International Journal of CAD/CAM Vol. 5, No. 1, pp. 99 ~ 109 (2005)

INTERNATIONAL JOURNAL OF CAD/CAM
www.ijcc.org

Ontology-Based Multi-level Knowledge Framework for a Knowledge Management System for Discrete-Product Development

Jaehyun Lee and Hyowon Suh*
Department of Industrial Engineering, KAIST, Daejeon, Korea

Abstract – This paper introduces an approach to an ontology-based multi-level knowledge framework for a knowledge management system for discrete-product development. Participants in a product life cycle want to share comprehensive product knowledge without any ambiguity and heterogeneity. However, previous knowledge management approaches are limited in providing those aspects: therefore, we suggest an ontology-based multi-level knowledge framework (OBMKF). The bottom level, the axiom, specifies the semantics of concepts and relations of knowledge so ambiguity can be alleviated. The middle level is a product development knowledge map; it defines the concepts and the relations of the product domain knowledge and guides the engineer to process their engineering decisions. The middle level is then classified further into more detailed levels, such as generic product level, specific product level, product version level, and manufactured item level, according to the various viewpoints. The top level is specialized knowledge for a specific domain that gives the solution of a specific task or problem. It is classified into three knowledge types: expert knowledge, engineering function knowledge, and data-analysis-based knowledge. This proposed framework is based on ontology to accommodate a comprehensive range of knowledge and is represented with first-order logic to maintain a uniform representation.

Keywords: **Ontology, Knowledge framework, Knowledge map, Expert system.**

1. Introduction

During the early 1990s, many researchers studied concurrent engineering to reduce the time-to-market [28]. To shorten the duration of product development, systematic management of product knowledge is required. Designers spend more than 70% of their working time searching and handling recently updated knowledge: such an unnecessary waste of time decreases the productivity of designers [17]. Stauffer *et al.* [26] studied why engineers use too much time to utilize the knowledge gained from past projects; the problem is that the past knowledge is not well organized. One reason for this problem is that designers do not have enough time to arrange the information and knowledge they have gained; moreover, companies not only disregard this knowledge as their asset, but they also do not budget enough for knowledge management (KM). According to the results of Court's empirical study [9], designers use about 30% of their personal knowledge during product development. However, in some cases, designers use up to 70% of their personal knowledge. Therefore, increasing the utilization of engineers' personal knowledge and knowledge sharing through KM is a critical leverage point in product development.

KM can be considered as the four basic processes of creating, storing/retrieving, transferring, and applying knowledge [1]. These four KM processes can be archived more easily if all relevant data, information, and knowledge are stored in an integrated knowledge-base. The knowledge-base should have a knowledge framework that accommodates comprehensive knowledge unambiguously using a uniform representation. However, the knowledge of engineer groups or knowledge of an individual engineer is currently scattered over numerous engineering documents, CAD models, engineering databases, expert systems, and so on. Consequently, we have designed an integrated knowledge framework to manage the explicit knowledge externalized from various knowledge sources and to store the knowledge with the related data and information integrally. Both managing the implicit knowledge and externalizing the implicit knowledge into explicit knowledge are indirectly supported by our approach.

This paper introduces an ontology-based multi-level knowledge framework for comprehensive knowledge management. The proposed knowledge framework suggests a structure for knowledge storage and retrieval. Previously, several knowledge management approaches have been proposed. However, these approaches were

*Corresponding author:
Tel: +82 (42) 869 3123
Fax: +82 (42) 869 3110
Homepage: http://ie1.kaist.ac.kr/~hwsuh/
E-mail: hw_suh@kaist.ac.kr

limited in accommodating several types of comprehensive knowledge without ambiguity and heterogeneity. The proposed approach is based on ontology, which is an emerging technology, but is not currently fully developed for knowledge management. In addition, we provide the lifecycle of our framework-based knowledge base. The lifecycle explains how to construct and use the knowledge framework.

In section 2, we discuss previous research and describe the proposed approach, an ontology-based multi-level knowledge framework, in section 3. The details of each level of the knowledge framework are discussed in sections 4 and 5. In section 6, we describe the lifecycle of our knowledge framework to explain how to define and use the proposed framework. Section 7 details the prototype built based on the proposed structure, and finally in section 8, a summary and considerations for future research are provided.

2. Previous Research

Product Development Knowledge: Court [9] discusses the three basic types of knowledge that engineering designers use and access to during their work:

- General knowledge: gained through everyday experiences and general education.
- Domain-specific knowledge: gained through study and experience within the specific domain where the designer works.
- Procedural knowledge: gained from experience in undertaking tasks within a domain.

On the other hand, Vincenti [29] suggested six categories of knowledge that most engineers have or use: fundamental design concepts, criteria and specifications, theoretical tools, quantitative data, practical considerations, and design instrumentalities. Ferguson [12] also proposed that engineering knowledge should include knowledge gained from the experimental data as well as the experience of experts.

The above studies classified engineering knowledge into several categories. However, the criteria for classification are not clear, and the concrete relationships between the categories are not discussed. Clear classification criteria and concrete relationships of the knowledge need to be defined so that they can be accommodated with the knowledge framework.

Engineering Knowledge Framework: Several previous studies adopted different knowledge frameworks for knowledge management systems. There have been several typical types of knowledge framework: PACT [10] and SHADE [18] developed agent-based collaborative product design systems based on ontology. Each agent had different problem-solving knowledge and the ontology was utilized to share the knowledge-base. This study showed the relationship between problem-solving knowledge and ontology, but it did not specify a detailed structure of the ontology. Recently, some ontology-based knowledge frameworks have been proposed. OntoEdit [27] enables engineers to define ontology and its rules. These rules can be viewed as problem-solving knowledge. This knowledge framework showed a detailed structure of ontology and the role of axioms, but it did not specify the relationship between ontology and the problem-solving knowledge. Another approach is Staab and Maedche [25]'s study: they separated the structure of ontology into axioms and the remaining parts. They classified the axioms and suggested some templates to easily define these axioms. Yoshioka *et al.* [31] proposed a 'meta-model' approach to integrate different design systems. Each of these systems had different problem-solving knowledge, and the 'meta-model' corresponded with the knowledge map. The semantics of the concepts were described in a concept dictionary, aiming to avoid the burden of developing axioms. However, problems remained in maintaining the semantics of the meta-model because of its informal specification. This model also showed the detailed roles of the knowledge map, but it ignored the role of formally specified axioms.

3. Ontology-Based Multi-level Knowledge Framework

We propose an ontology-based multi-level knowledge framework (OBMKF). 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 within the frame. Furthermore, for a uniform representation, the first-order logic (FOL) is adopted.

The previous studies suggest various types of product development knowledge. However, the criteria for the classification of these various types of knowledge are not clear and the concrete relationships between the categories are not discussed. We define two classification criteria: 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 knowledge criteria classifies the *task-specific knowledge* into three types according to the source: *engineering function knowledge, expert knowledge*, and *data-analysis-based knowledge*. Further details of the knowledge structure are discussed in section 4.

Bozsak *et al.* [6] define an ontology structure with six tuples: *concepts, relations, concept hierarchies, relation hierarchies, functions,* and *axioms*. This ontology structure is considered to be similar to the knowledge modeling ontologies defined by Heijst [16]. These ontology components can be used as a base in a

knowledge framework. The *concepts and relations* tuples generally represent the basic structure of a domain knowledge. Thus, the *common domain knowledge* can be represented by *concepts* and *relations*, including *concept hierarchies*, *relation hierarchies*, and *functions*. On the other hand though, *axioms* specify the semantics of *concepts* and *relations* so that the *semantics of knowledge* can be represented by *axioms*. In addition, the *task-specific knowledge* can also be defined using ontology because it specifies the quantified relationship between the *concepts* of ontology, which is the relationship between their instances. Thus, the three levels of knowledge are organized distinctly using an ontology structure. Among the three levels, the *common domain knowledge* level has similar concepts to the levels of ontology defined by Guarino [13]. This will be discussed further in section 5.

Ontology is usually expressed in a logic-based represen -tation; thus, detailed, accurate, consistent, sound, and meaningful distinctions among the concepts and relations can be made [15]. There are two typical representation approaches to describe ontology : FOL and description logic (DL). FOL has an appropriate syntax to describe the properties of objects; it provides inference algorithms such as forward chaining and backward chaining. KIF [21], Ontolingua [11], and Prolog [7] are FOL-based systems, and many studies, such as PSL [3], TOVE [22], and Engineering Ontology [4], use FOL representation approaches. On the other hand, DL has notations that are designed to easily describe definitions and properties of categories; it provides inference algorithms which are related to categorization, such as the structured subsumption algorithm and tableau algorithm [2]. CLASSIC, LOOM, and KRIS are DL-based systems [2].

The uniform representation should be as expressive as possible to convey the complex knowledge of the product development domain. In addition, the inference capability of the logic system should be as powerful as possible. Corcho and Perez [8] compared the expressiveness and the inference capability of several logical languages. They stated that there is a trade-off between the degree of expressiveness and the inference efficiency of a language: more expressive representations require rigorous inference capabilities.

Although the expressiveness of FOL and DL cannot be compared by themselves, the DL has an expressive power of certain subsets of the FOL, possibly augmented by counting quantifiers [5]. In addition, FOL can represent rules, but a pure DL cannot. Meanwhile, the inference capabilities of the two logics are also different: the inference capability of FOL focuses on how to process rules, but the inference capability of DL focuses on how to categorize concepts. Volz *et al.* [30] and Matheus *et al.* [23] studied the integration of OWL, which is a DL-based ontology language, with FOL rules to utilize the inference capabilities of FOL. However, it is not yet fully developed for practical application. Because the expressiveness of logic and the inference capability are important for many levels in our framework, we adopt FOL as a uniform representation. The FOL representation has sufficient expressiveness to represent the axioms as well as the concepts and relations; it also provides inferencing algorithms that can be a basis for task-specific knowledge [24].

4. Multiple Levels of Knowledge

As discussed above, we classify product development knowledge into three levels: *semantics of knowledge*, *task-specific knowledge*, and *common domain knowledge*. Hereafter, these levels will be called *Axiom, Knowledge Map* (*K-Map*), and *Specialized Knowledge for Domain* (*SKD*), respectively, to emboss the comprehensive ontology -based knowledge frame, as shown in Fig. 1. SKD is also further classified into three types and the K-Map has schema and multiple instance levels.

4.1. Axioms

The axioms specify the semantics of the concepts and relations in a domain using a logical representation to ensure both people and computers clearly understand the meaning without ambiguity. We define the axioms as fundamental properties of the concepts and relations as well as definitions. For example, let the '*subPartOf*' terminology represent a structure relation between two part concepts. The fundamental properties of the '*subPartOf*' relation are as follows:

- the '*subPartOf*' relation is irreflexible: a part cannot be a sub-part of itself.
- the '*subPartOf*' relation is anti-symmetrical: if part X is a sub-part of part Y, then part Y is not a sub-part of part X.
- the '*subPartOf*' relation is transitive: if part X is a sub-part of part Y and part Y is a sub-part of part Z, then sub-part X is a sub-part of part Z.

The definitions of the concepts and relations also can be described in logic. For a simple example, let one part be a direct sub-part of another part when no middle part exists between the two parts. We can define the '*directSubPartOf*' relation in logic:

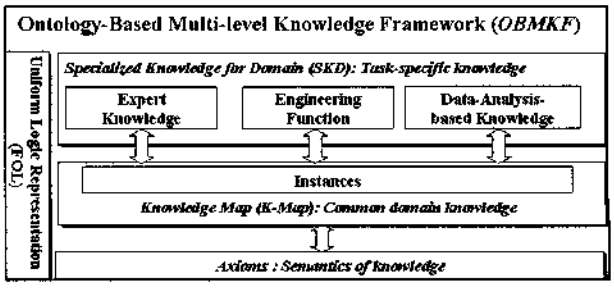


Fig. 1. Architecture of the Ontology-Based Multi-Level Knowledge Framework.

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	$(\forall p1, p2)\ subPartOf(p1, p2) \wedge \neg(\exists p3\ subPartOf(p3, p2) \wedge subPartOf(p1, p3)) \Leftrightarrow directSubPartOf(p1, p2)$

Fig. 2. Example of Axioms

- part X is a direct sub-part of part Y if, and only if, part X is a sub-part of part Y and a part Z does not exist such that part Z is a sub-part of part Y and part X is a sub-part of part Z.

We can translate the axiom into the FOL formulae, as shown in Fig. 2.

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

4.2. Knowledge Map (K-Map)

Fig. 3. shows the K-Map of a product development domain. The K-Map describes the common knowledge of a domain with a semantic network structure. The structure of the K-Map is composed of concepts, relations, relation functions, concept hierarchies, and relation hierarchies. The concept '*Part*' and the relation '*subPartOf*' are the realization or perception of real world objects and their relation by humans. The relation function, Rel, defines relationships between relations and concepts. If a '*subPartOf*' relation is related with two '*Part*' concepts, we can describe this logic as '*Rel*(*subPartOf*) = (*Part*, *Part*)'. The concept hierarchy is called taxonomy and is described with a concept hierarchy function 'H^C'. If the concept '*Assembly*' is a sub-concept of the concept '*Part*', then it can be represented in logic as 'H^C (*Assembly*, *Part*)'. The relation hierarchy can also be described in logic by the relation hierarchy function 'H^R'. For example, the '*directSubPartOf*' relation is a sub-relation of the '*subPartOf*' relation; therefore, it can be represented in logic: 'H^R(directSubPartOf, subPartOf)'. The relational hierarchy is omitted in Fig. 3. The K-Map can also be represented in the FOL formulae [19]. The K-Map should be agreed upon by domain experts and then used as the base of SKD. The agreed K-Map can be a reference model of a domain, and it can be further classified into several levels according to differing viewpoints.

4.3. Specialized Knowledge for Domain (SKD)

The SKD supports users' decisions because it provides solutions for users' problems. The SKD is represented based on the K-Map of the domain and provides a quantified relation between the instances of the K-Map. Since the SKD is concerned with specific tasks or problems, the logical formulae of the SKD are described with the concepts and relations in the K-Map and their instances.

We classify the SKD into three types according to the sources of knowledge: *engineering function knowledge*, *expert knowledge*, and *data-analysis-based knowledge*. Firstly, the *engineering function knowledge* comes from scientific theories and is represented by mathematical expressions. The *engineering function knowledge* is the numerical engineering relation between the concepts of K-Map. The following equation with a width of 'Leg' part and other factors is a good example. In the example, *Ws*, *X*, *St*, and *SF* are 'concepts' of the K-Map.

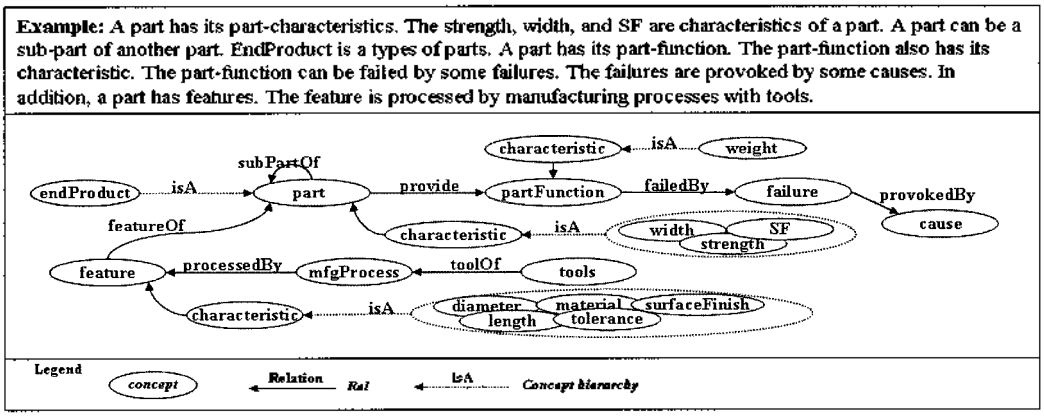


Fig. 3. Example of Graphical K-Map

(SKD-1):
$X = (Ws / St \times SF)^{1/2}$
X: width of a Leg; 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 is generally qualitative knowledge. The *expert knowledge* has an 'IF THEN' form similar to a production rule. The following logic is an example of the *expert knowledge*.

(SKD-2):
IF *feature.type* = *Closed_circular_hole* THEN *mfgProcess* = *Drilling*.
IF *mfgProcess* = *Drilling* AND *feature.material* = Wood THEN *tool* = *HandDrill*.

In the example, *feature.type*, *feature.material*, *mfgProcess*, and *tool* are the *concepts* from the K-Map, and *Closed_circular_hole*, *Wood*, *Drilling*, and *HandDrill* are the *instances* of those concepts. Thus, this knowledge provides solutions for problems or tasks.

Lastly, we can source knowledge from a burden of accumulated design data. We define this knowledge as *data-analysis-based* knowledge. Since it is extracted from the analysis of collected data using certain analysis methods, it has relations with specific data-analysis methods and the collected data. For an example of a *data-analysis-based knowledge*, the relation between the *Slackness* (an instance of 'FailureChar') and its correlated factors, *Hole & Peg Tolerance*, *Relative Surface Friction* (instances of 'CauseChar'), and Peg strength (instances of 'PartChar'), may be mined from a data cube with a linear regression model and a continuous relation can be obtained. The data cube concerning the correlated factors can be obtained from data tables that have accumulated data of the factors. The following logic is an example of the *data-analysis-based knowledge*.

(SKD-3):
$Y = -0.06^* X1 + -29.9^* X2 + 27^* X3 + 5.12$
Y: Slackness; *X1 : HolePegTolerence*; *X2 : Relative-SurfaceFriction*; *X3 : PegStrength*

The examples of SKD can also be represented in the FOL formulae. Fig 4 shows FOL examples of each

Type of SKD	FOL formulae of examples
Engineering function(SKD-1):	$(\forall\ w1\ w2\ w3)\ Width(w1) \wedge Weight(w2) \wedge MaterialStrength(w3) \wedge (SF = 3) \wedge W1 = squar((* (/ w2\ SF)\ w3))$.
Expert knowledge(SKD-2):	$(\forall\ y\ z1\ p)\ Feature(y) \wedge hasFeatureChar(y\ z1) \wedge FeatureType(z1) \wedge (= z1\ ClosedCircularHole) \Rightarrow (= p\ Drilling) \wedge mfgProcess(p) \wedge hasMfgProcess(y, p)$. $(\forall\ y\ c\ p\ t)\ Feature(y) \wedge Material(c) \wedge hasFeatureChar(y, c) \wedge ValueOf(c, wood) \wedge hasMfgProcess(y\ p) \wedge MfgProcess(p) \wedge (= p\ Drilling) \Rightarrow Tool(t) \wedge (= t\ HandDrill) \wedge ProcessedBy(p\ t)$
Data-analysis-based knowledge(SKD-3):	$(\forall\ y\ x1\ x2\ x3)\ Slackness(y) \wedge HolePegTolerence(x1) \wedge RelativeSurfaceFriction(x2) \wedge PegStrength(x3) \wedge (= y\ (+ 5.12\ (+ (* (-0.06)\ x1)\ (+ (* (-29.9)\ x2)\ (* 27.0\ x3)))))$.

Fig. 4. An Example of SKD.

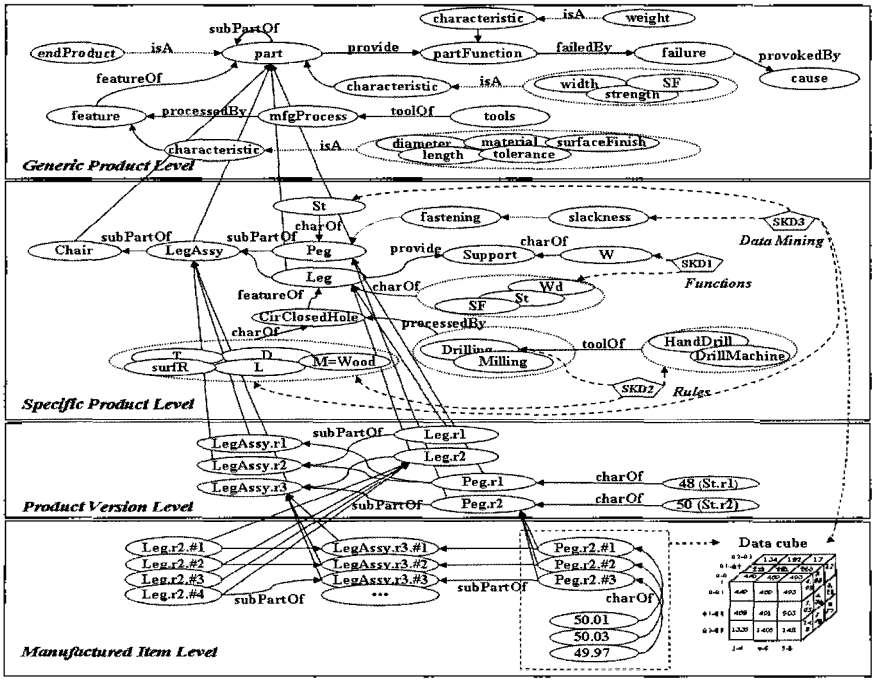


Fig. 5. 'Chair' Example of the Multi-levels of K-Maps

SKD type. When we implement OBMKF, the FOL formulae can be expressed differently according to the specific implementation language, such as Prolog, CLIPS, KIF, and others.

5. Multi-Levels of K-Map

In section 4.2, it shows that the K-Map has concepts and related instances. The concepts are related to a generic product and the instances are related to a specific product, such as a chair. Moreover, when design changes are required, several versions of the design need to be managed. In addition, the *data-analysis-based knowledge* requires information about the manufacturing item. Thus, the K-Map can be divided into several levels, such as generic product level, specific product level, product version level, and manufacturing item level, to cover the entire issues of a product lifecycle. Each level may have instantiation relationships with higher levels, while a higher level is a meta-model of the adjacent lower level. This relationship is similar to the relationships between the levels of ontology as defined by Guarino [13]. Both the SKD and Axioms are integrated into the four levels according to the application. However, since most necessary semantics of knowledge are defined in the generic product and specific product levels, axioms are especially important for those levels. Both the *expert knowledge* and *engineering function knowledge* are generally defined in the generic product level or specific product level, and consequently used for the product version level or the manufacturing item level. Although the *data-analysis-based knowledge* can also be defined in the generic product level or specific product level, they are generally developed using analyses of the data from the product versions or manufacturing items.

We show a visualized multi-level knowledge map for a 'Chair' product in Fig. 5. This shows the concepts and relations at each level and the instantiation relationships between the four levels.

6. OBMKF Lifecycle

In this section, we discuss the lifecycle of an OBMKF: it briefly shows how to build the knowledge base (KB) with an OBMKF, how to apply the KB to product development, and how to update the OBMKF-based KB (OBMKF-KB).

Building Stage: The building stage builds the OBMKF-KB including the K-Map, Axioms, and SKD. This stage is performed by domain experts and knowledge engineers. In building the OBMKF-KB, the K-Map is defined initially because the K-Map is common ground knowledge for the given domain. Next, the Axioms need to be defined because they are explicit specifications of the concepts and relations of the K-Map. Finally, the SKD should be built based on the concepts and relations of the K-Map and Axioms because the SKD is specialized knowledge for specific problems and requires the K-Map and Axioms. In this building stage, two levels of the K-Map, such as the generic product and specific product levels, need to be defined. The K-Map for a generic product is generated prior to the K-Maps of specific products because the K-Maps of specific products are instantiated from the K-Map of the generic product. Building the K-Maps, Axioms, or SKDs for generic products or specific products is a knowledge engineering process where knowledge engineers externalize the domain experts' implicit knowledge into explicit knowledge. Thus, our knowledge framework of OBMKF can be used not only to put the explicit knowledge in use, but also to acquire implicit knowledge.

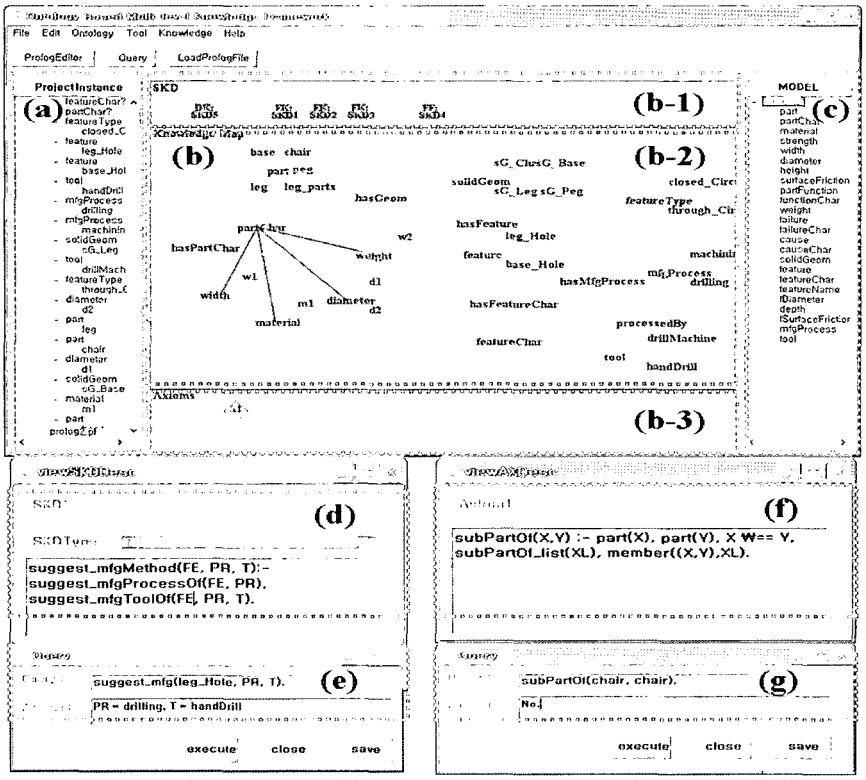
Application Stage: Applying the OBMKF-KB to product development means that product design engineers can develop their projects with the support of the KB, especially at the generic or specific product level. While they are developing their projects using the KB, their design outputs are stored at the product versions level of the K-Maps. Therefore, the product version levels of the K-Maps accumulate in the KB for every development project.

The K-Maps, Axioms, and SKDs developed in the building stage are used to guide design engineers or decide design parameters. Since the K-Maps have concepts and relations visualized as semantic networks, they can guide engineers to navigate the design parameters and their relationships. The axioms of each K-Map not only inform design engineers about the meaning of the concepts and relations of the K-Maps, but they are also utilized to infer new knowledge or validate it. Since the SKD is a type of task-specific knowledge, design engineers directly use it to solve their problems. Whenever engineers try to define the design data, they should check whether any related SKDs exist or not. If a related SKD exists, they should review the related SKD and decide whether or not to apply it to the design data.

Maintaining Stage: Maintaining or updating the KB needs to be done by domain experts and knowledge engineers. Updating the KB is related mainly to the generic or specific product levels. This maintenance updates a specific product level of a K-Map with the information from the product versions level, which is accumulated through the product development projects. Knowledge engineers can define new knowledge and revise existing knowledge to accommodate the new knowledge. In addition, the SKD and Axioms can be newly defined.

7. Prototype OBMKF

Prototyping of an OBMKF: We implemented a full



(a) Instance tree-view (b) Navigator (c) Concept tree-view (b-1) SKD area (b-2) K-Map area
(b-3) Axiom area (d) SKD window (e, g) Query window (f) Axiom window

Fig. 6. Main Screen-shot of OBMKF Prototype.

FOL representation of an OBMKF using Prolog. In this implementation, the predicates, variables, functions, logical connectivity, and so on are expressed in the Prolog format, which may not be the same as in FOL. The Prolog descriptions of the examples are fully developed on our website [20]. Furthermore, we developed a prototype system: the main screen shot of the prototype system is shown in Fig. 6.

The application supports users editing the ontology, visualizing the ontology, and executing queries. Figs 6(a), (b), and (c) show the basic user interfaces of the prototype. The tree-view on the right-side of the prototype shows the generic concepts for product development. These concepts are imported from the Prolog files. The visual navigator in the middle of the prototype shows the K-Map in a graphic manner. If an end-user drags a concept from the right-side tree-view and drops it into the navigator, the prototype system draws the concept and finds relations related to the concept. If a user inputs an instance of the concepts, the prototype suggests user input related instances of other concepts that have relations with the instantiated concept. This function allows a user to know what inputs are required based on the product ontology. All instances defined by the user are drawn in the visual navigator and listed on the left-side tree-view. The left-side tree-view classifies the concepts and instances between projects.

The navigator has many concepts and instances (Fig. 6(b-2)), and it also shows the SKD (Fig. 6(b-1)) and axioms (Fig. 6(b-3)) which are linked with the related concepts. The visual navigator has many nodes and edges so users may not easily identify what they want to know. Therefore, we can provide a categorized visualizing method to screen concepts and relations according to the users' viewpoints. The viewpoints include a taxonomical view, a specific-part view, specific -characteristic views, and so on. The end-user can define and review the SKD and axioms using other windows, i.e. Fig. 6(d) and (f). In addition, the user can ask Prolog queries through the query window (Fig. 6(e) and (g)). We utilized the SISTUS-prolog program and Java to execute the Prolog queries. We used C# language to visualize the K-Map in a window format.

Queries with the OBMKF Prototype: The implemented OBMKF can be used in answering engineers' questions. Thus, we developed some sample queries and to show the utility of the OBMKF. The detailed role of each level of the OBMKF, Axioms, K-Maps, and SKD, in the query processing is also described in the following queries and answers.

Query 1 : (Example of the knowledge map)

Query in English : What are the sub-parts of a 'chair' part?

Query in Prolog : ?- subPartOf(chair, chair).
Answer : No.

Knowledge Base

Concepts of generic product level – Concepts of specific product level

```
part(X) :- part_list(XL), member(X,XL). part_list([chair, leg, base, leg_parts, peg]).
partChar(X) :- material(X) ; strength(X) ; width(X) ; diameter(X) ; height(X) ; mass(X) ;
surfaceFriction(X).
material(X) :- material_list(XL), member(X,XL). material_list([m1]).
strength(X) :- strength_list(XL), member(X,XL). strength_list([s1, s2]).
width(X) :- width_list(XL), member(X,XL). width_list([w1]).
diameter(X) :- diameter_list(XL), member(X,XL). diameter_list([d1, d2]).
height(X) :- height_list(XL), member(X,XL). height_list([h1]).
mass(X) :- mass_list(XL), member(X,XL). mass_list([m2]).
surfaceFriction(X) :- surfaceFriction_list(XL), member(X,XL). surfaceFriction_list([sF1, sF2]).
partFunction(X) :- partFunction_list(XL), member(X,XL). partFunction_list([supporting, fastening]).
functionChar(X) :- weight(X). weight_list([w2]).
```

Concepts of generic product level – Concepts of specific product level

```
weight(X) :- weight_list(XL), member(X,XL).
failure(X) :- failure_list(XL), member(X,XL). failure_list([loose]).
failureChar(X) :- failureChar_list(XL), member(X,XL). failureChar_list([slackness]).
cause(X) :- cause_list(XL), member(X,XL).
cause_list([holePegTorelance, relativeSurfaceFriction, pegStrength]).
causeChar(X) :- causeChar_list(XL), member(X,XL). causeChar_list([t1, rF1, pS1]).
solidGeom(X) :- solidGeom_list(XL), member(X,XL).
solidGeom_list([sG_Chair, sG_Leg, sG_Base, sG_Peg]).
feature(X) :- feature_list(XL), member(X,XL). feature_list([leg_Hole, base_Hole]).
featureChar(X) :- featureType(X) ; depth(X).
featureType(X) :- featureType_list(XL), member(X,XL).
featureType_list([closed_Circular_Hole, through_Circular_Hole]).
depth(X) :- depth_list(XL), member(X,XL). depth_list([d3]).
mfgProcess(X) :- mfgProcess_list(XL), member(X,XL). mfgProcess_list([machining, drilling]).
tool(X) :- tool_list(XL), member(X,XL). tool_list([drillMachine, handDrill]).
```

Relations of generic product level - Relations Instances

```
hasPartChar(X,Y) :- part(X), partChar(Y), hasPartChar_list(XL), member((X,Y),XL).
hasPartChar_list([(leg, s1), (leg, w1), (leg, m1), (peg, d1), (peg, h1), (peg, s2), (peg, sF1), (peg, m2)]).
subPartOf_list([(leg_parts, chair), (base, chair), (peg, chair), (leg, leg_parts)]).
reqFunction(X,Y) :- part(X), partFunction(Y), reqFunction_list(XL), member((X,Y),XL).
reqFunction_list([(chair, supporting), (chair, fastening)]).
hasFuncChar(X,Y) :- partFunction(X), functionChar(Y), hasFuncChar_list(XL), member((X,Y),XL).
hasFuncChar_list([(supporting, w2)]).
failedBy(X,Y) :- partFunction(X), failure(Y), failedBy_list(XL), member((X,Y),XL).
failedBy_list([(fastening, loose)]).
provokedBy(X,Y) :- failure(X), cause(Y), provokedBy_list(XL), member((X,Y),XL).
provokedBy_list([(loose, holePegTorelance), (loose, relativeSurfaceFriction), (loose, pegStrength)]).
hasCauseChar(X,Y) :- cause(X), causeChar(Y), hasCauseChar_list(XL), member((X,Y),XL).
hasCauseChar_list([(holePegTorelance, t1), (relativeSurfaceFriction, rF1), (pegStrength, pS1)]).
hasGeom(X,Y) :- part(X), solidGeom(Y), hasGeom_list(XL), member((X,Y),XL).
hasGeom_list([(chair, sG_Chair), (leg, sG_Leg), (base, sG_Base), (peg, sG_Peg)]).
hasFeature(X,Y) :- solidGeom(X), feature(Y), hasFeature_list(XL), member((X,Y),XL).
hasFeatureChar(X,Y) :- feature(X), featureChar(Y), hasFeatureChar_list(XL), member((X,Y),XL).
hasFeatureChar_list([(leg_Hole, m1), (leg_Hole, closed_Circular_Hole), (base_Hole, d2), (base_Hole,
    d3), (base_Hole, sF2) ]).
hasFeature_list([(sG_Leg, leg_Hole), (sG_Base, base_Hole)]).
hasMfgProcess(X,Y) :- feature(X), mfgProcess(Y), hasMfgProcess_list(XL), member((X,Y),XL).
hasMfgProcess_list([]).
processedBy(X,Y) :- mfgProcess(X), tool(Y), processedBy_list(XL), member((X,Y),XL).
processedBy_list([]).
attrValueOf(A, V) :- attrValue_list(AL), member((A,V),AL).
attrValue_list([(safetyFactor, 10), (w3, 5),(w2, 50), (pS1, 5.3), (t1, 0.03), (rF1, 0.02), (m1, wood) ]).
```

Axioms

```
subPartOf(X,Y) :- part(X), part(Y), X \== Y, subPartOf_list(XL), member((X,Y),XL).
subPartOf(X,Y) :- subPartOf(X,Z), subPartOf(Z,Y).
```

SKD

```
suggest_mfgMethod(FE, PR, T):- suggest_mfgProcessOf(FE, PR), suggest_mfgToolOf(FE, PR, T).
suggest_mfgProcessOf(FE, PR):- feature(FE), featureType(FC), hasFeatureChar(FE, FC), FC ==
    closed_Circular_Hole, mfgProcess(PR), PR = drilling,.
suggest_mfgToolOf(FE, PR, T):- hasFeature(GE, FE), hasGeom(PA, GE), part(PA), material(M),
    hasPartChar(PA, M), attrValueOf(M, wood), mfgProcess(PR), PR == drilling, tool(T), T =
    handDrill.
check_attrValueOf(w1,Z) :- attrValueOf(w2, Z1), attrValueOf(w3, Z2), attrValueOf(safetyFactor, SF),
    ground(Z), Z >= sqrt(Z1/Z2 * SF).
check_attrValueOf(slackness,Z) :- attrValueOf(t1,X1), attrValueOf(rF1,X2),attrValueOf(pS1,X3), Z is
    (27.0*X1-29.9*X2-0.06*X3+5.12).
```

Fig. 7. Knowledge-Base in Prolog

Related Axiom : The ir-reflexive axiom of a 'subPartof' relation prohibits that the subPartOf relation can not have two identical parts as its domain and range.
subPartOf(X,Y) : -part(X), part(Y), X \== Y, ...
Related K-Map : The Rel function, subPartOf (P1, P2) :-part(P1), part(P2), verifies whether each arguments of 'subPartOf' is a part.
Related SKD : none.

Query 2 : (Example of the expert knowledge)
Query in English : What method is appropriate for making a closed circular hole of a 'leg' part?
Query in Prolog : ?- suggest_mfgMethod (leg_Hole, PR, T).
Answer : PR = drilling, T = handDrill.
Axioms : none.
K-Map : All inputs come from the Knowledge Map. The Rel functions of all relations verify weather each arguments are correct.
SKD : an expert knowledge, 'suggest_mfgMethod (FE, PR, T):- suggest_mfgProcessOf(FE, PR), suggest_mfgToolOf(FE, PR, T).' is utilized to answer for the query 2.

Query 3 : (Example of the engineering function)
Query in English : How long is the width of 'leg' part?
Query in Prolog : ?- suggest_attrValueOf(w1, X).
Answer : X = 10.
Axioms : The axiom serves the guarantee of the knowledge map's integrity. Omitted.
K-Map : The role is like the role of K-Map in query 2. Omitted.
SKD : The following engineering function is utilized ; suggest_attrValueOf(w1,Z) : -attrValueOf (w2, Z1), attrValueOf(w3, Z2), attrValueOf (safetyFactor, SF), ground(Z), Z is sqrt(Z1/ Z2 * SF).

Query 4 : (Example of the data-analysis-based knowledge)
Query in English : What is the value of 'slackness' of the 'loose' failure?
Query in Prolog : ?-suggest_attrValue Of (slackness, X).
Answer : X = 5.014
Axioms : The axiom serves the guarantee of the knowledge map's integrity. Omitted.
K-Map : The role is same like the role of K-Map in query 2. Omitted.
SKD : The following data-analysis-based knowledge is utilized;suggest_attrValueOf (slackness,Z) :- attrValueOf(t1,X1), attrValueOf (rF1,X2),attrValueOf(pS1,X3), Z is (27.0*X1 −29.9*X2-0.06*X3+5.12).

8. Conclusions

We suggested an Ontology-Based Multi-level Knowledge Framework (OBMKF) for the systematic storing and utilization of engineers' knowledge in product development. The reason this framework is an appropriate framework of knowledge management is that it provides an explicit and comprehensive knowledge structure in a uniform representation for several aspects of domain knowledge.

Although the framework has a structure which can consistently represent information from fundamental properties of concepts and relations to task-specific knowledge, further research is still required to make it a practical and commercial framework. We developed a prototype and applied it to a simple product development. Even though the framework has not yet been applied to more complex products, we believe that it can be applied. Therefore, we should apply the OBMKF to a practical example and find any problems that occur when we extend the applied scope of the framework. In addition, the framework needs to be integrated with an inferencing mechanism to solve practical problems. An XML-based representation of the framework is also necessary for internet-based collaborative environments. Building an integrated product ontology and best practice for the framework will encourage others to focus on practical knowledge management issues.

References

[1] Alavi, M and Leidner D.E., (2001), review: Knowledge Management and Knowledge management Systems: conceptual foundations and research issues, mis quaterly, **25**(1), 107-136.
[2] Baader, F., Calvanese, D., Mcfuinness, D.L., Nardi, D. and Patel-schneider, P.F., (2003), *The description logic handbook*, Cambridge university press.
[3] Bock, C. and Gruninger, M., (2004), PSL: A Semantic Domain for Flow Models, *Software and Systems Modeling Journal*, **4**(2), 209-231.
[4] Borst, P., Akkermans, H., and Top, J., (1997), Engineering ontologies, *International journal of human-computer studies*, **46**(2/3), 365-406.
[5] Borgida, A., (1996), On the relative expressiveness of description logics and predicate logies, *Artificial Intelligence*, **82**, 353-367.
[6] Bozsak, E., Ehrig, M., Handschuh, S., Hotho, A., Maedche, A., Motik, B., Oberle, D., Schmitz, C., Staab, S. and Stojanovic, L. (2002), KAON - Towards a Large Scale Semantic Web, *Lecture notes in computer science*, **2455**, 304-313.
[7] Bratko, I. (2001), Prolog programming for artificial intelligence, *Pearson education*, **57**, ISBN-0201-40375-7.
[8] Corcho, O. and Perez, A.G. (2000), A Roadmap to Ontology Specification Languages, *Lecture notes in computer science*, **1937**, 80-96.
[9] Court, A.W. (1998), Issues for integrating knowledge in new product development: reflections from an empirical study, *Knowledge-Based System*, **11**, 391-398.

[10] 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. (1993), PACT: An experiment in integrating concurrent engineering systems, *Computer*, **26**(1), 28-37.

[11] Farquhar, A., Fikes, R. and Rice, J., (1997), The Ontolingua Server: a tool for collaborative ontology construction, *International journal of human-computer studies*, **46**(6), 707-727.

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

[13] Guarino, N. (1997). Understanding, building and using ontologies, *International journal of human-computer studies*, **46**(2/3), 293-310.

[14] Gruber, T.R. (1993), A Translation Approach to Portable Ontology Specifications, *Knowledge Acquisition*, **5**, 199-220.

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

[16] Heijst, G. V., Sabina, F., Ameen, A. H., Guus S., and Mario S., (1995), A case study in ontology library construction, *Artificial Intelligence in Medicine*, **7**, 227-255.

[17] Kuffner, T.A. and Ullman, D.G. (1997), The information requests of mechanical design engineers, *Design Studies* **12**(1), 42-50.

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

[19] Lee, J.H. September, (2004), http://143.248.82.98/DEW05/OBMKF.html.

[20] Lee, J.H. September, (2004), http://143.248.82.98/DEW05/PROLOG.html.

[21] Genesereth, M. R., Knowledge interchange format, http://logic.stanford.edu/kif/dpans.html.

[22] Gruninger, M., (1997), Integrated Ontologies for Enterprise Modeling, *Enterprise, integration and modeling technology*, **1**, 368-377.

[23] Matheus, C. J., Kokar, M. M., Baclawski, K., and Letkowski, J. (2003), Constructing RuleML-Based Domain Theories on Top of OWL, Ontologies *Lecture notes in computer science*, **2876**, 81-94.

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

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

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

[27] Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer, R., and Wenke, D. (2002), OntoEdit: Collaborative Ontology Development for the Semantic Web, *Lecture notes in computer science*, **2342**, 221-235.

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

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

[30] Volz, R., Decker, S., and Oberle, D. (2003), Bubo - Implementing OWL in rule-based systems. *WWW2003 May 20-24, Bupest. Hungary.*

[31] Yoshioka, M. and Tomiyama T. (1997), 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, 57-70.

Jaehyun Lee is a Ph.D. candidate in concurrent engineering laboratory, department of industrial engineering, at Korea Advanced Institute of Science and Technology (KAIST). His research interests include product lifecycle management, ontology application to product development knowledge, and semantic web.

Hyowon Suh received the Ph.D. in the department of industrial engineering from West Virginia University, USA, in 1991. He had worked for Korea Institute of Industrial Technology (KITECH) from 1992 to 1995. He is a professor in the department of industrial engineering at KAIST since 1996. His research interests include CE/PDM/CPC/PLM, business process management/workflow management, and ontology/ knowledge-based system.

Jaehyun Lee

Hyowon Suh