データベース・システム　78－4
（1990．7．19）

複合オブジェクトのブラウジング手法とその評価

宇田川　佳久
三菱電機㈱情報電子研究所

ＣＡＤ／ＣＡＭなどのアプリケーションでは、過去に作成されたオブジェクトをブラウジングする機能が求められている。本文では、部分一致によるブラウジング・アルゴリズムについて論じている。この手法の主な特徴は、
(1) 概略図によってオブジェクトの選択条件を指定することができるグラフィック・インタフェース
(2) 階層構造を利用した効率的な関連オブジェクトの検索
(3) 選択した複合オブジェクトを順序づける一致尺度の導入
である。標準ＩＣデータベースに適用した結果を示し，効率とアルゴリズムの性質について論ずる。

A Technique to Browse Composite Objects and its Evaluation

Yosihisa UDAGAWA
Mitsubishi Electric Corporation
5-1-1 Ofuna, Kamakura City, Kanagawa, 247, JAPAN

ABSTRACT

Browsing a composite object is one of the problems arising in the applications where reuse of previously designed parts is of special importance. In this paper, we discuss a browsing algorithm based on partial match of composite objects. The main features of the algorithm are summarized as follows;
(1) a graphics user interface that allows a user to specify a selection condition by means of a rough sketch,
(2) use of aggregation hierarchies of composite objects to select related objects efficiently,
(3) introduction of matching measures to order the selected objects.

We have evaluated the algorithm with a standard IC database. Performance and characteristics of the browsing algorithm are discussed based on the experimental results.

1. Introduction

During the last few years, a number of research efforts have been directed at database management systems for advanced applications such as CAD/CAM, software engineering etc. In these applications, the objects are much more complex than conventional business applications. They are usually described by using a large number of records that are highly interconnected. A common theme running through many of the researches for engineering databases is organizing and retrieving objects having complex structures [1,3,4,7].

Among several approaches to support the composite object, we are interested in extending the relational data model because it is based on strong mathematical theories and because it is widely used in practical applications including engineering [1,3,7]. So far we have proposed an extended relational database called ADAM (Advanced Database with Abstraction Mechanism) and discussed the structural aspects of the model [6]. ADAM has been applied to the management of VLSI schematics.

In this paper, we are concerned with a browsing algorithm of a composite object. A database support to reuse previously designed objects is of special importance in CAD applications [5]. On browsing objects in the context of reusing objects, users of a design system require the following ;

(1) The users want to communicate with the system in terms of objects they are dealing with, e.g., schematics, layout geometry. A graphic user interface seems to be essential.
(2) Objects that partially satisfy given conditions should be selected and browsed to explore all of the related objects.
(3) The selected objects should be displayed in the order of degrees of how well they satisfy the conditions.

In this connection, we are well aware of many researches that discuss methods of selecting a composite object. Adiba [1], and Lorie et al.[2] introduces a notion of "logical path" or "logical pointer" to navigate a forest of composite objects efficiently. "Logical path" or "logical pointer" can be supported through slight modification of SELECT clause of the SQL language and proved to be useful in selecting a composite object based on Boolean conditions. However, explicit designation of "logical path" does not seem to be useful in browsing objects because browsing objects requires to explore all of the related objects. In other words, "logical path" seems to limit so much of search spaces that it may cut off many of possible access paths to the related objects. Actually we define "retrieving" and "browsing" as follows; retrieving is a selection of objects that satisfy all of the selection conditions, whereas browsing is a selection of objects that satisfy some of the conditions.

In this paper, we discuss a browsing algorithm of a composite object based on partial match. The algorithm takes advantage of an object hierarchy. We have introduced two measures to order the selected objects for user's convenience. We have also developed a user interface that allows a designer to specify selection conditions by means of a rough sketch of a composite object (a lure object, hereafter). We describe modeling capabilities of an extended relational database ADAM in the context of digital IC schematics. We also discuss a sophisticated browsing system called APPLE (A Prime Partner for Leading Engineers), and a user interface called EVE (Editor in Visual Environment), which allows users to edit and browse schematics only using terms in circuit design.

2. ADAM Data Model

2.1 Modeling Concepts

Engineering design is the process of building up a model of a complex artifact. The artifact may usually be decomposed into hundreds of other components and a component may be an assembly of smaller components. Thus a design object can usually be represented as a hierarchy of more primitive objects. An object built in this way is called a **composite object**.

Since design evolves over time, the design database has to support multiple version of a design object. Versions of an object are defined as objects that share the same specification and differ only in their implementation. A hierarchy that is composed of a collection of specific versions is called **configuration**.

ADAM data model supports all of the three modeling concepts, i.e., composite object, version and configuration.

2.2 Data Dictionary

To implement the modeling concepts discussed above, a data dictionary is developed. It consists of two relations. One is a relation DDD, which maintains identifiers and related information about versions of an object. The relation DDD consists of the following attributes:

(1) an object name (O_ID),
(2) a version identifier (VSN),
(3) change notification time (CNT),
(4) change approval time (CAT),
(5) last referenced time (LRT),
(6) a test level (TESL),
(7) a version derivation identifier (PVE).

The other is a relation CONFIGURATION that maintains the relationship between composites and components. The relation consists of tuples having the following attributes ;
(1) a parent object name (PARENT),
(2) a version identifier of the parent object (P.VSN),
(3) a child object name (CHILD),
(4) a version identifier of the child object (C.VSN),
(5) the number of child objects used to define the parent object (NUM).

Using the data dictionary, a user can get sufficient information to control composite objects, versions, configuration, creation and deletion of objects through a simple command. For example, the following command retrieves tuples relating to the components of M195 (or SN74195) with version #1.0. Figure 1 shows the resultant relation.

```
SELECT     *
FROM       CONFIGURATION
WHERE      PARENT = 'M195'
           P.VSN = '#1.0' ;
```

```
 LN  PARENT      P.VSN      CHILD       C.VSN      NUM
-------------------------------------------------------
  1  M195    D  #1.0   +  JNCAS   D  #1.0   +    4
  2  M195    D  #1.0   +  INTS    D  #1.0   +    7
  3  M195    D  #1.0   +  NOT2S   D  #1.0   +    3
  4  M195    D  #1.0   +  NOT3S   D  #1.0   +    1
  5  M195    D  #1.0   +  ANT2S   D  #1.0   +    7
  6  M195    D  #1.0   +  ANT3S   D  #1.0   +    2
  7  M195    D  #1.0   +  NAT2S   D  #1.0   +    1
```

Fig.1 Relation Describing Composite-Component Aggregation.

2.3 Describing Schematic through Relations

Structures of a schematic are represented in terms of relations. Current prototype system uses three relations below [21] ;

```
COMP( DIV_ID, DIV ),
TERM( T_ID, DIV_ID, CHR, XC, YC ),
CONN( L_ID, S_ID, DIV_S, D_ID, DIV_D ).
```

Figure 2 shows a set of relations that describe a schematic in figure 3. The relation COMP lists components of the schematic. Each component has a unique identifier (DIV_ID). A specific value 'D0' is assigned

```
COMP ( DIV_ID/ID, DIV/ABS ).
     1  D001, JNCAS(-2.0,+30.0)
     2  D002, JNCAS(-2.0,+16.0)
     3  D003, JNCAS(-2.0,+2.0)
     4  D004, JNCAS(-2.0,-12.0)
     5  D005, INTS(-12.0,+32.0)
     6  D006, INTS(-12.0,+18.0)
     7  D007, INTS(-12.0,+4.0)
          :
    21  D021, ANT3S(-22.0,-14.0)
    22  D022, ANT3S(-22.0,-18.0)
    23  D023, INTS(-25.0,-24.0)
    24  D024, INTS(-18.0,-24.0)
    25  D025, NAT2S(-23.0,-28.0)
    26  D0, M195(X,Y)
```

```
TERM ( T_ID/ID, DIV_ID/ID, CHR/ID, XC/AD, YC/AD ).
     1  Q0, D001, TP_N, -1.5, +32.0
     2  Q1, D001, TP_N, -1.5, +28.0
     3  I0, D001, TP_N, -8.5, +32.0
     4  I1, D001, TP_N, -8.5, +30.0
     5  I2, D001, TP_N, -8.5, +28.0
     6  I3, D001, TP_N, -5.0, +25.5
     7  Q0, D002, TP_N, -1.5, +18.0
          :
    90  I006, D0, TP_N, -30.0, -24.0
    91  I007, D0, TP_N, -30.0, -27.0
    92  I008, D0, TP_N, -30.0, -32.0
    93  Q000, D0, TP_N, , +32.0
    94  Q001, D0, TP_N, , +28.0
    95  Q002, D0, TP_N, , +18.0
    96  Q003, D0, TP_N, , +4.0
    97  Q004, D0, TP_N, , -18.0
```

```
CONN ( L_ID/ID, S_ID/ID, DIV_S/ID, D_ID/ID, DIV_D/ID )
     1  L001, I000, D0, I0, D014
     2  L002, I001, D0, I0, D016
     3  L003, I002, D0, I0, D018
     4  L004, I003, D0, I0, D020
     5  L005, I004, D0, I1, D021
     6  L006, I005, D0, I1, D022
     7  L007, I006, D0, I0, D023
          :
    57  L057, Q0, D019, I1, D012
    58  L058, Q0, D021, I1, D013
    59  L059, Q0, D020, I0, D013
    60  L060, Q0, D022, I2, D013
```

Fig.2 Relations Describing Circuit.

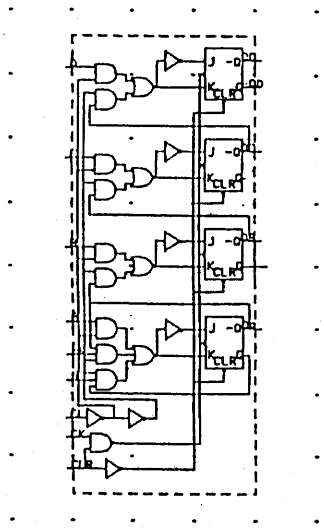


Fig.3 Example Circuit.

to represent the interface description of the schematic. Each item in the DIV column encodes an identifier of the schematic and geometric placement on a display. The relation TERM contains information about terminal points of the schematic. The relation CONN describes wires of the schematic.

3. Browsing Algorithm in ADAM

3.1 Overview

Given a lure schematic (a rough sketch) as the selection condition, the browsing algorithm in ADAM data model carries out the following processes;

(1) collecting related schematics or candidates that include some components of the lure schematic.

(2) calculating weights of each candidate.

A circuit is usually a composition of some component forming a hierarchy of circuits. By following the hierarchy upward, we can find candidates of a given lure. Information about a circuit and its components can be retrieved from the data dictionary through simple commands. Details are discussed in section 3.2 of the paper.

The next step is to weigh the candidates based on how well they match the lure schematic. Current prototype system adopts two measures. One is a measure, called RDMC (relative difference of matching components), which is calculated from the number of matching components among candidates and the lure schematic. The other is the number of nonmatching components, and called NNC. Intuitively, NNC is a measure representing how far two given objects differ from.

3.2 Calculating RDMC

Since a schematic is described in terms of relations in our database, we can define RDMC and NNC among the schematics based on the differences among the relations. In principle, since RDMC and NNC are the number of matching and nonmatching components of the schematics, as mentioned above, we can calculate them through a simple combination of set operations. However in practice, because a component can be referred many times by a schematic, simple set operations do not work well.

We have adopted the following algorithm. Let $N(x_i)$ and $M(x_i)$ be the number of a matching component x_i between a lure schematic X and a candidate S_j. We define the difference of matching components (DMC) between the lure X and the candidate S_j, denoted by DMC(X, S_j), as follows ;

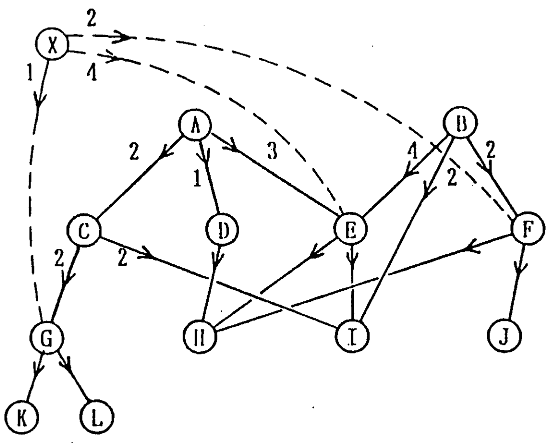


Fig.4 Example of Object Hierarchy.

$$\mathrm{DMC}(X, S_j) = \sum_{i=1}^{n} | N(x_i) - M(x_i) |$$

where n is the number of matching components between X and S_j.

Suppose schematics A and B are built up of lower level components C, D, E, F, etc. as shown in figure 4. And let a lure schematic X consists of components E, F and G. The candidates of the lure X are A, B and C because A, B and C include the matching components E, F and G.

Since E and F are matching components between the lure X and the candidate B, we obtain

$$\begin{aligned}\mathrm{DMC}(X, B) &= |N(E) - M(E)| + |N(F) - M(F)| \\ &= |4 - 4| + |2 - 2| \\ &= 0.\end{aligned}$$

In the same way, DMC between the lure X and the candidate C is

$$\begin{aligned}\mathrm{DMC}(X, C) &= | N(G) - M(G) | \\ &= |1 - 2| \\ &= 1.\end{aligned}$$

The object A includes three Es and two Cs and C includes two Gs. Since the object A includes components hierarchically, the definition of DMC is applied recursively, i.e. DMC between X and A is calculated as follows ;

$$\begin{aligned}\mathrm{DMC}(X, A) &= |N(E) - M(E)| + M(C) * \mathrm{DMC}(X,C) \\ &= |4 - 3| + 2*1 \\ &= 3.\end{aligned}$$

Note that the more a lure schematic and a candidate schematic match, the smaller value DMC(X, S_j) is produced. In fact, DMC(X, S_j) is zero if a lure schematic X and a candidate S_j are composed of the same number of matching components. Now we define "relative difference of matching components" (RDMC) among candidates to meet the human intuition, i.e., the more components match, the larger the value is.

$$\mathrm{RDMC}(X,S_j) = \mathrm{FMAX} - \mathrm{DMC}(X, S_j) + 1$$

where FMAX is the maximum value of DMC of the selected candidates. Thus RDMC is relative to a set of candidates. In the above example, since FMAX is equal to 3, we have

RDMC(X, B) = 4
RDMC(X, C) = 3
RDMC(X, A) = 1.

3.3 Calculating NNC

NNC, on the other hand, is defined based on the number of nonmatching components. Let N(y_i) and M(z_i) be the number of nonmatching components of a lure schematic X and a candidate schematic S_j , respectively. Then NNC between the lure X and the candidate S_j , denoted by NNC(X,S_j), is as follows ;

$$\mathrm{NNC}(X, S_j) = \sum_{i=1}^{n} N(y_i) + \sum_{i=1}^{m} M(z_i)$$

where n and m are the numbers of nonmatching components of the lure and the candidate, respectively.

NNCs among the lure X and the candidates B, C and A in figure 5 are as follows;

$$\begin{aligned} \mathrm{NNC}(X, B) &= N(G) + M(I) \\ &= 1 + 2 \\ &= 3 \\ \mathrm{NNC}(X, C) &= N(E) + N(F) + M(I) \\ &= 4 + 2 + 2 \\ &= 8 \\ \mathrm{NNC}(X, A) &= N(F) + M(C) * \mathrm{NNC}(X,C) + M(D) \\ &= 2 + 2*8 + 1 \\ &= 19 \end{aligned}$$

Note that in calculating NNC between X and A, the definition of NNC is applied recursively because the object A includes components hierarchically.

4. Implementation of Browsing Algorithm

4.1 Overall Algorithm

Figure 5 shows an overall algorithm of browsing schematics. All a user has to do is the following actions.

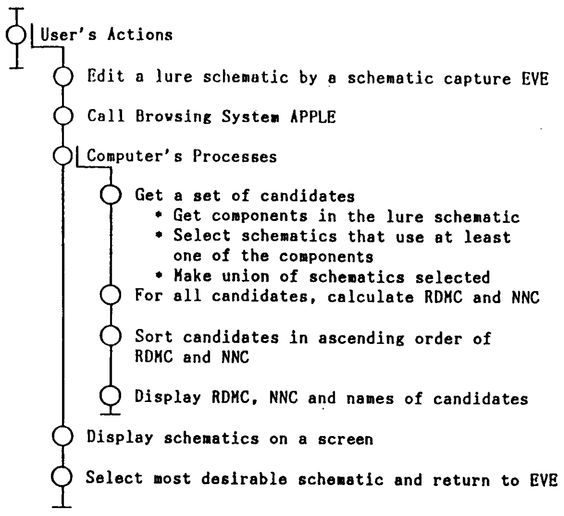

User's Actions
Edit a lure schematic by a schematic capture EVE
Call Browsing System APPLE
Computer's Processes
Get a set of candidates
• Get components in the lure schematic
• Select schematics that use at least one of the components
• Make union of schematics selected
For all candidates, calculate RDMC and NNC
Sort candidates in ascending order of RDMC and NNC
Display RDMC, NNC and names of candidates
Display schematics on a screen
Select most desirable schematic and return to EVE

Fig.5 Overall Browsing Algorithm.

(U-1) Edit a lure schematic by using the schematic capture EVE just in the same manner as usual schematics.
(U-2) Select an item APPLE in a command menu of EVE to call the browsing system APPLE.
(U-3) Display schematics on a screen by referring to the two measures, i.e. RDMC and NNC.
(U-4) Select the most appropriate schematic and return to the schematic capture EVE.

On the other hand, APPLE executes the following processes.
(A-1) Get a set of candidates of the lure schematic.
This process consists of the following subprocesses.
(A-1.1) Get components in the lure schematic given by a user.
(A-1.2) Select schematics that include at least one of the components by accessing the data dictionary.
(A-1.3) Build a set consisting of all names of schematics appearing in the selections A-1.2.
(A-2) For all candidates, calculate RDMC and NNC.
(A-3) Sort candidates in ascending order of NNC, and if there are multiple candidates with the same NNC, they are arranged by RDMC.
(A-4) Display RDMC, NNC and names of each candidate.

To get components of the lure schematic (or to implement(A-1.1)), the system simply executes a SQL command below.

```
SELECT    DIV
FROM      COMP
WHERE     DIV_ID /='DO' ;
```

To select schematics that use at least one of the components in the lure schematic (or to implement (A-1.2)), the following command should be executed, for each component, to the data dictionary.

```
SELECT    PARENT, P.VSN
FROM      CONFIGURATION
WHERE     CHILD = "name of component
                   of the lure schematic"
```

A set of candidates of the lure schematics is a union of results of the data retrieval.

4.2 Example of Browsing

Figure 6 shows a result of browsing that uses a SN7496 (M96) as the lure schematic. The system has selected 41 candidates for the lure. Just after APPLE is activated, the command and candidate panes of the browsing subwindow are displayed on the screen. A user can select any candidate from the candidate pane and display it on the browsing window to make sure the schematic is the one he/she is searching for. We have displayed two candidates that match more to the lure schematic, whereas two candidates that match less on the lower subwindows.

5. Evaluation of Browsing Algorithm

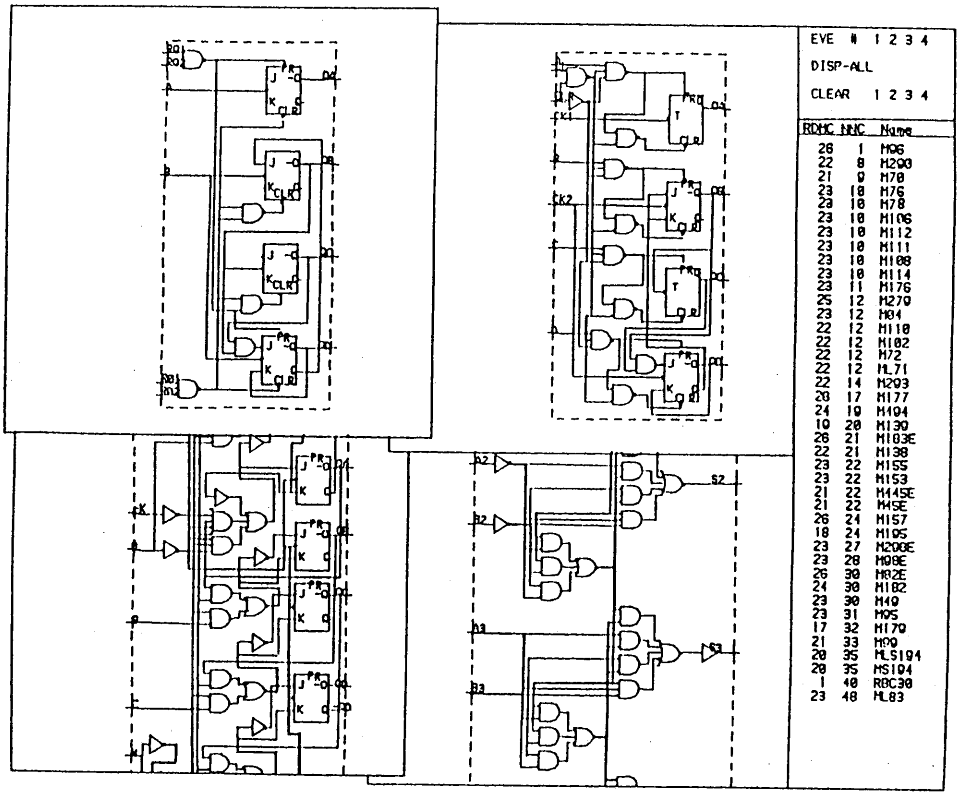
5.1 Performance

In this section, we present some measurement results of browsing experiments that we performed to evaluate the performance of the browsing algorithm.

5.1.1 Instrumentation

The algorithm is implemented on a MELCOM COSMO 900 II computer system that has approximately 4 MIPS processing power. The program is written in FORTRAN and consists of about 2000 lines of source code dedicated to the browsing module.

Performance experiments were carried out on a standard IC database. The relation



EVE # 1 2 3 4

DISP-ALL

CLEAR 1 2 3 4

RDMC	NNC	Name
26	1	M96
22	8	M290
21	9	M70
23	10	M76
23	10	M78
23	10	M106
23	10	M112
23	10	M111
23	10	M108
23	10	M114
23	11	M176
25	12	M270
23	12	M04
22	12	M110
22	12	M102
22	12	M72
22	12	ML71
22	14	M293
20	17	M177
24	19	M194
10	20	M130
26	21	M103E
22	21	M138
23	22	M155
23	22	M153
21	22	M145E
21	22	M45E
26	24	M157
18	24	M195
23	27	M200E
23	28	M98E
26	30	M82E
24	30	M182
23	30	M49
23	31	M95
17	32	M179
21	33	M90
20	35	MLS194
20	35	MS194
1	40	RBC30
23	48	ML83

Fig.6 Browsing Schematics.

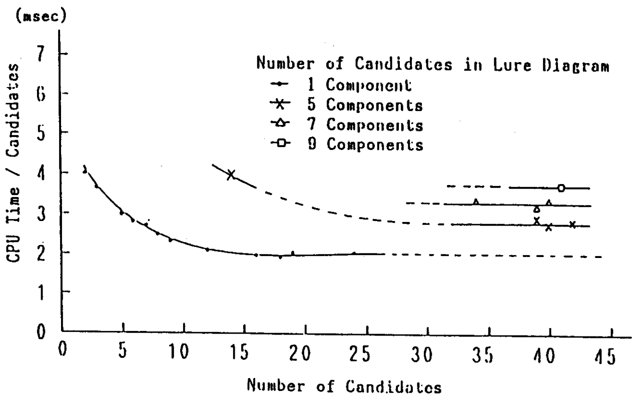


Fig.7 Execution Time of Retrieving Candidates.

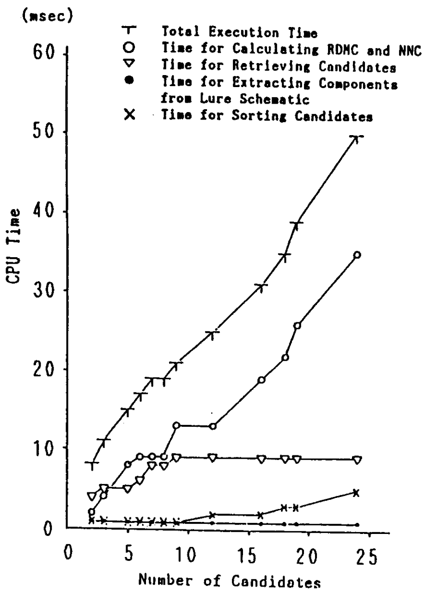


Fig.8 Execution Time Components.
(for lure consisting of one component)

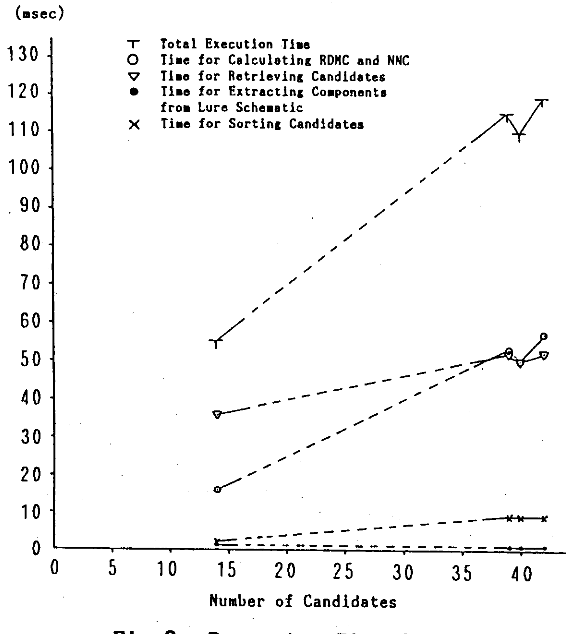


Fig.9 Execution Time Components.
(for lure consisting of five components)

DDD in the data dictionary contains approximately 300 tuples and the relation CONFIGURATION contains approximately 250 tuples.

5.1.2 Measurement

Figure 7 shows some measurement results of the browsing algorithm. We have used the following classes of lure schematics for the experiments, i.e.,
(1) lures consist of one component,
(2) lures consist of five components,
(3) lures consist of seven components,
(4) lures consist of nine components.

As for the lure schematics consisting of one component, 2 through 24 candidates are retrieved. As for other schematics, because they include multiple components, around 40 candidates are retrieved.

The CPU time per candidate is less than 4 milliseconds on a COSMO 900 II and decreases as the number of candidates increases. This is because the effect of overhead such as initializing variables and loop parameters are lessened as the number of candidates increases. Another observation is that more CPU time is needed as the number of components of a lure schematic increases.

5.1.3 CPU Time Components

As discussed in section 4, the algorithm consists of four major operations, i.e.,
(OP1) get components of a lure schematic,
(OP2) select candidates from the data dictionary for all unique components of a lure schematic, (OP3) calculate RDMC and NNC for each candidate,
(OP4) sort candidates.
We have measured the CPU time of each of the four operations.

Figure 8 shows details of the measurement results for lure schematics with one component. The CPU time to execute the operation OP1 is one millisecond and is constant for the number of candidates ranging from 2 to 24. The time to select

Table 1 RDMC and NNC.
(for lure consisting of 2-Input-NORs)

LURE	1		2		3		4		6		8		15	
CIRCUITS	RDMC	NNC	RDMC	NNC	RDMC	NNC	RDMC	NNC	RDMC	NNC	RDMC	NNC	RDMC	NNC
SN7455	4	2	2	2	1	2	1	2	1	2	1	2	1	2
SN7451	3	4	3	4	2	4	2	4	2	4	2	4	2	4
SN74155	3	14	3	14	2	14	2	14	2	14	2	14	2	14
SN74494	1	16	1	16	2	16	4	16	4	16	4	16	4	16
SN74182	4	18	2	18	1	18	1	18	1	18	1	18	1	18
SN7498	1	18	1	18	2	18	4	18	4	18	4	18	4	18
SN74298	1	19	1	19	2	19	4	19	4	19	4	19	4	19
SN7495	1	21	1	21	2	21	4	21	4	21	4	21	4	21
SN74195	2	22	2	22	3	22	3	22	3	22	3	22	3	22
SN7499	1	27	1	27	2	27	4	27	4	27	4	27	4	27
SN7449	2	31	2	31	3	31	3	31	3	31	3	31	3	31

Table 2 RDMC and NNC.
(for lure consisting of 2-Input-NANDs and 3-Input-NANDs)

LURE	(1,1)		(2,2)		(3,3)		(4,4)		(6,6)		(8,8)		(10,10)		(15,15)		(20,20)	
CIRCUITS	RDMC	NNC	RDMC	NNC	RDMC	NNC	RDMC	NNC	RDMC	NNC	RDMC	NNC	RDMC	NNC	RDMC	NNC	RDMC	NNC
SN74279	2	0	3	0	2	0	1	0	2	0	1	0	1	0	1	0	1	0
SN74177	1	5	2	5	3	5	4	5	3	5	2	5	2	5	2	5	2	5
SN74176	1	6	2	6	3	6	4	6	3	6	2	6	2	6	2	6	2	6
SN74293	8	5	6	6	4	7	2	8	1	10	2	12	4	14	9	19	14	24
SN74290	5	6	5	7	5	8	5	9	4	11	5	13	7	15	12	20	17	25
SN74155	1	9	1	10	1	11	1	12	4	14	9	16	11	18	16	23	21	28
SN7496	4	10	4	11	4	12	4	13	5	15	6	17	8	19	13	24	18	29
SN74139	1	11	1	12	1	13	1	14	4	16	9	18	11	20	16	25	21	30
SN74195	8	25	6	26	4	27	2	28	1	30	2	32	4	34	9	39	14	44
SN7449	5	31	5	32	5	33	5	34	4	36	5	38	7	40	12	45	17	50

candidates from the data dictionary is less than 9 milliseconds.

On the other hand, the execution time for the operation OP3 that calculates RDMC and NNC is roughly linear with the number of candidates. A quick sort algorithm is used to order the candidates on the basis of NNC and RDMC. The execution time for the operation OP4 is less than 5 milliseconds.

Figure 9 shows details of the measurement results for lure schematics with five components. The CPU time to select candidates from the data dictionary amounts to approximately five times larger than the corresponding CPU time in figure 8. This is because the data dictionary is accessed once for each unique component, i.e., five times.

5.2 Characteristics of Browsing Algorithm

In this section, we try to examine some characteristics of the browsing algorithm. One of our concerns to the browsing algorithm is "can users expect the approximately same sets of candidates for similar lures ?" In other words, we should know how the set of candidates changes as the lure changes. For the purpose, we have carried out experiments as follows.

5.2.1 Lure Consisting of One Kind of Components

First we have made experiments on the lure that consists of only 2-input-NOR gates. Table 1 lists RDMC and NNC of the candidate schematics as the number of 2-input-NOR gates varies from one through fifteen.

Through the experiments, order of the candidate circuits do not change because the candidates are sorted by NNC first, and NNC is relevant to the number of nonmatching components between the lure and the candidates. Since the lure consists only of the matching component, i.e., 2-input-NOR, NNC remains constant throughout the experiments.

From the table, we can see the following.

(1) RDMC for each candidate varies as the number of 2-input-NOR gates changes from one through four.

(2) The set of RDMCs remains constant when the number of the gates is more than four with the maxima of RDMC being four for SN74494, SN7498, SN74298, SN7495 and SN7499. Their RDMCs reach the maxima among the candidates because they include four 2-input-NOR gates and because other circuits include less than four 2-input-NORs.

5.2.2 Lure Consisting of Two Kinds of Components

Table 2 shows the results of experiments concerning the lure consisting of two kinds of components. (N,M) in the table indicates that the lure is composed of N 2-input-NANDs and M 3-input-NANDs. In the experiments, the number of the components ranges from one to twenty.

It is evident from the table that NNC changes as the lure changes, unlike the experiments before on the lure consisting of 2-input-NOR gates. This is because if a candidate schematic includes only one of the components of the lure , the number of the other component contributes to NNC. For example, SN74290 is composed of four 2-input-NANDs, one 2-input AND and four flip-flops. Thus the number of 3-input-NAND gates, which is a component of the lure, is added to NNC. NNC of SN74290 varies from six to twenty-five as the number of 3-input-NAND gates changes from one to twenty. However, if a candidate includes all of the components of the lure, NNC does not change. For example, since SN74177 and SN74176 includes both 2-input-NAND and 3-input-NAND gates, their NNCs are independent of the lure.

We can see from the table that RDMC changes more dynamically than NNC does. For example, for the lure (1,1), the maximum of RDMC is eight for SN74293 and SN74195. For the lure (6,6), it is five for SN7496, whereas for the lure (20,20), it is twenty-one for SN74155 and SN74139. Note that, in our definition, RDMC is related to the number of matching components. Thus the maximum of RDMC is attached to candidates when they include the same number of components as the lure does. For example, since SN74293 and ·SN74195 include one 2-input-NAND, they have the maximum of RDMC when the lure consists of one 2-input-NAND. As for SN74155 and SN74139, they include eight 3-input-NANDs. Because no other candidates include more 3-input-NANDs than SN74155 and SN74139 do, their RDMCs reach their maxima when the lure consists of more than eight 3-input-NANDs.

6. Conclusion

In this paper, we discussed the implementation and evaluation of an extended relational database called ADAM with special focuses on browsing composite objects. A database for design applications is required to handle composite objects that are usually described by a set of heterogeneous records.

In design processes, browsing previously designed objects are fairly important. We have developed a browsing algorithm for composite objects to support reuse of the objects. The algorithm takes advantage of aggregation hierarchies to select related composite objects and order them based on the two measures, i.e., the relative difference of matching components (RDMC) and the number of nonmatching components (NNC). The algorithm has been implemented and various kinds of experiments have been carried out with a standard IC database. The major points drawn form our study are as follows;

(1) The CPU time roughly linearly depends on the number of selected objects.
(2) Around 90% of the CPU time is consumed in calculating RDMC and NNC.
(3) RDMC is more sensitively affected by retrieval conditions than NNC is.

References

[1] Adida,M.E. "Modeling Complex Objects for Multimedia Databases", in Entity-Relationship Approach, North-Holland, 1987, pp.89-117.

[2] Lorie,R and Schek,H-J. "On Dynamically Defined Complex Objects and SQL", in Advances in Object-Oriented Datbase Systems, Lecture Notes in Computer Science No.334, Springer-Verlag, 1988, pp.323-328.

[3] Ketabchi,M.A and Berzins,V. "Mathematical Model of Composite Objects and Its Application for Organizing Engineering Dtabases", IEEE Trans. on Soft. Eng., Vol.14,No.1, Jan. 1988, pp.71-84.

[4] Kim,W., Ballou,N., Chou,H-T. et al. "Features of the ORION Object-Oriented Database System", in Object-Oriented Concepts, Databases and Applications, Addison-Weseley, 1989, pp.251-282.

[5] Meyer,B. "Reusability: the Case for Object-Oriented Design", IEEE Trans. on Soft. Eng., Vol.-13,No.3, March 1987, pp.50-64.

[6] Udagawa,Y. and Mizoguchi,T. "An Extended Relational Database System and its Application to Management of Logic Diagram", Proc. 12th Intl. Conf. on Very Large Databases, August 1986, pp.267-277.

[7] Wilkes,W., Klahold,P. and Schlageter,G. "Complex and Composite Objects in CAD/CAM Databases", Proc. 5th IEEE Intr. Conf. on Data Engineering, Feb. 1989, pp.443-450.