model annotation to build the correspondences of concepts between modeling languages and GPO. In the model annotation, users can choose this strategy provided the concepts in the models are very close to the concepts in the domain specific ontology. The strategy of simple reference is easy to apply and also makes it relatively simple for the machine to infer the mapping relationships without complicated algorithms.

Refined semantic relationships. Concepts used in process models are variously defined initially for different projects. Therefore, it might be difficult to find equally defined concepts in the domain specific ontology for process models. However, since they are still within one domain, there must be some semantic relationships between the concepts in models and in ontologies. In order to represent the semantic relationships precisely, we define some refined semantic relationships to link the concepts between models and ontologies for the model annotation.

The model annotation can be accomplished with the help of meta-model annotation, because the models related to domain information are usually artifacts, actor-roles, activities and exceptions. Artifacts and actor-roles are usually static concepts in the domain ontology. Activities and exceptions are usually related to the task concepts in the domain ontology.

Semantic relationships and the corresponding annotation denotations generally used for the model annotation are listed in Table 4.2. Both synonym and polysemy relationships are symmetrical. Note that *same_as* and *different_from* are not at the individual level like the OWL individuals relationships. Hypernym and hyponym are inverse relationships. Usually concepts in the ontology are more general, while model concepts are relatively concrete for specific projects. Thus, *kind_of* is more often used than *superConcept_of* when annotating model concepts with ontology concepts. We provide more human sense expressions of meronymic relationship for artifacts, actor-roles, activities and exceptions respectively. For artifacts, we use *part_of*, e.g. Engine is a part of Airplane; for actor-roles, we use *member_of*, e.g. Airline is a member of Air Alliance; for activities, we use *phase_of*, e.g. Flying is a phase of Traveling; for exceptions, we use *partialEffect_of*, e.g. Payment is cancelled is a partial effect of Booking has failed. The inverse relationship of meronym is holonymic relationship, which is seldom used in this framework because of the similar reason described for the hyponym relationship.

It is possible to represent some of the semantic relationships with OWL expressions, such as `owl:equivalentClass` for synonym, `owl:disjointWith` for polysemy, `rdfs:subClassOf` for hypernym and hyponym. Those OWL expressions can be used to displace the corresponding annotation notations for OWL inference. If an annotation model and a reference ontology are both represented in OWL, the inference can be made on the two OWL files. For example, a concept C_m in the annotation model is specified as *same_as* an ontology concept O_1, and O_1 is defined as `rdfs:subClassOf` another ontology concept O_2 in the ontology. An inference result could be C_m is sub-Class of (*kind_of*) O_2 too.

After the meta-model annotation, the model can be described as a GPO-annotated process model. Accordingly, the model annotation can be done directly in the GPO-annotated process model instead of the original model. Thereby, the model annotation notations will also be formalized in the process semantic annotation model in section

Table 4.2: Semantic relationships, corresponding annotation denotations, and OWL constructs

Semantic Relation-ship	**Annotation Denotation**	**OWL expression**
Synonym	*alternative_name*(terminology level) *same_as*(concept level)	 owl:equivalentClass
Polysemy	*different_from*	owl:disjoinWith
Hypernym	*kind_of*	rdfs:subClassOf
Hyponym	*superConcept_of*	rdfs:subClassOf
Meronym	*part_of*(Artifact) *member_of*(Actor-role) *phase_of*(Activity) *partialEffect_of*(Exception)	
Holonym	*compositionConcept_of*	
Instance	*instance_of*	OWL Individual

4.6.

4.6 Process Semantic Annotation Model

As described in section 4.4, a process model is represented by the GPO concepts after the meta-model annotation. The model annotation is then applied on the contents of the GPO-annotated model. In this section, we formalize the GPO-annotated model as a process semantic annotation model (**PSAM**). The domain ontology for a certain domain is built or selected by domain experts for the model annotation.

Definition 1. ***PSAM*** *containing the concepts of GPO and domain specific ontology is defined as follows.*

$PSAM = (AV, AR, AF, WP, I, O, \Theta^{pre}, \Theta^{pos}, E, PD)$

Where AV is a set of activities composing a process, AR is a set of actor-roles interacting with a process, AF is a set of artifacts participating in a process, WP is a set of workflow patterns, and each workflow pattern denotes an ordering of activities. I is a set of input parameters, O is a set of output parameters, Θ^{pre} is pre-conditions when a process starts, Θ^{pos} is post-conditions when a process ends, E is a set of possible exceptions occurring during a process. PD is a subset of the domain ontology (D) concepts, i.e. $PD \subseteq D$, including static ontology concepts and task ontology concepts.

Definition 2. *An activity can be considered as a simple process. Therefore an annotated activity is described as follows.*

$AV_i = (id, model\_fragment, name, alternative\_name, has\_Actor-role, has\_Artifact, has\_Input, has\_Output, is\_in\_WorkflowPattern\_of, has\_Precondition, has\_Postcondition, has\_Exception, has\_subActivity, same\_as, different\_from, kind\_of, superConcept\_of, phase\_of, compositionConcept\_of, instance\_of)$

Each element in a PSAM model has an *id* and *name* to uniquely identify the element. $Model\_fragment$ is the identifier of model fragment in the original process model for keeping the link between the annotated model fragment and its annotation information. $Alternative\_name$ provides a synonym of the name at the terminology level. Elements $has\_Actor-role$, $has\_Artifact$, has_Input, has_Output, $is\_in\_WorkflowPattern\_of$, $has\_\ Precondition$, $has\_Postcondition$, $has\_Exception$, $has\_subActivity$ denote the relationships between the activity and other related elements according to the GPO definition. The *ids* of the related elements are used in those relationships. We use $same\_as$, $different\_from$, $kind\_of$, $superConcept\_of$, $phase\_of$, $compositionConcept\_of$ to annotate the activities with the domain ontology concepts, i.e. using semantic relationship to map an activity to a concept in the domain ontology. $instance\_of$ is to specify the modeled activity is an instance of the domain ontology class.

Definition 3. *An actor-role is a person, agent or organization that interacts with an activity. The annotated actor-role is represented as follows.*

$$AR_i = (id, model\_fragment, name, alternative\_name, same\_as, different\_from, kind\_of, superConcept\_of, member\_of, compositionConcept\_of, instance\_of)$$

Definition 4. *An artifact is a thing consumed, used or produced in an activity.*

$$AF_i = (id, model\_fragment, name, alternative\_name, same\_as, different\_from, kind\_of, superConcept\_of, part\_of, compositionConcept\_of, instance\_of)$$

Definition 5. *A workflow pattern represents the type of the ordering of activities.*

$$WP_i = (id, model\_fragment, name, alternative\_name)$$

Refined workflow patterns are defined as follows.

$$Choice_i = (id, model\_fragment, name, alternative\_name, has\_inActivity, has\_outActivity, has\_logicConnector)$$

Where the cardinality of $has\_inActivity$ is 1. The $has\_logicConnector$ element of $ExclusiveChoice_i$, $MultipleChoice_i$ and $ParallelSplit_i$ has value `XOR`, `OR` or `AND` respectively.

$$Merge_i = (id, model\_fragment, name, alternative\_name, has\_inActivity, has\_outActivity, has\_logicConnector)$$

Where the cardinality of $has\_outActivity$ is 1. The $has\_logicConnector$ element of $SimpleMerge_i$, $MultipleMerge_i$, and $Synchronizatione_i$ has value `XOR`, `OR` or `AND` respectively.

$$Sequence_i = (id, model\_fragment, name, alternative\_name, has\_inActivity, has\_outActivity)$$

Where the cardinalities of both $has\_inActivity$ and $has\_outActivity$ are 1.

Definition 6. *Input and output are defined as parameters of an activity, which include value and data type. They are usually related to artifacts participating in the activity.*

$$I_i = (id, model\_fragment, name, alternative\_name, data\_type, related\_artifact)$$

$$O_i = (id, model\_fragment, name, alternative\_name, data\_type, related\_artifact)$$

If the same artifact related with both input and output parameters of an activity, the state of the artifact must change through this activity. We call it transformation.

Definition 7. *Precondition and postcondition are presented by expressions to constrain input and output. The constraints are usually used as contract in services or process composition.*

$$\Theta^{pre} = (id, model\_fragment, name, alternative\_name, related\_input)$$

$$\Theta^{post} = (id, model\_fragment, name, alternative\_name, related\_output)$$

Definition 8. *Exception happens in an activity and it can be handled by an activity.*

$$E_i = (id, model\_fragment, name, alternative\_name, handler\_Activity, same\_as, different\_from, kind\_of, superConcept\_of, partialEffect\_of, compositionConcept\_of, instance\_of)$$

Exception will be annotated using predefined exception types in a domain ontology. The activity handling the exception is pointed out by $handler\_activity$.

PSAM is modeled in OWL when it is implemented for annotation applications. Appendix G.1 presents the OWL representation of a complete PSAM [1].

4.7 A Simple Example of Process Semantic Annotation

There is a business process model to describe a very simple process of buying merchandise. It can be reused in any specialized and complicated purchase process. We assume it is originally built in EEML [77]. This purchase process contains only a task "purchase". There are two person roles in this task named "Client" and "Seller". The process starts with a milestone "agreed deal" and ends with a milestone "deal finishes". One flow links from "agreed deal" to the input port of the task "purchase" and another flow links from the output port of "purchase" to "deal finishes". A resource role "Order" coming to the input port is and another resource role "Receipt" is out of the output port. The EEML process model of a purchase process is illustrated in Figure 4.6.

We applied the semantic annotation approach to annotate the purchase model. The EEML modeling constructs are annotated with GPO concepts in meta-model annotation. In this case, EEML modeling constructs are mapped to the GPO concepts

[1] A complete PSAM contains the part of PSAM presented in this chapter and the extension part of PSAM in Chapter 5.

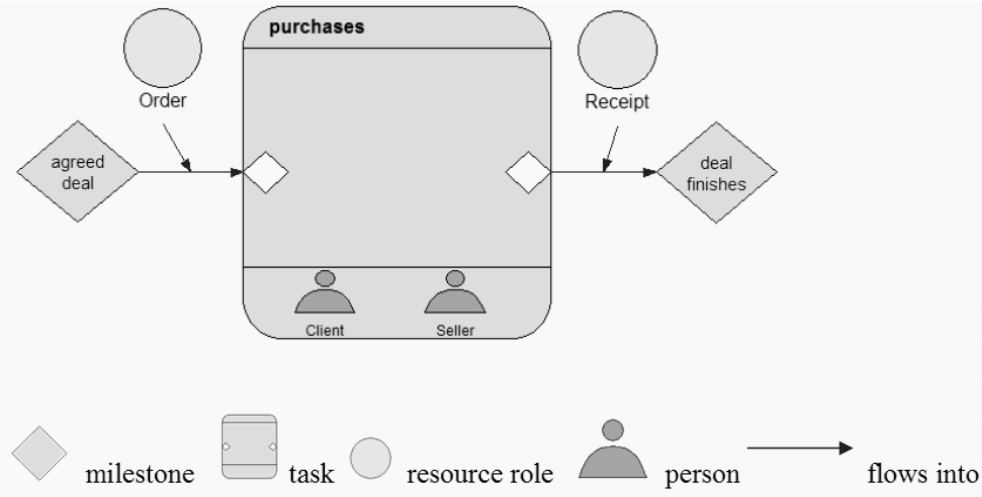


Figure 4.6: EEML process model example: purchase process

following the mapping rules for meta-model annotation. For example, the EEML `Task` is one-to-one mapped to the GPO *Activity*. Based on the meta model annotation, the GPO concepts will take the place of the corresponding process modeling constructs to describe the process. A domain ontology is employed to annotate the model contents which are described in the process annotation model. For instance, the EEML task "purchases" is a GPO *activity*, and this *activity* is annotated as a *kind of* domain ontology concept "buy" in the PSAM model. Figure 4.7 illustrates the annotation results of the purchase model.

We exemplify parts of the PSAM instance of annotation results which are represented in OWL. The example here is only a demo of the OWL representation. In the demo the data type of model_fragment is defaulted as URI and the data types of other properties are not specified, which can be compared with the PSAM instances in OWL from exemplars in Appendix H[2].

```
<GPO:Activity rdf:ID="av1">
  <GPO:model_fragment rdf:resource="&eeml_purchase;#oid6"/>
  <GPO:name>purchases</GPO:name>
  <GPO:alternative_name>Purchase</GPO:alternative_name>
  <GPO:has_Actor-role>
    <GPO:Actor-role rdf:resource="#ar1">
  <GPO:has_Actor-role>
  <GPO:has_Actor-role>
    <GPO:Actor-role rdf:resource="#ar2">
  <GPO:has_Actor-role>
  <GPO:has_Input>
    <GPO:Input rdf:resource="#in1">
  <GPO:has_Input>
```

[2]The exemplars are introduced in Chapter 7.

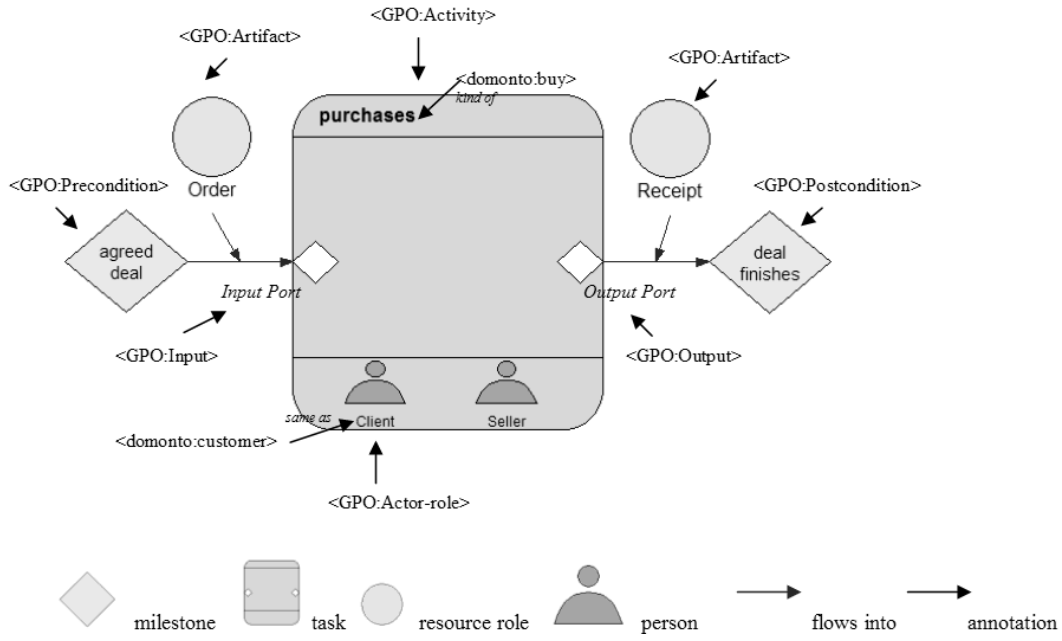


Figure 4.7: Annotated EEML process model example: purchase process

```
  <GPO:has_output>
    <GPO:Output rdf:resource="#out1">
  <GPO:has_Output>
  <GPO:kind_of rdf:resource="&domonto;#buy"/>
</GPO:Activity>
<GPO:Actor-role rdf:ID="ar1">
  <GPO:model_fragment rdf:resource="&eeml_purchase;#oid181"/>
  <GPO:name>Client</GPO:name>
  <GPO:alternative_name>Buyer</GPO:alternative_name>
  <GPO:alternative_name>Purchaser</GPO:alternative_name>
  <GPO:alternative_name>Customer</GPO:alternative_name>
  <GPO:same_as rdf:resource="&domonto;#customer"/>
</GPO:Actor-role>
<GPO:Actor-role rdf:ID="ar2">
  <GPO:model_fragment rdf:resource="&eeml_purchase;#oid182"/>
  <GPO:name>Seller</GPO:name>
  <GPO:alternative_name>Salesperson</GPO:alternative_name>
  <GPO:alternative_name>Salesman</GPO:alternative_name>
  <GPO:alternative_name>Vender</GPO:alternative_name>
  <GPO:same_as rdf:resource="&domonto;#vendor"/>
</GPO:Actor-role>
<GPO:Input rdf:ID="in1">
  <GPO:model_fragment rdf:resource="&eeml_purchase;#oid11"/>
  <GPO:name>Order</GPO:name>
  <GPO:alternative_name></GPO:alternative_name>
```

```
  <GPO:datatype rdf:resource="&xsd;#anyURI"
  <GPO:same_as rdf:resource="&domonto;#purchase_order"/>
</GPO:Input>
<GPO:Artifact rdf:ID="af1">
  <GPO:model_fragment rdf:resource="&eeml_purchase;#oid122"/>
  <GPO:name>Order</GPO:name>
  <GPO:alternative_name></GPO:alternative_name>
  <GPO:alternative_name>Salesman</GPO:alternative_name>
  <GPO:alternative_name>Vender</GPO:alternative_name>
  <GPO:same_as rdf:resource="&domonto;#vendor"/>
</GPO:Artifact>
```

4.8 Summary

In this chapter, we have proposed a semantic annotation framework to manage semantic heterogeneity of process models from the following perspectives: basic description of process models (profile annotation), process modeling languages (meta-model annotation), and process models (model annotation). Two ontologies are used for annotation purposes: General Process Ontology for meta-model annotation, and a domain ontology for model annotation. Furthermore, we have defined a set of mapping strategies for guiding users to annotate models.

As a formal result of the proposed framework, the process semantic annotation model (PSAM) provides a common semantic annotation schema for annotating semi-structured IS solutions. A PSAM model describes the process properties of a process model fragment in a way of representing process knowledge, and contents in the PSAM model are mapped to context references for domain semantic reconciliation. Process models are therefore transformed into process knowledge of IS solutions which are represented in the PSAM models after annotation. The PSAM models are supposed to be able to help the human and machine to understand heterogeneous process models with reconciled process semantics.

An extension of the semantic annotation framework will be elucidated in next chapter. The framework is extended by adding pragmatic metadata, namely providing a method to specify intentions of systems' owners, which are achieved through process models.

Chapter 5

Extension Semantic Annotation — Goal Annotation

In this chapter, we elucidate the annotation of process models and process model fragments with a goal ontology to specify organizational objectives (i.e. intentions of systems' owners) achieved by processes. The aim of goal annotation is to pragmatically facilitate recognizing process knowledge conveyed by heterogeneous process models based on the enriched intentional semantics of processes [95].

A goal ontology is a set of conceptualized goals and relationships among them. Based on the investigation of several goal modeling methods applied in requirements engineering and process modeling, we propose goal ontology design principles and semantic representations.

Goal annotation is a procedure to organize and define the process knowledge with goal ontology, i.e. building relationships between process models and pre-defined goal concepts. Defining those relationships is the major task of goal annotation, which indicates what relationships are supported in annotation and how annotation can be implemented. Consequently, the goal annotated models can be queried and reused in a goal-driven method with the goal ontology.

5.1 Goal-Driven Process Knowledge Discovery

As knowledge, business process models are required and reused for achieving business objectives or goals. On the other hand, a process model describes the capability and utility of a process, i.e. how the process achieves certain results. Therefore, goals can be used to describe what a knowledge user desires when searching and applying process models, which is known as goal-driven discovery approach. In such a way, the knowledge user does not need to specify models' identification such as name or location of the process models, but only specify his/her business objectives or goals to query the models. The search engine finds the process models through mapping the goal request and capability of process models. This approach provides a pragmatic way for users to discover knowledge because goals are obvious for users to specify their request. The discovery process is a black box for users who have no idea of the availability (existence and location) of desired process models. Goal-driven discovery is also transparent for

process models since there is no need for representations of process models but only for process capability specifications.

To facilitate the goal-driven discovery, capability of processes should be explicated in process models. However, most process modeling languages do not supply such a mechanism. Even if they are goal-supported modeling, they typically differ in various languages. Moreover, the representations of capability might also differ by terminology and conceptualization, which are common problems in any heterogeneous models. By extending our semantic annotation framework, a goal ontology is adopted to annotate process models for specifications of the process capability. With a goal ontology, the user can specify his/her goal request. The goal-driven discovery process is to match goal requests with goal annotation of process models through the goal ontology definitions.

In goal annotation, a goal ontology provides definitions of a set of business objectives or goals. In requirements engineering, business objectives or goals can be described in natural language or specified in goal models. Goal models are usually made for requirements elicitation and then used to derive process models. Goal modeling methods in requirements engineering provide references for building a goal ontology. When implementing a goal annotation, we need to establish the relationships between process models and the goal ontology. Goal annotation becomes one element of the semantic annotation framework which complements the approach from a pragmatic perspective.

5.2 Goal Ontology for Semantic Annotation

In this work, business goals are formalized in a goal ontology, which provides semantic representations of goals in a consensual way for different organizations. The focus in this section is goal ontology modeling. Firstly, we start from the discussion about the principles of goal ontology modeling in our approach. Based on the principles, we then define our goal ontology model in OWL.

5.2.1 Goal ontology design principles

Since we focus on process models, *the first design principle* is that the goals to be linked to process models should be process achievable. Thus, the research will not include the goals related to technical factors such as the usage of computing resources or financial aims like reaching a certain amount of gross profit. Furthermore, the process knowledge in our research is at model level not at instance level. Accordingly, goals defined for a certain domain are relatively abstract but not very concrete for an application. We are concerned with for example, the accomplishment of customers ordering of a cell phone but do not interested in the instance goal like satisfying Joan's ordering of SAMSUNG E568. In order to organize goals at a rational abstraction level, a goal ontology is domain specific and organized into categories.

As *the second design principle*, the corresponding relationships between a goal ontology and a process model should be easily built to facilitate the goal annotation. Many goal modeling languages have modeling constructs which are relevant to process modeling, so that those goal modeling approaches from the literature could be referred to model goal ontology concerning this principle. Distinct from those goal modeling languages, a goal ontology is used to normalize the semantic representation of agents'

intentions not to analyze goals for requirements elicitation, specification or validation. The goal analysis mechanism can find out from why to how, but a goal ontology just focuses on why. Hereby, we should tailor those goal modeling approaches for our purpose rather than just reuse them.

However, in the existing goal modeling methods, goals are represented informally or semi-formally, and they are not machine-understandable. A goal ontology provides references of goal concepts for annotation users to annotate process models, and the annotation results are supposed to be machine-understandable in order to enable machine to manipulate process knowledge discovery and reuse. Therefore, semantics of goal ontology should be both human- and machine-interpretable, which is *the third design principle*. We use OWL to represent a goal ontology and they are well modeled in classes, properties and relationships. The details of modeling the goal ontology in OWL are further elaborated in next section.

5.2.2 Semantic representations of a goal ontology

Following those principles, we make a meta-model of the goal ontology. In this meta-model, the goal ontology is defined based on goal categories and goal targets.

A goal is a condition or state of affairs in the world that stakeholders would like to achieve [40]. Soffer et al. [167] defines a goal as a set of stable states. In [205], state of a thing is described as the vector of values for all property functions of a thing. In the context of process modeling, states must be represented as values for properties of process and properties of objects involved in process. That is, a goal can be expressed as states of activities (e.g. "delivery is processed") or states of artifacts (e.g. "receipt is received"). The goal target could be *Activity* or *Artifact*. Usually the 'accomplished' is regarded as the goal state of an activity, whilst the state of an artifact has to be specified for different goals. Goal is an organizational concept and goals are held by stakeholders. 'Actor' is defined to represent the goal owner in i*/GRL and 'Agent' is applied in KAOS when analyzing the potential goal realizer. *Actor-role* is therefore the goal target in the goal ontology as well. In KAOS, goals are non-operational objectives and constraints are operational objectives. Although constraints are not goals, goals can be operated by constraints [21]. In this sense, *Constraint* is also the goal target. Some goals may be represented as constraints statements, e.g. the quality constraints (wrt. time, cost, security etc.) to processes according to domains. The relationships between *Goal* and those targets are simply defined because the purpose of the goal ontology is not to analyze the goal like those existing goal modeling methods. The targets show the different perspectives of viewing a goal. These targets are represented in the same way as the concepts in **PSAM**, which provides potential links between a goal ontology and process models.

In general, goals can be classified into hard goals and soft goals [110]. Hard goals are related to functional requirements and they are obviously supported by process. Soft goals are about global qualities of a system. Most of soft goals are related to non-functional requirements (colloquially "-ilities"). Soft goals do not have clear-cut criterion for their satisfaction, and they are satisfied when there is sufficient positive and little negative evidence for this claim [110].

Relations of goals should be specified in a goal ontology. A goal could be represented

in a relatively general sense (e.g. *security goal*) or in a specific way (e.g. *safe payment*), so the subsumption relationship should be supported. Relations between two goals having the same or different effects can be regarded as semantic equivalence or semantic difference. Decomposition relation – an important characteristic found in most goal analysis should be specified in the goal ontology too.

Based on the above analysis, semantics of goals in a goal ontology are specified through categories, expression perspectives and relationships. OWL `Classes` and `Properties` provide a way to define the semantics of goal concepts. Standing for goal categories, ***Hard Goal*** and ***Soft Goal*** are two upper level classes for all goals in general. Hard goals are functional and usually they are domain-dependent, so they are usually specific from different domains. Soft goals can be described generally in a set of "-ilities" (which are regarded as soft goal category), and also specific according to domains. Attributes and relations of a goal class are represented through `owl:DatatypeProperty` and `owl:ObjectProperty` respectively. Based on the analysis of expression perspectives, we know that a goal can be represented through specifying the targets such as *Activity*, *Artifact*, constraint and *Actor-role*. If those targets are already defined in domain ontology as classes, a goal can establish relationships with them. Otherwise, the goal uses attributes to represent them. To make the terminology consistent (with GPO), we name the relations/attributes as ***targetActivity***, ***targetArtifact***, ***targetRole*** and ***targetConstraint***. State is modeled as an attribute of ***Artifact***, which is associated with the ***targetArtifact*** of a goal.

The subsumption relationship (`owl:subClass`) is used to represent goal hierarchy. A simple part-whole relationship represent goal composition. OWL does not provide any built-in primitives for part-whole relations (as it does for the subclass relation). However OWL contains sufficient expressive power to capture most, but not all, of the common cases [200]. We therefore apply a simple part-whole relationship to represent the decomposition of goal concepts. The 'part' goals contribute the impacts to the 'whole' goal. The logic connections (OR, AND, XOR) between the parts are not considered in the goal ontology due to two reasons as follows. First, it is the representational capability of OWL. Second, such concrete goal analysis mechanisms are not necessary for a goal ontology. A goal ontology should be general to applications and decomposition of a goal depends on a specific application. A goal ontology is more like a taxonomy of goal concepts serving for semantically-aligned goal representation. Hence, the terminology presenting goal concepts in a goal ontology should be normalized. In addition, some general attributes such as ID, name, description are attached to a goal class to identify a goal concept. Semantic representation of a goal ontology is specified in the meta-model shown in Figure 5.1.

5.3 Relations between Process Models and a Goal Ontology

We can see that based on the design principles of a goal ontology modeling, underlying relations between process models and a goal ontology becomes clear. These relations can be used to derive goal annotation relationships of process models.

We start by clarifying some concepts involved in the relations. We assume that a

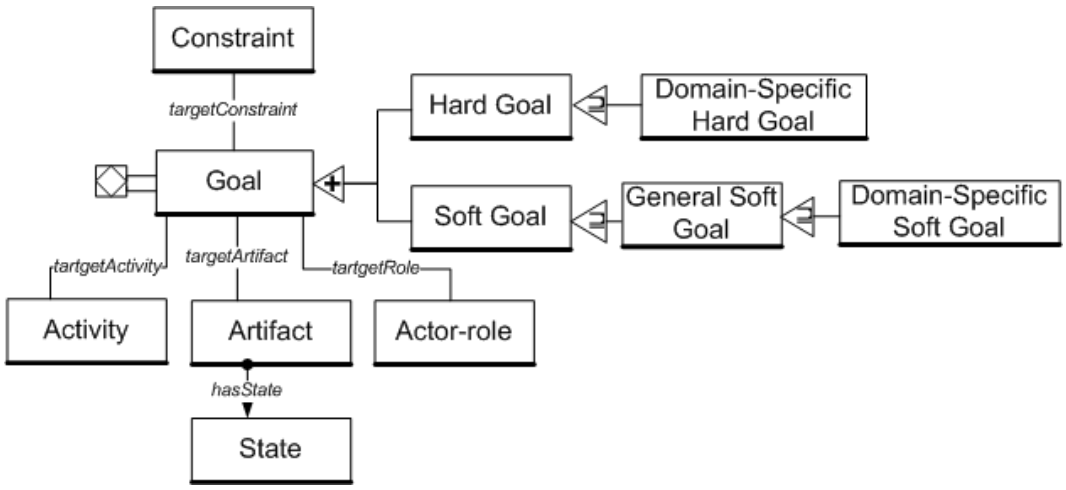


Figure 5.1: Meta-model of the proposed goal ontology

process model depicts a process, and a process consists of numbers of activities. As we have defined in GPO, an activity may be an atomic activity or a composite activity. In our semantic annotation framework, we define that a process model comprises a set of activities (***AV***) and an activity can be decomposed into sub-activities. If an activity in a process model is not an atomic activity[1] which is composed of a set of related activities, it is regarded as a process model fragment in this context. A goal (***g***) can be linked to a whole process model or to a process model fragment. We assume that process models are already organized into a decomposable hierarchy of activities referencing a domain ontology in the model annotation phase. Each level of activities in the hierarchy can be considered as goal annotation targets.

Definition 9. *In the semantic annotation framework, a process model (PM) can be partitioned into several process model fragments (PMF). Each PMF comprises a set of hierarchically organized and decomposable AV.*

$PM = PMF \bigotimes PMF$ *and* $PMF = AV \bigotimes AV$

Definition 10. *Any goal concept (g) in a goal ontology (G) is possibly related to an activity (av) in a PM/PMF:*

$$\forall(g, av) goalRelated(g, av) \qquad (a)$$

- *if the property* ***targetActivity*** *(av′) of a g is same or synonymous with av:*

$$\exists(av') targetActivity(g, av') \bigwedge av' = av \qquad (b)$$

- *if the property* ***targetArtifact*** *(af′) of a g is related to the output of av and the* ***State*** *(s') of af′ is the value of the* ***Output*** *(o) of the* ***Artifact*** *(af):*

$$\exists(af', af, s', o) targetArtifact(g, af') \bigwedge hasState(af', s') \bigwedge$$
$$s' = o \supset hasOutput(av, o) \bigwedge af' = af \supset relatedTo(o, af) \qquad (c)$$

[1]Note: An atomic activity can not be decomposed, but it is not an event either.

- *if the property* ***targetRole*** *(ar′) of a g is related to an* ***Actor-role*** *(ar) involved in av:*

$$\exists(ar')targeRole(g, ar') \bigwedge ar' = ar \supset hasActor - role(av, ar) \qquad (d)$$

- *if the* ***targetConstraint*** *(c′) expressed in a g is involved in (involvedIn)* ***pre-Condition*** *(pre),* ***postCondition*** *(post) or* ***Exception*** *(e) of av:*

$$\exists(c', pre, post, e)targetConstraint(g, c') \bigwedge (involvedIn(c', pre) \supset hasPrecondition(av, pre) \bigvee involvedIn(c', post) \supset hasPostcondition(av, post) \bigvee involvedIn(c', e) \supset hasException(av, e)) \qquad (e)$$

Therefore,

$$(a) \equiv (b) \bigvee (c) \bigvee (d) \bigvee (e)$$

Those definitions denote how an activity might be associated with a goal by specifying the facts of an activity that might affect a goal. Checking above cases through matching algorithms can automatically provide a list of possible goal annotations. The decisions of the desired goal annotations are left to the annotators. We hereby call such a mechanism semi-automatic goal annotation.

Definition 11. *A goal ontology (G) comprises a set of hard goals* G^h *and a set of soft goals* G^s*.*

$G = \{G^h, G^s\}$

The relations are further specified based on the context of the process models and the goal ontology. We define the relations as follows:

Definition 12. *Hard goals can be achieved by an activity or activities. I.e. the relation between an activity (av) and a hard goal (*g^h*) is* $\boldsymbol{achieves}(av, g^h)$*.*

Definition 13. *Soft goals can be positively or negatively satisfied by an activity or activities. I.e. the relation between an activity (av) and a soft goal (*g^s*) is* $\boldsymbol{positively\_satisfies}(av, g^s)$ *or* $\boldsymbol{negatively\_satisfies}(av, g^s)$*.*

Since the activities in the process models are decomposable, the relation between goals and a composite activity could be inferred based on the relations between goals and component activities.

Definition 14. *If an activity (av) is a component of another activity (*av^Δ*) in a process model/model fragment, av is the subactivity of* av^Δ*, i.e.* $\boldsymbol{subActivityOf}(av, av^\Delta)$*.* av^Δ *is a* ***Composite Activity*** *in that model.*

Usually the effects of hard goals achieved by a subActivity can contribute to its composite activity. That is,

Definition 15. *If av is the subactivity of* av^Δ *and av achieves* g^h*,* av^Δ *achieves* g^h*:*

$$(\forall(av, g^h)\exists av^\Delta)subActivityOf(av, av^\Delta) \bigwedge achieves(av, g^h) \longrightarrow achieves(av^\Delta, g^h)$$

However, the effects of soft goals can not be simply passed in the same way as hard goals. To a composite activity, the contribution of a soft goal from a subactivity might be enhanced or reduced by other subactivities which positively or negatively satisfy the same soft goal. The contribution could be calculated if the effects of soft goals are quantified. This issue is only simply considered in our current work by ***simple contribution calculation rules***. All effects of soft goals are regarded as the same. The contribution of a soft goal to a composite activity is determined by comparing the numbers of subactivities which positively satisfy and negatively satisfy the same soft goal. That is,

Definition 16. *Let a g^s is positively satisfied by N subactivities, and is negatively satisfied by M subactivities in a composite activity av'. Consequently,*

- ***positively_satisfies*** (av', g^s)*, if* $N > M$
- *the effect of g^s is counteracted for av', if* $N = M$
- ***negatively_satisfies*** (av', g^s)*, if* $N < M$

The relations between goals and activities defined in this section will be applied to build annotation links between the goal ontology and process models in the goal annotation. That is to say, the metadata schema of the goal annotation for process models is:

```
ActivityID
   achieves|positively satisfies|negatively satisfies
GoalID
```

5.4 PSAM with Goal Annotation

With goal annotation in the semantic annotation framework, the PSAM includes the references of the goal ontology.

$PSAM = (AV, AR, AF, WP, I, O, \Theta^{pre}, \Theta^{pos}E, PD, PG)$, where PG is a subset of goal ontology (G).

The annotated activity is also updated as follows and the complete PSAM can be represented in OWL (see Appendix G.1).

Definition 17. $AV_i = (id, model\_fragment, name, alternative\_name, has\_Actor-role, has\_Artifact, has\_Input, has\_Output, is\_in\_WorkflowPattern\_of, has\_Precondition, has\_Postcondition, has\_Exception, has\_subActivity, same\_as, different\_from, kind\_of, superConcept\_of, phase\_of, compositionConcept\_of, instance\_of, achieves|positively\_satisfies|negatively\_satisfies)$

5.5 Goal Annotation Procedure

In goal oriented requirements engineering, goal analysis and modeling is usually a top-down procedure — decomposing high level goals down to lower level goals and operational activities. Goal annotation is a bottom-up procedure — first annotating low level sub-activities and then annotating higher level activities and finally the whole process model with goal ontology. Focusing on the activity, we describe the goal annotation procedure accompanied with the meta-model annotation and model annotation in a UML Activity diagram in Figure 5.2. Through meta-model annotation, activities are identi-

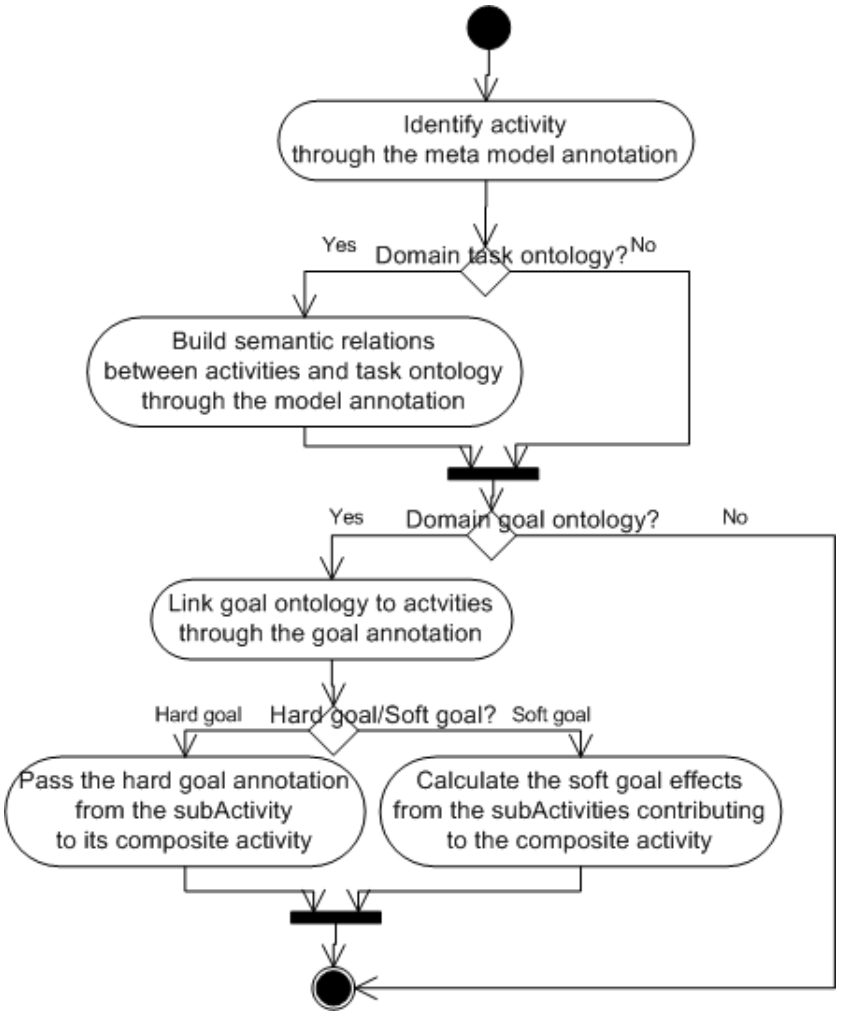


Figure 5.2: Goal annotation procedure

fied by the markup ***Activity*** in **PSAM**. In the model annotation phase, if a domain ontology is available as the activity references, the identified activities are annotated with concepts in domain ontology references via the semantic relations such as $same\_as$, $different\_from$, $kind\_of$, $superConcept\_of$, $phase\_of$, and $compositionConcept\_of$. If the domain goal ontology is available, the possible links between the activities and the goal ontology can be checked based on the relations described in section 5.3. We

employ the annotation from the component activities to the composite activities. The contributions of the goals annotated to the low level component activities can be passed to or calculated for their upper level composite activities.

In goal annotation, we modify goal annotation relations through *achieves_Goal*/*positively_satisfies_Goal*/*negatively_satisfies_Goal* in previously defined **PSAM** in Chapter 4. An example of goal annotation of an activity that achieves a hard goal is as follows:

```
<psam:Activity rdf:ID="ID">
  <psam:model_fragment rdf:resource="&MODEL_NAMESPACE#MODEL_ID">
  ...
  <psam:achieves rdf:resource="&GOAL_ONTOLOGY#GOAL_ONTOLOGY_CONCEPT"/>
```

5.6 Summary

As an extension of the semantic annotation framework, a goal annotation approach has been elaborated in this chapter. The goal annotation has been designed for a goal-driven management of heterogeneous process models on the conceptual level. Since business process models as IS solutions were created for achieving the intentions of systems' owners, a new problem to achieve similar intentions might reuse those solutions. Goal annotation helps humans and machines to locate the process knowledge by specifying the goals of process models.

The approach relies on a goal ontology, which provides common semantic representations of goal concepts. A goal ontology reconciles intentional semantics represented in heterogeneous models. The purpose of the goal annotation is twofold: 1) to enrich the semantics of the objectives of processes; 2) to provide a pragmatic way to find process knowledge based on business goals. Based on a set of goal ontology design principles, we proposed a way to describe the semantics of goals for process model annotation purpose. We have also discussed relations and rules between process models and goal ontologies. The formalization of the relations can be implemented to facilitate an automatic or semi-automatic goal annotation procedure in practice. In our semantic annotation framework for process models, process models are first described by PSAM (process semantic annotation model) after meta-model annotation, and then referenced with domain ontology in model annotation. Goal annotation is therefore employed based on the domain annotated PSAM models.

Chapter 6

Pro-SEAT (Process SEmantic Annotation Tool)

In order to facilitate semantic annotation of process models, we have developed a prototype of the semantic annotation tool — Pro-SEAT (**Pro**cess **SE**mantic **A**nnotation **T**ool). The tool implements the proposed semantic annotation framework. Development of the tool included planning system modules, defining data structure, designing annotation algorithms, and implementing functions and user interfaces. The tool supports all the phases of the semantic annotation procedure. In addition, a query-based model navigator is included in the tool.

6.1 Components of Prototype Environment

Pro-SEAT is an isolated application, independent of modeling tools. As an annotation tool, it is used to attach additional information in models. However, we need to import existing process models into the annotation tool. This requires the annotation tool to read models created by certain modeling tools. In this work, we take Metis[1], a modeling product from Computas partner Troux Technologies, as our modeling environment. Metis supports different modeling languages, such as UML, EEML and BPMN by its powerful Metamodel Developer. Models created by Metis are formated in XML which is analyzable by XML parsers. The annotation tool therefore contains functions to parse and read Metis models.

OWL technology is chosen for modeling ontology in the proposed semantic annotation approach. Therefore, the annotation tool should provide an OWL ontology browser and an ontology selection to support the ontology-based annotation. In the prototype, we integrate the Protégé-OWL API which can parse and manipulate OWL ontologies. An OWL ontology could be edited by any ontology editors, including the Protégé-OWL.

The current version of the semantic annotation tool — Pro-SEAT can read original process models created in Metis and OWL ontologies. The main task of the tool is to apply the annotation framework to build relationships between models and ontologies.

[1] http://www.computas.com/templates/Page____371.aspx

As an output of the system, the annotation result is stored in an OWL instance file, separately from the original process model.

6.1.1 Process modeling environment — Metis

Metis is an enterprise modeling environment. Metis provides a Metis Architect including:

- Model Browser. Provides end-users with ability to view (read-only) models published on the Internet or local area network.
- Model Annotator. Model reviewers can offer comments and feedback in "sticky note" style. Annotated models provide an easily accessed audit of proposed model changes and decisions.
- Model Editor. Creates visual models from enterprise data and can accommodate change on a detailed, operational level to assure information relevancy. It also allows users to publish models to a web server.
- Model Designer. Targets the more advanced modelers responsible for visual display and dynamic behavior of models, objects, and relationships.
- Model Developer. Offers a powerful development tool for advanced developers who need to create, adapt, or extend objects, relationships, and search criteria within a meta-model template.
- Data Import Facility. Allow users to visually model how external data should be imported into Metis.

Metis is a family of integrated products for visual model development and publishing plus a repository. Metis is a XML modeling tool and all models and meta-models are stored in an XML repository. The entire tool is configured by XML Schemas, and can be used to create or extend XML Schemas. The point-and-click publishing capability produces Web-browsable XML graphical models and views [51].

The XML files of models which are created in Metis are the inputs to Pro-SEAT. Therefore, Pro-SEAT should provide support to parse and read Metis generated models.

6.1.2 Ontology modeling environment — Protégé-OWL editor

The Protégé-OWL editor is an extension of Protégé [147] that supports the Web Ontology Language(OWL). The Protégé-OWL editor enables users to load and save OWL and RDF ontologies; edit and visualize classes, properties and SWRL rules [202]; define logical class characteristics as OWL expressions; execute reasoners such as description logic classifiers; edit OWL individuals for Semantic Web markup [148]. We make use of the Protégé-OWL editor as the ontology modeling environment to model the GPO, the domain ontology and the goal ontology. Those onologies are then saved in OWL files which will be loaded in the annotation tool. The annotation results — PSAM models generated from the annotation tool are in OWL. The results can be loaded in Protégé-OWL editor so that we can use Protégé-OWL editor to edit, visualize, and reason on the PSAM models independent from the annotation tool.

6.1.3 System modules in the semantic annotation tool — Pro-SEAT

Pro-SEAT includes a module to read and display the Metis process models as the source of process knowledge. Because meta-models and models generated by Metis are both in XML format, an XML parser is needed when reading models. There are several XML parsers available. Since it is a prototype, we did not examine the performance of different parsers at this stage. We chose XML DOM Parser from JAXP [179] to parse meta-models and models into DOM tree nodes, so that meta-models and models are displayed in a tree view. Besides, profile and meta-model annotation results are exported in XML format and they will be parsed when being loaded.

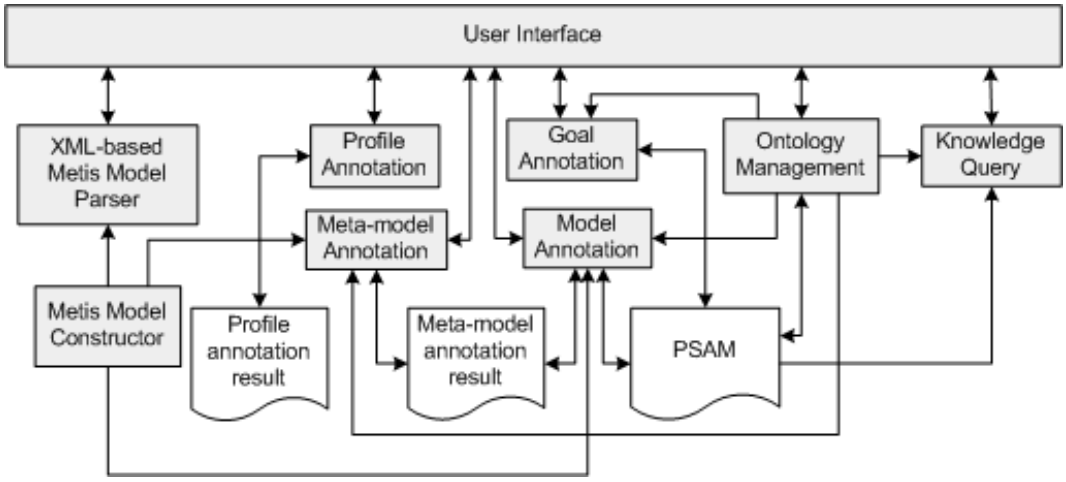


Figure 6.1: System modules of the prototype

In order to keep the Metis model structures and also facilitate the manipulation of models, a module of constructing parsed data into Metis model structures is developed in the system. In the module, a set of Java Classes and Interfaces are defined which follow the data structures of Metis.

An ontology management module is required for loading, parsing and manipulating OWL ontologies using the Protégé-OWL API [146]. The Protégé-OWL API is an open-source Java library for the Web Ontology Language and RDF(S). The API provides classes and methods to load and save OWL files, to query and manipulate OWL data models, and to perform reasoning [146]. By applying the Protégé-OWL API, we can have the Protégé ontology browser and the ontology selection panel integrated in Pro-SEAT. The PSAM model generated from a meta-model annotation file is an OWL file, and the annotation results are stored as instances of this OWL file. We can use the same module of handling the OWL ontology to open and save the annotation results.

The central modules of the system are those realizing the annotation phases defined in the semantic annotation framework: profile annotation module, meta-model annotation module, model annotation module, and goal annotation module. The main functions of those modules are connecting models and ontologies through semantic relationships which are defined as annotation properties in PSAM. The annotation results are saved respectively in a profile annotation result file, a meta-model annotation result file and a PSAM file.

A process knowledge query application module interacts with the OWL module by loading the ontology and the annotation results for an ontology-based query.

The user interface module manages interactions between a user and the system, so that it connects most system modules. Figure 6.1 specifies the interaction relations between those system modules.

6.2 Data Structure

In this section, we present the data structures using RML from the logical view to specify the interrelations between the entities participating in the system. We have adapted the MDD (Model Driven Development) methodology in the design and the implementation of the prototype. The structured entities in the design phase correspond to the ***Java Classes*** and ***Interfaces*** in the implementation.

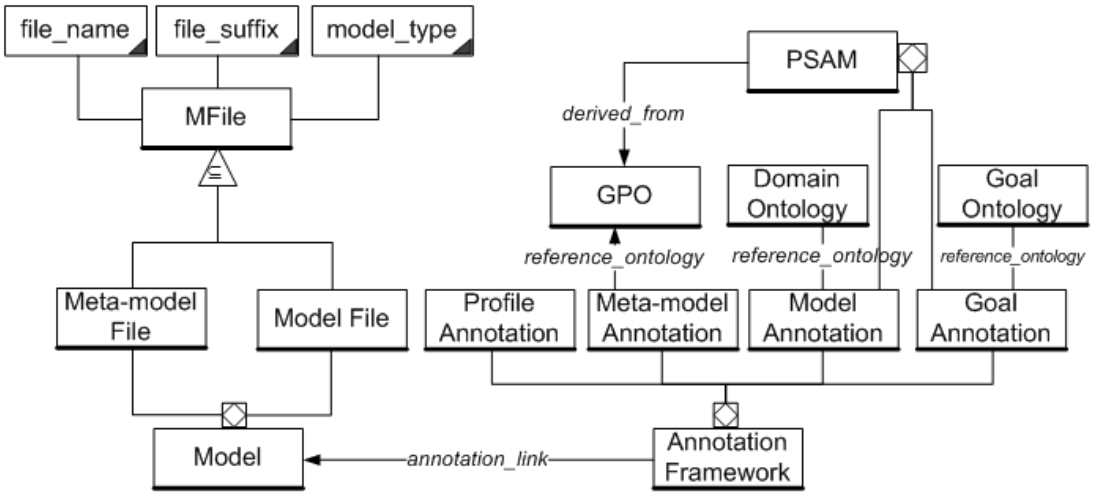


Figure 6.2: Structure of entities in the Pro-SEAT prototype

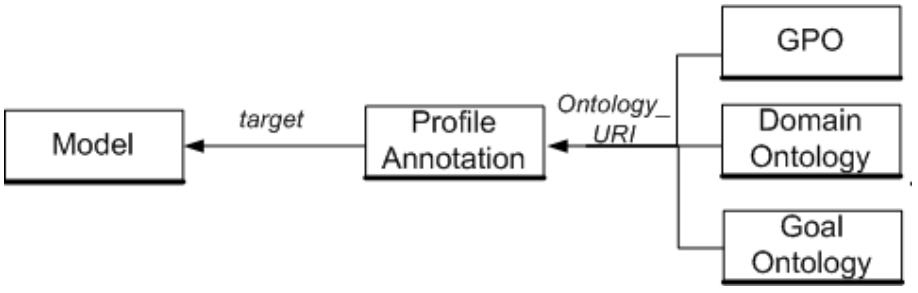


Figure 6.3: Structure of entities in the profile annotation

First, we provide an overview of data structures of the entities in Figure 6.2. Generally, an annotation structure is the annotation link between a model and the annotation framework. A model usually consists of meta-model file(s) and model file(s). Both

types of model files share same features such as "file name", "file suffix" and "file type". Profile annotation, meta-model annotation, model annotation and goal annotation are four components of the annotation framework. Ontologies (GPO, domain ontology and goal ontology) are the reference ontologies in the different annotation components. A PSAM file is derived from GPO and its content comprises the model annotation and the goal annotation. The schema of PSAM models we defined in section is in fact the extended GPO model in OWL, which is included in section G.1 of Appendix G.

Next we describe the data structure of each annotation component. In Figure 6.3, any model is the target of a profile annotation and URIs of reference ontologies are to be specified in the profile annotation.

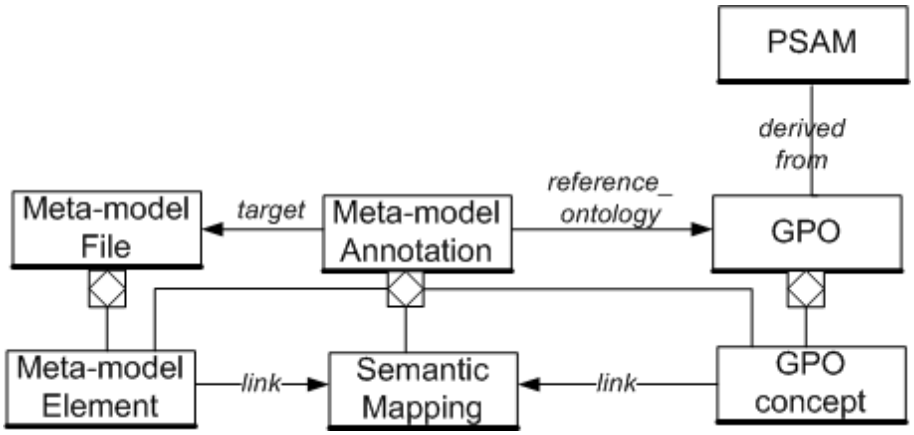


Figure 6.4: Structure of entities in the meta-model annotation

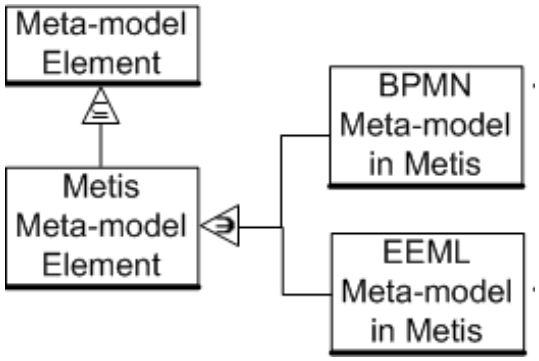


Figure 6.5: Metis meta-model structure

For a meta-model annotation (Figure 6.4), the meta-model file is the annotation target and the reference ontology is GPO. A meta-model file is composed of meta-model elements. GPO contains a number of the GPO concepts. The meta-model annotation results are composed of semantic mappings between meta-model elements and the GPO concepts. In the current version of the prototype, only Metis models are imported. In Figure 6.5, two specific modeling languages are exemplified in the data structure of the Metis meta-model elements.

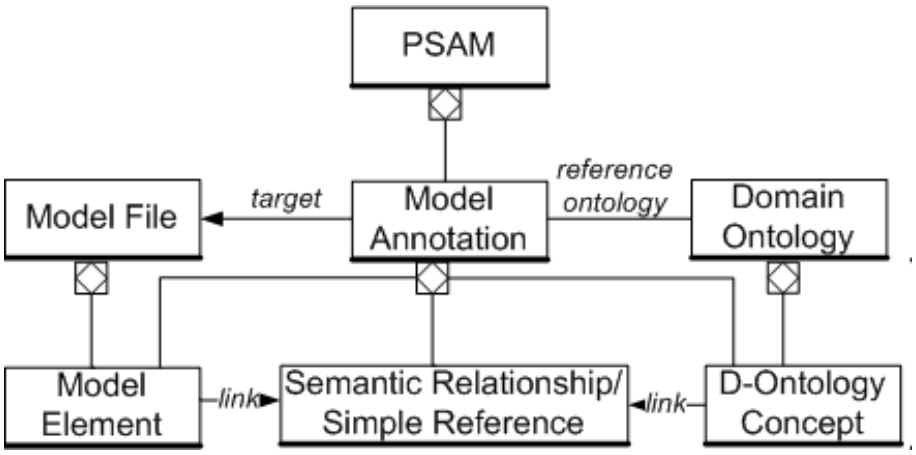


Figure 6.6: Structure of entities in the model annotation

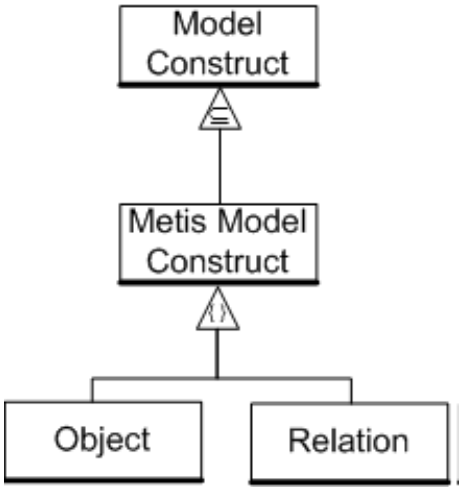


Figure 6.7: Metis model structure

Figure 6.6 displays the structure of entities in the model annotation, in which a model file is the annotation target. The model file is composed of model constructs. The ontological concepts defined in a domain ontology are referenced or semantically mapped to model elements in the model annotation. The Metis model element is one kind of model element and its members include `'Object'` and `'Relation'` — two abstract types in the meta-level definitions of the Metis model (Figure 6.7).

In Figure 6.8 we can see that the structure of entities in the goal annotation analogous with the model annotation. One distinct difference is that an intermediate granularity of the model — model fragment is between a model file and model elements. The goal annotation can link a goal ontology concept to a model fragment that comprises several model elements. In terms of the process model annotation, the process model fragments are composed of a set of activities (Figure 6.9).

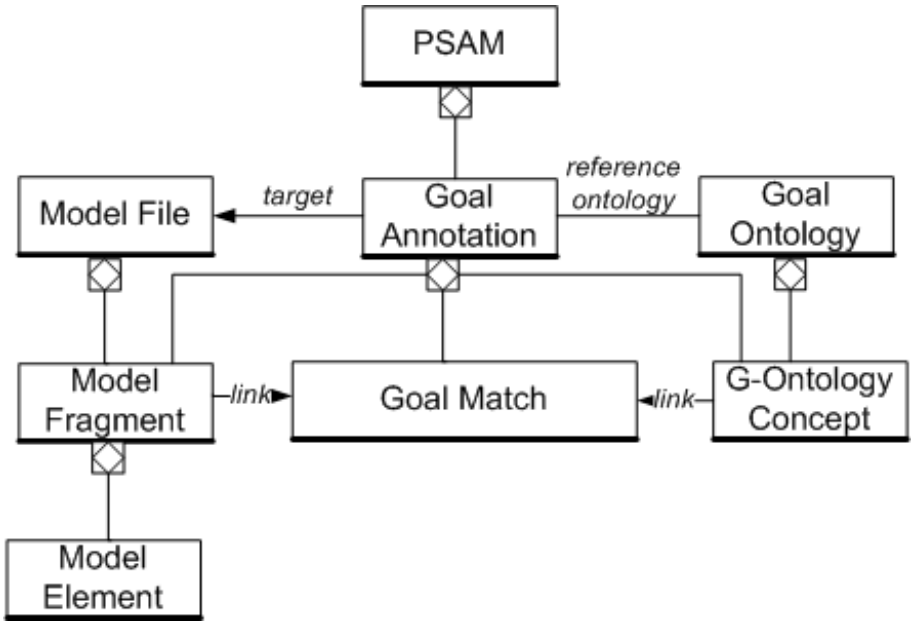


Figure 6.8: Structure of entities in the goal annotation

6.3 Goal Annotation Algorithm

In the current version of Pro-SEAT, there are no particular algorithms to implement the manual procedures of profile, meta-model and model annotations. However, based on the formal definitions of goal annotation (see Chapter 5), we design algorithms to semi-automate the goal annotation procedure. A semi-automatic goal annotation is implemented through the algorithms in the following cases.

- **A.** Match a target activity (av') of a goal (g) in the goal ontology (G) with the ***Activity*** (av) in a PSAM model.
- **B.** Match a target artifact (af') of a goal (g) in the goal ontology (G) with the ***Artifact*** (af) and ***Output*** (o) in a PSAM model.
- **C.** Match a target role (ar') of a goal (g) in the goal ontology (G) with the ***Act-role*** (ar) in a PSAM model.
- **D.** Match a target constraint (c') of a goal (g) in the goal ontology (G) with the ***Precondition*** (pre), ***Postcondition*** ($post$), and ***Exception*** (e) in a PSAM model.

To automate the matching, we exploit semantic mappings through both ontology comparison and string match between ontology references and model elements. To rank mapping results, weights are assigned to the different ways of mapping (σ is for a weight of ontology comparison. τ is for a weight of string match). In ontology comparison, three different semantic relationships applied in model annotation are taken into account in the assignment of weights. The synonym ("same_as") relationship is given the highest weight for a complete match ($\sigma = 1$), and the hypernym ("kind_of")

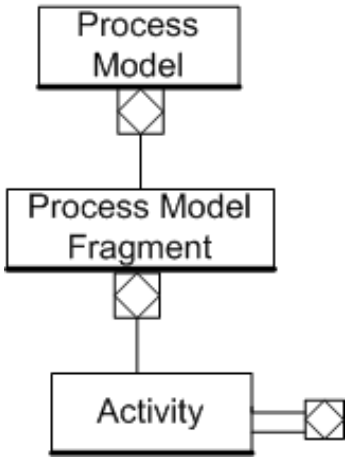


Figure 6.9: Structure of process model fragment

is given a lower weight for a subsumption relationship ($\sigma = 0.8$), and the meronym ("phase_of" for activity, "part_of" for artifact, "member_of" for actor-role) is given the lowest weight for a subset relationship ($\sigma = 0.5$). In the string match, an exact match gets higher weight than a sub-string match ($\tau = 1$ for exact match, and $\tau = 0.5$ for sub-string match).

Weights are also different for each case. We set different weight parameters for four cases matching. Values of parameters can be set by users. For example, α is used as the weight for calculation results of ontology comparison in case **A**, and ε is for calculation results of ontology comparison in case **C**. Details of the algorithm are described in Appendix D.

When matching a goal concept (g) with an activity (av) in models, all the four cases must be checked on g and av. The result of a match is a total weight by summing the calculation results from four cases. The higher total weight is, the better g matches av.

6.4 Functionality and User Interface

The main functionalities of Pro-SEAT in this prototype version are as follows: profile annotation, meta-model annotation, model annotation, goal annotation and query-based model navigation. UI (User Interface) for each annotation phase has a similar layout, which is built as a template and instantiated for different annotation phases. Figure 6.10 shows main user interface components in the Pro-SEAT tool.

A meta-model tree and a model tree are listed on the left side of the main frame. The meta-model and the model trees have the same hierarchical structure as the tree structure viewed from the Metis tool. The panel on the right side is for manipulating annotations and displaying results. If it is an ontology-based annotation, such as a meta-model, a model or a goal annotation, a reference ontology browser will be opened and embedded on the right annotation panel. A console is located at the bottom of the frame. Figure 6.10 is an instantiation of the user interface template for a meta-model annotation. Appendix E presents and visualizes the graphical user interfaces of

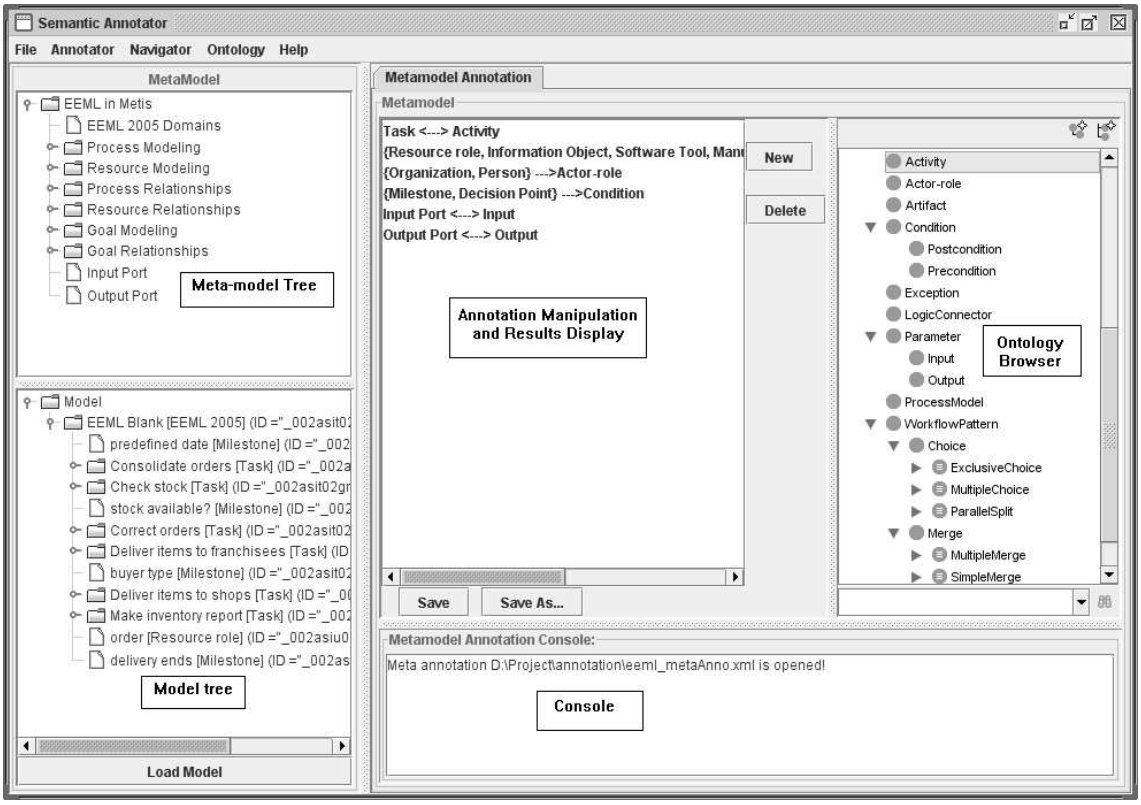


Figure 6.10: Main components of the UI in Pro-SEAT

Pro-SEAT for each annotation phase.

In the current version of Pro-SEAT, a navigator UI provides the function of query-based model navigation (Figure 6.11). The process knowledge user can select a reference ontology concept as the query word and submit the query to the system. During the query, the system can make inference based on the ontology. Therefore, the system would return the model elements annotated with not only the given ontology concept but also the inferred concepts of the given ontology concept. Inferences concerned in a query includes semantic equivalence, subsumption and aggregation relationships between concepts.

6.5 Summary

The proposed annotation approach could be realized through different tool implementations by various technologies. The prototype developed in this work shows one possible implementation. This annotation tool provides a way of annotating meta-models and model elements with OWL ontological concepts. The current version of the model annotation tool is mainly designed for interpreting models created by the Metis tool. It can be extended to import other models produced by different modeling tools. Because the tool integrates the Protégé-OWL API, it can load any OWL ontology created by Protégé.

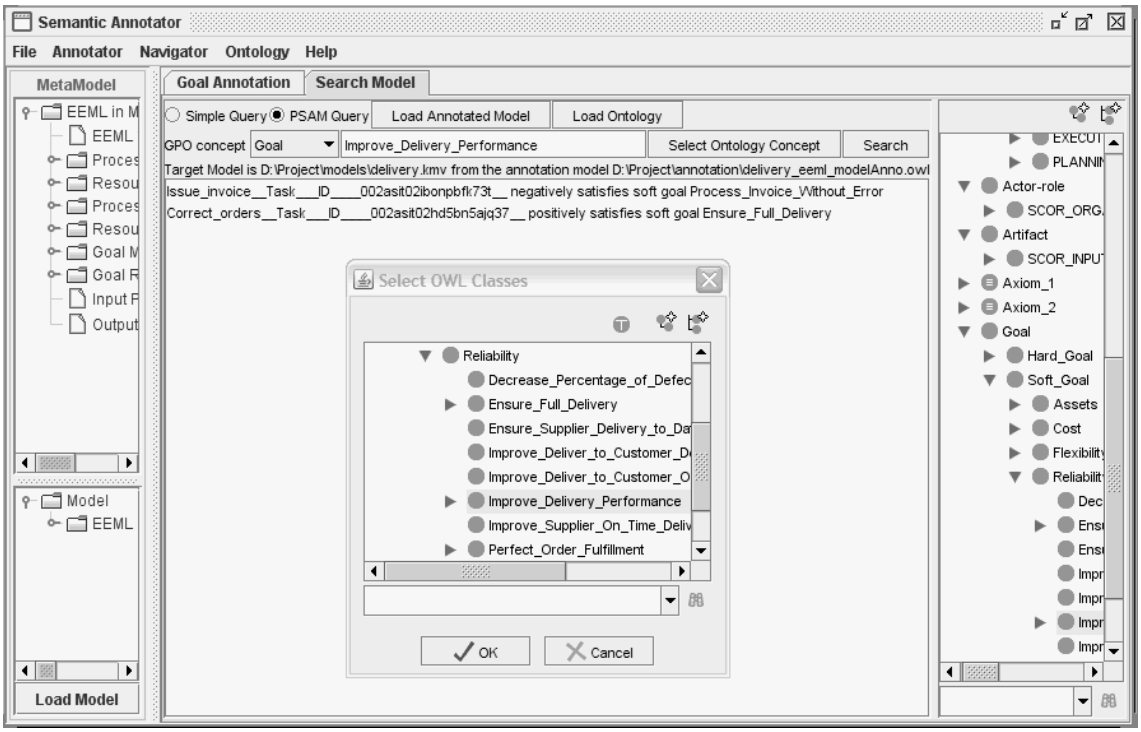


Figure 6.11: The UI of process knowledge navigator in Pro-SEAT

Annotation results are stored in OWL files which are separated from the original models. Links to the original models are specified in the annotation results. It indicates that one model may have more than one annotation file. Different annotation results might be used for different applications.

Annotation results are modeled in an annotation schema which is built on GPO – a process meta-model ontology. Hereby, the annotation schema is defined as ontology properties of GPO. The annotation schema can be extended through extending properties of GPO. The annotation schema is independent of this annotation tool and it can be interpreted by any OWL reader.

One of the limitations of the tool is a lack of the visual model explorer. Only a tree view of the model hierarchy is provided. Moreover, when re-generating the PSAM models from a meta-model annotation, the results of the model and the goal annotation will be lost. Most of the annotation work is still manual. Automatic annotation using the mapping algorithm can be considered in the further development of the annotation tool.

Chapter 7

Exemplar Studies and Application System

Having the methodology guidance and tool support, we can demonstrate the semantic annotation approach in exemplar studies. In the study, we have process models that describe a same business domain (logistics process) but are modeled in different modeling languages and by different enterprises. We need domain and goal ontologies about such a business domain for the annotation. Since there are no formal logistics ontology available, we formalize the SCOR (Supply Chain Operations Reference-model) [163] specifications into logistics domain and goal ontologies using OWL. Following the semantic annotation framework and methods, we deploy the annotation by using Pro-SEAT. Complete annotation results are used in the evaluation phase to validate the applicability of the semantic annotation approach in a process knowledge management application. Therefore, a system architecture is also depicted in this chapter to exploit a process knowledge management application based on the semantic annotation framework.

7.1 Semantic Annotation Procedure

The semantic annotation employed in the exemplar studies consists of four phases: profile annotation, meta-model annotation, model annotation and goal annotation (Figure 7.1). In a profile annotation phase, the annotator inputs basic information following the format of metadata which has been defined in Table 4.1. A general process ontology is employed to map process modeling constructs in a meta-model annotation. Based on the meta-model annotation results, PSAM can be initially generated for a new model annotation. The model content represented by PSAM is then annotated with the domain ontology in the model annotation phase. A goal annotation is employed as a succeeding step of the model annotation, i.e. annotating PSAM with goal ontology. Generally, the meta-model, model and goal annotation phases should be implemented step by step in a sequence, but the profile annotation can be made at any time. For example, the URI of the domain ontology can be inputed when the domain ontology is loaded for the model annotation. After completing all the four phases, process models become the process knowledge represented in PSAM models.

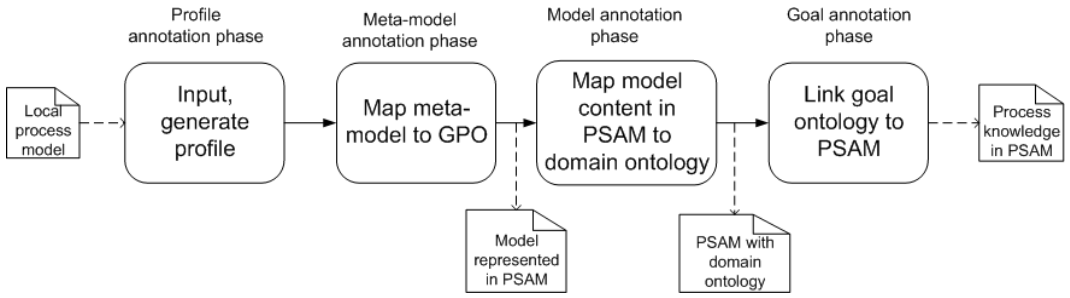


Figure 7.1: Semantic annotation process based on PSAM

7.2 Exemplars

Three process models have been chosen in order to explicate how the proposed semantic annotation framework and approach are applied in the process knowledge management of heterogeneous process models. The models are from two different enterprises, and they are created in different modeling languages. The business processes described in those models are from logistics domain. PM_A is from enterprise A and it is a purchase order process made in BPMN [12], whilst two other models PM_{B1} (an item receiving process) and PM_{B2} (an item delivery process) from logistics department in enterprise B are built in EEML [77]. The models depicts different detail of process knowledge, because two enterprises have different ways of dealing with their business. For example, the delivery process models for enterprise B are relatively simple compared with the delivery processing in enterprise A. However, both of them can achieve the same goals.

7.2.1 Sales logistics process in BPMN

The case of enterprise A is taken from [48]. PM_A is a big model covering most detail processes of sales logistics. The process model includes the following main activities — *Client RFQ processing*, *Client quotation processing*, *Standard order processing*, *Delivery processing*, *Shipping processing* and *Bill processing*.

Four actors are involved in PM_A — client, sales department, financial department, and logistics department. *Client RFQ processing*, *Client quotation processing*, *Standard order processing* and *Bill processing* are mainly carried out by sales department and interacted with client. *Delivery processing* and *Shipping processing* are the main activities in the logistics department. Both *Delivery processing* and *Standard order processing* need the task credit control in the financial department.

Client RFQ processing and client quotation processing

In PM_A, the client RFQ processing can produce the quotation created from the inquiry or the inquiry is rejected if it is invalid. Analogously, after the client quotation processing, the validity of the quotation is checked. The quotation might be rejected if it is invalid, otherwise it can be referred to by an order in the standard order processing.

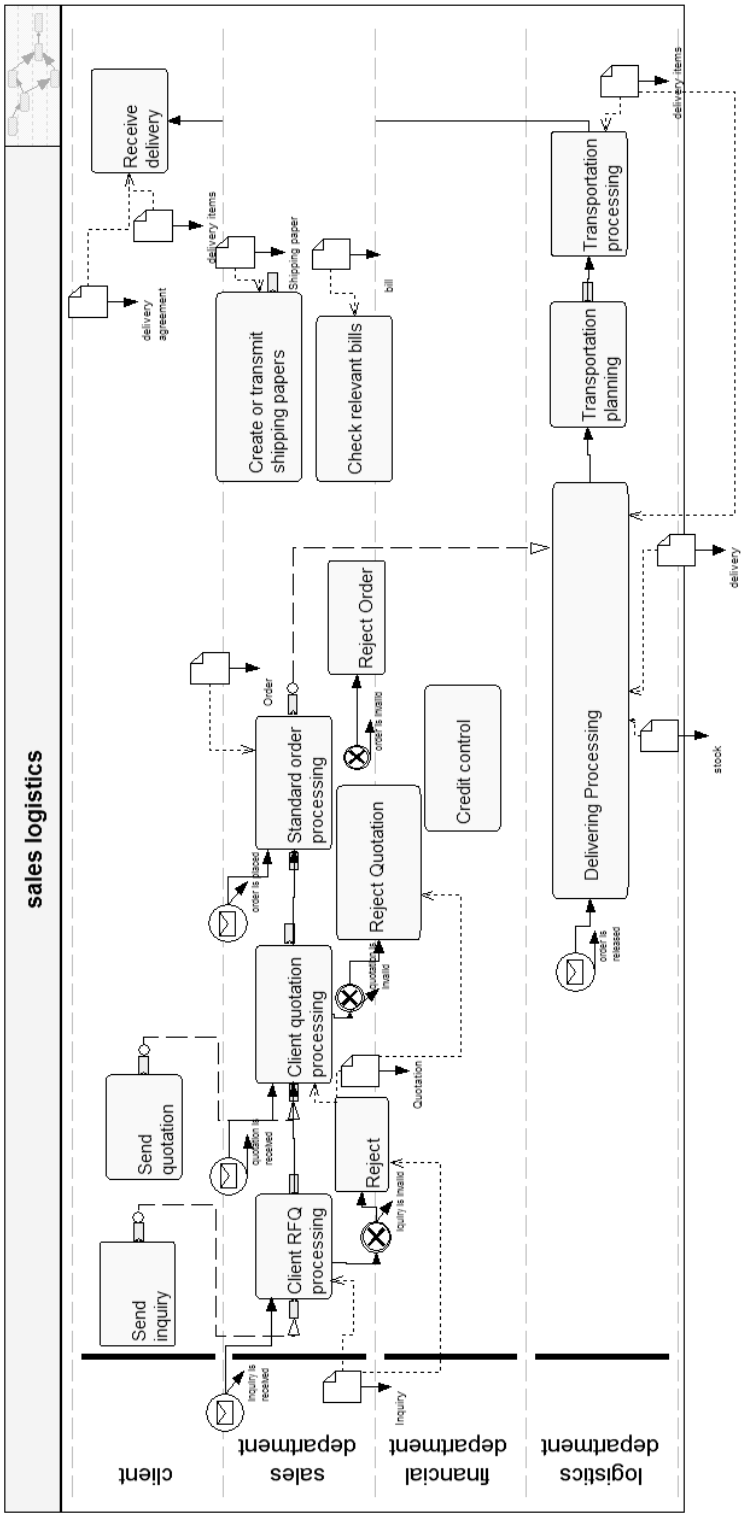


Figure 7.2: Sales logistics process of enterprise A in BPMN

Standard order processing and credit control

Standard order processing begins when an order is placed. In this processing, the order must be checked and edited accordingly. The credit control in financial accounting is carried out just following the task – edit order. After credit control, one of three tasks is possible: 1)accept order, 2)block order, or 3)reject order.

Delivery processing

The delivery will be opened when the delivery date takes place. Once opened, the credit control is carried out to check if the credit requirements are fulfilled. If the credit control is successful, a series of tasks follow, such as determine or transfer delivery route, determine serial numbers of delivery item, check the stock of the delivery items, determine picking place and delivery batch, check the availability of delivery items (fully or partially available). If the partial delivery is not allowed, the delivery quantity can be accordingly corrected or items are rejected. Next, the delivery items are picked completely or partially. The task – pick delivery items, can be iterated if the partially picked occurs and the picking is carried out again. Pack delivery items follows the task of picking delivery items. As a result, a delivery is created at the end of the delivery processing.

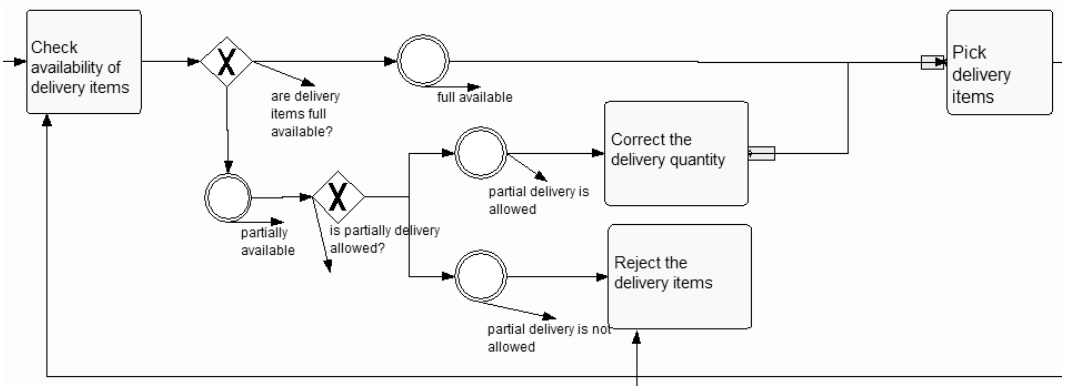


Figure 7.3: Checking availability of the delivery of enterprise A in BPMN

Shipping processing and bill processing

The shipping processing starts with the task of transportation planning after the delivery processing. Based on the planning, the transportation processing takes place. The shipping papers and the bill are consequently created and checked.

The original models of PM_A displayed in Figure 7.2. Some details of the delivering are illustrated in Figure 7.3 and Figure 7.4. The main processing and tasks described in the exemplar are modeled as the BPMN Logical Processes and are allocated into different Swimlanes which represent the four actors in the BPMN model. Sequences of the processing are represented by the control and flow notations.

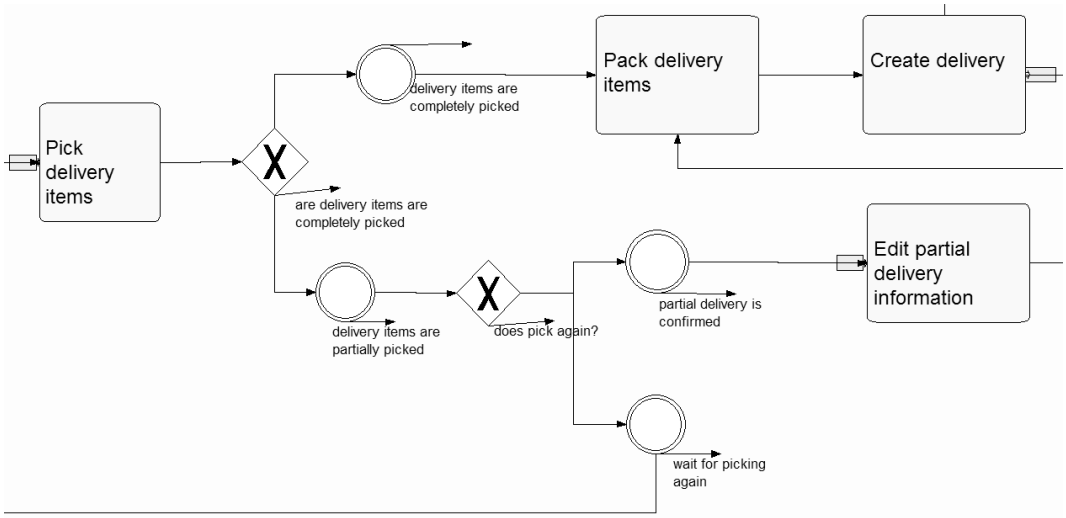


Figure 7.4: Picking, packing and create delivery of enterprise A in BPMN

7.2.2 The TelCo item receiving and delivery process in EEML

The second case is from INTEROP project [145]. Enterprise B is specialized in telecommunications, in production and distribution of batteries, as well as in retail sales of everyday technology products. The company is called TelCo. TelCo does not have its own warehouse but uses the services of logistics company – Orbit Ltd. But TelCo has its logistics department who is responsible for items receiving and delivering. Thus the main functions of logistics department are to make orders and control Orbit Ltd.

In PM_{B1} and PM_{B2}, the item receiving process and the item delivery process are described as a series of EEML tasks. The actors in the PM_{B1} are commercial department, logistics department and Orbit warehouse, whilst the actors in the PM_{B2} are logistics department, IT, sales department, financial department, Orbit warehouse, franchisee and shop. Those actors are modeled as the EEML organization in both models.

Item receiving process

PM_{B1} depicts the items receiving process of the TelCo company (Figure 7.5). The logistics department receives orders to suppliers about expected quantities of items, and then starts to receive the items. After receiving the items, the logistics department checks the items.

These are sub-tasks included in some tasks in the model because of three types of items receiving — regular orders to local suppliers, consignment and import. In the case of the regular orders to local suppliers, they deliver the items with protocols and issue an invoice. In the case of the import deliveries, sometimes there are differences in quantities received from a foreign supplier. The department issues a protocol for the whole quantity and another protocol for the deficit with minus quantity. The second protocol is followed by an export invoice or an invoice to the insurance company. The process details are described in the decomposition of the task *check import items* in Figure 7.6. After checking the items, the received items will be transferred to Orbit.

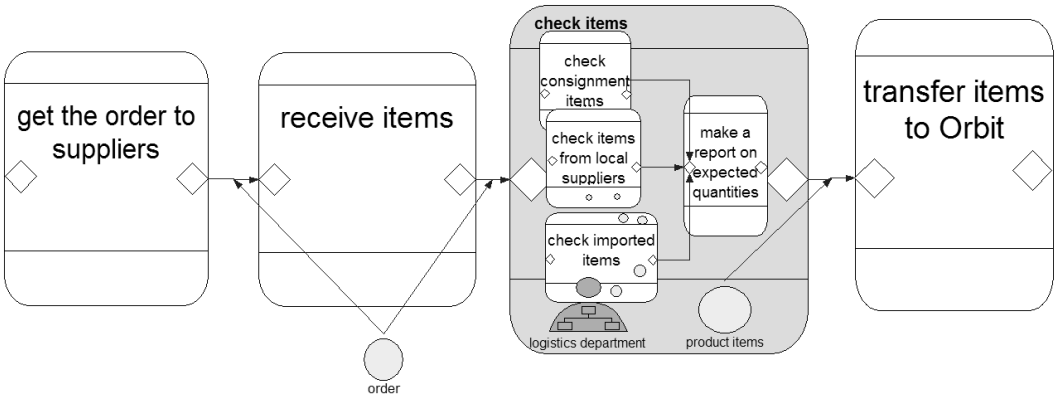


Figure 7.5: TelCo item receiving process

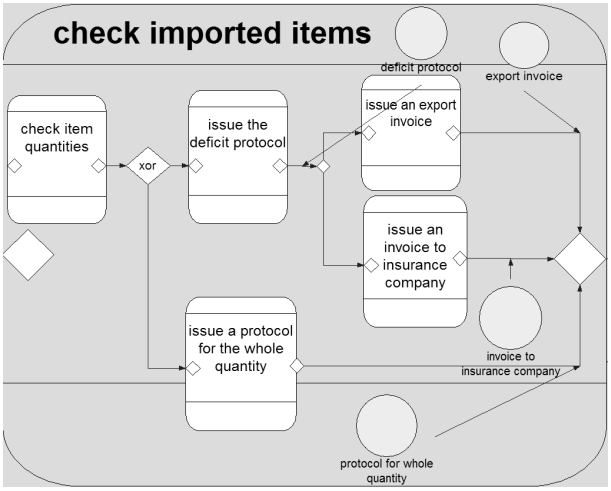


Figure 7.6: Decomposition of the *check items*

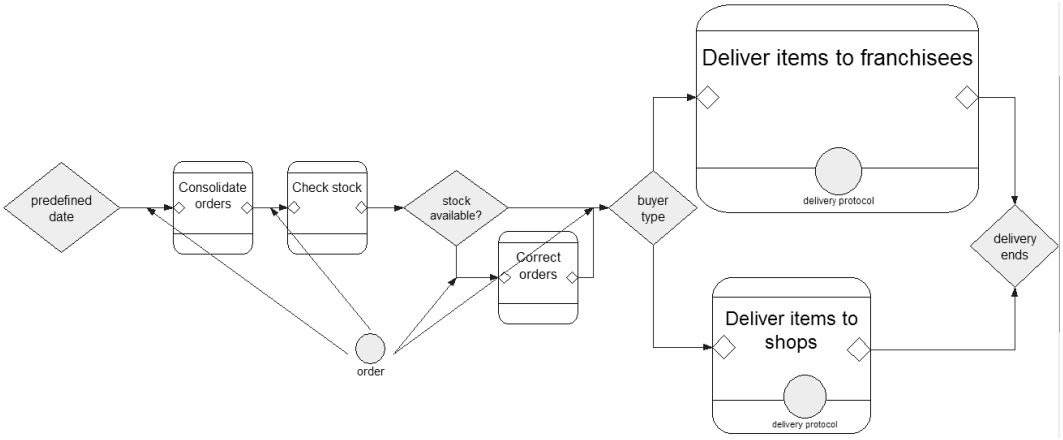


Figure 7.7: The item delivery process of enterprise B in EEML

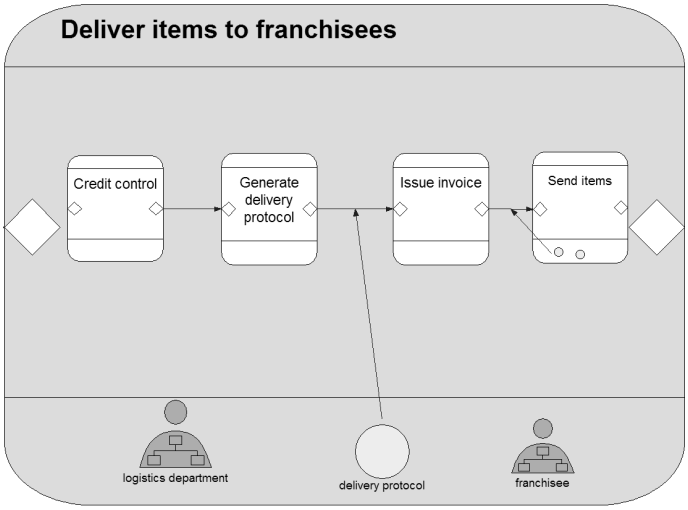


Figure 7.8: The delivery process to franchisees of enterprise B in EEML

Meanwhile the department sends a report to Orbit in order to inform them what to expect.

Item delivering process

PM_{B2} is the item delivery process in the TelCo case. Items are delivered to shops and franchisees. On a certain date (predefined) all orders from one shop are consolidated. The orders are checked against the available stock. The orders are corrected if there is not enough items in the stock. The order correction procedure is set by the sales department and it is processed using an external application in the IT department. After all the orders for a day are approved, items are to be delivered. If it is a franchisee order, it is followed by a delivery protocol and an invoice. The invoice is issued in the Orbit warehouse and is sent to the customer along with the items. If it is an order from a shop, there follows only a delivery protocol.

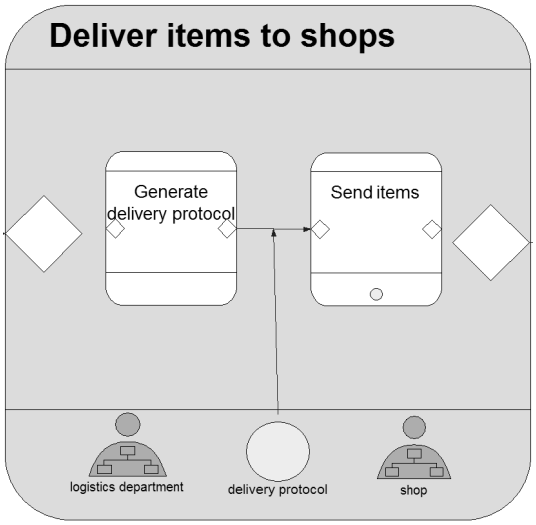


Figure 7.9: The delivery process to shops of enterprise B in EEML

The delivery processing of PM_{B2} describes only a part of the sales logistics process. PM_{B2} in EEML is illustrated in Figure 7.7, and some of the process detail is displayed in Figure 7.8 and Figure 7.9.

7.3 SCOR Reference Ontology

SCOR is a process reference model that has been developed and endorsed by the Supply-Chain Council as the cross-industry standard diagnostic tool for supply-chain management. A SCOR reference model has been developed to describe the standard business activities associated with all phases of satisfying a customer's demand. As standards, the model can be formalized as a reference ontology for supply-chain management domain.

There are three level process details in the reference model. The top level defines the scope and content of SCOR. Five process types — Plan, Source, Make, Deliver and Return are defined at this level. The second level is the configuration level defining the core "process categories". The third level is the process element level, decomposing the process categories into the process elements. The process element level consists of process element definitions, process element information inputs, and outputs, process performance metrics, best practices, system capabilities required to support best practices, and systems/tools. As an example, a logic flow of the SCOR level 3 process elements is depicted in Figure 7.10.

The level 3 process elements provide enough details of references for domain activities. In this work, we model domain ontology concepts based on the SCOR process elements at level 3 for model annotation purpose. The SCOR ontology from the INTEROP project [62] is extended. The extended ontologies include `SCOR_INPUT_OUTPUT`, `SCOR_MGM_PROCESS` and `SCOR_ORGANISATION`. `SCOR_INPUT_OUTPUT` are objects which are needed or produced by SCOR process elements. `SCOR_MGM_PROCESS` are the SCOR process elements at level 3 which are organized in a hierarchy following the SCOR

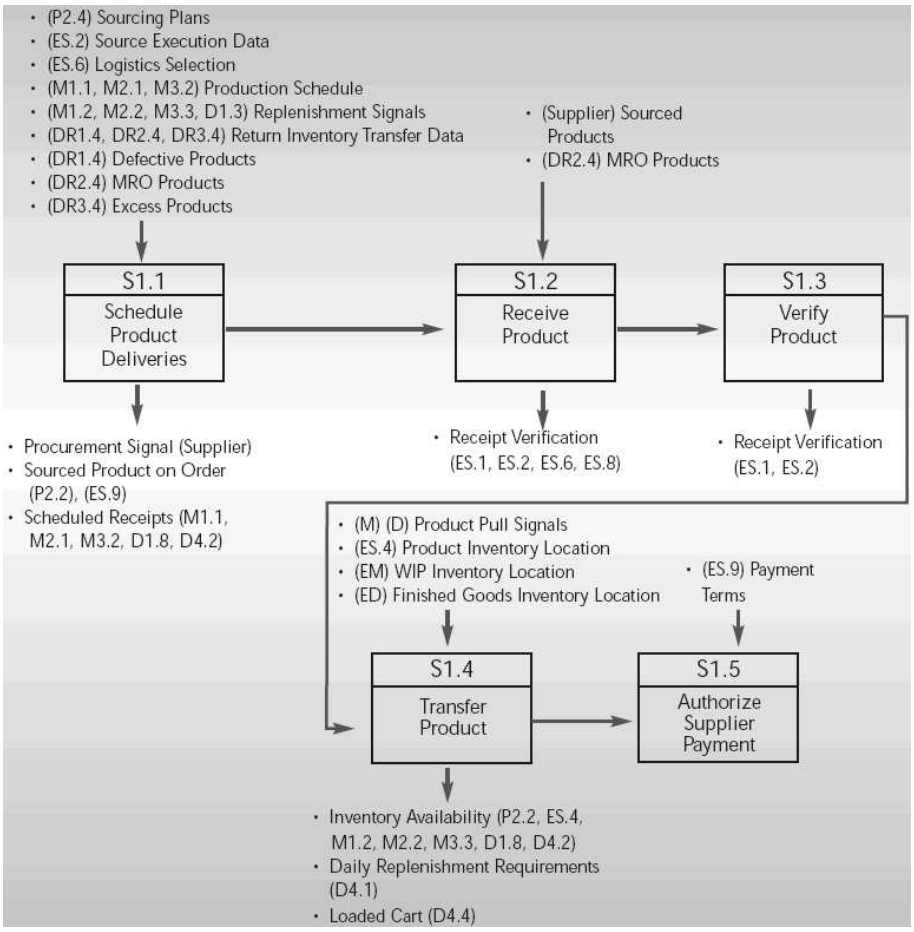


Figure 7.10: SCOR S1 process of sourcing stocked product [164]

levels (from level 1 to level 3). `SCOR_ORGANISATION` are organizations/persons who are involved in the process elements. The goal ontology in this domain is also from SCOR. Usually the hard goals are derived from process elements and also from their inputs and outputs. Performance attributes defined in SCOR can be modeled as a set of *General Soft Goal Category* such as **Reliability**, **Responsiveness**, **Flexibility**, **Cost**, and **Assets**. Domain specific soft goals are derived from the metrics of performance attributes [167].

The SCOR ontology is formalized in OWL. To organize those concepts, we categorize domain concepts with the 3A concepts (***Activity***, ***Artifact***, ***Actor-role***) defined in GPO. The purpose of the categories is to establish the mapping relationship between a PSAM model and the reference ontology when annotation. The categories and concepts are modelled as OWL Classes, and they are organized by a subsumption hierarchy.

The ontology concepts which are organized in "SCOR_MGMT_PROCESS" are sub-classes of ***Activity***, the input/output ontology concepts ("SCOR_INPUT_OUTPUT") are sub-classes of ***Artifact*** and the organizational ontology concepts ("SCOR_ORGANIZATIONAL") are sub-classes of ***Actor-role***.

Table 7.1: OWL definition of the SCOR ontology

OWL Ontology Class	Subsumption Relation	OWL Properties	Property Range
SCOR_MGMT_PROCESS	`owl:subClassOf` Activity	`hasInput`	*multiple* SCOR_INPUT_OUTPUT
		`hasOutput`	*multiple* SCOR_INPUT_OUTPUT
		`precedes`	*multiple* SCOR_MGMT_PROCESS
		`isPrecededBy`	*multiple* SCOR_MGMT_PROCESS
SCOR_INPUT_OUTPUT	`owl:subClassOf` Artifact	`isInputTo`	*multiple* SCOR_MGMT_PROCESS
		`isOutputOf`	*multiple* SCOR_MGMT_PROCESS
		`has_state`	*(data property inherited from Artifact*
SCOR_ORGANIZATIONAL	`owl:subClassOf` Actor-role		
Goal	*has sub-Class* Hard Goal, Soft Goal	`has_parts`	*multiple* Goal
		`part_of`	*multiple* Goal
		`targetActivity`	*multiple* Activity
		`targetArtifact`	*multiple* Artifact
		`targetRole`	*multiple* Actor-role
		`targetConstraint`	*(data property)*

Each SCOR_MGMT_PROCESS has object properties `hasInput` and `hasOutput` to specify inputs and outputs of each process element. The object properties `precedes` and `isPrecededBy` indicate logic flows between the process elements. Inversely, SCOR_INPUT_OUTPUT has the object properties `isInputTo` and `isOutputOf` to relate itself to SCOR_MGMT_PROCESS. A data property `has_state` is defined as a property of ***Artifact***. This property can be used associated with `isInputTo` and `isOutputOf` properties to indicate that a same artifact with different states represents different inputs/outputs.

Goal ontology concepts are organized into ***Hard*** and ***Soft Goals***. Soft goals are further organized into some general soft goal categories. We integrate the domain ontology and goal ontology into one OWL file, because they are both derived from SCOR descriptions. According to the semantic representation of the goal ontology in Chapter 5, the values of the object properties `targetActivity`, `targetArtifact`, and `targetRole` are from the domain ontology Activity, Artifact and Actor-role respectively. `TargetConstraint` is modeled as a data property since there is no corresponding ontological definition in the domain ontology. A pair of inverse properties `has_parts` and `part_of` are defined for a goal concept to represent the semantics of goal decomposition. An OWL model of the SCOR ontology is presented in Table 7.1.

The identified domain and goal concepts from the SCOR specifications are modeled as sub-Classes of the above categories. Values of the properties for a SCOR Class are specified through the OWL restrictions. As examples, Figure 7.11, Figure 7.12, Figure 7.13 and Figure 7.14 illustrate the sub-Classes of the SCOR_MGMT_PROCESS, SCOR_INPUT_OUTPUT, Hard Goal and Soft Goal in the Protégé-OWL editor.

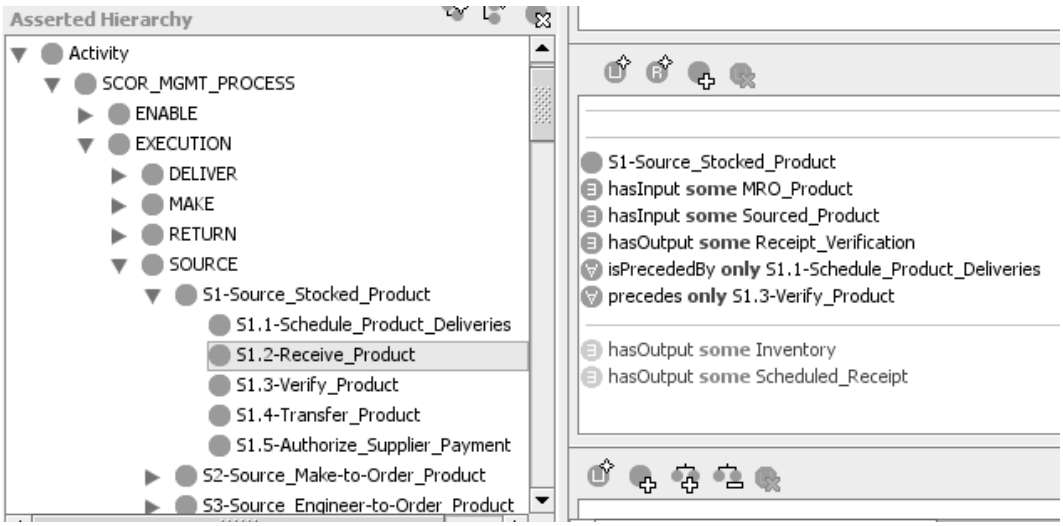


Figure 7.11: SCOR process element `S1.2 Receive Product` in Protégé-OWL editor

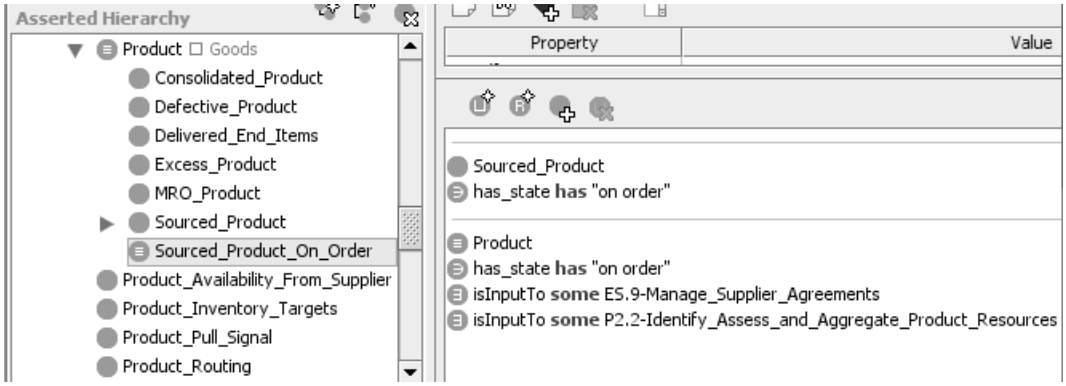


Figure 7.12: SCOR input/output `Sourced Product On Order` in Protégé-OWL editor

7.4 Annotation of Process Models with Pro-SEAT

In this section, we will describe the annotation of PM_A, PM_{B1} and PM_{B2} with the Pro-SEAT annotation tool. The annotation follows our semantic annotation framework and approach, including profile annotation, meta-model annotation, model annotation and goal annotation.

7.4.1 Profile annotation

The descriptive and technical information about models in the profiles are important in this scenario. Title and description of models may help to identify the model content in general. The title of PM_A is "sales logistics process" and the main processing is depicted in the description. The titles of PM_{B1} and PM_{B2} are "stock logistics process" and "delivery logistics process" respectively. In this scenario, the three models are classified under the same category "Business and Economy" [211] and domain is "SCO

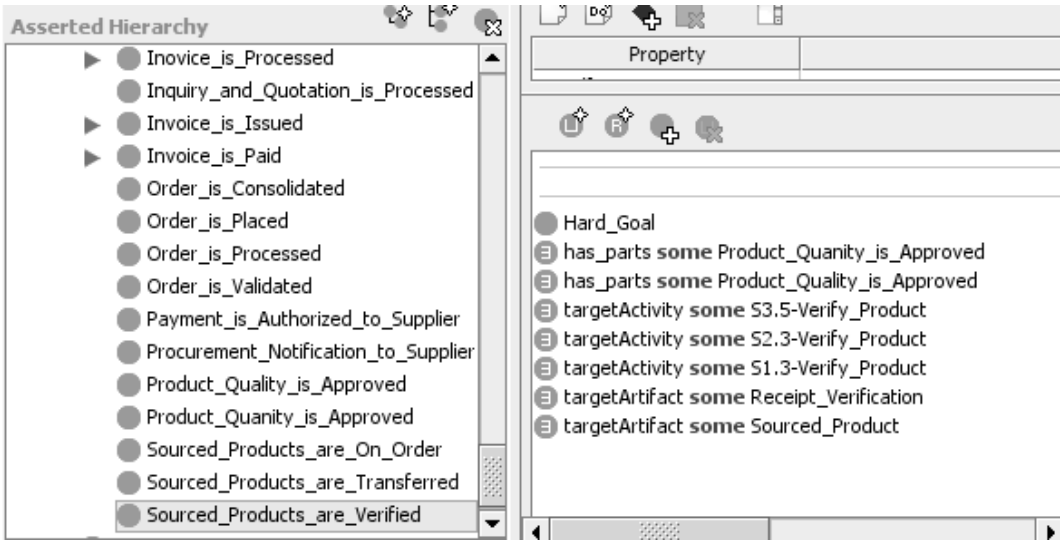


Figure 7.13: SCOR hard goal `Sourced Products are Verified` in Protégé-OWL editor

(Supply Chain Operations)". The SCOR ontology is therefore a domain ontology in the exemplar studies. We assume that the reference domain ontology applied in the annotation is SCOR (Supply Chain Operations Reference-model) [163]. The URI and URL of SCOR ontology are specified in the profile annotation. Technical information about PM_A shows that the modeling language is BPMN, the language_version is 1.0, and the modeling tool is Metis 5.2.2. PM_{B1} and PM_{B2} are also built with the tool Metis 5.2.2 but the modeling language is EEML and the version of EEML is 2005. Part of profile annotation of PM_A is exemplified in a tag language as follows.

```
...
<dc:title>sales logistics process</dc:title>
<profileAnno:category>Business and Economy</profileAnno:category>
<profileAnno:domain>SCO</profileAnno:domain>
...
<profileAnno:modeling_langugage>BPMN</profileAnno:modeling_language>
<profileAnno:language_version>1.0</profileAnno:language_version>
<profileAnno:modeling_tool>Metis 5.2.2<profileAnno:modeling_tool>
...
```

7.4.2 Meta-model annotation

The annotation of meta-models in this scenario includes annotating BPMN 1.0 and EEML 2005 with GPO. The mapping rules as defined in the semantic annotation framework, meta-model elements can be one-to-one, many-to-one or combined-to-one mapped to a GPO concept. Table 7.2 shows the mapping results for meta-model annotation.

During the implementation of the annotation tool, we made practical adjustments to the GPO as follows. ***WorkflowPattern*** is simplified by specifying sequences of activ-

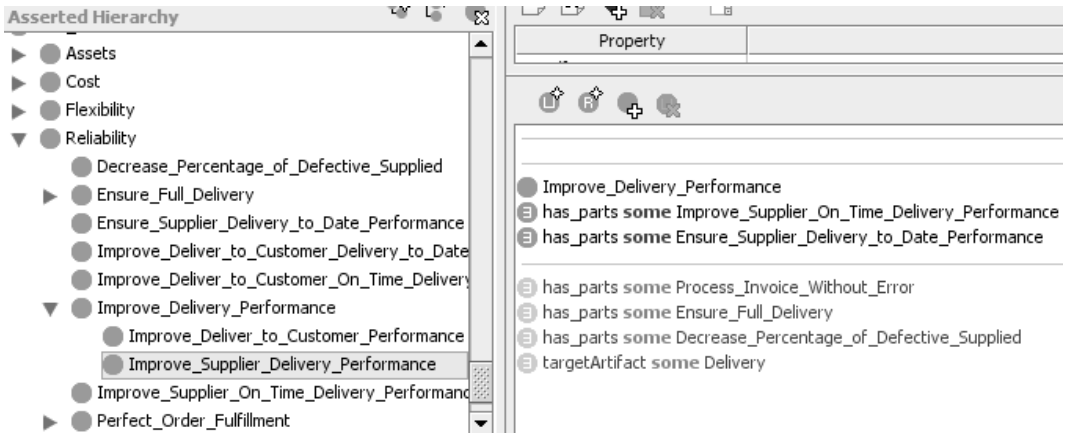


Figure 7.14: SCOR soft goal `Improve Supplier Delivery Performance` in Protégé-OWL editor

Table 7.2: Mapping the EEML and BPMN meta-model elements to the GPO concepts

GPO concepts	EEML elements in METIS 5.2.2	BPMN elements in METIS 5.2.2
Activity	Task	Logical Process
Actor-role	Organization, Person	Swimlane including Horizontal Swimlane and Vertical Swimlane
Artifact	Resource, Role, Information Object, Software Tool, Manual Tool, Material Tool, Location	Data Object, property 'Data' in Event
Input	Input Port	Input
Output	Output Port	Output
WorkflowPattern	Combination of Flow, Milestone and Decision Point	Combination of Event (including Start Event, Intermediate Event, End Event), Sequence Flow, Gateway
Precondition	Milestone, Decision Point	Event (including Start Event, Intermediate Event), Gateway
Postcondition	Milestone, Decision Point	Event (including End Event, Intermediate Event), Gateway
Exception	Decision Point	Event (when event type is error) together with Sequence Flow

ities with the properties `has_precedingActivities` and `has_succedingActivities` of ***Activity***. Moreover, mapping the same meta-model element to two different GPO concepts is not supported, because it produces the identical instances of different OWL Classes (e.g. a same milestone in the EEML model will be the instance of ***Precondition*** and the instance of ***Postcondition*** at the same time). This violates the OWL modeling principle that instances of different OWL Classes should be unique. A solution of the implementation of Pro-SEAT is to modify the GPO ontology by inserting an abstract concept ***Condition*** as the super-Class of ***Precondition*** and ***Postcondition***. The mapping between the modeling elements and Precondition/Postcondition is therefore transferred to the mapping between the modeling elements and ***Condition***. The distinction of ***Precondition*** and ***Postcondition*** (i.e. which model element is the instance of Precondition and which model element is the instance of Postcondition) can be specified in the model annotation phase through the properties `has_Precondition` and `has_Postcondition` of an ***Activity***.

After the meta-model annotation, three models can be represented using GPO

concepts through the meta-model mapping. The contents in different models are then expressed in a same process knowledge representation language, e.g. an *Activity* named "check availability of delivery item" has an $Actor-role$ called "logistics department" in enterprise A (originally represented in PM_A as *a logical process "check availability of delivery item" in a swimlane "logistics department"* in BPMN) vs. an *Activity* named "check stock" has an $Actor-role$ called "logistics department" in enterprise B (originally represented in PM_{B2} as *an organization object "logistics department" participates in the task "check stock"* in EEML).

7.4.3 Model annotation

In the model annotation phase, model elements as GPO instances will be annotated with reference ontologies. Since PM_A and PM_{B2} are about the delivery process, the reference ontologies for annotating the ***Activities*** are mainly from the category `DELIVER` under `SCOR_MGMT_PROCESS`. Furthermore, the products are stocked products so that the process elements of `D1 Deliver Stocked Product` are the corresponding references. For PM_{B1}, most of the Activities refer to the process elements of `S1 Source Stocked Product`. ***Artifacts*** and ***Actor-roles*** refer to the `SCOR_INPUT_OUTPUT` and `SCOR_ORGANIZATIONAL` for all three models.

Relationships between model elements such as `has_subActivity`, `has_Input`, `has_Output`, etc. (defined in GPO) should be instantialized in the model annotation phase. Some of the relationships can be directly transformed from the original process models, and some are specified by the annotation user. Such annotations can enrich the implicit semantics in the original models or compensate for the lost information caused by the transformation from the meta-model annotation.

7.4.4 Goal annotation

For the goal annotation, the semi-automatic goal annotation function of Pro-SEAT is run in the exemplar studies. Through the execution of goal annotation algorithms defined in section 6.3, the possible goal annotation results can be deduced automatically based on the model annotation information. Pro-SEAT provides a list of goal options by matching the SCOR goal ontology with the PSAM models. From the list, annotation users then select suitable goals manually to annotate the model fragments.

7.4.5 Annotation results

Complete annotation results of three models are presented in Appendix H in OWL. For the sake of representation clarity and limited space, in this section we represent parts of the annotation results of three models in Table 7.3, Table 7.4 and Table 7.5. The first column of the tables lists the PSAM Activity instances which are also the model elements of *Logic Process* in PM_A or *Task* in PM_{B1} and P_{B2}. The second column is the model annotation result: the ***Activities*** are annotated with SCOR process elements or categories through the semantic relationships `same_as`, `kind_of` and `phase_of`. The third and fourth columns are the goal annotation results. The SCOR goal ontology is associated with each Activity instance in the case of achieving certain hard goals,

positively and/or negatively satisfying soft goals. Details of the annotation analysis are depicted in Appendix F.

After annotating all the ***Activities***, ***Artifact***, ***Actor-role***, ***Input*** and ***Output*** of a PSAM model with domain ontology, the knowledge representation of those process models is aligned with the SCOR ontology. Thus the process knowledge is explicit and open to a third party who is interested in model exchange, system integration and business cooperation in the SCOR domain. Goal-oriented analysis and goal-driven search of process models and model fragments can be deployed based on the SCOR goal ontology which is referenced in the annotation. For instance, the annotation information can help an analyst to find out the processes which are related to the verification costs.

7.5 Process Knowledge Management System Based on Semantic Annotation

Generally knowledge management (KM) refers to a range of practices used by organizations to create, codify, and disseminate knowledge for reuse, awareness and learning within or across the organization. In the context of the thesis, process models from different enterprise systems are distributed process knowledge to be managed for the cooperation business. The hypothesis is that process models have been represented in our process annotation models — PSAM models in OWL, which is a way to represent process knowledge. However, enterprises need a service to manage the desired knowledge. Therefore, the management service should provide the query functions to set users' search conditions and then return search results. The search conditions can be set based on profiles or contents of process models. The query should not only be keyword-based but also ontology-based, so the semantic relationships among queries and models/model fragments should be machine-interpretable during the query. Moreover, users need the system to help them "discover" knowledge, to do this users need only provide the business goals and the system should return the process models/knowledge fulfilling the goals. Since the process knowledge is represented by process models, the search results are models or model fragments. No matter which way to represent the models — the visual display or the textual description, the knowledge management service should allow the users to navigate the knowledge conveniently, such as providing both a complete view and a partial view of models/model fragments.

Following our semantic annotation framework, an enterprise can use the semantic annotation client tool to annotate model resources. The client tool sends the annotation models to the server through which the process knowledge is stored into a knowledge repository. Since the annotation models are represented in OWL — XML-based files, the repository is an XML-based repository. The process knowledge query, discovery and navigation functions can also be implemented as user interfaces on client side. Knowledge users can edit the search queries and business goals on the client side and submit the queries and goals to the server. The search and discovery services are executed on the server. The search results are returned from the server to clients. Finally, users can navigate the returned process knowledge/models on the client side.

Table 7.3: Part of semantic annotation results of PM_A

PSAM Activity Instance	Semantic Annotation	Goal Annotation Relations	SCOR Goal Ontology
Client RFQ processing	`phase_of` D1.1 Process Inquiry and Quote	`achieves`	Inquiry and Quotation is Processed
Client quotation processing	`phase_of` D1.1 Process Inquiry and Quote	`achieves`	Inquiry and Quotation is Processed
Standard order processing	`same_as` D1.2 Receive Enter and Validate Order	`achieves`	Order is Consolidated; Order is Placed; Order is Processed; Order is Validated
Credit Control	`phase_of` D1.2 Receive Enter and Validate Order `kind_of` ED.5 Manage Deliver Capital Assets	`achieves`	Order is Validate
		`positively_satisfies`	Reduce Order Entry and Maintenance Costs; Increase Return on Deliver Capital Assets
		`negatively_satisfies`	Reduce Order Processing Time
Monitor delivery date	`phase_of` ED.1 Manage Deliver Business Rules	`positively_satisfies`	Improve Deliver to Customer Delivery to Date Performance; Improve Deliver to Customer On Time Delivery Performance
Check delivery items	`phase_of` D1.3 Reserve Inventory and Determine Delivery Date	`positively_satisfies`	Improve Deliver Performance; Raise Order Quantity Fillrate
Check availability of delivery items	`phase_of` ED.3 Manage Deliver Information	`positively_satisfies`	Ensure Full Delivery; Improve Deliver to Customer On Time Delivery Performance; Improve Delvier to Customer Delivery to Date Performance
Create delivery	`kind_of` ED.3 Manage Deliver Information	`achieves`	Delivery is Scheduled; Delivery Terms are Generated
		`positively_satisfies`	Ensure Full Shipment; Ensure Full Delivery
Transportation planning	`same_as` D1.7 Select Carriers and Rate Shipments	`achieves`	Delivery is Scheduled
		`positively_satisfies`	Ensure Full Shipment; Reduce Order Shipment Time
Transportation processing	`same_as` D1.12 Ship Product	`achieves`	End Items are Delivered

Table 7.4: Part of semantic annotation results of PM_{B1}

PSAM Activity Instance	**Semantic Annotation**	**Goal Annotation Relations**	**SCOR Goal Ontology**
Get the order to suppliers	`phase_of` S1.1 Schedule Product Deliveries	`achieves`	Sourced Product are On Order; Order is Placed
		`positively_satisfies`	Reduce Order Processing Time; Reduce Order Receipt Time
Check items from local suppliers	`kind_of` S1.3 Verify Product	`achieves`	Sourced Product are Verified
		`positively_satisfies`	Reduce Order Receipt Time
Check consignment items	`kind_of` S1.3 Verify Product	`achieves`	Sourced Product are Verified; Product Quality is Approved
		`positively_satisfies`	Reduce Order Receipt Time; Reduce Verification Costs
Check imported items	`kind_of` S1.3 Verify Product	`achieves`	Sourced Product are Verified; Procurement Notification to Supplier
		`negatively_satisfies`	Reduce Order Processing Time; Improve Supplier On Time Delivery Performance; Process Invoice Without Error; Reduce Order Processing Costs; Reduce Order Receipt Time; Reduce Verification Costs
check the item quantities	`phase_of` S1.3 Verify Product	`achieves`	Product Quantity is Approved
Issue the deficit protocol	`phase_of` ES.9 Manage Supplier Agreements	`achieves`	Procurement Notification to Supplier
		`negatively_satisfies`	Reduce Order Processing Time; Improve Supplier On Time Delivery Performance; Reduce Order Processing Costs; Reduce Order Receipt Time
Issue an export invoice	`phase_of` S1.5 Authorize Supplier Payment	`achieves`	Payment is Authorized to Supplier
		`negatively_satisfies`	Reduce Order Processing Time; Reduce Order Processing Costs; Reduce Order Receipt Time; Process Invoice Without Error
store items in Orbit	`kind_of` S1.4 Transfer Product	`achieves`	Sourced Products are Transferred; Available Inventory

Table 7.5: Part of semantic annotation results of PM_{B2}

PSAM Activity Instance	**Semantic Annotation**	**Goal Annotation Relations**	**SCOR Goal Ontology**
Consolidate orders	`same_as` D1.4 Consolidate Orders	`achieves`	Order is Processed; Order is Consolidated
		`positively_satisfies`	Reduce Order Processing Costs; Reduce Order Processing Time
Check stock	`phase_of` D1.3 Reserve Inventory and Determine Delivery Date	`achieves`	Order is Validated
		`positively_satisfies`	Raise Order Quantity Fillrate
Correct orders	`phase_of` D1.2 Receive Enter and Validate Order	`achieves`	Order is Validated; Order is Processed
		`positively_satisfies`	Ensure Full Delivery
		`negatively_satisfies`	Reduce Order Entry and Maintenance Costs; Reduce Order Processing Time
Credit control	`phase_of` D1.2 Receive Enter and Validate Order `kind_of` ED.3 Manage Deliver Information	`achieves`	Order is Validate
		`positively_satisfies`	Reduce Order Entry and Maintenance Costs
		`negatively_satisfies`	Reduce Order Processing Time
Generate delivery protocol	`kind_of` ED.3 Manage Deliver Information	`achieves`	Delivery Terms are Generated
		`positively_satisfies`	Improve Deliver to Customer Delivery to Date Performance; Improve Deliver to Customer On Time Delivery Performance
Issue invoice	`phase_of` D1.15 Invoice	`achieves`	Invoice to Customer is Issued
		`negatively_satisfies`	Reduce Invoice to Customer Processing Time; Reduce Invoice to Customer Processing Costs; Process Invoice Without Error
Send items	`kind_of` D1.12 Ship Product	`achieves`	End Items are Delivered
Deliver items to franchisees	`phase_of` D1 Deliver Stocked Product	`achieves`	Invoice to Customer is Issued; End Items are Delivered
Deliver items to shops	`phase_of` D4 Deliver Retail Product	`achieves`	End Items are Delivered
		`positively_satisfies`	Reduce Order Shipment Time

7.5.1 System architecture

The architecture of the system for process knowledge management is shown in Figure 7.15. Since the process knowledge management system is based on the semantic annotations, we include the annotation client tool in the system architecture. In the context of this thesis, the annotation client tool is Pro-SEAT. Annotators works with the annoation client too to annotate the original process knowledge resources. Annotators in this scenario should be model experts and domain experts to assure the annotation quality[1]. The original process knowledge resources — models and meta-models in XML files and XML schema files are stored in the local process model repository.

As references, ontologies are essential in our approach. Ontologies are stored on an ontology server. When annotators employ the annotation with the client tool, the selected ontologies must be loaded from the ontology server. The annotator should select appropriate domain ontologies and goal ontologies according to the ontology categories. The GPO ontology is only applied for the meta-model annotation.

Completing the annotation, annotators submit the annotated process models and/or the annotated meta-models to process knowledge repository server. It is allowed to modify the annotated models stored on the knowledge repository server but the authorization and the authentication are required (which we do not consider in this approach).

Knowledge users might be different from annotators. They do not know about the original process models but the system will assist them to access the desired process models. There are three applications for knowledge users, i.e. process knowledge query, discovery and navigation. Those applications are implemented by client user interfaces and application servers. Through the user interface, users can edit the queries or provide business goals. Queries can be keyword-based or ontology-based. For the ontology-based query, users can browse domain ontologies from the ontology repository and use ontology concepts to form a query. Goal ontologies can be applied by users to set goals for the knowledge discovery. When the application server receives queries or goal requirements from the client, the matchmaker component and the reasoner component co-work with ontologies to find process models. The reasoner uses description logic inference technology on both the ontological query and the ontological annotation, because ontologies and annotations are all in OWL. Based on the inference, the matchmaker matches user's query/goals with annotated models in the repository. The application server sends the search results to the client and the returned knowledge/models can be explored by the client. Users can navigate process knowledge/models from different views according to the process properties defined in PSAM.

7.5.2 Semantic reasoning

The underlying logic basis of OWL is description logic, which can provide the inference services of the knowledge for semantic reasoning (w.r.t. an ontology $\mathcal{O}$) [53]:

- **Consistency.** Check if the knowledge is meaningful. (Is $\mathcal{C}$ is satisfiable w.r.t. $\mathcal{O}$? i.e. $\mathcal{C}^I \neq \emptyset$ in some model $\mathcal{I}$ of $\mathcal{O}$?)

[1]The quality of annotation affects the quality of knowledge.

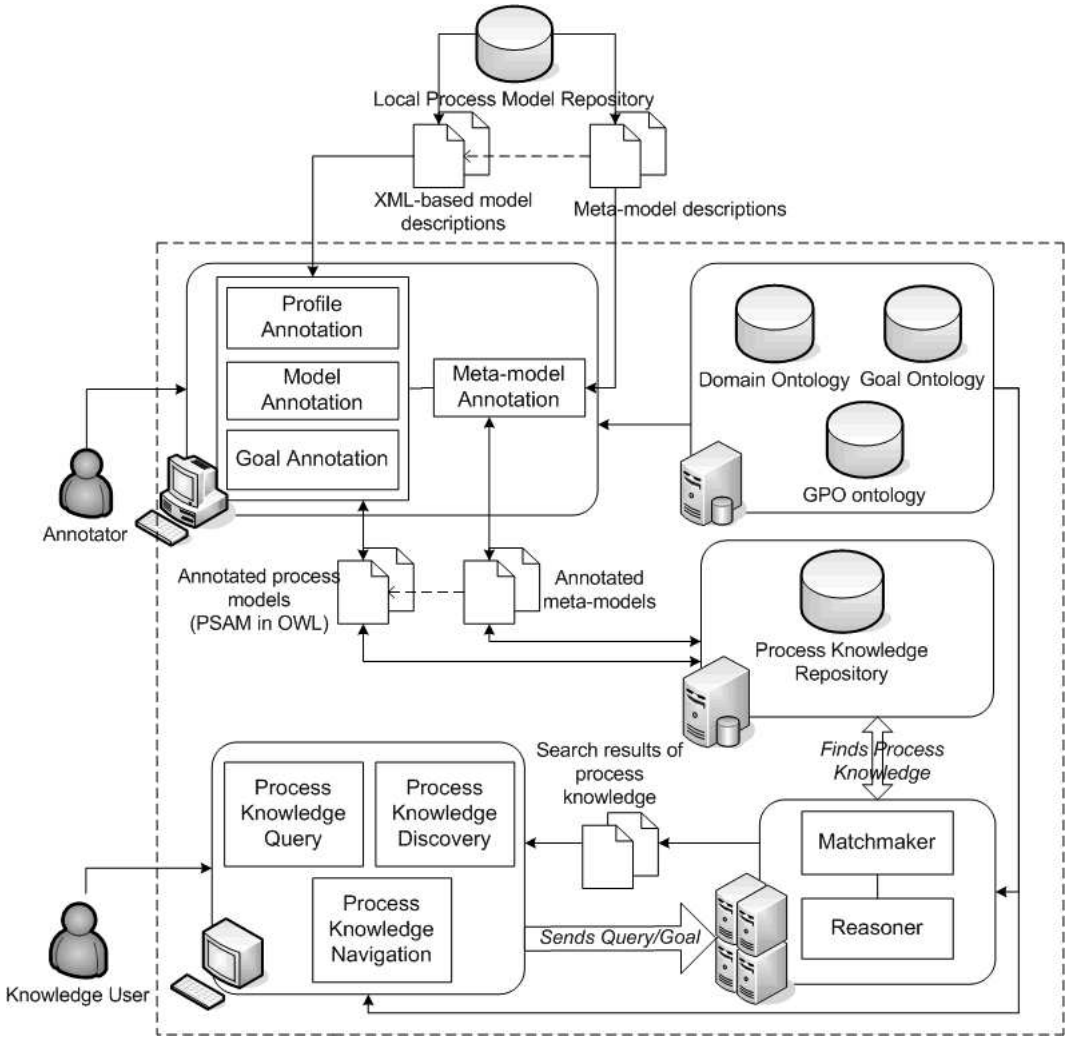


Figure 7.15: Architecture of the process knowledge management system based on semantic annotation

- **Subsumption.** Structure knowledge and computer a taxonomy. (Does $\mathcal{C}$ subsume $\mathcal{D}$ w.r.t. $\mathcal{O}$? i.e. $\mathcal{C}^I \sqsubseteq \mathcal{D}^I$ in every model $\mathcal{I}$ of $\mathcal{O}$?)
- **Equivalence.** Check if two classes denote same set of instances. (Is $\mathcal{C}$ equivalent to $\mathcal{D}$ w.r.t. $\mathcal{O}$? i.e. $\mathcal{C}^I = \mathcal{D}^I$ in every model $\mathcal{I}$ of $\mathcal{O}$?)
- **Instantiation.** Check if individual is an instance of a class. (Is x an instance of $\mathcal{C}$ w.r.t. $\mathcal{O}$? i.e. $x^{\mathcal{I}} \in \mathcal{C}^I$ in every model $\mathcal{I}$ of $\mathcal{O}$?)
- **Retrieval.** Retrieve the set of individuals that instantiate a class and a property.
  - Retrieve the set of i s.t. $i \in \mathcal{C}$ w.r.t.$\mathcal{O}$.
  - Retrieve the set of (x, y) s.t. $(x, y) \in \mathcal{R}$ w.r.t.$\mathcal{O}$.

Semantic reasoning can be useful in the ontology-based querying of process knowledge. The reasoning about the classes and properties of an ontology is often called as

T-Box inference, whilst A-Box inference is associated with instances of the classes [53]. In the process knowledge management system, the knowledge model PSAM is the instance of the GPO ontology and the contents in PSAM are from domain ontology (e.g. the SCOR ontology in the exemplar studies). Therefore, A-Box inference can be employed when retrieving the process model fragments which are represented as the GPO instances in the PSAM model. Meanwhile, T-Box inference can be applied on reasoning the semantic relationships of the ontological concepts between query and domain ontology.

7.5.3 Simple walkthrough examples

We use some annotation results from the exemplar studies to exemplify how the system architecture and semantic reasoning technique are applied in the application of process knowledge management. Three enterprise models have been annotated with GPO and SCOR ontologies. The annotated model fragments represented in PSAM are stored in the repository, which are to be retrieved, discovered and navigated by model users.

In one example, the model user needs to *look for some model fragments about the process which deal with "Bill" in both enterprise systems.* The following steps must be taken (formalized in (i), (ii), and (iii)):

1. The search question must be translated as a query using PSAM "language" derived from GPO ontology and terminologies from SCOR ontology.

    - The process is defined as ***Activity*** and "Bill" is instance of ***Artifact*** according to the GPO ontology.
    - The term "Bill" in the user's question is same as the concept "Bill" in the SCOR ontology.
    - The query submitted to the system is then interpreted as "Find the ***Activities*** which have ***Artifact*** `Bill`.", which is formalized in (i) with respect to the GPO and SCOR ontology.

2. The system analyzes and executes the query accompanied with reasoner.

    - The reasoner employs the T-Box inference to get all the equivalent concepts and all the sub-Classes of `Bill` in the SCOR ontology, which is formalized in (ii). In this case, `Invoice` is semantically equal to `Bill`, and `Bill of Materials` is a sub-Class of `Bill`.
    - The query is expanded by the three concepts, i.e. "Find the ***Activities*** which have ***Artifact*** `Bill`, `Invoice` or `Bill of Materials`".
    - According to the PSAM annotation, the model instances of ***Artifact*** can have the references of `Bill`, `Invoice`, and `Bill of Materials` through the semantic relationships such as `same_as`, `kind_of` and `part_of`. The query is hereby expanded based on the annotation information, which is formalized in (iii). A-Box inference is applied in (iii) to find out the model fragments (x, y) which are annotated as the instances of ***Activity*** and ***Artifact***.

Query: $?x|(x, Bill) \in \mathcal{R}$: `has_artifact`, $x \in Activity$ w.r.t $\mathcal{O}^{gpo}$, $\mathcal{O}^{scor}$ (i)

T-Box inference: $?\mathcal{C}|\mathcal{C} = Bill$ w.r.t. SCOR ontology $\mathcal{O}^{scor}$
$?\mathcal{C}|\mathcal{C} \sqsubseteq Bill$ w.r.t. $\mathcal{O}^{scor}$ (ii)

A-Box inference: $?y|(x, y) \in \mathcal{R}$: `has_artifact`, $y \in Artifact$ w.r.t. $\mathcal{O}^{gpo}$
$(y, C) \in \mathcal{R}$: `same_as`, `kind_of`, `part_of` w.r.t. $\mathcal{O}^{gpo}$ (iii)

After running query and inference tasks, the system finds the following model fragments which are about the process dealing with "Bill":

1. PM_A: "Check relevant bills". In the original model of PM_A, this Activity has an Artifact "bill" which is annotated with the SCOR ontology concept "Bill".

2. PM_{B2} "Send items". In the original model of PM_{B2}, this Activity has an Artifact "invoice" which is annotated with the SCOR ontology concept "Invoice" (an equivalent concept of "Bill" in the SCOR ontology).

Another example is about a goal query of process knowledge. The model user would like to *search the process model fragments having the impact on the goal of improving delivery performance*. Similar steps are undertaken formalized in (j), (jj), and (jjj):

1. The search question is translated as a query using PSAM, GPO and SCOR terminology.

   - Goal is only used to annotate ***Activity*** in PSAM, the search target is hence ***Activity*** of PSAM in the repository.
   - `Improve Delivery Performance` is a soft goal defined in the SCOR ontology, which is chosen as the keyword for the query.
   - An ***Activity*** in PSAM is annotated with a soft goal through the relationship `positively_satisfies` and `negatively_satisfies`. Therefore, the query submitted to the system is then interpreted as "Find any ***Activity*** which `positively_satisfies` or `negatively_satisfies` the goal `Improve Delivery Performance`". The query is formalized in (j).

2. The system analyzes and executes the query accompanied with a reasoner.

   - A reasoner employs the T-Box inference to get all the equivalent concepts, all the sub-Classes and all the composition members (`has_parts`) of `Improve Delivery Performance`, which is formalized in (jj):
     - The sub-Class inference results in `Improve Deliver to Customer Performance` and `Improve Supplier Delivery Performance`.
     - `Process Invoice Without Error`, `Ensure Full Delivery` and `Decrease Percentage of Defective Supplied` are the part-goals of `Improve Deliver to Customer Performance`.

  - Furthermore, through the transition of the inference, the composition members of `Improve Deliver to Customer Performance` (i.e. `Improve Deliver to Customer On Time Delivery Performance`, `Improve Deliver to Customer Delivery to Date Performance`) and of `Improve Supplier Delivery Performance` (i.e. `Improve Supplier On Time Delivery Performance`, `Ensure Supplier Delivery to Date Performance`) should also be involved in the query.

- The query is expanded by those inferred concepts, i.e. "Find the ***Activities*** which `positively_satisfies` or `negatively_satisfies` the goals `Improve Delivery Performance`, `Process Invoice Without Error`, `Ensure Full Delivery`, `Decrease Percentage of Defective Supplied`, `Improve Deliver to Customer Performance`, `Improve Supplier Delivery Performance`, `Improve Deliver to Customer On Time Delivery Performance`, `Improve Deliver to Customer Delivery to Date Performance`, `Improve Supplier On Time Delivery Performance`, and `Ensure Supplier Delivery to Date Performance`."
- When executing the query formalized in (jjj), C will be replaced by all the inferred concepts and A-Box inference is applied to locate the instance of ***Activity*** (x).

Query: $?x|(x, ImproveDeliveryPerformance) \in \mathcal{R}$: `positively_satisfies`, `negatively_satisfies`, $x \in Activity$ w.r.t $\mathcal{O}^{gpo}$, $\mathcal{O}^{scor}$ (j)

T-Box inference: $?\mathcal{C}|\mathcal{C} = ImproveDeliveryPerformance$ w.r.t. $\mathcal{O}^{scor}$
$?\mathcal{C}|\mathcal{C} \sqsubseteq ImproveDeliveryPerformance$ w.r.t. $\mathcal{O}^{scor}$
$?\mathcal{C}|ImproveDeliveryPerformance$ `has_parts` $\mathcal{C}$ w.r.t. $\mathcal{O}^{scor}$ (jj)

A-Box inference: $(x, C) \in \mathcal{R}$: `positively_satisfies`, `negatively_satisfies` w.r.t. $\mathcal{O}^{gpo}$ (jjj)

After running the query and inference, the system finds the following model fragments which might have an impact on the goal of improving delivery performance:

1. PM_A: "Check delivery items" (*positively_satisfies* `Improve Delivery Performance`); "Correct the delivery quantity" (*positively_satisfies* `Improve Deliver to Customer Performance`); "Check availability of delivery items" (*positively_satisfies* `Ensure Full Delivery`); "Monitor delivery date" (*positively_satisfies* `Improve Deliver to Customer On Time Delivery Performance` and `Improve Deliver to Customer Delivery to Date Performance`); "Edit partial delivery information" (*negatively_satisfies* `Ensure Full Delivery`).

2. PM_{B1}: "Receive items from local suppliers" (*positively_satisfies* `Ensure Supplier Delivery to Date Performance` and `Improve Supplier On Time Delivery Performance`); "Issue the deficit protocol" (*negatively_satisfies*

`Improve Supplier On Time Delivery Performance`); "Issue an export Invoice" (*negatively_satisfies* `Process Invoice Without Error`).

3. PM_{B2}: "Correct orders" (*positively_satisfies* `Ensure Full Delivery`); "Issue Invoice" (*negatively_satisfies* `Process Invoice Without Error`).

7.6 Summary

In this chapter, we have exemplified the semantic annotations— profile annotation, meta-model annotation, model annotation and goal annotation through two exemplar studies. A BPMN model and two EEML models have been selected from two different business organizations in a same business domain about logistics processing. The representations of two models have shown the diversities of modeling notations, conceptualizations, terminologies, and business processing ways. GPO as the semantic mediator for modeling languages, has been mapped to BPMN and EEML in the meta-model annotation phase. Two models have been transformed into the PSAM models with Pro-SEAT. The SCOR reference model provides the process templates and standards of logistics process, and it has been chosen as domain ontology for the model annotation and the goal annotation. The level 3 process elements with inputs and outputs from SCOR have been formalized as SCOR domain ontologies in OWL. In order to facilitate the semantic reference from the PSAM model to the domain ontology, the SCOR concepts have been organized in Activity, Artifact and Actor-role categories. The goal ontology has been derived from the process elements, the inputs and outputs, and the metrics of performance attributes of the SCOR specifications. The semi-automatic goal annotation has been conducted by Pro-SEAT based on the goal annotation algorithms.

To illustrate the applicability of the semantic annotation approach, we have outlined a process knowledge management system integrating the semantic annotation components. Making use of the Semantic Web technology, the semantic reasoning such as T-Box and A-Box inference has also been exemplified on the semantic annotation results for the process knowledge management purpose. In next chapters we will conduct more systematic evaluation and applicability validation of our method.

Part III

Evaluation

Chapter 8

Quality Evaluation of the Method

Evaluation of the method and prototype implementation consists of two parts: 1) quality analysis of semantic annotation framework and method, and 2) applicability validation based on annotation results. In this chapter, we focus on the quality of our method. We apply a quality framework — SEQUAL to provide a systematic analysis on the quality of GPO (General Process Ontology), PSAM (Process Semantic Annotation Model) and the annotation tool Pro-SEAT. The quality of the work is evaluated based on the use experience of the exemplar studies, which is also related to the applicability validation of the work in Chapter 9.

8.1 Evaluation Design

Since the underlying theory of our work is information modeling, we look at the criteria and metrics of the evaluation in the information modeling discipline. GPO and PSAM have been created for the knowledge representation of process models. If the PSAM definition in chapters 4 and 5 is regarded as a modeling language, GPO is the meta-model of it, and an instance of PSAM is a model. In our exemplar studies, the original EEML/BPMN process models are translated into the PSAM models in a common process knowledge modeling language based on GPO. We therefore apply a general quality framework of models and modeling languages in the evaluation. The quality framework is based on the semiotic theory[1], which is also the theoretical basis of our annotation framework. Hence the whole work is built and evaluated under the same theoretical foundation. The quality categories are selected from the quality framework and evaluated through the exemplar studies. The annotation tool is also evaluated according to the quality framework. However, since it is only a prototype, the performance of the system and the user interface is not taken into account in the evaluation.

8.2 Settings for the Quality Evaluation

In this section we discuss a general quality framework — SEQUAL [80], which is used for this evaluation work. A set of facts from the annotation approach and exemplar

[1]The quality categories in the quality framework correspond to the semiotic ladder [33].

studies are identified and explained to correspond to the quality categories which are defined in the quality framework.

8.2.1 SEQUAL

SEQUAL (SEmiotic QUALity framework) is the latest version of the general quality framework of models and modeling languages which was earlier reported in [73], [76], and [78]. This quality framework was developed with the objective of providing a systematic structure for evaluating the desirable properties for conceptual models. It has been useful in wider context of understanding quality in areas such as requirements engineering [74], enterprise modeling [78] [115], interactive modeling [77], and evaluating and comparing modeling languages such as UML [75], OWL [91].

Compared with other approaches and frameworks available for evaluating modeling approaches, SEQUAL provides a systematic evaluation method and it has three unique properties:

- It distinguishes between goals and means, i.e. by separating what to achieve from how to achieve it.
- It is closely linked to linguistic and semantic concepts, which particularly including the discussion on the terms of the semiotic theory — syntax, semantics, and pragmatics.
- It is based on a constructivistic world-view, recognizing that models are usually created as part of a dialogue between the participants involved in modeling, whose knowledge of the modeling domain and potentially the domain itself changes as modeling takes place.

Early version distinguished among three *quality categories* for conceptual models (syntactic, semantic, and pragmatic) according to steps on the semiotic ladder [33]. The *quality goals* corresponding to the categories are syntactic correctness, semantic validity and completeness, and comprehension (pragmatic). The framework distinguishes between *goals* and *means to reach the goals*. In the later extensions, more quality categories have been added so that the entire semiotic ladder is included, that is, *physical*, *empirical*, *syntactic*, *semantic*, *perceived semantic*, *pragmatic*, *social* and *organizational quality*. The main concepts of the framework and their relationships for quality of models are illustrated in Figure 8.1.

Physical quality is to check how the externalized model $\boldsymbol{M}$ could be persistent and available enabling the audience to make sense of it. Persistence is usually associated with the protection against loss or damage. It also sometime includes previous versions of the model, if these are relevant. Availability is dependent on its externalization and distributability.

Empirical quality focuses on model $\boldsymbol{M}$ itself. Empirical quality deals with the variety of elements distinguished, error frequencies, and ergonomics for CHI (Computer-Human Interaction) for documentation and modeling tools.

Syntactic quality is the correspondence between the model $\boldsymbol{M}$ and the language extension $\boldsymbol{L}$ of the modeling language. Syntactical correctness is the goal of this quality

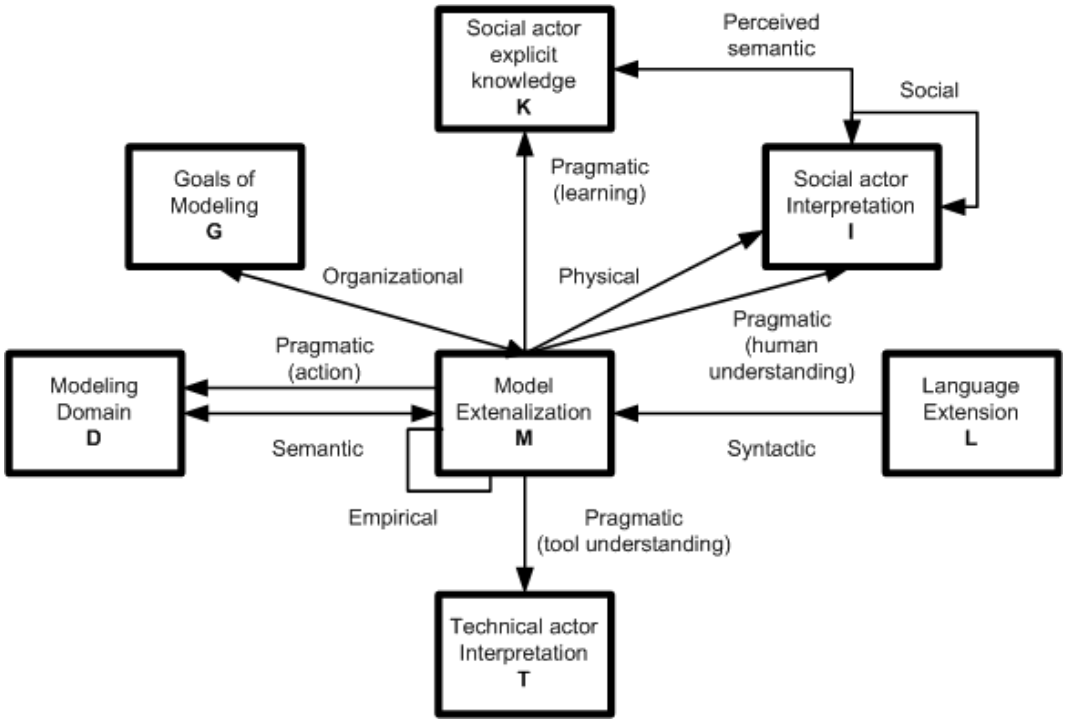


Figure 8.1: SEQUAL framework for discussing quality of models [80]

category, which requires that all statements in the model conform to the syntax and vocabulary of the language.

Semantic quality is the correspondence between the model ***M*** and the modeling domain ***D***. The semantic validity and the semantic completeness are goals for this quality.

Perceived semantic quality is the correspondence between the social actors' interpretation ***I*** of a model ***M*** and their current knowledge ***K*** of the domain ***D***.

Pragmatic quality is the correspondence between the model ***M*** and the audiences' interpretation of it ***I***. Usually **social pragmatic quality** (to what extent people understand the models) and **technical pragmatic quality** (to what extend tools can be made that can interpret the models) are differentiated. In addition, the pragmatic quality includes to what extent the participants are able to change the domain after interpreting the model or learning from it. The goal for pragmatic quality is comprehension, i.e. the degree to which a model can be understood.

Social quality focuses on social actors' interpretations ***I***. The goal of social quality is agreement among their interpretations.

Organizational quality is the correspondence between the model ***M*** and the modeling goals ***G***. Organizational goal validity and organizational goal completeness are related to this quality.

The quality framework of modeling languages keeps the same concepts used in the quality framework of models. However, the core concept is language extension ***L*** but not model externalization ***M*** anymore. Six quality categories of language quality are

identified as follows.

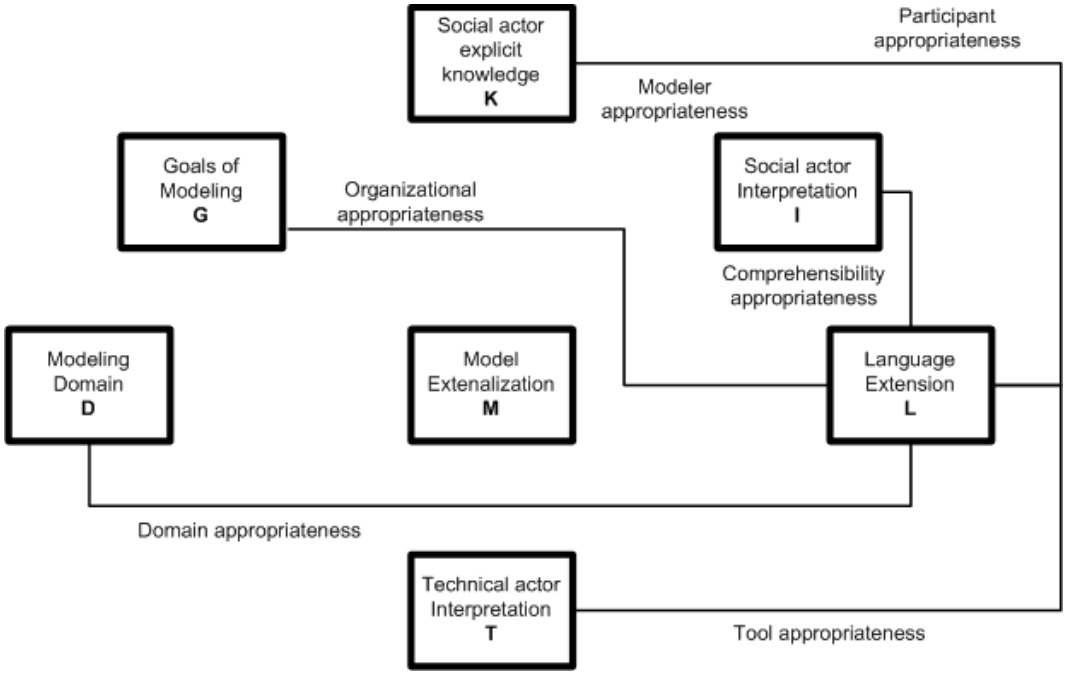


Figure 8.2: Language quality in the quality framework [80]

- **Domain appropriateness** indicates whether the modeling language addresses the problems of eliciting/representing relevant facts of the problem domain. Domain appropriateness is primarily a means to achieve semantic quality.

- **Participant appropriateness** indicates whether the modeling language corresponds to what the participants perceive as a natural way of working, i.e. the modeling language they have known. This is primarily a means to achieve pragmatic quality.

- **Modeler appropriateness** indicates whether the modeling language assists the modelers in externalizing their knowledge. Modeler appropriateness is primarily a means to achieve semantic quality.

- **Comprehensibility appropriateness** indicates whether the social actors are able to comprehend the models made in the modeling language. Comprehensibility appropriateness is primarily a means to achieve empirical and pragmatic quality.

- **Tool appropriateness** indicates whether the modeling language lends themselves to automated tool support or assists in support for reasoning. Tool appropriateness could be means to achieve syntactic, semantic, and pragmatic quality through formal syntax, mathematical semantics, and operational semantics, respectively.

- **Organizational appropriateness** indicates whether the model made in the modeling language achieves the organization's goals. This is the means to support organizational quality.

Figure 8.2 indicates how these appropriatenesses are related to the concepts in the quality framework. Those categories are related both to the meta-model and to the notation.

8.2.2 The facts corresponding to the quality categories from the exemplar studies

In order to apply the general quality framework, we firstly specify the facts corresponding to the sets in the quality framework. The quality categories relate to the relationships between the following facts.

Annotation model

In this scenario, the annotation process and the annotation result are evaluation targets. In our approach, the annotation result is an instance of the annotation model. Therefore, the PSAM model is the externalized model in the quality framework.

Goals of annotation

The annotation (the PSAM modeling in our approach) is to represent the knowledge stored in the existing process models through a set of agreed semantically-defined concepts and formats. Therefore, the goals of annotation depend on the original modeling goals in each case and also depend on the goals of knowledge management. A number of goals are identified from the cases as follows.

- G1 - The annotation should improve the readability of the existing process models.
- G2 - The annotation should help sharing process knowledge among different organizations within a domain.
- G3 - The annotation should help to analyze and validate the existing process models.
- G4 - The annotation should be helpful in the semantic reconciliation of models and to facilitate reuse and integration of models.

Modeling domain

In general, the modeling domain is about processes in the enterprise modeling domain. In this case, it is the SCO (Supply-Chain-Operation) domain.

Language extension

GPO is the meta-model of PSAM and determines the definitions of PSAM. Thus, GPO defines the syntax of the annotation model. Since GPO is created in OWL, a PSAM model is the instance of the OWL model and it has the syntactical features and constraints of OWL.

Modeler — model annotator

In this scenario, model annotators are modelers. They create the annotation by applying their modeling and domain knowledge. In the exemplar studies, we assume that the annotators of PM_A know the modeling language BPMN and their BPMN models quite well. The similar assumption for the annotators of PM_{B1} and PM_{B2} is the knowledge of the EEML modeling language and EEML models. Moreover, they understand thoroughly their business process and domain definitions. Such a role corresponds to the modeler in the quality framework.

Participant actor — annotation user

Annotation users are the consumers of the annotation results. In the cases, they make use of the annotation information in the process knowledge management activities, such as querying information, analyzing models, and eliciting/inferring interested knowledge. They correspond to the participant actors in the quality framework.

Annotation tool

The annotation tool is used to support the annotation procedure. The annotation tool — Pro-SEAT provides the functions for profile annotation, meta-model annotation, model annotation and goal annotation.

8.3 Quality Analysis

The quality evaluation includes the quality of GPO, the PSAM definitions, the annotated PSAM model instances and the annotation tool for the cases. GPO is defined as the meta-model of the annotation model, and the PSAM definitions are applied as the notation of the modeling. Thus, the language quality is analyzed on GPO and the PSAM definitions. The model quality is discussed based on the instances in the exemplar studies. The quality categories of SEQUAL are also applied to evaluate to which extent it improves the model quality.

8.3.1 GPO and PSAM quality evaluation

1. *Domain appropriateness.* GPO has been defined by referring most process modeling languages in enterprise modeling, and GPO includes the most vital and frequently used concepts in describing a business process. Compared with those languages, the GPO concepts are more general in the enterprise process domain. Accordingly, specific semantics in the domain can only be abstracted or encapsulated into a set in GPO. Such a quality evaluation can be made through the analysis of the meta-model annotation in the exemplars. For example, in EEML the *resources* could be the following different concepts — *Organization, Person, Information Object, Skill, Software Tool, Manual Tool*, and *Material Tool*. And the *resource role* is played by those *resources*. In the meta-model annotation, only two concepts ***Actor-role*** and ***Artifact*** in GPO are found to correspond

to those resources. It is obvious that more specified semantics from the organizational perspective will be missing when representing the EEML model by the GPO concepts. When annotating BPMN with GPO, an exact semantic mapping between *Data Object* and ***Actor-role***, between *Swimlane* and ***Artifact*** could be made without losing the semantics of the original BPMN models. With respect to the concepts of ***Activity***, ***Input***, ***Output*** and ***WorkflowPattern*** in GPO, it is relatively easy to map to both EEML and BPMN concepts. Thus, GPO is mainly designed from the process and function perspectives. The main concepts defined in SCOR are process elements, inputs and outputs. GPO provides enough concepts to represent such a domain. However, the relationships defined in EEML and BPMN cannot be directly mapped to the GPO concepts. Relationships between the GPO concepts are not applied during the meta-model annotation phase but directly generated in the PSAM model. Moreover, the annotation model is not only a process model but should support the ontology-based annotation. The relations linking the GPO concepts and the ontology concepts are specified in a PSAM model as well. All in all, GPO and PSAM have proper domain appropriateness.

2. *Participant appropriateness.* As we have discussed perviously, GPO is relatively simple compared with most enterprise process modeling languages. Since we assume that the annotators are process modeling specialists, learning GPO should not be a problem. In the profile, model and the goal annotation, we also assume the annotators are domain experts and understand the original models quite well. The only request for them is to have some knowledge about ontology and semantic relationships when employing the ontology-based annotation.

3. *Modeler appropriateness.* Since the GPO concepts are more general than a particular enterprise process modeling language, some specific semantics in the original models could not be conveyed in the PSAM model. However, as we argued before, GPO is not initially created as a comprehensive process modeling language. The intention of the proposal is to elicit the most important and useful process knowledge and to represent it in annotation models for knowledge management. In the meta-model annotation phase, a possible mapping from GPO to a particular modeling language has only three types — *one-to-one*, *one-to-many*, and *one-to-combination*, which means that the modeling language elements can be mapped to only one GPO concept. In the current prototype implementation many-to-one mappings are not supported. After the transformation based on the meta-model annotation, all the PSAM models have the same structure. Therefore the modeler can not make any other creative structure.

4. *Comprehensibility appropriateness.* Only 29 concepts are identified in the current version of GPO and the PSAM specification. The concepts and the relationships between them are relatively straightforward for annotation users to interpret just by reading the names. With respect to the relations linking the GPO concepts and the ontology concepts, only three semantic relationship categories (synonym, hypernym, and meronym) from PSAM are chosen for the exemplar studies. Also the name of those semantic relationships are specified according to the GPO

concepts. For example, the meronym is represented by "phase_of" for ***Activity***, "part_of" for ***Artifact*** and "member_of" for ***Actor-role***.

5. *Tool appropriateness.* PSAM is defined using a formal syntax and modeled in OWL. OWL is XML-syntactic compilable and it can be parsed by available commercial or non-profit parsers. The semantics of GPO and PSAM are also formally defined according to the OWL semantic definitions. The GPO concepts are modeled as `owl:Class` and the relations are modeled as `owl:ObjectProperty` and `owl:DatatypeProperty`. For our work, OWL DL is the basis of the annotation models. OWL DL was designed to support the existing Description Logic business segment and has desirable computational properties for reasoning systems [195]. The Protégé-OWL API is integrated in the annotation tool so that the syntax and the semantics of GPO and PSAM can be interpreted by the tool.

6. *Organizational appropriateness.* Due to the fair comprehensibility appropriateness and the application goals of this work, GPO and PSAM are theoretically easy to understand for audiences from different enterprise units. However, further empirical evidence is required to establish this claim.

8.3.2 Quality analysis of the annotation model instances

A specific annotation model is an instance of the PSAM model that is transformed from the original process models and annotated with ontological concepts. The quality of GPO and PSAM will impact the quality of the PSAM models. The evaluation of the PSAM models concludes how the model quality relates to the usability of the annotation results.

Physical Quality

We first look at how the knowledge of the domain has been externalized by the annotation models. Such a goal could be achieved through the means of domain appropriateness, participant language knowledge appropriateness and knowledge externalizability appropriateness. As we have discussed in the quality analysis of those appropriateness above, the PSAM models can present most information about the process and functional perspective. The EEML and BPMN models have presented the logistics processing comprehensively due to their expressiveness. Based on the meta-model annotation results, the original models are transformed into the PSAM models. Ideally, the transformation should keep exactly the same information as the original models. It is obvious that more one-to-one mappings in the meta-model annotation, more knowledge represented in the modeling elements can be preserved after the transformation. In the exemplar studies, five one-to-one mappings are in the annotation of BPMN and three are in the annotation of EEML. In EEML, different resources are specified and they all could be mapped to the GPO concepts Artifact and Actor-role respectively. Thus, the PSAM models of PM_A should have better physical quality than the ones of PM_B. Fortunately, those specific resources are not much applied in the original EEML models PM_B. That is, not much domain knowledge is lost because of the meta-model

annotation in this case. However, annotating the relationships defined in the modeling languages is not supported by this approach. The relationships between the GPO concepts are defined in GPO so that knowledge in the PSAM models are interpreted according to the GPO definitions. Compared with the original models, the PSAM models have additional knowledge — the ontological knowledge, which is introduced during the model annotation and the goal annotation. In the exemplar studies, the domain ontology is related to the SCOR standard, while the goal ontology provides the knowledge about the objectives of process and also the links to the SCOR process metrics. In chapter 9, we discuss what useful knowledge can be externalized by the PSAM models.

The PSAM models are then checked to determine if they are easily available and maintainable. The meta-model annotation result for each modeling language is stored in an XML file. Such results can be reused in generating the PSAM models by different models in the same modeling language. The generated PSAM models are OWL models, so that the PSAM models can be read and edited by any OWL editing tools. Model annotation and goal annotation is then made to the PSAM models. However, if there is any change of the meta-model annotation, the PSAM models based on the meta-model annotation result have to be re-generated. That means all the model and the goal annotation work made on the PSAM models will be lost.

Empirical Quality

There is no graphic notation for the annotation model. All the generated PSAM models are textual OWL files so that the readability of the model is poor without the tool support. The PSAM models are categories according to the GPO concepts. Since no graphic notation is used for the GPO concepts, the concepts such as "Activity", "Artifact", "Actor-role" are not distinct to the user. There is a limited number of categories since only eight GPO concepts are used in the exemplar studies. However, when the original model is large[2], the list of the instances for each GPO concept is long. For example, 33 instances of Activity are listed for the annotation in PM_A and only 14 instances of Activity in PM_{B2}. From the experience of using Pro-SEAT, it turns out that it is more difficult to find a desired instance in the PSAM model of PM_A compared with the one of PM_{B2}. The instance is named with the model textual title and its model id from the original process models. When two instances have the same model title but different model id, mistakes will be made by picking the wrong instance because of a confusion with the title.

Syntactic Quality

As an OWL model with classes and instances, the PSAM models should comply the OWL syntax. Since the PSAM models are generated from the meta-model annotation result, the premise is that the syntax of GPO and the mappings have been validated. GPO is created by Protégé and the correctness of the syntax is checked by Protégé. The meta-model mapping rules have also been set to comply with OWL syntax.

[2]The size of the model depends on the number of the model elements, i.e. the instances in the PSAM model.

Semantic Quality and Perceived Semantic Quality

The semantic quality of the annotation models depends on the semantic quality of both the original models and the annotation. We have assumed that the original process models are semantically correct and complete. The semantics of the generated PSAM models are consequently determined by the transformation from the original models to the PSAM definitions.

More semantics are introduced during the model and the goal annotation. The quality of such semantics is categorized into the perceived semantic quality in this approach because it corresponds to annotators' and annotation users' interpretations and their knowledge of the domain. Most annotation operations are manual, but Pro-SEAT supports the semi-automatic goal annotation which might help to achieve semantic validity and completeness. In Pro-SEAT, the ontology-based query interface provides the service to perceive the semantics of the annotation. The perceived semantic quality of annotation model is further validated through the applications and analyzed in chapter 9.

Pragmatic Quality

Since the annotated process models are OWL model instances, it is not difficult for an annotation user to understand the annotation schema and structure. Moreover, ontology is designed in a human understandable way and the annotation user is supposed to know about the domain. There is no problem for the annotation user to read the models, but it is hard for the user to see the whole picture of the models without the support of a visualization tool. However, the OWL models can be interpreted by any tools supporting OWL DL. Since the SCOR ontology provides explicit representation of conceptualization of supply chain domain, the references in the annotation help annotation users learn the domain and adapt processes of PM_A, PM_{B1} and PM_{B2}, which is evident in the applicability validation in Chapter 9. Besides, the pragmatic quality of PM_A, PM_{B1} and PM_{B2} represented in Pro-SEAT also depends on the pragmatic quality of Pro-SEAT, which is discussed in the following section.

Social Quality

One of the goals for the proposed approach is to help process knowledge sharing among different organizations within a domain. The ontologies are assumed to be the domain standards which are agreed by different audience. GPO is the meta-model ontology. The PSAM models are generated from this standard and the model content is annotated with domain standard. In the exemplar studies, SCOR is chosen to be the common standard and modeled as the domain ontology, because SCOR is already well known in many enterprises for the supply-chain management in practice. The original models PM_A, PM_{B1} and PM_{B2} are all within the supply-chain domain.

Organizational Quality

For the organizational quality of the PSAM models and the tool, we can check if the modeling goals can be fulfilled and addressed by the proposed approach.

- G1 - The annotation should improve the readability and comprehensibility of the existing process models.

  The semantic annotation enriches the model semantics by referencing the ontology concepts using semantic relationships. Thus, with the referenced ontology, the semantics of the model elements can be interpreted more correctly and completely by both human and machine. With respect to the pragmatic quality of Pro-SEAT we find that the annotation functions do not really address this goal. However the ontology-based query in Pro-SEAT provides the functions to navigate the process models. Since the annotation results are in OWL, the machine can read and reason the semantics in the annotation model. This goal is further analyzed in Chapter 9.

- G2 - The annotation should help process knowledge sharing among different organizations within a domain.

  From the discussion on the social quality, we find that this goal has been addressed by the annotation models. The applicability of the approach in Chapter 9 proves that the annotation models can fulfill this goal.

- G3 - The annotation should help to analyze and validate the existing process models.

  Satisfaction of this goal is checked and discussed in Chapter 9.

- G4 - The annotation should be helpful in model reuse and model integration.

  Such goal is related to G2. The annotation models have not been applied in any real applications of model transformation and model integration. The applicability of the annotation results are only described in the application scenario under the exemplar studies. This goal is further analyzed in Chapter 9.

8.3.3 Quality Evaluation of Pro-SEAT

The annotation tool is also the means to achieve quality annotation results. Thus the tool evaluation is to validate the performance of Pro-SEAT on achieving model quality.

- Physical Quality - Pro-SEAT provides the functions of importing the original process models, loading the ontology, editing and saving the meta-model annotation, the model and the goal annotation results. The current version of the tool is mainly designed for interpreting the models in XML created by Metis tool. The tool can only load the ontology in OWL format and the integrated ontology API is the Protégé 3.2.1 OWL API. There is no database repository deployed for the annotated process knowledge, but only the XML or OWL files and folders are used to manage the knowledge. Therefore, the benefit is that the knowledge is easy to distribute and could be published on the Web, while the disadvantage is the lack of a systematic way to manage the files and their inherent links. This might be compensated by integrating the CO_2SY system which is developed by Strašunskas [175] for managing the interrelationships between product fragments.

- Empirical Quality - The details of model annotation and goal annotation are displayed in the property fields for each instance in Pro-SEAT. The layout of those properties are not well organized in groups and the sequence of the properties is displayed variously each time when running the tool. Thus, it is hard to navigate the properties when manipulating the annotation. Nevertheless, the operation of the annotation is simple — just select the reference concept from the ontology tree by clicking or entering the reference value.

- Syntactic Quality - The prototype of the annotation tool does not provide the functions for performing a syntax check from the user interface, but invalid OWL models can not be parsed by the OWL API in Pro-SEAT.

- Semantic Quality and Perceived Semantic Quality. The current annotation tool does not support any semantic consistency or completeness checking during the annotation.

- Pragmatic Quality - Pro-SEAT does not provide a visualization of process models, annotation models and ontologies. The imported original process model from Metis is listed in the tree-view. The tree-view of the model keeps the same structure as in the Metis tool. It is not difficult for the Metis user to interpret the model in Pro-SEAT. The Protégé's tree-view of ontology is also integrated in Pro-SEAT so that the subsumption relationship between ontology concepts can be viewed in Pro-SEAT. The other relationships represented in OWL properties and constraints can not be displayed in Pro-SEAT, which would hide the complexity and help the annotator to identify the concepts easily. The annotation models are listed and navigated according to the GPO concepts. In Pro-SEAT, it is difficult to establish an overall view of the annotation model or check the inter-relationships between annotations. A model search interface is integrated in Pro-SEAT. Based on the annotation results, it facilitates the navigation of the process models and the annotation information by focusing on the GPO categories.

8.4 Requirements Satisfaction of the Semantic Annotation System

Recall the requirements we identified for the semantic annotation system in section 3.5. Next, we examine what and how the requirements for Pro-SEAT have been met in this section.

1. The system should be able to present and parse annotation source models which are originally in certain formats and representations.
   Pro-SEAT can present and parse the process models in XML format which is generated from Metis modeling tool.

2. The system should provide an ontology browser for the overview and manipulation of ontological knowledge.
   The integrated Protégé-OWL API provides the functionality to browse ontologies.

3. Semantic annotation schema or metadata should be supplied (pre-defined) or generated (ontology-based) by the system.
   OWL formated GPO ontology can be loaded in Pro-SEAT. After the meta-model annotation, the PSAM schema for model and goal annotation can be generated in Pro-SEAT.

4. The annotation procedure should be easily manipulated by users with the system, i.e. easy to locate "annotatee" (e.g. entity in source model) and "annotater" (e.g. concept in ontology) during the annotation.
   This requirement has been discussed through the analysis of empirical quality and pragmatic quality in section 8.3. Stronger evidence should be collected from more usability studies involving end-users.

5. The system should support the maintenance of annotation results (e.g. embedded vs. stand-off annotation).
   The annotation is stand-off annotation in this work so that it allows the dynamic changes of the user-specific annotations.

6. Multiple-ontology references (e.g. different levels of ontologies) might be supported in the system.
   So far, a top-level ontology for meta-model annotation and domain ontology for model and goal annotation are supported in this work.

7. Different types of annotations (e.g. instance identification, URI links, and other semantic relationships) might be supported.
   The main supported annotation types supported by Pro-SEAT are URI links and the semantic relationships such as synonym, hypernym and meronym.

8. Semi-automatic or automatic annotation might be considered in the system.
   The implementation of the goal-annotation algorithms contribute the semi-automation of the tool, which partially meets the requirement.

9. The system might be able to serialize the annotation results for reuse in different systems.
   The annotation result from Pro-SEAT is in OWL and it is serialized. Thus, this requirement is met through the test that the annotation result can be represented in Protégé-OWL Editor.

10. It might be possible to conduct semantic inference among ontology-based semantic annotations.
    The satisfaction of the requirement of semantic inference will be exemplified in next chapter.

8.5 Summary

As evaluation method for ontology-based semantic annotation is a new research topic as the emerging semantic annotation approaches applying the Semantic Web techniques. According to the author's knowledge, there is no systematic evaluation methodology for

semantic annotation approaches and tools. Maynard in [102] identified some requirements for ontology-based annotation tools such as expected functionality, interoperability, usability, accessibility, scalability and reusability. Usually, criteria and metrics for performance evaluation, such as precision, recall and F-measure [154], are defined for the evaluation of semi-automatic or automatic semantic annotations by using information extraction techniques. However, the evaluation is mainly for the semantic annotation of textual contents. The model features are certainly not concerned, but they are very important in our case of the semantic annotation of business process models. Moreover, those metrics are not sufficient for ontology-based information extraction, because the distinction between right and wrong is less obvious [102]. We do not apply any information extraction techniques in our system and the current prototype of the annotation tool mainly supports manual annotation.

We have chosen SEQUAL that has been widely and successfully applied in the information modeling area, to evaluate the proposed approach. Also SEQUAL shares the same theoretical foundation with our semantic annotation framework. According to the semiotic quality categories, the quality analysis has been made at both the metamodel level (GPO and the PSAM specifications) and model level (the PSAM model instances and Pro-SEAT), which make up our semantic annotation method.

Chapter 9

Validation of Applicability

In this chapter, we validate the applicability of annotation results derived from the proposed method. The validation is undertaken through an application scenario, where a new IS solution is modeled through selecting and reusing annotated process model fragments. The validation is supposed to check if the annotation method can facilitate process knowledge management, which is the general application objective of this work. Quality evaluation for Chapter 8 is further analyzed based on the discussion of the application results.

9.1 Validation Design

A walkthrough scenario of a process knowledge management application is deployed in this chapter based on the semantic annotation results from the exemplar studies of Chapter 7. In the scenario, the annotated process models (PM_A, PM_{B1} and PM_{B2}) are IS solutions presenting logistics process knowledge of different organizations, and a new IS problem (an integrated sales delivery solution for their cooperative business) is supposed to be dealt with by reusing and integrating some of the solutions which can achieve goals of the new system.

The annotation results which we got from the exemplar studies are the source data consumed by the application scenario. The annotation goals set in section 8.2.2 concretize a general application objective of our annotation method – to facilitate process knowledge management. The validation is therefore to measure the goals. Based on the goals, a set of requirements are derived. The requirements are implemented through a set of SWRL (Semantic Web Rule Language Combining OWL and RuleML) [202] rules and queries. The rules and queries are executed on the annotation results. The validation is thus to check how the returned query results fulfill those requirements in the context of the application scenario. In addition, semantic validity and semantic completeness of the annotation models are analyzed in the validation. The implementation tool of the application is Protégé-OWL with the SWRL editor and the Jess rule engine [1].

Accuracy of mapping between ontologies and models may affect the validation results. Therefore, we assume that the ontology-based annotation is accomplished by

[1] http://herzberg.ca.sandia.gov/jess/

modeling experts and domain specialists, who provide reliable mappings. Accuracy of the annotation is hereby not taken into account in the evaluation.

9.1.1 Application requirements

In this applicational scenario, users need to navigate models for browsing the process knowledge of the legacy IS solutions, search interesting knowledge candidates for building a new solution, select suitable model fragments from the candidates, and reuse the selected model fragments in a knowledge-based integration. We therefore elicit the following requirements to implement the application. The requirements are associated with the annotation goals (G1-G4) identified in section 8.2.2. If the annotation method fulfills the requirements of the application, it satisfies the annotation goals too.

- $RE1$ - *Navigation requirements.* For the purpose of browsing knowledge, navigate process models (PM_A, PM_{B1} and PM_{B2}) (related to G1). Such a requirement can be decomposed into following sub-requirements by specifying the interests of process properties.
  - $RE1.1$ - List Activities and their sub-Activities.
  - $RE1.2$ - List Activities and their preceding or succeeding Activities.
  - $RE1.3$ - List Activities and their Artifacts.
  - $RE1.4$ - List Activities and their Actor-roles.
  - $RE1.5$ - List Activities and their Preconditions or Postconditions.
- $RE2$ - *Search requirements.* For the purpose of locating interesting knowledge candidates, search the corresponding process model fragments across different business models by using the SCOR domain ontology (related to G2). Such a requirement can be decomposed into following sub-requirements by specifying the interests of ontological concepts.
  - $RE2.1$ - Find the model fragments of process models that reference to SCOR Management Process.
  - $RE2.2$ - Find the model fragments of process models that reference to SCOR Input/Output.
  - $RE2.3$ - Find the model fragments of process models that reference to SCOR Organizational.
  - $RE2.4$ - Find the model fragments of process models that reference to SCOR Goal.
- $RE3$ - *Semantics checking requirements.* For the purpose of selecting suitable model fragments, check the semantics of the process models and the annotation results (related to G3). Such a requirement can be decomposed into following sub-requirements by specifying the interests of process semantics defined in PSAM.
  - $RE3.1$ - Check the Artifacts related to the Input/Output of Activities.
  - $RE3.2$ - Check the Activity sequences.

  - *RE*3.3 - Check the Artifacts in sub-Activities.
  - *RE*3.4 - Check the Actor-roles in sub-Activities.
  - *RE*3.5 - Check the goal annotations in sub-Activities.
  - *RE*3.6 - Check the Precondition/Postcondition of Activities.
  - *RE*3.7 - Check the information flow from Outputs to Inputs.

- *RE*4 - *Knowledge discovery requirements*. For the purpose of reusing and integrating model fragments, acquire the implicit knowledge across different models through reasoning on ontological relationships (related to G4). Such a requirement can be decomposed into following sub-requirements by specifying the interests of potential relationships between process properties of different models.
  - *RE*4.1 - Find out semantic relationships between the Activities of different process models.
  - *RE*4.2 - Find out semantic relationships between the Artifacts of different process models.
  - *RE*4.3 - Find out semantic relationships between the Actor-roles of different process models.
  - *RE*4.4 - Find out goal relations between the Activities of different process models.
  - *RE*4.5 - Find out possible integration points among different process models.

9.1.2 SWRL rules and tool

SWRL is an acronym for Semantic Web Rule Language. As stated in [149]:

> It is intended to be the rule language of the Semantic Web. SWRL is based on a combination of the OWL DL and OWL Lite sub-languages of the OWL Web Ontology Language. It allows users to write Horn-like rules to reason about OWL individuals and to infer new knowledge about those individuals. These rules are expressed in terms of OWL concepts. SWRL is more expressive that OWL DL alone yet retains its formal semantics. It does so, however, at the expense of decidability.

We use SWRL to formalize the application requirements for each annotation model – a PSAM instance model in OWL. Running the SWRL rules shows the computational capability of the annotation models, which is one of the benefits of the proposed approach. Analyzing the rule-execution results helps check what knowledge can be derived from the annotation results and how the results fulfill the application requirements.

SWRL

SWRL rules are of the form of an implication between an antecedent (body) and consequent (head). The intended meaning can be read as: whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold. SWRL rules are written in terms of OWL classes, properties and individuals.

Both the antecedent and consequent consist of zero or more atoms. Atoms in these rules can be of the form C(x), P(x,y), sameAs(x,y) or differentFrom(x,y), where C is an OWL description, P is an OWL property, and x,y are either variables, OWL individuals or OWL data values. A "human readable" form of a SWRL rule is:

antecedent → *consequent*

where both antecedent and consequent are conjunctions of atoms written $a_1 \wedge ... \wedge a_n$. Variables are indicated using the standard convention of prefixing them with a question mark (e.g. ?x) [202].

SWRLTab and SQWRLQueryTab

The SWRLTab [151] is a development environment for working with SWRL rules in Protégé-OWL. It supports the editing and execution of SWRL rules. It also provides the mechanism to turn SWRL into a query language – SQWRL (**S**emantic **Q**uery-Enhanded **W**eb **R**ule **L**anguage). The SQWRLQueryTab [150] is a plug-in to the Protégé-OWL SWRLTab and it provides a convenient way to visualize the results of queries on an OWL ontology. In the evaluation, we employ SWRLTab to formalize the requirements from section 9.1.1 into SWRL rules in order to validate the annotation results. Most of the rules are queries which can return the query results in SQWRLQueryTab. The SQWRL provides SQL-like operations to retrieve knowledge from an OWL ontology.

9.2 Application Requirements in SWRL Formulation

In the exemplar studies, the BPMN model (PM_A) and the EEML models (PM_{B1} and PM_{B2}) are annotated by the proposed approaches. Those three models are to be shared with each other and integrated in a process knowledge management application. The application requirements are formalized into a set of SWRL queries and rules which are executed on those annotated models. The queries and rules are generally defined for all models, but they can be also specified by replacing the variables with certain values for specific models.

9.2.1 Formalize RE1 - Navigation requirements

RE1 is related to G1 (*The annotation should improve the readability and comprehensibility of the existing process models*). The sub-requirements identify what knowledge of the process models should be read directly from the annotation results. The requirements in RE1 are formalized in Table 9.1.

9.2.2 Formalize RE2 - Search requirements

RE2 is related to G2 (*The annotation should help sharing process knowledge among different organizations within a domain*). The search should look up all the three models without problems associated with semantic heterogeneity. A domain ontology which acts as the common understanding of the domain is essential in supporting the machine

Table 9.1: SWRL queries and rules for RE1

RE1	Rule Name	SWRL formulation
RE1.1	**QRule-Activity-subActivity**	$Activity(?x) \wedge has\_subActivity(?x, ?y) \rightarrow query : select(?x, ?y) \wedge query : orderBy(?x)$
RE1.2	**QRule-Activity-hasPrecedingActivities**	$Activity(?x) \wedge has\_precedingActivities(?x, ?y) \rightarrow query : select(?x, ?y) \wedge query : orderBy(?x)$
	QRule-Activity-hasSucceedingActivities	$Activity(?x) \wedge has\_succeedingActivities(?x, ?y) \rightarrow qurey : select(?x, ?y) \wedge query : orderBy(?x)$
RE1.3	**QRule-Activity-hasArtifact**	$Activity(?x) \wedge has\_Artifact(?x, ?y) \rightarrow query : select(?x, ?y) \wedge query : orderBy(?x)$
RE1.4	**QRule-Activity-hasActor**	$Activity(?x) \wedge has\_Actor - role(?x, ?y) \rightarrow query : select(?x, ?y) \wedge query : orderBy(?x)$
RE1.5	**QRule-Activity-hasPrecondition**	$Activity(?x) \wedge has\_Precondition(?x, ?y) \rightarrow query : select(?x, ?y) \wedge query : orderBy(?y)$
	QRule-Activity-hasPostcondition	$Activity(?x) \wedge has\_Postcondition(?x, ?y) \rightarrow query : select(?x, ?y) \wedge query : orderBy(?y)$

to understand the semantically-reconciled process knowledge from different organizations. In the PSAM models, an ontology is referenced by models through annotation relationships which are represented using OWL properties such as `same_as`, `kind_of`, `part_of`, `mapped_to`, `achieves`, `positively_satisfies` and `negatively_satisfies`. Therefore, those properties should be specified in the SWRL queries or rules (see Table 9.2).

9.2.3 Formalize RE3 - Semantic check requirements

RE3 is related to G3 (*The annotation should help to analyze and validate the existing process models*). It is used to infer the underlying process knowledge according to the annotation and to check the semantic completeness and semantic validity of annotation models. The SWRL formulations of RE3 are listed in Table 9.3. The queries for inferences are usually built through correlations to OWL properties. For example, besides the Object Property `has_Artifact` of Activity, the relationship between Activity and Artifact can also be inferred through connecting relations among Activity, Input, Output, and Artifact. RE3.1 (*Check the Artifacts related to the Input/Output of Activities*) is therefore formalized as **IRule-Activity-Input-hasArtifact**, **IRule-Activity-Output-hasArtifact** and **QRule-Activity-hasArtifact**. They can be used to infer the fact that if an Artifact is related to the Input or Output of an Activity, the Activity might have such an Artifact. Such inference tasks can be used to find possible missing annotations of `has_Artifact`.

RE3.2 (*Check the Activity sequences*) needs to figure out the ordinary sequence of activities, sequence of decomposable activities, iterative and inverse sequence. Sequences of decomposable activities sometimes are not easy to navigate directly from the model specifications. Therefore, **QRule-Activity-hasPrecedingActivities-hasSubActivity** and **QRule-Activity-hasSucceedingActivities-hasSubActivity** are formalized to infer such implicit semantics.

The properties of sub-Activities could be passed to their super-Activities, which are validated in RE3.3 (*Check the Artifacts in sub-Activities*), RE3.4 (*Check the Actor-roles in sub-Activities*) and RE3.5 (*Check the goal annotations in sub-Activities*). **IRule-**

Table 9.2: SWRL queries and rules for RE2

RE2	Rule Name	SWRL formulation
RE2.1	**QRule-Activity-sameas**	$Activity(?x) \wedge same\_as(?x, ?y) \rightarrow query : select?x, ?y \wedge query : orderBy(?y)$
	QRule-Activity-kindof	$Activity(?x) \wedge kind\_of(?x, ?y) \rightarrow query : select(?x, ?y) \wedge query : orderBy(?x)$
	QRule-Activity-phaseof	$Activity(?x) \wedge phase\_of(?x, ?y) \rightarrow qurey : select(?x, ?y) \wedge query : orderBy(?x)$
RE2.2	**QRule-Activity-Input-mappedto**	$Activity(?x) \wedge has\_Input(?x, ?y) \wedge mapped\_to(?y, ?z) \rightarrow query : select(?x, ?y, ?z) \wedge query : orderBy(?z)$
	QRule-Activity-Output-mappedto	$Activity(?x) \wedge has\_Output(?x, ?y) \wedge mapped\_to(?y, ?z) \rightarrow query : select(?x, ?y, ?z) \wedge query : orderBy(?z)$
	QRule-Activity-hasArtifact-sameas	$Activity(?x) \wedge has\_Artifact(?x, ?y) \wedge same\_as(?y, ?z) \rightarrow query : select(?x, ?y, ?z) \wedge query : orderBy(?z)$
	QRule-Activity-hasArtifact-kindof	$Activity(?x) \wedge has\_Artifact(?x, ?y) \wedge kind\_of(?y, ?z) \rightarrow query : select(?x, ?y, ?z) \wedge query : orderBy(?z)$
	QRule-Activity-hasArtifact-partof	$Activity(?x) \wedge has\_Artifact(?x, ?y) \wedge part\_of(?y, ?z) \rightarrow query : select(?x, ?y, ?z) \wedge query : orderBy(?z)$
RE2.3	**QRule-Activity-hasActor-sameas**	$Activity(?x) \wedge has\_Actor - role(?x, ?y) \wedge same\_as(?y, ?z) \rightarrow query : select(?x, ?y, ?z) \wedge query : orderBy(?z)$
	QRule-Activity-hasActor-kindof	$Activity(?x) \wedge has\_Actor - role(?x, ?y) \wedge kind\_of(?y, ?z) \rightarrow query : select(?x, ?y, ?z) \wedge query : orderBy(?z)$
	QRule-Activity-hasActor-memberof	$Activity(?x) \wedge has\_Actor - role(?x, ?y) \wedge member\_of(?y, ?z) \rightarrow query : select(?x, ?y, ?z) \wedge query : orderBy(?z)$
RE2.4	**QRule-Activity-achievesHardGoal**	$Activity(?x) \wedge achieves(?x, ?y) \rightarrow query : select(?x, ?y) \wedge query : orderBy(?y)$
	QRule-Activity-positivelysatisfiesSoftGoal	$Activity(?x) \wedge positively\_satisfies(?x, ?y) \rightarrow query : select(?x, ?y) \wedge query : orderBy(?y)$
	QRule-Activity-negativelysatisfiesSoftGoal	$Activity(?x) \wedge negatively\_satisfies(?x, ?y) \rightarrow query : select(?x, ?y) \wedge query : orderBy(?y)$

Activity-subActivity-hasArtifact and **IRule-Activity-subActivity-hasActor** are used to introduce the implicit annotation of `has_Artifact` and `has_hasActor-role` for the super-Activity. The effects of goals from the sub-Activities could be transferred to the super-Activities but it is not necessary for all cases. RE3.5 is therefore formalized as SWRL queries not inference rules by only listing the possible goal annotations transferred from the sub-Activities.

RE3.6 (*Check the Precondition/Postcondition of Activities*) is formalized into **QRule-Activity-hassamePrecondition** and **QRule-Activity-hassamePostcondition**. It attempts to find out the redundant Activities with the same conditions or the workflow patterns with the branches divided from the conditions. Besides, for RE3.7 (*Check the information flow from the Output to the Input*) the information flow (/output-input flow) can be identified through comparing the annotated ***Input*** and ***Output*** in **QRule-Activity-hasOuput-mappedto-sameonto-Activity-hasInput**. If an output has an ontology reference which is also referenced by an input, there might be an information flow between the output and the input.

9.2.4 Formalize RE4 - Knowledge discovery requirements

RE4 is related to G4 (*The annotation should be helpful in the semantic reconciliation of models to facilitate reuse and integration of models.*). This requirement is for analyzing the potential semantic relations between different models. However, SWRL rules can only be applied in one ontology file, such as the SWRL formulations of RE1, RE2 and

Table 9.3: SWRL queries and rules for RE3

RE3	Rule Name	SWRL formulation
RE3.1	**QRule-Activity-hasInput-relatedArtifact**	$Activity(?x) \wedge has\_Input(?x, ?y) \wedge related\_Artifact(?y, ?z) \rightarrow query : select(?x, ?y, ?z) \wedge query : orderBy(?x)$
	QRule-Activity-hasOutput-relatedArtifact	$Activity(?x) \wedge has\_Output(?x, ?y) \wedge related\_Artifact(?y, ?z) \rightarrow query : select(?x, ?y, ?z) \wedge query : orderBy(?x)$
	IRule-Activity-Input-hasArtifact	$Activity(?x) \wedge has\_Input(?x, ?y) \wedge related\_Artifact(?y, ?z) \rightarrow Activity(?x) \wedge has\_Artifact(?x, ?z)$
	IRule-Activity-Output-hasArtifact	$Activity(?x) \wedge has\_Output(?x, ?y) \wedge related\_Artifact(?y, ?z) \rightarrow Activity(?x) \wedge has\_Artifact(?x, ?z)$
RE3.2	**QRule-Activity-hasPrecedingActivities-hasSubActivity**	$Activity(?x) \wedge has\_precedingActivities(?x, ?y) \wedge has\_subActivity(?x, ?z) \rightarrow query : select?y, ?z \wedge query : orderBy(?y)$
	QRule-Activity-hasSucceedingActivities-hasSubActivity	$Activity(?x) \wedge has\_succeedingActivities(?x, ?y) \wedge has\_subActivity(?x, ?z) \rightarrow query : select?z, ?y \wedge query : orderBy(?y)$
	QRule-Activity-iterative-preceding	$Activity(?x) \wedge has\_precedingActivities(?x, ?x) \rightarrow qurey : select(?x)$
	QRule-Activity-iterative-succeeding	$Activity(?x) \wedge has\_succeedingActivities(?x, ?x) \rightarrow qurey : select(?x)$
	IRule-Activity-preceding-inverse-succeeding	$Activity(?x) \wedge has\_precedingActivities(?x, ?y) \rightarrow Activity(?y) \wedge has\_succeedingActivities(?y, ?x)$
	IRule-Activity-succeeding-inverse-preceding	$Activity(?x) \wedge has\_succeedingActivities(?x, ?y) \rightarrow Activity(?y) \wedge has\_precedingActivities(?y, ?x)$
RE3.3	**IRule-Activity-subActivity-hasArtifact**	$Activity(?x) \wedge has\_subActivity(?x, ?y) \wedge has\_Artifact(?y, ?z) \rightarrow Activity(?x) \wedge has\_Artifact(?x, ?z)$
RE3.4	**IRule-Activity-subActivity-hasActor**	$Activity(?x) \wedge has\_subActivity(?x, ?y) \wedge has\_Actor - role(?y, ?z) \rightarrow Activity(?x) \wedge has\_Actor - role(?x, ?z)$
RE3.5	**QRule-Activity-subActivity-transitive-achievesHG**	$Activity(?x) \wedge has\_subActivity(?x, ?y) \wedge achieves(?y, ?z) \rightarrow query : select(?x, ?y, ?z) \wedge query : orderBy(?z)$
	QRule-Activity-subActivity-transitive-positivelysatisfiesSG	$Activity(?x) \wedge has\_subActivity(?x, ?y) \wedge positively\_satisfies(?y, ?z) \rightarrow query : select(?x, ?y, ?z) \wedge query : orderBy(?z)$
	QRule-Activity-subActivity-transitive-negativelysatisfiesSG	$Activity(?x) \wedge has\_subActivity(?x, ?y) \wedge negatively\_satisfies(?y, ?z) \rightarrow query : select(?x, ?y, ?z) \wedge query : orderBy(?z)$
RE3.6	**QRule-Activity-hassamePrecondition**	$Activity(?x) \wedge Activity(?y) \wedge name(?x, ?n) \wedge name(?y, ?m) \wedge swrlb : notEqual(?n, ?m) \wedge has\_Precondition(?x, ?z) \wedge has\_Precondition(?y, ?z) \rightarrow query : select(?x, ?y, ?z)$
	QRule-Activity-hassamePostcondition	$Activity(?x) \wedge Activity(?y) \wedge name(?x, ?n) \wedge name(?y, ?m) \wedge swrlb : notEqual(?n, ?m) \wedge has\_Postcondition(?x, ?z) \wedge has\_Postcondition(?y, ?z) \rightarrow query : select(?x, ?y, ?z)$
RE3.7	**QRule-Activity-hasOutput-mappedto-sameonto-Activity-hasInput**	$Activity(?x) \wedge has\_Output(?x?p) \wedge mapped\_to(?p, ?o) \wedge Activity(?y) \wedge has\_Input(?y, ?q) \wedge mapped\_to(?q, ?n) \wedge swrlb : equal(?o, ?n) \rightarrow query : select(?x, ?y, ?o)$

RE3. For RE4, we have to run the SWRL rules of RE1, RE2 and RE3 on different models respectively and analyze all query results manually. For example, implementing RE4.1 (*Find out semantic relationships between the Activities of different process models*) needs the SWRL rules **QRule-Activity-sameas**, **QRule-Activity-kindof** and **QRule-Activity-phaseof** for RE2.1 (*Find the model fragments of process models that reference to SCOR Management Process*). In this application, those rules are run on all the three models, and SCOR Management Processes in the domain ontology are used as the common references (which are specified in the variables in the SWRL rule formulations) to analyze the relationships between the query results of three models. The relationships usually applied in the analysis are ontological relations such as OWL Class subsumption, OWL Class equivalent, ObjectProperty part-whole relationship and etc.

To implement RE4.5 (*Find out possible integration points among different process models*), we consider the following integration cases by running related SWRL queries and rules for the above requirements.

- **Case 1.** *Output and input.* If the outputs in one model can be mapped to the inputs in another model, then it is possible to integrate the two models through the outputs and the inputs. **QRule-Activity-Output-mappedto** and **QRule-Activity-Input-mappedto** are run on three models respectively. The variable of ontology concept ?z can be replaced by a specific domain concept.

- **Case 2.** *Sequence of activities.* If the Activities from the three annotation models that have references of the SCOR process elements, then a possible integration of those Activities from different models can be checked according to the sequence definition in SCOR ontology. SWRL queries for RE2.1 (*Find the model fragments of process models that reference to SCOR Management Process*) can be run in this case, and the variable ?y is specified with a SCOR process element in each query.

- **Case 3.** *Semantic relationship of activities.* If two Activities from different models have certain semantic relationships with each other according to the domain annotation, then there is a possibility for two Activities to be integrated through the relationships. The SWRL queries for RE2.1 (*Find the model fragments of process models that reference to SCOR Management Process*) can also be used in this case, and the annotation relationships such as `same_as, kind_of` and `phase_of` should be concerned in the integration analysis.

- **Case 4.** *Ontological relationship of goals.* If two Activities from different models are annotated with the goals and the goals are same or have certain ontological relationships, the possible integration of two models can be analyzed based on the goal annotation. Goal concept is specified to replace the ontology concept variable ?y in the queries **QRule-Activity-achievesHardGoal**, **QRule-Activity-positivelysatisfiesSoftGoal** and **QRule-Activity-negativelysatisfiesSoftGoal**. Through the ontological relationship `has_parts` or `rdfs:subClassOf`, the parts or the sub-class of this goal are also specified to replace ?y respectively in the queries.

The SWRL queries and rules above are edited in Protégé-OWL SWRLTab (see Figure 9.1). They are attached to all the OWL files of the annotation models and will

be executed on those OWL instances for the evaluation. Section G.2 of Appendix G presents a SWRL file of above queries and rules .

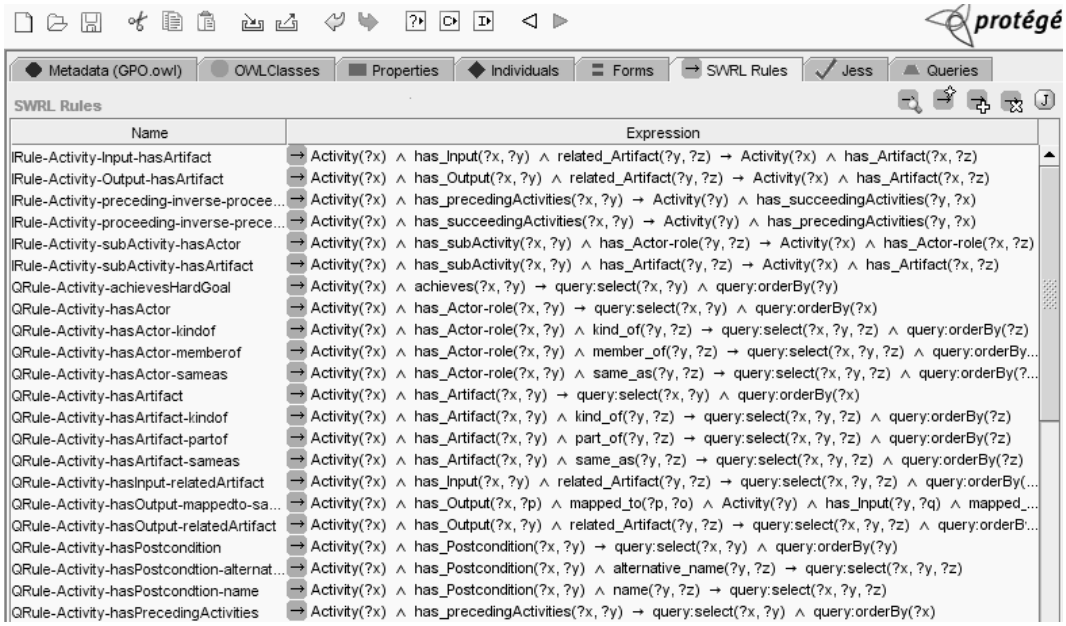


Figure 9.1: The SWRL rules in Protege-OWL SWRLTab

9.3 Applicability Validation in an Integration Application

The SWRL formulations for requirements are applied in the exemplar studies to validate the applicability of the proposed annotation approach. The validation is described by an application for integrating delivery process from enterprise A and B. The procedure of the implementation of this application is also the procedure of the validation.

The following steps have been undertaken for the integration application. In each step, some query questions are described by the application user, and the query questions are then inferred and answered by executing corresponding SWRL rules of the formalized requirements. Results are then analyzed and adjusted based on the semantic relationships of ontologies and models:

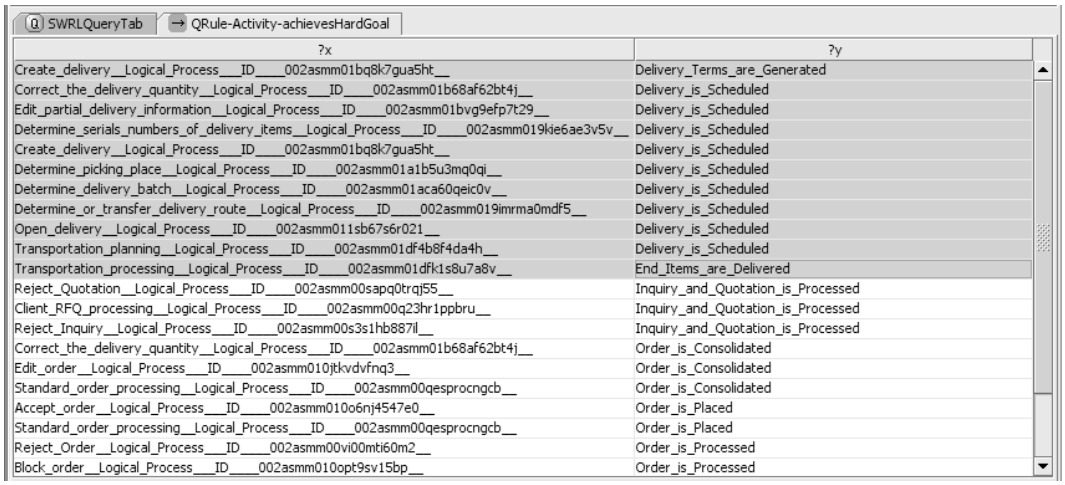
- **Step 1.** *Set application goals and get goal-relevant model fragments.*
  Running **QRule-Activity-achievesHardGoal** for RE2.4 (*Find the model fragments of process models that reference to SCOR Goal*) to find process model fragments achieving the integration goals.

  Query Question: Find model fragments from the PM_A, PM_{B1} and PM_{B2} which have impact on the goals `"Delivery_is_Processed"` and `"Invoice_to_Customer_is_Processed"`.

  Query Inference: Sub-goals of `"Delivery_is_Processed"` and `"Invoice_to_Customer_is_Processed"` are inferred to expand the query, such as `"Delivery_is_Scheduled"`, `"Delivery_Terms_are_Generated"`,

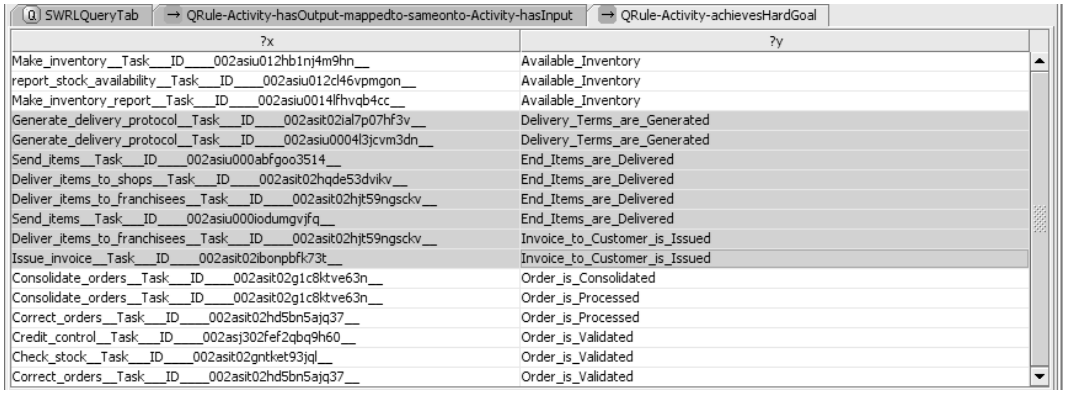
"End_Items_are_Delivered", "Invoice_to_Customer_is_Issued" and "Invoice_to_Customer_is_Paid".

Query Answer: As a result from executing the SWRL query, we have process model fragments (sub-processes) of "Delivering_Processing" in PM_A and "Deliver_items_to_franchisee" and "Deliver_items_to_shops" in PM_{B2} which are relevant to achieve those goals. The screen shot of the exemplified results in Protégé-OWL SWRL Query Tab are illustrated in Figure 9.2 and Figure 9.3.


SWRLQueryTab | QRule-Activity-achievesHardGoal

?x	?y
Create_delivery__Logical_Process___ID____002asmm01bq8k7gua5ht__	Delivery_Terms_are_Generated
Correct_the_delivery_quantity__Logical_Process___ID____002asmm01b68af62bt4j__	Delivery_is_Scheduled
Edit_partial_delivery_information__Logical_Process___ID____002asmm01bvg9efp7t29__	Delivery_is_Scheduled
Determine_serials_numbers_of_delivery_items__Logical_Process___ID____002asmm019kie6ae3v5v__	Delivery_is_Scheduled
Create_delivery__Logical_Process___ID____002asmm01bq8k7gua5ht__	Delivery_is_Scheduled
Determine_picking_place__Logical_Process___ID____002asmm01a1b5u3mq0qi__	Delivery_is_Scheduled
Determine_delivery_batch__Logical_Process___ID____002asmm01aca60qeic0v__	Delivery_is_Scheduled
Determine_or_transfer_delivery_route__Logical_Process___ID____002asmm019imrma0mdf5__	Delivery_is_Scheduled
Open_delivery__Logical_Process___ID____002asmm011sb67s6r021__	Delivery_is_Scheduled
Transportation_planning__Logical_Process___ID____002asmm01df4b8f4da4h__	Delivery_is_Scheduled
Transportation_processing__Logical_Process___ID____002asmm01dfk1s8u7a8v__	End_Items_are_Delivered
Reject_Quotation__Logical_Process___ID____002asmm00sapq0trqj55__	Inquiry_and_Quotation_is_Processed
Client_RFQ_processing__Logical_Process___ID____002asmm00q23hr1ppbru__	Inquiry_and_Quotation_is_Processed
Reject_Inquiry__Logical_Process___ID____002asmm00s3s1hb887il__	Inquiry_and_Quotation_is_Processed
Correct_the_delivery_quantity__Logical_Process___ID____002asmm01b68af62bt4j__	Order_is_Consolidated
Edit_order__Logical_Process___ID____002asmm010jtkvdvfnq3__	Order_is_Consolidated
Standard_order_processing__Logical_Process___ID____002asmm00qesprocngcb__	Order_is_Consolidated
Accept_order__Logical_Process___ID____002asmm010o6nj4547e0__	Order_is_Placed
Standard_order_processing__Logical_Process___ID____002asmm00qesprocngcb__	Order_is_Placed
Reject_Order__Logical_Process___ID____002asmm00vi00mti60m2__	Order_is_Processed
Block_order__Logical_Process___ID____002asmm010opt9sv15bp__	Order_is_Processed

Figure 9.2: The query result of **QRule-Activity-achievesHardGoal** on PM_A


SWRLQueryTab | QRule-Activity-hasOutput-mappedto-sameonto-Activity-hasInput | QRule-Activity-achievesHardGoal

?x	?y
Make_inventory__Task___ID____002asiu012hb1nj4m9hn__	Available_Inventory
report_stock_availability__Task___ID____002asiu012cl46vpmgon__	Available_Inventory
Make_inventory_report__Task___ID____002asiu0014lfhvqb4cc__	Available_Inventory
Generate_delivery_protocol__Task___ID____002asit02ial7p07hf3v__	Delivery_Terms_are_Generated
Generate_delivery_protocol__Task___ID____002asiu0004l3jcvm3dn__	Delivery_Terms_are_Generated
Send_items__Task___ID____002asiu000abfgoo3514__	End_Items_are_Delivered
Deliver_items_to_shops__Task___ID____002asit02hqde53dvikv__	End_Items_are_Delivered
Deliver_items_to_franchisees__Task___ID____002asit02hjt59ngsckv__	End_Items_are_Delivered
Send_items__Task___ID____002asiu000iodumgvjfq__	End_Items_are_Delivered
Deliver_items_to_franchisees__Task___ID____002asit02hjt59ngsckv__	Invoice_to_Customer_is_Issued
Issue_invoice__Task___ID____002asit02ibonpbfk73t__	Invoice_to_Customer_is_Issued
Consolidate_orders__Task___ID____002asit02g1c8ktve63n__	Order_is_Consolidated
Consolidate_orders__Task___ID____002asit02g1c8ktve63n__	Order_is_Processed
Correct_orders__Task___ID____002asit02hd5bn5ajq37__	Order_is_Processed
Credit_control__Task___ID____002asj302fef2qbq9h60__	Order_is_Validated
Check_stock__Task___ID____002asit02gntket93jql__	Order_is_Validated
Correct_orders__Task___ID____002asit02hd5bn5ajq37__	Order_is_Validated

Figure 9.3: The query result of **QRule-Activity-achievesHardGoal** on PM_{B2}

- **Step 2.** *Reconcile and align model semantics by using ontology as semantic mediator.*
  SWRL queries and rules are executed for RE4 (*Knowledge discovery requirements*) to find semantic relationships between ***Activity***, ***Artifact*** and ***Actor-role*** in those model fragments resulted from step 1. For example, we check the relationships between the Actor-roles involved in PM_A and PM_{B2}. We need to first find out what Actor-roles are involved in each process.

Query Question: Navigate Actor-roles in the processes of "Delivering_Processing" in PM_A and "Deliver_items_to_franchisee" and "Deliver_items_to_shops" in PM_{B2}.

Query Inference: In order to later find out the relationships between the Actor-roles of different models based on domain ontology, we need the query results including the model annotations by specifying which domain ontology concepts are referenced by the Actor-roles. Hence, **QRule-Activity-hasActor**, **QRule-Activity-hasActor-sameas**, **QRule-Activity-hasActor-kindof** and **QRule-Activity-hasActor-memberof** are used to query PM_A and PM_{B2}.

Query Answer: As results of **QRule-Activity-hasActor**, an Actor-role "logistics_department" is associated with the model fragments of "Delivering_Processing" in PM_A (Figure 9.4), and for PM_{B2} the Actor-roles "logistics_department", "financial_department", "Orbit_warehouse", "shop" and "franchisee" are queried out with respect to those process model fragments of "Deliver_items_to_franchisee" and "Deliver_items_to_shops" (Figure 9.5). The Actor-role "logistics_department" in PM_A is annotated with the SCOR ontology `"Logistics"` by the annotation relationship `"same_as"` (Figure 9.6). The Actor-role "logistics_department" in PM_{B2} is also `"same_as"` the SCOR ontology `"Logistics"` (Figure 9.7), while the Actor-roles "shop" and "franchisee" are both annotated with the SCOR ontology `"Customer"` by the annotation relationship `"kind_of"` (Figure 9.8). Besides, "financial_department" is `"same_as"` the SCOR ontology `"Finance"` and "Orbit_warehouse" is `"kind_of"` the SCOR ontology `"Warehouse"`.

SWRLQueryTab | QRule-Activity-hasActor | QRule-Activity-hasActor-sameas

?x	?y
Send_quotation__Logical_Process___ID____002asmm00pa5m38c6q3l__	client__Horizontal_Swimlane___ID____002asmm00oeom0j4eos6__
Create_or_transmit_shipping_papers__Logical_Process___ID____002asmm01cl28c...	sales_department__Horizontal_Swimlane___ID____002asmm00oeg111nl973__
Determine_or_transfer_delivery_route__Logical_Process___ID____002asmm019imr...	logistics_department__Horizontal_Swimlane___ID____002asmm00p3r9tn7e6c4__
Determine_delivery_batch__Logical_Process___ID____002asmm01aca60qeic0v__	logistics_department__Horizontal_Swimlane___ID____002asmm00p3r9tn7e6c4__
Monitor_delivery_date__Logical_Process___ID____002asmm011fpp64mv69i__	logistics_department__Horizontal_Swimlane___ID____002asmm00p3r9tn7e6c4__
Receive_delivery__Logical_Process___ID____002asmm00pbjlth9igtk__	client__Horizontal_Swimlane___ID____002asmm00oeom0j4eos6__
Transportation_planning__Logical_Process___ID____002asmm01df4b8f4da4h__	logistics_department__Horizontal_Swimlane___ID____002asmm00p3r9tn7e6c4__
Reject_Inquiry__Logical_Process___ID____002asmm00s3s1hb887il__	sales_department__Horizontal_Swimlane___ID____002asmm00oeg111nl973__
Pack_delivery_items__Logical_Process___ID____002asmm01bl32u5agihp__	logistics_department__Horizontal_Swimlane___ID____002asmm00p3r9tn7e6c4__
Delivering_Processing__Logical_Process___ID____002asmm0104dpp10il2e__	logistics_department__Horizontal_Swimlane___ID____002asmm00p3r9tn7e6c4__
Reject_Quotation__Logical_Process___ID____002asmm00sapq0trqj55__	sales_department__Horizontal_Swimlane___ID____002asmm00oeg111nl973__
Check_order__Logical_Process___ID____002asmm010jmggfpntci__	sales_department__Horizontal_Swimlane___ID____002asmm00oeg111nl973__

Figure 9.4: The query result of **QRule-Activity-hasActor** on PM_A

Result Analysis and Adjustment: There are no semantic relationships between `"Logistics"` and `"Customer"` in the SCOR ontology. A further step is hereby taken to search the corresponding Actor-roles for each other model. For example, search PM_A for the Actor-roles that have the annotation reference of `"Customer"`, and search PM_{B2} for the Actor-roles that have the annotation reference of `"Logistics"`.

By taking an example of `"Customer"`,

Query Question: Find the processes which have Actor-roles with the ontological reference of the SCOR ontology "Customer".

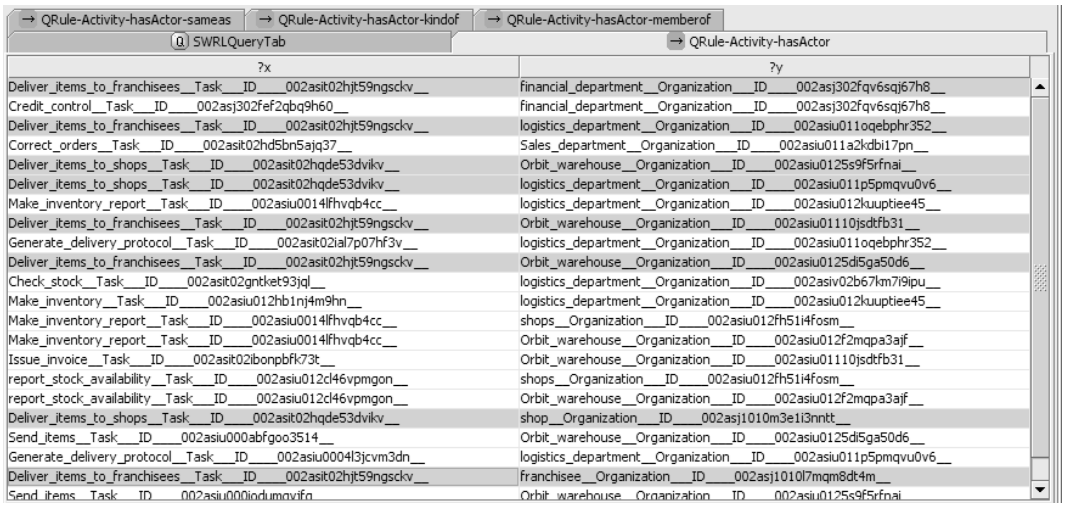


Figure 9.5: The query result of **QRule-Activity-hasActor** on PM_{B2}

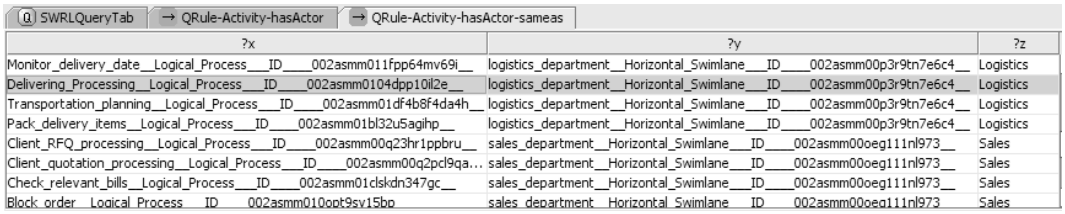


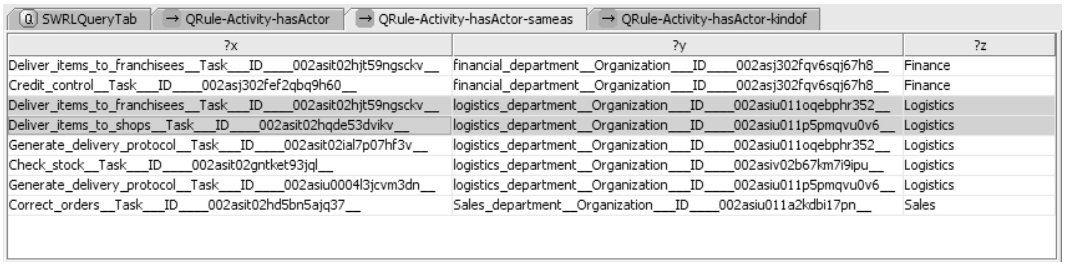
Figure 9.6: The query result of **QRule-Activity-hasActor-sameas** on PM_A

Query Answer: The "client" in PM_A is `same_as "Customer"`, and the Actor-roles "shop" and "franchisee" are `kind_of "Customer"` in PM_{B2}.

Query Analysis and Adjustment: "Shop" and "franchisee" in PM_{B1} (represented by the EEML modeling element `Organization`) are hence semantically aligned as sub-classes of "client" in PM_A (represented by the BPMN modeling element `Swimlane`).

Similar analysis of semantic relationships between the Activities is made by running **QRule-Activity-subActivity** to check their sub-activities and super-activities, and also running **QRule-Activity-phaseof**, **QRule-Activity-kindof** and **QRule-Activity-sameas** to find out how those Activities are interpreted according to the SCOR ontology. "Deliver_items_to_franchisee" and "Deliver_items_to_shops" in PM_{B2}, and "Delivering_Processing" in PM_A are all `phase_of "D1-Deliver_Stocked_Product"`. Since the Actor-roles "shop" and "franchisee" in PM_{B2} are sub-classes of "client" in PM_A, a corresponding adjustment of the semantic relationship between two models can be consequently made as "Deliver_items_to_franchisee" and "Deliver_items_to_shops" in PM_{B2} can be sub-activities of "Delivering_Processing" in PM_A.

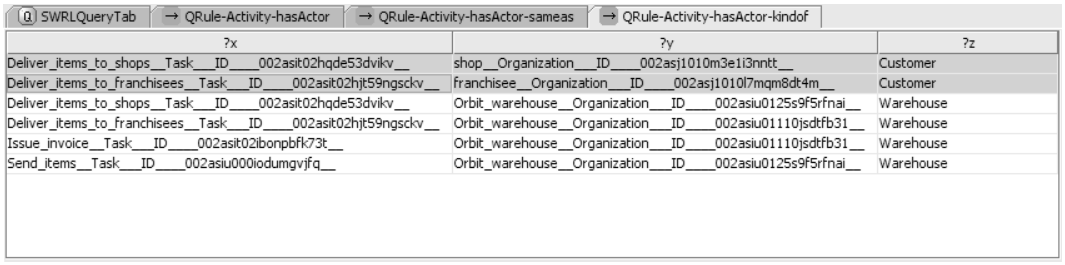
A possible integration path could be expressed as follows ("→" represents sequence flow, and "{}" represents sub-activities in the expression):



SWRLQueryTab | QRule-Activity-hasActor | QRule-Activity-hasActor-sameas | QRule-Activity-hasActor-kindof

?x	?y	?z
Deliver_items_to_franchisees__Task___ID____002asit02hjt59ngsckv__	financial_department__Organization___ID____002asj302fqv6sqj67h8__	Finance
Credit_control__Task___ID____002asj302fef2qbq9h60__	financial_department__Organization___ID____002asj302fqv6sqj67h8__	Finance
Deliver_items_to_franchisees__Task___ID____002asit02hjt59ngsckv__	logistics_department__Organization___ID____002asiu011oqebphr352__	Logistics
Deliver_items_to_shops__Task___ID____002asit02hqde53dvikv__	logistics_department__Organization___ID____002asiu011p5pmqvu0v6__	Logistics
Generate_delivery_protocol__Task___ID____002asit02ial7p07hf3v__	logistics_department__Organization___ID____002asiu011oqebphr352__	Logistics
Check_stock__Task___ID____002asit02gntket93jql__	logistics_department__Organization___ID____002asiv02b67km7i9ipu__	Logistics
Generate_delivery_protocol__Task___ID____002asiu0004l3jcvm3dn__	logistics_department__Organization___ID____002asiu011p5pmqvu0v6__	Logistics
Correct_orders__Task___ID____002asit02hd5bn5ajq37__	Sales_department__Organization___ID____002asiu011a2kdbi17pn__	Sales

Figure 9.7: The query result of **QRule-Activity-hasActor-sameas** on PM_{B2}



SWRLQueryTab | QRule-Activity-hasActor | QRule-Activity-hasActor-sameas | QRule-Activity-hasActor-kindof

?x	?y	?z
Deliver_items_to_shops__Task___ID____002asit02hqde53dvikv__	shop__Organization___ID____002asj1010m3e1i3nntt__	Customer
Deliver_items_to_franchisees__Task___ID____002asit02hjt59ngsckv__	franchisee__Organization___ID____002asj1010l7mqm8dt4m__	Customer
Deliver_items_to_shops__Task___ID____002asit02hqde53dvikv__	Orbit_warehouse__Organization___ID____002asiu0125s9f5rfnai__	Warehouse
Deliver_items_to_franchisees__Task___ID____002asit02hjt59ngsckv__	Orbit_warehouse__Organization___ID____002asiu01110jsdtfb31__	Warehouse
Issue_invoice__Task___ID____002asit02lbonpbfk73t__	Orbit_warehouse__Organization___ID____002asiu01110jsdtfb31__	Warehouse
Send_items__Task___ID____002asiu000iodumgvjfq__	Orbit_warehouse__Organization___ID____002asiu0125s9f5rfnai__	Warehouse

Figure 9.8: The query result of **QRule-Activity-hasActor-kindof** on PM_{B2}

$$
\begin{aligned}
&"Send\_inquiry"(PM_A) \rightarrow "Send\_quotation"(PM_A)\\
&\rightarrow "Credit\_control"(PM_{B2}) \rightarrow "Delivering\_Processing"(PM_A)\\
&\{"Deliver\_items\_to\_franchisee"(PM_{B2})\};"Deliver\_items\_to\_shops"(PM_{B2})\}\\
&\rightarrow "Ship\_items"(PM_{B2}) \rightarrow "Receive\_delivery"(PM_A)\\
&\rightarrow "Issue\_invoice"(PM_{B2}).
\end{aligned}
$$

- **Step 3.** *Re-order the sequence of integrating activities.*
  In Step 2, we are able to figure out the ontological relationships between the activities, but we need sequential relationship to integrate them as a process. For instance, the sequence of activities in the integration can be further refined through checking the sequential Activity and sub-Activities of "Send_quotation" in PM_A and the preceding Activities and sub-Activities of "Credit_control" in PM_{B2}. Applying SWRL queries and rules **QRule-Activity-hasSucceedingActivities** and **QRule-Activity-hasPrecedingActivities** for RE1 and **QRule-Activity-hasPrecedingActivities-hasSubActivity** for RE3.2 to rearrange the sequence of integrated process model fragments. Activity sequences are further checked with the SCOR domain ontology, and adjustment of the sequence and hierarchy of activities is made according to the integration context.

  Query Question: Navigate the succeeding Activities of "Send_quotation" in PM_A and the preceding Activities of "Credit_control" in PM_{B2}.

  Query Answer: "Client_quotation_processing" is found as the succeeding Activity of "Send_quotation" in PM_A (Figure 9.9), and "Check_stock" and "Correct_orders" are retrieved as the preceding Activities of "Credit_control" in PM_{B2}

(Figure 9.10).

SWRLQueryTab | QRule-Activity-hasSucceedingActivities

?x	?y
Open_delivery__Logical_Process___ID____002asmm011sb67s6r021__	Credit_control__Logical_Process___ID____002asmm00vrqmfbqngpt__
Pack_delivery_items__Logical_Process___ID____002asmm01bl32u5agih...	Create_delivery__Logical_Process___ID____002asmm01bq8k7gua5ht__
Pick_delivery_items__Logical_Process___ID____002asmm01b9r88vnk9d...	Pack_delivery_items__Logical_Process___ID____002asmm01bl32u5agihp__
Pick_delivery_items__Logical_Process___ID____002asmm01b9r88vnk9d...	Edit_partial_delivery_information__Logical_Process___ID____002asmm01bvg9efp7t29__
Pick_delivery_items__Logical_Process___ID____002asmm01b9r88vnk9d...	Check_availability_of_delivery_items__Logical_Process___ID____002asmm01aoegapa...
Send_quotation__Logical_Process___ID____002asmm00pa5m38c6q3l__	Client_quotation_processing__Logical_Process___ID____002asmm00q2pcl9qagna__
Transportation_planning__Logical_Process___ID____002asmm01df4b8f...	Transportation_processing__Logical_Process___ID____002asmm01dfk1s8u7a8v__
Transportation_processing__Logical_Process___ID____002asmm01dfk1...	Receive_delivery__Logical_Process___ID____002asmm00pbjlth9igtk__

Figure 9.9: The query result of **QRule-Activity-hasSucceedingActivities** on PM_A

SWRLQueryTab | QRule-Activity-hasSucceedingActivities

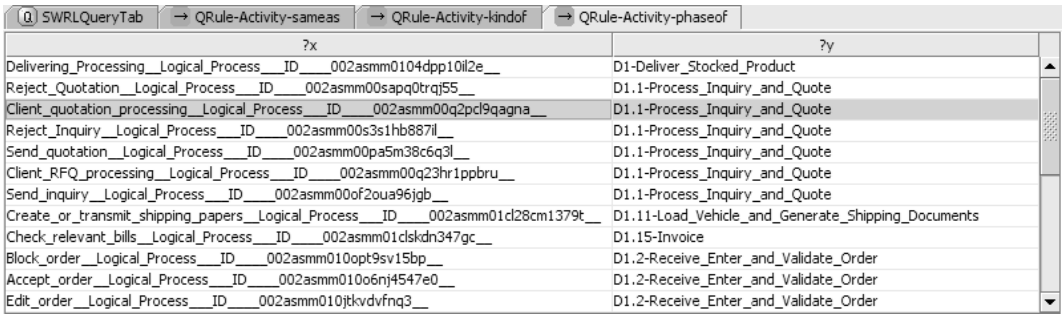
?x	?y
Check_stock__Task___ID____002asit02gntket93jql__	Credit_control__Task___ID____002asj302fef2qbq9h60__
Check_stock__Task___ID____002asit02gntket93jql__	Correct_orders__Task___ID____002asit02hd5bn5ajq37__
Consolidate_orders__Task___ID____002asit02g1c8ktve63n__	Check_stock__Task___ID____002asit02gntket93jql__
Correct_orders__Task___ID____002asit02hd5bn5ajq37__	Credit_control__Task___ID____002asj302fef2qbq9h60__
Correct_orders__Task___ID____002asit02hd5bn5ajq37__	Generate_delivery_protocol__Task___ID____002asiu0004l3jcvm3dn__
Correct_orders__Task___ID____002asit02hd5bn5ajq37__	Deliver_items_to_franchisees__Task___ID____002asit02hjt59ngsckv__
Correct_orders__Task___ID____002asit02hd5bn5ajq37__	Deliver_items_to_shops__Task___ID____002asit02hqde53dvikv__
Credit_control__Task___ID____002asj302fef2qbq9h60__	Generate_delivery_protocol__Task___ID____002asit02ial7p07hf3v__
Generate_delivery_protocol__Task___ID____002asit02ial7p07hf3...	Issue_invoice__Task___ID____002asit02ibonpbfk73t__
Generate_delivery_protocol__Task___ID____002asiu0004l3jcvm3...	Send_items__Task___ID____002asiu000iodumgvjfq__

Figure 9.10: The query result of **QRule-Activity-hasSucceedingActivities** on PM_{B2}

Query Analysis and Adjustment: The integrated sequence depends on i) the activity sequence definitions in original process models, ii) the reference sequence defined in the SCOR ontology, and also iii) the new integration requirements which might need different activity sequence from the former two. Adjustment includes mergence and decomposition of integrating activities. For instance, "Client_quotation_processing" in PM_A is annotated with the SCOR ontology concept of `"D1.1-Process_Inquiry_and_Quote"` (Figure 9.11), whilst "Correct_orders" in PM_{B2} is annotated as `phase_of "D1.2-Receive_Enter_and_Validate_Order"` (Figure 9.12). Therefore, "Correct_orders" in PM_{B2} can be succeeding activity of "Client_quotation_processing" in PM_A and then be adapted as a subActivity of "Standard_order_processing" in PM_A. While, the two Activities "Credit_control" in PM_A and in PM_{B2} can be merged into one Activity.

The result of the integration after this step is following ("/" is used to represent mergence):

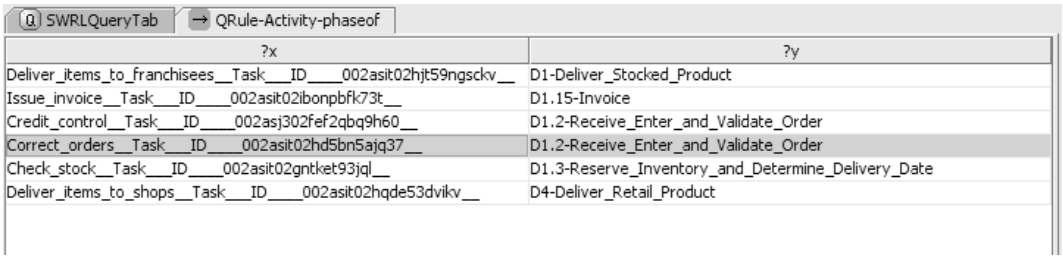
$$
\begin{aligned}
&\textit{"Send\_inquiry"}(PM_A) \rightarrow \textit{"Send\_quotation"}(PM_A) \rightarrow \\
&\textit{"Client\_quotation\_processing"}(PM_A) \rightarrow \textit{"Standard\_order\_processing"}(PM_A) \\
&\{...;\textit{"Correct\_orders"}(PM_{B2})\ ...\} \rightarrow \textit{"Credit\_control"}(PM_A/PM_{B2}) \rightarrow \\
&\textit{"Delivering\_Processing"}(PM_A)\{...;\textit{"Check\_stock"}(PM_{B2});...\} \\
&\rightarrow \textit{"Transportation\_processing"}(PM_A)/\textit{"Ship\_items"}(PM_{B2}) \\
&\rightarrow \textit{"Receive\_delivery"}(PM_A) \rightarrow \textit{"Issue\_invoice"}(PM_{B2}).
\end{aligned}
$$



SWRLQueryTab | QRule-Activity-sameas | QRule-Activity-kindof | QRule-Activity-phaseof

?x	?y
Delivering_Processing__Logical_Process___ID____002asmm0104dpp10il2e__	D1-Deliver_Stocked_Product
Reject_Quotation__Logical_Process___ID____002asmm00sapq0trqj55__	D1.1-Process_Inquiry_and_Quote
Client_quotation_processing__Logical_Process___ID____002asmm00q2pcl9qagna__	D1.1-Process_Inquiry_and_Quote
Reject_Inquiry__Logical_Process___ID____002asmm00s3s1hb887il__	D1.1-Process_Inquiry_and_Quote
Send_quotation__Logical_Process___ID____002asmm00pa5m38c6q3l__	D1.1-Process_Inquiry_and_Quote
Client_RFQ_processing__Logical_Process___ID____002asmm00q23hr1ppbru__	D1.1-Process_Inquiry_and_Quote
Send_inquiry__Logical_Process___ID____002asmm00of2oua96jgb__	D1.1-Process_Inquiry_and_Quote
Create_or_transmit_shipping_papers__Logical_Process___ID____002asmm01cl28cm1379t__	D1.11-Load_Vehicle_and_Generate_Shipping_Documents
Check_relevant_bills__Logical_Process___ID____002asmm01clskdn347gc__	D1.15-Invoice
Block_order__Logical_Process___ID____002asmm010opt9sv15bp__	D1.2-Receive_Enter_and_Validate_Order
Accept_order__Logical_Process___ID____002asmm010o6nj4547e0__	D1.2-Receive_Enter_and_Validate_Order
Edit_order__Logical_Process___ID____002asmm010jtkvdvfnq3__	D1.2-Receive_Enter_and_Validate_Order

Figure 9.11: The query result of **QRule-Activity-phaseof** on PM_A



SWRLQueryTab | QRule-Activity-phaseof

?x	?y
Deliver_items_to_franchisees__Task___ID____002asit02hjt59ngsckv__	D1-Deliver_Stocked_Product
Issue_invoice__Task___ID____002asit02ibonpbfk73t__	D1.15-Invoice
Credit_control__Task___ID____002asj302fef2qbq9h60__	D1.2-Receive_Enter_and_Validate_Order
Correct_orders__Task___ID____002asit02hd5bn5ajq37__	D1.2-Receive_Enter_and_Validate_Order
Check_stock__Task___ID____002asit02gntket93jql__	D1.3-Reserve_Inventory_and_Determine_Delivery_Date
Deliver_items_to_shops__Task___ID____002asit02hqde53dvikv__	D4-Deliver_Retail_Product

Figure 9.12: The query result of **QRule-Activity-phaseof** on PM_{B2}

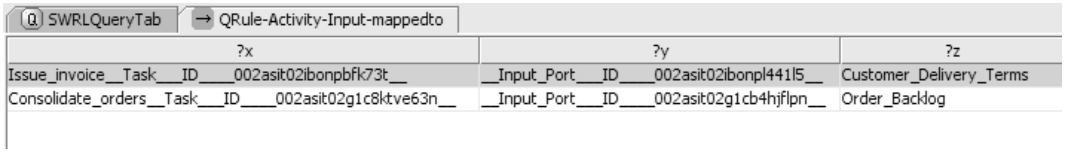
- **Step 4.** *Build output-input flow.*
  The step can be used to find the missing activities and also re-arrange the sequence of activities based on the output-input flow. For example, we can find an output of "Create_delivery" in in PM_A that matches an input of "Issue_invoice" in PM_{B2}.

  Query Question: Find out which Activity in PM_A produces the input of "Issue_invoice" in PM_{B2}.

  Query Inference: Such output-input mapping between different models need to be checked with the SCOR ontology as the mapping mediator, because the input/output parameters might be defined quite differently in two models. ***Input*** and ***Output*** of activities are checked through running **QRule-Activity-Input-mappedto** and **QRule-Activity-Output-mappedto** for RE2.2 (*Find the model fragments of process models that reference to SCOR Input/Output*).

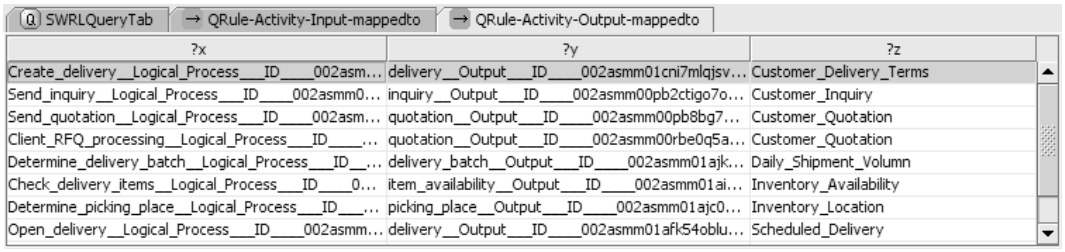
  Query Answer: **QRule-Activity-Input-mappedto** is executed on Activity of "Issue_invoice" in PM_{B2} and the input is mapped to `"Customer_Delivery_Terms"` (Figure 9.13). From the result of **QRule-Activity-Output-mappedto** in PM_A (Figure 9.14), Activity "Create_delivery" has an output "delivery" which is mapped to `"Customer_Delivery_Terms"` too.

  Query Analysis and Adjustment: Because the input of Activity "Issue_invoice" and the output of "Create_delivery" are both mapped to "Customer_Delivery_Terms", it is possible for the two Activities to be integrated with each other through the output-input flow. Not all the integrating points have strict integration sequences. For those non-strict integrating model fragments, more analysis have to be done manually based on the application


SWRLQueryTab | QRule-Activity-Input-mappedto

?x	?y	?z
Issue_invoice__Task___ID____002asit02ibonpbfk73t__	__Input_Port___ID____002asit02ibonpl441l5__	Customer_Delivery_Terms
Consolidate_orders__Task___ID____002asit02g1c8ktve63n__	__Input_Port___ID____002asit02g1cb4hjflpn__	Order_Backlog

Figure 9.13: The query result of **QRule-Activity-Input-mappedto** on PM_{B2}


SWRLQueryTab | QRule-Activity-Input-mappedto | QRule-Activity-Output-mappedto

?x	?y	?z
Create_delivery__Logical_Process___ID____002asm...	delivery__Output___ID____002asmm01cni7mlqjsv...	Customer_Delivery_Terms
Send_inquiry__Logical_Process___ID____002asmm0...	inquiry__Output___ID____002asmm00pb2ctigo7o...	Customer_Inquiry
Send_quotation__Logical_Process___ID____002asm...	quotation__Output___ID____002asmm00pb8bg7...	Customer_Quotation
Client_RFQ_processing__Logical_Process___ID____...	quotation__Output___ID____002asmm00rbe0q5a...	Customer_Quotation
Determine_delivery_batch__Logical_Process___ID__...	delivery_batch__Output___ID____002asmm01ajk...	Daily_Shipment_Volumn
Check_delivery_items__Logical_Process___ID____0...	item_availability__Output___ID____002asmm01ai...	Inventory_Availability
Determine_picking_place__Logical_Process___ID____...	picking_place__Output___ID____002asmm01ajc0...	Inventory_Location
Open_delivery__Logical_Process___ID____002asmm...	delivery__Output___ID____002asmm01afk54oblu...	Scheduled_Delivery

Figure 9.14: The query result of **QRule-Activity-Output-mappedto** on PM_A

context.

The final result of the integration application after above steps is presented ("|" is used to represent non-strict sequence):

$$"Send\_inquiry"(PM_A) \rightarrow "Send\_quotation"(PM_A) \rightarrow "Client\_quotation\_processing"(PM_A) \rightarrow "Standard\_order\_processing"(PM_A) \{"Correct\_orders"(PM_{B2})\} \rightarrow "Credit\_control"(PM_{B2}) \rightarrow "Delivering\_Processing"(PM_A)\{...;"Check\_stock"(PM_{B2});...;"Create\_delivery"\} \rightarrow "Issue\_invoice"(PM_{B2})|("Transportation\_processing"(PM_A)/"Ship\_items"(PM_{B2}) \rightarrow "Receive\_delivery"(PM_A)).$$

More details of the integration process are provided in Appendix F. By running the above SWRL queries and rules, the integration application has been implemented. The realized implementation validated the applicability of the proposed annotation approach through showing the capability of the annotation results to fulfill the requirements listed in section 9.1.1.

9.4 Discussion on Results of the Validation

9.4.1 Automatic vs. manual annotation

During the validation, we found that the results of the SWRL queries for RE1 (*Navigation requirements*) can display the most information of the models through the annotation, but still some semantics of original models are found missing or incomplete in the annotation models. The results of **QRule-Activity-subActivity** are the same as the original models, i.e. it is semantic completeness of the Activity composition for the annotation. Such completeness is guaranteed by Pro-SEAT's automatic transformation from Metis models to PSAM models

based on the meta-model annotation. However, the results of **QRule-Activity-hasPrecedingActivities**, **QRule-Activity-hasSucceedingActivities**, **QRule-Activity-hasPrecondition** and **QRule-Activity-hasPostcondition** are not complete because current Pro-SEAT does not support the automatic annotation of the sequence of Activities. We have to manually annotate such information.

When checking the results of **QRule-Activity-hasArtifact** and **QRule-Activity-hasActor**, it turns out that the automatic annotation of associating Artifact or Actor-role with Activity performs better on EEML models than on BPMN models. The reason is that EEML Resource Role (GPO:Artifact/Actor-role) is encapsulated in EEML Task (GPO:Activity) in Metis which behaves in the same way as GPO. However, the BPMN Logic Process (GPO:Activity) is encapsulated in BPMN Swimlane (GPO:Actor-role) but not in the reverse way. The relations can not be automatically transformed as `has_Actor-role` properties in PSAM models. Based on the above analysis, we can conclude that the function of automatic transformation should be improved in Pro-SEAT. RE1 (*Navigation requirements*) is almost fulfilled in spite of the missing annotation caused by the manual annotation.

9.4.2 Model analysis based on semantic relationships

The reference ontology is introduced in the SWRL queries for RE2 (*Search requirements*). The links between the model fragments and the ontology concepts are built through the semantic annotation. When executing the SWRL queries for RE2 (*Search requirements*) on the three model instances respectively, we found that the synonym (`same_as`) (i.e. semantic equivalent) relationship is mostly used in annotating ***Artifacts*** and ***Actor-roles***, while the hypernym (`kind_of`) and meronym (`part_of`, `member_of`) relationships are rarely used. Nevertheless, for ontology-based annotations of ***Activities*** the case is just reverse: more meronym (`phase_of`) relationships and hypernym than synonym are applied. Such phenomena is observed in all the three models. It shows that Artifacts and Actor-roles in the three models are not very specialized but relatively general and close to the SCOR standard. On the contrary, Activities in different models are quite various and meanwhile the modeling granularity of the Activity is smaller than the SCOR process elements.

9.4.3 Detecting missing annotation

In order to detect missing annotations, we run the query rules [2] together with corresponding inference rules [3]. For instance, when only running **QRule-Activity-hasArtifact** on PM_{B1}, 19 records are returned. If the query is run together with **IRule-Activity-subActivity-hasArtifact**, the results set consists of 30 records. By running **IRule-Activity-Input-hasArtifact** and **IRule-Activity-Output-hasArtifact** with **QRule-Activity-hasArtifact** the query returns 40 records. The results of the execution of those inference rules and queries on three models are listed in Table 9.4. We comparing record numbers of the query results, we can see that there are more missing annotation on Arifacts in PM_{B1} than in PM_A and in

[2] starting with "Q" in the formulation name (see Table 9.4)
[3] starting with "I" in the formulation name (see Table 9.4)

Table 9.4: Query results of the inferred annotation options

SWRL inference rules and queries	PM_A	PM_{B1}	PM_{B2}
QRule-Activity-hasArtifact	26	19	9
QRule-Activity-hasArtifact + IRule-Activity-subActivity-hasArtifact	27	30	12
QRule-Activity-hasArtifact + IRule-Activity-Input-hasArtifact	26	28	12
QRule-Activity-hasArtifact + IRule-Activity-Output-hasArtifact	32	33	11
QRule-Activity-hasArtifact + IRule-Activity-Input-hasArtifact + IRule-Activity-Output-hasArtifact	32	40	12
QRule-Activity-hasArtifact + IRule-Activity-Input-hasArtifact + IRule-Activity-Output-hasArtifact + IRule-Activity-subActivity-hasArtifact	33	47	15
QRule-Activity-hasActor	31	28	16
QRule-Activity-hasActor + IRule-Activity-subActivity-hasActor	32	28	23
QRule-Activity-hasPrecedingActivities	30	20	12
QRule-Activity-hasPrecedingActivities + IRule-Activity-succeeding-inverse-preceding	37	31	14
QRule-Activity-hasPrecedingActivities + IRule-Activity-succeeding-inverse-preceding + IRule-Activity-preceding-inverse-succeeding	37	31	14
QRule-Activity-hasSucceedingActivities	35	21	10
QRule-Activity-hasSucceedingActivities + IRule-Activity-preceding-inverse-succeeding	37	31	14
QRule-Activity-hasSucceedingActivities + IRule-Activity-preceding-inverse-succeeding + IRule-Activity-succeeding-inverse-preceding	37	31	14

PM_{B2}. The big discrepancy is between running **QRule-Activity-hasArtifact** alone and running **QRule-Activity-hasArtifact** together with **IRule-Activity-Input-hasArtifact**. It indicates that many Artifacts are allocated in the sub-Activities in PM_{B1}. Such knowledge is not explicitly represented in the original model so that it is difficult for an annotator to be aware of it when manually annotating. A similar case is observed when running **QRule-Activity-hasActor** and **IRule-Activity-subActivity-hasActor** on PM_{B2}. When comparing the three models we notice the following: most Actor-roles are modeled in the sub-Activities of PM_{B2}; no Actor-role is attached to the sub-Activities in PM_{B1}; since the way of modeling Actor-role in BPMN is different from EEML(see the previous paragraph), the annotation about `has_Actor-role` for each Activity is made manually but carefully (Only one annotation is missed by mistake). With respect to the annotation about the sequence of Activities, again the most missing annotations are found in PM_{B1}. The reason is still the hierarchy of the sub-Activities. Three levels of sub-Activities hierarchy prevents an annotator from picking all the preceding and succeeding Activities inherited from the super-Activities.

Not all of the inferred results should be considered as missing annotations. They are just optional annotations to disclose more implicit knowledge carried by the models. Some results of those queries also produce noise to the evaluation because they should not be the correct annotation. For example, the Activity v_n is the sub-Activity of V_1 and the Activity V_2 precedes V_1, but v_n is not the direct preceding Activity of V_2 because v_n is not the last sub-Activity of V_1. The query results of **QRule-Activity-subActivity-transitive-achievesHG**, **QRule-Activity-subActivity-transitive-positivelysatisfiesSG** and **QRule-Activity-**

subActivity-transitive-negativelysatisfiesSG can be used in the automatic goal annotation. In our method of "calculate the possible goal annotation from the subActivities" in the goal annotation, we synthesize the three inferences in the algorithms. Anyway, inference mechanisms should be taken into account in the development of the annotation tool for the sake of semi-automatic annotation.

9.4.4 Semantic validation

Some SWRL rules can be applied to validate semantic representations in both annotated and original process models. Two examples are provided here:

- When **QRule-Activity-hassamePostcondition** or **QRule-Activity-hassamePrecondition** is executed, the results list a set of Activities with same pre/postcondition. Through the observation, we can conclude two situations:

  1. The results are caused by the way of modeling **WorkflowPattern**: in an EEML model, the logic connection "join" or "choice" is modeled by an EEML Milestone, and the annotated workflow branches from this connection share a same GPO Condition ("eeml:milestone" is mapped to "gpo:condition" in the meta-model annotation). Two evaluation results are therefore brought out: a) The EEML model should be improved by changing the way of modeling to specify the semantics of different conditions. b) In the annotation model, conditions for different branches should be separate.
  2. Wrong model or redundant Activities with the same Conditions in the original models.

- **QRule-Activity-hasOutput-mappedto-sameonto-Activity-hasInput** is usually used to navigate the information flow between Activities, which also can be used for the semantic validation. For instance, two different Activities from PM_{B2} have a same local name "Generate_delivery_protocol", and the outputs of these two Activities are mapped to the same concept `"Customer_Delivery_Terms"` in the annotation model. According to the query results from **QRule-Activity-hasOutput-mappedto-sameonto-Activity-hasInput**, the outputs are supposed to be passed to the Activity "Issue_invoice" as the input. When we check the original model, one of the two Activities (named with "Generate_delivery_protocol") is a sub-Activity of "Deliver_items_to_franchisees", and it is indeed followed by the Activity "Issue_invoice" in the original model. However, the other "Generate_delivery_protocol" which is a sub-Activity of "Deliver_items_to_shops" has no link to "Issue_invoice" in the original model. Such observations can be concluded as follows:

  1. Two Activities "Generate_delivery_protocol" could be functionally implemented by one component.
  2. The annotation of the outputs of the two Activities should be specified distinctively.

9.5 Summary

Applicability of the proposed annotation method has been validated through a process knowledge management application based on annotated models. A set of requirements have been elicited for implementing the application. The validation has therefore been undertaken by checking the fulfillment of those requirements with the annotation results. A set of annotation goals in Chapter 8 has been associated with those requirements and measured during the implementation of the application. Analyzing the validation results also supplement the quality evaluation of Chapter 8. Moreover, the application has demonstrated how this work can gain the benefits from Semantic Web technology such as SWRL rules and Description Logic inferences through the validation.

Part IV

Synopsis

Chapter 10

Conclusions and Future Work

We conclude our work in this chapter. First, answers to the research questions and the achievements of the objectives are discussed. Then, the contributions are emphasized. Finally, limitations of current work and directions of future work are outlined.

10.1 Research Questions and Findings

The main research question presented in Chapter 1 is:

How can semantic interoperability of process models be improved by using semantic technologies such as ontologies in the process knowledge management applications?

Generally, this question has been answered by an ontology-based semantic annotation method, i.e. building the theory of a semantic annotation framework and a semantic annotation model based on a set of ontologies, and the design and implementation of an ontology-based semantic annotation tool, and as well as the applicability validation in business process knowledge management applications.

Furthermore, we elaborate the answers to more specific sub-questions.

- *RQ1. What kind of semantic interoperability problems exist in business process knowledge management?*
  This question is related to semantic heterogeneity of business process models. We first studied modeling basis of business process management and identified two levels semantic heterogeneity in process models — meta-model level and model level. In general, two cases of semantic discrepancies on both levels can be interpreted according to the relationships between *sign*, *referent*, and *concept* in the semiotic triangle: 1) various signs of referents refer to the same concepts; 2) synonymic signs of referents refer to different concepts. The discussion about modeling basis and semantic discrepancies is provided in Chapter 2, Chapter 3 and Chapter 4. We surveyed semantic heterogeneity of a number of process modeling languages by examining modeling constructs corresponding to the business process modeling perspectives in Chapter 3. Semantics of model contents are dependent on business domains, and only the general semantic discrepancies are considered in this work. The research scope is narrowed down to a study of models in one business domain which is exemplified in the exemplar studies in Chapter 7 and a walkthrough scenario in Chapter 9.

- *RQ2. What kind of ontologies are required for process knowledge management and how to represent them?*
  Ontologies can provides a standard and formal representation of a conceptualization, which can be used for semantic reconciliation. Through the analysis of semantic discrepancies at both levels, meta-model and model levels, we have determined that a process modeling ontology is needed for the meta-model level and a business domain ontology is necessary for the model level. A general process ontology — GPO has been proposed and the concepts and their relationships have been defined in a meta-model of process models in Chapter 4. A business domain is determined by the use cases so that the domain ontology is driven from the SCOR reference models. The ontological representations of SCOR have been exemplified in Chapter 7. Besides, a goal ontology representing the process objectives have been regarded as an additional requirement for managing process knowledge. The goal ontology is associated with the contextual semantics of process models, and moreover goal specifications can be used in the goal-driven process knowledge management. A general goal ontology representation has been defined in Chapter 5. In order to facilitate associating goals with processes, some concepts in GPO are reused in the goal ontology representation. A specific goal ontology is domain dependent and an example of the specific goal ontology of SCOR domain has been used in exemplar studies in Chapter 7. All the ontologies have been modeled in OWL with Protégé in order to make use of Semantic Web technologies in applications.

- *RQ3. What metadata are essential for process model interoperability and how are they defined concerning reference ontologies for reconciliation of the heterogeneous semantics of process models?*
  In Chapter 4 and Chapter 5, we have presented our semantic annotation framework, consisting of profile annotation, meta-model annotation, model annotation and goal annotation. The annotation metadata is also referred as annotation schema in this thesis. A set of metadata including Dublin Core has been categorized into the types of administrative, descriptive, preservation, technical and use to describe the profile information of process models in Table 4.1. In the meta-model annotation, different process modeling languages are mapped to GPO to reconcile the different modeling constructs with the common definitions in GPO. There are three cases of mapping: one-one, many-one and combination-one, which need to be specified in the meta-model annotation schema. GPO is the basis of the semantic annotation framework and it constructs the semantic annotation model for model annotation and goal annotation after the meta-model annotation. In the semantic annotation model, process knowledge in original models is represented in a common knowledge representation format. Model annotation is to refer model contents to domain ontology concepts. Simple reference is not sufficient for specifying the semantic discrepancies. We refined the reference through semantic relationships (such as synonym, polysemy, hypernym, hyponym, meronym, holonym and instance) which are usually used in ontology specifications, so that the model annotation looks like to build "intermediate ontologies" between the domain ontology concepts and the local model contents.

During the process knowledge management, the "intermediate ontologies" can be used to rank and infer the knowledge query results (see Chapter 8). Those semantic relationships are defined in the semantic annotation model together with the relationships defined in GPO. The semantic annotation model is the model annotation schema. Such model annotation schema is extended by goal annotation metadata `achieves`, `positively_satisfies` and `negatively_satisfies`, which are used to associate goal ontology concept with process model fragments. The semantic annotation schema is defined in OWL and a semantic annotation model is instantiated when it is generated as a result of meta-model annotation.

- *RQ4. How can Semantic Web technology to be incorporated in a tool using the proposed approach?*
  We have integrated the Protégé Java API into the prototype of our annotation tool — ProSEAT to manipulate the ontologies edited in Protégé (see Chapter 6). With the tool, users can browse the ontologies and select reference concepts from the ontology for the annotation. The tool supports manual mapping between GPO and process modeling languages, automatic generation of an OWL instances of a semantic annotation model for the model and goal annotation, manual model annotation and semi-automatic goal annotation. Annotation results are saved in the OWL instance model, which can be read by any OWL supported system. Semantic inference can be made on the annotation results using OWL DL reasoners such as Racer [61], FaCT++ [162], KAON2 [118], Pellet [96].

- *RQ5. How can we use the proposed approach to facilitate process knowledge management?*
  The annotation procedure with the annotation tool in exemplar studies has been elaborated in Chapter 7. Based on the annotation results, a process knowledge management application has been demonstrated in Chapter 9. We have validated the applicability of semantic annotation approach and results. The process knowledge management activities — process knowledge query and process model integration — have been analyzed through checking the satisfactions of a set of identified application requirements. Semantic Web technologies — OWL DL inference with SWRL rules has been applied in the applications and evaluations. The positive evaluation results proved the quality and applicability of the semantic annotation approach in the exemplified process knowledge management application.

From the answers to the research questions, we can associate them with the objectives specified in Chapter 1. We therefore conclude: the solutions to RQ1 achieve the objective "to investigate semantic heterogeneity issues in business process modeling"; the solutions to RQ2 and RQ3 achieve the objective "to explore a comprehensive annotation approach to deal with heterogeneous semantics of process knowledge with referenced ontology"; the solutions to RQ4 achieve the objective "to develop an annotation tool to implement the approach by applying Semantic Web technologies"; the solutions to RQ5 achieve the objective "to evaluate quality and use feasibility of the proposed approach and tool in supporting process knowledge management activities".

10.2 Summary of the Contributions

Based on the answers to the research questions and the achieved objectives, the work contributes to theoretical, methodological and practical body of knowledge.

- **Theoretical contribution:**
  - *Applying semiotic theory as the theoretical basis of the work.* Since our semantic annotation targets are process models at conceptual level, conceptual modeling theories have been adapted for this research. The meaning triangle based on the semiotic theory has been used to identify the semantic heterogeneity existing at meta-model and model levels. The relationships between ontology, model, meta-model and modeling language have been also analyzed through the semiotic framework in section 4.1.1.
  - *Survey and comparison of business process modeling languages and process ontologies according to the process perspectives defined in a Business Process Management Systems Paradigm (Figure 3.1).* The semantic heterogeneity of various business process modeling languages has been investigated through the language survey. Existing process ontologies have been surveyed and compared in order to grasp the common and core concepts for process semantics, which are the basis of our GPO.
  - *General Process Ontology (GPO).* Based on the investigation and analysis of process ontologies and business process modeling languages, a process ontology has been created for reconciling various process modeling constructs. GPO is a sep forward to the unified process ontology for the interoperability of process models.
- **Methodological contribution:**
  - *A semantic annotation framework for enriching the semantics of process models when treating them as knowledge.* A fundamental semantic annotation structure and annotation components (profile annotation, meta-model annotation and model annotation) depict how the process knowledge is organized through such annotation framework.
  - *An ontology-based annotation method.* A semantic annotation model has been formalized and a set of semantic mapping rules have been elaborated to guide users to apply the framework and conduct the annotation.
  - *A goal annotation method.* As the way of specifying the objective of processes, a goal ontology representation method has been proposed. Based on the underlying relationships between goal ontology and process models, semi-automatic goal annotation algorithms have been introduced to facilitate the annotation.
- **Practical contribution:**
  - *OWL representation of domain and goal ontologies driven from SCOR specifications.* Since domain and goal ontologies are domain dependent, SCOR

was chosen as the domain references for the annotation in the exemplar studies. SCOR specifications have not been originally represented in OWL ontologies. As the ontology engineering contributions, the SCOR ontologies in OWL representations in this work can be reused in other applications.

– *A prototype of the annotation tool.* The development of the annotation tool has provided the technical contribution to the system implementation by integrating Semantic Web technology. Also the prototype serves as a test-bed in the evaluation of our approach .

– *Quality evaluation and applicability validation.* The quality evaluation has presented how an evaluation procedure was conducted applying SEQUAL [80]. Through the applicability validation, a possible process knowledge management application has been demonstrated based on the annotation results. The ways of applying the relevant Semantic Web technologies such as DL reasoning and SWRL have also been presented in the application.

10.3 Limitations and Future Work

Semantic annotation enabling e-business and B2B is still an interesting topic in academic and industrial research. On one hand, the needs of semantic annotation for reconciling heterogeneous informations are obviously crucial for the cooperative business nowadays. On the other hand, there exist few comprehensive academic or industrial standards and mature off-the-shelf products for semantic annotation. Moreover, the related methodology and technology such as development and application of ontologies and Semantic Web, are still open issues for the ontology-based semantic annotation. Some limitations of the work and possible extensions are discussed in this section.

- Further automatic enhancement is needed to facilitate the annotation procedure. NLP techniques such as information extraction from model contents and AI techniques like machine learning might be considered in the annotation of semi-structured information too. Furthermore, model abstraction mechanism can be applied to categorize models.

- The work should be evaluated in various domains and applications. The approach has been evaluated on two models for one business domain. The exemplars are from literature sources, and the process models and domain ontologies are experimental design. The usability and applicability should be further validated through real cases studies.

- Evolution of ontologies and model changes have not been taken into account in the annotation. This issue is associated with the flexibility of the approach. Since no gold standard ontology is available and changes are impossible to avoid, the ontology will certainly evolve with the evolution of the domain knowledge, and process models might change too. When that happens, the annotation results will be inconsistent with models and ontologies. It should be possible to provide a mechanism to keep the annotation results for the unchanged parts of model and to evolve the ontology references when the ontology is modified.

- Semantics of relationships in original models have not been taken into account in this approach. Most of the annotation targets in our approach are entities/classes. Semantics conveyed by the relationships between the modeling entities/classes have not been preserved but transformed into the relationships between the GPO concepts, which might cause some loss of model semantics. However, the annotation of relationships is difficult due to the complexity and flexibility of the relationship representations in most semi-structured models.

- Relationship between the semantic annotations of process models at the conceptual level and at the execution level could be developed. Although process models in our application may be regarded as a service, the difference between a Web service and a process model is that a Web Service is executable. A Web service uses programming-like control constructs as their basic building blocks which are inadequate for all the modeling issues. Our approach can compensate for the inadequacy from a different modeling perspective, but more work needs to be completed.

Part V

Appendices

Appendix A

BPMN

This chapter presents the BPMN modeling elements and notations. The BPMN modeling elements consist of a set of core elements from which an entire set of the BPMN modeling elements are extended. The core elements support the requirement of a simple notation and define the basic look-and-feel of BPMN. The entire list of elements, including the core elements, help support requirement of a powerful notation to handle more advanced modeling situations [12]. The definition of the BPMN elements are standards from [12]. The complete notations are illustrated in [12], but this chapter illustrates the modeling notations implemented in Metis 5.2.2, which we have applied in our work.

A.1 BPMN Elements Categories

There are four basic categories of elements — Flow Objects, Connecting Objects, Swimlanes, and Artifacts. The elements classified into the four categories as follow.

- Flow Objects are the main graphical elements to define the behavior of a Business Process: Events, Activities, and Gateways.
- Connecting Objects provide ways of connecting the Flow Objects to each other or other information: Sequence Flow, Message Flow, and Association.
- Swimlanes are used to group the primary modeling elements: Pool and Lane.
- Artifacts model additional information about the Process: Data Objects, Group, and Annotation.

A.2 Flow Objects

A.2.1 Events

An Event is something that "happens" during the course of a business process. These events affect the flow of the process and usually have a cause (trigger) or an impact (result). Events are circles with open centers to allow internal markers to differentiate different triggers or results. There are three types of Events, based on when they affect the flow: Start, Intermediate, and End.

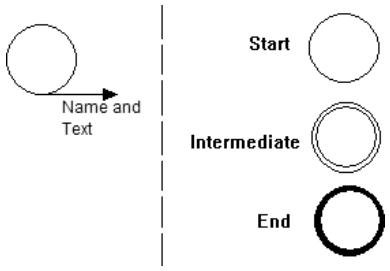


Figure A.1: BPMN Events

A.2.2 Activities

An Activity is a generic term for work that company performs. An activity can be atomic or non-atomic (compound). The types of activities that a part of a Process Model are: Process, Sub-Process, and Task. Tasks and Sub-Processes are either unbounded or a contained within a Pool. In Metis, notations of Tasks and Sub-Processes look same – round rectangles. Besides for those Activities, Input and Output can be attached. The notations of Activities are listed in Figure A.2.

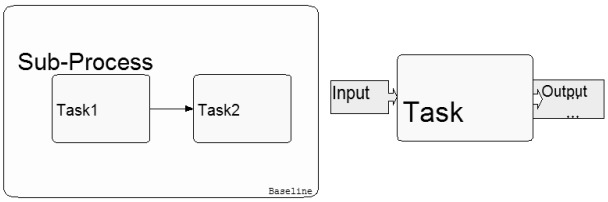


Figure A.2: BPMN Activities – Sub-Process, Task

A.2.3 Gateways

A Gateway is used to control the divergence and convergence of multiple Sequence Flow. Thus, it will determine branching, forking, merging, and joining of paths. Gateways are represented with diamonds and they can be specified with Gateway Control Types which are icons within the diamond shape (Figure A.3).

A.3 Connecting Objects

A.3.1 Sequence flows

A Sequence Flow is used to show the order that activities will be performed in a Process. A Sequence Flow can be further elaborated as Normal Flow and Exception Flow. A Normal Sequence Flow is usually notated by a line with an arrow connecting activities or gateways. Exception Flow occurs outside the Normal Flow of the Process and is based upon an Intermediate Event that occurs during the performance of the Process. Figure A.4 provides the graphical notation of a Normal Sequence Flow and an Exception Flow.

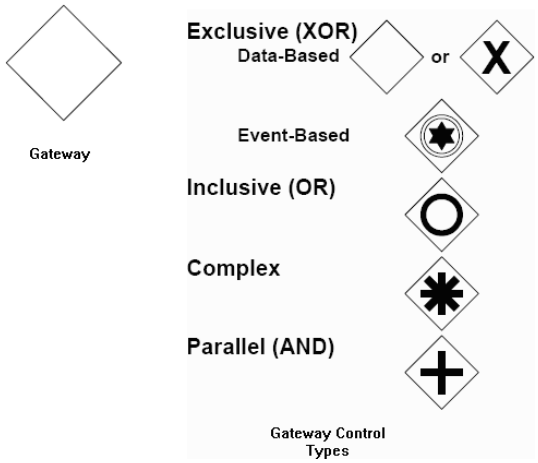


Figure A.3: BPMN Gateways

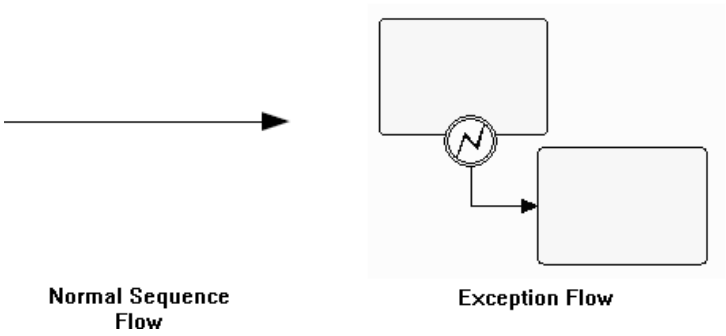


Figure A.4: BPMN Sequence Flows

A.3.2 Message flows

A Message Flow is used to show the flow of messages between two participants that are prepared to send and receive them. In BPMN, two separate Pools in the Diagram will represent the two participants (e.g., business entities or business roles). The notation for a Message Flow is illustrated in Figure A.5.

Figure A.5: BPMN Message Flow

A.3.3 Association

An Association is used to associate information with Flow Objects. Text and graphical non-Flow Objects can be associated with the Flow Objects. A dash line with an arrow is used to associate between Data Objects and Activities. A dash line without arrows is to link between Data Objects to a Sequence Flow or a Message Flow (Figure A.6).

Figure A.6: BPMN Associations

A.4 Swimlanes

A.4.1 Pool

A Pool represents a Participant in a Process. It is also acts as a "swimlane" and a graphical container for partitioning a set of activities from other Pools, usually in the context of B2B situations.

A.4.2 Lane

A Lane is a sub-partition within a Pool and will extend the entire length of the Pool, either vertically or horizontally. Lanes are used to organize and categorize activities. The Pool is notated as Swimlane Diagram and the Lane can be notated with Horizontal Swimlane or Vertical Swimlane, which are illustrated as Figure A.7.

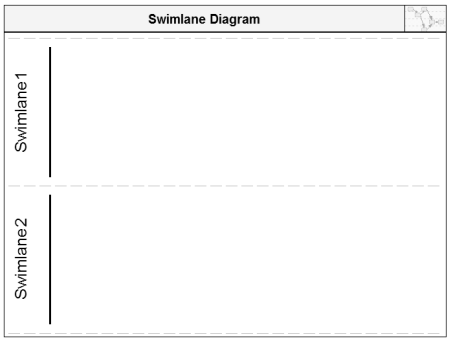


Figure A.7: BPMN Swimlanes – Pool and Lane

A.5 Artifacts

A.5.1 Data Object

Data Objects are considered Artifacts because they do not have any direct effect on the Sequence Flow or Message Flow of the Process, but they do provide information about what activities require to be performed and/or what they produce. The notation of Data Object is displayed in Figure A.8.

Grouping and Annotation are not supported by Metis 5.2.2 currently.

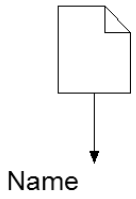


Figure A.8: BPMN Object Data

A.6 BPMN meta-model tree in Metis 5.2.2

The modeling elements are grouped into two categories in Metis, which are BPM Modeling Domain and Swimlane Diagram. In the Swimlane Diagram category, a Swimlane Diagram Object can group several Swimlanes and Swimlane can be either horizontal or vertical. Other modeling elements are in BPM Modeling Domain. Some additional modeling elements such as Input, Output, Control, Mechanism, Internal Flow are extended and categorized in the BPM Modeling Domain. The screen shot of the BPMN modeling constructs and notations in Metis 5.2.2 is illustrated in Figure A.9.

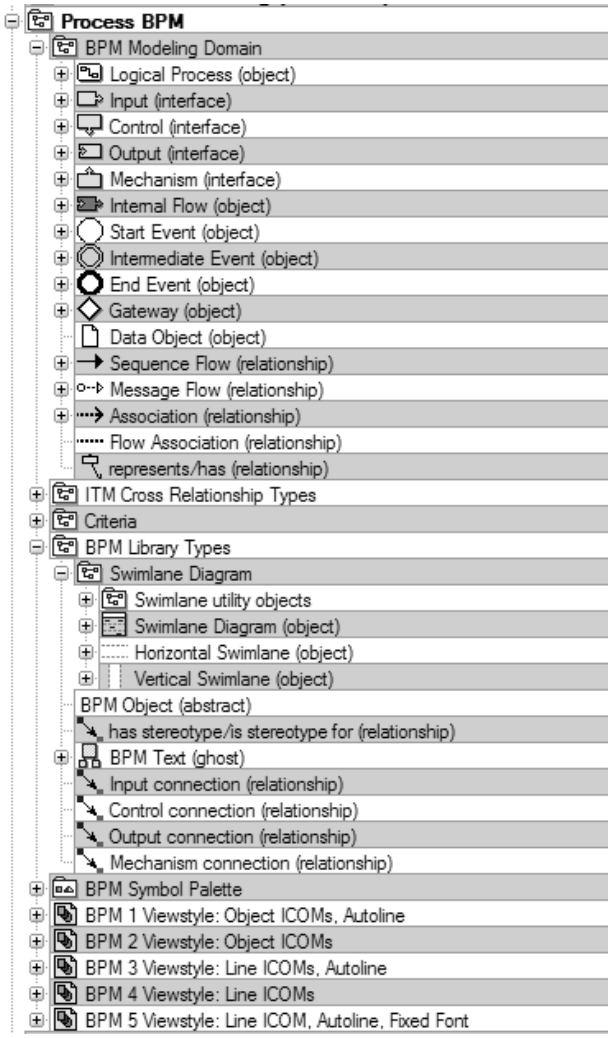


Figure A.9: BPMN modeling constructs and notations in Metis 5.2.2

Appendix B

EEML 2005

The language vocabulary of EEML 2005 is presented in this chapter. EEML can be used for process and enterprise modeling on different levels (both type and instance level). The language vocabulary is grouped into several domains focusing on different modeling perspectives.

Currently EEML includes four modeling domains — Process modeling, Resources modeling, Goal modeling and Data modeling (UML Class Diagram). In this research, we mainly concern the Process modeling and Resource modeling necessary for describing a business process.

B.1 Process Modeling Domain

B.1.1 Task

The Task concept should be used to represent a limited piece of work within a process. A task can be decomposed into smaller tasks, and, likewise, be a part of a larger task. The notation of a task is a round rectangle. Each task can have an Input Port and an Output Port which are the small diamonds attached to the left and right side within the round rectangle. A task-subtask-structure is shown in an onion-style notation as illustrated below (Figure B.1):

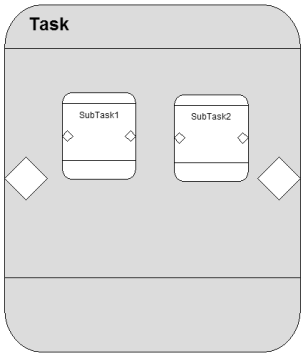


Figure B.1: EEML Tasks

B.1.2 Decision Point

A decision point models a (normally manual) decision performed as part of the overall process. The behavior of the decision point is largely defined by *Logical Relation* specified in a decision point, which describes how to handle multiple flows in or out from the decision point.

A default icon for the decision point is a diamond. And the logical relations can be specified within a diamond. Figure B.2 presents three types of decision points – OR, AND and XOR.

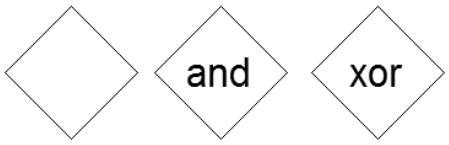


Figure B.2: EEML Decision Points

Figure B.3: EEML Milestone

B.1.3 Milestones

The Milestone concept is used to denote a decision point that is regarded to be of specific importance in the process it is a part of.

Examples: When ready to issue a contract, When ready to start a project.

The behavior of Milestone is largely defined by the property *Logical Relation*, which has similar semantics as described on decision-point above. The default icon for a Milestone is in Figure B.3.

B.1.4 Resource role

The term "role" designates some part played by some resource. A role should always be specified within a context. Typical examples of a role context are i) the organization as a whole, ii) organizational units within an organization iii) a task within the extended enterprise. Roles can be modeled in the context of Task, or externally, meaning the given (implicit) organizational context of the model. In addition, roles may be connected to flows, which is particularly relevant for the Roles of the object-type. Connecting object roles to flows is a convenient way to model document flow, for instance.

Roles may be considered as placeholders for resources, and are introduced to make it possible to talk about the use of resources without being concrete. In principal the

role types as such do not imply anything in terms of being active or passive. A role can be filled by any type of Resource. The default icon for Role is a circle (Figure B.4).

Figure B.4: EEML Resource role

B.2 Recourses Modeling Domain

Resources are things that play specific roles during the process execution or in the organization in general. The resource roles may vary greatly, - some resources are typically viewed as enablers, whereas others comprise the result (or part thereof) of the process.

EEML distinguishes between 6 types of resources, namely Person, Organization, Software Tool, Manual Tool, Material Object, and Information Object.

The typical way of connecting resources to processes is to assign them to resource roles included in the process, or attaching them to flows between tasks e.g. document flow.

The notations of those resources are illustrated in Figure B.5.

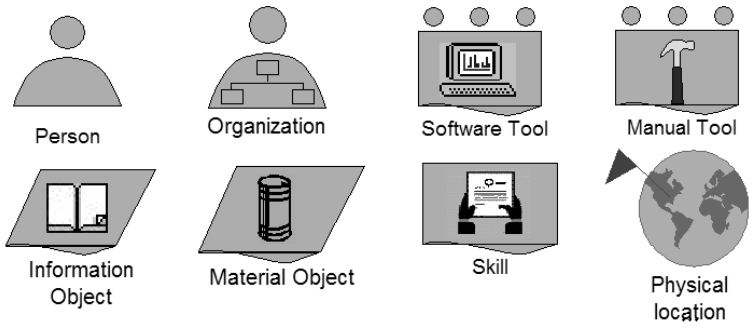


Figure B.5: EEML Resources

B.3 EEML modeling relationships

Besides the above object types, EEML defines a set of relationships to link those object types. The relationships are: flows into (between two Decision Points), has part (between a Task and its components, or between a Resource and its component Resources), has input port (between a Task and its Input Port), has output port (between a Task and its Output Port), has resource role (between a Task and its components of type Role), collaborates with (collaborations between two Tasks), is filled by (between Roles and its assignment of Resources), is candidate for (the possible filling of a Role

by a Resource), describe flow (linking a Resource role to a Flows Into), has member (between Organization and its components of Person).

B.4 EEML 2005 in Metis 5.2.2

The EEML 2005 modeling constructs are organized in different domains in Metis 5.2.2. Each domain consists of a set of object types and relationships. The metamodel tree structure of EEML in Metis 5.2.2 is displayed in Figure B.6.

Figure B.6: EEML 2005 modeling constructs in Metis 5.2.2

Appendix C

SCOR

SCOR — Supply Chain Operations Reference model [163] is a process reference model that has been developed and endorsed by the Supply Chain Council (SCC) as the cross-industry de facto standard diagnostic tool for supply chain management. The SCOR process reference model provides standard descriptions of management processes, a framework of relationships among the standard processes, standard metrics to measure process performance, management practices that produce best-in-class performance, standard alignment to features and functionality.

By applying process modeling building blocks, the model hereby can be used to describe supply chains that are very simple or very complex using a common set of definitions. SCOR provides three-levels of process detail, namely top level (Level 1), configuration level (Level 2), and process element level (Level 3). The three levels are illustrated in Figure C.1 [208].

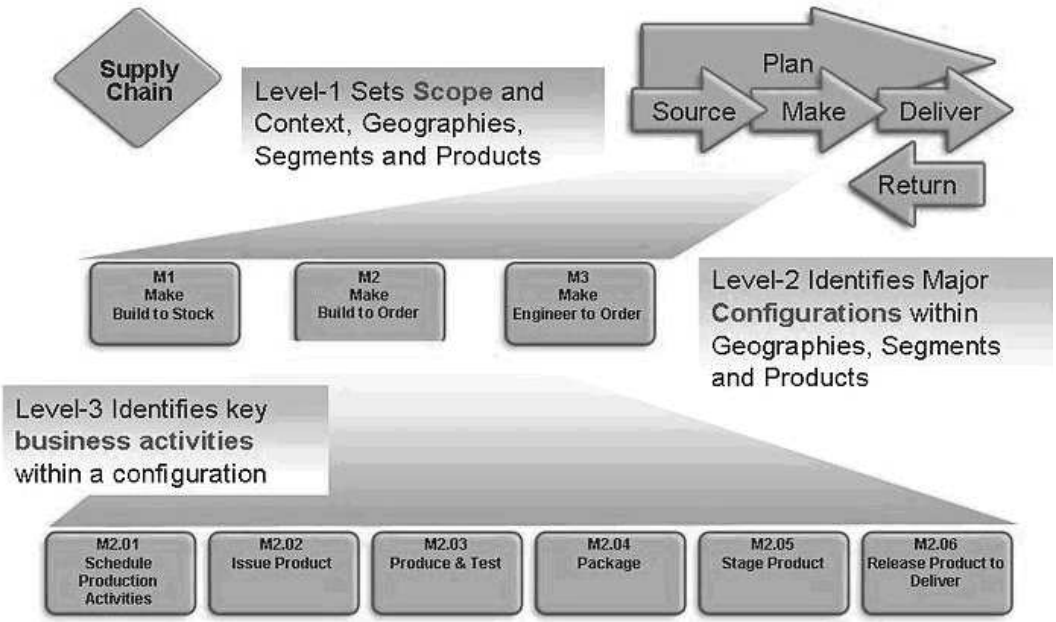


Figure C.1: Three levels in the SCOR scope [208]

- Level 1 defines the scope and content for the Supply Chain Opertations Reference-model. Here basis of competition performance targets are set.
- A company's supply chain can be "configured-to-order" at Level 2 from 30 core "process categories". Companies implement their operations strategy through the configuration they choose for their supply chain.

Level 3 defines a company's ability to compete successfully in its chosen markets, and consists of:

- Process element definitions
- Process element information inputs, and outputs
- Process performance metrics
- Best practices, where applicable
- System capabilities required to support best practices
- Systems/terms

The metrics are used in conjunction with performance attributes. The *Performance Attributes* are characteristics of the supply chain that permit it to be analyzed and evaluated against other supply chains with competing strategies. Most metrics in the Model are hierarchical just as the process elements are hierarchical. Level 1 Metrics are created from lower level calculations and are primary, high level measures that may cross multiple SCOR processes. Lower level calculations (Level 2 and metrics) are generally associated with a narrower subset of processes.

C.1 Level 1 Process Definitions

SCOR is based on five distinct management processes: Plan, Source, Make, Deliver, and Return.

- Plan - Processes that balance aggregate demand and supply to develop a course of action which best meets sourcing, production, and delivery requirements.
- Source - Processes that procure goods and services to meet planned or actual demand.
- Make - Processes that transform product to a finished state to meet planned or actual demand.
- Deliver - Processes that provide finished goods and services to meet planned or actual demand, typically including order management, transportation management, and distribution management.
- Return - Processes associated with returning or receiving returned products for any reason. These processes extend into post-delivery customer support.

Level 1 Metrics do not necessarily related to a SCOR Level 1 process. The Level 1 Metrics are the calculations by which an implementing organization can measure how successful they are in achieving their desired positioning within the competitive market space. Figure C.2 list a set of Level 1 Metrics defined in SCOR version 7.0.

Level 1 Metrics	Performance Attributes				
	Customer-Facing			Internal-Facing	
	Reliabilty	Responsiveness	Flexibility	Cost	Assets
Perfect Order Fulfillment	✔				
Order Fulfillment Cycle Time		✔			
Upside Supply Chain Flexibility			✔		
Upside Supply Chain Adaptability			✔		
Downside Supply Chain Adaptability			✔		
Supply Chain Management Cost				✔	
Cost of Goods Sold				✔	
Cash-to-Cash Cycle Time					✔
Return on Supply Chain Fixed Assets					✔

Figure C.2: Performance Attributes and Level 1 Metrics [164]

C.2 Level 2 Toolkit

At Level 2, each process can be further described by type. The SCOR process types are *Planning*, *Execution* and *Enable*. The SCOR configuration toolkit defines process categories by the relationship between a SCOR process and a process type. Practitioners select appropriate process categories from the SCOR configuration toolkit to represent their supply-chain configuration(s). The relationship of SCOR process, process type and process category are presented in Figure C.3. The overview of Level 2 process categories are outlined in Figure C.4.

"SCOR Configuration Toolkit"

		SCOR Process					
		Plan	Source	Make	Deliver	Return	
Process Type	Planning	P1	P2	P3	P4	P5	Process Category
	Execution		S1- S3	M1- M3	D1 - D3	R1-R3	
	Enable	EP	ES	EM	ED	ER	

Figure C.3: Process Categories [164]

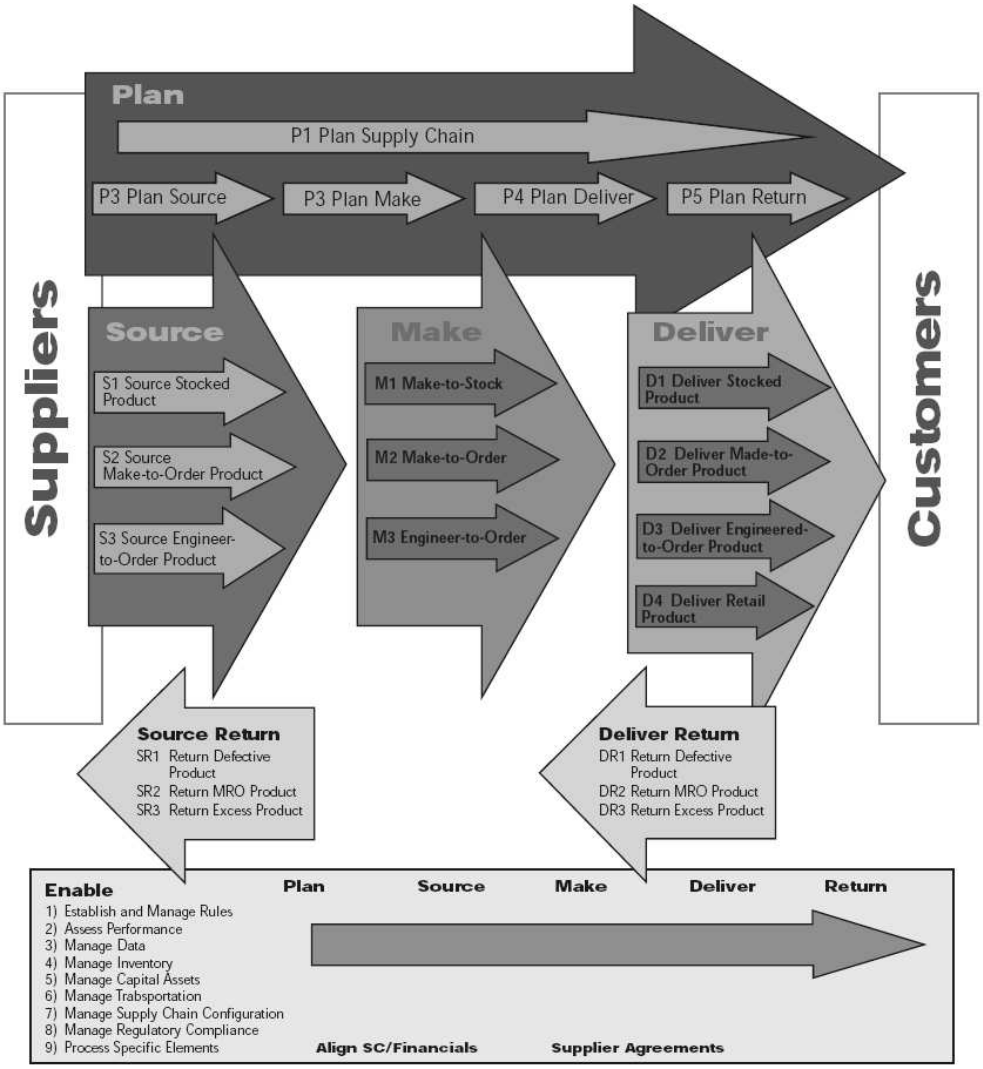


Figure C.4: Level 2 Toolkit [164]

C.3 Level 3 Process Elements

Level 3 presents detailed process element information, including process model, process element definition, performance attributes and accompanying metrics, for each Level 2 Process Category. Process details of S1 Source Stocked Product and D1 Deliver Stocked Product are exemplified in models in Figure C.5 and C.6. In the models, the process elements are ordered in sequence and specified with inputs and outputs. Figure C.7 provides some examples of metrics associated with the performance attributes for the process element `Schedule Product Deliveries`.

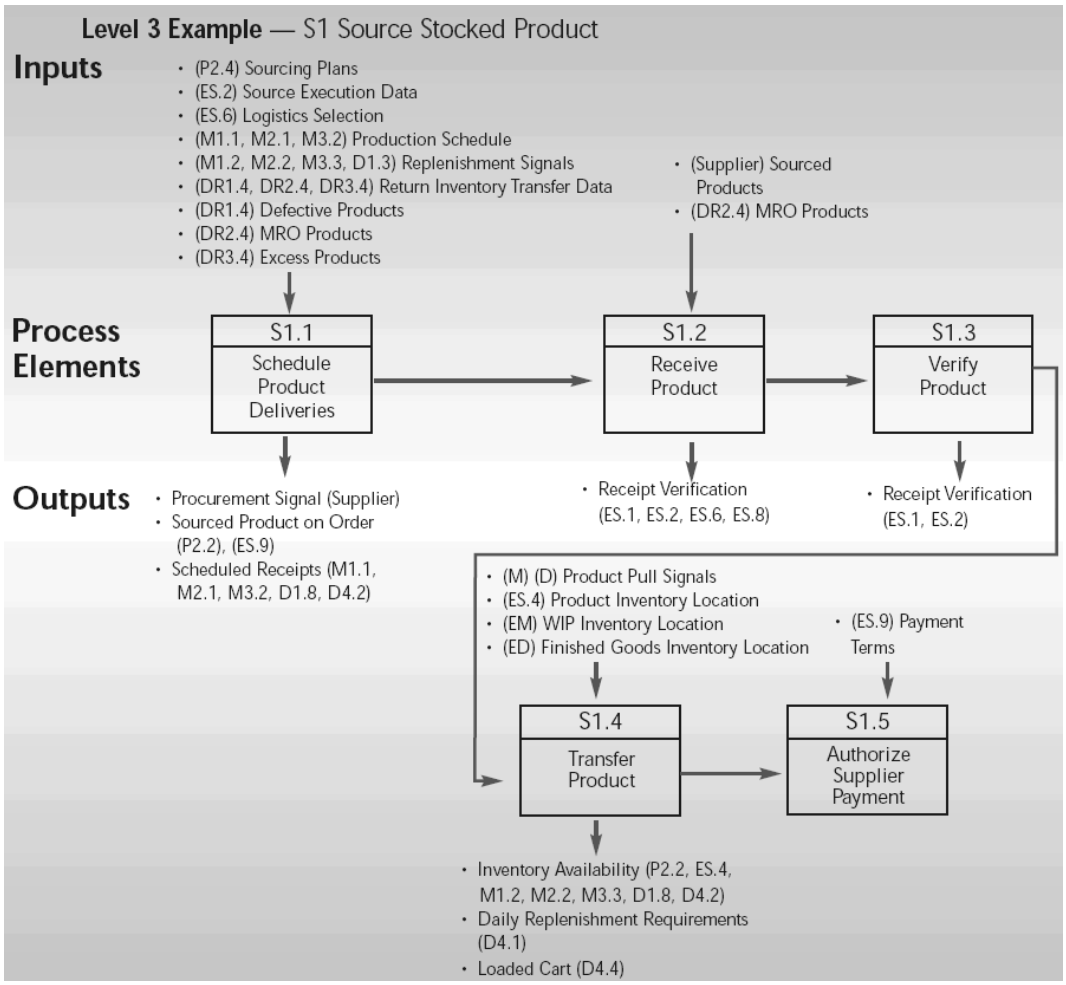


Figure C.5: Level 3 process elements of S1 Source Stocked Product [164]

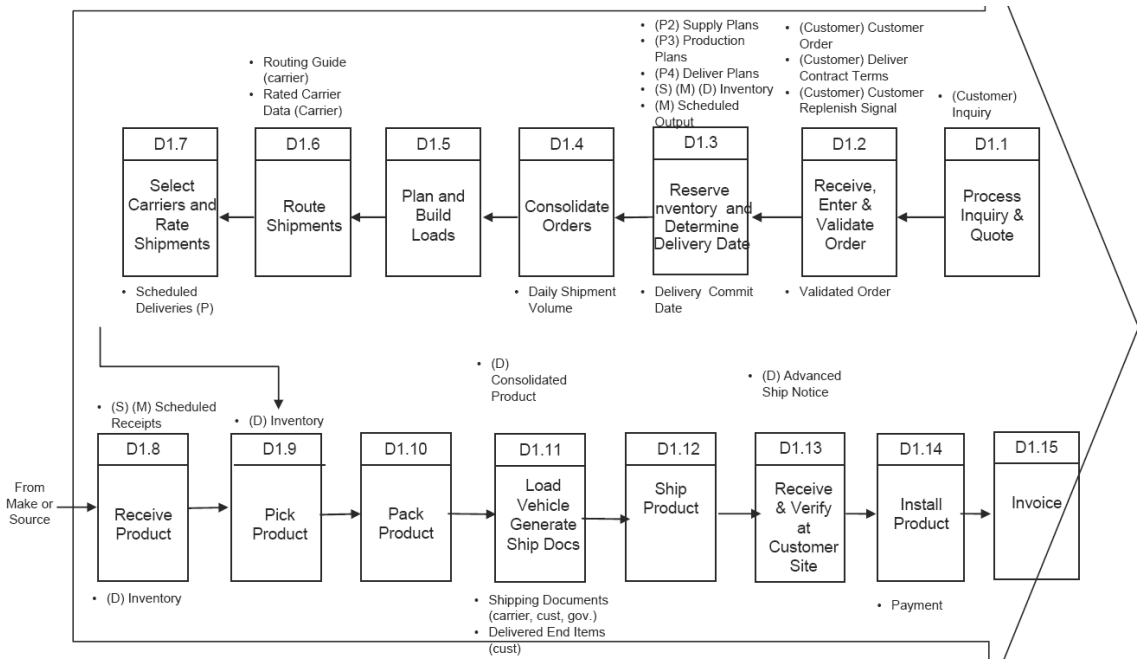


Figure C.6: Level 3 process elements of D1 Deliver Stocked Product [176]

Performance Attributes	Metric
Reliability	% Schedules Generated within Supplier's Lead Time % Schedules Changed within Supplier's Lead Time
Responsiveness	Schedule Product Deliveries Cycle Time
Flexibilty	None Identified
Cost	Schedule Deliveries Costs as a % of Product Acquisitions Costs
Assets	Return on Supply Chain Assets

Figure C.7: Level 3 metrics for S1.1 Schedule Product Deliveries [164]

Appendix D

Algorithm for Semi-Automatic Goal Annotation

The semi-automatic goal annotation is implemented through the algorithms in the following cases.

- **A.** Match a target activity (av') of a goal (g) in the goal ontology (G) with the ***Activity*** in a PSAM model(av).
- **B.** Match a target artifact (af') of a goal (g) in the goal ontology (G) with the ***Artifact*** and ***Output*** in a PSAM model (af, o).
  - **B.I** Match af' with the annotation of o.
  - **B.II** Match af' with the annotation of af which is directly associated with av through *has_Activity* relationship.
  - **B.III** Match af' with the annotation of af which is indirectly associated with av through *has_Output* and *related_Artifact* relationships.
- **C.** Match a target role (ar') of a goal (g) in the goal ontology (G) with the ***Act-role*** in a PSAM model (ar).
- **D.** Match a target constraint (c') of a goal (g) in the goal ontology (G) with the ***Precondition***, ***Postcondition***, and ***Exception*** in a PSAM model (pre, $post$, e).

For the automatic match, we exploit the semantic mapping through both ontology comparison and string match between ontology references and models. To rank the mapping results, weights are assigned to the different ways of mapping (σ is for a weight of ontology comparison. τ is for a weight of string match). In ontology comparison, three different semantic relationships applied in model annotation are taken into account in the assignment of weights. The synonym ("same_as") relationship is given the highest weight for complete match ($\sigma = 1$), and the hypernym ("kind_of") is given a lower weight for a subsumption relationship ($\sigma = 0.8$), and the meronym ("phase_of" for activity, "part_of" for artifact, "member_of" for actor-role) is given the lowest weight for a subset relationship ($\sigma = 0.5$). In the string match, the exact match

gets higher weight than the sub-string match ($\tau = 1$ for exact match, and $\tau = 0.5$ for sub-string match).

Weights are also different for each case. We set different parameters for weighing each case. The values of parameters can be set by users. In the algorithm, α stands for the weight of case **A** with ontology comparison and β for case **A** with string match. The case **B** is relatively complex. Sub-cases are distinguished for case **B**: case **B.I** – the target artifact af' of the goal concept is mapped to the annotation of the ***Output***; case **B.II** – the target artifact af' of the goal concept is mapped to the annotation of an ***Artifact*** which is directly associated with the activity; case **B.III** – the target artifact af' of the goal concept is mapped to the annotation of an ***Artifact*** which is indirectly related with the activity. An ***Artifact*** can be associated with an Activity through direct relation "has_Artifact" or through the indirect relation "related_Artifact" of the ***Output***. If the association is built through the indirect relation, the state of the target artifact af' should be checked to match the ***Output*** too. Parameter θ is used for weight of case **B.I**, γ is for case **B.II** with ontology comparison, and δ is for case **B.II** with string match. ϵ and μ are for case **B.III**. That the models are mapped to the goals by case **C** (matching the target role) is regarded less important than by the other cases, because the actor role is seldom considered as the stand alone target of a goal. Two parameters ε and ζ are applied in case **C** for ontology comparison and string match respectively. The values for the two parameters should be set lower than other parameters. No ontology comparison is employed in case **D**, so one parameter η is assigned in such case.

When matching a goal concept (g) with an activity (av) in models, all the four cases must be checked on g and av. The result of a match is a total weight by summing the weights from four cases. The higher total weight is, the better g matches av.

Algorithm 1 Goal Annotation Algorithm for Case **A**

Require: weight parameters α and β
initialize values of $weightforactivityonto$, $weightforactivityname$, σ, τ as 0
$onto = OD(av)$ {av is the activity to be annotated in goal annotation, and $OD(av)$ gets the domain ontological concept which was annotated to av in model annotation}

$r = SR(av, onto)$ {$SR(av, onto)$ gets the semantic relationship between av and $onto$}
for each target activity av' of a g in goal ontology **do**
  if o equals av' **then**
    if r is "same_as" **then**
      $\sigma = 1$
    else if r is "kind_of" **then**
      $\sigma = 0.8$
    else if r is "phase_of" **then**
      $\sigma = 0.5$
    end if
  end if
  $weightforativityonto = weightforactivityonto + \sigma$
  if $av.name$ equals $av'.name$ by string match **then**
    $\tau = 1$
  else if $av.name$ partly equals $av'.name$ by string match **then**
    $\tau = 0.5$
  end if
  $weightforactivityname = weightforactivityname + \tau$
end for
return $\alpha * weightforactivityonto + \beta * weightforactivityname$

Algorithm 2 Goal Annotation Algorithm for Case **B.I**

Require: weight parameters θ
initialize value of $weightforoutput$ as 0
$Arrayout[]$ is a set of output associated with av {av is the activity to be annotated in goal annotation. $out[]$ is got from *has_Output relationship*}
for each $output$ in $out[]$ **do**
  $onto = OD(out)$ {$OD(out)$ gets the domain ontological concept which was annotated to $output$ through *mapped_to* relationship}
  for each target artifact af' of a g in goal ontology **do**
    if $onto$ equals af **then**
      $weightforoutput = weightforoutput + 1$
    end if
  end for
end for
return $\theta * weightforoutput$

Algorithm 3 Goal Annotation Algorithm for Case **B.II**

Require: weight parameters γ, δ
  initialize value of $weightforartifactonto$, $weightforartifactname$ as 0
  $Arrayaf[]$ is a set of artifacts associated with av through $has\_Artifact$
  for each af in $af[]$ **do**
    $onto = OD(af)$ {$OD(af)$ gets the domain ontological concept annotated to af}
    for each target artifact af' of a g in goal ontology **do**
      $r = SR(af, onto)$ {$SR(ar, onto)$ gets the semantic relationship between af and $onto$}
      initialize σ, τ as 0
      if $onto$ equals af' **then**
        if r is "same_as" **then**
          $\sigma = 1$
        else if r is "kind_of" **then**
          $\sigma = 0.8$
        else if r is "phase_of" **then**
          $\sigma = 0.5$
        end if
      end if
      $weightforartifactonto = weightforartifactonto + \sigma$
      if $af.name$ equals $af'.name$ by string match **then**
        $\tau = 1$
      else if $af.name$ partly equals $af'.name$ by string match **then**
        $\tau = 0.5$
      end if
      $weightforartifactname = weightforartifactname + \tau$
    end for
  end for
  return $\gamma * weightforartifactonto + \delta * weightforartifactname$

Algorithm 4 Goal Annotation Algorithm for Case **B.III**

Require: weight parameters ϵ, μ
initialize value of $weightforoutartifactonto$, $weightforoutartifactname$ as 0 {Case **B.III**}
Array $oaf[]$ is a set of artifacts indirectly related to av through *has_Output* and *related_Artifact*
for each af in $oaf[]$ **do**
$onto = OD(af)$ {$OD(af)$ gets the domain ontological concept annotated to af}
for each target artifact af' of a g in goal ontology **do**
$r = SR(af, onto)$ {$SR(av, onto)$ gets the semantic relationship between af and $onto$}
initialize σ, τ as 0
if $onto$ equals af' **then**
if r is "same_as" **then**
$\sigma = 1$
else if r is "kind_of" **then**
$\sigma = 0.8$
else if r is "phase_of" **then**
$\sigma = 0.5$
end if
end if
if state s' of af' equals output o of av **then**
$weightforoutartifactonto = weightforoutartifactonto + \sigma + 1$
else
$weightforoutartifactonto = weightforoutartifactonto + \sigma$
end if
if $af.name$ equals $af'.name$ by string match **then**
$\tau = 1$
else if $af.name$ partly equals $af'.name$ by string match **then**
$\tau = 0.5$
end if
if state s' of af' equals output o of av **then**
$weightforoutartifactname = weightforoutartifactname + \tau + 1$
else
$weightforoutartifactname = weightforoutartifactname + \tau$
end if
end for
end for
return $\epsilon * weightforoutartifactonto + \mu * weightforoutartifactname$

Algorithm 5 Goal Annotation Algorithm for Case **C**

Require: weight parameters ε, ζ
  initialize value of $weightforactoronto$, $weightforactorname$ as 0
  $Arrayar[]$ is a set of actor-roles associated with av through *has_Actor-role*
  for each ar in $ar[]$ **do**
    $onto = OD(ar)$ {$OD(ar)$ gets the domain ontological concept annotated to ar}
    for each target role ar' of a g in goal ontology **do**
      $r = SR(ar, onto)$ {$SR(ar, onto)$ gets the semantic relationship between ar and $onto$}
      initialize σ, τ as 0
      if $onto$ equals ar' **then**
        if r is "same_as" **then**
          $\sigma = 1$
        else if r is "kind_of" **then**
          $\sigma = 0.8$
        else if r is "phase_of" **then**
          $\sigma = 0.5$
        end if
      end if
      $weightforactoronto = weightforactoronto + \sigma$
      if $ar.name$ equals $ar'.name$ by string match **then**
        $\tau = 1$
      else if $ar.name$ partly equals $ar'.name$ by string match **then**
        $\tau = 0.5$
      end if
      $weightforactorname = weightforartorname + \tau$
    end for
  end for
  return $\varepsilon * weightforartifactonto + \zeta * weightforartifactname$

Algorithm 6 Goal Annotation Algorithm for Case **D**

Require: weight parameters ε, ζ
initialize value of $weightforprecondition$, $weightforpostcondition$, $weightforexception$ as 0
$Arraypre[]$ is a set of preconditions associated with av through *has_Precondition*
initialize σ, τ as 0
for each *precon* in *pre*[] **do**
  for each target constraint c' of a g in goal ontology **do**
    if *precon.name* equals $c'.name$ by string match **then**
      $\tau = 1$
    else if *precon.name* partly equals $c'.name$ by string match **then**
      $\tau = 0.5$
    end if
    $weightforprecondition = weightforprecondition + \tau$
  end for
end for
$Arraypost[]$ is a set of postconditions associated with av through *has_Postcondition*

initialize σ, τ as 0
for each *postcon* in *post*[] **do**
  for each target constraint c' of a g in goal ontology **do**
    if *postcon.name* equals $c'.name$ by string match **then**
      $\tau = 1$
    else if *postcon.name* partly equals $c'.name$ by string match **then**
      $\tau = 0.5$
    end if
    $weightforpostcondition = weightforpostcondition + \tau$
  end for
end for
$Arrayexc[]$ is a set of exceptions associated with av through *has_Exception*
initialize σ, τ as 0
for each *exception* in *exc*[] **do**
  for each target constraint c' of a g in goal ontology **do**
    if *exception.name* equals $c'.name$ by string match **then**
      $\tau = 1$
    else if *exception.name* partly equals $c'.name$ by string match **then**
      $\tau = 0.5$
    end if
    $weightforexception = weightforexception + \tau$
  end for
end for
return $\eta * (weightforprecondition + weightforpostcondition + weightforexception)$

Appendix E

GUI of Pro-SEAT

The properties defined for annotating the profile are edit-able in the profile annotation UI (Figure E.1).

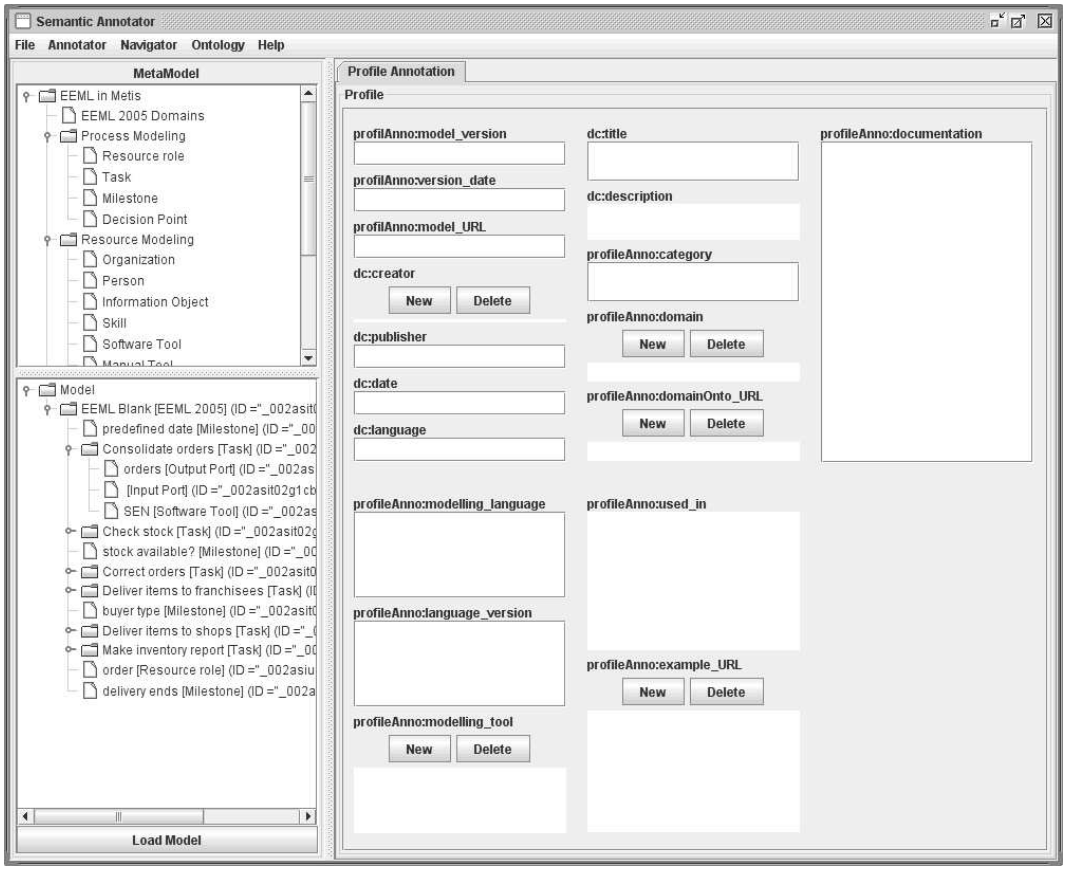


Figure E.1: Profile annotation UI in Pro-SEAT

Figure E.2 displays the main frame layout of the meta-model annotation UI. The tree view of the original process model is on the left, and on the right side of the annotation panel is the reference ontology browser (In this case, it displays the GPO

ontology). The meta-model annotation results are listed in the center of the frame. The manipulation of the meta-model annotation can be activated through the buttons beside the panel of the annotation result list.

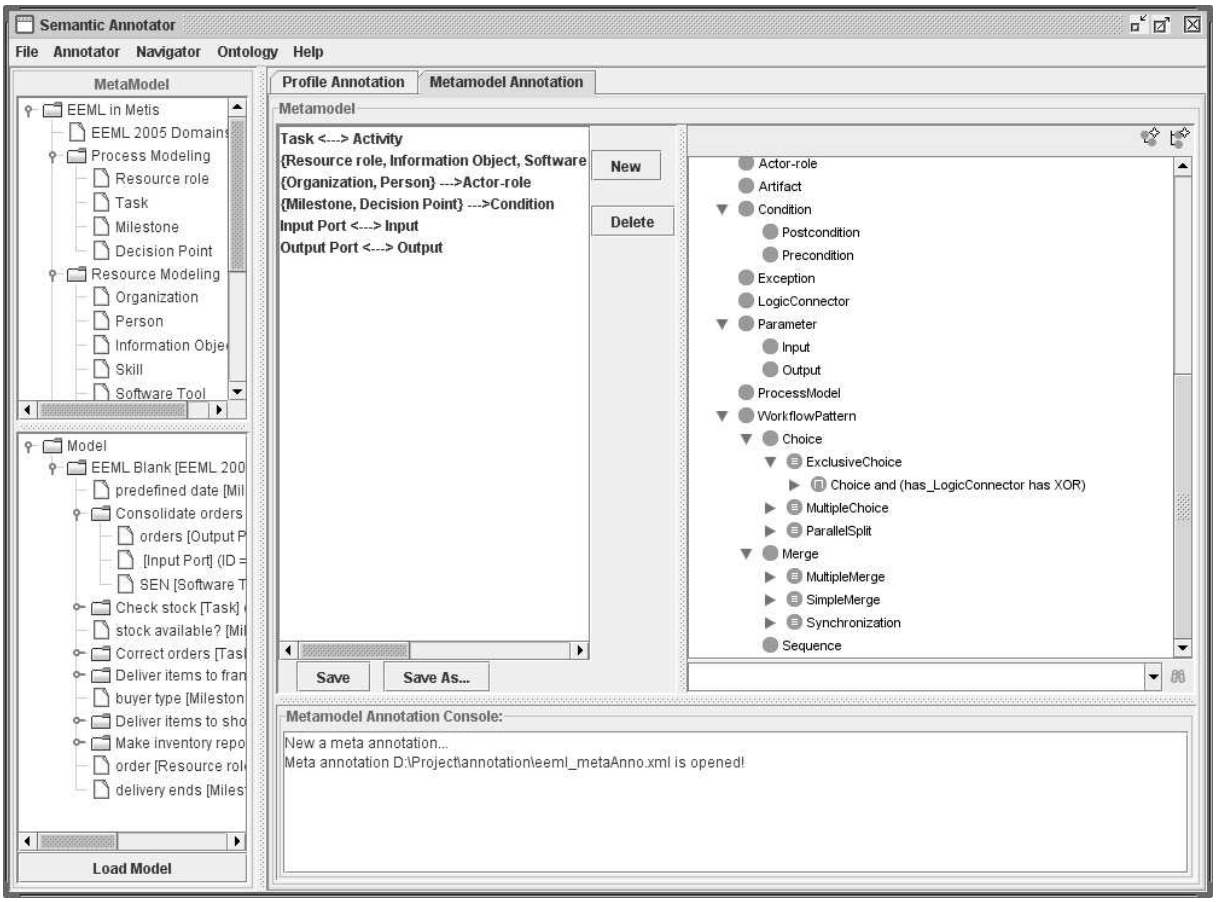


Figure E.2: Meta-model annotation UI in Pro-SEAT

Figure E.3 shows the activated dialogs in mapping the meta-model elements to the GPO ontology concepts.

In the model annotation UI (Figure E.4), a PSAM model can be generated for the model annotation from one meta-model annotation result. In a PSAM, the classes are GPO concepts and the model contents transformed from the original models are OWL instances of those GPO concepts. The model and the goal annotations are made to the instances of the generated PSAM models. The user can browse those instances by selecting a GPO concept in the annotation panel. For each model element/instance, there are a set of properties for editing the model annotation. Those properties are displayed between the panels of the model elements and the reference ontology. Some of the properties are the relationships defined in the GPO ontology (e.g. `has_subActivities`), and some are for linking the reference ontology (e.g. `same_as`). For those relationships of GPO, the values of the properties are still the model constructs/instances that can partly transferred from the original process model by the generation of PSAM. New values can be inserted or the values can be deleted by "New" and "Delete" buttons

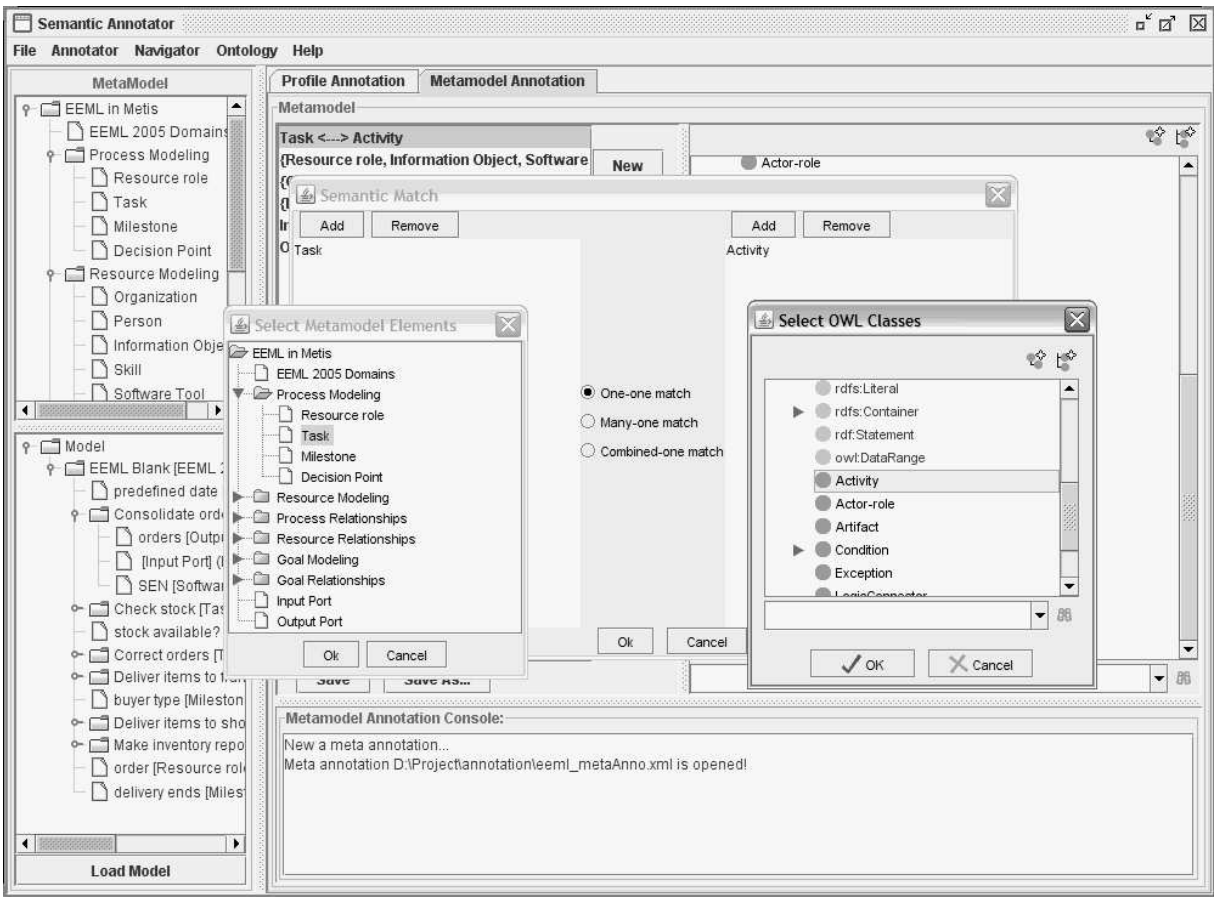


Figure E.3: Mapping meta-model to GPO in Pro-SEAT

for those properties. Since the value is also the annotatable model element/instance, "Annotate" button provides a short-path to browse this instance. For those links to the reference ontology, "Add" button can activate the ontology selection dialog and "Remove" button is for deleting the values of the property.

The goal annotation requires the generated PSAM model from the model annotation. Having the similar UI with the model annotation, the goal annotation properties are listed in the middle between the model elements and the reference ontology (Figure E.5). The target of the goal annotation is ***Activity*** and the sub-activity hierarchy is important for the automatic annotation, so that the model elements are listed as not the GPO instances but the model tree in the goal annotation UI. The manual annotation is to select the goal concept from the ontology selection dialog. When automating the goal annotation, a list of the goal options for the annotation is generated from the automatic goal annotation algorithm. The weights of the match are also displayed to indicate which goals have the higher priority for this annotation. Figure E.6 illustrates the results of the automatic goal annotation.

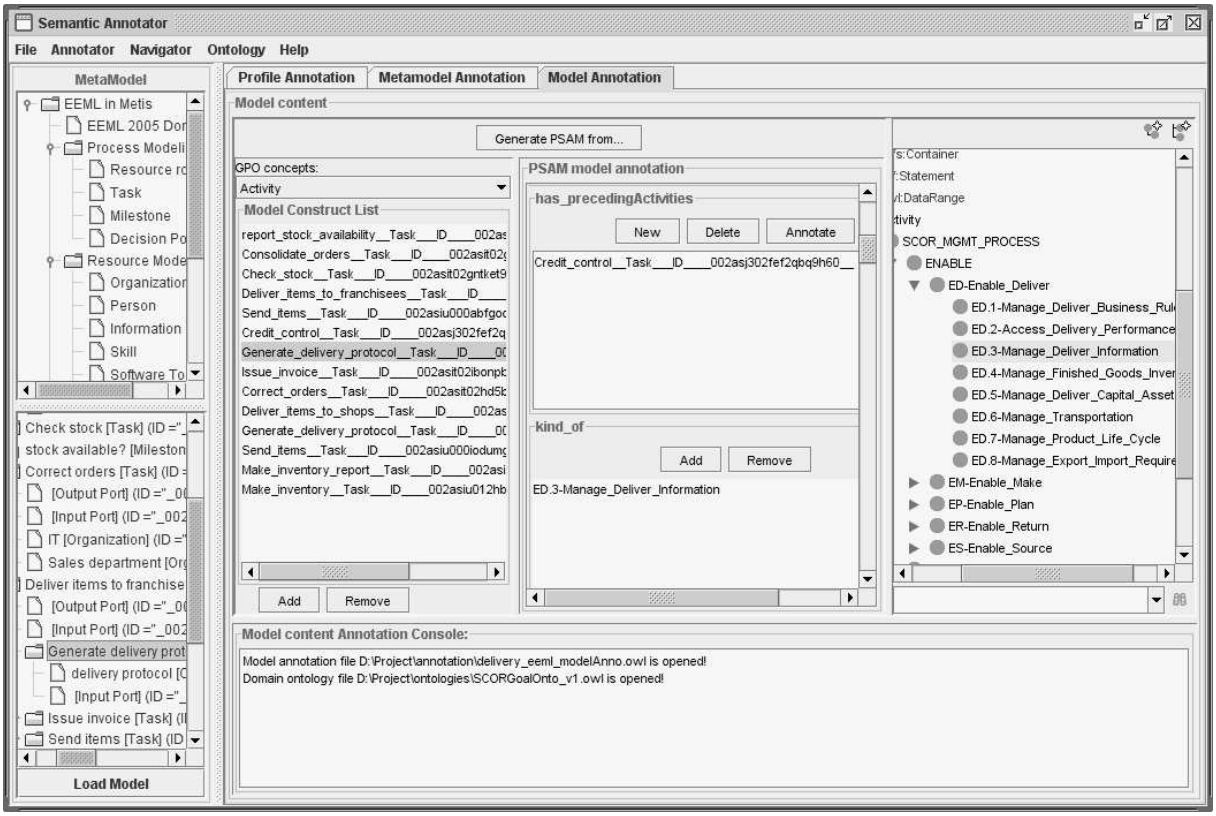


Figure E.4: Model annotation UI in Pro-SEAT

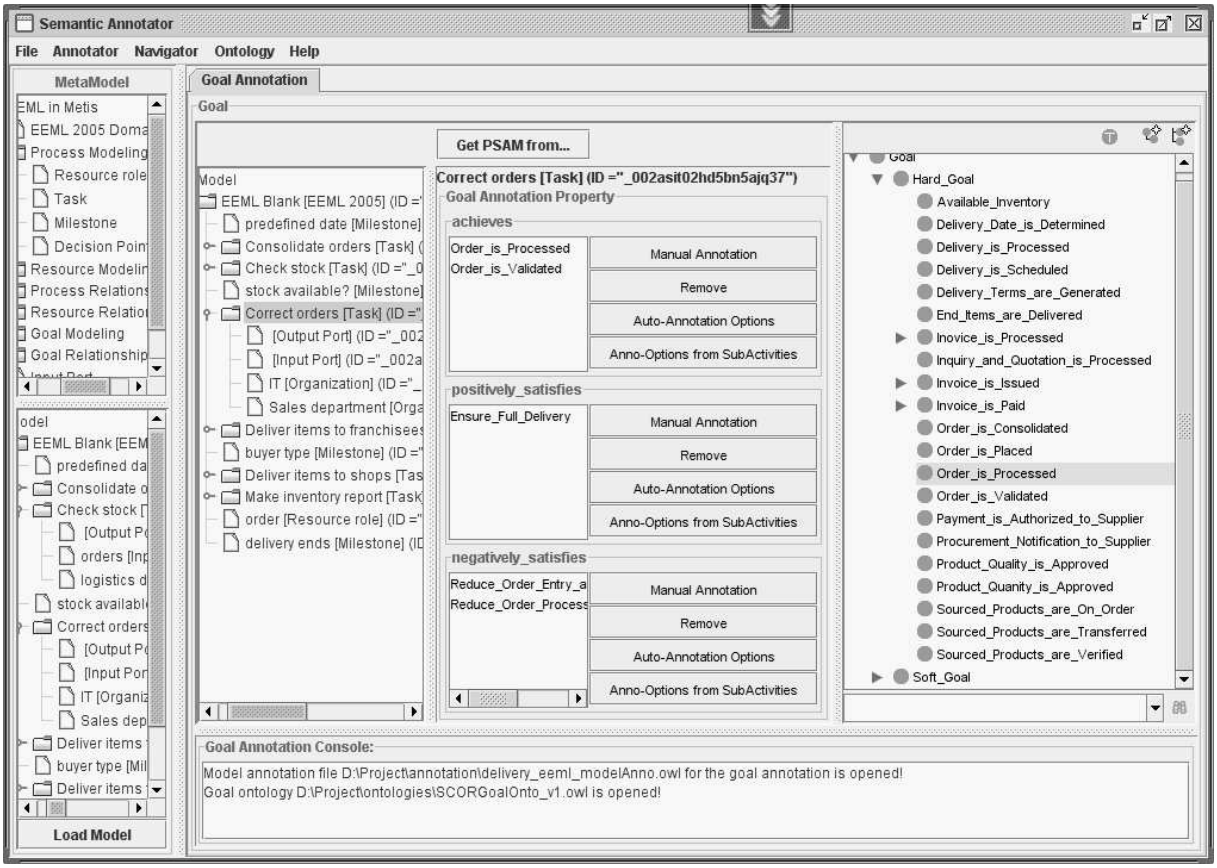


Figure E.5: Goal annotation UI in Pro-SEAT

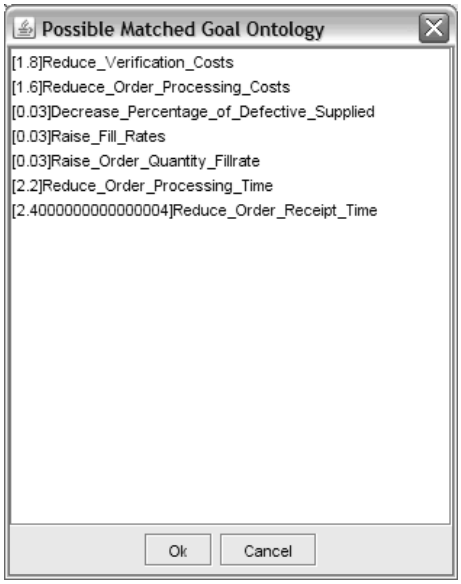


Figure E.6: Automatic goal annotation in Pro-SEAT

Appendix F

Analysis of the Annotation and the Integration Application in Exemplar Studies

In this chapter, we provide analysis details of exemplar studies. Section F.1 describes how SCOR ontology is annotated to PM_A, PM_{B1} and PM_{B2} with our semantic annotation approach. Based on the annotation results, an analysis of the integration application for the applicability of the semantic annotation approach is depicted in section F.2.

F.1 Semantic Annotation Based on SCOR ontology

Parts of the annotation results of the exemplar studies are displayed in Table 7.3, Table 7.4 and Table 7.5. Ten representive ***Activities*** out of 33 instances in PM_A are listed in Table 7.4. For the model annotation, `same_as` is applied when the ***Activity*** instance undertakes the same task as the SCOR process element. "Transportation planning" in PM_A does actually accomplish the carriers selection and shipments rating (`D1.7`), no more and no less work than that. When one ***Activity*** instance specifies the way of doing of a certain SCOR process element, `kind_of` can be used. "Credit control" is one way of managing the deliver capital assets (`ED.5`) in the context of PM_A. Usually the granularity of the Activity instances is smaller than the SCOR level 3 process element, i.e. they are decomposed parts of SCOR process elements. Those ***Activities*** are therefore `phase_of` the reference ontology concepts. Both "Client RFQ processing" and "Client quotation processing" are two phases of `D1.1 Process Inquiry and Quote`. As the goal annotation results, they are not obvious to see the inherent link between the ***Activity*** instance and the goal ontology in the table. The automatic goal annotation algorithm deduces the optional goal ontology concepts which have the features relevant to the model fragment of the Activity instance. But it does not mean that all the deduced goals are fulfilled by the Activity, the options are returned with the weights. Generally the higher weight indicates that the returned goal is more relevant to the Activity instance. Besides, the annotator will make the choices based on the context of process models and manual annotation is also necessary in

some cases. For the example of "Create delivery", the goal algorithm deduces two hard goals `Delivery Date is Determined` with weight '1.0' and `Delivery Date is Determined` with weight '0.06'. `Delivery Date is Determined` is not selected because of low weight. `Delivery is Processed` is further checked in the goal ontology and its parts `Delivery is Scheduled` and `Delivery Terms are Generated` are considered more precise than `Delivery is Processed` in the goal annotation of "Create delivery". Nine soft goals are also returned but only two goals with weight '1.8' (`Ensure Full Delivery` and `Ensure Full Shipment`) are selected by an annotator.

In PM_{B1}, "get the order to supplier" can be regarded as a phase of the activity ontology `Schedule Product Deliveries` from SCOR. Because there are three kinds of items receiving, they are respectively checked in sub-Activities of the activity "check items". Therefore, three sub-Activities all achieves `Sourced Product are Verified`. The procedure of "check imported items" is a little more complicated compared with the other two kinds of items receiving because it includes issuing deficit protocols and an invoice to insurance company. "Check imported items" are depicted with a serial of sub-Activities such as "issue the deficit protocol", "issue an invoice to insurance company", etc. Those negatively satisfy the soft goal `Reduce Verification Costs`. The consignment can simplify the check and item receiving procedure, so it positively satisfies `Reduce Verification Costs`. Issuing invoices are steps of "Authorize Supplier Payment", and "issue an export invoice" are to authorize payment to suppliers. The soft goal of `Process Invoice Without Error` will be risked by the invoice issue procedures. In the context of the TelCo use case, "store items to Orbit" is the way of `Transfer Product` and the result of the procedure is an `Available Inventory`.

PM_A starts the process from the order inquiry from the client, whilst PM_{B2} begins with the existing orders but the orders are managed and edited during the delivery process. It is hereby observed that the sequence of the ***Activities*** of PM_{B2} does not comply with the sequence of the SCOR `DELIVER`. The first Activity "Consolidate orders" accomplishes the same task as `D1.4 Consolidate Orders`. As the descriptions of the TelCo case, the "Consolidate orders" is executed by a computational software SEN which can `Reduce Order Processing Costs` and `Reduce Order Processing Time`. "Check stock" in PM_{B2} is more general compared with "Check delivery items" and "Check availability of delivery items" in PM_A so that no specific goals are annotated to "Check stock". Although both PM_A and PM_{B2} have the Activity "Credit control" and they are both `phase_of D1.2 Receive Enter and Validate Order`, they are not exactly same. "Credit control" in PM_A deals with the capital assets but in PM_{B2} it is just `kind_of ED.3 Manage Deliver Information`. Delivery protocol is treated as `Delivery Terms` and "Generate delivery protocol" of PM_{B2} is similar to "Create delivery" of PM_A. They are also the way of `ED.3 Manage Deliver Information`. However, the delivery protocol in PM_{B2} is mainly focus on the agreed delivery date so that `Improve Deliver to Customer Delivery to Date Performance` and `Improve Deliver to Customer On Time Delivery Performance` are selected as the goal annotation for "Generate delivery protocol". "Send items" serves in both "Deliver items to franchisees" and "Deliver items to shops" and should be specified as `kind_of D1.12 Ship Product` if they are using different ways to ship product. The items delivered to shops are retail product so "Deliver items to shops" is `phase_of D4 Deliver Retail Product` not `D1 Deliver Stocked Product`as "Deliver items to franchisees". Since

the shops are local shops, it usually takes less time to ship the products so that it `positively_satisfies Reduce Order Shipment Time`.

After annotating the low level activity elements, the goal contributions can be calculated to the upper level activities. Taking the example of the composite activity "Check items", we have annotated its component activities with hard goals and soft goals. "Check imported items" negatively satisfies `Reduce Verification Costs` and "check consignment items" positively satisfies `Reduce Verification Costs`, so the effects are counteracted for the composite activity "check items" if we apply the ***simple contribution calculation rules***. The hard goals "Sourced Products are Verified", "Procurement Notification to Supplier" and "Product Quantity is Approved" which are annotated to "check imported items", "check consignment items" and "check items from local suppliers" are simply passed to "check items". Applying the same rules, the goals related to the whole process model are specified through the goal annotation.

F.2 Integration Application Based On Semantic Annotation

An integration application in the exemplar studies is to integrate delivery processes of Enterprise A and B. We deploy this application based on semantic annotation results of process models.

The following steps have been undertaken for the integration application:

1. Running **QRule-Activity-achievesHardGoal** for RE2.4 to find process model fragments achieving the integration goals.

2. Using ontology as semantic mediator, SWRL queries and rules are executed for RE4 to find semantic relationships between ***Activity***, ***Artifact*** and ***Actor-role*** in those model fragments resulted from step 1.

3. Based on semantic relationships between model fragments, listing some possible integration paths.

4. Applying SWRL queries and rules **QRule-Activity-hasSucceedingActivities** and **QRule-Activity-hasPrecedingActivities** for RE1 and **QRule-Activity-hasPrecedingActivities-hasSubActivity** for RE3.2 to rearrange the sequence of integrating process model fragments.

5. Checking the results from step 4 with activity sequences in the SCOR domain ontology.

6. Adjusting the sequence and hierarchy of integrating activities according to the integration context.

7. Build output-input flow by checking ***Input*** and ***Output*** of activities through running **QRule-Activity-Input-mappedto** and **QRule-Activity-Output-mappedto** for RE2.2. The step can be used to find the missing activities and also re-arrange the sequence of activities based on the output-input flow.

Firstly, the goals `"Delivery_is_Processed"` and `"Invoice_to_Customer_is_Processed"` are chosen from the goal ontology as the integration goal. Running **QRule-Activity-achievesHardGoal** for RE2.4 on annotated models from both enterprises. There is no result from PM_A, PM_{B1} and PM_{B2} which is directly related to the two goals. However, reasoning from the goal ontology, `"Delivery_is_Processed"` has sub-goals (`has_parts`) `"Delivery_is_Scheduled"`, `"Delivery_Terms_are_Generated"`, and `"End_Items_are_Delivered"`. `"Invoice_to_Customer_is_Processed"` has sub-goals `"Invoice_to_Customer_is_Issued"` and `"Invoice_to_Customer_is_Paid"`. By running the queries about these sub-goals, we are aware that such knowledge is distributed in PM_A and PM_{B2} and mostly related to "Delivering_Processing" in PM_A and "Deliver_items_to_franchisee" and "Deliver_items_to_shops" in PM_{B2}.

Another integration concern in this application is the integration of `"Customer"` of two enterprises. In PM_A the BPMN Swimlane "client" is `same_as` the ontology concept `"Customer"` whilst in PM_{B2} the EEML Organization "shop" and "franchisee" are both `kind_of "Customer"`. When transforming or integrating two models, "shop" and "franchisee" can be modeled as sub-class of "client" in PM_A. The identical and the unique parts of "Deliver_items_to_franchisee" and "Deliver_items_to_shops" can be modeled as sub-Activities of "Delivering_Processing" in PM_A respectively. Moreover, we should check all the Activities relevant to "shop" or "franchisee" in PM_{B2} and the Activities relevant to "client" in PM_A. In the exemplar studies, the Activity "Send_inquiry", "Send_quotation", "Receive_delivery" in PM_A and the Activity "Deliver_items_to_shops" and "Deliver_items_to_franchisees" in PM_{B2} are returned in the query results. To check the underlying semantic relationships between those Activities, we can run **QRule-Activity-phaseof**, **QRule-Activity-kindof** and **QRule-Activity-sameas** to find out how those Activities mapped to the SCOR ontology. Both "Deliver_items_to_shops" and "Deliver_items_to_franchisees" in PM_{B2} are `phase_of D1-Deliver_Stocked_Product`, while in PM_A "Send_quotation" and "Send_inquiry" are `phase_of D1.1-Process_Inquiry_and_Quote` and "Receive_delivery" is `kind_of D1.13-Receive_and_Verify_Product`. It is obvious that the two Activities in PM_{B2} are too general (mapped to a process category of the SCOR level 2), so that we can check if they have sub-Activities by running **QRule-Activity-subActivity**. If there is no sub-Activities, the Activities in PM_A can be considered as the refined sub-Activities for PM_{B2} because `D1.1` and `D1.13` are sub-Activities of `D1` according to the SCOR ontology. If there are any sub-Activities, further check of those sub-Activities is to be deployed. Four sub-Activities are returned for "Deliver_items_to_franchisees": "Generate_delivery_protocol", "Credit_control", "Issue_invoice" and "Send_items". When checking the semantic mapping between those sub-Activities and the SCOR ontology, "Send_items" is `kind_of D1.12-Ship_Product`, "Credit_control" is `phase_of "D1.2-Receive_Enter_and_Validate_Order"` and "Issue_invoice" is `phase_of "D1.15-Invoice"`. By filling RE4.5, a possible integration path could be

$$\textit{"Send\_inquiry"}(PM_A) \rightarrow \textit{"Send\_quotation"}(PM_A) \rightarrow \textit{"Credit\_control"}(PM_{B2}) \rightarrow \textit{"Delivering\_Processing"}(PM_A)\{\textit{"Deliver\_items\_to\_franchisee"}(PM_{B2}\}; \textit{"Deliver\_items\_to\_shops"}(PM_{B2})\} \rightarrow \textit{"Ship\_items"}(PM_{B2}) \rightarrow$$

$"Receive\_delivery"(PM_A) \rightarrow "Issue\_invoice"(PM_{B2})$.

Certainly that is not a complete and fine integration. The succeeding Activities of "Send_quotation" in PM_A and the preceding Activities of "Credit_control" in PM_{B2} should be checked by **QRule-Activity-hasSucceedingActivities** and **QRule-Activity-hasPrecedingActivities** for RE1. Because "Credit_control" is a sub-Activity in PM_{B2}, the **QRule-Activity-hasPrecedingActivities-hasSubActivity** for RE3.2 should be executed as well. The returned succeeding and preceding Activities are then checked with the reference ontology based on the annotation. "Client_quotation_processing" and "Standard_order_processing" (succeeding Activities of "Send_quotation") in PM_A are mapped to `"D1.1"` and `"D1.2"` respectively. The sequence of "Check_stock" and "Correct_orders" (preceding Activities of "Send_quotation") in PM_{B2} are found not consistent to the SCOR ontology because "Check_stock" is `phase_of "D1.3-Reserve_Inventory_and_Determine_Date"` and "Correct_orders" is `phase_of "D1.2"`. The decision of adapting PM_{B2} to SCOR sequence in the integration model is made. Compared with "Standard_order_processing" in PM_A, "Correct_orders" has the same ontology reference `"D1.2"` and the same Actor-role "Sales". Therefore, "Correct_orders" can be adapted as a subActivity of "Standard_order_processing". Searching the Activities referencing `D1.3` in PM_A, the Activity "Check_delivery_items" is found so that it is considered to be merged with "Check_stock" in PM_{B2}. Analogously, the two Activities "Credit_control" in PM_A and in PM_{B2} are merged into one, and "Ship_items" in PM_{B2} is merged into "Transportation_processing" in PM_A just because they share the same ontology references.

$"Send\_inquiry"(PM_A) \rightarrow "Send\_quotation"(PM_A) \rightarrow$
$"Client\_quotation\_processing"(PM_A) \rightarrow "Standard\_order\_processing"(PM_A)$
$\{...;"Correct\_orders"(PM_{B2})\ ...\} \rightarrow "Credit\_control"(PM_A/PM_{B2}) \rightarrow$
$"Delivering\_Processing"(PM_A)\{...;"Check\_stock"(PM_{B2});...\}$
$\rightarrow "Transportation\_processing"(PM_A)/"Ship\_items"(PM_{B2})$
$\rightarrow "Receive\_delivery"(PM_A) \rightarrow "Issue\_invoice"(PM_{B2})$.

Checking the succeeding Activities of "Credit_control" in PM_A, a serials of delivery checking takes place before the Activity "Create_delivery" produces an output "delivery" which is mapped to `"Customer_Delivery_Terms"`. However, in PM_{B2} only one Activity "Generate_delivery_protocol" following "Credit_control" and has the output of "delivery_protocol". "Delivery_protocol" is annotated with `"Customer_Delivery_Terms"`. Such knowledge is got from running **QRule-Activity-Output-mappedto**. Hence the integration model can take the functions defined in PM_A to specify "Generate_delivery_protocol" in PM_{B2}. Besides, the input of Activity "Issue_invoice" is checked through **QRule-Activity-Input-mappedto** and it is also mapped to "Customer_Delivery_Terms". According to **Case 1** of RE4.5, "Create delivery" in PM_A can be followed by "Issue_invoice" of PM_{B2} in the integration model. Since there is no strict sequence requirement between "Transportation_processing" and "Issue_invoice", the two Activities can proceed parallely.

$$"Send\_inquiry"(PM_A) \rightarrow "Send\_quotation"(PM_A) \rightarrow$$
$$"Client\_quotation\_processing"(PM_A) \rightarrow "Standard\_order\_processing"(PM_A)$$
$$\{"Correct\_orders"(PM_{B2})\} \rightarrow "Credit\_control"(PM_{B2}) \rightarrow$$
$$"Delivering\_Processing"(PM_A)\{...;"Check\_stock"(PM_{B2});...;"Create\_delivery"\}$$
$$\rightarrow "Issue\_invoice"(PM_{B2})|("Transportation\_processing"(PM_A)/"Ship\_items"(PM_{B2})$$
$$\rightarrow "Receive\_delivery"(PM_A)).$$

Appendix G

Schema of PSAM Model and SWRL Rules

G.1 PSAM Model in OWL

Parts of the PSAM model are exemplified as follows. The complete model is available at http://www.idi.ntnu.no/~yunl/schema/GPOmetaOnto.owl

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns="http://www.owl-ontologies.com/GPO.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://www.owl-ontologies.com/GPO.owl">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://protege.stanford.edu/plugins/owl/protege"/>
  </owl:Ontology>

  <owl:Class rdf:ID="Actor-role">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="same_as"/>
        </owl:onProperty>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="kind_of"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="member_of"/>
        </owl:onProperty>
```

```
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#name"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
</owl:Class>

<owl:Class rdf:ID="Activity">
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#name"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#same_as"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#kind_of"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="phase_of"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:ID="Artifact">
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#name"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
```

```
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:ObjectProperty rdf:ID="has_Artifact">
  <rdfs:range rdf:resource="#Artifact"/>
  <rdfs:domain rdf:resource="#Activity"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="has_Postcondition">
  <rdfs:range rdf:resource="#Postcondition"/>
  <rdfs:domain rdf:resource="#Activity"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="way_of">
  <rdfs:range rdf:resource="#Activity"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="has_Input">
  <rdfs:domain rdf:resource="#Activity"/>
  <rdfs:range rdf:resource="#Input"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="related_Parameter">
  <rdfs:domain rdf:resource="#Condition"/>
  <rdfs:range rdf:resource="#Parameter"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="related_Actor">
  <rdfs:range rdf:resource="#Actor-role"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="has_succeedingActivities">
  <rdfs:domain rdf:resource="#Activity"/>
  <rdfs:range rdf:resource="#Activity"/>
</owl:ObjectProperty>

<owl:DatatypeProperty rdf:ID="alternative_name">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Activity"/>
        <owl:Class rdf:about="#Actor-role"/>
        <owl:Class rdf:about="#Artifact"/>
        <owl:Class rdf:about="#Exception"/>
        <owl:Class rdf:about="#Parameter"/>
        <owl:Class rdf:about="#Condition"/>
        <owl:Class rdf:about="#WorkflowPattern"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#kind_of">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Activity"/>
        <owl:Class rdf:about="#Actor-role"/>
        <owl:Class rdf:about="#Artifact"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#anyURI"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="filename">
  <rdfs:domain rdf:resource="#ProcessModel"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>

<owl:FunctionalProperty rdf:ID="id">
```

```
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Activity"/>
          <owl:Class rdf:about="#Actor-role"/>
          <owl:Class rdf:about="#Artifact"/>
          <owl:Class rdf:about="#Exception"/>
          <owl:Class rdf:about="#Parameter"/>
          <owl:Class rdf:about="#Condition"/>
          <owl:Class rdf:about="#WorkflowPattern"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#ID"/>
  </owl:FunctionalProperty>
</rdf:RDF>
```

G.2 Rules Definition in SWRL

Some rules defined in SWRL are exemplified as follows. The complete definition of the SWRL rules is available at http://www.idi.ntnu.no/~yunl/schema/swrl.txt

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns="http://www.owl-ontologies.com/GPO.owl#"
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    xmlns:swrlx="http://swrl.stanford.edu/ontologies/built-ins/3.3/swrlx.owl#"
    xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
    xmlns:query="http://swrl.stanford.edu/ontologies/built-ins/3.3/query.owl#"
    xmlns:temporal="http://swrl.stanford.edu/ontologies/built-ins/3.3/temporal.owl#"
    xmlns:tbox="http://swrl.stanford.edu/ontologies/built-ins/3.3/tbox.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:abox="http://swrl.stanford.edu/ontologies/built-ins/3.3/abox.owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:swrl="http://www.w3.org/2003/11/swrl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:swrla="http://swrl.stanford.edu/ontologies/3.3/swrla.owl#"
    xml:base="http://www.owl-ontologies.com/GPO.owl">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://swrl.stanford.edu/ontologies/built-ins/3.3/tbox.owl"/>
    <owl:imports rdf:resource="http://swrl.stanford.edu/ontologies/3.3/swrla.owl"/>
    <owl:imports rdf:resource="http://swrl.stanford.edu/ontologies/built-ins/3.3/abox.owl"/>
    <owl:imports rdf:resource="http://protege.stanford.edu/plugins/owl/protege"/>
    <owl:imports rdf:resource="http://swrl.stanford.edu/ontologies/built-ins/3.3/query.owl"/>
    <owl:imports rdf:resource="http://www.w3.org/2003/11/swrlb"/>
    <owl:imports rdf:resource="http://www.w3.org/2003/11/swrl"/>
    <owl:imports rdf:resource="http://swrl.stanford.edu/ontologies/built-ins/3.3/swrlx.owl"/>
    <owl:imports rdf:resource="http://swrl.stanford.edu/ontologies/built-ins/3.3/temporal.owl"/>
  </owl:Ontology>

 <swrl:Imp rdf:ID="QRule-Activity-Input-mappedto">
    <swrl:body>
      <swrl:AtomList>
        <rdf:rest>
          <swrl:AtomList>
            <rdf:first>
              <swrl:IndividualPropertyAtom>
                <swrl:argument2>
                  <swrl:Variable rdf:ID="y"/>
                </swrl:argument2>
                <swrl:propertyPredicate rdf:resource="#has_Input"/>
```

```
            <swrl:argument1>
              <swrl:Variable rdf:ID="x"/>
            </swrl:argument1>
          </swrl:IndividualPropertyAtom>
        </rdf:first>
        <rdf:rest>
          <swrl:AtomList>
            <rdf:first>
              <swrl:DatavaluedPropertyAtom>
                <swrl:argument1 rdf:resource="#y"/>
                <swrl:propertyPredicate rdf:resource="#mapped_to"/>
                <swrl:argument2>
                  <swrl:Variable rdf:ID="z"/>
                </swrl:argument2>
              </swrl:DatavaluedPropertyAtom>
            </rdf:first>
            <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
          </swrl:AtomList>
        </rdf:rest>
      </swrl:AtomList>
    </rdf:rest>
    <rdf:first>
      <swrl:ClassAtom>
        <swrl:argument1 rdf:resource="#x"/>
        <swrl:classPredicate rdf:resource="#Activity"/>
      </swrl:ClassAtom>
    </rdf:first>
  </swrl:AtomList>
</swrl:body>
<swrl:head>
  <swrl:AtomList>
    <rdf:first>
      <swrl:BuiltinAtom>
        <swrl:arguments>
          <rdf:List>
            <rdf:first rdf:resource="#x"/>
            <rdf:rest>
              <rdf:List>
                <rdf:first rdf:resource="#y"/>
                <rdf:rest>
                  <rdf:List>
                    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                    <rdf:first rdf:resource="#z"/>
                  </rdf:List>
                </rdf:rest>
              </rdf:List>
            </rdf:rest>
          </rdf:List>
        </swrl:arguments>
        <swrl:builtin rdf:resource="http://swrl.stanford.edu/ontologies/built-ins/3.3/
        query.owl#select"/>
      </swrl:BuiltinAtom>
    </rdf:first>
    <rdf:rest>
      <swrl:AtomList>
        <rdf:first>
          <swrl:BuiltinAtom>
            <swrl:arguments>
              <rdf:List>
                <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                <rdf:first rdf:resource="#z"/>
              </rdf:List>
            </swrl:arguments>
            <swrl:builtin rdf:resource="http://swrl.stanford.edu/ontologies/built-ins/3.3/
            query.owl#orderBy"/>
```

```
            </swrl:BuiltinAtom>
          </rdf:first>
          <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
        </swrl:AtomList>
      </rdf:rest>
    </swrl:AtomList>
  </swrl:head>
</swrl:Imp>

 <swrl:Imp rdf:ID="QRule-Activity-achievesHardGoal">
    <swrl:body>
      <swrl:AtomList>
        <rdf:first>
          <swrl:ClassAtom>
            <swrl:classPredicate rdf:resource="#Activity"/>
            <swrl:argument1 rdf:resource="#x"/>
          </swrl:ClassAtom>
        </rdf:first>
        <rdf:rest>
          <swrl:AtomList>
            <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
            <rdf:first>
              <swrl:DatavaluedPropertyAtom>
                <swrl:propertyPredicate rdf:resource="#achieves"/>
                <swrl:argument1 rdf:resource="#x"/>
                <swrl:argument2 rdf:resource="#y"/>
              </swrl:DatavaluedPropertyAtom>
            </rdf:first>
          </swrl:AtomList>
        </rdf:rest>
      </swrl:AtomList>
    </swrl:body>
    <swrl:head>
      <swrl:AtomList>
        <rdf:first>
          <swrl:BuiltinAtom>
            <swrl:arguments>
              <rdf:List>
                <rdf:first rdf:resource="#x"/>
                <rdf:rest>
                  <rdf:List>
                    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                    <rdf:first rdf:resource="#y"/>
                  </rdf:List>
                </rdf:rest>
              </rdf:List>
            </swrl:arguments>
            <swrl:builtin rdf:resource="http://swrl.stanford.edu/ontologies/built-ins/3.3/
            query.owl#select"/>
          </swrl:BuiltinAtom>
        </rdf:first>
        <rdf:rest>
          <swrl:AtomList>
            <rdf:first>
              <swrl:BuiltinAtom>
                <swrl:arguments>
                  <rdf:List>
                    <rdf:first rdf:resource="#y"/>
                    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                  </rdf:List>
                </swrl:arguments>
                <swrl:builtin rdf:resource="http://swrl.stanford.edu/ontologies/built-ins/3.3/
                query.owl#orderBy"/>
              </swrl:BuiltinAtom>
            </rdf:first>
```

```
          <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
        </swrl:AtomList>
      </rdf:rest>
    </swrl:AtomList>
  </swrl:head>
</swrl:Imp>

<swrl:Imp rdf:ID="IRule-Activity-subActivity-hasActor">
  <swrl:body>
    <swrl:AtomList>
      <rdf:first>
        <swrl:ClassAtom>
          <swrl:classPredicate rdf:resource="#Activity"/>
          <swrl:argument1 rdf:resource="#x"/>
        </swrl:ClassAtom>
      </rdf:first>
      <rdf:rest>
        <swrl:AtomList>
          <rdf:rest>
            <swrl:AtomList>
              <rdf:first>
                <swrl:IndividualPropertyAtom>
                  <swrl:argument2 rdf:resource="#z"/>
                  <swrl:argument1 rdf:resource="#y"/>
                  <swrl:propertyPredicate rdf:resource="#has_Actor-role"/>
                </swrl:IndividualPropertyAtom>
              </rdf:first>
              <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
            </swrl:AtomList>
          </rdf:rest>
          <rdf:first>
            <swrl:IndividualPropertyAtom>
              <swrl:argument1 rdf:resource="#x"/>
              <swrl:propertyPredicate rdf:resource="#has_subActivity"/>
              <swrl:argument2 rdf:resource="#y"/>
            </swrl:IndividualPropertyAtom>
          </rdf:first>
        </swrl:AtomList>
      </rdf:rest>
    </swrl:AtomList>
  </swrl:body>
  <swrla:isEnabled rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >true</swrla:isEnabled>
  <swrl:head>
    <swrl:AtomList>
      <rdf:rest>
        <swrl:AtomList>
          <rdf:first>
            <swrl:IndividualPropertyAtom>
              <swrl:propertyPredicate rdf:resource="#has_Actor-role"/>
              <swrl:argument1 rdf:resource="#x"/>
              <swrl:argument2 rdf:resource="#z"/>
            </swrl:IndividualPropertyAtom>
          </rdf:first>
          <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
        </swrl:AtomList>
      </rdf:rest>
      <rdf:first>
        <swrl:ClassAtom>
          <swrl:argument1 rdf:resource="#x"/>
          <swrl:classPredicate rdf:resource="#Activity"/>
        </swrl:ClassAtom>
      </rdf:first>
    </swrl:AtomList>
  </swrl:head>
```

```
    </swrl:Imp>
</rdf:RDF>
```

Appendix H

Annotation Results in Exemplar Studies — PSAM Instances in OWL

H.1 Annotation of PM_A

Parts of the annotation results of PM_A are illustrated as follows. The whole OWL file is available at http://www.idi.ntnu.no/~yunl/schema/PMA_bmp_modelAnno.owl

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns="http://www.owl-ontologies.com/GPO.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://www.owl-ontologies.com/GPO.owl">
  ...
  <Activity rdf:ID="Check_delivery_items__Logical_Process___ID____
    002asmm019p9am10kf6g__">
    <has_Output>
      <Output rdf:ID="item_availability__Output___ID____
        002asmm01aig8619d5uc__">
        <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >item availability</name>
        <mapped_to rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Inventory_Availability</mapped_to>
        <model_fragment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >_002asmm01aig8619d5uc</model_fragment>
        <related_Artifact>
          <Artifact rdf:ID="stock__Data_Object___ID____002asmm01e1g3iu6rr7q__">
            <model_fragment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >_002asmm01e1g3iu6rr7q</model_fragment>
            <same_as rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >Stock</same_as>
            <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >stock</name>
          </Artifact>
        </related_Artifact>
        <related_Artifact>
          <Artifact rdf:ID="delivery_items__Data_Object___ID____
```

```
            002asmm01dkqc62v0gnf__">
            <kind_of rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >Goods</kind_of>
            <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >delivery items</name>
            <model_fragment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >_002asmm01dkqc62v0gnf</model_fragment>
          </Artifact>
        </related_Artifact>
      </Output>
    </has_Output>
    <positively_satisfies rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Improve_Delivery_Performance</positively_satisfies>
    <positively_satisfies rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Raise_Order_Quantity_Fillrate</positively_satisfies>
    <phase_of rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >D1.3-Reserve_Inventory_and_Determine_Delivery_Date</phase_of>
    <has_Postcondition>
      <Condition rdf:ID="Is_items_in_stock___Gateway___ID____
        002asmm019tqpgr5teji__">
        <model_fragment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >_002asmm019tqpgr5teji</model_fragment>
        <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Is items in stock?</name>
        <alternative_name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >items are stored in the inventory</alternative_name>
      </Condition>
    </has_Postcondition>
    <has_Artifact rdf:resource="#stock__Data_Object___ID____
    002asmm01e1g3iu6rr7q__"/>

    <has_succeedingActivities rdf:resource="#Reject_the_delivery_items__Logical_Process
    ___ID____002asmm01b7291tj043o__"/>
    <has_Actor-role rdf:resource="#logistics_department__Horizontal_Swimlane___ID____
    002asmm00p3r9tn7e6c4__"/>
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Check delivery items</name>
    <model_fragment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >_002asmm019p9am10kf6g</model_fragment>
    <has_precedingActivities rdf:resource="#Determine_or_transfer_delivery_route
    __Logical_Process___ID____002asmm019imrma0mdf5__"/>
  </Activity>
  ...
  <ProcessModel rdf:ID="D__Project_models_sales_logistics_bpm.kmv">
    <filename rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >D:\Project\models\sales_logistics_bpm.kmv</filename>
  </ProcessModel>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.2.1, Build 365)  http://protege.stanford.edu -->
```

H.2 Annotation of PM_{B1}

Parts of the annotation results of PM_{B1} are illustrated as follows. The whole OWL file is available at http://www.idi.ntnu.no/~yunl/schema/PMB1_eeml_modelAnno.owl

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns="http://www.owl-ontologies.com/GPO.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
```

```
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://www.owl-ontologies.com/GPO.owl">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://protege.stanford.edu/plugins/owl/protege"/>
  </owl:Ontology>
  ...
  <Activity rdf:ID="make_a_report_on_expected_quantities__Task___ID____
  002ashn00tq5gtqi5ps3__">
    <has_Input>
      <Input rdf:ID="__Input_Port___ID____002ashn00tq5hh6820mc__">
        <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        ></name>
        <related_Artifact>
          <Artifact rdf:ID="order__Resource_role___ID____002ashm029nhieleknj3__">
            <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >order</name>
            <same_as rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >Order</same_as>
            <model_fragment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >_002ashm029nhieleknj3</model_fragment>
          </Artifact>
        </related_Artifact>
        <model_fragment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >_002ashn00tq5hh6820mc</model_fragment>
        <mapped_to rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Receipt_Verification</mapped_to>
      </Input>
    </has_Input>
    <has_Output>
      <Output rdf:ID="__Output_Port___ID____002ashn00tq5gthg1l9t__">
        <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        ></name>
        <model_fragment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >_002ashn00tq5gthg1l9t</model_fragment>
      </Output>
    </has_Output>
    ...
    <model_fragment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >_002ashn00tq5gtqi5ps3</model_fragment>
    <has_succeedingActivities rdf:resource="#transfer_items_to_Orbit__Task___ID____
    002ashm02257o1b2mk88__"/>
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >make a report on expected quantities</name>
    <has_precedingActivities rdf:resource="#check_imported_items__Task___ID____
    002ashn00svtm1opmcfb__"/>
    <has_precedingActivities rdf:resource="#check_consignment_items__Task___ID____
    002ashn00ti8hhhhnmv7__"/>
    <phase_of rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ES.4-Manage_Product_Inventory</phase_of>
    <has_Actor-role rdf:resource="#logistics_department__Organization___ID____
    002asiu01firmt85i7ok__"/>
  </Activity>
  ...
  <ProcessModel rdf:ID="D__Project_models_sales_logistics_bpm.kmv">
    <filename rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >D:\Project\models\item_receiving.kmv</filename>
  </ProcessModel>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.2.1, Build 365)  http://protege.stanford.edu -->
```

H.3 Annotation of PM_{B2}

Parts of the annotation results of PM_{B2} are illustrated as follows. The whole OWL file is available at http://www.idi.ntnu.no/~yunl/schema/PMB2_eeml_modelAnno.owl

```
<!-- Created with Protege (with OWL Plugin 3.3, Build 399)  http://protege.stanford.edu -->
<?xml version="1.0"?>
<rdf:RDF
    xmlns="http://www.owl-ontologies.com/GPO.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://www.owl-ontologies.com/GPO.owl">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://protege.stanford.edu/plugins/owl/protege"/>
  </owl:Ontology>
  ...
  <Output rdf:ID="delivery_protocol__Output_Port___ID____002asiu000413j1u57o3__">
    <mapped_to rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Customer_Delivery_Terms</mapped_to>
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >delivery protocol</name>
    <model_fragment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >_002asiu000413j1u57o3</model_fragment>
  </Output>
  <Actor-role rdf:ID="shops__Organization___ID____002asiu012fh51i4fosm__">
    <model_fragment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >_002asiu012fh51i4fosm</model_fragment>
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >shops</name>
  </Actor-role>
  <Condition rdf:ID="every_morning__Milestone___ID____002asiu012aqkekt00mj__">
    <model_fragment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >_002asiu012aqkekt00mj</model_fragment>
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >every morning</name>
  </Condition>
  ...
  <Activity rdf:ID="Make_inventory_report__Task___ID____002asiu0014lfhvqb4cc__">
    <has_Artifact>
      <Artifact rdf:ID="inventory__Resource_role___ID____002asiu01gctup01d1qn__">
        <model_fragment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >_002asiu01gctup01d1qn</model_fragment>
        <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >inventory</name>
      </Artifact>
    </has_Artifact>
    <model_fragment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >_002asiu0014lfhvqb4cc</model_fragment>
    <has_subActivity rdf:resource="#report_stock_availability__Task___ID____
    002asiu012cl46vpmgon__"/>
    <has_Artifact>
      <Artifact rdf:ID="available_stock__Resource_role___ID____002asiu01gbuangfov0s__">
        <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >available stock</name>
        <model_fragment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >_002asiu01gbuangfov0s</model_fragment>
      </Artifact>
    </has_Artifact>
    <achieves rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Available_Inventory</achieves>
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Make inventory report</name>
```

```
    <has_Output rdf:resource="#__Output_Port___ID____002asiu0014lfhclk3h2__"/>
    <has_Input rdf:resource="#__Input_Port___ID____002asiu0014lfrq31hu2__"/>
    <has_subActivity>
      <Activity rdf:ID="Make_inventory__Task___ID____002asiu012hb1nj4m9hn__">
        <achieves rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Available_Inventory</achieves>
        <model_fragment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >_002asiu012hb1nj4m9hn</model_fragment>
        <has_Output>
          <Output rdf:ID="__Output_Port___ID____002asiu012hb1nf120lp__">
            <model_fragment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >_002asiu012hb1nf120lp</model_fragment>
            <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            ></name>
          </Output>
        </has_Output>
        <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Make inventory</name>
        <has_Actor-role>
          <Actor-role rdf:ID="logistics_department__Organization___ID____
          002asiu012kuuptiee45__">
            <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >logistics department</name>
            <model_fragment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >_002asiu012kuuptiee45</model_fragment>
          </Actor-role>
        </has_Actor-role>
        <has_Input rdf:resource="#__Input_Port___ID____002asiu012hb210krl92__"/>
      </Activity>
    </has_subActivity>
  </Activity>
  ...
  <ProcessModel rdf:ID="D__Project_models_sales_logistics_bpm.kmv">
    <filename rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >D:\Project\models\delivery.kmv</filename>
  </ProcessModel>
</rdf:RDF>
```

Bibliography

[1] ALTOVA. What is the Semantic Web? http://www.altova.com/semantic_web.html, Last Visited: 2007.5.20.

[2] Scott W. Ambler. *The Object Primer: Agile Model Driven Development with UML 2, 3rd Edition.* Cambridge University Press, 2004.

[3] Answers.com. Answers definition of "semantics". http://www.answers.com/topic/semantics?cat=health, Last Visited: 2007.2.1.

[4] Murtha Baca. *Introduction to Metadata: Pathways to Digital Information.* Getty Information Institute, 2000.

[5] Kenneth Baclawski, Mieczyslaw M. Kokar, Paul A. Kogut, Lewis Hart, Jeffrey E. Smith, III William S. Holmes, Jerzy Letkowski, and Michael L. Aronson. Extending UML to support ontology engineering for the semantic web. In *Proc. of the 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools*, pages 342–360, London, UK. LNCS 2185, Springer-Verlag, 2001.

[6] Gregory Bateson. *Mind and Nature: A Necessary Unity.* Bantam, 1988.

[7] Carlo Batini and Maurizio Lenzerini. A methodology for data schema integration in the entity-relationship model. In *Proc. of the 3rd International Conference on Entity-Relationship Approach (ER 1983)*, pages 413–420. North-Holland, 1983.

[8] Thomas Beale. Archetypes: Constraint-based domain models for future-proof information systems. In *Kenneth Baclawski and Haim Kilov (Eds.) 11th Workshop on Behavioral Semantics: Serving the Customer at OOPSLA2002*, 2002

[9] Ken Beck, Joshy Joseph, and German Goldszmide. Learn business process modeling basics for the analyst. http://www.ibm.com/developerworks/webservices/library/ws-bpm4analyst/, 2005.

[10] Abraham Bernstein and Mark Klein. Towards high-precision service retrieval. In *Proc. of the 1st International Semantic Web Conference (ISWC 2002)*, volume 2342, pages 84–101. Lecture Notes in Computer Science, 2002.

[11] BPMI. Business process modeling language. http://xml.coverpages.org/BPML-2002.pdf, Last Visited: 2005.2.22.

[12] BPMI. Business process modeling notation Version 1.0 May 3, 2004. http://www.bpmn.org/Documents/BPMN%20V1-0%20May%203%202004.pdf, Last Visited: 2005.2.22.

[13] Christopher Brewster, Kieron O'Hara, Steve Fuller, Yorick Wilks, Enrico Franconi, Mark A. Musen, Jeremy Ellman, and Simon Buckingham Shum. Knowledge representation with ontologies: The present and future. *IEEE Intelligent Systems*, 19(1):72–81, 2004.

[14] Ang Chen. Business process management mind map. http://smv.unige.ch/~chen/?attachment_id=13, Last Visited: 2007.6.22.

[15] Peter P. Chen. The Entity-Relationship Model – Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.

[16] Fabio Ciravegna, Alexiei Dingli, Yorick Wilks, and Daniela Petrelli. Adaptive information extraction for document annotation in amilcare. In *Proc. of the 25th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR 2002)*, pages 451–451, New York, NY, USA, 2002. ACM Press.

[17] Christine Collet, Michael N. Huhns, and Wei-Min Shen. Resource integration using a large knowledge base in Carnot. *IEEE Computer*, 24(12):55–62, 1991.

[18] Jordi Conesa and Antoni Olivé. A method for pruning ontologies in the development of conceptual schemas of information systems. *Journal on Data Semantics*, Vol.5, 64–90, 2006.

[19] Stephen Cranefield, Stefan Haustein, and Martin Purvis. UML-based ontology modeling for software agents. In *Proc. of Workshop on Ontologies in Agent Systems, 5th International Conference on Autonomous Agents*, pages 21–28, 2001.

[20] Cycorp. The syntax of CycL. http://www.cyc.com/cycdoc/ref/cycl-syntax.html, Last Visited: 2005.3.11.

[21] Anne Dardenne, Axel van Lamsweerde, and Stephen Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20(1-2):3–50, 1993.

[22] Randall Davis, Howard Shrobe, and Peter Szolovits. What is a knowledge representation? *AI Magazine*, (1):17–33, 1993.

[23] SearchCIO Definitions. Business process management. http://searchcio.techtarget.com/sDefinition/0,,sid182_gci1088467,00.html, Last Visited: 2007.9.15.

[24] Linda G. DeMichiel. Resolving database incompatibility: An approach to performing relational operations over mismatched domains. *IEEE Transactions on Knowledge and Data Engineering*, 1(4):485–493, 1989.

[25] Rose Dieng-Kuntz. Corporate semantic webs. *ERCIM News Special Theme: Semantic Web*, (51), 2002.

[26] Dov Dori. Why significant UML change is unlikely. *Communications of the ACM*, 45(11):82–85, 2002.

[27] Mark Dowson. Iteration in the software process; review of the 3rd international software process workshop. In *Proc. of the 9th International Conference on Software Engineering (ICSE 1987)*, pages 36–41, Los Alamitos, CA, USA, 1987.

[28] William R. Durrell. *A Practical Guide to Data Administration.* McGraw-Hill, 1985.

[29] ARPA Knowledge Sharing Effort. Knowledge interchange format (KIF). http://www-ksl.stanford.edu/knowledge-sharing/kif/, Last Visited: 2007.7.20.

[30] Clarence A. Ellis and Gary J. Nutt. Workflow: The process spectrum. In *Proc. of NSF Workshop on Workflow and Process Automation in Information Systems*, 1996.

[31] University of Toronto Enterprise Integration Laboratory. Tove ontology project. http://www.eil.utoronto.ca/enterprise-modelling/tove/index.html, 2002.

[32] Michael Erdmann, Alexander Maedche, Hans-Peter Schnurr, and Steffen Staab. From manual to semi-automatic semantic annotation: About ontology-based text annotation tools. In *Proc. of the COLING 2000 Workshop on Semantic Annotation and Intelligent Content*, 2000.

[33] Eckhard D. Falkenberg, Wolfgang Hesse, Paul Lindgreen, Björn E. Nilsson, J.L.Han Oei, Colette Rolland, Ronald K. Stamper, Frans J.M. Van Assche, Alexander A. Verrijn-Stuart, Klaus Voss. FRISCO — A framework of information system concepts — The FRISCO Report. IFIP WG 8.1 Task Group FRISCO, 1998.

[34] Peter H. Feiler and Watts S. Humphrey. Software process development and enactment: Concepts and definitions. Technical report, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1991.

[35] Dieter Fensel. Ontologies: Dynamic networks of formally represented meaning. http://sw-portal.deri.at/papers/publications/network.pdf, Last Visited: 2004.3.2.

[36] Dieter Fensel, Ian Horrocks, Frank van Harmelen, Stefan Decker, Michael Erdmann and Michel C.A. Klein Oil in a nutshell. In *Knowledge Acquisition, Modeling and Management, Proc. of 12th International Conference on (EKAW 2000)*, pages 1–16. LNCS 1937, Springer, 2000.

[37] Daniela Florescu, Ioana Manolescu, and Donald Kossmann. Answering XML queries over heterogeneous data sources. In *Proc. of the 27th International Conference on Very Large Data Bases (VLDB 2001)*, pages 241–250. Morgan Kaufmann, 2001.

[38] Enrico Franconi. Tutorial on description logics for conceptual design, information access, and ontology integration: Research trends. In *1st International Semantic Web Conference (ISWC 2002)*, 2002.

[39] Leonidas Galanis, Yuan Wang, Shawn R. Jeffery, and David J. DeWitt. Locating data sources in large distributed systems. In *Proc. of the 29th International Conference on Very Large Data Bases (VLDB 2003)*, pages 874–885. Morgan Kaufmann, 2003.

[40] GRL. GRL ontology. http://www.cs.toronto.edu/km/GRL/, Last Visited: 2007.8.12.

[41] Thomas R. Gruber. The role of common ontology in achieving sharable, reusable knowledge bases. In *James F. Allen, Richard Fikes, and Erik Sandewall (Eds.) Proc. of 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 601–602, San Mateo, California, 1991. Morgan Kaufmann.

[42] Thomas. R. Gruber. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In *N. Guarino and R. Poli (Eds.) Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands. Kluwer Academic Publishers, 1993.

[43] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[44] Michael Grüninger, Katy Atefi, and Mark S. Fox. Ontologies to support process integration in enterprise engineering. *Computational & Mathematical Organization Theory*, 6(4):381–394, 2000.

[45] Nicola Guarino. Formal ontology and information systems. In *N. Guarino (Eds.) Proc. of the 1st International Conference on Formal Ontologies in Information Systems (FOIS 1998)*, pages 3–15. IOS Press, 1998.

[46] Nicola Guarino, Claudio Masolo, and Guido Vetere. OntoSeek: Content-based access to the Web. *IEEE Intelligent Systems*, (3):70–80, 1999.

[47] Nicola Guarino and Luc Schneider. Ontology-driven conceptual modeling. In *Proc. of the 21st International Conference on Conceptual Modeling (ER 2002)*, page 10, London, UK. LNCS 2503, Springer-Verlag, 2002.

[48] Prentice Hall. Chapter 5 sales logistics. http://www.phptr.com/content/images/0130853402/samplechapter/0130853402.pdf, Last Visited: 2006.11.15.

[49] Siegfried Handschuh, Steffen Staab, and Rudi Studer. Leveraging metadata creation for the semantic web with CREAM. In *Proc. of the Annual German Conference on Advances in Artificial Intelligence (KI 2003)*, pages 19–33, Berlin. LNCS 2821, Springer, 2003.

[50] Pat Hayes and Chris Menzel. IKL specification document. http://nrrc.mitre.org/NRRC/Docs_Data/ikris/IKLspec.pdf, Last Visited: 2007.4.19.

[51] Bob Hendry. Metis 3.3 by Computas. Wireless Business & Technology http://wbt.sys-con.com/read/41291.htm, Last Visited: 2007.10.25.

[52] Ian Horrocks. Ontology reasoning: Why and How. Invited talk on the Workshop of Reasoning on the Web at WWW2006. http://www.aifb.uni-karlsruhe.de/WBS/phi/RoW06/procs/horrocks.txt/, Last Visted: 2007.5.10.

[53] Ian Horrocks. Reasoning with expressive description logics: Theory and practice. In *Automated Deduction - CADE-18, 18th International Conference on Automated Deduction, Copenhagen, Denmark, July 27-30, 2002, Proceedings*, Lecture Notes in Computer Science, pages 1–15. LNCS 2392, Springer Berlin/Heidelberg, 2002.

[54] Karin Anna Hummel, Wolfgang Jochum, Stefan Leitich, and Bernhard Schandl. Supporting meetings with a goal-driven service-oriented multimedia environment. In *Proc. of the 1st ACM International Workshop on Multimedia Service Composition (MSC 2005)*, pages 55–65, New York, NY, USA, 2005. ACM Press.

[55] i*. An agent-oriented modelling framework. http://www.cs.toronto.edu/km/istar/, Last Visited: 2007.6.11.

[56] Annie I.Anton. Goal based requirements analysis. In *Proc. of the 2nd International Conference on Requirements Engineering (ICRE 1996)*, pages 136–144, 1996.

[57] LEKS IASI-CNR. AStar. http://leks-pub.iasi.cnr.it/Astar, Last Visited: 2007.2.1.

[58] IBM. Business process execution language for web services Version 1.1. http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel/ws-bpel.pdf, Last Visited: 2007.5.12.

[59] EXTERNAL Project in EU's IST programme. Extended enterprise resources, network architectures and learning (ist-1999-10091). http://research.dnv.com/external/default.htm, Last Visited: 2007.9.1.

[60] Inicio. EPC and eEPC. http://paginas.terra.com.br/negocios/processos2002/epc_e_eepc.htm, Last Visited: 2007.5.11.

[61] Technische Universität Hamburg-Harburg (TUHH) Institut für Softwaresysteme. Racer/racerpro: The first OWL reasoner in the market. http://www.sts.tu-harburg.de/~r.f.moeller/racer/, Last Visited: 2007.9.1.

[62] INTEROP. Interoperability research for networked enterprises applications and software. http://interop-noe.org/, Last Visited: 2007.4.8.

[63] Mustafa Jarrar, Jan Demey, and Robert Meersman. On using conceptual data modeling for ontology engineering. *Journal on Data Semantics (Special issue on Best papers from the ER, ODBASE and COOPIS 2002 Conferences)*, 2800:185–207, October 2003.

[64] Kurt Jensen. *Coloured Petri Nets – Basic Concepts, Analysis Methods and Practical Use*. Springer-Verlag, 1997.

[65] Håvard D. Jørgensen. Interactive process models. PhD thesis, Norwegian University of Science and Technology, 2004.

[66] Vipul Kashyap and Amit P. Sheth. Semantic and schematic similarities between database objects: A context-based approach. *VLDB Journal: Very Large Data Bases*, 5(4):276–304, 1996.

[67] Steven Kelly and Juha-Pekka Tolvanen. Visual domain-specific modeling: Benefits and experiences of using metacase tools. In *J. Bezivin, J. Ernst(Eds.) Proc. of the International workshop on Modeling Engineering, ECOOP*, 2000.

[68] Larry Kerschberg and Doyle Weishar. Conceptual models and architectures for advanced information systems. *Applied Intelligence*, 13(2):149–164, 2000.

[69] Ekkart Kindler. On the semantics of EPCs: A framework for resolving the vicious circle. Technical report, Institue fur Informatk, Universitat Paderborn, 2003.

[70] Atanas Kiryakov, Borislav Popov, Ivan Terziev, Dimitar Manov, and Damyan Ognynoff. Semantic annotation, indexing, and retrieval. *Journal of Web Semantics*, 2(1), 2005.

[71] KM. The knowledge machine. http://www.cs.utexas.edu/users/mfkb/RKF/km.html, Last Visited: 2007.6.12.

[72] Paul Kogut and William Holmes. AeroDAML: Applying information extraction to generate DAML annotations from web pages. In *Proc. of the 1st International Conference on Knowledge Capture (K-CAP 2001)*, 2001.

[73] John Krogstie, Odd Ivar Lindland, and Guttorm Sindre. Defining quality aspects for conceptual models. In *Proc. of the IFIP international Working Conference on Information System Concepts*, pages 216–231, London, UK. Chapman & Hall, Ltd., 1995.

[74] John Krogstie. Using quality function development in software requirements specification. In *Proc. of the 5th International Workshop on Requirements Engineering: Foundations for Software Quality (REFSQ 1999)*, pages 171–185, 1999.

[75] John Krogstie. Using a semiotic framework to evaluate UML for the development of models of high quality. *Unified Modeling Language: Systems Analysis, Design and Development Issues*, pages 89–106. IGI Publishing, 2001.

[76] John Krogstie and Arne Sølvberg. *Information Systems Engineering: Conceptual Modeling in a Quality Perspective*. Norwegian University of Science and Technology, Trondheim (NTNU). Unpublished book, 2003.

[77] John Krogstie and Håvard D. Jørgensen. Interactive models for supporting networked organisations. In *Proc. of the 16th International Conference on Advanced Information Systems Engineering*, pages 550–562. LNCS 3084, Springer-Verlag, 2004.

[78] John Krogstie and Sofie de Flon Arnesen. Assessing enterprise modeling languages using a generic quality framework. *Information Modeling Methods and Methodologies*, pages 63–79. Idea Group Publishing, 2005.

[79] John Krogstie, Csaba Veres, and Guttorm Sindre. Integrating semantic web technology, web services, and workflow modeling achieving system and business interoperability. In *International Journal of Enterprise Information Systems*, 3(1):22–41, 2007.

[80] John Krogstie. Integrated goal, data and process modeling: from TEMPORA to model-generated work-places. In *Paul Johannesson and Eva Soderstrom (Eds.) Information Systems Engineering: From Data Analysis to Process Networks*, IGI publishing, in print, 2008.

[81] Ontotext Lab. The KIM platform: Semantic annotation. http://www.ontotext.com/kim/semanticannotation.html, Last Visited: 2007.5.12.

[82] Charles Lakos. From Coloured Petri Nets to Object Petri Nets. In *Proc. of the Application and Theory of Petri Nets*, pages 278–297.LNCS 935, Springer-Verlag, Berlin, Germany, 1995.

[83] Charles Lakos. Pragmatic inheritance issues for object Petri Nets. In *Proc. of TOOLS Pacific Conference*, 1995.

[84] James A. Larson, Shamkant B. Navathe, and Ramez Elmasri. A theory of attributed equivalence in databases with application to schema integration. *IEEE Transactions on Software Engineering*, 15(4):449–463, 1989.

[85] Jintae Lee, Gregg Yost, and the PIF Working Group. The PIF process interchange format and framework. Technical report, MIT Center for Coordination Science, 1994.

[86] Maurizio Lenzerini. Data integration: a theoretical perspective. In *Proc. of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2002)*, pages 233–246, New York, NY, USA. ACM Press, 2002.

[87] Mauri Leppänen. An ontological framework and a methodical skeleton for method engineering — a contextual approach. PhD thesis, University of Jyväskylä, Jyväskylä, Finland, 2005.

[88] Alon Y. Levy. Combining artificial intelligence and database for data integration. In *Artificial Intelligence Today: Recent Trends and Developments*, pages 249–268, Berlin/Heidelberg, LNCS 1600, Springer. 1999.

[89] Alon Y. Levy, Anand Rajaraman, and Joann J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proc. of the 22nd International Conference on Very Large Databases*, pages 251–262, Bombay, India. VLDB Endowment, Saratoga, California. 1996.

[90] Manshan Lin, Heqing Guo, and Jianfei Yin. Goal description language for semantic Web service automatic composition. In *Proc. of IEEE/IPSJ International Symposium on Applications and the Internet (SAINT 2005), 31 January - 4 February 2005*, pages 190–196, Trento, Italy, 2005.

[91] Yun Lin, Jennifer Sampson, Sari Hakkarainen, and Hao Ding. An evaluation of UML and OWL using a semiotic quality framework. *Advanced Topics in Database Research Volume 4*, pages 178–200, Idea Group Publishing. Hershey, PA, USA, 2004.

[92] Yun Lin and Darijus Strašunskas. Ontology-based semantic annotation of process models. In *Proc. of 10th CAiSE/IFIP8.1/EUNO International Workshop on Evaluation of Modeling Methods in System Analysis and Design (EMMSAD05)* Porto, Portugal. 2005

[93] Yun Lin and Hao Ding. Ontology-based semantic annotation for semantic ineroperability of process models. In *Proc. of International Conference on Computational Intelligence for Modelling, Control and Automation (CIMCA 2005)* IEEE USA, 2005.

[94] Yun Lin, Darijus Strašunskas, Sari Hakkarainen, John Krogstie and Arne Sølvberg. Semantic annotation framework to manage semantic heterogeneity of process models. In *Proc. of the 18th International Conference on Advanced Information Systems Engineering (CAiSE 2006)* pages 433–446, Luxembourg, Luxembourg. LNCS 4001, Springer-Verlag, 2006.

[95] Yun Lin and Arne Sølvberg. Goal annotation of process models for semantic enrichment of process knowledge. In *Proc. of the 19th International Conference on Advanced Information Systems Engineering (CAiSE 2007)* pages 355–369, Trondheim, Norway. LNCS 4495, Springer-Verlag, 2007.

[96] Clark & Parsia LLC. Pellet. http://pellet.owldl.com/, Last Visited: 2007.11.1.

[97] LOOM. Loom knowledge representation systems. http://www.isi.edu/isd/LOOM/LOOM-HOME.html#OVERVIEW, Last Visited: 2007.8.19.

[98] Thomas W. Malone and Kevin Crownston. The interdisciplinary study of coordination. *ACM Computing Surveys*, 26(1):87–119, 1994.

[99] Thomas W. Malone, Kevin Crownston, and George A. Herman. *Organizing Business Knowledge — The MIT Process Handbook*. The MIT Press, Cambridge, Massachusetts, London, England, 2003.

[100] Frank Manola. Towards a web object model. http://www.objs.com/OSA/wom.htm, Last Visited: 2007.6.18.

[101] TOVE Manual. Chapter 3 an activity ontology for enterprise modeling. http://www.eil.utoronto.ca/tove/active/active32.html, Last Visited: 2007.6.22.

[102] Diana Maynard. Benchmarking ontology-based annotation tools for the semantic web. In *Workshop of Text Mining, e-Research and Grid-enabled Language Technology at UK e-Science Programme All Hands Meeting (AHM 2005)* Nottingham, UK, 2005.

[103] Raul Medina-Mora, Terry Winograd, Rodrigo Flores, and Fernando Flores. The action workflow approach to workflow management technology. In *Proc. of the 1992 ACM conference on Computer-supported cooperative work (CSCW 1992)*, pages 281–288, New York, NY, USA. ACM Press, 1992.

[104] Robert Meersman. Ontologies and databases: More than a fleeting resemblance. In *Proc. of OES/SEO Workshop*, 2001.

[105] Jan Mendling and Markus Nüttgens. EPC Markup Language (EPML) — an XML-based interchange format for Event-driven Process Chains (EPC). *International Journal of Information Systems and e-Business Management (ISeB)*, 4(3):245–263, 2006.

[106] Peter Mika, Daniel Oberle, Aldo Gangemi, and Marta Sabou. Foundations for service ontologies: Aligning OWL-S to DOLCE. In *Proc. of the 13th International Conference on World Wide Web (WWW2004)*, pages 563–572, New York, NY, USA. ACM Press, 2004.

[107] Randy Miller. Practical UML: A hands-on introduction for developers. Borland Software Corporation, Inc. 2003. http://dn.codegear.com/article/31863, Last Visited: 2007.5.12.

[108] Daniel Moldt and Rüdiger Valk. Object oriented petri nets in business process modeling. In *Business Process Management, Models, Techniques, and Empirical Studies*, pages 254–273, London, UK. LNCS 1806, Springer-Verlag, 2000.

[109] Mark A. Musen. Ontologies: Necessary — indeed essential — but not sufficient. *IEEE Intelligent Systems*, 19(1):77–79, 2004.

[110] John Mylopoulos, Lawrence Chung, and Eric Yu. From object-oriented to goal-oriented requirements analysis. *Communications of the ACM*, 42(1):31–37, 1999.

[111] Shamkant B. Navathe and Suresh G. Gadgil. A methodology for view integration in logical database design. In *Proc. of the 8th International Conference on Very Large Data Bases (VLDB 1982)*, pages 142–164, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc., 1982.

[112] Fred Nickols. Knowledge Management (KM) and process performance — implications for action. http://home.att.net/~nickols/KM_and_Processes.htm, Last Visited: 2007.6.17.

[113] Sergei Nirenburg and Yorick Wilks. What's in a symbol: Ontology, representation, and language. *Experimental and Theoretical Artificial Intelligence*, 13(1):9–23, 2001.

[114] Ikujiro Nonaka and Hirotaka Takeuchi. *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation.* Oxford University Press, 1995.

[115] Anna Gunhild Nysetvold and John Krogstie. Assessing business process modeling languages using a generic quality framework. *Advanced Topics in Database Research Volume 5* pages 84–101, Idea Group Publishing. Hershey, PA, USA, 2006.

[116] OASIS. Business Process Execution Language for Web Services (BPEL4WS). http://xml.coverpages.org/bpm.html#bpel4ws, Last Visited: 2007.4.5.

[117] OMG. MDA guide version 1.0.1. http://www.omg.org/docs/omg/03-06-01.pdf, Last Visited: 2004.1.10.

[118] AIFB (Institute of Applied Informatics and University of Karlsruhe Formal Description Methods). KAON2. http://kaon2.semanticweb.org/, Last Visited: 2007.7.17.

[119] IEEE. IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries Institute of Electrical and Electronics Engineers, 1991.

[120] National Institute of Standards and Technology. Process Specification Language (PSL) core. http://www.mel.nist.gov/psl/psl-ontology/pslcore_page.html, 2004.

[121] Charles Kay Ogden and Ivor Armstrong Richards. *The Meaning of Meaning.* London: Kegan Paul, 1923.

[122] Antoni Olivé. Conceptual schema-centric development: A grand challenge for information systems research. In *Proc. of the 17th International Conference on Advanced Information Systems Engineering (CAiSE 2005)* pages 1–15, Porto, Portugal. LNCS 3520, Springer-Verlag, 2005.

[123] OMG. Model driven architecture. http://www.omg.org/mda/, Last Visited: 2007.6.7.

[124] OMG. Meta Object Facility (MOF) specification version 1.4.1 formal/05-05-05. http://www.omg.org/docs/formal/05-05-05.pdf, Last Visited: 2007.6.7.

[125] OMG. Unified modeling language. http://www.uml.org/, Last Visited: 2007.6.7.

[126] BPMI & OMG. Business process management initiative. http://www.bmpi.org/, Last Visited: 2007.6.7.

[127] Ontotext.com. KIM platform. http://www.ontotext.com/kim/introduction.html, Last Visited: 2007.6.7.

[128] Andreas L. Opdahl and Brian Henderson-Sellers. Ontological evaluation of the UML using the Bunge-Wand-Weber model. *Software and Systems Modeling*, 1(1):43–67, 2002.

[129] OASIS Cover Pages. Event-driven Process Chain Markup Language (EPML) for business process modeling. http://xml.coverpages.org/ni2003-11-21-a.html, Last Visited: 2007.7.17.

[130] Hervé Panetto, Giuseppe Berio, Khalid Benali, Nacer Boudjlida, and Michaël Petit. A unified enterprise modeling language for enhanced interoperability of enterprise models. In *Proc. of the 11th IFAC INCOM2004 Symposium,*, 2004.

[131] Massimo Paolucci, Naveen Srinivasan, Katia Sycara, and Takuya Nishimura. Towards a semantic choreography of web. In *Proc. of the 1st International Conference on Web Services (ICWS 2003)*, pages 22–26, 2003.

[132] Jinsoo Park and Sudha Ram. Information systems interoperability: What lies beneath? *ACM Transactions on Information System*, 22(4):595–632, 2004.

[133] Abhijit A. Patil, Swapna A. Oundhakar, Amit P. Sheth, and Kunal Verma. METERO-S Web service annotation framework. In *Proc. of the 13th International Conference on World Wide Web (WWW 2004)*, pages 553–562, New York, NY, USA. ACM Press, 2004.

[134] Jean-Christophe R. Pazzaglia and Suzanne M. Embury. Bottom-up integration of ontologies in a database context. In *Alexander Borgida, Vinay K. Chaudhri, Martin Staudt (Eds.): Proc. of the 5th INternational Workshop on Knowledge Represenation Meets Databases (KRDB 1998): Innovative Application Programming and Query Interfaces*, pages 7.1–7.7. Seattle, Washington, USA, 1998.

[135] Adam Pease. Object Model Working Group (OMWG) Core Plan Representation, version 4. Technical report, Defense Advanced Research Projects Agency, 1998.

[136] Borislav Popov, Atanas Kiryakov, Angel Kirilov, Dimitar Manov, Damyan Ognyanoff, and Mirosla Goranov. KIM - semantic annotation platfrom. In *Proc. of the 2nd International Semantic Web Conference (ISWC2003)*, pages 834–849, 2003.

[137] Walter D. Potter and Larry Kerschberg. A unified approach to modeling knowledge and data. *Data and Knowledge (DS-2)*, pages 265 – 292, 1988.

[138] ATHENA Project. Deliverable DA1.3.1 report on methodology description and guidelines definition. http://www.athena-ip.org/, Last Visited: 2005.12.17.

[139] European Project. Athena (advanced technologies for interoperability of heterogeneous enterprise networks and their applications) project (ist-2003-2004). http://www.athena-ip.org, Last Visited: 2007.12.17.

[140] European Project. Interop (interoperability research for networked enterprises applications and software) project (ist-508011). http://www.athena-ip.org, Last Visited: 2007.7.27.

[141] INTEROP Project. Deliverable DEM 1 UEML2.1. http://www.interop-noe.org/, Last Visited: 2005.7.17.

[142] INTEROP Project. Deliverable DEM 2 roadmap for UEML and UEML2.1. http://www.interop-noe.org/, Last Visited: 2006.12.1.

[143] INTEROP Project. Deliverable DTG4.1 a practical experiment on semantic enrichment of enterprise models in a homogeneous environment. http://interop-noe.org/, Last Visited: 2006.12.1.

[144] INTEROP Project. UEML ontology overview. http://www.interop-noe.org/, Last Visited: 2006.12.1.

[145] INTEROP Project. Deliverable DTG4.2 experimental semantic enrichment of enterprise models for interoperability and its practical impacts. http://interop-noe.org/, Last Visited: 2007.8.6.

[146] Protégé. Protégé-OWL API. http://protege.stanford.edu/plugins/owl/api/, Last Visited: 2007.5.26.

[147] Protégé. What is Protégé? http://protege.stanford.edu/overview/index.html, Last Visited: 2007.5.26.

[148] Protégé. What is Protégé-OWL? http://protege.stanford.edu/overview/protege-owl.html, Last Visited: 2007.5.26.

[149] ProtegeWiki. SWRL Language FAQ. http://protege.cim3.net/cgi-bin/wiki.pl?SWRLLanguageFAQ, Last Visited: 2007.5.26.

[150] ProtegeWiki. SQWRLQueryTab. http://protege.cim3.net/cgi-bin/wiki.pl?SQWRLQueryTab, Last Visited: 2007.11.30.

[151] ProtegeWiki. SWRL Tab. http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTab, Last Visited: 2007.5.26.

[152] Sudha Ram and Venkataraman Ramesh. Schema integration: Past, current and furture. *A. Elmagarmid,. M. Rusinkiewicz, A. Sheth, (ed.) Management of Heterogeneous and Autonomous Database Systems*, pages 119–155, 1999.

[153] Lawrence Reeve and Hyoil Han. Survey of semantic annotation platforms. In *Proc. of the 2005 ACM symposium on Applied computing*, pages 1634–1638, New York, NY, USA. ACM Press, 2005.

[154] Cornelis Joost van Rijsbergen. *Information Retrieval (2nd edition)*. Butterworths, 1979.

[155] Colette Rolland. Modeling the requirements engineering process. In *Proc. of the 3rd European-Japanese Seminar on Information Modeling and Knowledge Bases*, 1993.

[156] Colette Rolland and Naveen Prakash. Bridging the gap between organizational needs and ERP funtionality. *Requirements Engineering*, 5(3):180–193. Springer London, 2000.

[157] Colette Rolland and Rim-Samia Kaabi. An intentional perspective to service modeling and discovery. In *Proc. of the 31st Annual International Computer Software and Applications Conference - Vol. 2- (COMPSAC 2007)*, pages 455–460, Washington, DC, USA. IEEE Computer Society, 2007.

[158] Michael Rosemann and Peter Green. Developing a meta model for the Bunge-Wand-Weber ontological constructs. *Information Systems*, 27(2):75–91, 2002.

[159] Gerard Salton. *Automatic Text Processing: The Transformation Analysis and Retrieval of Information by Computer*. Addison-Wesley, 1988.

[160] August-Wilhelm Scheer and Markus Nüttgens. ARIS architecture and reference models for business process management. *Business Process Management, Models, Techniques, and Empirical Studies*, Volume 1806/2000 pages 376–389, Springer-Verlag. London, UK, 2000.

[161] Craig Schlenoff, Michael Gruninger, Mihai Ciocoiu, and Jintae Lee. The essence of the process specification language. *Transactions of the Society for Computer Simulation International*, 16(4):204–216, 1999.

[162] The University of Manchester School of Computer Science. FaCT++. http://owl.man.ac.uk/factplusplus/, Last Visited: 2007.5.1.

[163] SCOR. SCOR model. http://www.supply-chain.org/page.ww?section=SCOR+Model\&name=SCOR+Model, Last Visited: 2007.5.10.

[164] SCOR. SCOR version 7.0 overview. http://www.supply-chain.org/cs/root/scor_tools_resources/scor_model/scor_model, Last Visited: 2006.3.22.

[165] Peter Senge. *The Fifth Discipline: The Art & Practice of the Learning Organization*. Doubleday-Currency, 1990.

[166] Kaarthik Sivashanmugam, John A. Miller, Amit P. Sheth, and Kunal Verma. Framework for semantic web process composition. *Special Issue of the International Journal of Electronic Commerce (IJEC)*, 9(2), 2004.

[167] Pnina Soffer and Yair Wand. On the notion of soft-goals in business process modeling. *Business Process Management Journal*, 11(6):663–679, May-June 2005.

[168] Arne Sølvberg. Data and what they refer to. In *Conceptual Modeling: Current Issues and Future Trends.*, pages 211–226. LNCS 1565, Springer-Verlag, 1999.

[169] Arne Sølvberg. Introdution to concept modeling for information systems. Norwegian University of Science and Technology, Trondheim, Norway, 2002.

[170] Arne Sølvberg, Sari Hakkarainen, Terje Brasethvik, Xiaomeng Su, Mihhail Matskin, and Darijus Strašunskas. Concepts on enriching, understanding and retrieving the semantics on the Web. *ERCIM News. Special Theme: Semantic Web*, (51), 2002.

[171] John.F. Sowa and John A. Zachman. Extending and formalizing the framework for information systems architecture. *IBM System Journal*, 31(3):590–616, 1992.

[172] Stefano Spaccapietra, Christine Parent, and Yann Dupont. Model indepentent assertions for integration of heterogeneous schemas. *VLDB Journal*, 1(1):81–126, 1992.

[173] Steffen Staab, Jurgen Angele, Stefan Decker, Michael Erdmann, Andreas Hotho, Alexander Maedche, Hans-Peter Schnurr, Rudi Studer, and York Sure. Semantic community web portals. In *Proc. of the 9th International World Wide Web Conference on Computer Networks : the International Journal of Computer and Telecommunications Netowrking*, pages 473–491, Amsterdam, The Netherlands. North-Holland Publishing Co., 2000.

[174] Steffen Staab, Rudi Studer, Hans-Peter Schnurr, and York Sure. Knowledge processes and ontologies. *IEEE Intelligent Systems*, 16(1):26–34, 2001.

[175] Darijus Strašunskas. *Domain Model-Centric Distributed Development — An Approach to Semantics-based Change Impact Management.* PhD thesis, Norwegian University of Science and Technology, 2006.

[176] Scott Stephens. Supply Chain Operations (SCOR) reference model and the integrated business reference framework. Speech on the Supply Chain International Conference, 2006.

[177] Heiner Stuckenschmidt. *Ontology-based Information Sharing in Weakly Structured Environments.* PhD thesis, Vrije University Amsterdam, 2003.

[178] Xiaomeng Su. *Semantic Enrichment for Ontology Mapping.* PhD thesis, Norwegian University of Science and Technology, 2004.

[179] SUN. JAXP java API for XML processing. http://java.sun.com/webservices/jaxp/, Last Visited: 2007.2.10.

[180] York Sure, Alexander Maedche, and Steffen Staab. Leveraging corporate skill knowledge — from ProPer to OntoProPer. In *Proc. of the 3th international Conference on Practical Aspects of Knowledge Management*, 2000.

[181] Semantic Web Services Language (SWSL) Committee. Semantic Web Services Framework (SWSF). http://www.daml.org/services/swsf/1.0, Last Visited: 2007.5.3.

[182] Katia Sycara, Massimo Paolucci, Anupriya Ankolekar, and Naveen Srinivasan. Automated discovery, interaction and composition of semantic web services. *Journal of Web Semantics*, 1(1):2–27, 2003.

[183] Troux Technologies. Metis. http://www.troux.com/, Last Visited: 2007.7.10.

[184] Advanced Knolwedge Technology. MnM. http://kmi.open.ac.uk/projects/akt/MnM/, Last Visited: 2007.6.22.

[185] Systems Thinking. Knowledge management — emerging perspective. http://www.systems-thinking.org/kmgmt/kmgmt.htm#anex, Last Visited: 2007.6.22.

[186] Eran Toch, Avigdor Gal, and Dov Dori. Automatically grounding semantically-enriched conceptual models to concrete web services. In *Proc. of 24th International Conference on Conceptual Modeling (ER 2005)*, pages 304–319. LNCS 3716, Springer, 2005.

[187] Computer Science Department University of Georgia. METEOR-S: Semantic web services and processes. http://lsdis.cs.uga.edu/projects/meteor-s/, Last Visited: 2007.6.22.

[188] Mike Uschold and Michael Grüninger. Ontologies: Principles, methods, and applications. *Knowledge Engineering Review*, 11(2):93–155, 1996.

[189] VA. A tutorial on the zachman framework for enterprise architecture. http://www.va.gov/oirm/architecture/EA/theory/tutorial.ppt, Last Visited: 2007.3.5.

[190] Wil M.P. van der Aalst. Formalization and verification of event-driven process chains. *Information and Software Technology*, 41(10):639–650, 1999.

[191] Wil M.P. van der Aalst, Ana P. Barros, Arthur H.M. ter Hofstede, and Bartek Kiepuszewski. Advanced workflow patterns. In O. Etzion en P. Scheuermann, editor, *Proc. of 7th International Conference on Cooperative Information Systems (CoopIS 2000)*, pages 18–29. LNCS 1905, Springer-Verlag. Berlin, 2000.

[192] Axel van Lamsweerde. Building formal requirements models for reliable software. In *Proc. of the 6th Ade-Europe International Conference Leuven on Reliable Software Technologies*, pages 1–20, London, UK. Springer-Verlag, 2001.

[193] Maria Vargas-Vera, Enrico Motta, John Domingue, Mattia Lanzoni, Arthur Stutt, and Fabio Ciravegna. MnM: Ontology driven semi-automatic and automatic support for semantic markup. In *Proc. of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2002)*, pages 379–391, London, UK. LNCS 2473, Springer-Verlag, 2002.

[194] Olegas Vasilecas and Diana Bugaite. Ontology-based elicitation of business rules. In *Proc. of Information Systems Development (ISD 2005)*, pages 795–806, Sweden. Springer-Verlag, 2006.

[195] W3C. OWL web ontology language guide. http://www.w3.org/TR/2004/REC-owl-guide-20040210/, Last Visited: 2007.5.2.

[196] W3C. OWL web ontology language reference. http://www.w3.org/TR/2004/REC-owl-ref-20040210/, Last Visited: 2007.5.2.

[197] W3C. OWL web ontology language overview. http://www.w3.org/TR/owl-features/, Last Visited: 2007.5.2.

[198] W3C. OWL-S: Semantic markup for web services. http://www.w3.org/Submission/OWL-S/, Last Visited: 2004.10.24.

[199] W3C. RDF vocabulary description language 1.0: RDF schema. http://www.w3.org/TR/rdf-schema/, Last Visited: 2006.12.1.

[200] W3C. Simple part-whole relations in OWL ontologies. http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/index.html, Last visited: 2006.3.17.

[201] W3C. Semantic annotations for WSDL and XML schema. http://www.w3.org/TR/sawsdl/, Last Visited: 2007.9.20.

[202] W3C. SWRL: A semantic web rule language combining OWL and RuleML. http://www.w3.org/Submission/SWRL, Last Visited: 2007.8.19.

[203] W3C. Web service semantics — WSDL-S. http://lsdis.cs.uga.edu/library/download/WSDL-S-V1.html, Last Visited: 2005.10.5.

[204] Yair Wand and Ron Weber. An ontology model for an information system. *IEEE Transactions on Software Engineering*, 16(11):1282–1292, 1990.

[205] Yair Wand and Ron Weber. On the deep structure of information systems. *Information System Journal*, 5:203–223, 1995.

[206] Branimir Wetzstein, Zhilei Ma, Agata Filipowska, Monika Kaczmarek, Sami Bhiri, Silvestre Losada, Jose-Manuel Lopez-Cobo, and Laurent Cicurel. Semantic business process management: A lifecycle based requirements analysis. In *Proc. of Workshops on Semantic Business Process and Product Lifecycle Management (SBPM 2007) at the 4th European Semantic Web Conference (ESWC 2007)*, pages 1–11. CEUR WS, 2007.

[207] Wikipedia. Metamodeling. http://en.wikipedia.org/wiki/Meta-modeling, Last Visited: 2007.12.12.

[208] Wikipedia. Supply-chain operations reference. http://en.wikipedia.org/wiki/SCOR-model, Last Visited: 2007.11.10.

[209] WSMO.org. Web Service Modeling Ontology (wsmo). http://www.wsmo.org/, Last Visited: 2007.11.10.

[210] WSMO.org. D3.1 v0.1 WSMO primer. http://www.wsmo.org/TR/d3/d3.1/v0.1/, Last Visited: 2007.8.16.

[211] Yahoo! Yahoo!directory. http://dir.yahoo.com/, Last Visited: 2007.7.12.

[212] Eric Yu, Lin Liu, and Ying Li. Modeling strategic actor relationships to support intellectual property management. In *Proc. of the 20th International Conference on Conceptual Modeling (ER 2001)*, pages 164–178, London, UK. LNCS 2224, Springer-Verlag, 2001.

[213] Eric Yu and John Mylopoulos. Why goal-oriented requirements engineering. In *Proc. of the 4th of International Workshop on Requirements Egnineering: Foundations of Software Quality*, 1998.

[214] John.A. Zachman. A framework for information systems architecture. *IBM System Journal*, 26(3):454–470, 1987.