Formal models for ontologies

Project: ANR - IMPEX

Authors: IMPEX consortium

Contact: yamine@enseeiht.fr

Date May 2015

Contents

Chapter 1

Introduction

This report presents an overview of ontology models and the way they are set up in engineering. The objective of the work presented in this report is to show how knowledge domain models can help in the definition of high quality design models.

This report is organized as follows. Chapter 1 describes the main characteristics of ontologies. Formal modeling aspects related to the ontology models are overviewed. In chapter 2, we address the problem of design model annotations by references to ontologies. Finally, chapter 3 outlines the next research directions to be addressed in the IMPEX project.

Chapter 2

Design models and ontologies

This chapter describes the stepwise methodology we have defined in order to allow model designers to handle explicitly domain knowledge. The proposed approach consists in associating a domain ontology (concepts and associated constraints) to a design model. This association, is performed thanks to a defined annotation mechanism. The defined association mechanism entails a loosely coupling of the ontology and of the annotated models. Indeed, no modification nor evolution of the design models is required. Moreover, ontologies and models may evolve asynchronously.

2.1 Ontologies are good candidates for domain knowledge modeling

As mentioned above, a lot of efforts have been devoted to the study of ontologies and their applications in the area of semantic web and information retrieval. Several approaches for describing, designing and formalizing ontologies for these application domains have been proposed by many authors. Models [13, 8, 37, 28, 43, 40], browsers like Protégé[1] [32, 1] or PlibEditor[2], reasoners [6, 23, 24, 36], annotators [17, 25] and XML-based translators [9, 45] have been developed to engineer such ontologies and establish formal links with the studied domain objects like texts, images, videos, signals, ... Most of these approaches paid a lot of attention to the use of ontologies to interpret these objects and/or to provide classifications of these interpreted objects.

In this section, we aim to to give our view of domain ontologies with a specific focus on system engineering and model engineering. We study how ontologies can be used to enrich design models by relevant domain properties and thus to increase the expressivity of such design models. First at all we give our point of view on domain models expressed by ontologies and design models.

[1] http://protege.stanford.edu/
[2] http://www.plib.ensma.fr/

2.1.1 Domain modeling: Ontologies

Definition

Gruber defines an ontology as *an explicit specification of a conceptualization* [21]. Another definition relies on the notion of dictionary, where [31] considers a domain ontology as a *formal and consensual dictionnary of categories and properties of entities of a domain and the relationships that hold among them.* Here, an entity represents any concept belonging to the considered domain. The term *dictionary* entails two major concepts. First, it makes explicit the existence, through a constructive definition or declaration, of entities in the domain under consideration and second that any entity or relationship described in this ontology is directly referencable, for any purpose and from any context, independently of the other entities or relationships. Reference is carried by a symbol defining an identifier. This identification symbol may be either a language-independent identifier, or a language-specific set of words. However, whatever this symbol is, and unlike in linguistic dictionary, it directly denotes a domain entity or relationship. Each *description* of each entity or relationship is formally *stated* using an ontology modeling language equipped with a formal semantics. It allows automatic reasoning and consistency checking.

Some fundamental characteristics

A domain ontology is an explicit conceptualization of domain entities and relationships. Ontology definitions will fulfil three fundamental criteria [31].

1. **Formality.** An ontology is a conceptualization with an underlying formal semantics and equipped with reasoning capabilities. It is expressed within a modeling language equipped with a satisfaction relationship ($\models_O$) offering interpretation capabilities (e.g. checking that an instance belongs to the model defined by the ontology) and an entailment relationship ($\vdash_O$) to handle proofs (e.g. proving that a statement can be derived from axioms and theorems defined by the ontology). As a consequence, checking properties expressed over the ontology-defined concepts and individuals, becomes possible, thanks to the associated theory, either by automatic or semi-automatic reasoning techniques.

2. **Consensuality.** Agreement on the conceptualization defined by an ontology will be reached for a large community of users. This community is not restricted to users or to developers of a specific application: it gathers all the potential users and developers of other applications related to the conceptualized domain. Consequently, an ontology will be shared by several applications and design models. For example, ISO 13584-compliant (PLIB) [40, 43] product ontologies are defined according to a formal standardization process. They are published as ISO and/or IEC international standards. This criterion excludes conceptual models defined for a specific application.

3. **Capability to be referenced.** As stated in the previous definition, each concept defined in an ontology is associated to an identifier provided to allow

applications to refer this concept from any environment. Moreover, this concept can be referenced whatever is the ontology model set up to describe this concept.

2.1.2 System Modeling: Design Models

Design models address the definition of models of systems to be realized. They are described within modeling languages and they correspond to abstractions of the considered system. Semantics of the modeling language is given by a satisfaction relationship ($\models_M$) checking if a model is satisfiable and an entailment relationship ($\vdash_M$) defining a proof system where properties can be proved from axioms, theorems and proof rules applications). Various analyses, properties checking, models manipulations, etc. are performed on such models depending on the provided modeling language, its semantics, its associated verification procedure and on the abstraction level where models are defined. As a consequence, different models of a same system may be produced along the design process. Thus, several heterogeneous models expressed with different modeling languages are produced. That heterogeneity may lead to ambiguities in the interpretation of the system characteristics and/or behavior.

2.1.3 Domain Modeling: Ontologies versus System Modeling: Design Models

In [31], authors have proposed a comparison of ontologies and design models. Modeling languages are required to express both ontologies and design models. These modeling languages offer different verification techniques according to their semantics. As such, both ontologies and design models are models. They define a conceptualization of a part of the subject concerned by through models defined within modeling languages. So, one may guess that from this point of view, ontologies and design models are similar, since they share a common goal, namely modeling. However, if we consider the three criteria identified above in section 2.1.1, we can identify some significant differences.

Design models are equipped with formal semantics In this sense, they fit the *formality* criterion. They are grounded on rigorously defined semantics and associated to property verification systems which use reasoning and logical theories. However, according to [31], design models are closely related to the system under design. In other words, a design model *prescribes* and *enforces* which system characteristics *will* be available in the model to perform a specific analysis or treatment. Indeed, as mentioned above, a single system may have different design models (safety oriented model, real time model, energy consumption model and an architectural model for example). From this observation, we can conclude that the *consensuality* criterion is not (or partly) fulfilled by design models. Finally, if we consider naming processes in design models and design modeling languages, there is no rule requiring a single identification of entities (variables, constants, states, events, etc.) manipulated by design models. A typical example of such a naming

rule relates to the description of point coordinates, where a pair of floats (v_1, v_2) may be interpreted differently in a model referring to polar coordinates (r, θ) and in another model referring to cartesian coordinates (x, y). The entity identification is unique in the context of the considered design model, but it may be used with a different semantics in another model. So, the *capability to be referenced* criterion is not fulfilled by design models.

Previously identified differences do not constitute a drawback. The simultaneous use of both ontologies and design models in an engineering context, strengthens the modeling processes.

The subject of this paper is not to oppose ontologies and design models. The previously identified differences advocate for the use of both ontologies and design models in a single framework. Ontologies carry relevant informations usually handled implicitly in design models. When ontologies and design models are integrated (or composed), domain properties may be explicitly used in the design models and in the associated system development processes.

2.1.4 Ontology Models

Several ontology modeling languages were developed during last ten years, as for example, Ontolingua [19] for artificial intelligence applications, DAML+OIL [13], RDFS [8] and OWL [37, 28] for Web applications, and PLIB [43, 40] for engineering applications.

Main Ontology Models Characteristics

In [18], authors have identified some relevant characteristics associated to ontology modeling languages. These characteristics have been extended with new ones in order to handle the system engineering aspects.

- **Words and concepts.** Ontology models offer the capability to describe words and concepts. Various relationships are offered by these languages: between words, between concepts and between words and concepts. The presence of such relationships leads to two ontologies design processes [31]: from words to concepts (e.g. semantic web) or from concepts to words (e.g. engineering catalogues).

- **Strong typing.** Ontology modeling languages are equipped with a type system characterizing classes, properties, relationships and domain values. According to the modeling language, this type system may be more or less a strong type system. For example, the PLIB ontology modeling language uses a strong typing system (e.g. unit types are built-in types) while description logics do not require strong typing. Types are useful for indexation, and thus for the definition of persistent frameworks like semantic databases [10, 16, 38, 27, 39, 44] to sore both ontologies and their instances.

- **Algebraic operators** may be associated to the types available in the ontology language. Operators on classes like union, intersection, etc. are available in

most of the ontology modeling languages. For example, operators on reals, are available in the PLIB ontology model. They make it possible to express property derivation (e.g. diameter equals two times a ray). These defined algebraic operators define abstract data types and give complete definition of the data types discussed above.

- **Constraints description** constructs are offered by the ontology modeling language to define constraints on classes, properties or on whole ontology. In the engineering domain, the more the constraint description language is rich, the more the expressed concepts of the ontologies are precisely described. Checking these constraints depend on the used constraint solving techniques associated to the ontology modeling language.

- **CWA v.s. OWA.** Closed world assumption (CWA) implies that a complete knowledge is known and, if a fact is not a consequence of the ontology model, then its negation is, while in open world assumption (OWA) this reasoning is no longer available. In general, CWA is used in system engineering, while semantic web considers OWA.

- **Context modeling.** In the engineering domain, the context in which a property is defined is important [41]. At the ontology level, the domain of a context dependent property is not only its class, but it is also a context description (usually a class). For example, the definition of the lifetime (property) of a tyre (class) depends on the average temperature of use (context of use). Note that PLIB offers built-in constructs for such properties.

- **Inheritance and instanciation.** Classes may be linked by single or multiple inheritance relationships. Inheritance helps to factorize objects with the same properties, it also contributes to the definition of the subsumption relationship. Instances of a class reperesent the individuals, and an individual may belong to a single class (mono-instanciation) or to several classes (multi-instanciation). Ontology modeling languages offer different forms of inheritance and instanciation. For example OWL supports multiple inheritance and multi-instancitation while PLIB supports single inheritance and mono-instanciation.

- **Reasoning.** In ontology engineering, reasoning essentially concerns subsumption (e.g. to link ontologies classes in case of integration), class membership checking (e.g. for migration of instances from one ontology class to another one) and classification (e.g. to build new class hierarchies according to some criteria). Other logical aspects of reasoning concern the specific properties of the underlying logic like symmetry, reflexivity, equivalence etc are useful for knowledge inference. Different reasoning techniques and tools have been developed. One can cite [6, 23, 24, 36], running in central memory have been defined. Like for the model checking tehnique, these approaches may face the problems of memory saturation or space exploration. Other reasoning technique more commonly used by formal methods have also been set up to handle proof of properties in ontologies. These approaches, which proved scalable, use

theorem provers like COQ with [4, 15] or Event-B [2] to infer ontologies properties.

- **Exchange formats**. All the ontology modeling languages offer exchange formats based on the XML language. When expressed in this exchange format, classes and their instances can be interpreted in the in different contexts of use.

2.1.5 Ontologies in Engineering

Our work does not address semantic web applications. In our study of design models, we have been involved in the engineering area. We focus on domain ontologies where the whole knowledge on the domain is described in the provided ontology. Due to the system engineering targeted application domain, we use ontologies conforming to the PLIB ontology model [3, 22, 42, 29]. This ontology model advocates the use of strong typing with a rich type system (similar to the one of a programming language more specific types like units), property derivation with algebraic operators corresponding to the defined types, first order logic and set theory as a constraint language, CWA and context dependent properties.

Like in usual engineering practices and unlike OWL, additional models may be added to a technical object description. Indeed, a set of different functional models, each one representing a particular discipline-specific representation (e.g., safety, real time, energy consumption, geometry procurement, simulation, etc.) can be ssociated to a given technical object described within the PLIB ontology model.

Finally, a number of domain ontologies based on this model already exist. Examples are the ISO 13584 and ISO 15926 (e.g. mechanical fasteners, measure instruments, cutting tools) and IEC 61360 (e.g. electronic components, process instruments) series of ontologies developed within international standardization organizations (e.g. ISO, IEC) or national ones (e.g. JEMIMA[3] CNIS[4]) that cover progressively all the technical domain.

2.1.6 Ontologies and Annotations

One of the main usage of ontologies is annotation. Let us consider a set of entities available in a given copus. These entities may be words or sentences in a document, images or videos, entities of a design model, etc. By annotation, we denote the link that may exist between an ontology concept (class, instance, property, etc.) and an entity of the considered corpus. The annotation process consists in defining and running a set of rules leading to the production of annotations. This process may be completely automated, semi-automatic with user validation or completely interactive. Automatic annotation proved powerful in the area of the semantic web and natural language processing where the entities of the corpus are words appearing in texts. Several tools (or annotators) have been developed for various ontologies

[3]Japan Electric Measuring Instruments Manufacturers Association,
[4]Chinese Institute for Standardization

and different natural languages [7, 14, 17, 25, 26]. Other approaches to annotate images and multi-media documents have also been developed [11].

In the area of system design, the objective of model annotation is to increase model interoperability. Consensual domain ontologies are shared by different system models corresponding to different engineering views. Annotations allow the designer to link different entities of different system models to ontology concepts. Reasoning at the ontology level makes it possible to check some domain properties. Model annotations have been produced using semi-automatic and/or interactive approaches. Automatic annotation is not recommended in such application domains. For example, model annotations have been produced for product life management (PLM) models in [34], for petroleum engineering models in [35, 5] or for aircraft system modeling in [46]. All these examples used a controlled annotation techniques being either semi-automatic or interactive.

2.1.7 Ontologies and Formal Models

The previous sections presented an overview of the fundamental characteristics of ontologies and ontology modeling languages. We have shown that different ontology languages can be set up to describe domain ontologies. Application domains, semantics, constraints expressiveness, reasoning capabilities, assumption on the universe of discourse etc. are some of the characteristics to be assessed before designing a domain ontology.

Moreover, when an ontology modeling language (with its own $\models_O$ and $\vdash_O$) and a design modeling language (with its own $\models_M$ and $\vdash_M$) are used concurrently, there is a need of *semantic alignment* due to different semantics of these two languages. This topic is outside the scope of this paper.

In the remainder of the paper, we consider that domain ontologies are described within set theory and first order logic. We show that such domain ontologies are useful to strengthen formal system development and verification in the engineering domain. More precisely, we study two illustrative case of formal modeling: model checking techniques and a proof and refinement based technique with Event-B. Both of these two modeling languages are expressed with set theory and first order logic as well. Thus, the semantic mismatch no longer present in our case. We will use $\models$ and $\vdash$ to denote the satisfaction and entailment relationships for ontology and design models modeling languages.

Chapter 3

Annotations to bridge ontologies and system design

3.1 What if the ontology and the design model were linked ?

Usually, design models do not handle, in an explicit manner, the knowledge of the application domain or context where models are designed. Therefore, some useful properties available in the domain knowledge are not considered by the design models, more, these properties could be violated by the design model. For instance, the nose gear velocity system measures the velocity on the ground of an aircraft. To do so, it uses a 16-bit register variable to store the number of cycles of the wheels (to compute the distance travelled and the aircraft speed). Thereofore, it is important to *know* the maximal length of the runway on earth in order to check that the chosen size of the register variable (16-bit) is correct. In terms of system engineering, the determined size of this register comes from flight mechanics, more precisely from the lift of the aircraft. Without an explicit definition of this knowledge, engineers would not be able to set up such a value for the register size.

So, linking knowledge domains, expressed by ontologies, with the design models strengthens the designed models and support more verifications, since the properties expressed in the ontologies will be part of the designed models. Moreover, thanks to the capability to reference domain entities, it becomes possible to avoid ambiguous definitions of the same entity in two different models.

Model annotation is the mechanism classically set up to link domain ontologies with design models. It consists in defining specific relationships to relate ontology concepts with models entities. The annotation mechanism extends the one that has been defined by the semantic web community to annotate web pages [7, 14, 17, 25, 26] or images [11].

3.2 Modeling languages

Different modeling languages may be used for building both ontologies, design models and annotation models. These languages may have the same semantics and

verification procedure, but these may be different. Two situations occur. First, as mentioned in section 2.1.7, the semantics and the verification procedure can be expressed in a single modeling language. In this case, there is no semantic mismatch and both design models, ontologies and annotation can be formalized in the same modeling language. The second option considers different semantics of both modeling languages. This case is out of the scope of this paper, it requires semantic alignment. Several approaches to align semantics have proposed in the literature, they are based on the definition of institutions as models [20, 12, 12, 33].

3.2.1 Our approach

Our approach advocates the exploitation of domain knowledge, carried out by ontologies, in design models. We propose a stepwise methodology, composed of four steps, to establish a formal link between these two models. The approach is based on the definition of an annotation mechanism that represents this link. The definition of this mechanism depends on the used modeling languages for both ontologies and design models. Fig. 3.1 shows the overall schema of the approach.

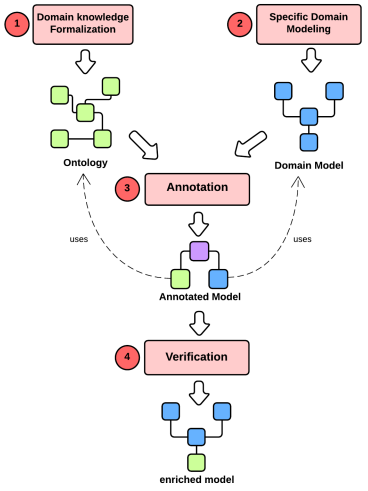


Figure 3.1: A four steps methodology for integrating domain knowledge and design models.

1. **Formalization of Domain Knowledge.** Domain information are formalized in an ontology modeling language. Concepts, entities, relationships, constraints, rules, etc. are explicitly defined. The result is a formal ontology expressed in the chosen ontology modeling language. The semantics of this language and the associated verification techniques are used to establish properties of the ontology. The expressive power of this language has an impact on the defined ontologies (e.g. different constraint description languages may be used). Finally, the ontology shall be defined independently of any context

of use. It may also be built from already existing ontologies (e.g. standard ontologies).

2. **Definition of design models.** Any formal modeling language is used to describe design models. Within this formal modeling language, users define and formalize specific design models corresponding to a given specification. Different analyses allowed by the modeling language and its associated verification technique may be performed on the designed model.

3. **Annotation of design model by references to ontologies.** Using specific mechanisms available in the formal modeling language, annotation of design models are explicitly described. Annotation consists of defining specific relationships between between design models entities and ontology concepts. Different relationships are available, they have their specific properties. For example, the Is_a relationship can be used to assert that a given design model entity *is a*n ontology concept (annotation by *subsumption*). Annotation is made explicit in the design models thanks to the use of these relationships.

4. **Expression and verification of properties.** Once design models are annotated by domain ontologies, the proof context of the design models is enriched by the domain properties expressed in the ontology. It becomes possible to check on the one side the consistence of the design models already established before annotation (they may be no longer correct after annotation) and on the other hand other properties that emerged after annotation.

At the end of this process, a new design model enriched with new information of the domain knowledge is obtained. This model makes an explicit representation of domain concepts and properties borrowed from the ontology thanks to the annotation.

3.2.2 Some Comments

1. It is important that the specified and used ontologies are defined in a consensual manner by the stakeholders involved in the system under design. Moreover, they should have relationships with the domain of the design model.

2. Steps 1 and 2 of the previous methodology are independent. They may be run in parallel. Ontologies may be defined prior to the design model or they may preexist.

3. In the semantic web area, lot of efforts are devoted to the definition of automatic annotation mechanisms [7, 14, 17, 25, 26, 11]. There the annotated models are documents in general and ontologies usually exploit terms rather than concepts. The definition of the annotations may be realized either by manual, semi-automatic or automatic processes[34, 35, 5, 46, 2]. In this paper, we are concerned with formal design models targeting system design. Therefore, our approach relies on an interactive annotation performed by the designer.

3.2.3 Associated theory

Following the previously defined stepwise methodology to make explicit domain knowledge expressed by ontologies in design models, we propose a general formal setting in which such a methodology can be deployed for specific formal methods.

1. **Formalization of Domain Knowledge.** An ontology is described with an ontology modeling language. It defines axioms $A_{O_1}, \ldots, A_{O_m}$ and proof deduction rules rules from which properties i.e. theorems $T_{O_1}, \ldots, T_{O_n}$ may be deduced.

   - Ontologies shall be sound (healthiness of the ontology). This means that there exists a model M_O that satisfies the axioms of the ontology. We write $M_O \models A_{O_i}$ for $0 \leq i \leq m$
   - Each theorem can be deduced from the axioms and the other theorems. We write $A_{O_1}, \ldots, A_{O_m} \vdash T_{O_1}$ and $A_{O_1}, \ldots, A_{O_m}, T_{O_1}, \ldots T_{O_{i-1}} \vdash T_{O_i}$ for all $2 \leq i \leq n$

2. **Definition of design models.** The studied systems are described in the chosen modeling language. If the modeling language supports properties verification, then properties may be expressed and checked. Axioms $A_1, \ldots A_k$ and theorems $T_1, \ldots T_l$ describing the model properties are defined.

   - Described system models shall be sound (healthiness of the design model). This means that there exists a model M_D that satisfies the axioms defined for the system model. We write $M_D \models A_i$ for $0 \leq i \leq k$
   - Each theorem expressing properties on the design model can be deduced from the axioms and the other demonstrated theorems. We write $A_1, \ldots, A_k \vdash T_1$ and $A_1, \ldots, A_m, T_1, \ldots T_{i-1} \vdash T_i$ for all $2 \leq i \leq l$

3. **Annotation of design model by references to ontologies.** Annotation consists in integrating domain knowledge expressed by ontologies in the design model.

   - Integrated axioms define a sound annotation (healthiness of the annotated model). There exists a model satisfying the axioms of both the ontology and the design model. We write $M \models A_i \wedge A_{O_j}$ for $0 \leq i \leq k$ and $0 \leq j \leq m$

4. **Expression and verification of properties.** The properties $T_1, \ldots, T_l$ shall be re-proved again once the model has been enriched by ontologies. Moreover, new emerging properties $P_1, \ldots, P_t$ may be inferred from the annotated model.

   - The properties of the design model before annotation need to be re-proved again. Indeed, the ontology may have brought relevant information that falsify 0 or more properties. We write $A_1, \ldots, A_k, A_{O_1}, \ldots, A_{O_m} \vdash T_1$ and $A_1, \ldots, A_k, A_{O_1}, \ldots, A_{O_m}, T_{O_1}, \ldots T_{O_n}, T_1, \ldots T_{i-1} \vdash T_i$ for all $2 \leq i \leq l$

- When domain knowledge described by ontologies is embedded in the design models, new properties $P_1, \ldots P_t$ may arise. They should be proved. We write: $A_1, \ldots, A_m, A_{O_1}, \ldots, A_{O_m}, \vdash P_i$ for all $0 \leq i \leq t$

Remark. As mentioned in section 2.1.7, we have assumed that the same deduction logic (with $\vdash$ and $\models$) is associated to both the ontology and the design models. If this is not the case, alignment of the semantics of the ontologies and of the models should be performed. This is out of the scope of this paper.

3.3 An illustrating example

To demonstrate how our approach works, we have chosen to use a simple illustrative example. This example is related to the management of *students diplomas*. Below, we show the set up design model and the used ontology. They will be reused, later in this report, at each step of the proposed approach.

Remark. We have chosen to use the UML class diagrams to describe the different models used in this report. However, this choice is not restrictive and the proposed method can be set up for any modeling language. Particularly, in the case of the IMPEX project, we target the use of the Event-B method.

3.3.1 Design model for students registration

In this example, we consider an information system describing a set of students. The information system considers students registered for different diplomas like *Bachelor, Engineer, Master, PhD, Licence, etc.* Each student is characterized by the last hold diploma and the prepared diploma (next diploma). Moreover, this information system is constrained by the fact that a student cannot register for a PhD if he/she does not hold a master degree.

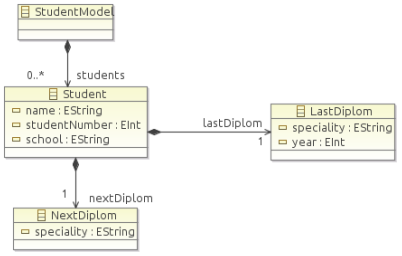


Figure 3.2: The *Students* design model

Figure 3.2 depicts a UML class diagram describing a part of the information system related to the management of students. In this model, a student holds a diploma (degree) represented by the last graduation diploma he obtained. A student, represented by the *Student* class with the attributes *name, studentNumber* and *school*, representing his name, his student number and the school where he/she

studies. The *LastDiplom* class describes the last diploma hold by a student, with *speciality* and *year* attributes for the chosen speciality and a year of graduation. Last, the *NextDiplom* class describes the next diploma a student is willing to prepare.

Moreover, a constraint named *phdInscritpion* on the *NextDiplom* class is defined. It represents a model invariant associated to this design model. It asserts that a student registering for a *PhD* diploma needs to hold a *master* diploma to be allowed to register for a PhD.

```
class NextDiplom
  {
   attribute speciality : String[?];
   invariant phdInscritpion:
        if self.speciality.equalsIgnoreCase('phd') then
               self.oclContainer().oclAsType(Student).
               lastDiplom.speciality.
          equalsIgnoreCase('master')
else
    true
endif;
}
```

This design model *prescribes* the following.

1. It considers attributes represented by the datatype *String*. Such a typing can be considered as a weakness of the model. Errors due to misspelling of the values of the attributes can lead to errors. A *Diploma* class describing classes would have improved this model.

2. It considers that each student of the model already holds a diploma and is willing to prepare another one.

3. Finally, it defines a specific constraint requiring that each student willing to register for a *PhD* needs to hold a master (*phdInscritpion* constraint).

3.3.2 An ontology of diplomas

Diplomas and their characteristics represent a central knowledge for the previously defined model. A knowledge model to *describe* the diploma knowledge through diplomas characteristics, rules and constraints can be defined. Several candidate ontologies are possible. Below, we show two examples of such ontologies.

Figure 3.3 shows part of the *Diploma* ontology representing existing diplomas. This ontology represent different types of knowledge.

- *Structural*. It uses the subsumption relationship (here represented by the *is_a* relationship) to define two categories of diplomas. *LMDDiplom* and *ClassicalDiplom* describing respectively the *Licence, Master* and *PhD* diplomas and the other diplomas like an *Engineer* diploma.

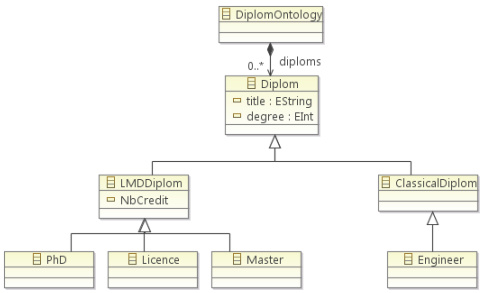


Figure 3.3: The *Diplomas* ontology

- *Descriptive.* Several descriptive attributes, like *title, degree* in the *Diplom* class describing the name and the level of a given diploma and *NbCredit* describing the credit number required for each diploma, are defined.
- *Behavioral.* A nonstructural constraint on the model says that the class *Master* is equivalent to the class *Engineer*. It is written in the ontology modeling language as

  ```
  Equivalent_Class (Master, Engineer).
  ```

  In UML, this constraint is represented by *equivalent* class linking the *Master* and *Engineer* classes of the same ontology.

Another possible ontology is depicted on figure 3.4. This ontology describes explicitely that the class *PhD* is associated to the *Master* class through the *required* association. Note that the semantics of this association describes that to hold a PhD, it is needed to hold a master degree. This constraint cannot be verified at this level. It is given to describe this constraint.

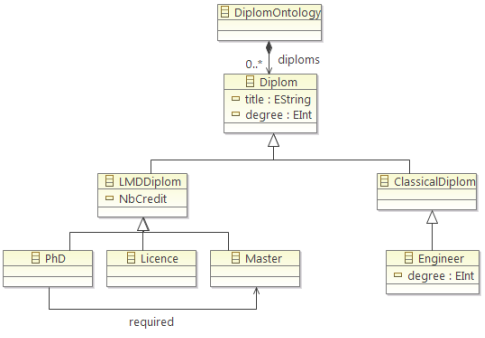


Figure 3.4: The *Diplomas* ontology: second version

3.3.3 Link of ontology and design model

The constraint stating that, to register for a PhD, a student must hold a master degree, on the design model is too restrictive. Indeed, when processing the design

model, there is no way to register an *Engineer* for preparing a *PhD*, since there is no information, in the design model, stating that an Engineer diploma is equivalent to a Master diploma.

This information can be obtained from the ontology. If the design model was able to exploit the properties expressed by the ontology, the design model would accept Engineers to register for a PhD.

3.3.4 Modeling languages

Different modeling languages may be used for building both ontologies, design models and annotation models, leading to heterogeneous models. In order to integrate both models in a single setting, two solutions are possible. The first one consists in using a single modeling language where all the models are described. The second one consists in using a single modeling language supporting meta-modeling capabilities. Then, each modeling language is described as a specific meta-model.

In this work, the developed approach uses the second option. Model Driven Engineering techniques are set up. Meta-models of each manipulated models are defined in order to define an annotation model in an uniform setting, and to ease the prototyping.

3.4 Model annotation: three cases

In step 3, relations, defining model annotations, are established between the design model entities and the ontology concepts. Three annotation mechanisms are identified. Annotations map design model entities (classes, properties, datatypes, associations, etc.) and ontology concepts (classes, properties, associations, etc.).

3.4.1 Annotation by inheritance using the Is_a relationship

The *Is_a* relationship, defines a subsumption relationship [30]. In this case, a concept of the ontology subsumes an entity of the design model. The mapping relationship is a subsumption (*is_a*) relationship. All the properties, attributes, rules and constraints that apply to the ontological concept become applicable for the design model entity.

The annotation by inheritance maintains the ontological reasoning and preserves it in the business model. But, note that due to the inheritance of all the resources issued from the ontological concept, all these resources are expressible at model level but some of them may not be valuable after annotation. This relationship is usually defined in an *a priori* setting where the design models assume the availability and existence of ontologies.

Figure 3.5 shows how the *LastDiplom* class of the *students* design model is annotated by the *Engineer* class of the *Diplomas* ontology.

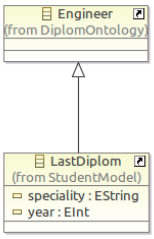


Figure 3.5: Annotation by inheritance using the *Is_a* relationship.

3.4.2 Annotation by partial inheritance using the Case_of relationship

The *Is_case_of* relationship is also a subsumption relationship. It defines a partial inheritance relationship [30]. This relation behaves like the *Is_a* relationship, except that it does not require that *all* the properties and constraints to be inherited. Indeed, only some of the properties and of the constraints of the ontology class are imported. The annotation mechanism is in charge of selecting which properties and constraints are imported.

Some of the domain restrictions (constraints) formalized in the ontological classes participating to the annotation may not be expressible in the model if the properties they are related to are not valuable in the model.

The main advantage of this approach is flexibility, it can be set up *a posteriori*.

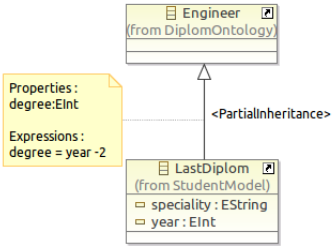


Figure 3.6: Annotation by partial inheritance using the *Is_case_of* relationship.

Figure 3.6 shows how the *LastDiplom* class of the *students* design model is annotated by the *Engineer* class of the *Diplomas* ontology using partial inheritance. Here the property *degree* of the *Engineer* class of the ontology is mapped with the *year* property of the *LastDiplom* class of the design model using an algebraic relation stating that $degree = year - 2$.

3.4.3 Annotation by association

is_a and *Is_case_of* relationships are built-in relationships usually available in the ontology modeling languages. It may happen that some annotations use specific relationships defined by the users. These relationships are themselves characterized by

ontologies. The process enables the connection between the ontological classes and model classes by association and eventual specification of the properties relations.

In this case, the ontological reasoning contained in the knowledge model can not be preserved at the annotated design model. The properties of the annotation shall be borrowed from the ontology which defines this annotation relationship.

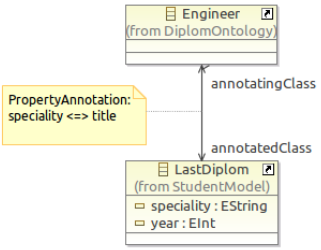
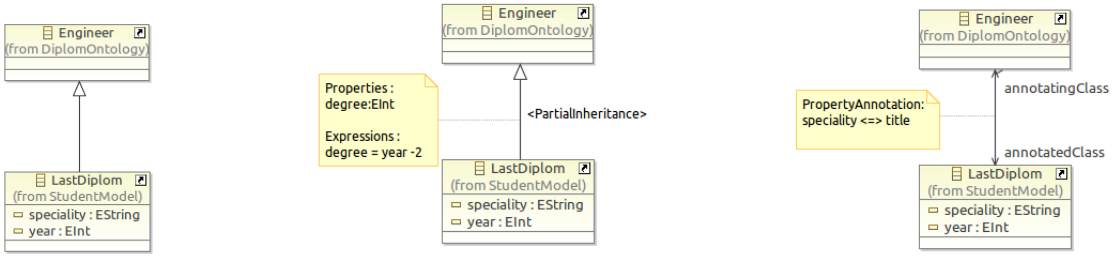


Figure 3.7: Annotation by association

Figure 3.7 shows how the *LastDiplom* class of the *students* design model is annotated by the *Engineer* class of the *Diplomas* ontology. Here, the property *title* of the *Engineer* class of the ontology is mapped with the *speciality* property of the *LastDiplom* class of the design model using an equivalence relation.

3.4.4 Annotation of the case study

Figure 3.8 illustrates the three identified possible annotations of design models.



(a) Annotation by inheritance

(b) Annotation by partial inheritance

(c) Annotation by association

Figure 3.8: Annotations mechanisms

The annotation process may be manual, semi-automatic or an automatic one. In the case of the association based annotation, an ontology of annotation relations is required. Moreover, the associated reasoning shall be specified and carried by the definition of these relations.

3.4.5 Annotation meta-models

As mentioned above, the annotation mechanisms shall be described in the modeling language. We recall that we have chosen the UML class diagrams as a modeling language. A consequence of this choice, is the use of the *Is_a* relationship. It is a built-in relationship and does not need to be re-defined. The *Is_Case_of* and *Association* annotation relations need to be defined within the modeling language (here UML). Two meta-models (one for each type of annotation) describing these mechanisms are defined. They link design model entities and ontology concepts at the meta-model level.

The meta-model for annotation by Is_Case_of

Figure 3.9 shows the defined meta-model for describing the *Case_of* based annotation.

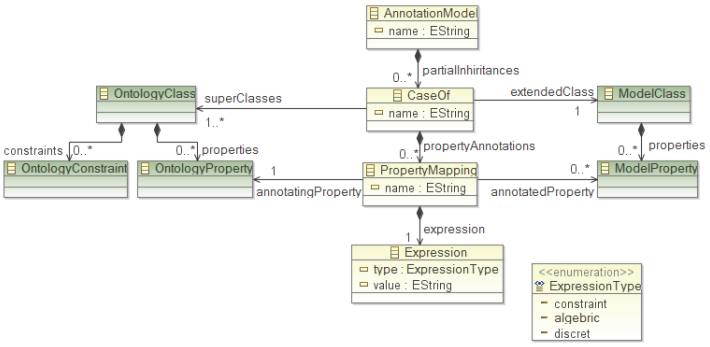


Figure 3.9: Annotation by *Case_of* meta-model

- *AnnotationModel* represents the entry point of the annotation models.
- The *CaseOf* concept represents the *Is_Case_of* relations that may exist between design model entities and ontology classes.
- *PropertyMapping* class models the relations between design model properties and ontology properties.
- *ModelClass* references the Class entity of a design model.
- *OntologyClass* references the Class concept of an ontology.
- *Expression* class represents the type of the relation that may exist between ontology and design model properties. They may be of different types: *algebraic* (e.g. X = Y+5), *discrete* (e.g. equivalence between clothes sizes like S $\Longleftrightarrow$36 , M$\Longleftrightarrow$38 and L$\Longleftrightarrow$40) or *constraint* (e.g. X+Y $\geq$ Z). This meta-model does not give the whole expression language. This language can be given by a classical class diagram describing expressions.

The meta-model for annotation by Association

Figure 3.10 shows the proposed meta-model for describing the *Association* based annotation.

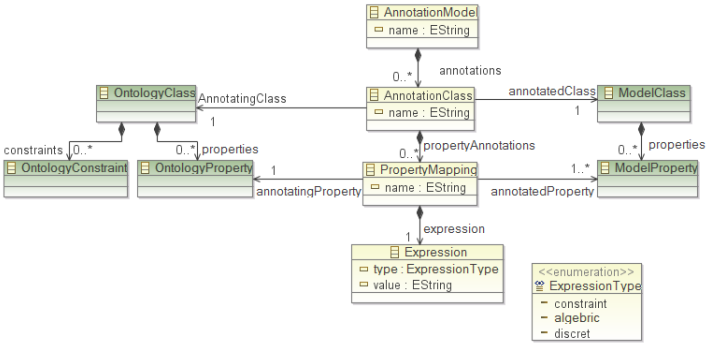


Figure 3.10: Annotation by *Association* meta-model

- *AnnotationModel* represents the entry point of the annotation models.
- *ClassAnnotation* represents an association relation between ontology concepts and design model classes (class A is associated to class B).
- *PropertyMapping* represents the relations between design model properties and ontology properties.
- *Expression*, similarly to the *Is_case_of* annotation, makes reference to the type of correspondence between the properties (algebraic, discrete or constraint).

3.5 Properties expression and verification

The last step of the approach analyzes the obtained annotated design models through formally established links with the ontology. This annotation process leads to the enrichment of the original design model with new relations, properties, constraints and rules. Ontological properties and classes are considered to be available in the enriched model if they have been selected or linked to model properties during the annotation process (third step the approach). It may happen that these relations, properties and constraints could not be expressed at the design model level and thus not evaluable at instantiation level due to the absence of attributes to express them and of the values of these attributes (instances). These constraints become meaningless. At this level, an analysis of the obtained relations, properties, constraints and rules issued from the annotation is necessary after an annotation by *Is_Case_Of* or by *association* because these two types of annotation offer the possibility of having only certain ontological properties in the enriched design model. The annotation by *is_a* does not suffer from this drawback since all ontological constraints in the design model can be expressed (all the properties of the annotating ontological classes are inherited in the design model).

Chapter 4

Conclusion and future work

4.1 Conclusion

This report gives an overview of domain knowledge modeling within ontologies. We have shown that different ontology models are available. A set of characteristics has been made explicit, they allow users to make a clear distinction between ontology models. In a second part, we have presented a general setting defined to handle explicitly domain knowledge in design models. The realized work aims at making explicit and integrating domain knowledge in design models. In general, these properties are not explicit when design models are developed. They are implicitly handled by the engineers through a set of hypotheses. The objective of the defined setting is to enrich these design models with domain knowledge in order to embed new domain properties in these design models and verify these properties. A simple example has been used to illustrate the defined approach.

The main interests of this approach can be summarized in the following points.

- Asynchronous evolution of both the ontology and the design models. Indeed, ontologies and models may evolve independently.
- There is no modification of the annotations in case the ontologies evolve.
- The approach supports an incremental design process where annotations can be added incrementally.
- The developed approach relies on semi-formal techniques based on the UML modeling language and model driven engineering techniques.
- The defined ontology is a simple one. There is a need to define ontologies corresponding to the avionic domain, by domain experts.

4.2 Ongoing work

The next step consists in defining formal models allowing to handle formal verification techniques and make it possible to handle explicit domain knowledge in such

formal models. The interest of this work would be to evaluate if formal methods can be handled by this approach in case design models are formally described.

Another important work consists in describing domain ontologies that stick to different application domains. It is also important to formalize the domain information contained formatting tables, glossaries, standards and informal descriptions.

Bibliography

[1] A free, open-source ontology editor and framework for building intelligent systems, howpublished = `http://protege.stanford.edu/`.

[2] Y. Aït-Ameur, J. P. Gibson, and D. Méry. On implicit and explicit semantics: Integration issues in proof-based development of systems - version to read. In *Leveraging Applications of Formal Methods, Verification and Validation. Specialized Techniques and Applications - 6th International Symposium, ISoLA 2014, Imperial, Corfu, Greece, October 8-11, 2014, Proceedings, Part II*, volume 8803, pages 604–618. Springer Verlag, 2014.

[3] Y. Ait-Ameur and H. Wiedmer. *General resources*. ISO-IS 13584-20. ISO Genève, 88 pages, 1998.

[4] P. Barlatier and R. Dapoigny. A type-theoretical approach for ontologies: The case of roles. *Applied Ontology*, 7(3):311–356, 2012.

[5] N. Belaid, S. Jean, Y. Aït-Ameur, and J. Rainaud. An ontology and indexation based management of services and workflows application to geological modeling. *IJEBM*, 9(4):296–309, 2011.

[6] S. E. P. Bijan. Pellet: An owl dl reasoner. In *International Workshop on Description Logics (DL2004)*, pages 6–8, 2004.

[7] K. Bontcheva, V. Tablan, D. Maynard, and H. Cunningham. Evolving GATE to Meet New Challenges in Language Engineering. *Natural Language Engineering*, 10(3/4):349–373, 2004.

[8] D. Brickley and R. V. Guha. *RDF vocabulary description language 1.1: RDF schema*. W3C Recommendation 10, 25 February 2014. Available at `http://www.w3.org/TR/rdf-schema/`.

[9] J. Broekstra and A. Kampman. SeRQL: An RDF query and transformation language. In *SWAD Europe Workshop on Semantic Web Storage and Retrieval*, 2004.

[10] J. Broekstra, A. Kampman, and F. v. Harmelen. Sesame: A generic architecture for storing and querying rdf and rdf schema. In *Proceedings of the First International Semantic Web Conference on The Semantic Web*, ISWC '02, pages 54–68, London, UK, UK, 2002. Springer-Verlag.

[11] A. Chebotko, Y. Deng, S. Lu, F. Fotouhi, and A. Aristar. An ontology-based multimedia annotator for the semantic web of language engineering. *Int. J. Semantic Web Inf. Syst.*, 1(1):50–67, 2005.

[12] M. Codescu, T. Mossakowski, and O. Kutz. A categorical approach to ontology alignment. In P. Shvaiko, J. Euzenat, M. Mao, E. Jiménez-Ruiz, J. Li, and A. Ngonga, editors, *Proceedings of the 9th International Workshop on Ontology Matching collocated with the 13th International Semantic Web Conference (ISWC 2014), Riva del Garda, Trentino, Italy, October 20, 2014.*, volume 1317 of *CEUR Workshop Proceedings*, pages 1–12. CEUR-WS.org, 2014.

[13] D. Connolly, I. Horrocks, D. McGuinness, F. Patel-Schneider, and A. Stein. Daml+oil reference description. *World Wide Web Consortium*, 2001.

[14] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, N. Aswani, I. Roberts, G. Gorrell, A. Funk, A. Roberts, D. Damljanovic, T. Heitz, M. A. Greenwood, H. Saggion, J. Petrak, Y. Li, and W. Peters. *Text Processing with GATE (Version 6)*. 2011.

[15] R. Dapoigny and P. Barlatier. Modeling ontological structures with type classes in coq. In *Conceptual Structures for STEM Research and Education, 20th International Conference on Conceptual Structures, ICCS 2013, Mumbai, India, January 10-12, 2013. Proceedings*, volume 7735 of *Lecture Notes in Computer Science*, pages 135–152. Springer, 2013.

[16] H. Dehainsala, G. Pierra, and L. Bellatreche. Ontodb: An ontology-based database for data intensive applications. In *Proc. of the 12th Int. Conf. on Database Systems for Advanced Applications (DASFAA'07). LNCS*. Springer, 2007.

[17] S. Desprès and S. Szulman. Terminae method and integration process for legal ontology building. In M. Ali and R. Dapoigny, editors, *Advances in Applied Artificial Intelligence, 19th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2006, Annecy, France, June 27-30, 2006, Proceedings*, volume 4031 of *Lecture Notes in Computer Science*, pages 1014–1023. Springer, 2006.

[18] C. Fankam, Y. Aït-Ameur, and G. Pierra. Exploitation of ontology languages for both persistence and reasoning purposes - mapping plib, OWL and flight ontology models. In *WEBIST 2007 - Proceedings of the Third International Conference on Web Information Systems and Technologies, Volume WIA, Barcelona, Spain, March 3-6, 2007.*, pages 254–262. INSTICC Press, 2007.

[19] A. Farquhar, R. Fikes, and J. Rice. The Ontolingua Server: a Tool for Collaborative Ontology Construction. *International Journal of Human Computer Studies (IJHCS)*, 46(6):707–727, 1997.

[20] J. Goguen and R. Burstall. Introducing institutions. In E. Clarke and D. Kozen, editors, *Logics of Programs*, volume 164 of *Lecture Notes in Computer Science*, pages 221–256. Springer Berlin Heidelberg, 1984.

[21] T. R. Gruber. Towards Principles for the Design of Ontologies Used for knowledge sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*. Kluwer Academic Publisher's, 1993.

[22] P. Guy, A.-A. Yamine, and S. Eric. ISO (660p), GenÃve, 2003.

[23] V. Haarslev and R. Möller. Description of the RACER system and its applications. In *Working Notes of the 2001 International Description Logics Workshop (DL-2001), Stanford, CA, USA, August 1-3, 2001*, volume 49 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2001.

[24] V. Haarslev and R. Möller. RACER system description. In *Automated Reasoning, First International Joint Conference, IJCAR 2001, Siena, Italy, June 18-23, 2001, Proceedings*, volume 2083 of *Lecture Notes in Computer Science*, pages 701–706. Springer, 2001.

[25] S. Handschuh and S. Staab. CREAM: creating metadata for the semantic web. *Computer Networks*, 42(5):579–598, 2003.

[26] S. Handschuh, R. Volz, and S. Staab. Annotation for the deep web. *IEEE Intelligent Systems*, 18(5):42–48, 2003.

[27] S. Harris and N. Gibbins. 3store: Efficient bulk RDF Storage. In *Proceedings of the 1st International Workshop on Practical and Scalable Semantic Systems (PPP'03)*, pages 1–15, 2003.

[28] P. Hitzler, M. Krötzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph, editors. *OWL 2 Web Ontology Language: Primer*. W3C Recommendation, 27 October 2009. Available at `http://www.w3.org/TR/owl2-primer/`.

[29] ISO13584-42. Industrial automation systems and integration parts library part 42 : Description methodology : Methodology for structuring parts families. Technical report, International Standards Organization, 1998.

[30] S. Jean. *OntoQL, an exploitation language for ontology-based databases*. Theses, Université de Poitiers, Dec. 2007.

[31] S. Jean, G. Pierra, and Y. Ait-Ameur. Domain Ontologies: A Database-Oriented Analysis. In *Web Information Systems and Technologies, International Conferences, WEBIST 2005 and WEBIST 2006. Revised Selected Papers*, Lecture Notes in Business Information Processing, pages 238–254. Springer Berlin Heidelberg, 2007.

[32] H. Knublauch, R. W. Fergerson, N. F. Noy, and M. A. Musen. The protÃ©gÃ© owl plugin: An open development environment for semantic web applications. pages 229–243. Springer, 2004.

[33] C. Lange, O. Kutz, T. Mossakowski, and M. Grüninger. The distributed ontology language (DOL): ontology integration and interoperability applied to mathematical formalization. In J. Jeuring, J. A. Campbell, J. Carette, G. D. Reis,

P. Sojka, M. Wenzel, and V. Sorge, editors, *Intelligent Computer Mathematics - 11th International Conference, AISC 2012, 19th Symposium, Calculemus 2012, 5th International Workshop, DML 2012, 11th International Conference, MKM 2012, Systems and Projects, Held as Part of CICM 2012, Bremen, Germany, July 8-13, 2012. Proceedings*, volume 7362 of *Lecture Notes in Computer Science*, pages 463–467. Springer, 2012.

[34] Y. Lu, H. Panetto, Y. Ni, and X. Gu. Ontology alignment for networked enterprise information system interoperability in supply chain environment. *Int. J. Computer Integrated Manufacturing*, 26(1-2):140–151, 2013.

[35] L. S. Mastella, Y. Aït-Ameur, S. Jean, M. Perrin, and J. Rainaud. Semantic exploitation of engineering models: An application to oilfield models. In A. P. Sexton, editor, *Dataspace: The Final Frontier, 26th British National Conference on Databases, BNCOD 26, Birmingham, UK, July 7-9, 2009. Proceedings*, volume 5588 of *Lecture Notes in Computer Science*, pages 203–207. Springer, 2009.

[36] B. Motik. KAON2 - scalable reasoning over ontologies with large data sets. *ERCIM News*, 2008(72), 2008.

[37] W. OWL Working Group. *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation, 27 October 2009. Available at `http://www.w3.org/TR/owl2-overview/`.

[38] Z. Pan and J. Heflin. Dldb: Extending relational databases to support semantic web queries. In *In PSSS*, pages 109–113, 2003.

[39] M. J. Park, J. H. Lee, C. H. Lee, J. Lin, O. Serres, and C. W. Chung. An efficient and scalable management of ontology. In *Proceedings of the 12th International Conference on Database Systems for Advanced Applications (DASFAA'07)*, volume 4443 of *Lecture Notes in Computer Science*. Springer, 2007.

[40] G. Pierra. Context-explication in conceptual ontologies: the plib approach. In *Proceedings of the 10th ISPE International Conference on Concurrent Engineering (CE 2003), Vol. Enhanced Interoperable Systems*, volume 26, page 2003, 2003.

[41] G. Pierra. Context representation in domain ontologies and its use for semantic integration of data. *Journal on Data Semantics*, 10:174–211, 2008.

[42] G. Pierra and E. Sardet. *ISO 13584-32 â Industrial automation systems and integration â Parts library â Part 32: Implementation resources: OntoML: Product ontology markup language*. ISO, 2010.

[43] G. Pierra and H. Wiedmer. Industrial automation systems and integrationparts librarypart 42: methodology for structuring part families. Technical report, Technical Report ISO DIS 13584-42, International Organization for Standardization, 30 May 1996. ISO/TC 184/SC4/WG2, 1996.

[44] M. Stocker and M. Smith. Owlgres: A scalable owl reasoner. In *The Sixth International Workshop on OWL: Experiences and Directions*, 2008.

[45] J. Trinkunas and Q. Vasilecas. A graph oriented model for ontology transformation into conceptual data model. *Information Technology and Control*, 36(1A), December 2007.

[46] D. S. Zayas, A. Monceaux, and Y. Aït-Ameur. Knowledge models to reduce the gap between heterogeneous models: Application to aircraft systems engineering. In R. Calinescu, R. F. Paige, and M. Z. Kwiatkowska, editors, *15th IEEE International Conference on Engineering of Complex Computer Systems, ICECCS 2010, Oxford, United Kingdom, 22-26 March 2010*, pages 355–360. IEEE Computer Society, 2010.