Proceedings of the TMCE 2004, April 13-17, 2004, Lausanne, Switzerland, Horváth & Xirouchakis (eds.)
© 2004 Millpress, Rotterdam, ISBN 90 5966 018 8

MANAGEMENT OF DESIGN KNOWLEDGE FOR KNOWLEDGE-BASED CAD

Yutaka Nomaguchi
Department of Computer-Controlled Mechanical Systems
Osaka University
Japan
noma@mech.eng.osaka-u.ac.jp

Yoshiki Shimomura
Research into Artifacts, Center for Engineering (RACE)
The University of Tokyo
Japan
simomura@race.u-tokyo.ac.jp

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 our 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.

KEY WORDS

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 *et al.* 2003). Some groups have been implementing design knowledge management systems (i.e., Yoshioka *et al.* 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 allows designers 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 beforehand 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 our 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 to encode and maintain the knowledge because it's easier for a 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 our 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.

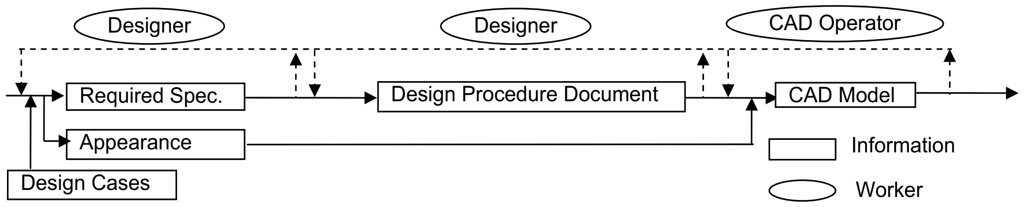


Figure 1 Information flow of design activities (before introducing knowledge-based CAD)

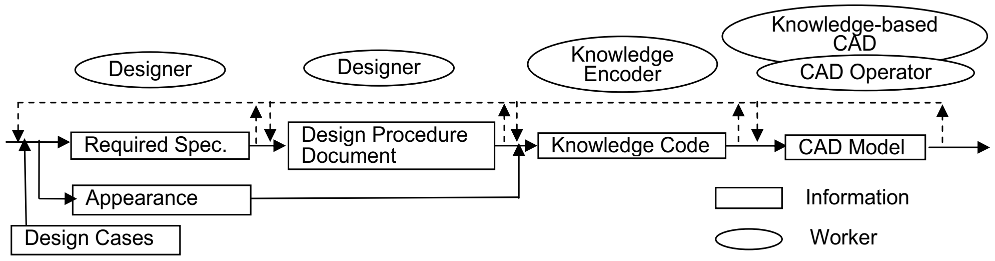


Figure 2 Information flow of design activities (after introducing knowledge-based CAD)

2. EFFECTS 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, with 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.
- A design procedure, which is a procedure of a design operation applied to a CAD model. The design procedure can be reused in a future design.

In this section, the effects and issues of knowledge-based CAD are described using a case in which a car-component manufacturer introduces knowledge-based CAD into the design division.

Figure 1 depicts the workflow and information-flow in the design division before the knowledge-based CAD was introduced. Rectangular nodes represent information, and oval nodes represent workers who process the information. These flows generally contain backward processes, which are depicted by dashed lines.

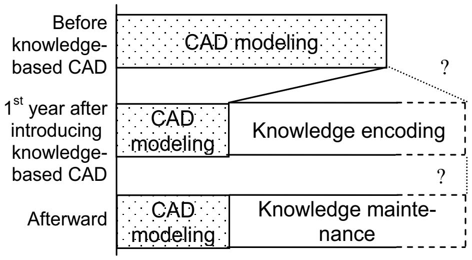


Figure 3 Variation of loads by the introduction of knowledge-based CAD

At the upstream stage of the flow, a designer plans the required specifications and required appearance of a product/component by referring former design cases. Through this stage, the physical and geometric attributes required for a product/component are decided as an alternative solution.

In the next stage, the designer composes a document called a design procedure document (DPD), which describes how to determine the attributes of a required specification determined in a previous stage and design rules/constraints among attributes. The principal aim 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. Although it is burden for a designer to compose a DPD, the task plays a crucial role for the manufacturer because design knowledge is explicitly acquired by composing a DPD. A DPD not only makes the communication among designers and CAD operators smooth but also promotes the reuse of design knowledge for the efficiency of future design.

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

The introduction of knowledge-based CAD changes the flows, as shown in Figure 2. Although the stage of composing a DPD still remains, the stage of encoding knowledge is added anew. 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. Figure 3 depicts an estimated variation of loads of designers through each year by introducing a knowledge-based CAD. At the first year of introducing a knowledge-based CAD, the load of CAD modeling is reduced by the use of a knowledge-based CAD. However, designers confront the load of the new stage *encoding knowledge*. Although the loads required by this stage cannot be estimated at this time, it might overcome the reduction of the load of CAD modeling. The details of this stage are discussed in the following paragraphs.

A knowledge encoder carries out this stage by interpreting the DPD. The problem is that a knowledge encoder should have not only design knowledge for interpreting but also knowledge of a knowledge code of a particular knowledge-based CAD. However, our investigation of the manufacturer clarified that the expertise to take charge of a knowledge encoder 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.

Even if this stage ware carried out, another problem would occur. Once knowledge has been encoded, it is difficult to maintain. In the research field of expert systems, knowledge accumulated in a computer should be maintained by capturing new concepts and removing mistakes, contradictions, and redundancy (Roth 1985). Unless knowledge is retained, it results in rigid design activities and a reduction in productivity. This is why a knowledge encoder should be used to retain knowledge. However, it is challenging for humans to decipher knowledge codes that were designed to be read by computers. This makes the load of knowledge maintenance so heavy that the beneficial effect of a knowledge-based CAD may disappear (see the Afterward in Figure 3).

In addition, the grammar of knowledge codes is not incompatible among CADs. It is necessary to encode

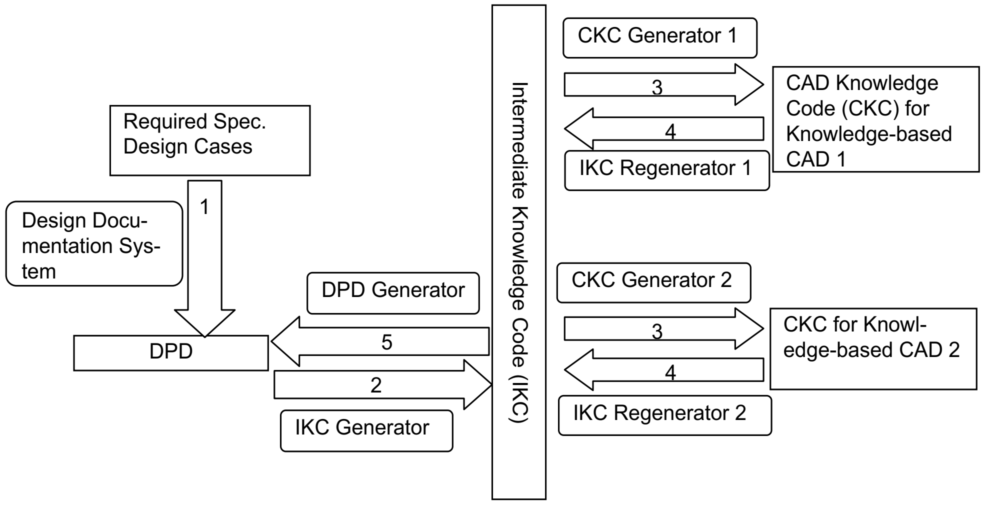

Required Spec.
Design Cases
1
Design Documentation System
DPD Generator
DPD
5
2
IKC Generator
Intermediate Knowledge Code (IKC)
CKC Generator 1
3
4
IKC Regenerator 1
CAD Knowledge Code (CKC) for Knowledge-based CAD 1
CKC Generator 2
3
4
IKC Regenerator 2
CKC for Knowledge-based CAD 2

Figure 4 the architecture of C^3

knowledge again when changing from one knowledge-based CAD to another.

These problems are vital when a manufacturer considers introducing a knowledge-based CAD. To solve these problems, a methodology should be established to encode and maintain design knowledge with a small load. The approach here is to use document-based knowledge management. The methodology covers the following tasks to be supported by using a computer system.

1. Knowledge acquisition by composing DPD;
2. Generation of a knowledge code based on DPD;
3. Maintenance of design knowledge on DPD; and
4. Mutual exchange of knowledge codes among various knowledge-based CAD systems

3. FRAMEWORK OF DOCUMENT-BASED KNOWLEDGE MANAGEMENT

In this section, the framework of a document-based knowledge management for a knowledge-based CAD system is proposed It is called C^3 (Cubic: CAD knowledge Code Capacitor. The architecture of C^3 is depicted in Figure 4.

3.1. Outline of C^3

C^3 supports five processes, (I) composition of a DPD; (II) conversion of a DPD to an intermediate knowledge code; (III) conversion of an intermediate knowledge code to various CAD knowledge codes; (IV) conversion of a CAD knowledge code to an intermediate knowledge code; and (V) conversion of an intermediate knowledge code to a DPD. The user can carry out the tasks stated in the previous section by combinations of these processes. The correspondence of processes and tasks is shown in Table 1. An "X" in an intersection indicates the correspondence between a process and a task, and the process number corresponds to the Roman numeral followed by the process description in this paragraph.

Table 1 Correspondence of processes and tasks of C^3

Process No. / Task	I	II	III	IV	V
1. Knowledge Acquisition	X				
2. Generation of a knowledge code		X	X		
3. Maintenance of design knowledge		X	X	X	X
4. Mutual exchange of knowledge code			X	X	

During the first process (expressed as "1" in Figure 4), C^3 helps a designer to compose a DPD in order to explicitly acquire design knowledge. This process can be supported by a design documentation management system, which is been developed (Nomaguchi 2002). In this paper, the detail are not discussed.

The process to convert a DPD into various CAD knowledge codes consists of the following three steps, (i) generation of an Intermediate Knowledge Code (IKC) (expressed as "2" in Figure 4), (ii) generation of a CAD Knowledge Code (CKC) (expressed as "3" in Figure 4), and (iii) regeneration of an IKC from a CKC (expressed as "4" in Figure 4). During process "2," the IKC generator converts a description of a DPD into an IKC, which is described in an independent format against each knowledge-based CAD. Then, during process "3," an IKC is converted into a knowledge code of each knowledge-based CAD by the CKC generator. A CKC is defined for each knowledge-base CAD; therefore, different CKC generators should be respectively provided for each CAD. Furthermore, we suggest the mutual conversion between different CKCs through the IKC. This is realized by the IKC regenerator, which reconverts each CKC into an IKC again.

For easier maintenance of the knowledge code, an IKC is converted into a DPD by the DPD generator (expressed as "5" in Figure 4).

3.2. Phases of C^3

The phases of the C^3 framework stated above are arranged according to a situation of a design activity. C^3 can support the following two phases:

- Phase-1
  This is for the time when a manufacturer introduces C^3 for the first time. Users of C^3 should carry out the entire procedure of the framework, namely, process "1" to process "5."
- Phase-2
  This is for a time after the introduction of C^3. Phase-2 is further classified into the following three sub-phases.
  1. When the required specification is changed, the users should start from process "1" to compose new design procedure documents with the revised required specifications. Then, the users generate an IKC and a CKC again from new design

procedure documents using the IKC/CKC generators.

2. When the design procedure documents are revised, the users should start from process “2” to generate an IKC and a CKC from the revised design procedure documents using the IKC/CKC generators.
3. When a knowledge-based CAD is added anew, the users should start from process “3” to generate a CKC for the added knowledge-based CAD

3.3. Components of C^3

This subsection describes the details of each component in C^3.

Intermediate Knowledge Code

The intermediate knowledge code (IKC) should be able to express the design rules, design constraints and design procedures without depending on knowledge code formats or specific operations of each knowledge-based CAD. The advantages provided by defining an IKC are: (i) we can clearly define generic concepts, as this is necessary to represent design knowledge, and (ii) in this why, we can easily add a new knowledge-based CAD into this framework by defining the relationships between generic concepts and CAD-specific concepts. Concerning this character of the IKC, we collected 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 5 depicts the example of IKC we defined. The part expressed between “<” and “>” in the figure is a variable name. The IKC we defined here may be updated by adding knowledge-based CADs.

IKC Generator

The IKC generator is the component that converts natural language descriptions in a DPD into IKCs. The IKC generation process consists of the following two steps: (i) a syntactic analysis of the description of a DPD, and (ii) the generation of an IKC, which is matched with the syntactic structure analyzed in step (i) by applying the IKC generation rule. The IKC generation rule has a syntactic pattern of a document description of DPD in the condition part and a syntactic pattern of IKC in the conclusion part. See the upper side of Figure 6. The IKC generator retrieves a rule which matches a syntactic pattern of a document description, unifies variables, and generates IKC. An example of a generated IKC is shown in the lower side of Figure 6.

CAD Knowledge Code (CKC) Generator

The CAD knowledge code (CKC) is a knowledge code which is specific for a knowledge-based CAD. The CKC generator is a component to convert an IKD to a CKC by the CKC generation rule. Because we use CATIA V5 for a prototyping, as state in Section 4, we developed the CKC generator according to the knowledge code format of CATIA V5. The upper side of Figure 7 depicts an example of a CKC generation rule for CATIA V5. The CKC generation rule is specific for a knowledge-based CAD, although the converting mechanism of the CKC generator is generic.

The CKC generation rule has the syntactic pattern of an IKC in the condition part and a syntactic pattern

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

Figure 5 Examples of a generic operation for IKC

Example of IKC generation rule

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

Matched description

Create an offset plane from face-A upward by 5 mm

Generated IKC
```
(#create_offsetplane
        ((#object face-A)
        (#direction upward)
        (#distance 5)))
```

Figure 6 Example of an IKC generation rule and example of matching

of CKC in the conclusion part. The upper side of Figure 7 shows an example of the CKC generation rule.

```
Example of CKC generation rule
  Condition□
    (#create_offsetplane
         ((#object <Plane>)
         (#direction <Direction>)
         (#distance <Value>)))
  Conclusion□
    Dim ?parameter As Parameters
    Set ?parameter = &part.Item("#object")
    Dim ?offsetplane As HybridShapePlaneOffset
    Set ?offsetplane =
    &factory.AddNewPlaneOffset(?parameter,
    #distance, #direction)
```

```
IKC matched with the CKC
    (#create_offsetplane
         ((#object 'top-surface')
         (#direction Upward)
         (#distance 5)))
Generated CKC
    Dim hybridShapeSurfaceExplicit1 As Parame-
    ters
    Set hybridShapeSurfaceExplicit1 =
    part1.Item("A1")
    Dim hybridShapePlaneOffset1 As Hybrid-
    ShapePlaneOffset
    Set hybridShapePlaneOffset1 = fac-
    tory1.AddNewPlaneOffset(hybridShapeSurfac
    eExplicit1, 5, False)
```

Figure 7 CKC and CKC generation rule

```
Condition:
  Parallel.Name = <name>,
   Mode = <mode>,
  Type = <type>,
  Curve = <curve>,
  Support = <support>,
  Offset.Mode = <offsetmode>,
  Length = <length>,
  Bothside = <bothside>,
  Direction = <direction>
Conclusion:
  ((crate_trim_line)
       (((name)(<name>))
       ((object1)(<curve>))
       ((object2)(<support>))
       ((direction)(<direction>))
       ((distance)(<length>))))
```

Figure 8 Example of the IKC regeneration rule

```
Condition:
  ((create_trim_line)
       (((name)(<name>))
       ((object1)(<curve>))
       ((object2)(<support>))
       ((direction)(<direction>))
       ((distance)(<length>))))
Conclusion:
  <name>        Create a line by offsetting <curve>
  with the support of <support> to the direction of <di-
  rection> by <length> mm
```

Figure 9 Example of DPD generation rule

IKC Regenerator

The IKC regenerator is a component that converts a CKC into an IKC by the IKC regeneration rule. The IKC regenerator has a syntactic pattern of CKC in its condition part and a syntactic pattern of the IKC in its conclusion part (see Figure 8). The IKC regeneration rule is specific for a knowledge-based CAD although the converting mechanism of the IKC regenerator is generic.

By using the CKC generator and the IKC regenerator, C^3 realizes the mutual exchange of CKC among plural knowledge-based CADs. In addition, the IKC regenerator contributes to DPD generation with the DPD generator.

DPD Generator

The DPD generator is a component that converts an IKC into a DPD by the DPD generation rule. The DPD generation rule has a syntactic pattern of an IKC in its condition part and a syntactic pattern of a DPD description in its conclusion part (see Figure 9).

4. PROTOTYPING

4.1. Implementation

We have been developing a prototype system of C^3 that 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 using the C++ programming language on Windows2000. We used CATIA V5 of Dassault Inc. as a sample knowledge-based CAD for this prototype system. The prototype system is composed of an IKC generator, a CKC generator for CATIA V5, an IKC regenerator for CAIA V5, and a DPD generator.

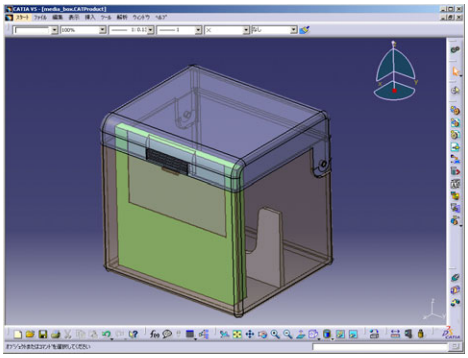
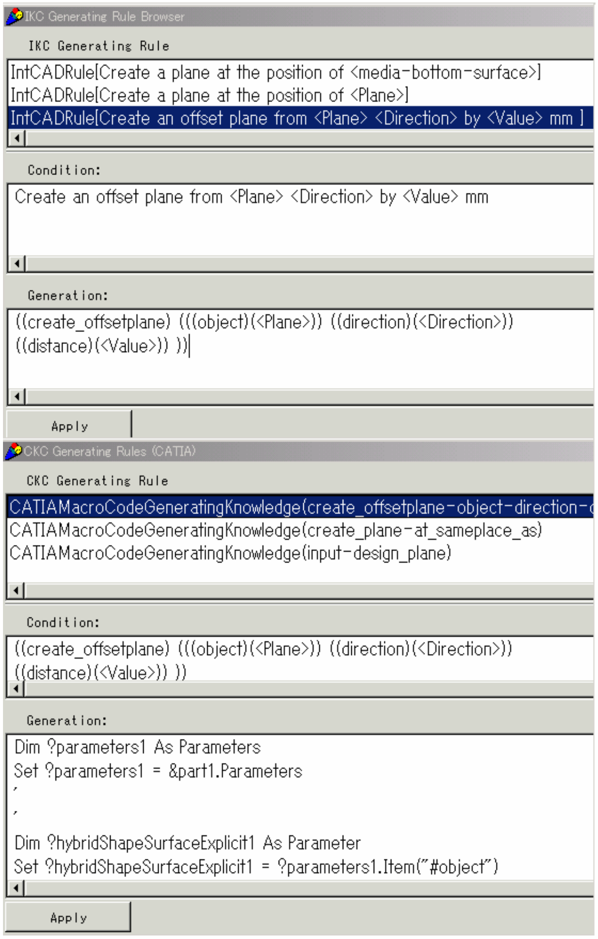


Figure 10 Media case

	Name	Design procedure
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 11 A part of the design procedures document used for design procedures



```
IntCADRule[Create a plane at the position of <media-bottom-surface>]
IntCADRule[Create a plane at the position of <Plane>]
IntCADRule[Create an offset plane from <Plane> <Direction> by <Value> mm ]
```

```
Create an offset plane from <Plane> <Direction> by <Value> mm
```

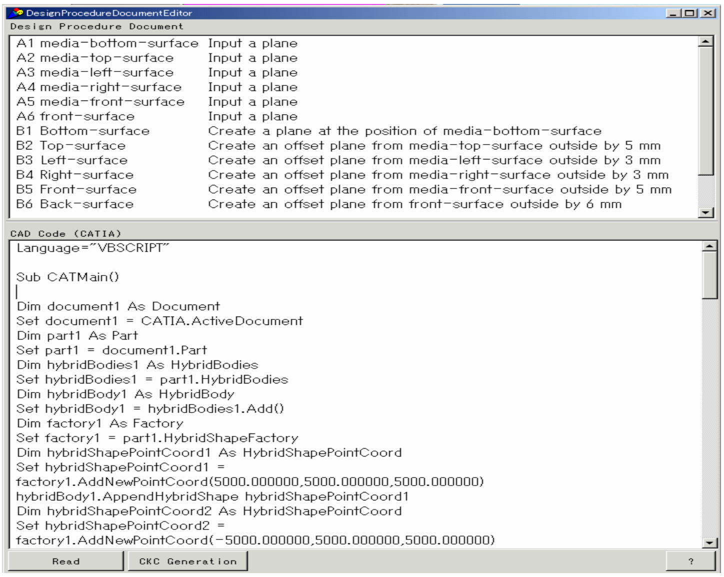
```
((create_offsetplane) (((object)(<Plane>)) ((direction)(<Direction>))
((distance)(<Value>)) ))
```

```
CATIAMacroCodeGeneratingKnowledge(create_offsetplane-object-direction-d
CATIAMacroCodeGeneratingKnowledge(create_plane-at_sameplace_as)
CATIAMacroCodeGeneratingKnowledge(input-design_plane)
```

```
((create_offsetplane) (((object)(<Plane>)) ((direction)(<Direction>))
((distance)(<Value>)) ))
```

```
Dim ?parameters1 As Parameters
Set ?parameters1 = &part1.Parameters
'
'
Dim ?hybridShapeSurfaceExplicit1 As Parameter
Set ?hybridShapeSurfaceExplicit1 = ?parameters1.Item("#object")
```

Figure 12 IKC generation rule browser (top) and CKC generation rule browser (bottom)



DesignProcedureDocumentEditor

Design Procedure Document

A1	media-bottom-surface	Input a plane
A2	media-top-surface	Input a plane
A3	media-left-surface	Input a plane
A4	media-right-surface	Input a plane
A5	media-front-surface	Input a plane
A6	front-surface	Input a plane
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
B6	Back-surface	Create an offset plane from front-surface outside by 6 mm

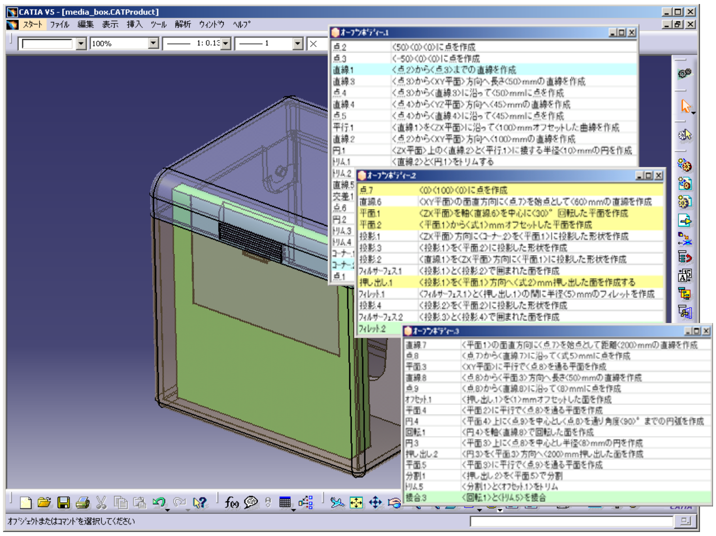
CAD Code (CATIA)

```
Language="VBSCRIPT"

Sub CATMain()

Dim document1 As Document
Set document1 = CATIA.ActiveDocument
Dim part1 As Part
Set part1 = document1.Part
Dim hybridBodies1 As HybridBodies
Set hybridBodies1 = part1.HybridBodies
Dim hybridBody1 As HybridBody
Set hybridBody1 = hybridBodies1.Add()
Dim factory1 As Factory
Set factory1 = part1.HybridShapeFactory
Dim hybridShapePointCoord1 As HybridShapePointCoord
Set hybridShapePointCoord1 =
factory1.AddNewPointCoord(5000.000000,5000.000000,5000.000000)
hybridBody1.AppendHybridShape hybridShapePointCoord1
Dim hybridShapePointCoord2 As HybridShapePointCoord
Set hybridShapePointCoord2 =
factory1.AddNewPointCoord(-5000.000000,5000.000000,5000.000000)
```

Figure 13 Processing DPD by C^3



CATIA V5 - [media_box.CATProduct]
スタート ファイル 編集 表示 挿入 ツール 解析 ウィンドウ ヘルプ
100%
オブジェクトまたはコマンドを選択してください

オープンボディー.1

点.2	<50><0><0>に点を作成
点.3	<-50><0><0>に点を作成
直線.1	<点.2>から<点.3>までの直線を作成
直線.3	<点.3>から<XY平面>方向へ長さ<50>mmの直線を作成
点.4	<点.3>から<直線.3>に沿って<50>mmに点を作成
直線.4	<点.4>から<YZ平面>方向へ<45>mmの直線を作成
点.5	<点.4>から<直線.4>に沿って<45>mmに点を作成
平行.1	<直線.1>を<ZX平面>に沿って<100>mmオフセットした曲線を作成
直線.2	<点.2>から<XY平面>方向へ<100>mmの直線を作成
円.1	<ZX平面>上の<直線.2>と<平行.1>に接する半径<10>mmの円を作成
トリム.1	<直線.2>と<円.1>をトリムする
トリム.2	
直線.5	
交差.1	
点.6	
円.2	
トリム.3	
トリム.4	
コーナー.1	
コーナー.2	
点.1	

オープンボディー.2

点.7	<0><100><0>に点を作成
直線.6	<XY平面>の面直方向に<点.7>を始点として<60>mmの直線を作成
平面.1	<ZX平面>を軸<直線.6>を中心に<30>° 回転した平面を作成
平面.2	<平面.1>から<式.1>mmオフセットした平面を作成
投影.1	<ZX平面>方向に<コーナー.2>を<平面.1>に投影した形状を作成
投影.3	<投影.1>を<平面.2>に投影した形状を作成
投影.2	<直線.1>を<ZX平面>方向に<平面.1>に投影した形状を作成
フィルサーフェス.1	<投影.1>と<投影.2>で囲まれた面を作成
押し出し.1	<投影.1>を<平面.1>方向へ<式.2>mm押し出した面を作成する
フィレット.1	<フィルサーフェス.1>と<押し出し.1>の間に半径<5>mmのフィレットを作成
投影.4	<投影.2>を<平面.2>に投影した形状を作成
フィルサーフェス.2	<投影.3>と<投影.4>で囲まれた面を作成
フィレット.2	

オープンボディー.3

直線.7	<平面.1>の面直方向に<点.7>を始点として距離<200>mmの直線を作成
点.8	<点.7>から<直線.7>に沿って<式.5>mmに点を作成
平面.3	<XY平面>に平行で<点.8>を通る平面を作成
直線.8	<点.8>から<平面.3>方向へ長さ<50>mmの直線を作成
点.9	<点.8>から<直線.8>に沿って<8>mmに点を作成
オフセット.1	<押し出し.1>を<1>mmオフセットした面を作成
平面.4	<平面.2>に平行で<点.8>を通る平面を作成
円.4	<平面.4>上に<点.9>を中心とし<点.8>を通り角度<90>° までの円弧を作成
回転.1	<円.4>を軸<直線.8>で回転した面を作成
円.3	<平面.3>上に<点.8>を中心とし半径<8>mmの円を作成
押し出し.2	<円.3>を<平面.3>方向へ<200>mm押し出した面を作成
平面.5	<平面.3>に平行で<点.9>を通る平面を作成
分割.1	<押し出し.2>を<平面.5>で分割
トリム.5	<分割.1>と<オフセット.1>をトリム
接合.3	<回転.1>と<トリム.5>を接合

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

4.2. Design Session

To test and validate our methodology, we prepared a DPD of a media case as an example and tested the generation of the CKC for the CATIA V5 by the prototype system. Figure 10 depicts a CAD model of a media case constructed by processing a DPD on C^3. Figure 11 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 10 could be automatically generated. Figure 12 depicts the ICK/CKC generation rule browser, by which the user can define and edit IKC/CKC generation rules. Figure 13 depicts the generation of CKC on the prototype system. The prototype system could also regenerate the DPD of the media case from the CAD model (see Figure 14).

We determined that the ability of our framework to generate a knowledge code and regenerate DPD was validated. From the viewpoint of knowledge management, we consider that our framework facilitates the acquisition and maintenance of knowledge for knowledge-based CAD on the basis of documents written in a natural language. This method is easier for humans than the maintenance of a direct knowledge code.

5. SUMMARY

We proposed a document-based methodology to manage design knowledge by utilizing a knowledge-based CAD and C^3, which is a framework to implement our methodology. We developed a prototype system of C^3 that generates encoded knowledge for knowledge-based CAD automatically as a result of processing a design document written in a natural language. Our future works include a quantitative validation of how much our framework could reduce the designer's load to encode and maintain knowledge.

ACKNOWLEDGMENTS

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 the design session in Section 4.

REFERENCES

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

Dassault, Inc., CATIA homepage, http://www.catia.com/.

Electronic Data Systems, Inc., Unigraphics homepage, http://www.eds.com/products/plm/unigraphics/.

Nomaguchi, Y. and Tomiyama, T., (2002), "Design Knowledge Management based on the Model of Synthesis," Proceedings of The Fifth IFIP WG5.2 Workshop on Knowledge Intensive CAD, Malta, pp. 62-81.

Roth, F. H., Waterman, D. A. and Lenat, D. B., (1985), "Building Expert Systems," Addison-Wesley Publishing Company, Inc.

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," Proceedings of the ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Chicago, DAC-48745 (in CD-ROM).

Yoshioka, M. and Shamoto, Y., (2003), "Knowledge Management System for Problem Solving - Integration of Document Information and Formalized Knowledge - ," Proceedings of the ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Chicago, CIE-48217 (in CD-ROM).