클래스 부품의 재사용을 위한 객체의 추출과 이해

한 정 수† · 송 영 재††

요 약

정보저장소에 저장된 클래스 부품들은 검색, 추출과정을 통한 부품의 다양한 시각적 정보가 부족하고 정확한 정보의 표현 기능이 요구되고 있다. 따라서 본 연구는 클래스 부품을 정보저장소에 저장하기 위해 객체의 구문분석 방법과 추출 Viewer 기능을 연구하였다. 분석과정에서는 클래스명, 멤버함수, 속성 등을 추출하고 클래스 자체정보와 클래스 사이의 상속관계, View를 위한 다이어그램 정보 등을 추출하였다. 또한 추출한 분석정보를 시각적으로 표현할 수 있는 추출 Viewer 기능을 구현하여 클래스 부품의 재사용과 생성, 삽입, 삭제 기능을 보였다. Viewer 기능은 객체의 상속관계와 클래스에 대한 구성정보를 보여주며, 단순한 부품의 재사용에 그치지 않고 클래스를 생성할 수 있는 기능을 추가하여 클래스 부품의 효율적 관리와 재사용성을 높이도록 하였다. 따라서 본 연구는 클래스 부품 정보, 상속관계, 계층도의 정보와 프로세스 등으로 분류하였고 추출 Viewer 기능을 통한 객체의 이해력을 높이고, 객체 생성을 위한 프로토타입을 지원한다.

Extraction and Comprehension of Objects for Class Components Reuse

Jung-Soo Han† · Young-Jae Song††

ABSTRACT

Class components in a repository require exact information representation by reason of insufficiency of various visual information through search and extraction. In this paper, we have described syntax-analysis method and viewer facilities about class components. In the components analysis, we extracted class information which consists of class_names, methods, attributes, class inheritance relationship, and graphic information. Viewer represents extracted class information and creates a new class. Also, it provides facilities of reuse, insertion, and deletion. Not only Viewer represents class hierarchy diagram, and shows detail information about each class, but also it creates new classes. In this paper, we implemented a Viewer using the components, inheritance, diagram information and process. And we enhanced understanding of programs through viewer, and supported prototype for class creation.

1. 서 론

객체 지향 방법론(Object-Oriented Methodology)의 영향으로 객체의 효율적 관리와 사용 방법에 관한 관심이 높아지고 있으며, OMT(Object Modeling Technique), OOA/OOD 등의 객체지향 방법론이 등장하였다[1]. 또한 이들의 단점을 보완한 UML(Unified Modeling Language) 등의 방법론이 제시됨에 따라 객체지향 소프트웨어를 개발하는데 필요한 많은 객체들이 요구되고 있다. 이러한 객체들을 효율적으로 저장, 관리, 검색, 재사용 등의 활용을 위한 주요 역할을 하는 것이 정보저장소(Repository)이다. 클래스 부품(Class Components)은 기존의 소프트웨어를 분석하여 재사용 가능한

† 정 회 원 : 경희대학교 전자계산공학과 박사과정
†† 종신회원 : 경희대학교 전자계산공학과 교수
논문접수 : 1998년 12월 29일, 심사완료 : 1999년 2월 6일

부품단위로 정보저장소에 저장한 후 사용된다[2,3].

기존의 연구는 객체의 재사용을 위한 부품검색 부분과 문서화 정보에 의존하고 실제 재사용에 대한 제한적 경험을 바탕으로 이루어져왔다. IBM's Repository Manager/MVS[3]은 명세 도메인(Specification Domain)에서 객체를 ER에 의한 추상화(abstractions)와 그 명세(specification)로 제한하고 있다. IBM의 분산데이타베이스인 DDM[4]은 클래스 라이브러리를 지원하나 DDM 자체기능의 라이브러리에 한정하고 있고, "재사용을 위한 처리기"[12] 역시 텍스트적인 방법론으로 구조적 방법론을 사용한 Team-work[8]과 비교하고 있다. 클래스 부품의 추출과정에 있어서는 클래스들의 계층관계를 나타내는 추출 Viewer 기능이 정보의 표현에 한정되어 있어서 부품들의 생성, 삭제, 삽입 기능이 없어서 그 효율성이 떨어지고 있다. 따라서 클래스 부품을 보다 효율적으로 저장하기 위한 객체의 추출방법과 객체지향적 재사용 방법론이 대두되고 있다[5,6,11,13].

이러한 문제점을 개선하기 위하여 클래스 부품의 분석 정보를 규칙적이고 효율적으로 저장하고, Viewer 기능을 이용한 재사용성을 향상시킬 수 있도록 클래스 부품의 다양한 정보를 제공하였다. 객체지향 재사용을 위한 정보저장소는 객체에 대한 정보를 클래스이름(class_name), 멤버함수(methods), 속성(attributes) 등으로 분류하여 추출하고, 각 클래스에 대한 상속관계정보로 구성하였다. 또한 이들 정보를 이용하여 객체지향 방식으로 클래스 계층도에 필요한 객체 사이의 상속관계 정보와 그래픽 정보를 추출하여 Viewer 기능을 통한 재사용성을 실험하였다. 또한 Viewer 기능은 재사용 부품의 클래스 계층도 뿐만 아니라 새로운 클래스를 직접 생성(create), 삽입(insert)할 수 있도록 하고 이때에 클래스 부품에 대한 혼합된 정보를 재구성할 수 있도록 구현하였다. 또한 사용자가 클래스의 정보를 쉽게 볼 수 있도록 하고 이들의 상속정보(inheritance information) 역시 계층도에 표현하여 부품의 이해와 계층도상에서 부품선택을 통한 재사용의 효율성을 높이도록 하였다.

본 연구는 재사용을 위한 객체를 선택하면 이와 관련된 정보를 보여줌으로써 재사용 부품에 필요한 상, 하위 객체 역시 자동으로 선택되도록 하여 재사용에 도움이 되도록 구현하였다. 정보저장소로부터 정보를 검색 추출하는 과정과 그 결과를 재사용하기 위한 그래픽 표현 방법의 기법을 기술하였다. 객체지향 다이어그램에서 표현되는 그래픽 의미를 정보저장소에 저장하여 이 저장소로부터 계층화가 이루어지고, 클래스 부품의 계층화를 통한 분석된 도메인 정보를 손쉽게 추가, 삭제하는 등의 작업으로 소프트웨어 재사용의 효율성을 보이도록 하였다. 정보저장소와 추출 Viewer는 Visual C++ 6.0 MFC Library와 Visual Foxpro 5.0을 이용하여 구현하였다.

2. 분석 정보

분석 정보는 그 응용에 따라 삭제, 추가, 상속, 실체화, 추출등의 동작을 통하여 재사용할 수 있다[7]. 관련 시스템에 대한 정보를 분석 저장하여 이를 바탕으로 재사용하고자 하는 클래스(class) 사이의 상속(inheritance) 관계를 다이어그램으로 표현이 가능하도록 설계하였다.

2.1 클래스 정보

클래스 정보는 클래스이름, 멤버함수, 속성, 수퍼클래스와 서브클래스의 관계로 구성되고 이 정보를 이용하여 각 클래스와의 상속관계를 중심으로 정보를 추출한다. 이를 위해서 클래스 부품을 구문 분석하여 클래스내의 모든 정보를 상호 관련정보로 구축한다. 또한 각 클래스들의 관계정보도 추출한다. 다음의 클래스 구조에서 *Diagram*은 *drawboard(), name(), height(), draw()*의 멤버함수와 *scale, x, y*의 속성을, 클래스 *TextBlock*은 2개의 멤버함수와 1개의 속성을 갖고 있다. 또한 두 클래스는 서로 상속관계에 있다. 이들에 대한 정보는 각 클래스 자체정보와 관계정보로 이루어진다. 여기에 다이어그램 정보인 상속관계(Rel), 위치(Location), 클래스(Box), 각 레벨의 클래스 개수(Class_Num) 등의 다이어그램 정보를 연결시켜 저장한다. 클래스 개수는 윈도우에 클래스에 대한 위치선정에 사용된다. 이때의 정보는 클래스 원시코드사이에 관계정보가 링크 된다.

예)
```
class Diagram
{
  int scale, x,y;
  DrawBoard();
name();
height();
draw();
}
```

```
classTextBlock:Diagram
 {
  int ChangeScale;
  Text();
  Property();
 }
```

- 클래스 정보

  (Diagram)

  →(DrawBoard,name,height,draw)

  →(scale,x,y)

  (TextBlock)

  →(Text,Property) →(ChangeScale)

  ....

- 클래스관계정보

  (Diagram) → (TextBlock) ....

- 다이어그램 정보

  Rel → Class_Num → Location → Box

  (상속관계) (클래스 번호) (위치정보) (클래스 표현)

클래스의 구문 분석은 프로그램 파일을 입력받아 먼저 토큰(token) 단위로 클래스 이름을 모두 추출한 다음 처음 클래스 명에서 다음 클래스 명이 나타날 때까지 비교하면서 멤버함수(methods)와 속성(attributes)들을 추출한다. 이와 같은 클래스 수만큼의 반복으로 클래스 정보를 추출한다. 이때 클래스 계층도를 위한 각 레벨마다의 클래스 개수가 카운트되어 클래스의 다이어그램 정보와 링크되어 계층도로 표현될 때 위치선정이 자동으로 정해질 수 있도록 하였다.

◆ **클래스 구문분석 알고리즘**

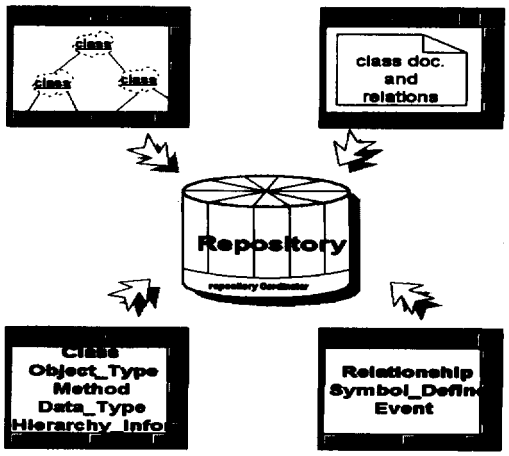
```
    class_count = 1;
do while
    if( "};" exist)
        endClass = location
    else endClass  Not exist
        exit
    endif
   if("class" exist)
        startClass = location
        mm = class_count
        mC&mm have strings from "class" to "};"
         /* &mm = 1로 변환 */
        class_count = class_count+1
    endif
end do
totalc=class_count  /* 전체 클래스 개수 */
/* mc1,mc2,..에 class 단위로 분류되어 저장 */
```

2.2 클래스와 다이어그램 정보

클래스정보와 클래스 사이의 관계정보가 (그림 1)과 같이 정보저장소에 저장된다. 이때 이들 정보는 객체지향 다이어그램 도구를 위한 정보로 변환시켜야한다. 이들 정보는 클래스 정보와 클래스 사이의 관계정보, 기호정보, 위치정보를 통합한다. 재사용 도구의 클래스 표현은 하나의 좌표공간을 선택하여 수퍼 클래스(super class)를 중심으로 상호 균등하게 표현되어야 하므로 클래스 계층도의 각 레벨단위마다 클래스의 개수 정보를 얻어야 한다. 따라서 저장소에 저장되는 정보는 클래스 추출 정보와 클래스 상속정보, 상속정보를 바탕으로 한 연결(line)정보가 있어야한다. 따라서 상속정보를 바탕으로 각 클래스는 박스(box)로 연결되어 그 안에 클래스 명(text)이 표시된다. 상속정보는 화살표로 연결된다. 또한 윈도우 상에서 클래스를 마우스로 클릭했을 때 클래스의 구성 정보와 소스 코드가 나타날 수 있도록 클래스 기호 안의 좌표를 인식시켜 이 좌표가 클래스의 선택을 가능하도록 하였다.

```
INFO :: Name + Contents + Box
    -> Rel + Class_Num +Location
```

정보의 표현은 객체를 박스(box)로 정의하고 화살표는 상속관계를 의미한다. 또한 다중상속은 한 개 이상의 화살표로 표시되며, 클래스 계층도에 표현되도록 하였다.



(그림 1) 클래스의 정보구조

(Fig. 1) Information Structure of Classes

◆ 정보 추출

클래스 정보추출은 스트링(예, CLASS), 클래스명, 레코드 번호(클래스 수, 부모클래스 번호, 각 레벨에서의 고유번호), 수퍼클래스, 레벨번호, 서브클래스, 멤버함수리스트, 변수리스트 순으로 추출한다. 또한 전체 클래스 코드를 나타내기 위한 클래스 단위의 코드 정보에도 링크되도록 하였다. 이는 클래스 수만큼의 반복적 추출 과정을 실행하여 정보를 얻는다. 다음의 정보 추출과정은 Visual Foxpro 5.0으로 구현하였다.

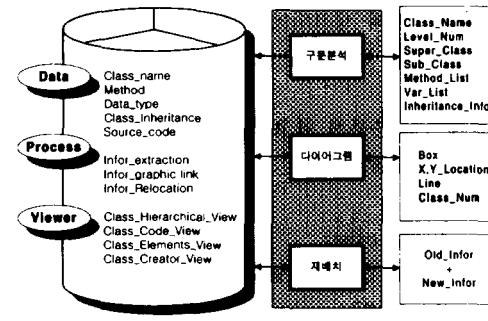
```
do while !eof()
  m=  "CLASS::"
    + classname   /* 클래스 이름 */
    + "{("
    + Class_Num  /* 클래스 번호 */
    + Super_Class_Num /* 수퍼클래스 번호 */
    + Level       /* 계층도 레벨 번호 */
    + Level_Num  /* 레벨에서의 클래스 번호 */
    + "),"
    + "SubClass("
    + SubClass_Names  /* 서브클래스 명 */
    + "),FunList("
    + Fun_Name      /* 멤버함수 명 */
    + "),VarList("
    + Var_Name      /* 변수 명 */
    + ")};"

classinfor have  m information from infor_table
enddo

/* shape 클래스의 관계 정보 예 */
/*
CLASS::shape{(1,0,0,0), SubClass(ball, rect, tr
ia, noshape, Wheel) ,FunList(set, shape, floodfill,
setcolot, fill, setcolor, init), VarList(int xCo,
int yCo, int LInecolor,int fillcolor)};
*/

// 소스코드 추출
do while ( source file != eof )
 write string "(CLASS(" , class_number, and ")"
    read class name
    read source code linked class name
  where
    class name links string
    from classname_string_num
    to  "};" string_num
enddo
```

3. 정보저장소

정보저장소는 도구들이 정보를 공유하면서 개발하고자 하는 시스템에 관한 정보를 생성하고 관리하는 기본적인 기능을 가지고 있다. 각 도구들은 분석 및 설계 정보를 획득하여 이를 분석하고 제어하며, 명세서 정보를 프로그램 코드나 문서로 변경시키는 기능을 수행한다. 정보저장소의 기능을 세분화하면 도구의 통합, 시스템 정보공유, 소프트웨어 재사용, 시스템 문서와 코드의 자동화 생성, 프로젝트 관리 및 제어 등의 기능을 갖고 있다[9,10].

3.1 정보저장소 구조

본 연구에서 설계한 정보저장소는 저장되는 각각의 클래스 정보를 (그림 2)와 같이 데이터(Data), 프로세스(Process), 뷰어(Viewer)의 3단계로 분류하여 구성하였다.



(그림 2) 정보저장소 구조
(Fig. 2) Repository Structure

데이터는 구문분석으로 추출된 객체의 클래스 단위로 이루어진다. 프로세스 수행 과정은 소스 코드를 읽어 각 클래스를 추출하여 클래스 사이의 관계를 계층적으로 표현해 준다. 이때 각 클래스는 자신이 가지고 있는 부품 정보(클래스명, 멤버함수, 속성, 원시코드)와 함께 추출된다. 사용자는 클래스 계층도에서 자신이 원하는 클래스를 삭제하고 추가하는 기능을 이용하여 새로운 계층도를 만들 수 있다. 계층도에서 사용자가 원할 경우 상위 클래스에 연결된 하위 클래스가 다른

상위 클래스와의 다중상속 관계가 아닌 경우에는 자동 삭제된다. 그 다음에 변형된 계층도를 새로운 이름으로 저장한 후에 다시 읽어들이면 새로운 도메인 정보가 생성되어 재사용을 가능케 하였다. 정보저장소는 크게 객체의 정보와 다이어그램 정보로 구성된다.

객체의 정보는 클래스, 멤버함수(method), 데이터형(data_type)으로 구성되어 구문분석 단계에서 DB 파일에 저장된다. 다이어그램 정보는 기호(symbol), 사건(event), 관계정보(relation_information), 위치(location), 클래스의 수(class_number) 등으로 구성된다. 다이어그램 정보를 이용하여 그래픽으로 표현할 때 나타나는 객체들은 객체 내부의 정보뿐만 아니라 객체들 사이의 상속관계 정보와 이의 표현을 위한 위치정보도 포함하고 있어야한다. 또한 정보저장소에서 상속관계의 정보를 가지고 이의 재사용을 위해 도식적으로 표현하기 위해서는 정보의 재배치가 이루어져야 하기 때문에 각 객체의 수(Obj_Num)에 관한 정보 역시 포함하고 있어야한다. 이들 정보를 바탕으로 새롭게 생성된 객체들을 다시 계층도로 표현해야 하기 때문이다. 이처럼 정보저장소는 구문분석을 통한 클래스 정보와 View를 위한 정보로 구성된다.

3.2 정보저장소 기능

정보저장소를 구성하는 부품정보를 위해 여러 개의 프로세서가 메인모듈로 이루어져 정보저장소 내의 정보를 처리하게 된다. 정보저장소 내에 사건(event)이 발생하게 되면 정보추출 전 정보저장소의 모든 컴포넌트를 초기화하는 작업이 필요하며, 정보저장소내의 모든 컴포넌트 초기화 작업이 이루어지면 구문분석 단계에서 정보추출 프로세스가 진행된다. 이때 부품정보에 필요한 클래스의 이름(Class_name), 멤버함수(Method), 속성(Attribute) 등과 같은 컴포넌트들이 정보저장소에 저장되기 위하여 추출된다. 추출과정은 클래스 구성의 기본요소인 콜론(:), 콤마(,) 브레이스({,}), 스트링, 공백으로 이루어지기 때문에 클래스 이름을 먼저 추출하는 과정이 필요하다. 클래스 이름을 모두 추출한 후 다시 각 클래스에 대한 브레이스({,}) 사이를 분석하여 멤버함수를 추출한다. 추출과정에서 클래스 계층도를 위한 정보가 필요하기 때문에 각 클래스마다 클래스 수(Class_Num), 레벨, 서브클래스, 멤버함수, 변수 리스트 등의 순서로 저장한다. 멤버함수는 클래스 내부 즉, 브레이스({,}) 안에서 괄호를 기준으로 공백을 제외한 문자열을 멤버함수로 간주한다.

```
CLASS::shape{(Class_Num, Level, Par_Num,
Level_Num), SubclassList, FunList, VarList}
```

◆ **클래스명 추출 알고리즘**

```
for class_count=1 to total_count
  mm= class_count
  a= location of ":" /* 수퍼클래스 */
  b= location of "," /* 다수의 수퍼클래스 */
  c= location of "{" /* 클래스명 경계 */
  ma=a   mb=b   mc=c
  store " " to  mSU1,mSU2
 //클래스명  추출
 if( superclass Not exist)
   do while
     delete space /*공백문자제거*/
     decrease mc counter (mc-1)
     calculate location of class_name
   enddo
   mmc = location of class_name
   do while
     jump characters /*공백문자통과*/
     mc is last character location
          of class_name
   enddo
     read classname from space
                  to   space or "{"
 else  /* superclass exist */
     do while
      delete space from ":"
      decrease mc counter (mc-1) mc
     enddo
     mmc = location of class_name
     do while
       jump characters from ":"
       mc is last character location
            of class_name
     enddo
     read classname from space
                  to   space or ":"
  end if
end for
```

◆ **멤버함수(method) 추출 알고리즘**

```
do while
```

```
if( "(" and ")" exist )
    startClass = location of "("
    mm = class_count
    sub_count = startClass-1
endif
do while
  read string from "(" in reverse
  if ( char = "[" or "]"  or " ~ " )
        sub_count = sub_count - 1
        continue
  endif
  if( string Not 0-9 or a-z)
      exit
  endif
      sub_count = sub_count - 1
enddo
 read string from space to space or "("
 where
   string is from sub_count+1
        to startClass - sub_count - 1

 delete space from string
if ( method exist one more )
   a method linked other methods with  ","
endif
  class_count = class_count+1
enddo
 m_fun=   superclass
        + "("              // 문자열
        + classname        //클래스명
        + "("              //문자열
        + method lists     //멤버함수
        + ")"              //문자열
```

3.3 프로세스

프로세스는 클래스의 구문분석과 View를 위한 정보를 처리하는 과정으로 크게 두 가지의 정보가 저장된다. 하나는 클래스의 고유정보이며, 다른 하나는 클래스간의 관계를 나타내는 relationship이다. 클래스의 고유정보는 클래스의 이름을 대표하며 각 클래스의 속성이 함께 저장된다. 그리고 각 클래스간의 상속(inheritance), 관계(relationship) 등에 관한 정보를 갖는 relation_information이 저장된다.

정보저장소 내에 저장된 class_relation에 의해 구성된 그래픽 정보로 각 정보(class)간의 관계를 알 수 있으며, 이로부터 저장소 내의 정보를 관리할 수 있게된다. 저장소에 새로운 정보가 수정되거나 추가, 삭제되는 경우 관계정보를 참조하여 변화하는 관계에 포함된 정보가 각 정보들의 관계에 따라 함께 수정될 수 있다. 이때 관계정보에 포함된 모든 정보가 완성될 때까지 반복된다. 다음은 프로그램을 분석하여 이들 클래스들 사이의 정보를 추출하여 정보저장소에 저장된 결과이다. 추출관계 정보는 객체의 검색결과로서 slot.inf에 저장된다. 추출된 클래스 사이의 관계정보 의미는 다음과 같다.

```
CLASS::shape{(1,0,0,0),SubClass.....
CLASS::tria{(4,1,1,2),SubClass.....
infor{TC(10), TLebel(1,5,4)};

//  추출 정보 의미
    Class_Num              : 4
    계층상의 Level         : 1
    Parent_Num             : 1
    각 Level에서의 고유번호 : 2
    TC                     : 전체 클래스 수 10
    TLevel      : 각 Level 상의 클래스 수(1,5,4)
```

(그림 3)의 계층도를 참조하면 총 3 계층(Level)으로 되어있고 각 Level 마다 1, 5, 4개(TLevel)의 클래스로 구성되어있고 전체 클래스 수는 10(TC)이다. *tria* 클래스는 전체 클래스 중에서 4번째이고 레벨은 0,1,2 중에서 1 Level이 된다. 그리고 부모(parents) 클래스는 *shape* 하나이므로 1의 정보를 갖고 1 레벨에서의 위치번호는 0,1,2,3,4 중에서 2의 정보를 갖는다. 그 결과 (4,1,1,2)의 정보가 추출된다. 그 다음에는 멤버함수 리스트와 변수리스트가 추출된다. 따라서 Viewer 기능은 이 정보를 바탕으로 box, line, text를 이용하여 클래스의 계층도를 표현하게 된다.

◆ 추출된 클래스 사이의 관계 정보(slot.inf)

```
CLASS::shape{(1,0,0,0),SubClass(ball, rect, tr
ia, noshape, Wheel),FunList(set, shape, floodfill,
setcolot, fill, setcolor, init), VarList(int
xCo,int yCo,int LInecolor,int fillcolor)};
CLASS::ball{(2,1,1,0),SubClass(Cherry,Grape)
,FunList(fill, ellipse, init, draw, set, set,
shape, ball), VarList()};
CLASS :: rect{(3,1,1,1),SubClass(Square),Fun
```

```
List(lineto, moveto, fill, rectangle, init, draw,
set, set, shape, rect), VarList());
CLASS :: tria((4,1,1,2),SubClass(Pyramid),
FunList(fill, polygon, init, draw, set, set,
shape, tria), VarList());
CLASS::noshape((5,1,1,3),SubClass(Cherry,Gra
pe, Square,Pyramid), FunList(rectangle, set
color, erase),VarList());
CLASS::Cherry((6,2,2,0),SubClass(),FunList
(draw , erase, draw, set, set, set, ball,
Cherry), VarList());
CLASS :: Grape((7,2,2,1),SubClass(), FunList
(draw,erase, draw, set, set, set, set, ball,
Grape),VarList());
CLASS :: Square((8,2,2,2),SubClass(),FunList
(draw, erase, draw, set, set, set, rect
,Square), VarList());
CLASS::Pyramid((9,2,2,3),SubClass(),FunList
(draw, erase, draw, set, set, set,tria,
Pyramid), VarList());
CLASS :: Wheel((10,1,1,4),SubClass(),
FunList(draw, set, set, set, set, set,
Wheel),VarList());
infor(TC(10), TLevel(1,5,4));
```

4. 클래스 부품 Viewer

지금까지 클래스 부품을 정보저장소에 저장하기 위하여 프로그램의 구문분석을 통한 정보 추출과정을 데이터와 프로세스로 나누어 설계하였다. 이 장에서는 정보저장소의 부품들을 시각화하기 위한 스키마 정보 구성 방법을 알아보고, Viewer 기능을 통하여 클래스 계층도, 클래스 정보 표현, 프로토타입을 이용한 객체 생성과정을 알아보고자 한다.

4.1 스키마 정보

객체지향 재사용 시스템은 스키마와 이와 관련된 정보를 하부구조에서 연결시킬 수 있도록 구현하였다. 이는 객체의 깊이와 너비를 계산한 후 각 테이블에 맞도록 객체를 표현한다. 재사용을 위한 정보를 정보저장소로부터 객체지향 다이어그램으로 표현하고, 객체를 선택하면 그 객체에 대한 정보가 나타난다. 또한 객체를 생성할 때에는 다른 도메인상의 객체를 복사하거나, 프로토타입 기능을 이용하여 상속관계, 멤버함수 타입, 변수 타입 등의 지원을 통한 새로운 객체를 생성할 수 있도록 구현하여 재사용 가능하게 하였다.

다이어그램 과정은 추출정보 결과로부터 최상위 클래스일 경우 중심좌표를 구하여 위치가 선정된다. 다음 레벨부터는 레벨단위의 클래스의 수만큼 순서적으로 위치가 선정된다. 위치를 선정한 후 각 클래스에 box가 그려지고 그 안에 클래스 명(text)이 표시된다. 또한 상위 클래스의 정보를 받아 상속(inheritance) 관계가 표시된다. 객체 선택은 마우스를 이용하여 각 클래스의 정보를 쉽게 이해할 수 있도록 하였다. 이를 위하여 각 클래스(box)의 위치(x,y좌표)를 기억하고 있어서 마우스를 이용하여 클래스를 더블클릭하면 선택된 클래스의 위치를 인식하여 소스 코드를 새로운 윈도우를 생성하여 볼 수 있도록 하였다. 추가적인 기능으로는 계층도 윈도우 왼쪽에 클래스에 대한 다중상속(Multiple inheritance) 관계를 표현하여 객체의 이해도를 높이도록 하였다. (그림 3)에서 다중상속 클래스는 "*"로 표시하였다.

◆ **객체의 다이어그램 알고리즘**

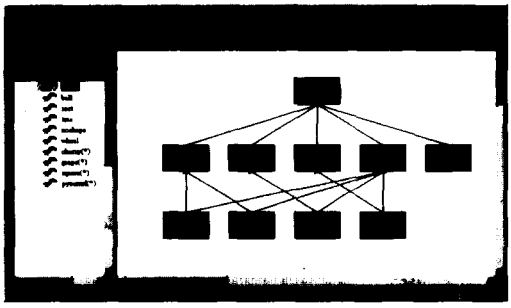
```
for all total class and each level class
  if(level_count exist and zero level
     a class is located at center
                 in table size
   /* root 인 경우 테이블의 중심좌표를 구함 */
   else from 1 level to end level
      classes are located from left side
                 in defined size
   endif
end for    //클래스 표현
for all total class count, add count+1
    drawing the box(class)
    write text(classname)
end for
int cnt=0; //상속관계 표현
for all total class count, add count+1
  if inheritance exist
   while(cnt< tablesize)
    if subClass exsit
    x,y position move to  /* 좌표이동*/
          center position of class box of top
    line between classes /클래스 상속관계*/
    line arrow /* 화살표 좌우 그리기 */
   end if
```

```
    end while
  end if
end for
```

◆ 객체 선택 코드

객체를 선택하면 그 객체에 대한 정보를 보여주는 기능을 한다.

```
//마우스 클릭 후 해당 클래스명 검색
for all total class count, add count+1
    if mouse point is in any class box
                (left, right, top, bottom )
       class_name of the box is loaded
    end if
    for all total class count, add count+1
      if class_name is included in class lists
            // 검색된 클래스정보 표현
          dialog_box loaded
          write source linked class_name
        end if
      end for
```

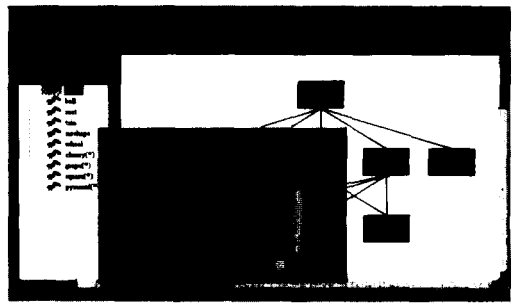


(그림 3) 클래스 뷰
(Fig. 3) Class View

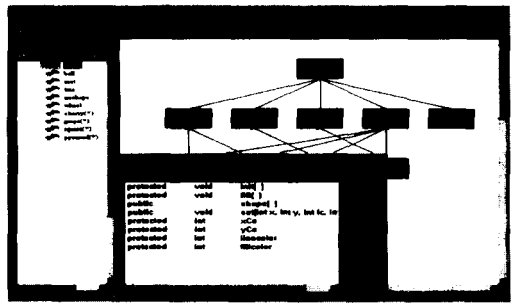
4.2 클래스 뷰(View)

(그림 3)은 정보저장소로부터 slot.inf 파일정보를 도식으로 표현해 주고있다. slot.inf 파일은 클래스들 각각의 정보와 클래스 사이의 상속정보, 도식화를 위한 그래픽 정보를 가지고 있다. 이때 클래스 계층도(class hierarchy diagram)는 클래스 사이의 상속관계 정보를 보여준다. 다중상속은 "*" 기호로 나타내어 상, 하위 계층의 하나 이상의 연결관계를 보다 이해하기 쉽도록 하였다. 계층도의 각 클래스 정보는 (그림 4)에서 처럼 클래스를 선택할 수 있도록 하여 클래스 내에 마우스를 클릭하면 즉, *shape* 클래스를 선택하면 선택된 클래스의 원시 코드가 나타난다. 또한 클래스의 구성 요소에 대한 정보를 보기 위해서는 클래스 계층도에서 객체를 선택하면 (그림 5)와 같이 *shape* 클래스의 Access Mode, Type, Declaration에 대한 정보를 표현해 준다. 이와 같이 클래스의 계층도에서 정보를 이용하여 클래스의 이해도를 높이고 재사용이 가능하도록 하였다.

```
INFO :: Name + Contents + Box
     -> Rel + Location + Class_Num
```



(그림 4) 클래스의 정보
(Fig. 4) Class Information



(그림 5) 클래스 구성요소
(Fig. 5) Class Elements

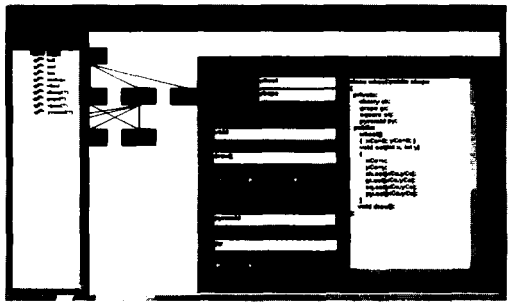
4.3 객체의 생성

Viewer는 재사용을 위한 객체의 정보표현 뿐만 아니라 직접 객체를 생성할 수 있는 기능을 가지고 있다. (그림 6)은 클래스 계층도에서 클래스 *wheel*의 생성과정을 보여준다. 객체를 생성하기 위해서는 클래스명(*wheel*)을 입력하고 *wheel*에 대한 부모클래스(base class)가 있으면, 여기서는 *shape*를 입력하면 된다. 멤버함수(function)에 대한 형식(type)을 입력하고 멤버함수명(*draw*)을 입력한다. 이때 클래스 생성에 도움을

주기 위하여 이 멤버함수가 *public, protected, private* 중에 어떤 엑세스 모드에 적용되는 지를 선택할 수 있도록 해준다. 또한 변수 입력 과정에서도 변수의 형(Variable Type)과 선언(Var_Declaration)란에 각각 *pyramid*와 *py*를 입력하고 엑세스 모드를 선택한다. 그 후 이의 반복적 수행과정을 거치면 오른쪽 부분에 클래스의 프로토타입이 형성되고 기타 코드는 텍스트 형식으로 입력하면 된다. 이와 같은 과정을 Viewer 기능에 포함시켜 정보를 입력함으로써 상속관계 정보를 얻어 클래스 계층도에서 쉽게 객체의 생성(Create)이 가능하도록 하였다.

여기서 재사용할 정보를 이동시키기 위해서는 선택된 객체와 관계(Rel)가 이동할 때에 그래픽 정보와 이의 표현을 위한 재배치 과정이 이루어진다. 이때의 정보는 이동될 곳의 정보와 연결을 시도한다. 이때 객체의 정보(INFO)와 재사용될 객체의 관계정보를 이용하여 기존의 객체와 연결이 될 수 있도록 객체의 수와 위치정보(Obj_Num, Location)가 이용되어 재배치 도식을 위한 정보가 생성된다.

```
Relocate :: INFO + Pre_Rel(Obj_Num)
            + Now_Rel(Obj_Num)
```



(그림 6) 클래스 생성
(Fig. 6) Class Creation

5. 비교 분석

클래스 부품의 정보분석과 추출 View 기능에 대한 기존 시스템들과의 비교가 부품성격, 부품의 구성항목, 재사용방법, Viewer 기능 등의 특성들을 기준으로 하여 <표 1>에 나타나 있다.

본 연구에서 정보저장소는 데이터, 프로세스, View -er로 구성되며 클래스 부품, 구성요소, 계층도를 보여준다. 재사용 가능한 객체를 분석하여 이들 클래스 정보와 클래스 사이의 관계 정보를 이용하여 정보저장소를 구성하였다. 정보저장소에서의 재사용 가능 부품을 다이어그램으로 표현함으로써 쉽게 이해하고 재사용의 활용을 높이도록 하였다. 또한 구성 정보를 이용하여 클래스의 생성이 가능하도록 하여 시스템 설계시 그 효율성을 높이도록 개발하였다.

〈표 1〉 기존의 시스템과 비교
〈Table 1〉 Comparison of existing systems

항목 / 시스템	부품 성격	구성 항목	재사용 방법	Viewer
Repository /MVS	함수	함수형 데이터형	함수 호출	추상화 명세화
DDM	클래스	클래스	제한적 라이브러리 제공	×
Teamwork	함수	함수형 데이터형	×	×
CARS	클래스	클래스	객체의 생성상속	클래스 계층구조
객체분석 도구[11]	클래스	클래스	객체의 생성상속	클래스 계층구조, 클래스 구성요소
재사용 처리기[12]	함수	함수형 데이터형	텍스트 명령어를 이용한 재사용	×
본 논문의 제시방법	클래스	클래스 상세정보	객체의 생성상속	클래스 계층구조, 클래스 구성요소, 소스코드, 프로토타입 생성

"재사용 처리기[12]"와 비교해 보면 재사용 처리기는 수작업을 통한 언어 처리기로 구성되어 있어서 프롬프트 상에서의 명령어를 통하여 재사용 부품을 생성한다. 또한 부품을 명세서로 국한시키기 때문에 실질적인 부품 재사용이라 할 수 없고, 뷰(view) 기능이 없어서 부품에 대한 정보의 이해성이 떨어진다. 또한 "객체지향 클래스 라이브러리 시스템[13]"은 클래스의 다중 상속에 대한 객체의 이해도가 부족하다. 따라서 본 연구는 클래스의 이해를 위한 클래스 계층도의 다중 상속 관계를 표현하였다. 또한 Viewer에 재사용 기능

을 추가함으로서 클래스 부품을 통한 도메인 상의 정보를 쉽게 표현할 수가 있으며, 클래스 계층도에서 필요한 클래스를 생성할 수 있도록 하여 재사용의 효율성을 향상시켰다. 또한 Teamwork[8]은 구조적 설계방법과 OOA 기능이 뛰어나지만 재사용성에 있어서는 구조적 분석기법을 통한 프로세스 참조 기능뿐이다. 따라서 재사용 측면에서는 기존의 시스템에 대한 구조적 설계 방법론을 이용하기 때문에 이해가 어렵고 검색기능 역시 결여되어 사용하기가 어렵다는 문제점을 안고 있다. 그리고 도메인 정보가 결여되어 재사용을 위한 정보 추출이 불가능하기 때문에 본 연구는 클래스의 재사용 도구와 정보추출 그리고 도메인 정보의 효율적 이용으로 인한 재사용성이 뛰어남을 알 수 있었다. 또한 Viewer 기능에 클래스의 다중상속(Multiple Inheritance) 관계를 클래스 사이에 하나 이상의 라인 기호로 나타내어 프로그램의 이해도를 높이도록 하고 클래스 계층도에 심볼 "*"를 이용하여 프로그램의 구조를 쉽게 이해할 수 있도록 하였다. 뿐만 아니라 클래스 부품을 구성하고 있는 원시 코드(Source Code), 클래스 구성(Access Mode, Type, Declaration) 부분을 표현함으로서 클래스 부품을 재사용하기 위한 이해도를 높였다. 클래스 생성(Create) 기능에서는 생성 과정에 프로토타입을 제공함으로서 상속관계 정보를 이용한 클래스 계층도를 쉽게 구성할 수 있도록 하였다.

6. 결 론

본 연구는 새로운 소프트웨어 요구시 이전의 유사한 도메인 분석을 통하여 클래스 부품 재사용을 위한 정보저장소를 데이터(data), 프로세스(process), 뷰(viewer)로 분류하고 프로그램 이해와 재사용을 위한 뷰(viewer)의 기능을 높이도록 구현하였다. 클래스 부품을 다이어그램으로 표현하기 위해 구문분석을 통한 추출과정을 클래스 계층구조, 부품구성요소, 코드표현, 클래스 생성이 가능하도록 하였다. 추출과정에서는 구문분석을 통한 클래스, 멤버함수, 변수 등의 추출과정을 클래스명을 중심으로 토큰단위로 멤버함수와 기본 변수 명을 추출하고 클래스 상속관계의 계층화 방법, 추출된 부품의 상세 정보를 통하여 보다 이해가 빠르고 추가, 삭제 기능을 이용한 재사용 과정을 보였다. 또한 다중상속관계를 보여주고 클래스의 원시 코드 뿐만 아니라 클래스 구성요소를 참조할 수 있도록 하였고, 클래스 생성과정에서는 프로토타입을 만들어서 클래스를 다이어그램 정보와 함께 생성하도록 하였다. 본 연구는 클래스 부품의 재사용을 위한 Viewer 기능을 개발함으로서 부품의 다양한 정보를 이용할 수 있었고 기존의 재사용 시스템과 비교하여 그 효율성을 증명하고 재사용 시스템 구축을 용이하도록 하였다.

그러나 본 연구의 구현과정에서 클래스 부품들의 다중상속에 대한 정확한 정보 이해의 부족으로 삭제과정에서 사용자가 스스로 판단해야하는 어려움이 있었다. 이의 해결을 위한 객체의 일반화(generalization)와 집단화(aggregation) 개념 도입이 요구된다. 또한 재사용을 위한 부품 검색방법이 필요하고 이를 위한 도메인 분석 과정이 요구된다. 앞으로의 연구 방향은 도메인 분석정보를 통한 재사용의 응용에 관한 연구와 CASE의 분석, 설계 단계에 연관성이 있도록 하고, 현재 도메인 정보의 검색방법이 연구되고 있다.

참 고 문 헌

[1] Rumbaugh, J. et al., "Object-Oriented Modeling and Design," Prentice-Hall, 1991.

[2] James Petro and Michael E. Fotta, "Model-Based Reused Repositories-Concepts and Experience," IEEE Computer Society Press-Technical Council on Software Eng., pp.60-69, 1995.

[3] J. M. Sagawa, "Repository Manager Technology," IBM System Journal, Vol.29, No.2, pp.209-227, 1990.

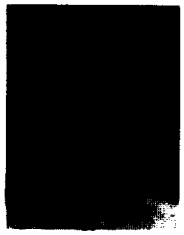
[4] R.A. Demers, J.D. Fisher, S.S. Gaitonde and R.R. Sanders, "Inside IBM's Distributed Data Managem ent architecture," IBM System Journal, Vol.31, No.3, pp.459-487, 1992.

[5] Udo Hahn, Mattthias Jarke, Thomas Rose, "Teamwork Support in a Knowledge-Based Information Systems Environment," IEEE Transactions on Software Engineering, pp.467-481, 1991.

[6] Karen E. Smith, Stanley B.Zdonik, "Intermedia : A Case Study of the Differences Between Relational and Object-Oriented Database Systems," OOPSLA 87, pp.452-465, 1987.

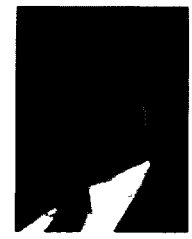
[7] Arango G., "Domain analsys methods, Software Reusability," Ellis Horwood, pp.26-37, 1994.

[8] Teamwork/OOA Release 5.0, CADRE Technologies Inc. July, 1993.
[9] R.Prieto-Diaz and P.Freeman, "Classifying Software for Reusability," IEEE Software, Vol.4, No.1, pp.6-16. Jan., 1987.
[10] Carma McClure, "CASE is Software Automation," Prentice-Hall, Inc., pp.53-69, 261-279, 1989.
[11] 김재생, 송영재, "재사용에 기반한 객체들의 정보분석과 자동화에 관한 연구," 한국정보처리 논문지, 제4권 2호, pp.384-394, Feb., 1997.
[12] 김지홍, 송영재, "도메인 분석 정보의 재사용을 위한 처리기의 설계 및 구현," 한국정보처리 논문지, 제2권 4호, pp.499-508, 1995.
[13] 정계동, 권오진, 최영근, "프로그램 이해 지원과 재사용을 위한 객체 지향 클래스 라이브러리 설계 및 구현," 한국정보처리 논문지, 제5권 제6호, pp.1507-1521, 1998.



한 정 수

e-mail : jshan@soft.kyunghee.ac.kr
1990년 경희대학교 전자계산공학과 (학사)
1992년 경희대학교 전자계산공학과 (석사)
1995년 경희대학교 전자계산공학과 박사과정 수료
1993년~현재 경희대학교 전자계산공학과 박사과정
관심분야 : 소프트웨어공학, S/W 재사용, CASE 도구



송 영 재

e-mail : yjsong@nms.kyunghee.ac.kr
1969년 인하대학교 전기공학과(공학사)
1976년 일본 Keio University 전산학과(공학석사)
1979년 명지대학교 대학원 졸업 (공학박사)
1971년~1973년 일본 Toyo Seiko 연구원
1982년~1983년 미국 Univ. of Maryland 전산학과 연구교수
1985년~1989년 IEEE Computer Society 한국지회부회장
1984년~1989년 경희대학교 전자계산소장
1976년~현재 경희대학교 전자계산공학과 교수
1993년~1995년 경희대학교 교무처장
1996년~1998년 경희대학교 공과대학장
1998년~현재 경희대학교 기획조정실장
관심분야 : 소프트웨어공학, OOP/S, CASE 도구, S/W 개발도구론, S/W 재사용