Development of Knowledge Code Converter for Design Knowledge Management

Yutaka Nomaguchi*
Department of Computer-controlled Mechanical Systems, Osaka University

Yoshiki Shimomura‡
RACE (Research into Artifact, Center for Engineering), The University of Tokyo

Abstract

This is a report on a new methodology to manage design knowledge by utilizing a knowledge-based CAD and a prototype system named C^3 (Cubic; CAD knowledge Code Capacitor) **, which is being developed using the methodology. C^3 facilitates (i) the automatic generation of a knowledge code for a knowledge-based CAD by processing design documents written in a natural language, such as English or Japanese, and (ii) automatically generation of a design document written in a natural language from the knowledge code. The features of the system facilitate document-based design knowledge management which reduces the designer's load to encode and maintain design knowledge, because it is easier for a designer to treat a natural language description than a coded description.

Keywords: knowledge management, knowledge-based CAD, design document, natural language analysis

1 Introduction

Knowledge management is a crucial issue for manufacturers [Dieng 2000] because the power of knowledge has been recognized as a very important resource for preserving valuable heritage, learning new things, solving problems, creating core competences, and initiating new situations for both individuals and organizations [Liao 2003]. Even in the field of design research, knowledge management has become a hot topic in recent years [Mekhilef and Deshayes 2003]. Some groups have been implementing design knowledge management systems (i.e., [Yoshioka and Shamoto 2003]).

Because of this concern, commercial knowledge-based CAD systems, which are equipped with knowledge bases to store design rules and design constraints, have been released in succession i.e., CATIA (Dassault, Inc.) and Unigraphics (Electronic Data Systems, Inc.,). A knowledge-based CAD is expected to allow a designer to adopt semi-automated design activities by accumulated design rules and design constraints and, thus, reduces the lead-time for the design. However, in order to utilize knowledge-based CAD, it is necessary to encode knowledge before-hand and continue to maintain the encoded knowledge. This results in a heavy workload for designers.

This is a report of a new methodology to manage design knowledge by utilizing a knowledge-based CAD and a prototype system named C^3 (Cubic; CAD knowledge Code Capacitor) based on the methodology. C^3 facilitates (i) the automatic generation of a knowledge code for knowledge-based CAD by processing design documents written in a natural language, such as English or Japanese, and (ii) the automatic generation of a design document written in a natural language from the knowledge code. The features of the prototype system reduce the designer's load for encoding and maintaining the knowledge because it's easier for the designer to treat a natural language description than a coded description.

There are four other sections in this paper. In Section 2, the effects and issues of knowledge-based CAD are discussed based on the study of actual design activity in a car-component manufacturer. The analysis of the study is followed by a proposal of the methodology of document-based knowledge management in Section 3. Section 3 is an identification of the outline of C^3. In Section 4, the implementation of C^3 is explained, and a design session is carried out on C^3 to show the power of C^3. Finally, Section 5 is a summary of the key points.

2 Effect and issues of knowledge-based CAD

A knowledge-based CAD is a brand-new system that is expected to support effective design activity with equipped knowledge bases. Knowledge that can be accumulated in knowledge-based CAD is categorized into the following three types.

A design rule, which describes design operations and their condition.
A design constraint, by which design parameters should be satisfied. When a CAD model does not meet a constraint, the knowledge-based CAD warns and prompts a designer to modify the model.

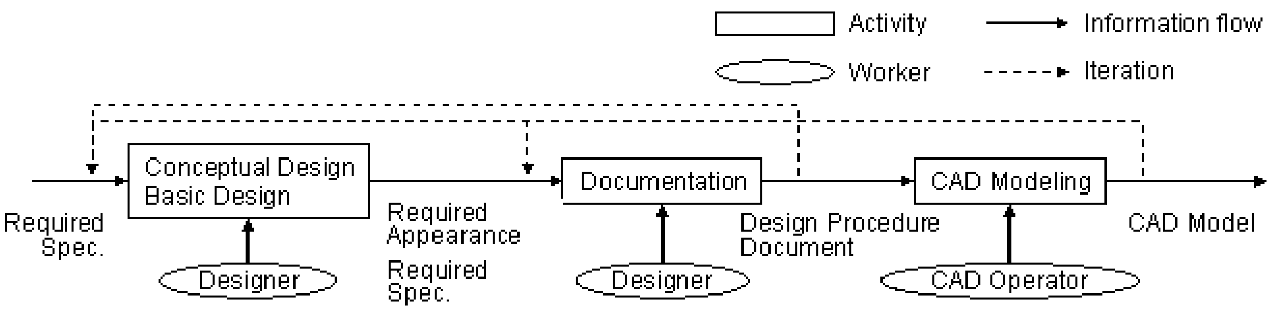

Figure 1. Work-flows of design activities (before introducing knowledge-based CAD)

----
*e-mail: noma@mech.eng.osaka-u.ac.jp
†e-mail: simomura@race.u-tokyo.ac.jp
**Maxis C^3 is the registered trademark of Maxis Inc.,

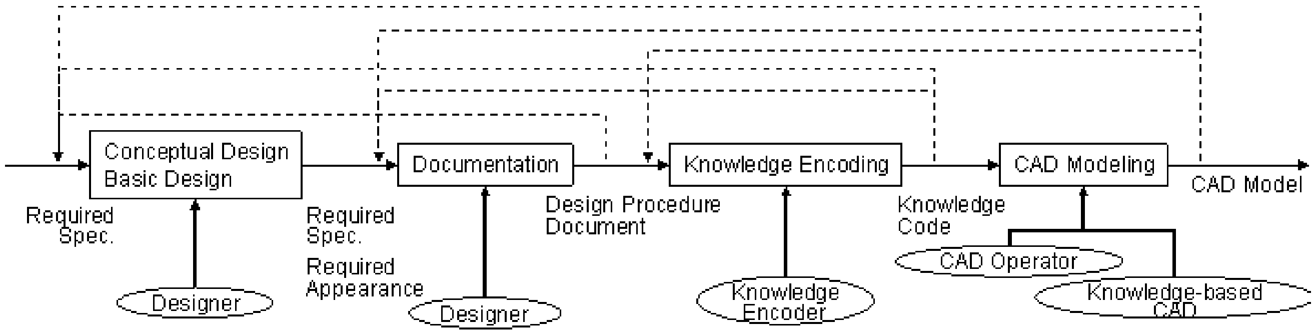

Figure 2. Workflow of design activities (after introducing knowledge-based CAD)

A design procedure, which is a procedure of an operation applied to a CAD model.

In this section, the effects and issues of knowledge-based CAD are clarified by study of an actual case of a car-component manufacturer which introduces knowledge-based CAD into the design division.

Figure 1 depicts the workflows of the design division before the knowledge-based CAD was introduced. At the upstream of the flows, a designer plans a required specifications and required appearance of a product/component. Through this stage, a physical and geometric attributes required for a product/component are decided as a conceptual design solution.

Next, a designer composes a document called a design procedure document (DPD), which describes procedure to determine the attributes to meet the required specification and the design rules/constraints among the attributes. The principal purpose of composing DPD is to instruct a CAD operator, who is not usually an expert of design, to build a geometric model of a product/component without misunderstanding the design rationale. The DPD doest not only makes the communication among a designer and a CAD operator smooth but also promotes the explicit acquisition of design knowledge and its reuse for the efficiency of future design. This is why the manufacturer makes much of composing the DPD, although it is burden for the designer to compose it.

At the final stage of this workflow, the CAD operator interprets the DPD based on the required specification and then builds a CAD model on a CAD system.

The introduction of knowledge-based CAD changes the workflows as shown in Figure 2. The work to encode knowledge is added anew after the work to compose the DPD. The encoded knowledge, which consists of the design rules, constraints, and the definition of design procedures, realizes semi-automatic design by knowledge-based CAD. As a result, it reduces the loads of CAD modeling carried out by CAD operators.

However, the case study also clarifies some suspicions for introductory effect of knowledge-based CAD.

The first one of them is concerning about the new work, “knowledge encoding,” which is carried out by interpreting the DPD. The problem is that a person who performs knowledge encoding should have not only design knowledge for interpreting DPD but also knowledge of the knowledge code of the particular knowledge-based CAD. However, according to the interview to the manufacturer, such a double-role expertise does not exist. This is why either a designer or a CAD operator should take charge of a knowledge encoder, although it is a heavy load for both. The second one is that the format of knowledge codes is not incompatible among CADs. It is necessary to encode knowledge again in case the manufacturer executives decide to change the knowledge-based CAD to the other vender’s one. This means that the introduction of knowledge-based CAD might be risk for the manufacturer from the viewpoint of knowledge reusing. The third one is that .it is difficult to maintain knowledge after knowledge has been finally encoded. In the research field of expert systems, it has been pointed out that knowledge accumulated in a computer should be maintained by capturing new concepts and removing mistakes, contradictions, and redundancy [Roth 1985]. Unless knowledge has been maintained, it results in rigid design activities and a reduction in productivity. This is why a knowledge encoder should have kept on maintaining knowledge. However, it is challenging for humans to read knowledge codes that were formatted to be processed by computers. This must make the load of knowledge maintenance so heavy that the beneficial effect of a knowledge-based CAD may disappear.

The above suspicions for introductory effect of knowledge-based CAD are quantitavily verified by the following study on annual loads on (a) CAD modeling, (b) knowledge encoding and (c) knowledge maintenance before and after introducing a knowledge-based CAD. This study was based on the interviews to the designer and the CAD operators concerning of the design of a part of automobile, of which model-change is conducted three times a year. Table 1 depicts the results. Note that the unit is man-day. According to this study, it is sure that introducing knowledge-based CAD decreased the load of CAD modeling (24 to 2); however, knowledge encoding process takes so many lords (50) that the total load has gone up by three times the load before the introduction. After the second year of the introduction, the load of knowledge encoding decreases because many of the knowledge encoded in the first year can be reused. However, knowledge maintenance still requires considerable load (8). This is why the load reduction by introducing knowledge-based CAD remains a little (24 to 17), although the load of CAD modeling falls in less than one tenth of the load before the introduction.

Table 1 Variation of loads by the introduction of knowledge based CAD

	Loads (Man-Day)			
	CAD Modeling	Knowledge encoding	Knowledge maintenance	Sum.
Before introducing knowledge-based CAD	24	-	-	24
1st year after introducing knowledge-based CAD	2	50	10	62
Afterward	2	9	8	17

The above case study unveils that the key to utilize knowledge-based CAD is knowledge encording and knowledge maintenance. Besides, the knowledge code should be managed in a generic format which is independent from a specific knowledge code format. The approach in this research is document-based knowledge management, whereby the knowledge code is managed by the format of the design procedure document written in a natural language such as Japanese or English, and mutual exchange between the knowledge code and the design procedure document is conducted. This approach required the following three tasks:

1. Converting DPD to a knowledge code;
2. Mutual exchange of knowledge codes among various knowledge-based CAD systems; and
3. Converting a knowledge code to DPD

3 Framework of kowledge code converter

The knowledge code converter C^3 (Cubic: CAD knowledge Code Capacitor) facilitates the above three tasks in order to realize knowledge management by utilizing knowledge-based CAD. This section explains the framework of C^3.

3.1 Intermediate knowledge code

At first, this research introduces a concept of Intermediate Knowledge Code (IKC). IKC is the knowledge code which is able to express the design rules, design constraints and design procedures without depending on specific knowledge code formats or specific operations of each knowledge-based CAD. To introduce such a knowledge code, generic concepts required to describe design knowledge should be clearly divided from concepts specific for knowledge-based CAD. Once such classification could be established, it becomes easier to mutually exchange knowledge codes between diverse knowledge-based CAD by mapping concepts of IKC and CAD-specific knowledge codes.

Making a line passing two points:
```
(create_line
        ((#point1 <Point>)
        (#point2 <Point>)))
```
Offsetting a plane:
```
(create_offsetplane
        ((#object <Plane>)
         (#direction <Direction>)
         (#distance <Value>)))
```

Figure 3 Examples of a generic operation for IKC

Some related works to IKC can be found. STEP (Standard for Exchange of Product Model Data: ISO 10303) is the famous standard format of CAD data, which is independent from specific CADs to express geometry and product information. In the research field of knowledge sharing, KIF (Knowledge Interchange Format) is suggested as the generic format for knowledge description [Genesereth 1992]. As compared to these proceeded successes, IKC is the new trial to describe design knowledge, which includes design procedures, design rules, design constraints, and geometric operations

This research defined more than 100 generic CAD operations, such as "offsetting (parallel displacement) a plane” and “making a line passing two points” as IKCs after consultation with designers who mastered the operations of plural CADs. Figure 3 depicts the example of IKC. An IKC consists of one operation name, i.e., “create_line” and “create_offsetplane” and plural operand names, i.e., “#point1” and “#object.” A word between “<” and “>,” which follows an operand name, is a variable name, which refers description of DPD in the definition of DPD to IKC conversion rule stated below. Note that the IKC defined here may be updated by adding knowledge-based CADs.

3.2 Functions of C^3

By introducing IKC, the three tasks stated in Section 2 is developed to the following four functions:

I. To convert the description of DPD to IKC;
II. To convert IKC to CAD-specific knowledge code, called CKC (CAD Knowledge Code);
III. To convert CKC to IKC;
IV. To convert IKC to the description of DPD;

Table 2 shows the correspondences between the three tasks and the four functions. An “X” in an intersection indicates the correspondence.

3.3 Components of C^3

In order to realize the four functions, this research develops the four components, which correspond to the function, respectively, *DPD to IKC converter*, *IKC to CKC converter*, *CKC to IKC converter* and *IKC to DPD converter*. Figure 4 depicts the architec-

Table 2. Correspondence of processes and tasks of C^3

Function. / Task	I	II	III	IV
1. Converting DPD to a knowledge code	X	X		
2. Mutual exchange of knowledge codes		X	X	
3. Converting a knowledge code to DPD			X	X

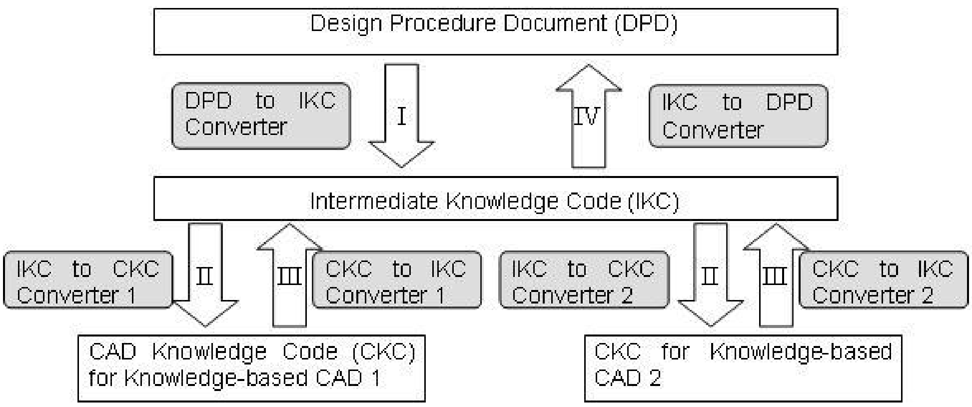

Figure 4. The architecture of C^3

ture of C^3 which consists of the above four components. In Figure 4, an arrow with a Roman number represents function; and a grey rectangular node represents a component. The detail of each component is as follows.

DPD to IKC converter

The DPD to IKC converter is the component that converts natural language descriptions in the DPD into the IKCs. The description in the DPD should be written in the format depicted in Figure 5. Each line in this format corresponds to one description of DPD, which consists of the following three slots:

Line number: a serial number issued for each description;
Reference name: a name used to designate the geometric object created by the description of the line.
Description: text information which describes a design procedure, a design rule or constraint. A word between '<' and '>' means a reference name referring to the geometric object which is created by the other line's description.

No.	Ref. Name	Description
B1	Bottom-surface	Create a plane at the position of <Media-bottom-surface>
B2	Top-surface	Create an offset plane from <Media-top-surface> outside by 5 mm
B3	Left-surface	Create an offset plane from <Media-left-surface> outside by 3 mm
B4	Right-surface	Create an offset plane from <Media-right-surface> outside by 3 mm
B5	Front-surface	Create an offset plane from <Media-front-surface> outside by 5 mm

Figure 5. Design Procedure Document

```
DPD: Create an offset plane
         from <Plane> <Direction> by <Value>mm
IKC: (create_offsetplane
                 ((#object <Plane>)
                 (#direction <Direction>)
                  (#distance <Value>)))
```

Figure 6. Example of DPD to IKC conversion rule

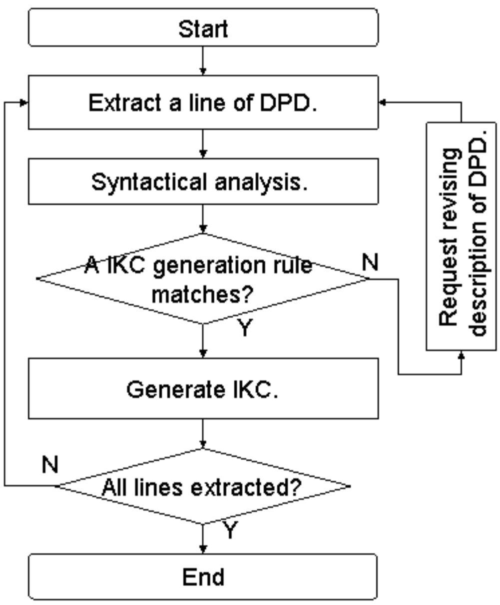

Figure 7. Algorithm of DPD to IKC converter

In order to convert DPD to IKC, the converter uses the DPD to IKC conversion rule, which has a syntactic pattern of a document description of DPD in the condition part and a syntactic pattern of IKC in the conclusion part. For example, the DPD to IKC conversion rule depicted in Figure 6 matches to a DPD description "Create an offset plane from face-A upward by 5 mm;" and the converter generates IKC "(create_offsetplane ((#object face-A) (#direction upward) (#distance 5)))." When no rule matches to DPD description, the DPD to IKC converter decides that the description includes errors, and alerts the user to correct the description of DPD. The algorithm of the DPD to IKC converter is depicted in Figure 7.

IKC to CKC converter

The IKC to CKC converter is the component to convert the IKC to the CKC by the IKC to CKC conversion rule. Because CATIA V5 is used for a prototyping as stated in Section 4, this research developed the IKC to CKC converter according to the knowledge code format of CATIA V5. Figure 8 depicts an example of an IKC to CKC conversion rule for CATIA V5. A word beginning with "?" in CKC is an internal variable whose identical name is automatically assigned by CATIA. The IKC to CKC conversion rule is specific for a knowledge-based CAD, although the converting mechanism of the IKC to CKC converter is generic.

The IKC to CKC conversion rule has the syntactic pattern of the IKC in the condition part and the syntactic pattern of the CKC in the conclusion part. The IKC to CKC converter picks up each IKC, and search the rule which matches to it. For example, IKC " (create_offsetplane ((#object face-A) (#direction upward) (#distance 5)))" matches to the conversion rule depicted in Figure 8, and it is finally converted to CKC depicted in Figure 9. Because the converter manages the relationships between variable names and reference names in IKC ("face-A" and "upward") and their internal description in CKC ("A1" and "False"), the former can be automatically converted to the later.

CKC to IKC converter

The CKC to IKC converter is the component that converts the CKC into the IKC by the CKC to IKC conversion rule. This conversion rule has the syntactic pattern of CKC in its condition part and the syntactic pattern of the IKC in its conclusion part (see Figure 10). The CKC to IKC conversion rule is specific for a knowledge-based CAD although the converting mechanism of

```
IKC：
  (create_offsetplane
        ((#object <Plane>)
        (#direction <Direction>)
        (#distance <Value>)))
CKC：
  Dim ?parameter As Parameters
  Set ?parameter = &part.Item(<Plane>)
  Dim ?offsetplane As HybridShapePlaneOffset
  Set ?offsetplane =
  &factory.AddNewPlaneOffset(?parameter, <Value>,
  <Direction>)
```

Figure 8. IKC to CKC conversion rule

```
Dim hybridShapeSurfaceExplicit1 As Parameters
Set hybridShapeSurfaceExplicit1 = part1.Item("A1")
Dim hybridShapePlaneOffset1 As HybridShapePla-
neOffset
Set hybridShapePlaneOffset1 = fac-
tory1.AddNewPlaneOffset(hybridShapeSurfaceExplicit
1, 5, False)
```

Figure 9. Example of CKC

CKC:

Parallel.Name = <name>, Mode = <mode>,
Type = <type>, Curve = <curve>,
Support = <support>, Offset.Mode = <offsetmode>,
Length = <length>, Bothside = <bothside>,
Direction = <direction>

IKC:

((crate_trim_line)
((#name <name>) (#object1 <curve>)
(#object2 <support>)
(#direction <direction>) (#distance <length>)))

Figure 10. Example of the CKC to IKC conversion rule

IKC:

((create_trim_line)
((#name <name>)
(#object1 <curve>)
(#object2 <support>)
(#direction <direction>)
(#distance <length>)))

DPD:

Ref. Name	Description
<name>	Create a line by offsetting <curve> with the support of <support> to the direction of <direction> by <length> mm.

Figure 11. Example of IKC to DPD conversion rule

the CKC to IKC converter is generic.

By using the IKC to CKC converter and the CKC to IKC converter, C^3 realizes the mutual exchange of CKC among plural knowledge-based CADs.

IKC to DPD converter

The IKC to DPD converter is the component that converts the IKC into the description of the DPD by the IKC to DPD conversion rule. This conversion rule has the syntactic pattern of the IKC in its condition part and the syntactic pattern of the DPD description in its conclusion part (see Figure 11).

The IKC to DPD converter and the CKC to IKC converter contribute to DPD generation.

4 Prototyping

4.1 Implementation

The authors have been developing a prototype system of C^3 which can generate a knowledge code from a document description as well as a document description of DPD from a knowledge code. C^3 is implemented by C++ programming language on Windows2000. CATIA V5 of Dassault Inc, is used as a sample knowledge-based CAD for this prototype system. The prototype system is composed of the DPD to IKC converter, the IKC to CKC converter for CATIA V5, the CKC to IKC converter for CAIA V5, and the IKC to DPD converter. The current version of the prototype system support conversion of DPD written in Japanese and English to CKC, although conversion of CKC to DPD only supports Japanese.

4.2 Design session

To test and validate the methodology, the authors prepared a DPD of a media case as an example, and conducted the generation of the CKC for the CATIA V5 by the prototype system. Fig-

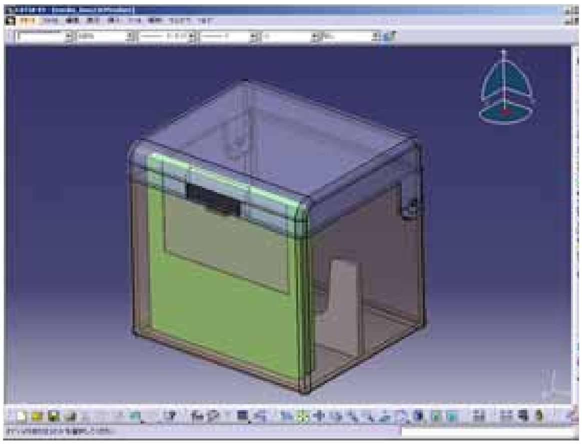

Figure 12. Media case

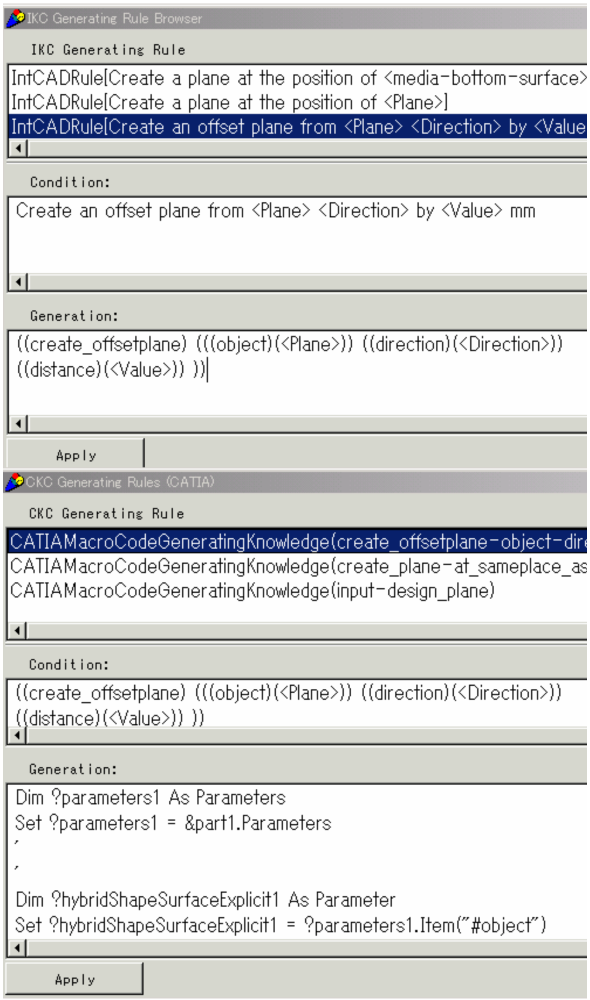

Figure 13. DPD to IKC conversion rule browser (top) and IKC to CKC conversion rule browser (bottom)

Table 3. Variation of loads by the introduction of knowledge based CAD with C^3

	Loads (Man-Day)			
	CAD Modeling	Knowledge encoding	Knowledge maintenance	Sum.
Before introducing knowledge-based CAD with C^3	24	-	-	24
1st year after introducing knowledge-based CAD with C^3	2	25	5	32
Afterward	2	6	3	11

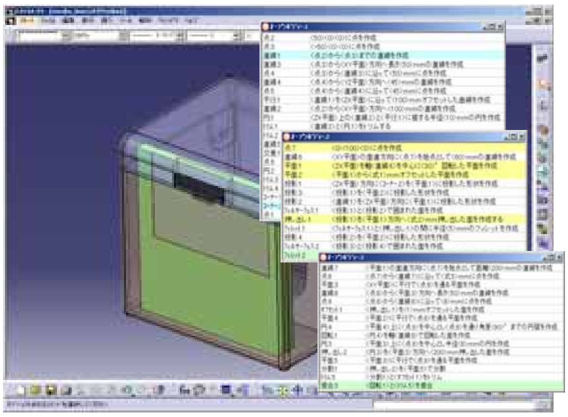

Figure 14. Regeneration of DPD (written in Japanese) from the CAD model

ure 12 depicts the CAD model of the media case. Figure 5 depicts a part of the DPD of the media case. The prototype system could generate a CKC for CATIA V5, by which the CAD model depicted in Figure 12 could be automatically generated. Figure 13 depicts the DPD to ICK / IKC to CKC conversion rule browser, by which the user can define and edit conversion rules. The prototype system could also regenerate the DPD of the media case from the CAD model (see Figure 14. Note that generated DPD is written in Japanese).

4.3 Validation

According to the above experiment, it is confirmed that C^3 can generate a knowledge code and regenerate DPD, by which C^3 facilitates the encoding and maintaining knowledge for knowledge-based CAD on the basis of documents written in a natural language. From the viewpoint of knowledge management, it is easier for humans to manage a natural language description than to directly manage a digital description of knowledge codes. The ability of C^3 might serve the utilization of knowledge-based CAD.

To support the above discussion about the effect of C^3, the authors conducted the study on annual loads on (a) CAD modeling, (b) knowledge encoding and (c) knowledge maintenance before and after introducing a knowledge-based CAD with C^3 as the comparison experiment to the study stated in Section 2. Table 3 depicts the results. Because enough data for long-term analysis have not yet accumulated, the numbers of “Afterward” row in the study of this section includes prediction.

At the first year of introducing knowledge-based CAD with C^3, loads of CAD modeling drastically decreases as shown in Section 2; but the total load increases by 8 man-days because the loads of knowledge encoding and knowledge maintenance arise. However, the amount increased is smaller than that of the case in which knowledge-based CAD is solely introduced depicted in Table 1. This is because C^3 facilitates to encode/maintain knowledge at the format of natural language description in DPD so that the designers’ workload decreases.

Afterward the second year, the total load will decline to a half of that before introducing knowledge-based CAD with C^3, because the load of knowledge encoding decreases to one fifth of that of the first year. The decreased amount of loads balances introductory cost of knowledge-based CAD.

According to the above study, C^3 takes a crucial role to utilize knowledge-based CAD and to raise its power to manage design knowledge.

5 Conclusion

This paper reports the framework of the knowledge code converter C^3 to utilize knowledge-based CAD towards design knowledge management, and validates its introductory effect by conducting the case study. The core ideas of this research are under patent application in Japan (Serial No. 2002- 338832, 2003-305667) and the United States of America (Serial No. 10/716,557).

Acknowledgements

We would like to acknowledge Mr. Ichiro Koike and Mr. Noriyasu Goto of Maxis Inc, who co-operated with the study of load variation by introducing a knowledge-based CAD in section 2 and 4.1, and the design session in section 4.2. C^3 was developed under the CATIA V5 CAA (Component Application Architecture) Adopter license, which Maxis Inc, acquired from Dassault Inc.

References

DIENG, R. 2000. Knowledge Management and the Internet, IEEE Intelligent Systems, Vol. 15, No. 3, (2000), pp. 14-17.

GENESERETH,M.R. 1992. Knowledge Interchange Format, In *Proceedings of the Conference of the Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann Publishers, pp. 599-600.

LIAO, S. 2003. Knowledge Management Technologies and Applications - Literature Review from 1995 to 2002, Expert Systems with Applications, Vol. 25, pp. 155-164.

MEKHILEF, M. AND DESHAYES, P. 2003. Knowledge Management: A Concept Review, In *Proceedings of the ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, DAC-48745 (in CD-ROM).

ROTH, F. H., et al. 1985. Building Expert Systems, Addison-Wesley Publishing Company, Inc.,

YOSHIOKA, M. AND SHAMOTO, Y. 2003. Knowledge Management System for Problem Solving - Integration of Document Information and Formalized Knowledge -, In *Proceedings of the ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, CIE-48217 (in CD-ROM).