Zeitschrift: IABSE reports = Rapports AIPC = IVBH Berichte

Band: 68 (1993)

Artikel: Developing engineering knowledge bases

Autor: Yang, Xiaofeng / Sheng, Kanghen

DOI: https://doi.org/10.5169/seals-51874

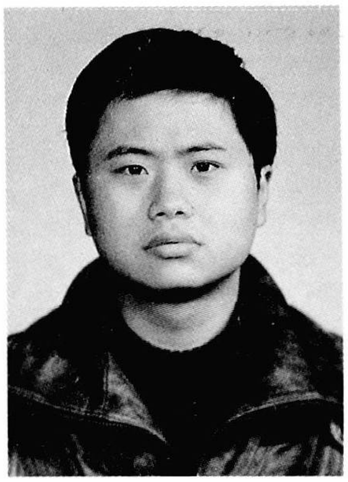
Developing Engineering Knowledge Bases

Evolution des bases de données en génie civil

Entwicklung wissensbasierter Datenbanken im Ingenieurwesen

Xiaofeng YANG
Software Eng.
Shanghai Maritime Univ.
Shanghai, China

Kanghen SHEN
President, Prof.
Shanghai Maritime Univ.
Shanghai, China



Yang Xiaofeng, born 1966. got his MS in Computer Science at Shanghai Maritime Univ. His research work is focused on object oriented methodology, knowledge-based systems and Programming Environment

SUMMARY
Engineering standards are widely used in the process of engineering design and manufacture. They are the records of collective and mature knowledge of the profession. The article reports on various software packages developped for application of such knowledge-based systems.

RÉSUMÉ
Les normes du génie civil font l'objet d'un usage très large dans le projet et la construction de structures. Elles reflètent l'ensemble collectif des connaissances de la profession. Le présent article décrit différents logiciels développés pour l'application de tels systèmes à bases de connaissances.

ZUSAMMENFASSUNG
Die Baunormen werden im Bauprozess vom Entwurf bis zur Ausführung vielseitig verwendet. Sie wiederspiegeln das kollektive Wissen des Berufs. Im Artikel werden verschiedene Datenbankprogramme beschrieben, welche entwickelt worden sind, um die Anwendung dieser wissensbasierten Systeme zu erleichtern und damit ihren Wirkungsgrad zu erhöhen.

1. INTRODUCTION

1.1 Engineering codes and KECOMS

Engineering standards or codes are widely used in the process of engineering design and manufacture. They are the records of collective and mature knowledge of the professions. A volume of engineering standards or codes is actually a knowledge base of the engineering profession in a concise literal form.

By now, various engineering codes have been used in CAD/CAM. In (1) (2), engineering codes have been successfully treated as knowledge bases and a knowledge based codes management system, KECOMS was developed.

1.2 The function and composition of KECOMS

KECOMS is a software package which supports engineers to acquire, translate and manage the defined knowledge forms of engineering codes, and therefore to build engineering knowledge bases. With these knowledge bases users can inquire the contents of codes when giving a subject of the request, determine the engineering design constraints according to the design requirements, and verify the conformance of engineering activities according to the related set of codes.

The implemented KECOMS includes following components:
(1) a text editor; (2) a graphic editor; (3) a translator of a code knowledge input language - TOL; (4) an information network manager; (5) a classifying tree manager; (6) a data dictionary manager; (7) the inquiry and reasoning system.

1.3 Knowledge representation structures in KECOMS

Various knowledge representation techniques are introduced in KECOMS to represent and prcess the codes knowledge, including decision tables, production rules, classifying trees, frames and procedures. A multi-level codes knowledge structure is adopted. Based on this structure, knowledges in the codes knowledge base are classified into four kinds: the data knowledge, the provision knowledge, the organization knowledge and the application knowledge. Different knowledge takes the different representation structure: the provision knowledge is represented by the decision table structure, the organization knowledge by the semantic network and classifying tree structure and the application knowledge by production rules.

The code processing technique was founded by S.J. Fenves (3)(4). Using decision tables and information networks, he proposed a model as the basis of the standard processing. In KECOMS, the decision tables and information networks are seperated from the code processing. A relevant representation language for decision table, called TOL, was invented. The following is a TOL represented decision table example from a knowledge base of the case "The Tentative Specification of Harbour Work".

Ex.1: a decision table which is represented as a TOL program

```
  code jtj22087 5-1-32;
  datum L5_1_32 is logical for result;
  datum ReInConc(c,1),BiaPull(c,1),Ring(c,1),Tangent(c,1)
        is character*20 for cls;
  datum DSCON(c,1) is character*20 for input;
 datum  Nz(a,1), Agl(a,1),eO(a,1),rg(a,1) is float for input
```

```
datum K1(a,1) is float for refer ta 3-2-2_t1;
......
datum Betae(a,1) is float for refer dt 5-1-32_1;
ccondition 1: ReinConc=="T";
......
ccondition 5: DSCON=="dscon3";
action 1: (y,y,y,y,y)=>L5_1_32=K1*K2*Nz<=Rgl*Agl*Betae/(1+eO/rg);
end
```

1.4 The coming of OOKSDE

For the complexity of problems in reality, KECOMS takes a mixed knowledge representation structure. It is really a complex task for a system to manage so many kinds of different knowledge structures efficiently. As a result, the knowledge and the processing was not seperated well and some structures took their simple forms in the implemented system. It also results in the limited usage of KECOMS. To solve the problems, OOKSDE (an Object Oriented Knowledge base System Development Environment) is developed.

The initial goal of OOKSDE is to succeed the functions of KECOMS. However, the ultimate goal of OOKSDE is to expand the processing techniques to various engineering professional knowledge processing. The result of our recent researches includes following achievement: an object-oriented knowledge representation method (OOKRM); an object knowledge base design method; an OO knowledge representation language (SMULA) and an object knowledge base system development environment (OOKSDE) (being developed).

2. THE OVERVIEW OF OOKSDE

2.1 The structure and composition of OOKSDE

OOKSDE is a knowledge object base system development environment. The system developed by OOKSDE is a knowledge based system, which contains a collection of various kinds of objects. When combined with the object codes interpreter and the object manager, the system can be on executing. It can perform the operations, such asinquiring code provision content, verifying the engineering design and creating the engineering design constraints etc.. What the developed system may do is determined by the characteristics of the knowledge objects created and developed by the user.

OOKSDE is an integrated environment. It is composed of the following related parts:
(1) a full-screen object editor. It provides the convenient creating, editing and modifying of text contents of knowledge objects.
(2) a knowledge object base manager. It gives the routine processing of objects, such as retrieving, storing, copying and deleting of objects. It also provides mechanism to retrieve and store the data about relations between objects in the object base.
(3) an object inference controller. It performs the operations to control and monitor object evaluating.
(4) a running code interpreter. It performs the real-time interpreting of executable programmed codes in objects. Operating with the object inference controller, it implements the knowledge objects inference. They both make the core of inference in the knowledge object base system.
(5) a user interface generator. It helps to generate the user interface objects for the knowledge base system being developed. The inter-

face objects need to be interpreted by the running code interpreter before going on operation.
(6) a user-friendly interface of environment. It is the interface through which user can do operations on objects and activate the functions provided by the environment.

Figure 1 reveals the components of OOKSDE and their relation in the environment.

2.2 The user interface of OOKSDE

Figure 2 provides the first-stage user interface of OOKSDE.

OOKSDE is designed for developing one and only one knowledge base system during a particular period. More than one knowledge base system can exist at the same time, but only one of them can be active. The process of developing a knowledge base system is a long period task.

It is supposed that user with OOKSDE can define the class object of knowledge, input the actual knowledge body objects and modify the text content of objects through selecting and entering the (Edit) icon. By selecting the (Compile) icon, objects entered are translated into inner represented forms. Through the (Browser) icon, user can browse the objects stored in system, and can inquire the relation between them. By selecting and entering the (Debug) icon, user can execute and debug the system already built up. The environment also provides the on-line help facilities to give the help information for the system operation, errors and language etc..

3. SMULA, THE KNOWLEDGE REPRESENTATION LANGUAGE IN OOKSDE

3.1 The overview of SMULA

Based on our researching experience in KECOMS, OOKSDE is supposed to be different from other knowledge base system development environments and expert system shells. It provides powerful facilities to handle the engineering knowledge, especially for engineering codes. SMULA, is the knowledge representation and processing language in OOKSDE. In its design and implementation, the following features have considered: (1) the C-PASCAL based language structure is adapted; (2) to represent the special engineering knowledge characteristics, several unique super types are introduced; (3) for processing large scale knowledge bases and introducing reusablity of knowledge , object compiling, template technique and the technique for long period task are introduced in.

A SMULA program can be seperated into two parts, class definition and object instancing. The class definition is for the definition of knowledge object structure. It is composed of two parts. The object instancing is for acquiring actual knowledge body. It is different from each other for the four kinds of objects. See Figure 3.

```
                                       ┌-- field definition
             ┌---- class definition----┤
             |                         └-- method specification
SMULA ---────┤                           ┌- Interface object
             |                           ├- Main object
             └- object instancing -------┤
                                         ├- Knowledge object
                                         └- Enumerate object
```

(Figure 3) The program structure of SMULA

3.2 Some important features of SMULA

Ex.2 : Definition of class object DECISION_TABLE. It is the SMULA represented decision table structure.

```
   CLASS  DECISION_TABLE of STRUCT
   /*  STRUCT is the super class in OOKSDE  */
   {
   (1)    def  PNO : string[10];
   (2)    def  TEXT_ID : IDENT;
   (3)    def  TEXT_V  :  text;
   (4)    def  C
   (5)      C_begin
   (6)         C_expr  :  EXPR;
   (7)      C_end    /* condition parts of decision table  */
   (8)    def  A
   (9)      A_begin
   (10)      action_condition  :  array[1..*] of char;
   (11)           action_expr  :    EXPR;
   (12)     A_end    /*  action parts of decision table */
          /* other method are omitted here  */
          ......
          method  get_dt_result() : value_set;
          /*  evaluate the value of decision table  */
          {
              var_begin
                int   i,j,succ,k;
              var_end;  /* declaration of inner variables */
              for (i=1;i<=STRUCT<-number_of:A;i++) {
                 succ = 1;
                 for (j=1;j<=STRUCT<-number_of:C;j++)  {
                   if (action_condition[i][j] != 'I') {
                      k=C_expr j <-Object_value;
                      if (k != 0) {
                         if (action_condition[i][j] != 'Y') {
                            succ = 0;
                            exit;
                         }
                      }
                   else {
                     if (action_condition i  j  != 'N') {
                         succ = 0;
                         exit;
                     }
                   }
                 }  /*   for  j  */
                 if (succ) return(action_expr  - Object_value);
              }    /*   for i  */
              return(nul);
          }
          method  Object_value() : value_set;
          {
              self  - get_dt_result;
          }
   }          /*  DECISION_TABLE   End  */
```

3.2.1 Super types

There are several super types pre-defined in SMULA. They are the expression, the variable and the object statement.

An object of an expression type (EXPR, see Ex.2, (6) (11)) is a string which conform the grammar and semantic structure defined in

OOKSDE. It is needed frequently in processing engineering codes knowledge to represent the mathematical formula. The adoption of expression type makes the acquiring of mathematical formula much conveniently.

An object of a name appears in an expression is treated as an object of a variable type (VARIDENT). The type (VARIDENT) can be re-defined as an object class. This mechanism implements the variable with user defined features.

The object statement type (OBJ_STAT) is also a string. But it must conform the grammar of object activating statement in SMULA. The actual effect is achieved when the name of (OBJ_STAT) type is replaced by its value - the executable object activating statement, and when the statement is interpreted. This operation is called macro substitude in SMULA.

With these super types pre-defined in environment, it becomes a simple task to input the knowledge body objects. Ex.3 presents an actual decision table in SMULA.

Ex.3 : Following is a provision represented as a decision table. The provision c5s1p32 is selected from "The Tentative Specification of Harbour Work". It is instanced according to the object class defined in Ex.2.

```
    KB  c5s1p32  of DECISION_TABLE
    {
    (1)  PNO = "c5s1p32";
    (2)  TEXT_ID = "TXT_c5s1p32";
    (3)  C  ReinConc == 'T';
    (4)  .....
    (5)  C  DSCON == "dscon3";
    (6)  A (Y,Y,Y,Y,Y),K1*K2*Nz<=Rgl*Agl*Betae/(1+e0/rg);
    }
```

3.2.2 Template technique

Template is a special date abstrction mechanism, with three different forms: simple, multiple and nested. Statement (1)-(3) in Ex.2 are simple template. Statement (4)-(7) and (8)-(12) in Ex.2 are multiple template.

The different between simple and multiple template is that the simple one can be instanced just one time but the multiple one can be instanced more than one time. See Ex.3, (1) (2) are the instanced fields of simple template and (3)-(5) are the instanced fields of multiple ones.

With other features, such as repeated definition and default mechanism, template technique greatly simplifies the processing of data abstraction. The language TOL in KECOMS is now in OOKSDE replaced by the class object definitions. Comparing with Ex.1, code provision c5s1p32 in Ex.3 is much more concisely defined.

Reference (6)(7) gives some more examples showing the representation ability of SMULA. Most of commonly used knowledge representation structures can be represented by SMULA.

3.2.3 Object compiling and long period task technique

In OOKSDE, the compilation is based on a single object. The process of building a knowledge base system is treated as a long period task with all temperary statuses are recorded. This makes the developing a knowledge base system can be implemented step by step during a period of fairly long time.

4. THE PROCESS OF INFERENCE IN KNOWLEDGE OBJECT BASE

The inference of knowledge object base system is the process of evaluating objects. The "sailing" from object to object implements the process of inference. During the process of verifying the comformance of engineering design, the main operation is to evaluate relevant code provisions. In another word, it is to evaluate the objects of relevant decision tables. For short, we consider the evaluating of the object in Ex.3, c5s1p32. To be understood clearly, Ex.4 gives another object represented in SMULA.

Ex.4 Following is an instanced data table object represented by SMULA. The data table is attached to provision c3s2p2.

```
    KB  c3s2p2_t1    of  STATIC_TABLE
    {
        PNO = "c3s2p2_t1";
        t_data = (1.55,1.45,1.65,1.50,1.65,1.50);
        COL  ((AxPull=='T') (Crook=='T')) & (ReinConc=='T');
        ......
        COL  ((AxPull=='T') (Crook=='T')) & (PressConc=='T');
        LINE  Ladcomb=="design";
        LINE  Ladcomb=="proof";
    }
```

When all needed objects are available in the knowledge base, the following relation is automatically built up in the knowledge base system or in OOKSDE. In Figure 4,[<--] means the left one can be evaluated after the right ones have been evaluated.

```
    c5s1p32<-- ReinConc (input)
           <-- ...... (input)
           <-- K1 <-- c3s2p2_t1 <-- PresConc (input)
                                <-- ...... (input)
           <-- K2 <-- c3s2p2_t2 <-- Chocond (input)
            ......
           <-- Betae <-- c5s1p32_1 <-- e0 (input)
                                   <-- ...... (input)
```

(Figure 4) The relation built up around provision c5s1p32

Suppose the following values are input:
ReinConc = 'T'; DSCON = "dscon3"; Ladcomb = "proof; chocond = "chocond2"; e0 = 5; Agl = 14.5; ......
The evaluation begins with the object c5s1p32. The following is the evaluating chain during inference.

```
The goal of inference : the value of c5s1p32;
1-1)activate the method Object_value in c5s1p32,
1-2)executing the method in class DECISION_TABLE,
  2-1)evaluate field C, the result is (Y,Y,Y,Y,Y),
  2-2) evaluate field A, that is to evaluate  the expression
       K1*K2*Nz<=Rgl*Agl*Betae/(1+e0/rg),
  2-3)activate the method (expr_value) to get the value of the
      expression,
    3-1)evaluate the variable K1,
    3-2)activate the method (Object_value) in VARIDENT,
    3-3)activate the method (Object_value) in c3s2p2_t1,
      4-1)evaluate COL field, get the result COL[1] = true,
      4-2)evaluate LINE field, get the result LINE[2]= true,
      4-3)get the data t_data[1,2] = 1.45,
      4-4)return the value 1.45,
    3-4)get the value, K1=1.45,
    3-5)...... /* evaluate following variables */
    3-i)get the value, K2=1.05, Rgl=550, ...
        input value, Nz=4000, e0=5, ...
  2-4)the result is 1, return the value,
1-3)get the value of c5s1p32, the inference process ends.
```

5. CONCLUSION

OOKSDE is a development environment for developing common-purposed and large-scaled knowledge base systems. For its unique features of processing code knowledge, it is also an efficient tool to develop engineering knowledge base system. For early development of OOKSDE, it is being implemented using C++.

REFERENCES

1. SHEN KANGCHEN etc, Knowledge Based Engineering Code Management System - KECOMS and Its Interface to Engineering Databases, Proceedings of the International Conference on Expert Systems in Engineering Application,Oct. 1989.

2. SHEN KANGCHEN, etc., ECOMS: A Good Aid for Building Engineering Knowledge Bases, IV-ICCCBE, Tokyo, July 1991.

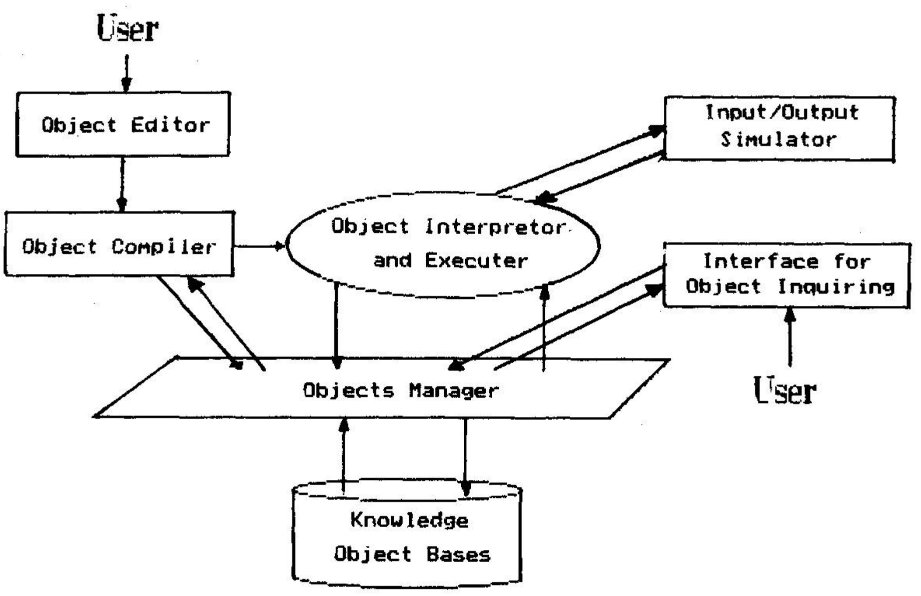
3. FENVES S.J., Software for Analysis of Standards. Computing in Civil Engineering, New York, 1979.

4. FENVES S.J., Representation of the Computer-Aided Design Process by a Network of Decision Tables. Computer & Structures, Vol.3, p.1099-1107, 1973.

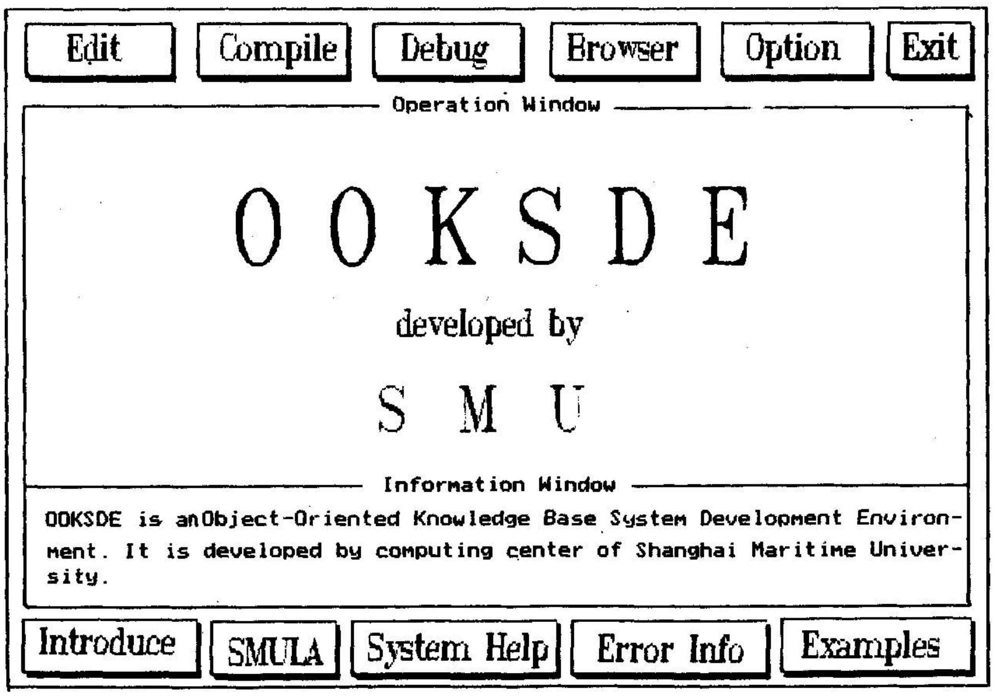
5. YANG XIAOFENG, LI HONG, The Study of OO Knowledge Representation Method and OO Knowledge Bases, Proceedings of CAAI-7, April 1992.

6. YANG XIAOFENG, LI HONG, The Study of Representation of Engineering Code Knowledge, Proceedings of CAAI-7, April 1992.

7. YANG XIAOFENG, SHEN KANGCHEN, SMULA - the Knowledge Representation Language in OOKSDE, To be published on Proceedings of IEEE TENCON'93, Beijing, Oct. 1993.



(Figure 1) The structure of OOKSDE



OOKSDE is anObject-Oriented Knowledge Base System Development Environment. It is developed by computing center of Shanghai Maritime University.

(Figure 2) The first-stage user interface of OOKSDE