Ontology-based Knowledge Framework for Product Development

Jaehyun Lee[1], Hyowon Suh [2] and Soon Hung Han[3]

[1]Department of Industrial Engineering, KAIST in Korea, knight@kaist.ac.kr
[2]Department of Industrial Engineering, KAIST in Korea, hwsuh@kaist.ac.kr
[3]Department of Mechanical Engineering, KAIST in Korea, shhan@kaist.ac.kr

ABSTRACT

Although information systems let manufacturing companies have a lot of development data, engineers do not know how to manage those huge data. So knowledge management is required to guide the engineers with comprehensive knowledge. In this paper, we propose the ontology-based knowledge framework for the product development. The framework composes of the three levels of knowledge. Lowest level is axioms, which explicitly and formally specify the semantics of concepts and relations. Middle level is a knowledge map, which defines the common domain knowledge that domain experts agree with. The knowledge map can guide engineers which design data are required for their tasks. Highest level is a specialized knowledge for domain which gives the solution of a specific task or problem. The specialized knowledge is classified into three knowledge types; expert knowledge, engineering function and data-analysis-based knowledge. The framework has a uniform representation; first order logic to integrate the three levels smoothly. We implement a prototype of the framework using prolog and test the example queries to show the effectiveness of the framework.

Keywords: Ontology, Knowledge framework, Engineering Knowledge.

1. INTRODUCTION

During the early 1990s, many researchers studied the concurrent engineering to reduce the time-to-market[19]. To shorten the duration of product development, a systematic management of product knowledge is required. Designers spend more than 70% of their working time to search and handle the recently updated knowledge, so that such an unnecessary waste time decreases the productivity of designers[8]. Stauffer et al. [16] studied why engineers take too much time to utilize knowledge of past projects. The problem is that past knowledge is not well organized. One reason of the problem is that designers do not have enough time to arrange information and knowledge they have. Another reason is not only companies do not regard the knowledge as their assets, but also they do not have enough budgets for knowledge management (KM). According to the result of Court's empirical study [4], designers use about 30% of their private knowledge during product development. They use even 50~70% of the private knowledge in some cases. Therefore, increasing the utilization of engineers' private knowledge and knowledge sharing through KM is a critical leverage in product development.

Nowadays there exists an overall consensus that the process of building a knowledge-base system is seen as a modeling activity[17]. Since the product development knowledge is complex and types of the knowledge are various, a knowledge framework is required to assist knowledge engineers with modeling the comprehensive knowledge of product development detail, clearly, and consistently. Thus, this paper introduces multi-level knowledge framework for comprehensive knowledge management. Previously, several knowledge management approaches have been proposed. However, the approaches are limited in accommodating comprehensive knowledge of several types without ambiguity and heterogeneity. The proposed approach is based on ontology which is emerging technology, but not fully developed for knowledge management. The details will be discussed in the following sections. In section 2, we discuss the previous researches and describe the proposed approach, ontology-based knowledge framework (OBKF) in section 3. The details of each level of knowledge framework are discussed in section 4. In section 5, we discuss the system architecture for OBKF. The example of the framework for a product development domain is also presented in section 6. Lastly in section 7, summary and considerations for future research are provided.

2. PREVIOUS RESEARCH

Many previous researches adopt different knowledge frameworks for their systems. We can classify those frameworks into three types like Fig. 1.

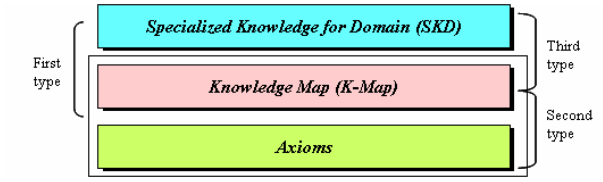

Fig. 1. Previous research classifications

First type framework composed of the problem-solving-knowledge and ontology. For example, PACT[5] and SHADE[9] developed agent-based collaborative product design systems based on ontology. Each agent has different problem-solving-knowledge, and the ontology is utilized for sharing knowledge-base. Another example is the OntoEdit[18], which supports ontology engineers collaboratively developing ontology. It provides the engineers with functions; editing ontology and rules. The rules can be viewed as the problem-solving-knowledge. Second type framework is two layers approach for ontology. Bozsak et al.[1] said ontology is composed of 5-tuples; concepts, relations, concept hierarchies, functions, and axioms. The structure of knowledge can be represented with the 4-tuples of the ontology definition; concepts, relations, concept hierarchies, and functions. The semantics of knowledge is formally and explicitly specified by axioms. Thus, we can divide ontology tuples into knowledge map and axioms. The knowledge map is the structure of knowledge, and the axioms specify the semantics of concepts and relations in knowledge map. A typical example of the framework is Staab and Maedche[15]'s study. They separate the structure of ontology into axioms and others. They classified axioms based on their experience of developing ontology, and suggested some templates of axioms to develop axioms easily. Third type framework composed of the problem-solving-knowledge and the knowledge map. For example, Yoshioka[21] proposed 'meta-model' approach to integrate different design systems. Each system has different problem-solving-knowledge, and the 'meta-model' is correspondent with the knowledge map. The semantics of concepts are described within a concept dictionary. This approach tried to avoid a burden of developing axioms by using the dictionary. However, the dictionary approach still had problems for maintaining the semantics of the meta-model because of the informal specification of the semantics.

Each type researches studied different knowledge frameworks, and their merits and defects are also different. The first one showed the relationship between problem-solving-knowledge and ontology, but it did not specify the detailed structure of ontology. The second one showed detailed structure of ontology and role of axioms, it did not show the relationship between ontology and problem-solving-knowledge. The third one showed detailed roles of the knowledge map, but it ignored the role of formally specified axioms. Therefore, we suggest that our knowledge framework integrates the three levels of knowledge. Highest level is a 'specialized knowledge for domain'(SKD), which stands for the problem-solving-knowledge. Middle level is 'knowledge map'(K-Map), which shows the structure of knowledge. Base level is 'axioms', which describe the semantics of the knowledge map. We will explain the framework more detail in section 3.

3. ONTOLOGY-BASED KNOWLEDGE FRAMEWORK (OBKF)

We propose an Ontology-Based Knowledge Framework. In this approach, the knowledge is categorized and structured for comprehensive, unambiguous and homogeneous knowledge management. For this, the knowledge is represented based on ontology. In addition, typical types of knowledge such as engineering functions, expert rules, and data-analysis-based knowledge are specified and accommodated with the frame. And for a uniform representation, first-order logic (FOL) is adopted.

Knowledge Structure: The previous researches [3, 6, 20] suggest various types of product development knowledge. However the criteria of classification are not clear and the concrete relationships between the categories are not discussed. We define two criteria for the classification, the role of knowledge and the source of knowledge. The role of knowledge classifies the knowledge into three levels according to the contribution of the knowledge; task-specific knowledge, common domain knowledge and semantics of knowledge. The source of the knowledge classifies the task-specific knowledge into three types according to the source; engineering function knowledge, expert knowledge and data-analysis-based knowledge. The details of the knowledge structure are discussed in section 4.

Ontology-based: Bozsak et al.[1] define an ontology structure with 6-tuples; concepts, relations, concept hierarchies, relation hierarchies, functions, and axioms. These ontology components can be used for a base of knowledge framework. The concepts and relations generally represent the basic structure of domain knowledge. Thus, common domain knowledge can be represented with concepts and relations including concept hierarchies, relation hierarchies and functions. The other hand, axioms specify the semantics of concepts and relations so that the semantics of knowledge can be represented with axioms. In addition, the task-specific knowledge can also be defined with ontology because it specifies the quantified relationship between concepts of ontology, which is the relation between their instances. Thus, the three levels of knowledge are concretely organized with an ontology structure.

Uniform Representation: Ontology is usually expressed in a logic-based representation, so that the detailed, accurate, consistent, sound and meaningful distinctions among the concepts and relations can be made [7]. Addition to that, logical representation is appropriate for a uniform representation.

The uniform representation should be as expressive as it can express the complex knowledge of product development domain. In addition, its reasoning capability should be as powerful as it can. Corcho and Perez [2] compared the expressiveness and the reasoning capability of several logical languages. They said that there is a trade-off between the degree of expressiveness and the reasoning efficiency of a language. The more expressive representation requires the rigor reasoning capabilities. Based on this, we adopt the FOL as a representation method for the three levels of our framework. The FOL representation has sufficient expressiveness so that it can represent axioms as well as concepts and relations. It also provides reasoning algorithms which can be a basis for task-specific knowledge [14].

4. THE THREE LEVELS OF KNOWLEDGE

As discussed above, we classify the product development knowledge into three levels; task-specific knowledge, common domain knowledge and semantics of knowledge. In what follows, we are naming them Specialized Knowledge for Domain (SKD), Knowledge Map (K-Map) and Axiom respectively for embossing the comprehensive ontology-based knowledge frame. In addition, SKD is classified into three types. We can represent the architecture of the Ontology-Based Knowledge Framework (OBKF) in a simple diagram as Fig. 2.

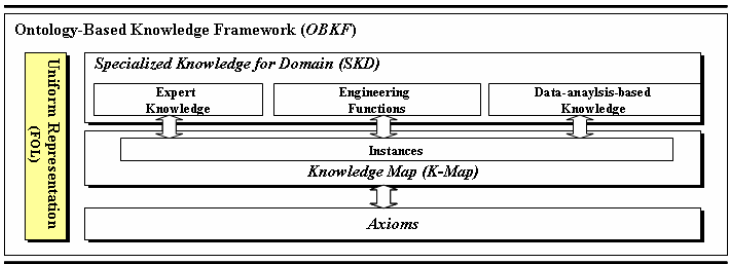

Fig. 2. Architecture of the Ontology-Based Knowledge Framework (OBKF)

4.1 Level 1: Axioms

The axioms specify the semantics of concepts and relations in a domain by logical representation, so that both people and computers clearly understand the meaning of them. Axioms can be generally defined as the basic factual information from which useful conclusions can be derived[13]. However we define the axioms as fundamental properties of concepts and relations as well as definitions of them. For example, let the '*subPartOf*' terminology present a structure relation between two part concepts. The fundamental properties of '*subPartOf*' relation are followings:

- '*subPartOf*' relation is ir-reflexible; a part is not sub-part of itself.
- '*subPartOf*' relation is anti-symmetry; if a part X is a sub-part of a part Y, then the part Y is not a sub-part of the part X.
- '*subPartOf*' relation is transitive; if a part X is a sub-part of a part Y and the part Y is a sub-part of a part Z, then the part X is a sub-part of the part Z.

The definition of concepts and relations also can be described in logic. To take a simple example, let a part be a direct-Sub-Part-Of another part when there exist no middle part between the two parts. We can define the '*directSubPartOf*' relation in logic like following:

- If a part X is a direct sub-part of a part Y, then the part X is a sub-part of the part Y and a part Z does not exist such as the part Z is a sub-part of the part Y and the part X is a sub-part of the part Z .

We can translate the axiom logics into FOL formulae like Fig. 3.

Axioms	FOL formulae
1) *subPartOf* is non-reflexible	$(\forall p) \neg subPartOf(p, p)$
2) *subPartOf* is anti-symmetric	$(\forall p1, p2)\ subPartOf(p1, p2) \Rightarrow \neg subPartOf(p2, p1)$
3) *subPartOf* is transitive	$(\forall p1, p2, p3)\ subPartOf(p1, p2) \wedge subPartOf(p2, p3) \Rightarrow subPartOf(p1, p3)$
4) Definition of '*directSubPartOf*' relation	$(\forall p1, p2)\ directSubPartOf(p1, p2) := subPartOf(p1, p2) \wedge \neg(\exists p3) \wedge subPartOf(p3, p2) \wedge subPartOf(p1, p3)$

Fig. 3. An example of Axioms

The fundamental properties and definitions in axioms keep the consistency of a knowledge base. When new knowledge is introduced or previous knowledge is evolved in a knowledge base, it can be verified with axioms. For example, if an engineer defines that a part A is a direct sub-part of a part B, the definition of '*directSubPartOf*' prevents others from inserting another part between the two parts.

We can classify axioms into two groups. One is a group of 'classified-axioms', and the other is a group of 'customized-axioms'. The 'classified-axioms' is a set of axioms which can be generally grouped, because they have very similar meanings. For example, the 'non-reflexible' meaning of 1) axiom in Fig. 3 can be frequently shown in other binary relations. Such fundamental properties of a binary relation are frequently shown in most ontology. Therefore we can group them as a set, and model them easily. The 'customized-axioms' is a set of axioms which is not included in the 'classified-axioms'. The customized axioms are too domain specific, so that they are hard to be grouped. The 4) axioms in Fig. 3 is an example of the customized axiom.

4.2 Level 2: Knowledge Map (K-Map)

The K-Map describes the common knowledge of a domain with a structure like a semantic network. The structure of the K-Map is composed of concepts, relations, relation function *Rel*, concept hierarchy and relation hierarchy and functions. The concepts and relations are the realization or perception of the real world objects by humans. They are denoted by terminologies. Therefore, one can state that the concepts and relations define the terminologies used in the K-Map. The relation function *Rel* defines a relationship between the relation and concepts. For example, if '*subPartOf*' relation is related with two '*Part*' concepts, we can describe this logic like '*Rel*(*subPartOf*) = (*Part*, *Part*)'. The concept hierarchy is called taxonomy. We can describe the concept hierarchy with a concept hierarchy function 'H^C'. If a concept '*Material*' is a sub-concept of a concept '*PartChar*', then it can be represented in logic like 'H^C (*Assembly*, *Part*)'. The relation hierarchy also can be described in logic by a relation hierarchy function 'H^R'. For example, the '*directSubPartOf*' relation is a sub-relation of the '*subPartOf*' relation, so that it can be represented in logic; 'H^R (*directSubPartOf*, *subPartOf*)'.

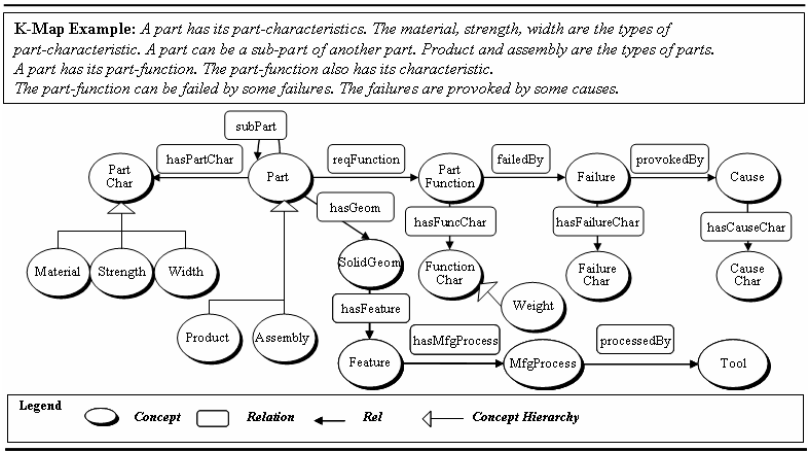

Fig. 4. Example of graphical K-Map

A K-Map of a product development domain can be visualized as shown in Fig. 4. The relational hierarchy is omitted. The K-Map can also be represented in FOL formulae. The K-Map should be agreed by domain experts so that the K-Map can be the base of SKD. The agreed K-Map can be used as a reference model of the domain.

4.3 Level 3: Specialized Knowledge for Domain (SKD)

The SKD supports users' decision, because it roles solving user's problems. The SKD is represented based on the K-Map of domain, and the SKD can provide relational information among the instances of the K-Map. Since SKD concerns about specific tasks or problems, the logical formulae of SKD are described with the instances of concepts and relations as well as the K-Map.

We classify the SKD into three types based on the sources of knowledge; engineering function, expert knowledge, and data-analysis-based knowledge. Firstly, engineering function comes from the scientific theories, and it is represented in mathematical expression. Following equation with a width of 'Leg' part and other factors is a good example of this. In the example, *Ws*, *X*, *St*, and *SF* are 'concepts' of the K-Map. The engineering function is the numerical engineering relation among the concepts of K-Map or the instances of the concepts .

- (SDK-1): $X \geq (Ws / St \times SF)^{1/2}$

X: width of a Leg part *Ws: student's weight*
St: material's tensile strength *SF: Safety Factor*

Secondly, expert knowledge represents the knowledge extracted from experts' experience or intuition, and it is generally qualitative knowledge. The expert knowledge has a 'IF.. THEN..' form like a production rule. Following logical formula is an example of the expert knowledge.

- (SDK-2): IF *feature.name* = *Through_circular_hole* AND *part.material*=*Wood* THEN *tool* = *HandDrill*.

In the example, '*feature.name*', '*part.material*' and '*tool*' are the 'concept' in K-Map, and '*Through_circular_hole*', '*Wood*', and '*HandDrill*' are the instances of those concepts. Thus, this knowledge represents the specialized knowledge about the 'Hole-making'; manufacturing process.

Lastly, the data-analysis-based knowledge is extracted from the analysis of collected data, so that it has relations with specific data-analysis method and the collected data. A data collector, for example, analyzes a relationship between the '*Slackness*' instance of a '*FailureChar*' concept and the instances of a '*CauseChar*' concept; '*Hole & Peg Tolerance*', '*Relative Surface Friction*', and '*Peg strength*'. He can collect a '*DataCubic*' data set from a data table in Fig. 5, so that he can know the relations among the data at a glance. He can also apply a 'Linear Regression' method to analyze the data table. After all, he can get a data-analysis-based knowledge like the equation in Fig. 5.

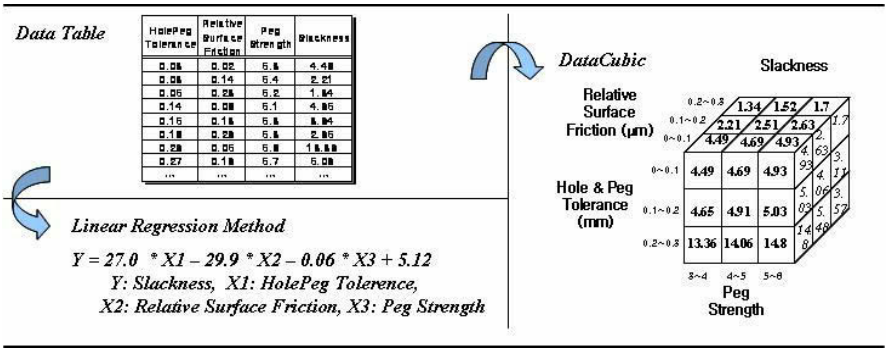

Fig. 5. Example of data-analysis-based knowledge

5. A SYSTEM OF ONTOLOGY-BASED KNOWLEDGE FRAMEWORK (OBKF)

The OBKF can be applied to a knowledge management system so that the system can handle comprehensive and integrated knowledge in a uniform manner. We suggest the architecture of an OBKF-based knowledge management system in this section. The architecture defines which modules are required for the knowledge management system. A realized prototype system will be discussed with an example in the section 6.

The examples of SKD also can be represented in FOL formulae. Fig. 6 shows FOL examples of each SKD types.

Type of SKD	FOL formulae of examples
engineering function □(SKD-1):	*(∀w1 w2 m1) Width(w1) ∧ Weight(w2) ∧ Material(m1) ∧ w1 ≥ squar((* (/ w2 SF) w3)).*
expert knowledge □(SKD-2):	*(∀x y z1 z2 p t) hasFeature(x y) ∧ hasFeatureChar(y z1) ∧ FeatureName(z1) ∧ (= z1 Closed_circular_hole) ∧ hasPartChar(x z2) ∧ Material(z2) ∧ (= z2 Wood) ∧ hasMfgProc(y p) ∧ (= p HoleMaking) ∧ ProcessedBy(p t) → (= t HandDrill)*
Data-analysis-based knowledge □(SKD-3):	*(∀ y x1 x2 x3) Slackness(y) ∧ HolePegTolerence(x1) ∧ RelativeSurfaceFriction(x2) ∧ PegStrength(x3) ∧ (= y (+ 5.12 (+ (* (-0.06) x1) (+ (* (-29.9) x2) (* 27.0 x3))))).*

Fig. 6. An example of SKD.

5.1 OBKF System Architecture

We design an OBKF system architecture considering four requirements. Firstly, an OBKF system helps expert users organize their knowledge easily and it also should help novice users utilize the accumulated knowledge easily. Secondly, an OBKF system should be integrated with legacy systems such as PDM (Product Data Management), ERP (Enterprise Resource Planning), SCM (Supply Chain Management) and others. They have a huge database related to product development. So the OBKF system should have an interface mechanism with their database and other functionality. Thirdly, an OBKF system should guide novice users to do their task correctly. The K-Map in OBKF has domain concepts and relations. If a user defines an instance of a concepts, the K-Map shows which concepts should be defined according to the concept's relations. This mechanism is useful when novice users do not know which information they should define. Lastly, comprehensive reasoning ability is required to manage three knowledge types in SKD. It is often happened that the three different knowledge types is applied to a same problem. On the other hand, there may be a conflicts during the application of three knowledge types. We should be able to handle all the cases which can be happened in reasoning processes. Fig. 7 shows the OBKF system architecture which meets the requirements.

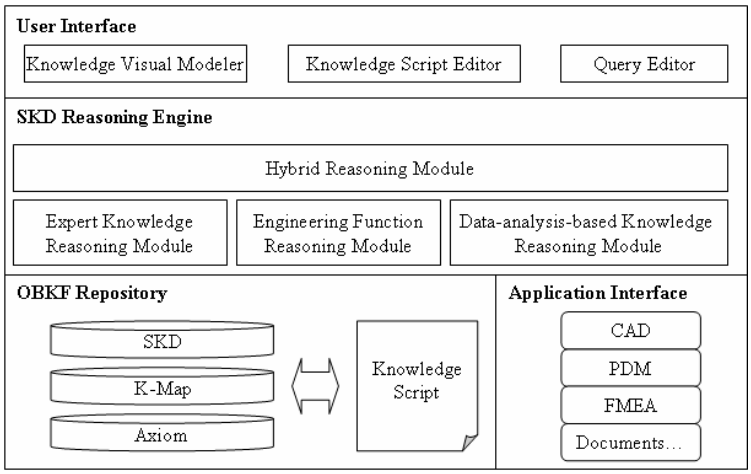

Fig. 7. OBKF System Architecture

User Interface: The user interface is composed of three components. First one is a visual modeler, second one is a script editor, and last one is a query editor. When we design the user interface, we should consider users' ability. Generally they are not familiar with any knowledge representations. So they need an intuitive knowledge modeling interface. Since K-Map comes from the semantic network, users can model it as a network diagram with a visual

modeler. However axioms and SKD can not be described in a visual modeling manner, so the knowledge scripts are required for users to represent them. The script editor is useful for the experts of knowledge engineering. The query editor helps users describe queries which describe what they want to know. It also can be an interface to execute the queries.

SKD Reasoning Engine: The three different SKD have different knowledge structure each others. For example, the expert knowledge has an IF-THEN form, but the engineering function has a mathematical equation form. Data-analysis-based knowledge has an experimental data which is the source of the knowledge, and the knowledge is depends on the data. Thus, we can not handle them with a single inference engine. The three reasoning modules in Fig. 7 are in charge of each SKD type, and the hybrid reasoning module is responsible for the integrated control of the three reasoning modules. If there are any conflicts or competitions, the hybrid reasoning module resolves them by user interactions, predefined rules, or trade-offs.

OBKF Repository: Since OBKF system manages data as well as knowledge in a manufacturing company, a repository should be considered to handle those data and knowledge. The structure of OBKF can be a reference to design the repository. The structures of SKD, K-Map, and axioms are reflected in the schema of the repository. In addition, we consider a script manner to handle those data and knowledge. Because many knowledge representations are described in script manner, and reasoning modules also require such a script for their processing. Therefore, we also need a script repository.

Application Interface: Since legacy systems have been accumulated a lot of information, we can extract product development knowledge from them. So the OBKF system needs application interfaces to the legacy systems such as CAD, ERP, SCM, and others. The interfaces require schema mapping between OBKF database and the other applications. Following the mapping, the information of the legacy systems can be imported as instances of K-Map.

6. A PROTOTYPE SYSTEM OF OBKF

In this chapter, we show the prototype system of the ontology-based knowledge framework, which is applied to a 'chair' product development domain. The prototype shows the three levels of knowledge graphically. The full FOL descriptions of the examples are more developed on the web site [10].

6.1. OBKF Prototype System

We implement the full FOL representation of OBKF using Prolog. In this implementation, predicates, variables, functions, logical connectivity, and etc. are expressed in Prolog format which may not be as those in FOL[12]. Prolog descriptions of the examples are fully developed on the web site [12]. Furthermore, we develop a prototype application for OBKF. The main screen shot of application is shown in Fig. 8.

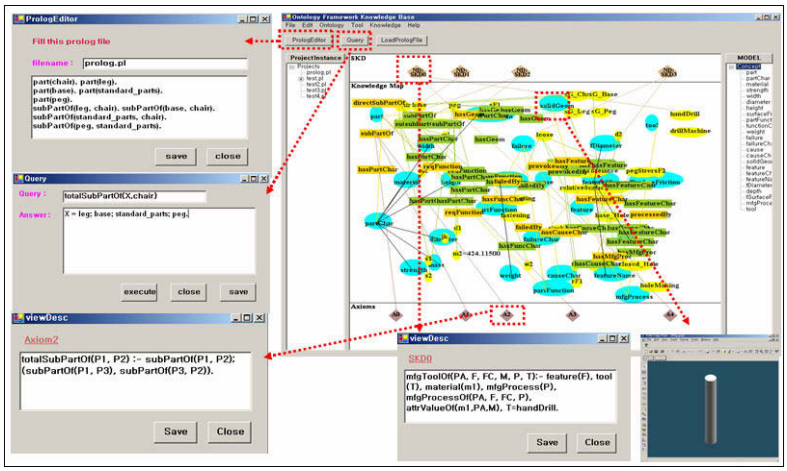

Fig. 8. The Screen Shot of OBKF Prototype.

The application supports users editing the ontology, visualizing the ontology, executing queries. We utilize the SISTUS-prolog program and Java to execute the Prolog queries, and C# language to visualize the ontology in window form. In addition to that, the prototype guides a product designer with the K-Map; which information should be

entered. The sample SKD knowledge provides answers for some sample queries. The detail role of each level of OBKF in the query processing is described in the following section. We also implement partial integration with commercial CAD and PDM system through the prototype. The updated information of CAD or PDM systems can be imported into the prototype, and the information in the prototype also can be exported into those systems.

6.2. Queries with the OBKF System

The implemented OBKF can be used in answering engineers' questions. So, we develop some sample queries, and show the utility of the OBKF. We will emphasize the use of axioms, K-Map and specialized knowledge for domain while the queries are resolved.

Query 1: (Example for the K-Map)

- Inputs:
  part(chair). part(leg). part(base). part(standard_parts). part(peg).
  subPartOf(leg, chair). subPartOf(base, chair).
  subPartOf(standard_parts, chair).
  subPartOf(peg, standard_parts).
- Query in English: What are the sub-parts of a 'chair' part?
- Query in Prolog: *?- subPartOf(X, chair).*
- Outputs: *X = leg; base; standard_parts; peg.*
  - ✓ Axioms: The *transitive axiom* of a 'subPartof' relation allows the system to traverse the product structure.
  - ✓ K-Map: The *Rel* function, *subPartOf(P1, P2) :- part(P1), part(P2).*, verifies whether each arguments of 'subPartOf' is a part.
  - ✓ SKD: *none*.

Query 2: (Example for the expert knowledge)

- Inputs:
  part(leg). material(m1). solidGeom(sG_Leg).
  feature(leg_Hole). featureName(closed_Hole).
  tool(handDrill).
  hasPartChar(leg, m1). hasGeom(leg, sG_Leg).
  hasFeature(sG_Leg, leg_Hole).
  hasFeatureChar(leg_Hole, closed_Hole).
  hasMfgProc(leg_Hole, holeMaking).
- Query in English: What tool is appropriate for making a closed hole of a 'leg' part?
- Query in Prolog: *?- toolOf(leg, closed_Hole, X).*
- Outputs: *X = handDrill*.
  - ✓ Axioms: The axiom, a part should have at least one solid geometry, serves a guarantee of the K-Map's integrity.
    part(X) :- solidGeom(Y), hasGeom(X, [Y|_]).
  - ✓ K-Map: All inputs come from the K-Map. The *Rel* functions of all relations verify weather each arguments are correct.
  - ✓ SKD: The expert knowledge SKD-1 in Fig. 6 is applied.
    toolOf(X, closed_Hole, handDrill) :- part(X), material(A), attrValueOf(A, X, wood), hasGeom(X, Z), hasFeature(Z, F), hasFeatureChar(F, FC), FC = closed_Hole, tool(handDrill).

7. CONCLUSIONS

We suggested Ontology-Based Knowledge Framework for the systematic storing and utilization of engineers' knowledge in product development environments. The reason why OBKF can be the framework of knowledge management is that it serves explicit knowledge structure, and it is represented in a uniform representation; FOL. Although OBKF has the structure which can consistently represent from fundamental semantics of concepts and relations to domain knowledge, further researches still remain to make it practical and commercial. Researches for an integrated product ontology and best practice for OBKF will encourage others to focus on knowledge management issues for the collaborative product development domain.

REFERENCES

[1] Bozsak, E., Ehrig, M., Handschuh, S., Hotho, A., Maedche, A., Motik, B., Oberle, D., Schmitz, C., Staab, S. and Stojanovic, L., KAON - Towards a Large Scale Semantic Web, *Lecture notes in computer science*, No.2455, 2002, pp 304-313.

[2] Corcho, O. and Perez, A.G., A Roadmap to Ontology Specification Languages, *Lecture notes in computer science*, No.1937, 2000, pp 80-96.

[3] Court, A.W., The relationship between information and personal knowledge in new product development, *International Journal of Information Management*, Vol.17, No.2, 1997, pp 123-138.

[4] Court, A.W., Issues for integrating knowledge in new product development: reflections from an empirical study, *Knowledge-Based System*, No.11, 1998, pp 391-398.

[5] Cutkosky, M.R., Engelmore, R.S., Fikes, R.E., Genesereth, M.R., Gruber, T.R., Mark, W.S., Tenenbaum, J. M. and Weber, J. C., PACT: an experiment in integrating concurrent engineering systems, *Computer*, Vol. 26, No. 1, 1993, pp 28-37.

[6] Ferguson, E.S., *Engineering and the Mind's Eye*, MIT Press, Cambridge, MA, 1992.

[7] Heflin, J., OWL web ontology language use cases and requirements W3C Recommended, February, 2004. http://www.w3.org/TR/webont-req/

[8] Kuffner, T.A. and Ullman, D.G., The information requests of mechanical design engineers, *Design Studies*, Vol. 12, No. 1, 1997, pp 42-50.

[9] Kuokka, D. R., McGuire, J. G., Pelavin, R. N. and Weber, J. C., SHADE: Technology for knowledge-based collaborative engineering, *Artificial intelligence in collaborative design*, 1994, pp. 245-262.

[10] Lee, J.H., September 1, 2004. http://143.248.80.90/CAD05/FOL.html

[11] Lee, J.H., September 1, 2004. http://143.248.80.90/CAD05/PROLOG.html

[12] Lin, J., Fox, M.S. and Bilgic, T., A Requirement Ontology for Engineering Design, *Concurrent engineering, research, and applications*, Vol. 4, No.3, 1996, pp 279-292.

[13] Russel, S. and Norvig, P., Artificial Intelligence, 2nd edition, *Prentice Hall*, 1995, p.255.

[14] Russel, S. and Norvig, P., Artificial Intelligence, 2nd edition, *Prentice Hall*, 1995, pp.272-315.

[15] Staab, S. and Maedche, A., Axioms are objects, too: Ontology Engineering Beyond the Modeling of Concepts and Relations, *Workshop on Ontologies and Problem-Solving Methods*, Berlin, 2000.

[16] Stauffer, L.A., and Ullman, D.G. Fundamental process of mechanical designers based on empirical data, *Journal of Engineering Design*, Vol. 2, 1991, pp. 113-125.

[17] Studer, R., Benjamins, V.R., and Fensel, D., Knowledge engineering: principles and methods, *Data knowledge engineering*, Vol. 25, 1998, pp. 161-197.

[18] Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer, R., and Wenke, D., OntoEdit: Collaborative Ontology Development for the Semantic Web, *Lecture notes in computer science*, No. 2342, 2002, pp. 221-235.

[19] Syan C.S., and Menon U., Concurrent Engineering: Concepts, implementation and practice, Chapman & Hall, 1994.

[20] Vincenti, W.G., *What engineers know and how they know it: analytical studies from aeronautical engineering*, John Hopkins University Press, 1990.

[21] Yoshioka, M. and Tomiyama T., Pluggable metamodel mechanism: A framework of an integrated design object modelling environment, *Computer Aided Conceptual Design `97*, Proceedings of the 1997 Lancaster International Workshop on Engineering Design CACD'97, Lancaster University, 1997, pp. 57-70.