객체의 개념적 인식과 논리적 분석에 의한 재공학 툴에 대한 연구

김 행 곤†

요 약

소프트웨어 재공학은 시스템 생명주기 전반에 걸쳐 생산성과 품질향상을 가지게 하며 소프트웨어 유지보수성을 새로운 기법과 유지보수 툴의 적용을 통해 기존 시스템의 성능을 향상시킨다. 또한 기존 시스템의 이해성을 높이고 시스템 전반에 걸쳐 설계 구조나 자료구조와 같은 소프트웨어 컴퍼넌트를 추출하는데도 필요하다. 이들 컴퍼넌트는 시스템 개발 또는 재개발시 재사용된다. 기존의 객체 지향 과라다임은 소프트웨어 유지보수성을 향상시키는 방법으로 알려지고 있다. 그러나 객체지향의 개념적 통합을 위한 객체, 속성, 오퍼레이션의 인식과 객체 클래스의 구성과 같은 문제점을 가지고 있다. 따라서 본 논문에서는 객체지향 시스템의 재공학 기본 방법론과 객체지향 과라다임의 재공학을 위한 개념인식에 대해 논하며 또한 기존 절차 중심으로 개발된 프로그램을 객체지향 시스템으로 변경하는 재공학 툴에 대해 논한다. 이 툴은 객체지향 인식에서 개념적 무결성 문제를 해결하는 장점을 가진다.

A Study on the Reengineering Tool with Concepts Recognition and Logical Analysis of Objects

Haeng-Kon Kim†

ABSTRACT

Re-engineering has the potential to improve software productivity and quality across the entire life cycle. It involves improving the software maintenance process and improving existing systems by applying new technologies and tools to software maintenance. Re-engineering can help us understanding existing systems and discover software components(e.g., design structure, data structure) that are common across systems. These common components then can be reused in the development (or redevelopment) of systems, thereby significantly shortening the time and lessening the risk of developing systems. The Object-Oriented paradigm has been known to improve software maintainability. There still exist many problems in recognizing object, attributes and operations that are conceptually integrated and constructing of object class. In this paper, we propose a method that defines a fundamental theories of re-engineering and a concept recognition for object-oriented paradigm. We also describe the re-engineering tool that translates the existing procedure-oriented program into object-oriented system. This tool has a strength to solve the conceptual integrity problem in object-oriented recognition.

1. 서 론

급속히 발달하는 하드웨어 개발 기술에 힘입어 저렴한 컴퓨터가 널리 보급되면서 컴퓨터가 사용되는 응용분야가 점차 확대되고 컴퓨터 사용인구도 증가하게 되었다. 이로인해 소프트웨어가 점차 복잡한 일도 처리 하도록 요구되어지고 소프트웨어 수요도 양적으로 증가되고, 기존의 단순한 작업만을 수행하던 소프트웨어의 수준도 점차 복잡, 전문화 되어 가고 있다. 그러나 소프트웨어는 사용자들의 요구가 이전보다 복잡해지고, 고도의 기술을 지니고 있는 소프트웨어 전문가의 부족, 소프트웨어 개발도구와 방법론이 소프트웨어를 개발하는 능력을 혁신적으로 향상시키지 못하게 되는 등의 이유로 소프트웨어의 공급이 수

*본 논문은 1995년도 한국학술진흥재단의 공모과제 연구비에 의해 연구되었음

† 정 회 원 : 대구효성가톨릭대학교 컴퓨터공학과 조교수

논문접수 : 1995년 5월 16일, 심사완료 : 1995년 10월 20일

요를 따르지 못하게 되었으며〈표 1〉, 사용자가 요구하는 소프트웨어가 점차 복잡해지고 그 규모가 방대해짐에 따라 개발된 소프트웨어의 정확성도 의심을 받게되는 등 소프트웨어의 위기를 맞게 되었다[1].

그러나 객체지향 패러다임이 많은 사람의 관심을 끌게되고 보편화 되면서 소프트웨어의 생산성 향상을 위한 한 수단으로 주목을 받고 있다. 이는 각각의 소프트웨어 모듈을 하나의 객체로 만들어 두었다가 새로운 소프트웨어를 개발할때 재사용할 수 있도록 하는 것으로 소프트웨어 생산성을 향상시키는 것이 궁극적인 목적이다.

기존의 패러다임으로 개발된 시스템들은 사용자들의 새로운 요구와 환경의 변화, 개발 당시의 기술보다 뛰어난 새로운 기술이 개발됨에 따른 새로운 요구, 그리고 시간이 흐를수록 수정 양이 많아져 처음 설계 정보와의 불일치성으로 인해 기존의 시스템을 새로이 만들도록 요구 되어지고 있다.

이러한 상황들이 기존에 개발된 시스템들이 재공학 되어야 할 필요성을 증대 시키고 있다. 그러나 이러한 변화 요구에 기존의 시스템들은 잘 적응하지 못하는데 문제가 있다. 그 이유는 기존의 패러다임으로 개발된 시스템들은 표현상 어떤 변경이 있을 경우 종속된 모든 모듈들이 영향을 받게 되는 경향이 있다.

〈표 1〉 소프트웨어 수요와 생산성의 비교
〈Table 1〉 Demands and productivity of software

증가 대상	20년간 증가율	연 증가율
수 요	100 배	12 %
소프트웨어 생산성	1.8배	4 %
개발 인력 규모	10 배	4 %

기능적 분해 방법으로 개발된 시스템들은 데이터의 표현법이 설계 초기에 결정되기 때문에 상태에 관한 정보가 공유될 때에는 시스템의 결합도를 증가시키는 경향이 있어 요구사항 변경으로 인한 설계 변경에 잘 적응하지 못하기 때문이다. 따라서 기존의 패러다임으로 개발된 시스템은 확장성(extensibility), 재사용성(reusability), 생산성(productivity)에 한계가 있다. 그러나 객체지향 패러다임은 이러한 문제점을 잘 극복하고 있다.

본 논문에서는 프로그래밍 패러다임과 재공학에 대한 개념 및 객체지향 패러다임의 재공학 기본 이론을 정의하고 기존의 절차중심으로 개발된 프로그램을 객체지향 시스템으로의 변형을 위한 재공학 방법론(reengineering methodology)을 제시한다. 또한 기존 프로그램에서 객체의 인식을 통한 클래스를 구축하는 툴인 CROS(Concept Recognition and refinement Object-oriented System)에 대해 논한다. 이 CROS는 객체지향 개발의 문제점 중의 하나인 객체인식에 대한 방법에 중점을 두어 기존의 프로그램에서 보다 양질의 객체를 추출하기 위한 객체인식 방법을 제공하며 정제를 통한 객체의 무결성을 입증한다.

2. 연구배경

2.1 관련연구

최근 각종 객체지향 언어가 개발됨에 따라 기존 시스템을 객체지향 시스템으로 재공학하는 많은 시도들이 이루어 지고 있다. 이 논문과 관련된 연구로는 〈표 2〉와 같다[2, 3, 4, 5].

Zimmer는 기존 코드를 객체 모듈 형태로 재구성하는 방법을 시도하였고,Jacobson은 기존 시스템을 수동으로, 그리고 점차적으로 객체지향 시스템으로 재구성하는 방법을 제안하였으며, Feldman은 Fortran 프로그램을 자동으로 C나 C++로 변환해주는 방법을 제안하였다. 이 방법은 객체를 인식하거나 생성할 수 없으며, 따라서 객체지향 코드를 생성할 수 없다. 자료 추상화 장치(Data Abstraction Facility)를 제공할수 있도록 포트란 언어를 확장하는 방법이 Miller에 의해 시도된 바 있으며, Dietrich는 기존 시스템들을 객체지향 시스템으로 변환하기 위해 기존 시스템들 위에 객체지향 인터페이스를 만들어 연결시켜 기존 시스템을 객체지향 시스템으로 변환하는 시도가 있었다. Waters는 코드 추상화를 통해 변환 과정을 시도하였다. 이 방법은 어구(cliches)나 계획(plans)을 인식하여 원시 코드의 자료로 사용하도록 하였으며 목적 코드를 생성하는 새로운 언어에 의해 제공되는 더욱 의미

있는 구조를 사용하였다.

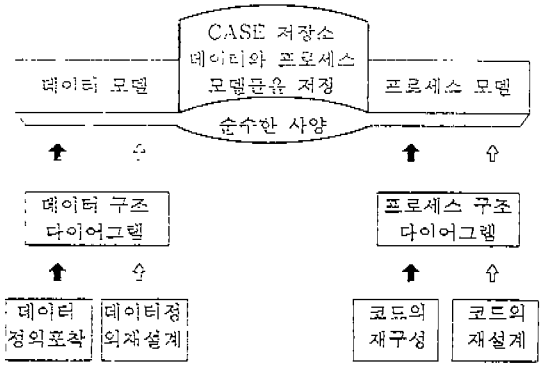
〈표 2〉 관련 연구표
(Table 2) Related works [2, 3, 4, 5]

연 구 자	연구내용	인식(객체, 클래스,메소드)정도	구 현	비 고
Hausler 90	COBOL 프로그램을 재 구성	인식 안됨	구현중	수동 접근
Zimmer 90	FORTRAN 프로그램을 재 구성	수동으로 객체인식	구현중	〃
Jacobson 91	기존코드를 재 구성	인 식	구현중	〃
Waters 88	FORTARN을 ADA로 변환	계획을 인식	안 됨	어구 라이브러리 필요
Feldman 90	FOTRAN을 C,C++ 변환	인식안됨	구현됨	자동변환
Ong 93	FORTRAN을 C++ 재공학	객체,클래스인식	구현중	자동변환
AO. ET 93	C를 C++로 재공학	객체,클래스, 메소드 인식	구현중	자동변환

그러나 이러한 접근 방식은 원시 코드에서 나타날 수 있는 모든 가능한 계획을 미리 예측해야 하므로 먼저 거대한 라이브러리를 만들어야 하고 변환기를 만들때에는 프로그램의 변화를 상세히 기술하기 위해 언어를 사용하여야 한다. Hausseler는 프로그램 실행에 필수적인 두가지 요소인 구조적 관점과 기능적 관점을 제시하였다. 구조적 관점은 응용하는데 필요한 구성요소의 계층구조 관계를 나타내고, 기능적 관점은 응용에 의해 계승된 기능화의 계층구조를 나타낸다. 그밖에 많은 사람들이 기존 시스템을 객체지향 시스템으로 자동으로 변환하는 방법에 대해 연구를 하고 있으나 이는 현재 어려운 상태이다.

2.2 역공학

소프트웨어 역공학은 전통적 개발단계 진행방향의 역으로 거슬러 올라가 기존 코드나 데이터로 부터 설계사양이나 요구분석서를 복구시키는 것이다. (그림 1)은 역공학에서 수행되는 기능들을 나타내고 있는데, 데이터 기능에 대한 역공학은 그림의 왼편에, 프로세스 기능에 대한 역공학은 오른편에 나타나 있다.



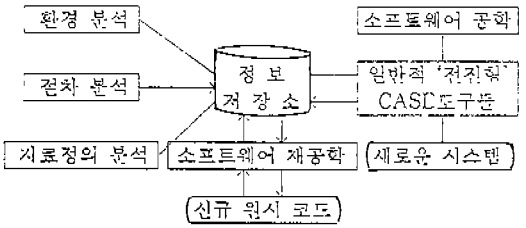
〈➡ : 역공학, ⇨ : 순공학〉
(그림 1) 역공학과 순공학의 과정
(Fig. 1) Reverse and forword engineering

(그림 1)에서 보여주는 바와 같이 데이터 정의를 사양으로 변환시키는 과정은 3단계로 이루어 지는데, 먼저 원시코드에서 물리적인 데이터 정의를 포착하고 데이터 구조 다이어그램에 포함되어 있는 도식의 표현들을 개선하며, 그 다음이 데이터 구조 다이어그램을 개념적 다이어그램(데이터 모델)으로 변환 시킨다. 여기서 개념적 다이어그램이란 일반적으로 엔티티 관계 다이어그램(E-R 다이어그램)유형으로 표현되는 순수한 사양을 말한다. 프로세스 부분에 대한 역공학은 비구조적인 프로그램 코드를 CASE 툴의 저장소에 입력될 수 있는 상위 설계 사양으로 변환시키는 과정을 요구하고 있는데 매우 어려움이 많다.

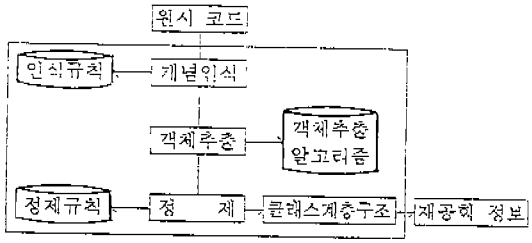
그러나 문제는 원래의 소프트웨어 분석 및 설계에 사용되었던 여러 가정들이나 고려되었던 대안들이 코드 안에 내재되어 있지 않고 사라져 버렸기 때문에 데이터를 이해할 수 있는 지식기반 시스템이 필요하다. 기존 코드로부터 객체를 생성하는 것도 최근 많이 연구하고 있으나 어려운 문제로 남아있다.

2.3 소프트웨어 재공학과 CASE

CASE는 다양한 도구들과 개발 방법론의 집합으로 그 체계를 이룬다. CASE는 발전하면 할수록 그 중심을 지키는 정보 저장소(repository)에 크게 의존한다. (그림 2)는 재공학과 역공학이 CASE의 정보 저장소와 연관 관계가 있음을 나타내고있다.



(그림 2) 소프트웨어 재공학 - 역공학과 CASE 정보저장소
(Fig. 2) Re-engineering, Reverse engineering and CASE repository



(그림 3) CROS의 구조
(Fig. 3) CROS architecture

3. 시스템 설계 및 구현

3.1 개 요

모든 시스템은 한정된 생명주기를 갖는다. 기존의 개발된 소프트웨어의 유지보수 과정을 통한 변화(Δ)는 초기의 시스템 구조를 서서히 파괴한다. 시간이 지남에 따라 이러한 변화에 따른 유지 보수 비용은 전체 개발 노력의 70% 이상을 차지하고 있으며 이 비중은 점점 증가할 것이다 [11]. (그림 3)은 개념인식과 정제를 통한 객체지향 시스템으로 재공학하기 위한 제한된 시스템인 CROS(Concept-recognition and Refinement Object-Oriented System) 구조이다.

이러한 모델의 첫번째 특징은 코드에만 의존하지 않고, 실세계의 응용영역에 대한 객체 모델링을 한다. 객체 모델링을 통해 다음과 같은 장점을 얻을 수 있다. 객체 모델링을 통해 시스템에 존재할 수 있는 객체의 종류를 파악할 수 있고 코드에서 객체를 추출할 때 얻기 힘든 개념적 무결성(conceptual integrity)문제를 해결할 수 있다. 기존의 객체지향 언어를 이용한 객체지향 시스템으로의 재공학 방법론에서 제공하기 힘들던 클래스 계층 구조를 객체모델링을 할 때 만들어진 클래스 계층구조를 활용함으로써 실제 코드상에서 추출된 객체의 구조를 만드는데 도움이 된다.

또한 이로써 객체의 재사용을 향상시킬 수 있다. 실세계의 응용영역에서 객체모델링을 통해 얻어진 객체들은 시스템에 존재하는 객체의 경계가 된다. 그래서 코드의 분석을 통하여 각 객체의 실제적인 속성 및 연산을 찾을 때 이들 객체를 인식할 때보다 효율적으로 객체를 찾을 수 있다. 두번째 특징은 코드를 통해 객체를 인식하는 방법의 다양화이다. 기존의 방법들이 코드에서 객체를 인식할 때 코드에 존재하는 데이타 및 그에 대한 기능을 중심으로 객체를 인식하고자 하였다.

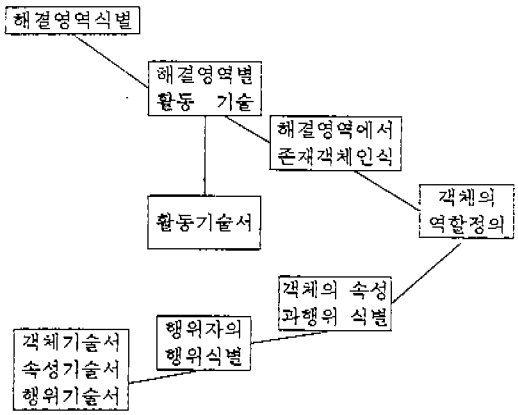
본 논문에서는 자료 및 그에 대한 기능을 중심으로 객체를 인식할 수 있는 새로운 방법을 제시하며, 코드에서 일어나는 외부 객체 및 다른 소프트웨어 시스템과의 인터페이스를 이용하여 객체를 인식하는 방법을 제시하였다. 세번째 특징은 실세계의 응용영역에 대한 객체 모델링만으로 객체를 추출할 때 속성 및 연산을 찾기위해 많은 노력이 필요했으나 이들을 코드에서 찾게 함으로써 속성 및 연산의 추출을 용이하게 하였다.

3.2 개념인식 및 추출

본 논문에서는 개념인식과 추출을 위한 객체지향 재공학은 실세계의 영역에서 객체 모델링을 통해 인식된 객체들에 대한 문서와 원시 코드에서 개념인식의 단계를 설정하여 프로그래밍 개념과 추상화 개념 인식을 통하여 시스템의 기능 및 원시 코드의 각 루틴 및 변수의 기능을 파악하고, 이들 정보들을 이용하여 객체지향 시스템으로 재설계할 때 해결 영역(solution domain) 및 실세계의 응용 영역에 존재하는 객체를 추출한다.

3.2.1 객체 모델링을 통한 객체의 인식

실세계의 응용영역에서 객체 모델링은 응용 시스템 분석 과정에서 그들의 상호작용을 파악한 다음 각 객체에 부과된 책임 즉, 행위를 인식한다. 여기에서 시스템 개발 동기와 동기별로 어떠한 사건이 일어날 것인가 하는 점을 찾는다.



(그림 4) 모델링을 통한 객체 인식
(Fig. 4) Object recognition with modeling

이에 따라 동기를 부여한 제안자와 이의 요청에 따르는 행위자의 활동이 어떠한 것이 있는가를 인식하게 된다. 즉 객체와 그들의 속성 및 행위을 식별하게 된다. 이 과정에서 객체와 그들 객체들의 속성과 행위에 대한 문서가 만들어진다 (그림 4).

3.2.2 언어개념 인식을 통한 객체추출

프로그램은 일반적으로 언어개념과 추상화 개념으로 분류되는데 언어개념은 변수, 모듈, 문장들을 포함하는 구문적 개체이다. 이러한 개념은 언어의 구문적 특성에 의해 정의되어진다. 기존의 원시코드는 주로 기능분해와 탑-다운 방식에 의해 구조화 프로그램으로 개발이 되었다. 즉 시스템의 코드를 기능적 응집도(functional cohesion)를 갖는 블럭으로 분할하여 작성되었다. 이러한 블럭들을 잘 분석해보면 객체들을 찾을 수 있다. 또한 전역 변수같은 개체는 객체의 속성이 될 가능성이 많이 있으므로 객체의 속성은 단위 블럭내의 변수가 될 가능성이 많다. 그러나 이들 단위 블럭이 어느 객체의 연산이 되는가를 판단하는 것이 문제가 된다. 이 문제를 해결하기 위해 단위 블럭을 일정한 카드 형식에 맞추어 단위 블럭의 행위를 기입하고, 이 카드의 내용을 단위 블럭을 클러스터링(clustering)하여 개체 추상화(entity abstraction)를 지니는 객체를 추출한다. 즉 언어개념에서 구문적 개체 개념인식을 통하여 객체를 추출한다. 추출 방법은 첫째 단위 블럭을 인식하고, 둘째 객체 인식을 위한 카드를 작성하고, 셋째 이를 이용하여 객체를 추출한다.

(1) 단위 블럭 추출

속성이 될 수 있는 많은 자료들은 코드상에 전역변수 및 매개변수의 형태로 나타난다. 그러므로 이들 변수들을 분석하면 속성을 찾을 수 있다. 또한 이들에 행해지는 행위를 중심으로 기능적 응집도를 이루도록 코드에서 단위 블럭을 인식하고 이를 오퍼레이션으로 정의할 수 있다. (그림 5)에서 lParam 매개변수는 마우스의 위치의 값을 갖고 있다. 이 값을 이용하여 마우스의 위치를 이동하는 단위 루틴을 인식한 예이다. 이러한 단위 루틴은 기능적 응집도를 지녀야 하며, 객체의 오퍼레이션으로 정의한다. 그러나 한 단위 루틴이 여러개의 기능을 수행하면 좋은 루틴이 되지 못한다. 그러므로 하나의 단위 루틴은 하나의 기능을 수행하기 위하여 꼭 필요한 내용만 포함될 수 있도록 코드를 분할하여 단위 루틴이 기능적 응집도가 높도록 해야한다.

```
case WM_MOUSEMOVE:
    if(wParam & MK_LBUTTON && nCount
    <1000)
    points [nCount ++] = MAKEPOINT
    (lParam);
    hdc = GetDC(hwnd);
    SetPixel (hdc,LOWORD(lParam),HIWORD
    (lParam),0L);
  ReleaseDC(hwnd,hdc);
  }
  return 0;
```

(그림 5) 매개변수를 통한 단위 루틴 인식
(Fig. 5) Unit routine recognition with parameter

중요한 변수를 중심으로 단위 루틴을 만들어야 한다. 각 서브루틴 혹은 블럭내에는 여러 개의 변수들이 존재한다. 그러나 이들 변수는 실제 기능의 중심이 되는 변수와 그 변수에 대한 기능을 수행하기 위한 보조적인 데이타를 가지고 있는 변수들로 구성되어진다. 그러므로 단위 루틴을 만들 때 실제 기능의 중심이 되는 변수에 대한 동작이 일어나는 코드상의 범위를 단위 루틴으로 만들고 가능하면 상위루틴이 되게 한다. (그림

6)에서 points[i].x, points[i].y, points[j].x, points[j].y는 작업 대상이 되는 클라이언트 윈도우에서 점의 좌표(x,y)를 설정해 줄 뿐 중요한 것은 그 좌표끼리 무엇을 하느냐하는 동작일 것이다. 이러한 동작과 관계를 맺어보면 선을 긋는다, 원을 그린다등이 있을 것이다. 여기서 이들 모두를 포함할 수 있는 상위 루틴을 찾으면 DRAW라는 하나의 단위 루틴으로 인식이 가능하다.

```
case WM—DRAW:
      hdc = BeginPaint(hwnd,&ps);
      for(i=0;i(nCount −1;i++)
        for(j=1; j(nCount; j++)
          {
          MoveTo(hdc,points[i].x, points[i].y);
          LineTo(hdc,points[j].x, points[j].y);
          }
```

(그림 6) 기능 변수를 통한 단위 루틴 인식
(Fig. 6) Unit routine recognition with function variables

(2) 템플릿 작성

앞 절에서 만들어진 각 단위 루틴을 객체를 인식하기 쉬운 형태의 다음 템플릿(그림 7)에 맞추어 그 기능을 작성할 때 필요한 방법을 제시한다. 아래 템플릿은 단위루틴의 기능이 객체에 정보를 전송하거나 정보를 받아들일때 사용할 수 있는데 작성 방법은 다음의 규칙을 따른다.

① 항에는 그 단위 루틴의 기능을 나타낼 수 있는 동사를 기술한다.

② 항에는 자료를 포괄할 수 있는 외부 개체나 자료구조등을 기술한다.
: 내용을 기술할 때 실세계의 응용영역을 객체 모델링하여 나온 객체를 참고한다. 객체 모델링을 통해 얻어진 객체들은 현재의 응용영역에서 존재하는 객체의 경계를 설정하고 있다.

③ 항에는 단위루틴의 기능 및 단위 루틴의 중심되는 변수가 어느 객체에 사상될 수 있는가를 파악하여 그 객체의 이름을 기술한다.

동일한 동작을 수행하는 루틴이 코드상의 여러 서브루틴에 반복되어 있을 수 있으므로 이 들을 추출할 때 동일한 이름으로 기술되도록 해야 한다.

① 동 사	② 변 수	From/To	③ 변 수

(그림 7) 템플릿
(Fig. 7) Templete for object recognition

(3) 객체의 추출

앞 절에서는 객체의 메소드가 될 수 있는 단위 루틴을 객체를 인식하기 쉬운 템플릿에 작성하는 방법을 기술하였다. 이 절에서는 전절에서 만들어진 템프릿을 이용하여 객체를 추출하는 방법을 살펴보겠다.

1 단계 : 작성된 각 템플릿의 ③항이 같은 템플릿을 하나로 그룹화 하고, 외부 개체 혹은 실세계의 응용영역에 대해 객체 모델링을 통해 나온 객체를 참조하여 이들을 객체로 정의한다.
2 단계 : 그룹화된 각 템프릿들의 ②항을 객체의 속성으로 한다.
3 단계 : 그룹화된 템플릿의 동사를 객체의 메소드로 한다. 그룹화된 각 템플릿의 항목들을 참조하기에 맞도록 적당한 이름을 부여한다.

3.2.3 프로그램 개념 인식을 통한 객체추출

추상화 개념으로 계산문제 해결방법의 언어에 대한 독립적인 사고를 표현하는데 일반적인 코딩 전략이라든가 자료구조, 알고리즘등이 프로그램밍 개념에 속한다. 여기서 시스템내의 표준 자료구조 및 사용자 정의 자료구조는 전역변수들을 개념적 속성으로 하여 정의된 하나의 객체라 정의할 수 있다. 이 절에서는 원시 코드상에서 사용되는 전역변수를 이용하여 객체를 인식하는 방법을 기술한다. 일반적으로 전역변수는 객체의 속성 및 객체가 될 가능성이 많이 있다. 그러므로 이들 전역변수가 하나의 객체로 정의될 수도 있고 전역변수들 사이의 관계를 위주로 하여 클러스터링하여, 하나로 클러스터링된 전역변수들을 속성으로 하는 하나의 개념적 객체를 정의할 수 있다. 전역변수를 이용한 객체의 인식 과정은 먼저 전역변수를 클러스터링하는 단계와 클러스터로 부터 객체와 속성을 추출하는 단계로 구성된다.

(1) 전역변수 클러스터링

객체인식을 위하여 다음의 세가지 방법으로 전역변수를 클러스터링한다.

① C에서 struct나 union, Fortran의 Commom 같은 자료구조를 이루는 전역변수는 하나로 클러스터링한다. 소프트웨어 시스템에서 관련된 개체를 나타내는 자료구조는 함께 사용 혹은 수정되기 때문에 그룹화되기 쉽고 이를 자료구조는 객체의 인스탄스 변수에 대응된다. 그리고 이들 자료구조들의 모임은 객체가 될 수 있다.
② 공통의 자료를 지니고 있는 전역변수들은 하나로 클러스터링한다.
③ 코드의 템플릿 작성에서 나온 단위루틴을 이용하여 전역변수들을 클러스터링 한다.

(그림 8) 객체 인식을 위한 클러스트링 방법
(Fig. 8) Clustering for object recogrition

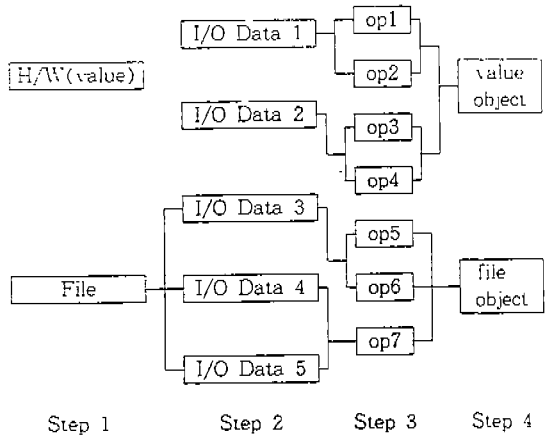
(2) 객체 및 속성추출

앞 절에서 클러스터링한 각 클러스터(cluster)를 각각 하나의 객체로 추출한다. 이때 클러스터링된 전역변수들은 클러스터링을 통해 정의된 개념적 객체의 속성이 된다.또한 단위루틴들은 전역변수들 중 각 단위루틴의 동작의 핵심이 되는 전역변수가 속하는 개념적 객체의 메소드가 된다. 또한 이들 전역변수를 사용하는 루틴에서 전역변수에 대한 이들의 상태를 수정 또는 이용하는 메소드를 추출한다.

3.2.4 인터페이스를 통한 객체 추출

소프트웨어는 외부개체와 인터페이스를 하며 이들 외부개체들은 객체로서, 이들과의 인터페이스는 객체지향 방식으로 구현할 수 있다. 객체지향 시스템을 만들기 위해서는 소프트웨어와 인터페이스를 하는 외부개체를 가상기계(virtual machine)로 내부적으로 구현할 필요가 있으며 이들 외부개체를 객체로 정의할 수 있다. 그러므로 기존 시스템에 구현된 인터페이스를 분석하여 객체를 추출한다. (그림 9)에 인터페이스로부터 객체를 인식하는 과정을 나타내었다.

시스템과 인터페이스하는 외부개체에는 사용자와 인터페이스하는 하드웨어(CRT, 키보드, 마우스, 프린터등)와 외부 소프트웨어와 같은 것들이 있으며, 이들은 역공학을 수행하는 과정에서 얻을 수 있다. (그림 9)의 예를 볼때 1 단계에서는 시스템에 존재하는 외부개체로서 value 및 화일이 존재한다.



(그림 9) 인터페이스로 부터 객체 인식
(Fig. 9) Object recognition through interface

I/O를 수행하는 루틴들은 네트워크 관련 라이브러리 기능, 통신방법, I/O관련 시스템과 소프트웨어 라이브러리와 같은 것들이 있다. 또한 이들 I/O를 수행하는 루틴의 매개변수중에 이들 I/O 데이타가 들어있을 가능성이 많다. (그림 9)의 2 단계에서는 1 단계에서 얻은 각 외부개체별로 일어나는 I/O루틴 및 I/O데이타를 찾아 기술한 형태이다. 즉 (그림 9)에서 하드웨어(Value)에 I/O되는 루틴 및 I/O 데이타를 찾은 결과 I/O 데이타 1(I/O 루틴 1), I/O 데이타 2(I/O 루틴 2), I/O 데이타 3(I/O 루틴 3)이 구해진다.

I/O를 수행하는 루틴에서 I/O되는 데이타를 속성으로 둔다. 그러나 연산을 추출하기 위해 I/O 데이타에 설정되는 값을 분석하여 의미를 추출하고 이들 의미를 대상으로 각 연산을 만든다.

(그림 9)의 경우 I/O 데이타 1에 설정되는 값에 따라 연산 1 및 연산 2가 추출된다. 이러한 경우의 예를 들면, value에 대해 on/off를 조절하는 데이타를 인터페이스할 경우 value_status =0은 value를 On, value_status=1은 value를 off 하라는 의미를 지니고 있다. 이들을 의미별로 value on( ), value—off( )라는 각각의 연산을 정의할 수 있다. (그림 10)은 이들 입력되는 nCmdShow 값에 따라 윈도우를 활성화되고 아

이콘화되는 메소드를 인식한 예를 나타내고 있다.

몇개의 I/O 데이타를 조합하여 하나의 의미를 만들 수 있고 이것이 하나의 연산이 된다. (그림 9)에서 I/O데이타 4와 I/O 데이타 5를 합하여 하나의 연산이 되는 경우도 있다. 예를 들면, (그림 11)과 같이 메뉴를 만들기위해 여러개의 printf문을 사용할 경우, 이로부터 4개의 I/O루틴(I/O Data)을 얻을 수 있으며 이들을 조합하여 보면 CRT에 메뉴를 출력하기 위한 루틴들임을 알 수가 있다. 그러므로 조합하여 하나의 연산 즉, Display_Menu( )을 추출할 수 있다.

```
if(!InitInstance(hInstance,nCmdShow)
    return(FALSE);
while (GetMessage(&msg,null,null,null))
    {
    TranslateMessage(&msg);
    DispatchMessage(&msg);
    }
 return(msg,wParam);
    :
    :
```

(그림 10) 입출력 값에따른 메소드 인식
(Fig. 10) Method recognition through I/O value

객체를 추출하기위해 하드웨어, 외부 소프트웨어 및 화일의 경우 이들을 하나의 객체로 한다. 즉, 실세계의 개체를 소프트웨어 경계내에 구현하기위한 가상기계(Virtual Machine)기능을 하는 객체로 된다. 예를 들면, (그림 9)의 외부개체인 value의 경우 그와 관련된 모든 연산을 모두 갖는 객체가 된다. 그리고 실세계의 응용영역에 대한 객체 모델링에서 추출한 객체, 전역변수에서 추출된 객체, 템플릿 작성을 통해 인식한 객체를 참고하여 I/O 루틴을 그룹핑하는데 이용한다.

3.3 정 제

추출된 객체들은 중복되게 찾아지는 것들도 존재하고 이들 중 객체의 인식은 동일하나 서로에서 찾아진 속성 및 연산들이 상호 누락된 것들이 존재하기도 하여 서로 보완적인 기능을 지니고 있는 경우도 존재한다. 또한 어떤 경우 어느 한 가지 방법에서 추출된 객체의 동일한 속성이나

```
printf("1. File");
printf("2. Edit");
printf("3. Run");
printf("4. Compile");
```

(그림 11) 메뉴 표시
(Fig. 11) display_menu( )

같은 연산들이 서로 다른 객체에 소속되기도 한다. 또한 코드를 대상으로 객체를 추출한 결과 객체지향 시스템의 효과를 확실히 살리지 못하는 경우가 발생한다. 따라서 각 방법에서 추출한 객체를 보다 합리적인 객체가 되도록 이들을 결합과 정제를 하고 이들을 좀더 재사용 가능하도록 재설계 해야 한다. 정제 방법은 (그림 12)에 나타내었다.

추출된 객체를 정제(refinement)를 할때 가급적 실세계에서 추출된 객체의 내용을 중심으로 정제한다. 코드에는 프로그래머의 편리를 위해 코딩할 때 필요한 데이타를 기술하지 않고 표현되는 경우가 있다. 이런 경우 코드에서 속성을 완전히 기술하는 것이 용이하지 않다. 또한 유지보수되는 과정에서의 많은 수정으로 인해 코드를 분석할때 실수가 발생하기 쉬우므로 코드를 통한 객체의 인식에 잘못이 따를 경우가 있다. 그러므로 우리가 이들을 이용하여 객체를 추출할때 이들을 전적으로 의지하는 것은 상당히 위험하다. 그러나 실세계의 응용영역의 객체에 어느정도 사상(mapping)을 시켜 결합 및 정제를 하는것이 좋다.

중복되게 인식된 객체들의 속성 및 연산들을 정제할 때 이들 속성 및 연산들의 중복도 유의해야 하며, 이때 각 방법에서 중복되게 인식된 객체 및 속성 및 연산들의 이름이 다르게 정의될 수도 있으므로 이에 주의하여 정제해야 한다.

① 모델링된 객체의 내용을 중심으로 정제하라.
② 중복된 객체는 하나로 결합하라.
③ 객체들 간에 속성의 공유가 생길 경우
  - 실세계의 객체에 따르라.
  - 속성을 가장 많이 사용하는 객체에 편입하라.
④ 속성이 동일한 두 객체를 하나로 결합하라.
⑤ 객체는 속성들의 상태를 변경하는 연산을 가져야 한다.

(그림 12) 정제 규칙
(Fig. 12) Refinement rules

실세계의 응용영역에 대한 객체 모델링에 의해 인식하지 못한 객체를 템플릿 작성에서 찾은 경우 다른 객체들과의 중복이 일어나는가 확인한다.

객체들간에 속성의 공유가 생길 경우 첫째, 실세계의 객체에 따른다. 둘째, 개념적 무결성을 지니도록 적절한 객체로 편입을 시킨다. 셋째, 그 속성을 가장 많이 사용하는 객체에 편입을 시킨다. 이때 이들을 속성으로 지니고 있던 객체에서의 각종 연산들도 수정이 되어야 한다.

모든 속성이 동일한 두 객체를 하나로 결합하고, 단일 속성(single attribute)을 지닌 객체는 피한다. 객체는 객체내의 속성들의 상태를 변경하는 몇개의 연산을 포함해야 한다.

3.4 클래스 계층구조 구성

추출된 객체들은 객체간의 상속을 얻을 수 있도록 하고 이들을 통해 객체의 재사용성을 높일 수 있다. 객체들간의 일반화-특수화(generalization specialization)관계를 이용하여 클래스 계층구조를 만든다. 이때 실세계의 응용영역에서 객체 모델링을 할때 만든 객체 구조를 이용하면 클래스 계층구조를 만드는데 많은 도움이 된다. 객체의 클래스 계층구조를 만들기 위한 과정을 (그림 13)에 나타내었다.

① 공통의 속성 및 연산을 찾아 하나의 범주로 하라.
② 공통의 속성및 연산을 추출하라.
③ 클래스 계층구조를 만들라

(그림 13) 클래스 계층구조 구성
(Fig. 13) Construction of class hierarchy

4. 적용 예

앞에서 언급한 단계별 적용 예로 윈도우즈 프로그램을 사용하였다. 적용 예로 사용한 프로그램은 키보드와 마우스를 사용하여 단순한 입출력을 행하는 단순한 프로그램이다. MS-WINDOW상에서 실행되며 Borland C로 구현된 원시 코드를 Visual C++언어로 재구현하게 된다.

4.1 개념인식 단계

개념인식 단계에서는 기존 시스템을 분석하여 CRT, 마우스,키보드 등의 외부개체가 존재함을 인식하였고 기존 시스템의 인터페이스로는 사용자 인터페이스를 위한 CRT, 마우스, 키보드에 대한 인터페이스 등이 존재했다. 윈도우의 기본 데이타 구조인 hInstance, hPrevinstance, cCmdShow, lParam등의 전역변수가 존재한다.

4.2 객체 추출 단계

실세계의 응용영역을 분석하여 객체를 추출했으며 그 결과 많은 객체를 추출할 수 있었다. 그 추출된 객체의 일부는 (그림 14)와 같다.

```
Window
public:
  Handle hInstance
  Handle hPrevInstance
  int    nCmdShow

  int    MessageLoop
()
  .          .
  .          .
```

```
Mouse
protected:
  Bool LeftButton;
  Bool RightButton;
  int init—x, init—y;
public:
  WinMouse( ... )
  .          .
  .          .
virtual void WMLButto-
nup(...)
```

(그림 14) 추출된 객체
(Fig. 14) Extracted object

개념인식을 통한 객체추출은 윈도우즈 프로그램에서 모든 윈도우즈에서 참조하는 전역변수를 사용하여 클래스를 정의한 예를 (그림 15)에 나타내었다.

```
class windows {
public:
static HANDLE hInst-
ance
static HANDLE hPrevI-
nstance
static int nCmdShow
int MessageLoop()
int void Register()
int ShowWindow()
void Update()
: : : }
```

```
Class   WinMouse:public
MainWind{
protected :
  Bool LeftButton;
  Bool RightButton;
  int init—x, init—y;
public:
  WinMouse( ... )()
    .          .
    .          .
virtual      void
WMLButtonup( ... )
virtual      void
WMLButtonup( ... ) }
```

(그림 15) 전역 변수를 이용한 클래스 정의
(Fig. 15) Class definition with global variable

인터페이스를 통한 객체의 추출은 사용자 인터페이스를 통해 입력되는 메세지를 처리하는 다음

의 함수의 인식을 통해 함수의 동작 대상이 되는 마우스 객체를 추출할 수 있었다(그림 16).

```
Window::WinMouse(PTWindowsObject, aParent, . . .
 {
 Init_x=Init_y=0;
 x=30, y=0;
 LeftButton=RightButton=FALSE;
 }
 : :
```

(그림 16) 함수의 인식
(Fig. 16) Function recognization

4.3 정 제

추출된 객체들을 모델링을 통해 추출된 객체를 참조하여 원시코드에서 추출된 객체들과 함수들을 결합하여 완전한 객체를 정의한다.

```
Class WinMouse:public MainWind{
 protected:
 BOOL LeftButtonPress;
 BOOL RightButtonPress;
 int init_x, init_y;
 public:
 WinMouse( ... )
 : :
 virtual void WMMButtonup( ... )
 virtual void WMLButtonup( ... )}
```

4.4 클래스 계층구조 구성

추출된 객체들을 계층구조를 형성한다.

초기화 클래스

```
Class MainWind {
 public:
      static HANDLE hInstance
      static HANDLEhPrevInstance
      static int nCmdShow
      static int MessageLoop()
        :          :
HANDLE Main::hInstance=0; }
```

수퍼 클래스

```
Class Window {
    protected:
      HWND hWnd;
    public:
      HWND
GetHandle(void)
        {return
hWnd;};
      BOOL   Show(int
nCmdShow)
        {return
Show...};
              : : :
    };
```

서브 클래스

```
Class     WinMouse:public
MainWind{
 protected:
 BOOL LeftButtonPress;
 BOOL
RightBUttonPress;
 public:
 WinMouse( . . . )
 : :
 virtual       void
WMMouseMove(.....)
 virtual       void
WMLButtonDown(...) }
```

5. 결 론

지금까지 기존 시스템을 객체지향 시스템으로 재공학하기 위해 많은 시도가 있었다. 본 논문에서는 객체의 무결성을 위하여 실세계의 응용영역을 모델링하여 얻은 정보를 이용하였다.객체의 인식과 구축을 통해 객체지향 시스템으로의 재공학 방법론에서 제공하기 어려웠던 클래스 계층구조를 구축하기가 보다 용이하고 객체 인식으로 얻어진 컴퍼넌트는 객체의 재사용성을 향상 시킬 수 있는 장점이 있으며 기존 여러 종류의 프로그램 언어의 다양한 객체 인식 기법을 제공한다.

본 연구에서는 클래스 계층 구조 및 객체간의 인터페이스 방법은 아직 미흡하고 인식되어진 객체들로서 기존 프로그램의 기능을 만족시킬 수 있는가에 대한 검증이 미비하다. 그리고 인식되어 많은 객체들을 정제하는 과정에서 많은 어려움이 있다. 향후 연구 방향으로는 객체간 인터페이스 방법론 및 툴의 구현과 객체를 추출하는 방법에 있어 유효한 객체를 쉽게 추출하는 방법론 및 개념 인식 방법론에 기초한 자동화된 툴의 구현 그리고 다양한 언어에의 개념적용을 위한 연구가 필요하다.

참 고 문 헌

[1] R. Balzer, T.E.Cheatham and C. Green, "Software Technology in the 1990's : using a new paradigm", Computer, pp. 39-45, November 1983

[2] J.A. Zimmer, "Restructuring for style", Software Practice and Experience, Vol. 20(4), pp.365-389, April 1990

[3] W. Miller, et al, "Adding Data abstraction to Fortran software", IEEE Software, pp.278-285, November 1988

[4] I, Jacobson, "Re-engineering of old system to an Object-oriented architecture", OOPSLA '91 Proceeding ,pp.73-89, 1991

[5] S.J Feldman, et al, "A Fortran-to-C Converter", Computing science, Technical Report No.149, pp.149-372, AT&T Bell Lab, 1990

[6] 김행곤,현창문, "소프트웨어 개념인식과 변형에 대한 연구",한국정보과학회,제21권,1호, pp.647-652, 1994,

[7] E. Chikofsky, "Reverse Engineering and Design Recovery : A Taxanomy", IEEE Software, pp.13-17, Jan 1990

[8] J. Martin, J. Odell, Object oriented analysis and design, Prentice Hall,1990

[9] John D. Mcgregro, David A. Sykes, Object-Oriented Software Development : Engineering Software for Reuse, Van Nostrand Reinhold, 1992

[10] B. Mayer, Object-oriented Software Construction , Englewood Cliffs, N.J. Prentice Hall,1989

[11] 강성구, "기능 데이타, 인터페이스 분석에 의한 개념적 객체 추출 방법", 포항공대 정보산업 대학원, 1994

김 행 곤

1985년 중앙대학교 전자계산학과 졸업(학사)
1987년 중앙대학교 대학원 전자계산학과 졸업(이학석사)
1991년 중앙대학교 대학원 전자계산학과 졸업(공학박사)
1978년~79년 미 항공우주국 객원연구원
1987년~90년 한국전기통신공사 전임연구원
1988년~89년 AT&T 객원연구원
1990년~현재 대구효성 가톨릭대학교 컴퓨터공학과 조교수
관심분야 : 객체지향시스템 설계, 사용자 인터페이스, 소프트웨어 재공학, 유지보수 자동화 툴, CASE