Draft: Metamodels towards Improved Domain Modeling for Semantic Inferencing

Paul Witherell, Anantha Narayan, JaeHyun Lee, Sudarsan Rachuri
National Institute of Standards and Technology
Gaithersburg, MD 20899

***Abstract*- As information requirements have increased, domain models have become increasing complex and difficult to manage. Though domain-specific languages have been developed for domain experts, increasing their expressivity and decreasing their complexity, their effectiveness is often limited by their implementation. This paper will discuss current domain modeling practices, specifically in the form of OWL and SWRL within the context of product development. We then recommend a set of best practices to account for domain context while promoting application-specific domain modeling. We then propose that a metamodel can be used to incorporate these practices in early domain modeling. We discuss what factors should l be considered in such a metamodel, and finally outline further development in future work.**

I. INTRODUCTION

Many knowledge management systems have been built on the premise that adding explicit structure to domain context creates environments that are not only human-readable, but also computable. These systems often provide computability through rules and grammar, or schema and syntax, and significant research has been done in this area in the form of language development. This research is often tailored to meeting the language expressivity requirements of a specific domain. In practice, the effectiveness of knowledge management systems often comes down to not only the expressivity of the modeling language, but also how the domain is modeled. This paper will explore an approach to improving this effectiveness by utilizing domain-specific metamodels to account for language capabilities and application requirements during domain modeling stages.

This paper will focus on the knowledge management in the product development domain. In particular, this paper will discuss the use of domain-specific metamodels to support inferencing in product development knowledge management systems developed with the Semantic Web's OWL (Web Ontology Language) and SWRL (Semantic Web Rule Language).

This paper will begin by discussing the evolution and use of knowledge management systems in product development. Next, the advantages and disadvantages of adding structure to domain-specific context will be discussed, specifically structure through ontologies and the Semantic Web. Several possible inferencing capabilities in product development are presented to highlight the potential of a well-structured knowledge base. Finally, this paper will propose a set of best practices to develop domain models most suitable for inferencing and propose a metamodel approach to as a means to support these recommendations.

II. KNOWLEDGE MANAGEMENT IN PRODUCT DEVELOPMENT

Knowledge management (KM) needs have become industry mainstays as industry has become progressively more reliant on distributed processes and projects have become larger and more detailed. Continued increases in information requirements (1) have led to the continued need for new and advanced methods to capture and share this information. These advancements are necessitated by the importance of proper knowledge management to successful product development. One could argue that the integrity of the product development process as a whole depends on sound knowledge management, as errors accrued throughout these processes can be very costly. It has also been shown that the cost of errors is drastically reduced the earlier the error is detected (2). Facts such as these stress the need for dependable knowledge management in both early and later product development phases.

The advantages offered by many of the advanced knowledge management systems used in product development are highly dependent on the methods used to capture and represent information. The more capable knowledge management systems will often manage not only necessary information, but also instructions on how to use it. As researchers at NIST note, "In order to support reuse of engineering knowledge, a representation must convey additional information that answers not only 'what' questions about a design, but also 'how' and 'why' questions (1)." This additional comprehensiveness can be

addressed by capturing higher-level, or meta, knowledge from engineers, such as the rationale behind an engineer's decisions during the design process. To capture and manage such information, research has sought to develop more structured knowledge management systems, beginning by adding further structure to knowledge bases.

Early works with the development of knowledge bases in engineering focused on product knowledge representation, including work by deKleer and Brown (3), Iwasaki and Chandrasekaran (Iwasaki & Chandrasekaran, 1992), Alberts and Dikker (4), Henson et al. (5), Goel et al. (6)(7), Qian and Gero (8), Ranta et al. (9), and Umeda et al (10). These works laid the groundwork for formally defining and representing product information, such as the categorization of product information into form, function, and behavior in the NIST Design Repository Project (1).

Many researchers began to adopt the form of objects and relationships for formalized product representation, such as that seen NIST Design Repository, (11), and this became a focus of research in advanced knowledge management systems. The inclination towards this approach can be attributed to the formal information structure offered that, when instantiated, represents knowledge that can then be operated upon by computers and humans alike (12). The potential applications of knowledge management systems based on computable representations have driven research in languages and expressivity for representing product development knowledge.

A significant amount of research in knowledge management techniques has focused on meeting the KM needs of a domain and representing information in such a way that is practical for the end user. The next session will discuss research in the area of domain specific languages.

Take Away: KM in Product Development is important; There has been significant research in the area, much of it focusing on KR and Languages.

III. MODELING LANGUAGES

This section will discuss the motivation for and use of Domain-specific modeling languages (DSMLs). It will also discuss their capabilities, and how many of their capabilities are derived from their respective General-purpose modeling languages (GPMLs). It will then discuss the use of ontologies as a compromise between DSMLs and GPMLs, followed by an introduction to the Semantic Web.

A. Domain-Specific Languages

Model driven engineering is the discipline of designing and using domain specific modeling languages to simplify and automate engineering design and analysis. Creating information models of domains requires both domain experts and information modeling experts. Domain experts provide requirements of the information model based on their domain experience, while information modeling experts can construct the information model in a proper way by considering the syntax and semantics of modeling languages, limitations of modeling language expressivity, as well as efficient information structure. Domain-specific modeling languages (DSML) are developed to bring the abstraction level of modeling languages down to the domain level, tailoring the language to reduce language complexities that may be encountered by domain experts. DSMLs consist of domain-specific concepts and relationships so that domain experts can also understand and develop domain information models. DSMLs intentionally hide many complexities of information representation in general-purpose modeling languages (GPML) and essentially become metamodels of a GPML for a specific domain. Figure 1 categorizes several GPMLs and DSMLs based on their knowledge representation abilities.

Light-weight knowledge representation → Heavy-weight knowledge representation

Knowledge representation paradigms	Unstructured	Structured	Object-oriented	Logic-based	
General-purpose modeling language	Natural language	SQL, XML	UML	Description logic, OWL	First-order logic, KIF, CLIPS
Domain-specific modeling language	-	ebXML, RossettaNet, …	CPM2/OAM, SysML, …	OPML, SPMM, …	PSL, …

Fig. 1 Modeling languages

GPMLs have their abstract syntax, concrete syntax, and semantics. The abstract syntax explains significant elements of the language, while the concrete syntax describes how the language expression looks. Light-weight GPMLs define the semantics of domain concepts informally, whereas heavy-weight GPML can define them explicitly and formally. DSMLs also have abstract syntax, concrete syntax, and semantics. The abstract syntax of DSMLs defines domain-specific concepts and relationships. DSMLs sometimes create a new concrete syntax, or they inherit the concrete syntax of a GPML. Semantics of domain concepts and relationships are also a part of a DSML, and they can be axioms or definitions of domain-specific concepts and relationships. In short, though DSMLs are tailored for a specific domain they often take advantage of a GPML used in development.

Significant advancements have been made in DSML's in attempt to accurately represent and model domain-specific information. Once a DSML is well designed and implemented for a domain, domain experts can get the following advantages by using it.

- Domain experts can build their information models by themselves without the burden of learning generic-purpose modeling languages.
- Domain experts can understand information models so that communication between domain experts and knowledge modeling experts becomes easy.
- DSMLs allow validation of information consistency at the domain level.

These efforts focus on simplifying the job of the domain expert by reducing levels of abstraction by increasing the expressivity of the modeling language. While this creates a more powerful language, it also reduces its flexibility. Product development is rarely regulated to a single domain, and often has to cross domain boundaries. Therefore, any DSML adopted for product development needs to be modified or extended efficiently, or should be able to support multiple domains. In (13), desired product modeling capabilities were categorized into four categories. These capabilities enable designers to express portions of a product model, and combine them with contributions of other designers. The capabilities fall into:

- Generalization (taxonomies, refinement).
- Interconnections as components (reuse, inheritance, decomposition, interconnections between interconnections).
- Behavior as relations and interconnections.
- Models of device and environment (designs and requirements, total systems).

Ontologies are able to satisfy each of these language requirements, while at the same time offering the flexibility to be tailored to multiple domains. To this end, many researchers have turned to Semantic Web for developing knowledge management systems for product development.

B. Ontologies and Semantic Web

Ontologies have the ability to mimic many of the advantages offered by domain specific languages by providing a formalized means to add new context to existing structure, in other words they can easily be made DSMLs from GPMLs. Ontologies first were made popular as a knowledge modeling technique used in AI (14), and their ability to create and operate on domain specific vocabulary and knowledge has long been of interest to the scientific community(15).

Ontologies may vary in many different respects, such as domain, language, comprehensiveness, and expressiveness. The expressiveness, determined by the logic used in their development language, plays a major role in determining its capabilities. One of the most common languages used in ontology development is OWL (Web Ontology Language), which is based on Description Logic (DL).

A cornerstone of the Semantic Web, OWL provides a computable, explicit domain information structure, subsequently catering to both computational management systems and distributed knowledge bases. Driven by DL, OWL allows for relationships such a holynmic (subsumption) and meronomic (part-of) to be formed within a knowledge base. OWL axioms define a class by assigning necessary and/or sufficient characteristics to a class. Axioms also allow for restrictions to be placed on classes, such as a cardinality restriction. Such axioms open the door for DL reasoning mechanisms (e.g. consistency checking, subsumption, equivalence, etc.) to act on concepts and relationships between concepts and explicitly represent otherwise implicit knowledge.

The expressivity of OWL can be extended through SWRL, or the Semantic Web Rule Language. SWRL extends OWL both syntactically and semantically. Developed from Rule ML (16), the SWRL extension of OWL creates a much more expressive language than either OWL or Horn clauses individually Beyond the abilities of Horn clauses, the expressive power of SWRL also allows "existentials" to be expressed in the head of a rule, (17). SWRL's Horn clause(18)"if-then" capabilities allow inferences to be drawn on the assertion component, or ABox, of a knowledge base, based on relationships defined in the TBox, or terminological component. Such inferences are essential in providing additional functionalities to

an OWL knowledge base, as SWRL allows conclusions to be drawn based on existing knowledge.

Take Away: There has been a longstanding effort to meet the needs of knowledge representation in product development. These DSML can be considered metamodels of the domain, and their efforts often focus on simplifying the job of the domain expert by adding domain-specific context. Ontologies are able to meet the language requirements of Product Development DSMLs.

IV. KNOWLEDGE MANAGEMENT WITH ONTOLOGIES

The previous section highlighted the fact that significant research has been done on meeting domain needs for knowledge management through tailored language and ontologies. This section will discuss the motivation for this research, namely the applications and advantages offered by ontologies as foundations for knowledge management tools.

A. Ontologies and Knowledge Managment

When knowledge management systems are built using logical languages, the end result is often a tool with reasoning capabilities. Reasoning can lead to inferences that can reduce redundancy, check for consistency, and classify instances in a knowledge base.

The success of domain modeling and reasoning with ontologies is influenced by the application domain and the intentions of the ontology. Notable successes include the Human Genome ontologies (MeSH, SNOMED-CT, and ICD9-CM [1]) and GIS (Geographical Information Systems). In product development, ontologies have emerged as building blocks for many knowledge management techniques and DSMLs. Notable applications include product knowledge interoperability and life-cycle management. The following subsection will discuss some specific capabilities that can be realized through the use of ontologies for KM in product development.

B. Ontologies in Product Development

Different types of logical inferencing can be performed on an ontology depending on the language used to develop it. By complimenting the expressivity of Description Logic with Horn clauses, a broader spectrum of the more expressive first-order logic can be replicated with an OWL and SWRL knowledge base. When modeling in an ontology, DL relationships inherently become part of the knowledge associated with an artifact as well, for instance if the concept of "car" is subsumed by the concept "vehicle", than without explicitly stating it, it can be inferred that "car" is a type of "vehicle," or make the statement a "car" has four wheels. Xenia et al (An Analysis of Description Logic Augmented with Domain Rules for the Development of Product Models) evaluated the expressivity of DL based on the general information modeling requirements for product development. While the DL axioms provide a mechanism for capturing traditional product and process knowledge, the Horn clauses provide a mechanism for capturing and expressing less traditional information in the form of "if-then" statements.

In today's product development processes domains often develop independently, especially with the utilization of distributed and concurrent design approaches. Many approaches to support these approaches involve the creation of workflow environments or central repositories where information is stored and shared. In industry, companies have begun to employ systems engineers as a means for harmonizing the development of a product, both sequentially and concurrently. The increasing demand for this position highlights the desire and need for improved and streamlined communication between domains within product development. Inferencing on a structured knowledge base is another way of replicating much of this work by 1) reducing redundancy and 2) facilitating consisting across independently developed domains.

Additionally, relationships between each stage of product development can be fully exposed and made computable so software tools can help engineers understand interactions between knowledge and anticipate the impact of changes to a product. The ability to have a real time awareness of the progression of designs of individual components during the design process can greatly simplify the process and reduce margins of error. As Boston et al note, readily accessible knowledge during these (early) critical phases of the product development process can drastically reduce costly errors and ultimately lead to a more efficiently developed product (19). The introduction of logical inferenceing to a structured knowledge base can support important facets associated with product knowledge management, such as:

- Minimization of redundancy in the knowledge instantiation process
- Maintaining of consistency during the knowledge instantiation process
- Corroboration of knowledge instantiations

Logic can be used to assist the engineer in his or her decision making, eliminating infeasible decision alternatives and assisting in the knowledge capturing

[1] International Classification of Diseases, 9th revision, Clinical Modification

process. Ultimately, however, the inferencing capabilities of a domain model depend on the expressivity of the language used to model it and the structure of the modeled domain. The realization of many of these capabilities depends not only on the language used, but also how the domain model is developed.

Take away: Ontologies and the Semantic web can be very powerful tools when applied correctly.

V. DEVELOPING A DOMAIN MODEL

Languages such as OWL provide significant flexibility and allow information to be represented in many ways, such as objects or as data. The effectiveness of a tool can be greatly influenced by the domain model on which the information is stored. A domain model should be tailored to take properly take advantage of the inferencing capabilities offered by many of modeling languages. Factors that should be accounted for include not only the domain in question, but also to the use-intention of the model. This section will first discuss current domain modeling practices and then propose a set of recommended practices that will consider not only domain requirements, but also application requirements when modeling a domain.

A. Common Modeling Practices

Common domain modeling practices usually follow the lead of the development of domain specific languages, developing models to suit the needs of the domain. When modeling a domain, the objective is to satisfy the needs of the domain as best possible, including modeling domain concepts and relationships. What is often not considered during development, however, is how the domain model is going to be used. As discussed earlier, the reason for the adaptation of knowledge management systems with formal, explicit structure to context is often to utilize the reasoning capabilities that are offered. The result is often a domain model with considerable redundancy and unnecessary complexity. These complex models then must be carefully studied and analyzed to understand their capabilities.

The inferencing capabilities of a knowledge base are determined by the structure of the knowledge base and the language used in its development. For instance, consider again the statement "A car has four wheels." If this sentence where of an ontology of English grammar, each work would have the same importance. However, if this sentence is meant to describe a type of vehicle, the words "car," "four," and "wheel" become important concepts that may be used to distinguish a car from perhaps a boat, or a plane. If this sentence where meant to describe types of "wheeled vehicles," the concept of wheel may no longer be as important, and instead the concepts of "car" and "four" become the mechanisms for reasoning. It is important to remember as new concepts are added to act as reasoning mechanisms the complexity of the domain also increases. By understanding the reasoning capabilities and identifying necessary reasoning mechanisms during the early development of a domain model, less complex and more effect domain models can be developed.

One of the main obstacles in deploying knowledge management frameworks is understanding where inferencing should be done to exploit the inherent logic. One of the common criticisms of ontologies is the size they can reach and the perception they can become unmanageable to the point where the advantages provided by the structure and explicitness they offer are overshadowed by their complexities. By accounting for the application of a domain model in early modeling stages, the model can be simplified and the language exploited to meet the application needs. A metamodel to support domain-specific inferencing would allow the domain experts to better tailor the development of their knowledge base and consider inferencing aspects early in the development stage. Towards this, we will now present best-practices that such a metamodel should facilitate.

B. Recommended Practices

To create a good, quality domain model, it is important to identify which are the most significant concepts in the domain, how they are related to each other, and how the domain models will be used in practice. It is important to identify what kind of information we would like to be able to infer from domain models, and capture the concepts and relationships that will allow this kind of inferencing to be performed. Here, we present a short list of guidelines (based on ref: "Design Guidelines for Domain Specific Languages", Gabor Karsai et. al.) that will allow us to design better models for inferencing.

Identify language use early: We must be able to foresee the scenarios in which the DSML may be used. It may not always be possible to completely identify all usage scenarios; however, sufficient attention must be paid to the most commonly expected ones. For instance, identify the kind of information we would like to infer from the domain models, and capture the concepts and relationships necessary to make these inferences.

Ask questions: Once the usages have been identified, ask a number of questions about each use case. When is the inferencing performed? Who is performing the inferencing? What other languages and tools are involved in the inferencing?

Compose existing languages when possible: It may often be the case that one or more existing languages capture all the concepts and relations of your domain. In these cases, it is beneficial to compose these languages to create a new composite language. The advantage of using OWL is that composition is fairly easy and often automatable. The composed language will have stronger inferencing abilities than the individual languages it is composed from.

Capture only the necessary domain concepts and keep it simple: It is very easy to get into the trap of identifying and listing every domain concept you can think of. Capturing too many domain concepts could make the language rigid and limited in its application. The language must present concepts that are generic enough to be flexible, while capturing the key ideas of the domain. This reflects on the previous guideline about identifying use cases early, and capture only those concepts that contribute to these uses.

Be mindful of efficiency: To be practically useful, it must be possible to perform automated inferencing on large models in a reasonable amount of time. In other words, the inferencing must be scalable. Keeping the language simple is one requirement for this. On the other hand, identifying the types of inferencing early allows us to capture the concepts and relationships tailored to addressing its complexity and scalability.

In the next section we will discuss how metamodels be used to assist in addressing each of these points to create more usable, efficient domain models.

Take away: When developing a domain model, we should consider not only the needs of the domain, but also the intended application. This is often not done in current practices.

VI. METAMODELS TO SUPPORT MODELING FOR APPLICATION AND DOMAIN-SPECIFIC INFERENCING

Metamodels for domains are developed to consider domain constructs through domain requirements, concepts and attributes. When creating a computationally-driven domain model, not only should the domain and language constructs be considered, but also the intended application. We propose that a metamodel for domain-specific inferencing can provide crucial guidance during the development of a domain model to understand how the domain can be reasoned on. By developing a domain model with a clear purpose or application intent, more effective, efficient models can be created. The notion of a metamodel to support inferencing is meant to increase the effectiveness of a knowledge base while decreasing its complexity.

It has been shown that metamodels are an effective way for communication. They are especially useful for providing a visuzalztion mechanism for text-based syntax, such as that seen by OWL and SWRL modeling.**Invalid source specified.**. They can also be used to provide best-practices and outlines. Figures 1-3 show metamodels**Invalid source specified.** of GPMLs (OWL and SWRL) and a DSML (CPM). As discussed throughout the paper, each of these comes with their own advantages/disadvantages. When developing a domain model, elements from each should be carefully considered.

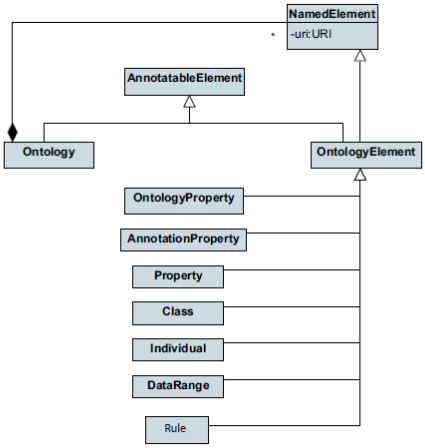

Figure 2. OWL Metamodel

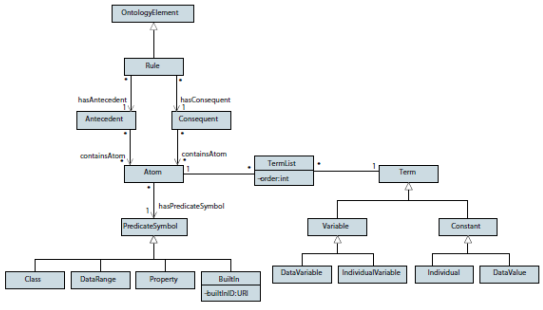

Figure 3. SWRL Metamodel

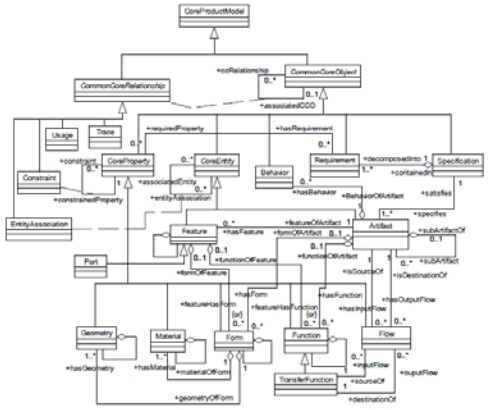

Figure 4. CPM Metamodel

Many factors need to be considered during the development of a domain model, and even more when also considering how a domain model will be reasoned upon. The reasoning mechanisms depend on the domain being modeled and the objective of the domain model. The metamodel should consider what inferencing capabilities could be used in the domain model, i.e. the DL axioms and Horn clauses offered by OWL and SWRL. A metamodel should consider the extensiveness that a domain needs to be modeled. While there are many details that a domain modeler may want to capture, in practice it is best to understand which ones are necessary and productive. A metamodel should consider which attributes should be reasoning mechanisms, and which may have little effect on the models inferencing capabilities knowing the models intentions. These are a few issues that can significantly influence the complexity and effectiveness of a domain model used for inferencing. Each will be discussed further in the following paragraphs.

When discussing what an inferencing metamodel might look like, it is important to understand the variances that may exist in both the elements and relationships used in the inference. One such consideration involves the elements and relationships offered by the logic employed. Metamodels built specifically to the language must consider language constructs, language representation, and language expressiveness. Such a metamodel (See Figures 1 &2) offers no insight into how a domain should be modeled in order to exploit inferencing functionality. The metamodel will only mimic the constructs of the syntax of the inferencing language used. While this is useful to understand language expressivity, domain considerations must also be made.

By having a better understanding of how the structure of a domain can be exploited during the development phase, models can become more efficient and the knowledge that can be gained through inferecing can be increased. For instance when modeling with OWL, one should not only consider the application context, but also the Horn implications when modeling in DL. This can be done by understanding the type of inferencing that you are most likely to need based on domain context, and how this inferencing is done. Domain context and application needs are important considerations, but it also important to consider them while also considering available expressivity. These considerations will influence the inherent structure of the domain model.

The complexity of the structure is significantly influenced by not only understanding attributes of significance, but also the extent of their significance. For instance, the example was given on reasoning to determine a vehicle was a car based on the concept of having four wheels. However, what was not discussed was to what extent the concept of wheel should be expressed. For instance consider again the "A car has four wheels" sentence. Many domain modeling languages provide mechanisms for distinguishing between object attributes and string attributes. While both may allow an attribute to be expressed explicitly, object attributes allow for further detail. However, with this additional detail comes additional complexity. Tradeoffs such as these are issues that should be, yet not always are, considered during the development of domain models. Metamodels will help domain modelers better understand how to address application needs and what the tradeoffs are of the model.

Take away: Metamodels have been created for domains and languages, we can create a metamodel that can also take into consideration application intentions. Such a metamodel could be used to create more efficient km tools.

VII. DISCUSSION

This paper discussed improving domain modeling practices for domain applications through improved metamodels. While domain specific modeling languages have become established practices, their development is often catered to meet the needs of a domain. As information complexities increase and knowledge management application progress, further emphasis needs to be placed on understanding the context for which a domain is being modeled. To this end, guidelines were proposed and a metamodel was theorized that could be used to improve domain modeling practices. While this paper presented the idea, continued research is needed before a functional metamodel can be proposed. In future research we will begin the modeling phase of a metamodel for domain specific inferencing using OWL and SWRL within in the product development domain. From this

work we hope to abstract a template to develop further metamodels for additional domains and languages.

REFERENCRES

1. *The NIST Design Repository Project.* **Szykman, S., et al.** s.l. : Springer-Verlag, 1998, Advances in Soft Computer-Engineering Design and Manufacturing.

2. *Measuring and Managing Quality in the Engineering Design Process.* **Finn, G.A.** s.l. : High Mountain Press, April 1999, CATIA Solutions Magazine.

3. *Assumptions and ambiguities in mechanisticmental models.* **de Kleer, J. and Brown, J.S.** [ed.] D., & Stevens, E.L. Genter. Hillsdale, NJ : Erlbaum, 1983, Mental models, pp. 155–190.

4. *Integrating standards and synthesis knowledge using the YMIR ontology.* **Alberts, L.K. and Dikker, F.** [ed.] J.S., Sudweeks, F. Gero. Boston : Kluwer Academic, 1992, Artificial Intelligence in Design, pp. 517–534.

5. *Towards an integrated representation of function, behavior and form, computer aided conceptual design.* **Henson, B., Juster, N. and de Pennington, A.** [ed.] J., Oh, V. Sharpe. Lancaster : s.n., 1994. 1994 Lancaster Int.Workshop on Engineering Design. pp. 95–111.

6. *Explanatory interface in interactive design environments.* **Goel, A., et al.** [ed.] J.S. Gero. Boston : Kluwer Academic, 1996, Artificial Intelligence in Design.

7. *KRITIK: an early case-based design system.* **Goel, A., Bhatta, S. and Stroulia, E.** [ed.] M., Pu, P. Maher. Mahwah, NJ : Erlbaum, 1996, Issues and Applications of Case-Based Reasoning to Design.

8. *Function–behavior–structure paths and their role in analogy based design.* **Qian, L. and Gero, J.S.** 4, 1996, Artificial Intelligence for Engineering Design, Analysis and Manufacturing, Vol. 10, pp. 289–312.

9. *Integration of functional and feature based product modeling—the IMS0GNOSIS Experience.* **Ranta, M., et al.** 5, 1996, Computer-Aided Design, Vol. 28, pp. 371–381.

10. *Supporting conceptual design based on the function–behavior–state modeler.* **Umeda, Y., et al.** 1996, Artificial Intelligence for Engineering Design, Analysis and Manufacturing, Vol. 10, pp. 275–288.

11. *Design repositories: engineering design's new knowledge base.* **Szykman, S., et al.** s.l. : IEEE, 2000, Intelligent Systems and Their Applications, Vol. 15, pp. 48-55.

12. *Ontologies for supporting engineering analysis models.* **Grosse, I., Milton-Benoit, J. and Wileden, J.** 1, 2005, Artificial Intelligence for Engineering Deisign, Analysis, and Manufacturing, Vol. 19, pp. 1-18.

13. *Ontological Product Modeling for Collaborative Design.* **Bock, C. E., et al.** 4, 2010, Advanced Engineering Informatics, Vol. 24, pp. 510-524.

14. *An Ontology for Engineering Mathematics.* **Gruber, T. and Olsen, G.** [ed.] J., Torasso, and Sandewall, E Doyle. San Mateo, CA : Morgan Kaufman. Proc. Fourth International Conference on Principles of Knowledge Representation and Reasoning. pp. 258-269.

15. *Component-Based Support for Building Knowledge-Acquisition Systems.* **Musen, M. A., et al.** Beijing : s.n., 2000. Conference on Intelligent Information Processing (IIP 2000) of the International Federation for Information Processing World Computer Congress.

16. *MOF-RuleML: The Integrate.* **Wagner, G., Tabet, S. and Boley, H.** Boston : s.n., October 2003. 2003 OMG Meting.

17. *Using Vampire to Reason with OWL.* **Tsarkov, D., et al.** 2004. 3rd International Semantic Web Conference.

18. *On sentences which are true of direct unions of algebras.* **Horn, A.** 1956, Journal of Symbolic Logic, Vol. 16, pp. 14-21.

19. *Life-cycle management of supplier literature: the pertinent issues.* **Boston, O. P., Culley, S. J. and McMahon, C. A.** 3, 1999, The Journal of Product Innovation Management, Vol. 16, pp. 268-281.

20. *Towards a Standardized Engineering Framework for Distributed, Collaborative Product Realization.* **Choi, H., et al.** Chicago, IL : s.n., September 2. 2003 ASME Computers and Information in Engineering Conference.