Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

A concept-based approach to modelling shared ontology-based models for industrial applications.

Corniere, A. * **Fortineau, V.** * **Paviot, T.** * **Lamouri, S.** *

* *Arts et Métiers Paristech, 151 bd de l'hopital, Paris, France*

Abstract: Ontologies have emerged as a new modelling paradigm in industry during the last decade. They propose new design paradigms for information modelling that engineers must be aware of and manage, whenever they aim to design consistent models. In this paper, guidelines are proposed as well as an industrial illustration in order to help modellers to anticipate and well manage the specificities of ontology modelling. An illustration based on family modelling and an industrial example about classification of parts based on their basic shapes are proposed.

Keywords: Ontology, modelling, PLM, manufacturing, concept, methodology, UML, properties, OWL, OWL 2, SWRL, OntoClean

1. INTRODUCTION

Ontologies, and especially inference ontologies based on OWL syntax have emerged the last decade as a new modelling paradigm for industrial applications. Indeed, ontology-based modelling is far different from UML-based modelling, to which most engineers are more used. Therefore, there is a need of explaining to modellers what are the specificities of inference ontologies, and how to avoid misusing these tools.

The aim of this paper is to explore the issue of modelling a system or a certain reality using inference ontologies. Systems considered are industrial products in general, but the proposed approach in this paper could be used in more generic cases: modelling business activities, knowledge management, etc. The present paper, while comparing UML-based and ontology-based paradigms, provides guidelines to modellers, in order to help them to create consistent ontology-based models for industrial applications, that can be later shared or merged between various users.

The present paper is therefore organized as follows: section 2 provides a state-of-the-art of ontology-based models for the industry and of existing methodologies. Then section 3 explains why ontologies are (should) be concept-based models, and provides guidelines to achieve such a concept-based approach. Section 4 illustrates this proposition with an industrial application. Finally, section 5 proposes a large discussion and section 6 concludes this study.

2. ONTOLOGY-BASED MODELS FOR INDUSTRIAL PURPOSE

2.1 State-of-the-Art of existing models

As explained in Fortineau et al. [2013b], there exist several ontology-based models for industrial applications. Those models can be standard models transformed into an ontology, like OntoSTEP (Barbau et al. [2012]) which is a transformation of STEP; the transformation of the Open Assembly Model (OAM) made by Fiorentini et al. [2007]; or the transformation of the Semantic Object Model (SOM) by Matsokis and Kiritsis [2011]. There also exist *ad hoc* models, which involve various phases of the product lifecycle, like the Product Design Ontology (Catalano et al. [2009]) for the design phase, OntoPDM (Panetto et al. [2012]) and PRONTO (Vegetti et al. [2011]) for the manufacturing phase and logistics activities, and the Linked Design Ontology (Kiritsis et al. [2012]) for knowledge management.

2.2 Methodologies to design ontology-based models

Most of the previous contributions follow a design methodology to create the ontology. Catalano et al. [2009] followed the On-To-Knowledge methodology (Staab et al. [2001]) which is, like most existing methodologies, a step-based approach: a procedure is given in order to help ontology designers, in the form of generic guidelines. Other well known methodologies are Anemone (Özacar et al. [2011]), Diligent (Pinto et al. [2004]), and Neon (Suárez-Figueroa [2010]) methodology. Those methodologies describe the fundamental activities to design an ontology, but very few explain how to evaluate the ontology formalization in an ontological language (like the Web Ontology Language (OWL) for instance). They only focus on consistency checking. However, there may exist several different formalizations of the same conceptualization that are consistent. But the ontology consistency does not ensure a good formalization of the model. In this paper, we therefore specifically focus on the ontology formalization.

Most of the evaluation methods proposed within the existing methodologies are empirical, based on the confrontation of the obtained taxonomy to business users approbation. However the evaluation of the content of the ontology is crucial when the model aims to be shared between various users, which is often the case in industrial applica-

tions. The OntoClean project provides a semi-automated evaluation of the ontology formalization. Guarino and Welty [2009], while studying the philosophical definition of an ontology, propose four meta-properties to evaluate the formalization: identity, unity, rigidity and dependency. The study of those properties provides an objective and semi-automated evaluation of the chosen formalization.

2.3 Requirements for designing industrial ontology-based models

It is important to explicit why ontologies have been identified as an improved and new paradigm for designing industrial models. According to the literature ()Fortineau et al. [2013b]), ontologies improve model richness, because they enable to model information from different levels of abstraction. Moreover, since they can deal with non-canonic data (i.e. an instance may belong to several classes), ontologies enable model merging and sharing: a single ontology may represent several viewpoints. As a consequence, they can improve semantic interoperability between various business systems. Moreover, as Dekkers et al. [2013] indicate, they may link knowledge-based models with existing PLM systems.

Thus, the main requirements to design ontology-based models for industrial applications are to ensure a semantic richness and to enable model sharing, reuse and merging. This last requirement extremely constrains the manner the industrial ontology is formalized, and is compliant to the primarily given definition of an ontology: "a formal specification of a shared conceptualization" (Borst and Akkermans [1997]).

In the next section, we propose an approach to design ontologies that takes into account the specificities of ontological modelling. The following method has been established based on the literature review - notably the work of Guarino and Welty [2009] - and on our own experience in designing ontologies for industrial purpose (Fortineau et al. [2013c], Fortineau et al. [2013a], Fortineau et al. [2013d]). Therefore, we invite modellers to follow a *concept-based approach* to formalize industrial ontologies and we provide an industrial use case in section 4.

3. ONTOLOGY-BASED MODELLING: A CONCEPT-BASED APPROACH

3.1 Concept-based approach VS UML approach

As explained previously, a majority of existing ontology-based models are a translation of former UML-based models. UML deals with canonic data: besides generalization properties, an instance belongs to only one class, and semantic properties are attached to specific classes (they do not exist by themselves). As a consequence, UML leads to define classes as a set of instances, from a business viewpoint: the modeller gather instances into groups (classes) corresponding to what he specifically needs to model.

Because of ontology's reasoning abilities, a similar modelling approach is often followed while formalizing an ontology. However, the UML paradigm and the ontological paradigm are different, notably due to the Open World Assumption (see section 4.1), to the definition of object properties from domains and ranges, and to non-canonic data modelling.

Moreover, the reasoning can classify instances into corresponding classes by itself . Then, it is common in the literature to apprehend the taxonomy of classes with sets and subsets, and try to figure out most behaviour from Venn diagrams. This leads to complex and domain-specific taxonomies that are not defined considering properties *essential* to a concept but in order to match the intended behaviour. Thus, most classes definitions are built beforehand and properties are set up later on. An example of this approach is the well known *Pizza ontology* that many modellers use as a starting point when learning ontologies. While it can produce consistent ontologies, it leads to design business-oriented ontologies, which is contradictory to the aim of sharing ontologies between various users. If the model has to be shared, then it needs to be as generic as possible, and concepts (classes) must be defined according to their intrinsic and objective properties. Moreover, concept definition must be robust, to suffer a possible model re-design.

We call this modelling approach a *concept-based* approach, which is based on the following statements:

- classes represent concepts that must be defined according to their essential properties;
- defined axioms are better based on object properties than on taxonomies;
- avoiding using inference abilities in order to add behaviour skills to the model;
- in an Open World, one should be aware that antagonist properties are information, but absence of data is not.

3.2 Defining classes from conceptual definitions.

The OntoClean project focuses on meta-properties of classes in an ontology. While this approach allows to verify and to validate models for consistency, it remains difficult to design a model using ontologies, partly because the Open World Assumption is difficult to manage, since we are used to manage definite sets, while classes are not all definite.

Using the class to distinguish the *essential* properties of the instances, as described in Guarino and Welty [2009] makes it relevant to the sense it has to a human, while leaving the ontology consistency to process. Based on properties only, a class may regroup (or 'fetch') instances that would have been missing if fetched through a "class intersection" filter only.

The paradigm for modelling with ontologies is very different from those of most other models. The very concept of classes is not usual : instead of the usual set disjoint from the other classes, ontologies allow for a single element to be included in several classes. They also allow classes to be subclasses of each other. Following the works of the OntoClean project, we advise to consider classes as *concepts*.

Consequently, the relation of *subduction* of classes can be considered as a *more restrictive* concept then the parent class. A sub-class represents a *kind of* its parent class. This

approach mimics the way human reasoning operates and takes advantage of the non-canonic data in ontologies (e.g. a blue cubic `Thing` can -and should- appear in classes `Cube` and its parent `Parallelepiped` and `Blue`).

3.3 Defining classes from semantic properties (instead of taxonomy)

How could we define classes in order for them to represent conceptual categories ? Our proposition is to define classes according to the essential properties (Guarino and Welty [2009]) that define them. This can be done by defining classes through axioms, and whenever possible, make these axioms rely on properties of the instances themselves.

A short example may help consider the implications of this approach: in an ontology representing the family relationships, how to propose a robust defining axiom for `GrandParent`, a class intended to regroup people known to have at least one grandchild?

There are two ways of thinking this out: create a class `Parent`, regrouping people known for having a parental relation to at least one other human, then defining `GrandParent` as the members of Parent whose children also happen to be `Parent`.

The other way is to define `GrandParent` as the group of people who have a child who has a child.

- `Parent`≡ `hasKid some Thing`
  `GrandParent`≡ `hasKid some Parent`
- `GrandParent`≡ `hasKid some (hasKid some Thing)`

Those two models are equivalent, in regard to the `GrandParent` class, while both models differ on where the information originates from: in one case, `GrandParent` depends on the class `Parent`, in the other it builds itself from the instances and their properties.

This makes a great difference: in one case, the inferred reason facts are based on the taxonomy of the ontology, more specifically the defining axiom for `GrandParent` is based on the class `Parent`; while in the other case, the object properties (i.e. the relations between individuals) let the ontology build itself *based on the information the object properties carry.* This also makes the instances data and properties easier to use in another ontology, which, as we will see later in this article, may reflect another context in which the entities have to be modelled.

Assume the definition of a `Parent` has to be broadened to model the relations to adopted children. In the first model the `GrandParent` class is also modified, whereas in the other model, the `GrandParent` class remains unchanged, its essential definition being expressed a defining axiom. In terms of model evolution, the second model is more robust.

3.4 Context-dependant reasoning

Shared ontologies, and shared data, implies for the models to be used in different contexts. According to the context, data will be given sense in, it will produce information (Tsuchiya [1993]). For example, the same data regarding a part will not be used in the same processes in a production environment as in the package design or in the product specification.

The distinction between data and information is especially relevant when dealing with open world reasoning. It is common, in automated decision making, to treat *the absence of data* as *negative information.* Ontologies though are designed to process information, which is likely to be generated by external tools and incorporated into the ontology at a later time. This same file defining the instances and the data they carry might be used by another processing environment where this data does not hold the same information.

This data might be for instance used in the design phase to represent and help out with the interactions of different sub-systems in a product, in the production context the material data and geometry information might help in production design. In the middle of the product life, the references of those same instances (parts), processed and enriched from the maintenance network, allow (with cross-referencing ontologies) to reference other products which may need maintenance. In the end of life, recycling can be designed according to all this information.

Those different contexts have to be modelled differently, and they may need different tools because they do not use the same information or reasoning paradigm [1], but they all use the same data that is relevant to the same parts.

In the previously mentioned example of a family ontology, different axioms help to verify the consistency of data: the class `Error` is intended to regroup the oddities, even though they do not make the ontology inconsistent. For example, a parent cannot be younger than any of his children. Using OWL2 & SWRL basic comparison, we detect this situation and "label" them into the class `Error`. To help the reviewing of the model we also create an object property (`WrongLink`) to trace the two instances implied in this error. This rule reflect the logical context in which we conceive a family: parents are older than their children.

```
has_Child(?parent,?kid)
  ^ age(?parent,?age1)
  ^ age(?kid,?age2)
  ^ swrlb:greaterThan(?age2,?age1)
-> Error(?parent) ^ Error(?kid)
  ^ WrongLink(?parent,?kid)
```

Another set of SWRL rules can infer blue-eyed children, using knowledge from the context of genetics that children of blue-eyed parents have blue eyes too.

```
is_Father_of(?father,?kid)
^ is_Mother_of(?mother,?kid)
^ eyes_color(?father,"blue")
^ eyes_color(?mother,"blue")
-> eyes_color(?kid,"blue")
```

Ontologies, when designed with a conceptual approach, are very suited to infer information, because of their reasoning abilities. In this case, provided there is enough information in the model to infer the colour of the `kid`'s eyes, the reasoner infers this property itself. This kind of reasoning can be very useful in an industrial environment, as it makes the information *as complete as possible* by inference.

[1] A closed-world assumption can help with managing negation for instance.

Those two SWRL rules describe how different elements in a same ontology can serve different kinds of reasoning: one is, in the context of common-sense logic, to check for errors about the `age`, while the other one models a basic rule in the context of genetics.

This in one of the reasons we advise defining classes through axioms and basing them on properties : this way a set of rules and axiom can easily be understood as elements of context defining a concept.

3.5 Elementary reasoning in the Family ontology

Ontology reasoning abilities helps make tedious or repetitive tasks automatic: to allow the reasoner to infer the ancestry of our instances in the family ontology, we introduce some rules:

- `has_Child(?parent,?kid)` `-> is_Ancestor_of(?parent,?kid)`
- `is_Ancestor_of(?x,?y) ^ is_Ancestor_of(?y,?z)` `-> is_Ancestor_of(?x,?z)` this is equivalent to making `is_Ancestor_of` transitive

We can then use this property to detect the descendants of a particular person, with a DL request:

`(Person and (is_Ancestor_of value John))` defines the class of all the persons *known to be* in `John`'s family tree. This DL expression can of course also define a class.

4. INDUSTRIAL APPLICATION : A MANUFACTURING PROCESS DECISION HELPER (SIMPLIFIED)

In this example we consider an ontology designed for manufacturing. It is constructed from geometric models, which are run through a set of geometric tests. The instances of the ontology then model parts references, and are given properties according to the results of those tests.

Some of these properties are:

- `shape_isLong`: property of a part whose envelope has one preponderant dimension in comparison to the others,
- `shape_isFlat`: property of a part whose envelope has one significantly smaller dimension than the others,
- `shape_envCyl`: boolean property of a part whose envelope is a cylinder ,
- `shape_envBox`: boolean property of a part whose envelope is a box,
- `shape_hasPlanes`: boolean property of a part presenting flat surfaces,
- `shape_planeAxes`: data properties (may be multiple) representing the directions of the normals to the planes, if any.
- `shape_width`, `shape_height`, `shape_length`: data properties (integers) representing the size of the part's shape.

From those properties, a series of classes can be constructed:

- `LathablePart` ≡ `Part and ((shape_envCyl value true) or (shape_isLong value true))`

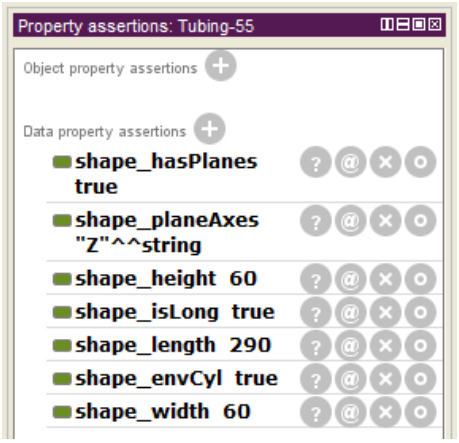

Fig. 1. Example of `shape_xx` properties for a tube

- `LathableInThreeAxesLathePart` ≡ `LathablePart and shape_planeAxes some {"X" ,"Y"}`
- `MillablePart` ≡ `Part and (shape_envBox value true)`
- `MillableNeedsFourAxesMillPart` ≡ `MillablePart and shape_planeAxes some not {"X","Y","Z"}`
- `CutoutPart` : `Part and (shape_isFlat value true)`

Those classes then gather the parts that are *eligible* to each process. One can also be more restrictive in the class defining axioms including specifics of the machines available :

```
PartMillableOn_XC300 : MillablePart and
    (shape_width some float[< 300]) and
    (shape_planeAxes some {"X","Y","Z"})
```

This class for instance regroups parts that are small enough to be processed in the XC300 mill. It also checks for planes correctly oriented for this 3-axis mill. Yet, the ontology cannot detect whether there are other planes as well, as this is a inference relevant to closed-world reasoning. This can potentially lead, in industrial applications, to undesired effects some of which are discussed later in this article.

Note the machines specific characteristics may be modelled in an ontology as well: they could be instances of the class `Machine`. Their status and specific capabilities may be used in the processes office; while their maintenance and usage history, being processed in another ontology, may help the maintenance department keep track of incoming tasks.

For many applications, ontologies are complement to databases:

while the database holds the information, ontologies are relevant in the knowledge and information management, individuals of an instance being akin to lines in the database, the reasoning ability of the ontology allows to infer information and make the information complete and consistent, even if some more information is added to the model later on.

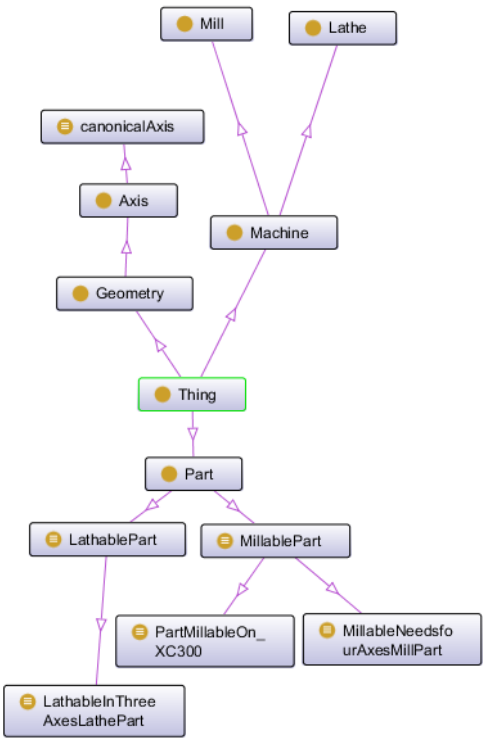

Fig. 2. Part of the class taxonomy for a manufacturing process decision helper

4.1 Some implications of the Open World Assumption in modelling

The Open World Assumption states that, in the absence of information, the reasoning can't assume there is information. This comes from the postulate that information is by nature incomplete, even if data appears to be. One benefit from this is that any inferred result in an ontology is backed with information that guarantees it (Drummond and Shearer [2006]) and guarantees it will keep its consistency whenever information is added to the model. Yet, reasoning under the Open World Assumption is fairly different from usual thinking.

Considering the family ontology introduced in part 3, and considering a class can be constructed to group the ancestry for a particular individual, let us define :

`Johns_Ancestors` ≡ `(Person and (is_Ancestor_of value John))`

In a closed-world environment, people who are not ancestors to `John` would be collected by the request :

`(Person and not (Johns_Ancestors))`

In the state of information described, no individual can be included in this class [2]. The explanation for this is that in an open world assumption where any information is susceptible to be added, there is no explicit information preventing other individuals to be included in `Johns_Ancestors`.

[2] though `John` himself can be included in this class if `is_Ancestor_of` is defined as irreflexive.

The action of listing people *known to the model* and not part of John's ancestry is by definition to be executed in the closed model of known individuals.

In a similar way, our industrial example cannot detect parts that do *not have planes* other than along X,Y,or Z. A class defined by `(not (shape_planeAxes some not {"X","Y","Z"})` is empty though not a subclass of `Nothing`.

We recommend not trying to mix paradigms in the design of an ontology, but to consider instead the use of another work environment to process information in a closed-world assumption. The use of separate reasoning for different paradigms (Open World Assumption and Closed World Assumption) makes it easier to keep the ontology model remain consistent along time and across users.

There could be explicit information providing closure to some instances (e.g. a class with a cardinality restriction), but as of now there is no way we know to provide closure of information on an instance itself,and as this would be equivalent to stating the information is complete for said instance, we would advise against it for the difficulties it would yield.

While the Open World Assumption makes some deductions impossible, the ontologies are especially apt for information management, as the reasoning facts remain consistent as more information is added to the model. This reasoning paradigm makes them a tool to consider in situations where information is likely to be added to the model along time, such as in product lifecycle management.

5. DISCUSSION

Some cases are not as manageable using ontologies than using closed-world models. We are used to work with definite sets, and this makes the Open World Assumption one of the most tangible difficulties for modellers turning to ontologies. The Open World Assumption itself doesn't allow for negative statements which can be disturbing to modellers. It is common to try working around what appears to be a limitation to the model, because the modelling paradigm is very different of the tools and techniques modellers are used to.

Sharing ontologies or merging them in such situations can be very difficult, as workarounds on one part and another can make the merged ontology inconsistent, or as the classes and instances on different branches of the classes taxonomy may not model homogeneous concepts.

In the domain of information and knowledge management thought, considering classes as models for concepts, and subsequently considering subduction as a restriction of parent concept, makes classes consistent to each other. This conceptual approach, mostly based on the instances properties, lets the reasoner infer most of the classes taxonomy. It also helps to produce consistent ontologies, according to the principles of the OntoClean project (Guarino and Welty [2009]).

We believe modellers have to waive their UML habits while modelling with ontologies, designing ontologies for information enrichment instead of behaviour design; relying on the concept a class represents instead of the set

of individuals it is expected to be; and let the model be enriched with more information instead of trying to reason on the lack of information.

However, their modelling paradigm being such, other tools may be needed to allow for the reasoner to manage information. Analysis and information-tracking tools beforehand, to provide the information needed to the model; and specific tools to perform the specific closed-world reasoning afterwards. While this implies a number of specific tools, it also seems to us that it is relevant regarding the level of abstraction of ontology-based models.

6. CONCLUSION

The emergence of ontologies in the industrial modelling area forces modellers, that are more used to UML-based models, to completely transform the way they formalize a given model. Indeed, inference ontologies provides new design paradigms, whose most crucial are: the Open World Assumption, non-canonic data, and semantic and independent object properties. This paper provides guidelines to modellers that enable to anticipate and consider those new paradigms when formalizing an ontology. More concrete explanations are then given using an industrial use case.

However, the proposed guidelines do not provide a full and automated evaluation of the obtained ontology. According to the authors, only the OntoClean approach enables an objective evaluation of ontologies, but this evaluation is limited to the ontology taxonomy and classes relationships. A more complete evaluation is a necessary perspective of the present work, that could be extremely useful to the industrial community.

Finally, a short-term perspective of the present paper is to provide and diffuse to the community a step by step tutorial to design a simple ontology (the family ontology) that follows the guidelines proposed in this paper, in order to help modellers that are new in ontology modelling. This contribution will be diffused through the IFAC TC51 Working Group on ontologies website [3].

REFERENCES

R. Barbau, S. Krima, S. Rachuri, A. Narayanan, X. Fiorentini, S. Foufou, and R.D. Sriram. OntoSTEP: Enriching product model data using ontologies. *Computer Aided Design*, 44(6):575–590, 2012.

P. Borst and H. Akkermans. An ontology approach to product disassembly. *Journal of Computer Science*, 1319:33–48, 1997.

C.E. Catalano, E. Camossi, R. Ferrandes, and V. Cheutet. A product design ontology for enhancing shape processing in design workflows. *Journal of Intelligent Manufacturing*, 20 (5):553–567, 2009.

R. Dekkers, C.M. Chang, and J. Kreutzfeldt. The interface between "product design and engineering" and manufacturing: A review of the literature and empirical evidence. *International Journal of Production Economics*, 63:749–755, 2013.

N. Drummond and R. Shearer. The Open World Assumption. Technical report, University of Manchester, UK, 2006.

[3] https://sites.google.com/site/tc51wgontology/contributions

X. Fiorentini, I. Gambino, V.C. Liand, S. Foufou, S. Rachuri, C. Bock, and M. Mani. Towards an ontology for Open Assembly Model. International Conference on Product Lifecycle Management, Milan, Italy, 2007.

V. Fortineau, T. Paviot, , and S. Lamouri. Expressing business rules from business expertise to formal implementation: An application to the nuclear industry. Doctoral Workshop, Internation Conference on Product Lifecycle Management, PLM13, Nantes, France, 2013a.

V. Fortineau, T. Paviot, and S. Lamouri. Improving the interoperability of industrial information systems with description logic-based models - the state of the art. *Computers in Industry*, 64:363–375, 2013b.

V. Fortineau, T. Paviot, and S. Lamouri. 5 root concepts for a meta-ontology to model product along its whole lifecycle. 11th IFAC workshop on Intelligent Manufacturing Systems (IMS'13), Sao Paulo, Brazil, 2013c.

V. Fortineau, X. Paviot, T. Fiorentini, L. Louis-Sidney, and S. Lamouri. Expressing formal rules within ontology-based models using SWRL: an application to the nuclear industry. *International Journal On Product Lifecycle Management*, sousmis, 2013d.

N. Guarino and C.A. Welty. *Handbook on Ontologies*, chapter An overview of OntoClean, pages 201–220. 2009.

D. Kiritsis, S. El Kadiri, A. Perdikakis, and A. Milicic. Design of fundamental ontology for manufacturing product lifecycle applications. International Conference on Advances in Production Management Systems, Rhodes Island, Greece, 2012.

A. Matsokis and D. Kiritsis. Ontology applications in PLM. *International Journal of Product Lifecycle Management*, 5:84–97, 2011.

T. Özacar, Ö. Öztük, and M.O. Ünalir. ANEMONE: an environement for modular ontology development. *Journal of Data and knowledge engineering*, 70:504–526, 2011.

H. Panetto, M. Dassisti, and A. Tursi. ONTO-PDM: Product-driven ONTOlogy for Product Data Management interoperability within manufacturing process environment. *Knowledge based engineering to support complex product design*, 26(2):334–348, 2012.

H.S. Pinto, S. Staab, and C. Tempich. Diligent: Towards a fine-grained methodology for distributed, loosely-controlled and evolving engineering of ontologies. In *ECAI*, volume 16, page 393, 2004.

S. Staab, H.P. Schnurr, R. Studer, and Y. Sure. Knowledge processes and ontologies. *IEEE Intelligent Systems*, 16 (1):26–34, 2001.

M.C. Suárez-Figueroa. *NeOn Methodology for building ontology networks: specification, scheduling and reuse*. PhD thesis, Informatica, Spain, 2010.

S. Tsuchiya. Improving knowledge creation ability trough organizational learning. *Proceedings of International Symposium on the Management of Industrial and Corporate Knowledge, ISMICK, Compiègne, France*, 1993.

M. Vegetti, H. Leone, and G. Henning. PRONTO: An ontology for comprehensive and consistent representation of product information. *Engineerings Applications of Artificial Intelligence*, 24(8):1305–1327, 2011.