재사용 소프트웨어 컴포넌트의 합성과 릴레이션쉽에 관한 연구

김 치 수†

요 약

소프트웨어 개발시에 소프트웨어 개발팀의 생산성을 향상시키기 위한 방법 중의 하나가 한번 사용한 소프트웨어 컴포넌트를 재사용하는 것이다. 이러한 재사용할 수 있는 소프트웨어는 종종 소프트웨어 라이브러리로써 구성된다. 라이브러리로부터 선택된 소프트웨어 모듈을 효과적으로 재사용하기 위해서 사용자가 쉽게 찾을 수 있는 충분한 정보를 필요로 한다.

본 논문에서는 소프트웨어 컴포넌트 재사용을 위해 클래스간의 릴레이션쉽 정보를 제공하고 사용자가 원하는 컴포넌트가 라이브러리에 없는 경우 클래스에서 원하는 데이터와 함수만을 가져와 합성할 수 있는 툴을 설계 및 구현하였다.

A study on the Composition and Relationship of Reusuable Software Components

Chi Su Kim†

ABSTRACT

One of different methods to increase the productivity of software development team is to reuse the software components which were used once. Such a reusable software components are often organized software libraries. In order to reuse effectively the software modules selected from a library, the users need information enough to search for the modules easily.

This paper designs and implements the tool which provides the information about the relationship between classes to reuse software components, and makes a new class by combining the data and functions from existing classes.

1. 서 론

소프트웨어 공학의 중대 관심사인 소프트웨어 품질 향상과 생산성 증대를 위한 방법의 하나로 재사용 소프트웨어를 이용한 프로그래밍 방법이 연구되어 왔다. 기존에 개발된 소프트웨어 사이에는 40-60%가 유사하다는 것이 밝혀졌으며[1], 따라서 유사한 소프트웨어를 재사용한다면 50-80%의 생산성 향상을 기대할 수 있음이 조사되었다[2]. 이러한 연구는 재사용 소프트웨어에 대한 연구의 필연적인 근거를 제공하였으며 특히 추상자료형의 사용은 특정분야에 국한

※이 논문은 1994년도 한국학술진흥재단의 공모과제 연구비에 의하여 연구.

† 정 회 원:공주대학교 전자계산학과

논문접수:1995년 10월 30일, 심사완료:1996년 6월 5일

된 사용을 배제하고 구현상의 복잡성을 줄이는데 결정적인 역할을 하여 왔다.

따라서 이미 만들어진 소프트웨어를 재사용한다면 개발 기간을 단축시킬 수 있고, 프로그래머의 노력을 감소시킬 수 있으며, 잘 실행되고 있는 소프트웨어를 재사용함으로써 신뢰성을 보장받을 수 있다. 그 결과 테스팅의 노력이 감소되고, 유지보수가 용이해지는 등의 여러 잇점을 얻을 수 있다[3, 4]. 그러나 재사용 방법이 원하는 수준까지의 생산성을 향상시키기 위해서는 먼저 해결되어야 할 과정들이 있다. 그것은 빈도수가 높게 발생하는 추상화를 식별해야 하고 그러한 추상화를 캡슐화해야 한다. 그리고 재사용을 위한 툴들이 개발되어야 한다. 이러한 문제를 해결하기 위해 초기의 시도는 객체지향 라이브러리가 아닌 일반적인 프로시듀어와 함수를 포함하는 소프트웨어 라이브러리를 만드는 것이었다. 그러나 최근에는 객체지향 방법이 많이 연구되면서 객체지향 라이브러리 컴포넌트가 더욱 더 독창적이고 재사용하기가 쉽다는 것을 말해주고 있다[5].

재사용 소프트웨어 라이브러리가 아닌 일반적인 소프트웨어 컴포넌트로 이루어진 라이브러리가 성공하지 못하는 이유 중의 하나는 그들이 단지 거대하고 컴포넌트간의 관련성이 내포되지 않은 함수와 프로시듀어라는 점이다. 사용자가 소프트웨어를 쉽게 재사용하기 위해서는 컴포넌트간의 관련성이 잘 나타나야 한다. 그리고 유사한 소프트웨어로부터 필요한 기능을 가져와 합성하여 사용할 수 있도록 제공해 주어야 한다.

기존의 언어 디자인과 소프트웨어를 개발하는 데는 함수나 프로시듀어의 관점이 주로 사용되어왔다. 따라서 큰 시스템을 여러 부분으로 나눌 때 함수 관점을 토대로 한 방법이 주로 사용되어 왔다. 이것은 큰 시스템이 프로시듀어나 함수의 집합으로 구성됨을 알 수 있다. 그러나 Grady와 Booch는 또 다른 관점으로 객체와 action의 개념을 사용한 모델을 제안하였다[5]. 소프트웨어에서 객체에 관한 정보는 자료구조 혹은 자료 추상화에 의해 표현된다. 이러한 관점을 사용해 개발된 소프트웨어 컴포넌트의 재사용을 지원해주기 위해 라이브러리는 단지 프로시듀어 집합이 아닌 자료 추상화 또는 객체의 집합으로 구성된다.

본 논문에서의 라이브러리 컴포넌트는 함수나 프로시듀어가 아닌 추상자료형을 사용한다. C++의 클래스 메카니즘은 재사용을 위한 추상자료형을 잘 표현할 수 있으며 자료 추상화와 자료구조 은폐 등의 기능을 제공함으로써 언어의 선택에 있어서는 객체지향언어의 C++를 사용한다.

본 논문에서는 사용자가 원하는 컴포넌트를 찾는 과정을 돕기위해 클래스가 재사용 컴포넌트로 존재할 수 있도록 클래스에 관한 정보와 클래스간의 릴레이션쉽을 제안하였으며, 존재하지 않을 경우 유사한 컴포넌트를 선택해 그곳에서 필요한 데이터와 함수를 가져와 합성하여 새로운 클래스를 생성하는 시스템을 설계 및 구현하였다.

2. 상속 클래스의 합성

본 장에서는 클래스의 합성과 사용자에게 합성의 판단에 도움을 줄 수 있는 상속 다이어그램과 클래스 합성시의 정보 부재로 인한 합성 오류를 최소화하고 사용자가 편리하게 합성할 수 있도록 합성 도구 및 합성 함수를 제안하고 구현하였다.

2.1 컴포넌트 합성의 관련분야 연구현황

컴포넌트의 합성에 대해서는 프로그래밍 언어론적인 접근과 데이터 베이스 측면에서의 접근이 있을 수 있다.

Goguen[6]은 제너릭 기능을 절차적으로 강조하는 파라메타화 프로그래밍 기법을 도입한 OBJ 언어를 통하여 여러가지 합성 방법을 제시하고 있다. 소프트웨어 재사용을 지원하기 위한 OBJ언어는 Theories와 View, Module Expression 이라는 세가지 부분으로 구성되어 있다. Theories 부분에서는 실제변수(actual parameter)의 대체를 위한 형식변수(fomal parameter)가 가져와야 하는 여러가지 특성에 대해 정의하고 있다. View는 Theories에서 정의된 일반적인 모듈이 응용영역의 특정한 상황에 맞도록 인스탄스화시키는 부분으로써 디폴트 기능을 함께 제공해주고 있다. Module Expression 부분에서는 여러가지의 함수집합에 의해 기존 모듈을 변형, 합성하는 일을 수행하며 여기에서 Goguen은 HIDE, VISIBLE, ENRICH 와 같은 세가지 합성 함수를 제시하고 있다. 이 방법은 OBJ언어가 컴파일러 형식으로 구성되어 있으므로 대화 환경에서 프로

그램을 개발할 수 없다는 단점을 가지고 있다.

또 다른 접근방법으로 Ketabchi와 Berzins[7]는 CAD 용 데이터 베이스의 오브젝트 합성에 관심을 두고 연구했으며, 복합 오브젝트에 대한 컴퍼넌트의 분할(partitioning), 응집(aggregation), 정제(refinement) 방법을 대수적 연산(algebric operation)에 근거하여 정의하였으며, 이들 연산에 의한 컴퍼넌트 합성을 연구하였다.

본 연구에서는 오브젝트를 데이터 베이스의 관점에서 보아 단순히 데이터의 계층구조로 이해하여 합성을 연구한 Ketabchi와 Berzins의 방법보다는 처음부터 데이터 및 프로시듀어 자체를 오브젝트로 간주하여 파라메터화를 통해 오브젝트 합성을 연구한 Goguen의 방법에 그 기초를 두었다.

2.2 클래스 합성의 정의와 필요성[8]

클래스 재사용의 형태는 세가지로 나누어 볼 수 있다.

첫째는 재사용 라이브러리에 저장된 클래스를 그대로 가져와 사용하는 것이다. 그러나 사용자가 원하는 클래스가 존재하지 않을 경우에는 설계과정에서 모듈의 세분이 더 가능한가를 보고 그렇지 않을 경우에는 다른 클래스에서 기능을 파생시켜 작성할 수 있는가를 타진한다. 두번째로 상위클래스로부터 그 특성을 물려받아 상속을 하여 사용하는 것이다. 이경우 클래스는 응용환경에 맞도록 매개변수를 통한 변형이 필요하다. 이것은 제너릭 메카니즘으로 가능하나 C++에서는 이 메카니즘이 제공되지 못하고 있다.

따라서 기본적인 매개변수 전달방법인 값의 호출(call by value) 대신 포인터 변수를 사용하는 주소호출(call by address) 방식의 제너릭 방법을 사용할 수 있다.

제너릭 방법을 사용하는 예를 살펴보면 다음과 같다.

(그림 1)의 클래스 리스트는 문자열에 대한 클래스를 선언한 것이며 이를 기본 클래스로 하여 기능을 상속받는 (그림 2)의 int_stack은 정변수를 도메인 구현영역으로 하고 있다.

이때에는 기본클래스인 리스트로부터 인스턴스 파라미터의 포인터를 물려받아 변수의 형태 변화없이 재사용 가능하게 한다.

그러나 제너릭 타입을 이용하더라도 원하는 클래스로서 부적합할 경우에는 세번째로 다른 클래스에서 필요한 부분을 가져와 재구성하여 사용하는 것이다. 아무리 정교한 ADT 개념으로 구현되었고 또한

```
class list
{
private :
      node* head ;
      int size ;
public :
      list ( int s ) { head=0 ; size=s ;}
      void insert ( char* a ) ;
      void append ( char* a) ;
      char* get ( ) ;
      void clear ( ) ;
      list ( ) { clear ( ) ; }
}
```

(그림 1) 문자열에 대한 리스트
(Fig. 1) List of string

```
class int_stack : list
{
public :
  void push (int a ) { list :: insert ( (char* )& a ) ; }
  int pop ( ) { return* ( ( int* ) list :: get ( ) ); }
  int_stack ( ) : ( size_of ( int ) ) { }
 ) ;
```

(그림 2) int_stack
(Fig. 2) int_stack

완전한 모듈 테스트를 거친 컴포넌트라 할지라도 이들이 사용될 수 있는 응용분야의 다양성으로 인하여 ADT의 일반성이 응용영역의 특수성을 모두 흡수할 수 없다는 점이며, 이러한 이유로 인하여 전체목적 코드의 40-60% 만을 재사용할 수 밖에 없다는 재사용 기법의 한계에 도달한다.

따라서 본 논문에서의 클래스 합성이란 세번째의 경우처럼 다른 컴퍼넌트들로부터 필요한 기능등을 가져와 재구성하여 사용하는 것을 말한다.

2.3 상속 다이어그램[9]

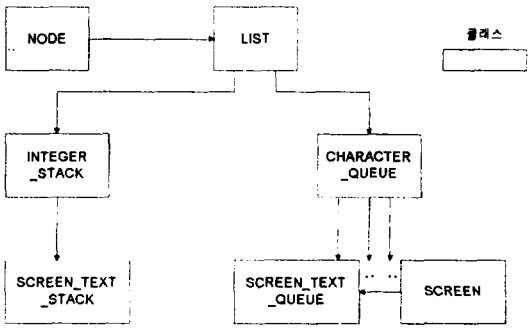
재사용 클래스의 합성을 위해 클래스 라이브러리는 가능한한 C++의 공개부 정보를 자료 추상화와 정보은닉을 준수하는 범위 내에서 제공해 주어야 한다. 이러한 정보 제공의 방법으로 상속 다이어그램을 이용할 수 있다.

상속 다이어그램이란 클래스간의 종속관계와 함께 클래스들 사이에 전달될 수 있는 데이터 및 멤버함수들에 대한 정보를 한눈에 알아볼 수 있도록 구성된 다이어그램이다.

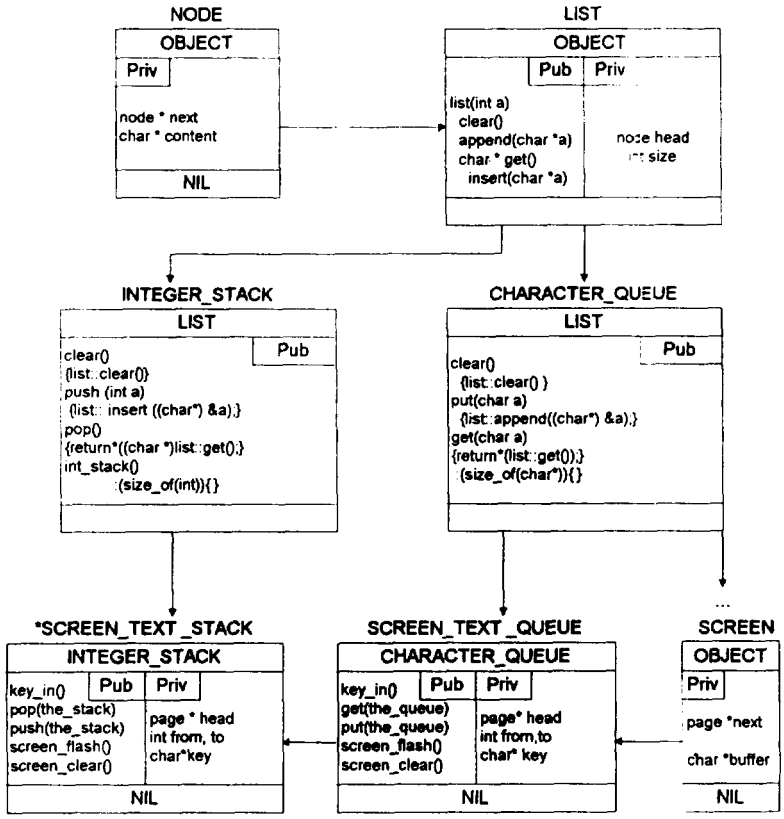
각 상속다이어그램은 상위 클래스와 파생클래스들에 대한 포인터를 가지고 있으며, 클래스의 공개부에서 제공하는 멤버함수들과 비공개부의 데이터구조를 사용자에게 보여줌으로써 클래스들간의 상속 정보를 가능한 한 자세하게 알 수 있도록 해준다.

그 적용예로서 (그림 3)은 다중화면을 이용한 문장편집기의 다이어그램 일부로서 SCREEN_TEXT_QUEUE 클래스는 최초의 화면으로부터 현재의 차례대로 볼 수 있게 하며 SCREEN_TEXT_STACK은 그 반대의 차례로 화면을 보여주는 클래스이다.

(그림 3)에서 보여주듯이 LIST 를 기본클래스로 하여 파생클래스 INTEGER_STACK과 CHARACTER_QUEUE를 구성하였고 또한 CHARACTER_QUEUE를 기본클래스로 하여 SCREEN_TEXT_QUEUE를



(그림 3) 다중 편집기의 클래스다이어그램
(Fig. 3) Classdiagram of multi_editor



(그림 4) 다중 스크린 편집기의 상속다이어그램
(Fig. 4) Diagram of multi_Screen Editor

구성하였다.

SCREEN_TEXT_STACK을 구성함에 있어 재사용될 적합한 클래스가 없으므로 기존의 클래스를 중심으로 합성할 경우, 키보드로부터의 명령어 입력 부분은 SCREEN_TEXT_QUEUE로부터, 다중화면의 스택처리는 INTEGER_STACK 클래스로부터 멤버함수의 상속을 받으려 한다. 만일 클래스 다이어그램에만 의존할 경우 사용자는 그러한 복합상속의 가능성만을 SCREN_TEXT_QUEUE의 서술자와 INTEGER_STACK의 서술자로부터 확인받을 수 있을 뿐이며, 그러한 합성의 타당성을 판단할 상세한 정보는 ADT 구현집합을 살펴보기 전에는 알기가 힘들다.

사용자에게 상속에 대한 좀더 자세한 정보를 주기 위한 방법으로 상속 다이어그램을 이용하여 (그림 3)의 문장 편집기 상속관계를 나타내면 (그림 4)와 같다. (그림 4)는 클래스 합성함수를 적용하여 기본클래스 LIST로 부터 매개변수화하여 생성된 파생클래스 INTEGER_STACK과 CHARACTER_QUEUE의 생성과정과 클래스합성의 예로서 클래스 SCREEN_TEXT_QUEUE와 INTEGER_STACK을 합성하여 합성클래스 SCREEN_TEXT_STACK을 구성하는 과정을 보인 것이다. 상속 다이어그램을 통하여 ADT-서술자가 제공해주는 클래스의 정보보다 더 자세한 정보를 얻을 수 있다.

예를들어 클래스 CHARACTER_QUEUE의 멤버함수 put과 get은 각각 기본 클래스 List의 멤버함수인 append와 get으로부터 상속되어 생겨난 함수이며 이들은 다시 파생클래스 SCREEN_TEXT_QUEUE의 멤버함수 put과 get을 형성시킨다는 것을 알 수 있다.

본 논문에서의 합성은 두가지 방법으로 할 수 있다.

첫째는 주메뉴에서 새로운 클래스를 생성하고 그곳에 다른 클래스에서 데이타 및 함수를 드래그하여 원하는 곳으로 가져온다.

두번째는 본 논문에서 제시하는 Public, Private, Add, Delete 등의 합성함수를 이용하여 합성을 할 수 있다.

3. 부품 정보와 릴레이션쉽

3.1 관련 연구

이절에서는 컴포넌트에 대한 정보와 릴레이션쉽을 나타내는 기존 연구된 대표적인 시스템의 특징을 알아본다.

3.1.1 C Information Abstraction System[10]

CIA는 C 프로그램으로부터 객체 및 객체간의 관계에 관한 정보를 데이터 베이스에 저장하여 C 프로그램의 구조를 분석하는 시스템이다. 이 시스템은 대규모 프로그램의 구조를 분석하는데 매우 유용하나 교차 참조, 제어흐름 및 자료흐름에 대한 정보 제공은 미흡하다. 또한 객체의 정보를 알고자 할 때 객체의 형을 명시해야 하는 불편한 점도 있다.

3.1.2 MasterScope[11]

MasterScope는 함수 정의를 분석하고 함수들의 호출에 대한 트리구조를 인쇄하는 PrintStructure라는 프로그램으로부터 시작하여, Interlisp환경하에서 사용자의 프로그램을 분석하고 교차 참조의 이해를 돕는데 사용된다. 이 시스템은 Interlisp에서 파일과 편집 패키지를 가지고 있으며, 프로그램의 변경시 재 분석을 하여 보관된 정보가 항상 일치하도록 하고 있다.

3.1.3 Cscope[12]

Cscope C 심볼의 사용도를 조사하고 함수와 매크로 정의를 찾고 수정할 수 있다. Cscope는 상호 참조 파일을 만들고 각 심볼에 참고 정보를 저장한다.

3.1.4 PIE브라우저[13]

PIE브라우저는 클래스의 상속성과 클래스간의 메시지 전송관계를 보여주고, 텍스트 자료를 클래스에 연관시켜줄 수 있는 기능이 있다. 이런 종류의 브라우저는 클래스의 모임이 아주 작을 경우에 주로 시험되었으며, 클래스의 크기가 커질 경우에는 그 유용성이 아직은 의문시되고 있다.

이들 관련 연구들을 참조하여 본 논문에서 개발된 툴은 다음과 같은 특징을 가지고 있다.

1) 호출구조 및 시스템과 클래스의 구성 명시

대상 프로그램의 분석을 통해 클래스들을 분류하고 이를 상속에 근거하여 호출구조를 표시함 으로써 각 시스템 단위의 클래스들에 대한 구성도를 나타내었다.

2) 시스템의 내부구조와 코드의 이해성 증대

각 클래스에 속하는 인스턴스들의 코드를 볼 수 있도록 하여 클래스의 기능을 파악하게 하 고 또한 시스템을 손쉽게 이해할 수 있도록 하였다.

3) 적절한 그래픽 및 텍스트 이용

분리 저장된 소스프로그램은 사용자의 요구에 따라 일목요연하게 제공되는 것이 매우 중요하 다. 사용자의 편리성 및 이해성을 높이기 위해 적절한 그래픽과 텍스트를 사용하여 출력을 가시화 하였다.

4) 재사용성 증대

시스템 내부에 감추어져 있는 클래스와 인스턴스 그리고 상호관계를 파악함으로써 부적절한 클래스와 상호관계를 바로잡을 수 있다.

3.2 C++의 이해 기법[14]

3.2.1 C++의 이해에 대한 문제점

프로그램에 대한 정보를 얻는 방법은 정적인 방법으로써 프로그램의 소스코드를 직접 읽거나 프로그램과 관련된 문서를 읽는 방법이 있다. 또한 프로그램을 직접 수행시켜 실행 순서를 탐색하거나, 자료의 경로를 추적하고 동적 기억 공간을 조사할 수 있는 동적인 방법이 있다.

위 방법 중에서 문서는 완벽하게 갖추어져 있을 수도 있지만 빠지거나 혹은 불완전하고 최근의 바뀐내용이 변경이 안된 경우 부정확할 수 있다. 또 프로그램을 실행해 봄으로써 프로그램의 동적인 정보를 살펴보는 것은 매우 유용할 수 있으며, 단지 원시 코드를 읽는 것으로부터 얻을 수 없는 프로그램의 특성을 알 수 있는 장점이 있다.

그러나 원시코드는 프로그램의 이해를 돕는 가장 기본적이고 중요한 단서이다. 종래의 구조적 프로그램을 이해하는 기본 단위는 모듈이나 함수가 된다. 그러나 C++와 같은 객체지향 프로그램에 적용되었을 때 프로그램을 이해하는 기본단위는 객체 혹은 클래스가 된다. 특히 객체지향프로그램의 기본단위인 객체와 클래스외에 상속성, 다형성, 동적 바인딩 등과 같은 개념에 대한 이해의 지원이 필요하다.

C++나 Smalltalk와 같은 객체지향 프로그램언어는 그 속성상 상속성을 제공하기 때문에 프로그램의 복잡도가 커져서 프로그램을 이해하기가 대단히 어려워진다. 보통 상속의 수준이 2~3일 때가 적당하며, 수준이 그 이상으로 늘어나면 프로그램을 이해한다는 것은 불가능하다고 알려져 있다. 또한 가상 함수와 오버로딩의 허용은 함수 호출과 흐름에 대한 분석을 대단히 어렵게 만든다.

3.2.2 C++의 이해 방법

객체지향 프로그램의 이해를 지원하는 방법은 다음과 같은 기본적인 사항이 필요하다.

① 시스템 구조 분석 : 원시 프로그램의 분석을 통하여 시스템의 구성을 보여주고, 상속성과 다형성에 근거를 둔 클래스의 분류 및 구조를 나타낼 수 있어야 한다.

② 관계 분석 : 클래스의 상속 구조 그래프, 화일 관계 그래프, 멤버 함수를 사용하는 객체에 대한 정보 등을 분석하고 나타낼 수 있어야 한다.

③ 정보 저장소 : 원시 프로그램이나 문서의 분석으로 얻어진 정보들은 관계형 데이터베이스에 적절히 저장하여, 사용자의 필요에 의해서 질의어를 통하여 간편하게 제공될 수 있어야 한다.

④ 사용자 편의성 : 사용자가 필요로 하는 정보들은 적절한 그래픽과 텍스트의 형태로 알아보기쉽게 표현될 수 있어야 한다.

⑤ 확장성 : 원시 프로그램의 이해를 통하여 얻어진 결과들은 그것에 그치지 않고 재사용이나 재공학(reengineering), 재구성(restructuring) 등에 효율적으로 적용될 수 있어야 한다.

프로그램 이해의 기본단위인 클래스는 다음 세가지 특성으로 분리되어질 수 있다. 이것은 프로그램 이해를 위한 정보 수집의 기본 단위가 될 수 있어야 한다.

① 클래스 : 모든 문서와 원시 프로그램 내부에 있는 클래스의 이름을 수집하고, 클래스의 종류를 구분한다.

② 클래스의 속성 : 각 클래스는 속성을 갖고 있으며, 클래스가 속하는 화일, 이름, 클래스의 위치, 역할 등이 있다.

③ 클래스간의 관계 : 각 클래스들끼리는 서로 상관관계에 놓여 있을 수 있다. 이런 관계에서의 예를들면, 상속성이나 메세지 교환에 의한 클라이언트 서버(client-server) 관계 등이 있다.

상속성과 다형성, 그리고 동적 바인딩은 객체지향 프로그램의 이해에 있어서 상당한 난제이다. 일반적

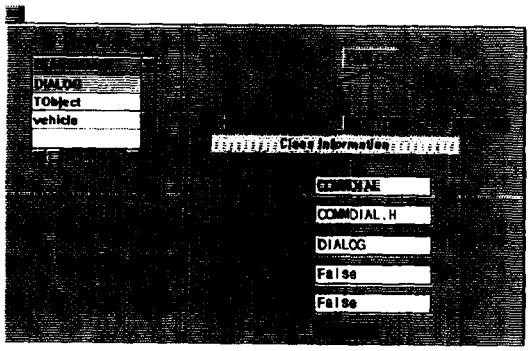
으로 지금까지 진행되어 온 연구에 의하면[15], 상속성을 이해하기 위해서 클래스 브라우저(class browser)를 사용하여 이해를 돕고 있으며, 다형성의 경우에는 외부 종속 그래프(external dependency graph)를 이용하여 해결하고 있다. 동적 바인딩의 경우에는 모든 가능한 경우를 탐색하거나, 사용자가 개입하여 해결하는 방법 등이 있다.

3.3 컴포넌트 정보

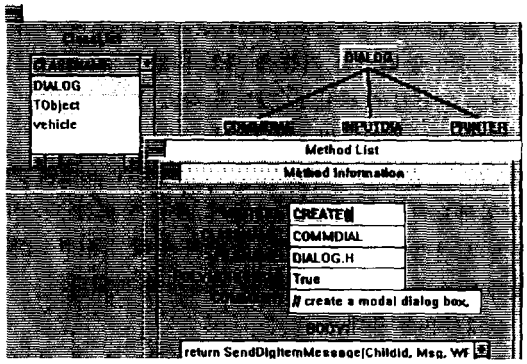
사용자가 원하는 컴포넌트를 찾는데 도움을 줄 수 있는 컴포넌트에 관한 정보는 본 논문에서 다음처럼 요약할 수 있다.

클래스 정보	화일 정보	Method 정보	Member 정보
·클래스명 ·화일명 ·base class ·Derived class ·abstraction class ·friend class ·key word ·서술부 ·Depth	·화일명 ·Header file ·설명부	·Method ·classname ·filename ·polymorphism ·friend method ·comment ·code	·classname ·member data ·type ·comment

다음은 그 실행 예를 보인 것이다.



(그림 5) 클래스에 관한 정보
(Fig. 5) Information of class



(그림 6) 메소드에 관한 정보
(Fig. 6) Information of method

3.4 릴레이션쉽[16]

본문에서 사용되는 릴레이션쉽은 다음과 같다.

3.4.1. Classification (a, i)

Classification은 Group i에 대해 서술부에 공통적인 특성을 정의해 놓고 만일 ADT a가 Group i와 연관된 서술부를 만족한다면, ADT a는 Group i의 멤버가 된다. 여기서 서술부란 Group에 대한 공통적인 성질을 정의해주는 부분이다.

트리에 대한 서술부는 다음처럼 정의할 수 있다.

· 트리는 하나 이상의 노드인 유한 집합이다.
· 특별히 루트라고 하는 노드가 존재한다.
· 나머지 노드들은 Disjoint Set T1, T2, ... Tn (n)=0)으로 분할(Partition) 되었다.

예를 들면, 다음과 같다.

Classification (Binary_tree, Tree)
Classification (B_tree, Tree)
Classification (Threaded_binary_tree, Tree)
Classification (KJU_student, Person)
Classification (KJU_faculty, Person)
Classification (KJU_policeman, Person)

3.4.2 Generalization (a, b)

ADT a 도메인이 ADT b 도메인의 부분집합이라면 ADT a는 ADT b의 Generalization이라 한다. 예를들

어, Person의 도메인이 "(name, sex, birthdate)"이고 ADT KJU_student의 도메인이 "(name, sex, birthdate, address, age, deathdate)"라면 Person은 KJU_student의 Generalization이 된다.

Generalization (Person, KJU_student)

3.4.3 Specialization (a, b)

Specialization은 Generalization의 역이다. 따라서 Generalization (b, a)가 성립하면, ADT a는 b의 Specialization이라 한다.

3.4.4 Generic_of (a, b)

ADT b가 ADT a의 인스턴스 매개변수에 대해 하나 이상의 값을 대치함으로써 a 로부터 얻을 수 있다면 ADT a는 ADT b의 Generic_of라 한다.

C++에서는 제너릭 메카니즘이 없지만 기본적인 매개변수를 사용하는 주소 호출 방식을 사용해 제너릭의 기능을 수행할 수 있다.

Generic_ of (list, int_stack)

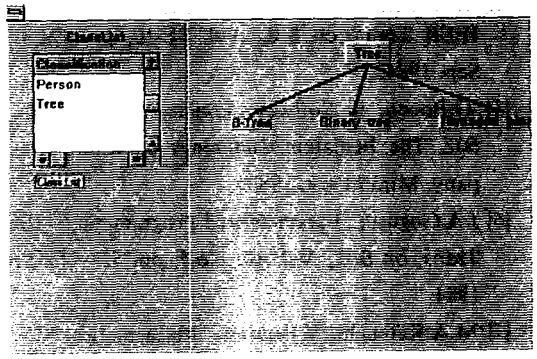
3.4.5 Uses (a, b, i)

만일 ADT a의 i번째 구현집합이 ADT b를 참조(Reference)한다면 ADT a는 ADT b를 Uses 한다고 한다. 만일 다음의 조건에 대해 하나 혹은 둘다 만족한다면 ADT a의 구현 집합은 ADT b를 참조한다.

- ADT a의 오퍼레이션의 구현집합이 b의 오퍼레이션 중 하나를 호출 (invoke) 한다.
- ADT a의 도메인이 ADT b도메인의 Term으로 정의되었다. 또는 임의의 도메인 표현과 관련된 최소한의 하나의 매개변수, 변수, 타입, 상수 혹은 ADT a의 함수 구현집합이 ADT b의 도메인의 Term으로 정의되었다.

예를 들면, ADT "person"이 ADT "date"에 의해 birthdate, marriage_date, death_date를 정의한다고 가정했을 때 다음과 같다.

Uses (Person, date, #1)



(그림 7) Classification의 예
(Fig. 7) example of Classification

4. 결 론

본 논문에서는 C++의 클래스를 중심으로 사용자가 라이브러리에서 원하는 컴포넌트를 검색할 수 있도록 컴포넌트에 대한 정보와 컴포넌트간의 관련성에 대한 정보를 제공해 주고, 사용자가 원하는 컴포넌트가 없는 경우 가장 유사한 컴포넌트를 가져와 합성할 수 있도록 합성함수를 사용하거나 화면상에서 직접 드래그하여 합성하는 시스템을 구현하였다.

앞으로의 연구과제는 지속적인 연구를 통해 보다 안정적이고 효율적인 관리 시스템이 되도록 적절한 저장모델(repository model)을 도입하는 것이 필요하다.

앞으로 개선 보완될 과제는 프로그래머가 정의하는 클래스 상속 구조가 적절하지 않을 경우 이를 인식하여 프로그래머에게 알려주는 기능이 있다면 보다 많은 도움을 주리라 생각된다.

참 고 문 헌

[1] W.B. Rauch-Hindin, "Reusable Software," Electronic Design, pp. 176-194, Feb. 1983.

[2] C.V. Ramamoorthy, A.Parkasy, W.T.Tasi & Y.

Usuda, "Software Engineering: Problems Perspectives," IEEE computer, pp. 191-209, Oct. 1984.

[3] E.Horowitz & J.B.Munson, "An Expansive View of Reusable Software," IEEE Trans. on S.E., Vol. SE-10, pp. 477-487, Sep. 1984.

[4] T.A.Standish, "An Essay on Software Reuse," IEEE Trans. on S.E., Vol. SE-10, pp. 494-497, Sep. 1984.

[5] G.Booch, in Software Engineering with ADA, 502, The Benjamin/Cummings publishing Company, Minlo Park, 1983.

[6] J.A.Goguen, "Parameterized Programming," IEEE Trans. on S.E., Vol. 10, No.5, pp. 528-543, Sep. 1984.

[7] M.A.Ketabchi, V.Berzins, "Mathematical Model of Composite Composite Object and It's Application for Organizing Engineering," IEEE Trans. on SE., Vol. 14, No.1, pp. 71-84, 1988.

[8] Wendy B. Rauch-Hindin, "Special Series on System Integration," Electronic Design, Electronic Design and System & Software, Feb. 3, 1983.

[9] 김치수, 서동수, 이경환, "상속 기능을 갖는 소프트웨어 컴포넌트의 합성에 관한 연구," 한국 정보과학회 학술 발표 논문집, Apr. 1989.

[10] Yie-Farm Chen, Michael Y. Nishimoto, C. V. Ramamoorthy, "The C information abstraction system," IEEE trans. on Software Eng., Mar. 1990.

[11] W. Teitelman, L. Masinter, "The Interlisp programming enviornment," IEEE Computer, Apr. 1981.

[12] J. L. Steffen, "Interactive examination of a C program with Cscope," In Proc. USENIX Assoc. winter Conf., Jan. 1985.

[13] I.P. Goldstein, D.G. Bobrow, "A layerd app-Roach to software design," Rep. CSL-80-5, Xerox, 1980.

[14] 이원영, 최은만, "객체지향 프로그램의 이해를 지원하는 시스템에 관한 연구," 한국정보과학회 봄 학술발표논문집, Vol. 22, No. 1, 1995.

[15] Norman Wilde, Ross Huitt, "maintenance support for object oriented programs," IEEE trans. on Software Eng., Dec. 1992.

[16] D.W.Embley, S.N.Woodfield, "A Knowledge Structure for Reusing Abstract Data Types," 9th Int. Conf. on S.E., pp. 360-368, 1987.

김 치 수

1984년 중앙대학교 전자계산학과(학사)

1986년 중앙대학교 대학원 전자계산학과(공학석사)

1990년 중앙대학교 대학원 전자계산학과(공학박사)

1990년~1992년 공주교육대학교 전임강사

1992년~현재 공주대학교 전자계산학과 조교수

관심분야: 소프트웨어공학(특히, 객체지향 분석 및 설계, 소프트웨어 재사용)