□ 기술해설 □

소프트웨어 재사용을 위한 CASE 구축

중앙대학교 이경환*

● 목 차 ●

1. 소프트웨어 재사용의 의미와 역할

1.1 소프트웨어 재사용의 의미와 역할

소프트웨어의 정렬(sort), 검색(search), 큐, 스택 등과 같은 모듈들은 많은 소프트웨어 개발에서 중복하여 나타난다. 이미 개발되어 있는 소프트웨어와 새로 개발할 소프트웨어가 특정 부분에 있어서만 다르고 나머지는 대체로 같거나 비슷하다는 점은 소프트웨어의 재사용 방법이 소프트웨어 개발에서 매우 효과적이라는 설명을 해준다. Lanergan과 Poynto[1]은 소스 코드의 40~60%가 하나 이상의 응용 소프트웨어분야에서 반복된다고 보고하였고, Kapur[2]는 프로그램에 나타난 함수들 중에서 약 70%가 유사하다는 연구를 하였으며, Jones[3]는 85%가 같은 내용이거나 일반적인(유사한)내용이라고 보고하였다.

소프트웨어개발에 필요한 지식을 표준화하고 축적하여 소프트웨어를 개발하는 과정을 통해서 재사용의 효율을 최대화 시킬 수 있다. 재사용할 수 있는 지식은 다음의 두가지로 구분할 수 있다.

① 선언형 지식(declarative knowledge)
소프트웨어 최종 생성물과 중간 생성물에 관한 지식

② 처리 절차형 지식(procedural knowledge)
소프트웨어 최종 생성물과 중간 생성물을 생산하는 절차에 관한 지식

일반적으로는 처리 절차형 지식을 포함한 선언형 지식을 재사용하는데 이들을 소프트웨어 공학적인 입장에서 재사용하는 대상은 다음과 같이 구분할 수 있다.

① 선언형 지식의 표준화→여러가지 표준화 양식과 문서, 프로그램 부품

② 처리절차형 지식의 표준화→개발 절차의 규정, 개발 지원 도구

즉, 생성된 양식과 문서, 그리고 프로그램 부품과 이들을 생산한 절차와 지원 도구들이 모두 재사용의 대상이 된다. 개발 절차와 도구의 재사용은 표준화와 방법론, 그리고 CASE 도구들

* 종신회원

로서 자동화된 재사용 시스템들이고 표준화된 양식과 문서, 그리고 프로그램 부품을 재사용할 수 있는 시스템을 개발해야 된다. 재사용의 대상들은 개발계획 단계에서 시스템의 개념, 요구정의, 분석과 설계 등과 같은 상세한 부품들로 나타난다. 그러나 본 논문에서는 재사용을 지원하는 CASE 도구를 중심내용으로 제한하게 됨으로서 프로그램 부품을 중심으로 한 재사용 시스템을 대상으로 설명한다.

소프트웨어의 재사용은 소프트웨어개발의 생산성의 향상에 목적을 두고 다음과 같이 정의할 수 있다.

$$\text{생산성} = \frac{\text{가치}}{\text{작업량}} = \frac{\text{가치}}{\text{생산량}} * \frac{\text{생산량}}{\text{작업량}}$$

$$= \frac{\text{가치}}{\text{작업량}} = \frac{\text{생산량}}{\text{개발량}} * \frac{\text{개발량}}{\text{작업량}}$$

$$= \text{품질(가치 생산성)} * \text{증폭율} * \text{개발효율}$$

소프트웨어의 품질은 그 소프트웨어가 갖는 가치로 결정한다. 소프트웨어가치는 사용자의 요구를 잘 반영한 설계에 있다. 증폭율은 기존의 소프트웨어를 재사용하는 비율로 결정하고 재사용 비율이 클수록 증폭율이 커진다. 개발 효율은 협의의 생산성으로 새로 개발할 소프트웨어의 작업 효율을 나타낸다. 가치 생산성은 사용자의 요구에 부합하도록 개발함으로써 높아지고, 프로토타이핑과 설계 검토 기법 등을 통해서 성취할 수 있다. 증폭율은 기존의 소프트웨어의 재사용 비율을 높이기 위하여 표준화, 구조화, 그리고 부품화 기법을 도입한다. 개발 효율은 자동화를 목표로 하여 고급언어, 개발 지원 도구와 개발 환경을 통해서 이룩할 수 있다. 이상과 같은 내용을 바탕으로 해서 재사용을 지원하는 소프트웨어 개발방법론은 다음과 같이 정리할 수 있다.

① 기존의 표준양식과 표준 부품을 활용한다.
② 표준양식과 표준 부품을 조합하기 위한 상위 구조의 패턴(모듈)을 활용한다.
③ ①과 ②항에 포함된 고도의 지식으로 설계자가 판단하여 지원 도구를 활용한다.

1.2 재사용과 프로토타이핑

프로토타이핑의 목적은 실행 가능한 프로토타입을 먼저 확보하고 사용자의 요구에 대한 시스템 양식의 타당성을 분석하여 수정 보완시키는 방법에 의해서 사용자와 개발자가 공동으로 목표한 시스템을 개발한다. 이때 타당성 분석은 다음과 같은 사항을 중심으로 평가한다.

① 사람과 컴퓨터의 인터페이스의 타당성
② 복잡한 알고리즘의 실현성
③ 목표한 성능의 실현성

프로타이핑의 적용을 위해서는 소프트웨어 개발의 생명 주기도 수정해야 되는데 무엇보다도 사용자 요구를 파악하는 "what"과 소프트웨어를 동작시켜서 원하는 정보를 출력시키는 "how"를 구별하여 분석해야 한다. 개발자가 사용자의 요구를 파악하기 어려운 것은,

① 사용자의 요구가 불완전하거나 불명확하고,
② 사용자와 설계자의 대화의 차이가 발생하며,
③ 사용자의 요구를 구조화(모듈화) 하기가 어렵기 때문이다.

이러한 사용자 요구(what)의 문제를 실현방법(how)와 구별하되 상호연관되는 메시지 전달 사항을 파악하고 what에 대응된 정보처리수단

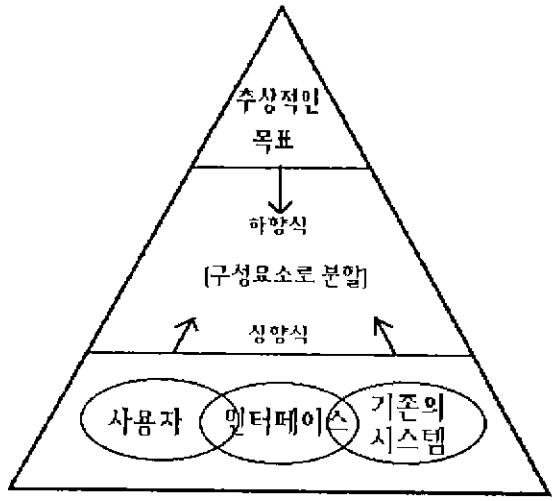

그림 1 소프트웨어 설계의 지적 피라미드

(how)을 고려하여 설계한다. 따라서 프로토타이핑 기법에서 재사용 방법을 적용하면 효과적이다. 소프트웨어 개발은 추상적인 사용자의 요구를, 구현할 수 있는 사람과 기계간의 인터페이스와 시스템 내부에 설계될 기능으로 표현하는 과정이다.

그림 1에서 보면 추상적인 목적 개념으로부터 하향식으로 구성 요소를 분할시켜 나가는 구현 과정과, 기존의 설계 양식과 부품을 바탕으로 상향식으로 조합해 나가는 과정이 피라미드 골격 안에서 조합해 나간다.

하향식과 상향식 방법을 조합시키는 과정은 일반적으로 세가지 방법이 있다. 첫째로 대상범위 한정법은 프로토타이핑의 대상 범위를 가능한 적게 하는 방법이다. 즉 프로토타이핑의 목적을 다음과 같이

① 사람과 컴퓨터의 인터페이스의 타당성
② 특정한 기능의 구현 가능성
③ 성능의 평가

로 한정하고 이에 대응한 개발 대상을 제한시켜서 개발하는 것이다. 둘째로 재사용 방법은 기존에 표준화된 소프트웨어를 재이용 함으로서 새로 개발할 부분을 최소화 하는 것이다. 4세대 언어를 사용하여 재사용 방법을 적용한 프로토타이핑의 개발 공정은 그림 2와 같다.

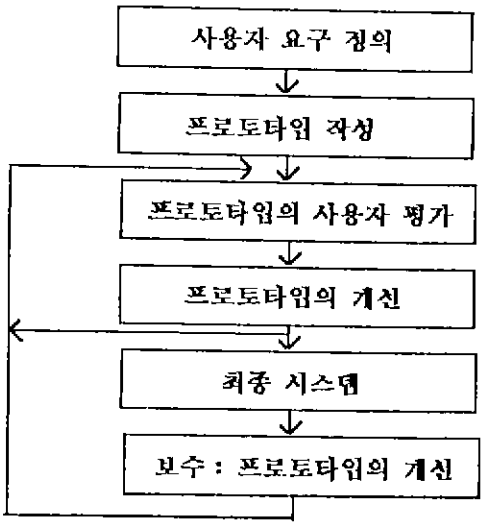

그림 2 4세대 언어를 사용한 프로토타입의 개발 공정

프로토타이핑에 재사용 방법을 적용하는 과정을 CASE 도구로 개발할 수가 있다. 본 장의 주제는 프로토타이핑 기법을 재사용 시스템에 도입한 CASE 도구로 개발하여 소프트웨어를 개발하는 방법을 설명한다. 셋째로 실행가능한 양식 방법이다. 실행가능 양식법은 추상화 레벨의 고급 언어를 사용하여 양식을 기술한다. 따라서 형식적인 양식을 세밀하게 정의하는 양식(what)에서 실현방법(how)으로 변환 과정을 자동화 시킨다. 그림 3은 실행가능한 양식 방법에 의한 프로토타이핑 과정을 설명한 파라다임이다.

재사용법과 실행가능 양식법은 지적 피라미드의 하위 구조 부분의 개발을 생략하고(부품의 이용과 프로그램 자동 생성과 자동 변환) 상위 구조 부분을 새롭게 작성한다. 재사용법은 응용분야를 한정하고 표준화한 대신에 실행가능 양식법은 범용화에 중점을 두고 사용한다.

2. 소프트웨어 재사용시스템의 도입

2.1 재사용의 애로점과 해결 방법

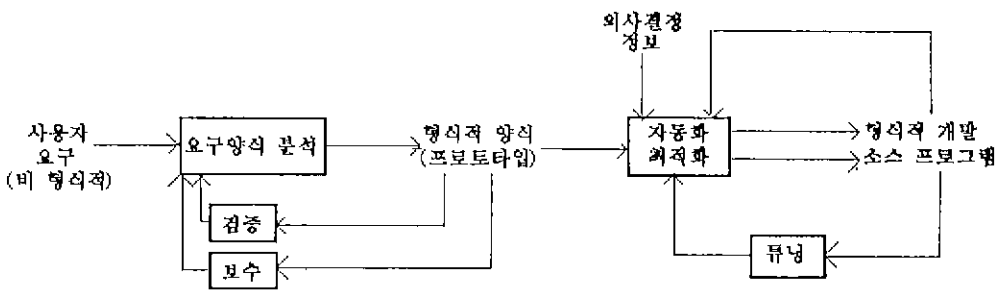

그림 3 자동화 개발의 파라다임

소프트웨어 재사용은 전혀 새로운 기술이 아니고, 지금까지 소프트웨어 시스템의 개발에서 지금까지 자주 사용되어 오던,

- 자금 변환 모듈
- 수학 계산 함수
- 통계 처리 함수
- 입출력 처리 루틴

등의 형태로 라이브러리화 되어 재사용해 왔던 기술의 일반화와 CASE 도구를 결합시키고자 한 것이다.

이러한 개념을 체계화 시켜서 도구로 개발하여 재사용의 효율을 높이고자 하는 것이 최근의 추세이다. 체계적인 재사용 스시템의 효과는

- 개발기간의 단축
- 개발 비용의 절감
- 소프트웨어품질 향상
- 생산성 향상
- 시스템 구축에 관한 지식의 공유
- 중간 생성물 및 최종 생성물의 공유

등으로 사용할 수 있다. 이러한 효과를 거두기 위해서 재사용할 수 있는 지식을 다음과 같이 구체적으로 나열할 수 있다.

- 프로토타입
- 자료 타입과 자료 모델
- 프로그램의 구조적 골격(frame work)
- 프로그램의 코드단위
- 소프트웨어 패케이지
- 재사용 가능한 생명 주기상의 중산 생성물

그러나 재사용의 개념을 소프트웨어 개발 현장에서 적용하려고 하면 여러가지 애로점이 많다. 이러한 애로점을 해결하는 것은 재사용의 도입을 촉진시키고 그 효과를 최대화 시키게 되므로 재사용 시스템의 구축에 새로운 요구사항이 될 수 있다.

① '소프트웨어 재사용'이란 무엇인가?
1장에서 설명한 재사용의 의미와 요구를 구현 시킬 수 있는 방법론과 재사용의 대상을 구현하여 개발한다.

② 여러가지 응용시스템에 공통되는 특성을 가진 부품을 찾기 어렵다. 이를 위해서는 소프트웨어의 표현 기법을 찾고 공통되는 속성을 기술하는 방법론을 연구해야 한다. 객체지향 방법을 도입하여 객체를 추출하고 이들을 추상자료형으로 표현하는 연구가 하나의 해결 방안이다.

③ 재사용을 위한 사전 계획이 필요하다.
재사용을 도입하기 위한 CASE 도구와, 이 도구를 활용할 수 있는 부품화 과정을 표준화 시켜야 한다. 부품화를 위한 소프트웨어의 분류 방법과 재사용의 모델링 기법을 표준화 시켜야 한다.

④ 프로그래밍 언어에서 종속성을 갖는다.
프로그램 코드를 부품으로 재사용하거나 분석과 설계문서를 재사용할 때 채용하는 프로그래밍언어 이외의 다른 언어사용에서 일반성이 결여 될 수 있다. 이러한 문제는 객체 지향 방법론과 같은 기법을 도입함으로서 C와 C++를 공용할 수 있는 인터페이스의 설계를 재사용시스템이 지원할 수 있게 한다. 재사용 시스템에 API(Application Programming Interface) 기능을 도입하는 것은 이러한 목적 때문이다.

⑤ 부품의 재부구조 뿐만이 아니라 인터페이스 요구를 만족해야 한다.
부품의 추상화 방법에 의해서 정보 은닉과 캡슐화를 보장하고 부품을 building할때 멤버 함수나 메소드와 같은 기법에 의해서 부품의 조합을 위한 인터페이스를 표준화 시켜야 한다.

⑥ 소프트웨어 부품의 표현과 분류의 방법론을 연구해야 한다.
재사용의 효율을 높이기 위해서는 부품을 추상화 시켜서 표현하고 부품간의 승계(inheritance) 관계를 정의해야 한다. 이러한 승계, 부품의 분류 방법(classification)에 따라서 재사용의 효과가 좌우된다. 대수적 방식을 도입하거나 위상수학의 개념, 그리고 카테고리 디론 등을 도입하는 연구가 진행되고 있다.

⑦ 재사용의 관리를 지원하는 기법이 필요하다.
재사용은 finding, browsing, 그리고 building의 세가지 원리를 지원하는 통합기능의 도구에 의해서 지원되어야 한다. 이러한

도구는 평가 할 수 있는 방법론에 의해서 개발되어야 한다. 이러한 방법론 중에는 객체 지향 방법론(Object-Oriented paradigm)이 많이 도입되고 있다.

⑧ 재사용의 효과는 장기적으로 나타난다.
개발된 재사용 시스템은 여러가지 응용영역별로 라이브러리를 구축해야만 재사용의 효율이 높다. 표준화된 객체 지향 방법론에 의해서 부품을 개발하는 장기적인 계획에 의해서 이루어지고 개발팀 별로 재사용의 필요성 인식과 표준화된 객체모델링 방법의 교육과정이 필요하다.

⑨ 프로그램의 표준화가 어렵다.
프로그램의 표준화는 소프트웨어 분류 방법에 의해서 객체를 부품으로 개발하고 C++과 같은 객체 지향 언어에 의해서 코딩해야 한다. 프로그램의 표준화는 자료 추상화 표현에 의해서 승계 구조, 다중연산(Polymorphism) 등의 기능을 부여해야 하고 이러한 부품의 표준화는 분류방법에 따라서 재사용의 효율이 평가된다.
앞에서 언급한 바와 같이 대수적인 구조에 의한 양식의 표현 방법과 위상 수학의 이론, 그리고 카테고리 이론 등을 객체 지향 방법론에 적용해야 하므로 많은 기초 연구가 필요하다.

그러나 이와 같은 애로점은 계속된 연구개발로 하나하나 해결되어가고 있으므로 재사용 시스템에 의한 재사용 라이브러리의 구축은 객체 지향 방법론을 적용하고 분류하기가 쉬운 영역의 라이브러리로부터 개발해 나가야 한다.

2.2 재사용 가능한 부품

앞에서 언급한 재사용 가능한 여러가지 형태를 구체적으로 설명해 보면, 재사용의 대상은 소프트웨어 시스템의 개념(모델링) 분석과 설계 문서, 프로그램 코드를 부품으로 생각할 수 있다. 이를 위해서는 체계화된 프로젝트 계획과 문서의 표준화, 소프트웨어 개발에 필요한 여러가지 정보들을 재사용 시스템안에 저장해야 한다. 코드 단위의 재사용을 목표로하여 재사용 가능한 부품의 형태별로 구분하여 설명해 보자.

2.2.1 프로그램 부품(parts)

객체 지향 방법에 의해서 추출된 객체(object)를 C++언어로 코딩하여 부품을 만들거나 포트란이나 코볼같은 언어를 택하여 부품을 작성 할 수 있다. 어떤 경우에도 공통성과 일반성을 가진 부품의 개발이 필요하고 공유 라이브러리의 구성과 표준화된 재사용의 인식하에서 개발되어야 한다. 재사용의 부품이 만족해야될 특성은 다음과 같다.

- 확장성
- 정형성
- 자기정의가 잘된요소
- 수정의 용이성
- 재사용의 효율성
- 프로그래밍 언어에 독립성
- 기술서 작성의 용이성
- 검증의 효율성
- 간결한 이터페이스

이러한 부품 특성 등은 포트란이나 코볼과 같은 기존의 언어로 작성할 경우에는 주의해야 되고, C++와 같은 객체 지향 언어로 이러한 점을 감안하여 필요한 기능을 보유하고 있다.

부품을 재사용하고 있는 실례를 보면 다음과 같다.

Raytheon회사
3,200 코볼 소스코드 모듈
Harford 보험회사
35 소스 코드 모듈
(15개의 프로그램과 20개의 서브 루틴)
AT & T
1,000개의 C 언어 부품
GTE Data Services
220개의 재사용 가능한 부품
960,000 라인에 이르는 코볼, C, 어셈블리 부품

2.2.2 페케이지

소프트웨어 재사용을 완전한 프로그램의 재사용과 프로그램 일부의 재사용으로 나눌 수 있다. 부품의 재사용을 일부의 재사용이라고 한다면

패케이지의 재사용을 완전한 재사용으로 볼 수 있다.

패케이지의 재사용은

- 'AS-IS'의 일반 형태로 완전한 프로그램을 파라미터화 하고
- 파라미터를 통해서 프로그램 내부의 수정을 할 수 있게 한다. 결점 적용 분야가 특수하게 한정 되고 변경하려고 할 때는 더 많은 인력과 비용이 필요하다.

많이 적용되는 분야로는 급여 계산, 은행 및 보험처리, 회계, 네트워킹, 재고 관리, 시스템 소프트웨어 패케이지의 재사용 효율이 높다.

패케이지를 얻는 방법으로는 다음과 같은 점에 유의해서 개발한다.

- 현재 또는 향후 필요한 사항을 상세히 파악
- 해양 분야에 가능한 모든 패케이지들은 조사
- 패케이지들의 분서를 모두 파악
- 패케이지들이 충분한 적용성(파라미터를 통한)을 갖는지 확인하고, CASE 도구를 이용하여 패키이지를 평가하고, 수정하고 관리 가능 여부를 파악
- 재사용 가능한 패케이지들을 체계적으로 저장
- 판매회사를 평가
- 기존의 사용자 그룹을 확인

2.2.3 Templates

Texas Instrument사는 IEF(Information Engineering Facility)의 CASE 도구와 함께 사용되어질 Template을 재사용 부품으로 제공한다. 그 내용을 보면 다음과 같다.

제품명 : General Ledget Template

내 용 : 30개의 *entity type*
40개의 transaction
15개의 batch procedure로 구성된 IEF설계 모델
→600,000라인의 코볼코드 생성

재사용을 지원하기 위해서 수정 가능한 사용자 지침서와 널리 쓰이는 데이타 베이스를 마련하고 있다.

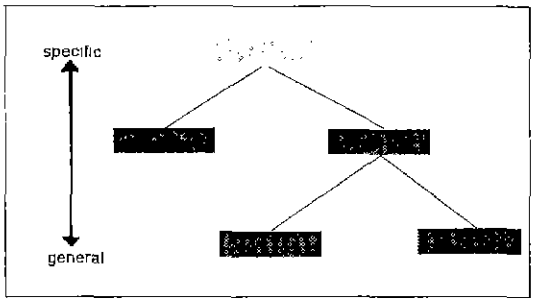

그림 4 부품의 계층 구조

2.2.4 Frames

재사용 가능한 부품의 계층 구조는 그림 4와 같이 구성한다.

상위계층의 Frame들은 하위단계의 Frame들을 수정하거나 모아서 원하는 프로그램을 작성할 경우 코볼 응용 프로그램만을 생성하는데 제한하고 METRON/CAP(코볼 응용 프로그램 코드 생성기)의 CASE 도구와 함께 사용하고 있다.

일반적으로 많이 사용되는 Frames들은 다음과 같은 것들이 있다.

BROWS. F
datafile의 내용을 출력하는 코드
STANDARD.F
코볼의 *division*을 제공
SCROLL.F
레코드 scrolling
LINK.F
프로그램 link를 지원
CONCAT.F
임의의 크기를 갖는 두 스트링을 결합
MATCH.F
두 스트링을 비교
FILE.F
화일의 개/폐쇄/읽기/쓰기를 지원하는 코드
PARMS.F
파라미터의 전송을 수행
SDATE.F
날짜와 시간의 출력 형태 제공

2.3 재사용의 사례: GTE Data Services

다음에는 재사용 기술을 효과적으로 도입하고

있는 GTE Data Services회사의 사례를 들어 보자.

2.3.1 회사소개

사업영역 : GTE전화 회상에 필요한 업무 지원시스템 개발

역점부서 : 2,000명으로 구성된 개발 환경 부서

2.3.2 주사용 언어 : 코볼과 C 언어

2.3.3 작업환경 : IBM 대형, DEC, HP, Tandem, LAN으로 연결된 PC

2.3.4 초기 접근 방법 :

공통된 프로그램 Utility를 수집하여 asset으로 저장하고 소스코드가 아닌 Object Module로 수집하고 개별 프로그램사용에서 프로그래머에 의해 수정 되지 않고 Read-Only 방법으로만 사용된다.

2.3.5 결과 및 효과 :

1년에 수집된 asset수가 300개 정도이고 그 수가 많아 질수록 효율이 감소된다. 현재 220개정의 asset만 보유하고 코볼, C, 어셈블러로 작성된 960,000라인만 재사용 시스템에 포함하고 있다. Table Control System에서 현저한 생산성 증가를 보이고 있다.

2.3.6 장기계획

PACBASE Repository 구입하여 다음과 같은 정보 서비스를 할 수 있도록 계획하고 있다.

- asset의 위치 파악
- asset의 수정
- 변경의 영향 이해
- 재사용 수준과 질에 관한 통계자료 수집, 관리
- asset 카탈로그를 통한 browsing이 용이하도록 보완한다.

IAE(Information Asset Engineering)를 창설하여 35명으로 구성한 소프트웨어 Productivity Group을 조직하고 CASE Group안에서 활동하게 하고 있다. IAE의 운영은 재사용 전문가를 채용하여 GTE의 재사용 카달로그를 만들고 관리, 재사용 성공사례를 수집하고 홍보하여 asset사용 지침서를 작성하여 배포하고 있다. 100명~150명의 개발자당 한명의 비율로 재사용 전문가를 채용함으로서 프로젝트 시작부터 참여하여 요구분석 단계에서 부터 asset이 포함될 수 있는 비율을 결정한다. 현재 개발 프로젝트의 경우 20~25%의 생산성과 유지보수/확장 프로젝트의 경우, 15%의 생산성을 향상 시키고 있다.

3. 재사용 시스템 도구 : CARS

3.1 CARS 설계사상

소프트웨어 개발자들은 일반 응용 분야에서 고객의 편리성과 그들의 업무 효율성, 생산성을 높여 주기 위한 다양한 도구들을 개발하는데 주력해왔던 경향이 있다. 그러나, 이제는 우리 개발자들의 편리함, 개발업무 처리의 효율성, 생산성에 관심을 가져야 한다는 주장이 많다. CARS (Computer Aided Reuse System)는 이러한 요구를 충족시키기 위하여 개발되었다.

개발된 응용 프로그램을 영역별로 부품화하여 라이브러리를 구축하므로써 유사한 작업에 드는 노력을 절감할 수 있다. 이는 단순한 코드의 재사용으로 프로그래머의 코딩 시간이나 테스팅 노력을 절감하여줄 뿐아니라 코드를 생성하게된 설계 개념의 재사용을 가능하게 하므로써 오랜 경험자들의 설계 방식을 자연스럽게 이어갈 수 있도록 한다.

생산성 향상에 중요한 요인이 되는 부분이 표준화이다. 코드를 작성하는 양식의 표준화, 문서화를 이루는데 필요한 표준화뿐 아니라 좋은 설계를 위한 표준화, 나아가서는 사용되는 용어들의 표준화를 통해 일관된 인터페이스 유지를 보장한다. 이렇게 표준화를 유도하므로써 팀 단위의 협동 작업을 원활하게 하고, 개발자간의 의사 교류의 장애를 줄이므로써 생산성 향상을 가져온다. 기존의 라이브러리는 함수들의 목적 코드의 집합으로 구성되어 있어 개발자의 의도와 다를 경우는 스스로 새로운 함수를 만들어 써야 했다. 그러나 CARS를 통한 재사용은 기존 라이

브러리에 저장된 부품의 코드에 새로운 기능의 추가, 삭제, 변경을 자유롭게 도와준다. 소프트웨어를 개발하는 과정을 자세히 살펴보면, 매일 작성되는 코드는 어느 시점에서 비슷한 응용 분야, 또는 다른 응용 분야에서 작성되었던 것일 수 있다. 경험이 많은 프로그래머라면 자신의 머리속에 무수히 많은 소프트웨어 모듈을 넣어 두고 필요할 때마다 기억을 더듬어 다시 사용할 것이다. 이러한 모듈들을 하나의 통합된 라이브러리로 구축하므로써 개발자 자신은 물론 다른 개발자들도 재사용할 수 있도록 한다. 그러므로 CARS는 부품을 개발하여 저장하고 검색하여 재사용할 수 있는 과정까지 지원하고 있다. 최근에 객체지향에 대한 전문개발자나 일반 소프트웨어 사용자들의 관심이 고조되고 있다. 그러나 객체지향 기술은 단순한 환경의 전환 즉, 기계의 변경이나 프로그래밍 언어의 변화로 이루어지는 것이 아니다. 객체지향 기술로의 전이는 프로그래머의 사고의 전이를 요구한다. 즉, 소프트웨어를 구성하는 요소들에 대한 변화를 인식해야만 한다. 소프트웨어 개발단계에서 이루어지는 활동들도 약간씩의 변화를 가져오게 된다. CARS는 객체지향 프로그래밍은 물론 분석과 설계를 지원하는 방법론을 정립하였다.

3.2 CARS 구성도

CARS의 전체 구성도는 그림 5와 같다.

3.2.1 검색 및 이해 지원 시스템

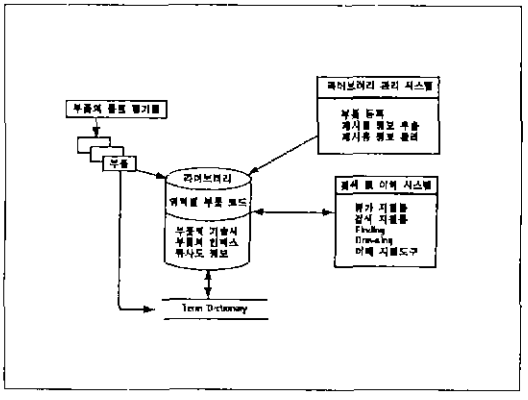

그림 5 CARS 구성도

1) 검색 지원 기능

CARS는 특별한 질의어를 가지고 있지 않으므로 질의어 작성을 위한 구문의 암기화 의미 이해의 노력이 요구되지 않는다. 질의어는 찾고자하는 부품의 성질을 기술하면 CARS 검색 시스템이 기술서를 이용하여 라이브러리를 검색하고 질의에 가장 부합되는 부품을 제공한다.

소프트웨어 모듈의 경우 질의와 완전히 일치하는 부품이 없는 경우가 발생한다. 일반적인 데이타 베이스에서는 이 경우, 검색에 실패한 것으로 간주하지만, CARS는 질의와 가장 부합되는 순서대로 차선의 소프트웨어 부품을 제공한다. 소프트웨어 부품은 수정이나 첨삭이 가능하므로 가장 유사한 부품의 검색은 개발자의 수고를 덜어준다.

2) 이해 지원 기능

소프트웨어 부품을 이해하는데 필요한 요소를 정의하고 이를 나타내는 다양한 형태의 다이아그램을 제공하고 있다.

① 클래스 다이아그램
부품의 구조적 성질을 나타내는 그림으로 부품을 구성하는 외부 인터페이스를 보여준다.

② I/O 차트
부품을 구성하는 인터페이스는 하나의 함수로 대응되어 함수를 구성하고, 함수 사용시에 필요한 입력 매개변수와 출력 자료를 시각화하여 보여주므로써 부품의 사용방법을 알게 한다.

③ 관련 다이아그램
클러스터를 구성하는 부품들은 상호 밀접한 관련을 갖고 있으므로 이들 관계를 표현하여 주는 다이아그램이다.

④ 플로우 차트
프로그램의 내부 구조를 나타내기 위한 도구이다.

3) 검색의 효율성

검색의 효율성을 두 가지 관점으로 평가한다[4]. 첫째 본 검색 환경에서의 질의어 작성이 용이한가? 둘째 질의 작성에 사용된 비제한적 용어를 제한된 단어로 대체해 질의로 사용하는 것이 바람직한가?

이를 평가하기 위해 검색 과정을 다시 살펴보면 아래와 같다. 질의는 재사용자가 필요한 기능을 자연어로 기술하면 된다. 예를 들어 재사용자가 "윈도우를 스크린상에 그리고, 그 윈도우의 배경색을 빨강색으로 변경"하는 기능을 수행하는 부품을 라이브러리에서 찾고자 할 경우 "Draw a window on the screen and change the background color of window red"로 작성하면 된다. 이는 질의를 작성하는 재사용자의 경험과 지식에 따른 용어의 자유로운 선정이 가능한 자연어를 사용하게 한 것으로 특별한 질의어 습득의 노력이 필요없다는 장점을 지닌다. Diaz가 제안한 Facet 단위의 질의어 작성법과 비교하기 위하여 Diaz 가 제안한 질의어 작성과정을 살펴보면 다음과 같다. Diaz 방식에 따르면 라이브러리가(기능, 도메인, 응용영역, medium, 시스템 타입, 언어, setting)의 8개 facet단위로 구성된 함수들의 집합이므로 facet에 해당하는 질의를 입력해야 한다. 이 방법은 다양한 기능을 지니고 있는 객체지향 라이브러리의 부품을 검색하기 위해서는 여러번의 질의를 작성한 후 부울 (boolean)오퍼레이션을 수행해야 하는 단점을 지닌다. 두번째 고려 사항은 재사용자의 경험을 기반으로 작성된 질의의 대체 필요성과 효율에 관한 평가이다. 질의의 재구성 단계는 더 광범위한 영역으로의 확장과 질의의 수정을 수행하는 것이다. 이는 라이브러리를 구성하는 부품에 대한 인덱스가 제한된 언어로 사용되어 있으므로 사용자의 지식을 라이브러리 시스템이 이해 가능한 형태로의 변형 과정이다. 이런 재구성의 과정을 통해 수정/확장된 질의를 바탕으로 검색을 수정하므로써 보다 광범위한 질의처리가 가능해지는 장점을 갖는다. 그러나 단점으로는 본 논문에서는 Diaz방법에서 채택한 질의의 축소/상세화 과정은 고려하지 않았으므로 보다 상세한 정보의 이용을 통해 얻을 수 있는 검색의 정확도 효과는 얻을 수 없었다.

4) 검색 결과의 평가

라이브러리 검색 결과는 대개 "recall"과 "precision"에 의해 평가되는데, "recall"이란 관련없는 부품의 배제 기능에 관련된 평가 기준이며, "precision"이란 관련된 부품의 검색 능력에 관한 평가이다. 이를 그림으로 설명하면 다음 그림 6과 같다.

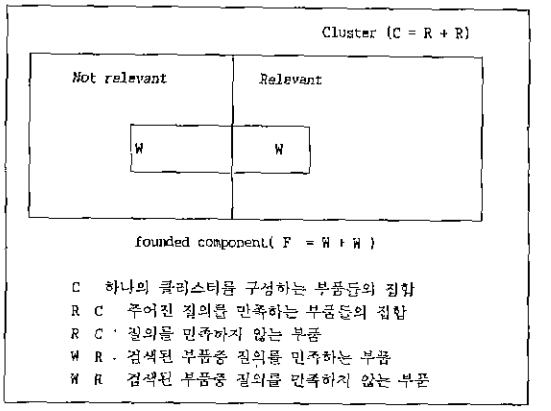

그림 6 "recall"과 "precision"

$$\text{Recall} = \frac{W}{F}$$

$$\text{Precision} = \frac{W}{R}$$

검색의 예에서 보았듯이 본 시스템에 질의 1)과 같은 질의를 주었을때 가장 근접한 기능을 수행하는 부품으로 Reg_polygon이 검색되어졌으며 또한 클러스터 기반 부품 검색을 통해 Rectangle, Square, Polygon, Figure, Circle, Close_figure 등의 부품을 유사한 기능을 갖는 부품으로 검색되었다. 주어진 질의를 바탕으로 검색한 라이브러리는 Graphic/Window 라이브러리로 라이브러리를 구성하는 클래스 부품의 갯수는 87개이고 각 부품을 구성하는 실제 메소드를 포함하면 전체 600여개의 함수를 갖고 있다. 그러나 Graphic/Window 라이브러리에는 질의에 부합되는 기능을 수행하는 부품으로 Rectangle, Square, Polygon, Triangle,Close_figure, Circle, Ellipse 등이 존재한다. 이는 저장된 부품들중 사각형과 유사한 그림을 그리거나 그려진 물체에 색을 지정하는 기능을 담당하는 부품이 7개 존재한다는 것을 의미한다. 이 경우

$$\text{Recall} = \frac{W}{F} = \frac{4}{6} = 0.66$$

$$\text{Precision} = \frac{W}{R} = \frac{4}{7} = 0.57$$

대표인덱스	관련 인덱스
insert	put, append,
attribute	color, width, hieght, font,
display	refresh, show, draw
delete	erase, wipe_out, delete_all

그림 7 Term Dictionary의 일부 예

의 결과를 얻을 수 있다.

3.2.2 TERM DICTIONAR

본 논문의 검색 환경은 정규 표현의 사용을 전제로 하므로 동일한 의미를 지니는 여러 단어들은 제한된 단어로 대체되어야 한다. 이를 지원하기 위하여 인덱스 추출과정의 결과를 바탕으로 동일한 의미의 인덱스, 더 광역의 의미를 지닌 인덱스, 더 좁은 범위의 인덱스들, 이들을 대표할 수 있는 인덱스로 Term dictionary를 구축되게 된다. Term dictionary의 구축 과정에서 자동으로 추출된 인덱스를 바탕으로 라이브러리 관리자에 의해 이루어지는 것을 가정한다. 객체지향 파라다임이 지닌 특성 중의 하나인 클래스 이름이나 클래스를 구성하는 메소드의 이름을 정의시 비정규적인 이름의 사용보다는 범용적이고 제한된 단어의 사용을 유도한다는 점에서 라이브러리 관리자에 의한 Term dictionary작성은 어느 정도 자연스러운 작업이다.

본 논문이 대상으로 하는 WINDOW라이브러리에 대해 구축된 Term dictionary의 일부를 예로 들면 그림 7과 같다.

Term dictionary에 기술되는 내용을 살펴보면, "insert"는 "put", "append" 등의 인덱스와 유사한 기능을 담당하는 서로 다른 인덱스 이름으로 추출되어 "insert"의 대표 인덱스로 대체되게 되며, "attribute"의 대표인덱스의 경우는 관련 인덱스가 유사한 기능의 표현보다는 "color", "hieght", "font"등이 "attibute"에 속하는, 포함관계를 나타내고 있다. 또한 "delete"는 동일한 기능의 인덱스 뿐아니라 더 좁은 의미의 "delete_all", "wipe_out" 을 관련 인덱스로 정의한다.

즉, Term dictionary에 기술되는 내용은 대표 인덱스와 기능적으로 유사성을 지니고 있거나, 대표 인덱스에 포함될 수 있는 속성 또는 의미적으로 대표 인덱스보다 좁거나 넓은 의미를 지닌 인덱스이다.

그림 7과 같이 작성된 Term Dictionary는 추출된 인덱스중 관련 인덱스에 속하는 인덱스를 대표 인덱스로 대체하고 관련 인덱스의 가중치를 대표 인덱스의 가중치로 합산한다.

3.2.3 부품의 품질 평가 시스템

본 서브시스템은 추출전 객체를 클래스의 부품으로 작성한 후에 그품질을 체크하는 기능을 갖는다.

1) 클래스안에서의 메소드 가중치

클래스안에서 메소드의 가중치(weight method per class)를 정의하고[10] 객체의 복잡도를 정하며 메소드들은 객체의 성질이고 한 객체의 복잡한 성질들의 집합의 수에 의해서 결정되기 때문이다.

메소드의 수와 메소드의 복잡도는 그 메소드들을 포함한 객체를 개발하고 유지보수하는데 요구되는 노력과 시간을 나타낸다.

한 객체내의 메소드수가 크면 클수록 그 하위 클래스에 미칠 잠재적인 영향력이 커지게 되는데, 이는 객체내에 정의된 모든 메소드들이 하위 클래스에 상속되기 때문이다. 메소드의 수가 많은 객체들은 일반적인 응용분야에서 보다 많이 이용되는데 재사용 가능성이 극히 제한 된다.

2) 상속 트리의 깊이 (Depth of Inheritance Tree : DIT)

클래스에 대한 상속 깊이는 해당 클래스의 DIT메트릭스이다.

DIT는 성질들의 범위와 관련되어 얼마나 많은 상위 클래스들이 해당 클래스에 잠정적인 영향을 줄수있는가를 측정하는 메트릭스이다.

계층구조상에 깊이 있는 클래스는 상위에서의 많은 메소드들을 상속받기 때문에 보다 복잡하다. 상속 트리가 깊으면 깊을수록 많은 메소들와 클래스들이 트리를 구성하게 되므로 설계 복잡도는 증가하게 된다. 그러므로 특정 클래스가 상속 받는 메소드들의 재사용에 의해 설계되기 위하여 계층구조상의 어느 깊이에 위치하는가를 측정하는 것이 필요하다.

3) 서브클래스들의 갯수(Number of Children : NOC)

NOC=클래스 계층 구조상에서 한 클래스에 직접 종속된 서브클래스들의 수

NOC는 DIT와 마찬가지로 성질들의 범위와 관계가 있다. 이것은 얼마나 많은 서브클래스들이 부모 클래스들의 메소드들을 상속 받게 되는가를 측정하는 척도이다.

일반적으로 클래스 계층구조상에서 트리의 넓이(breadth) 보다 깊이(depth)가 더 유용한데, 그이유는 상속을 통하여 메소드의 재사용을 증가시키기 때문이다. 그런면에서 모든 클래스가 비슷한 서브 클래스들의 수를 갖도록 구성된 계층구조는 바람직하지 못하다. 클래스가 계층구조의 상위에 위치할 수록 하위 클래스 보다도 서크 클래스들을 포함한다.

4) 객체간의 결합도 (Coupling between object : CBO)

한 클래스의 객체간의 결합도(CBO)는 다른 클래스와 관련된 비상속성의 갯수로 정의된다.

만일 두 객체가 서로 작용할 때(하나의 메소드들이 다른 것의 메소드나 인스턴스 변수를 사용) CBO는 한 객체가 다른 객체와 결합되어졌다는 개념과 관련된다.

상속 계층의 과도한 결합도는 모듈과 설계에 손해가 되며, 재사용을 막는 좋지 못한 효과가 있다.

만일 A객체가 B와 결합되고 B는 C와 결합되었다면 이것은 C가 A와 결합되었다는 것을 의미하지는 않는다. 그러므로 결합도에서 결합 법칙 (associative law)은 성립하지 않는다. 모듈화를 증가시키고 캡슐화를 높이기 위하여 내부 객체간 결합을 최소화 하여야만한다.

결합의 수가 많으면 설계에서 다른 부분을 변경하였을 때 그 파급영향이 커지게 되기 때문에 유지보수의 어려움이 증가된다.

결합도의 측정은 설계에서 여러 부분들을 테스팅하는데 얼마나 복잡한가를 측정하는 것과 같다. 그러므로 내부 객체간의 결합도가 높을수록 보다 철저하고 엄격한 테스팅이 필요하다.

5) 클래스에 대한 반응(Response For a Class : RFC)

RFC는 클래스에 대한 반응 집합의 절대값으로 정한다.

반응 집합은 그 객체에서 이용하는 메소드의 집합이며 그 기수는 객체의 특성들에 대한 측정이다. 객체로부터 호출되는 메소드들을 갖는 반응집단은 객체간의 인터페이스(통신) 역시 포함한다.

만일 하나의 메시지에 대한 반응으로 많은 수의 메소드가 호출된다면, 그 객체의 테스팅과 디버깅은 보다 더 복잡하게 된다. 한 객체로부터 호출될 수 있는 메소드의 수는 객체의 복잡도를 증가시키게 된다.

6) 메소드의 응집력 결핍(Lack of Cohesion Method : LCOM)

메소드 M1,M2……Mn을 갖는 클래스 C1 에 대하여 {Ii}가 메소드에 의해 사용되는 인스턴스 변수들의 집합이라고 하자.

LCOM=집합들의 교집합에 의해 허용되는 공통부분을 갖지 않는(disjoint) 집합의 수

이것은 메소드들의 유사성 정도의 개념을 사용하는데 C1 클래스에서 메소드의 유사성의 정도는 다음과 같이 주어진다.

$$(\ ) = \{I1\} \cap \{I2\} \cap \cdots \cap \{In\}$$

만일 공통된 인스턴스 변수가 없다면 유사성의 정도는 0이다. 그러나 이것은 메소드들 각각이 유일한 인스턴스 변수들 집합에 대하여 작용하는 경우와하나의 메소드만이 변수들의 유일한 집합에 대하여 작용하는 경우를 구별하지 않는다. 공통 부분이 없는 집합들의 수는 해당 클래스에서 전혀 관계가 없는 집합들이 존재하지 않는다면 메소드들의 유사성의 정도가 크다는 것을 의미한다. LCOM은 인스턴스 변수들과 객체의 메소드들에 밀접한 관계를 갖고 있으며, 한객체의 특성을 측정하는 척도가 된다.

한 클래스에서 메소드의 응집은 객체들의 캡슐화를 증진시키기 때문에 바람직하다. 응집력의 결핍은 클래스가 두개 또는 그 이상의 서브클래스로 분할 되어야 한다는 것을 암시한다. 메소드들의 불일치 정도에 대한 측정은 클래스들 설계시에 결점들을 정의하는데 도움을 준다. 낮은

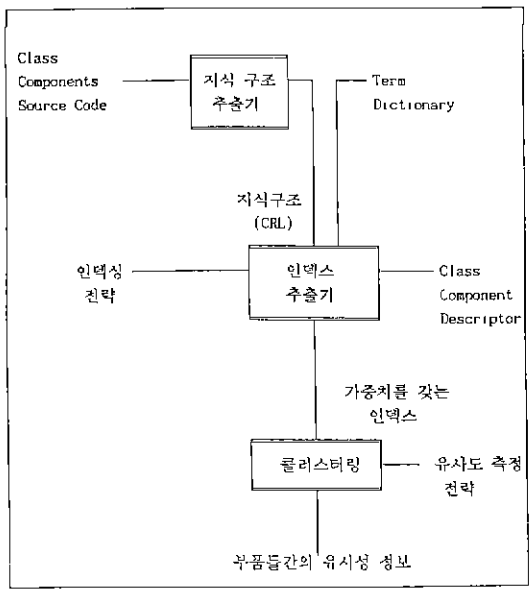

그림 8 라이브러리 구축 과정

응집도는 복잡도를 증가시키며 결국 설계 과정중 에러의 가능성을 증가시킨다.

3.2.4 라이브러리 관리시스템

본 장에서는 객체지향 기법에 기반한 지식 구조와 정보 검색을 통한 부품의 인덱싱 방법을 바탕으로 본 논문에서 구축한 재이용 시스템을 기술한다. 재이용 시스템에 있어 가장 기본이 되는 것은 라이브러리, 라이브러리 분류 방법, 지원시스템이다.

본 논문에서는 window라는 특정 영역의 의도에 따라 라이브러리를 설계 및 구축하고 지원 환경으로 검색 시스템을 구현하였다.

라이브러리의 구축 과정은 그림 8과 같다. 지식 구조를 추출하기 위하여 특정 영역의 클래스를 입력으로 분석한 지식 구조를 표현한다. 구축된 지식 구조 추출기는 C++로 작성된 부품만을 입력으로 받아들여 클래스의 속성, 클래스의 행위, 그리고 클래스의 관련성을 분석한다. 본 논문에서는 인덱스나 질의의 대치 과정에서 필요한 Term Dictionary구축은 수작업으로 수행되며 Term Dictionary와 부품 기술자를 바탕으로 가중치를 갖는 인덱스들을 각 부품별로 추출한다. 인덱스 정보는 유사도 측정의 기반이 된다. 얻어진 지식 구조, 인덱스, 유사도 정보, Term Dictionary로 이미 작성된 부품의 소스 코드와 기술자와 함께 해당 영역의 라이브러리를 구축하게 된다. 또다른 한가지 방법은 새로운 클래스를 개발할 때, 객체를 직접 생성함으로서 사용할 수 있는데 이러한 경우에 재사용을 고려한 클래스가 추상화된 클래스로 설계되면 이 클래스로 부터 객체를 생성할 수 있다. 클래스를 이와 같이 객체를 직접 생성함으로서 재사용할 수도 있지만, 일반적으로 필요한 클래스를 찾아 적절한 수정을 거친후 사용할 수 있다. 상대적으로 구조적 프로그래밍 환경에서의 재사용 방법은 모듈 상호 결합 언어와 같은 새로운 언어를 사용함으로서 가능하게 할 수 있다.

3.2.5 사용자인터페이스 시스템

이 시스템은 사용자가 원하는 사용자 인터페이스 컴포넌트를 재사용할 수 있는 환경을 제공한다. 즉 사용자 인터페이스영역을 위한 재사용 시스템이라고 할 수 있으며 이 시스템은 다음과 같이 구분한다.

① GUI구축기

사용자에게 그래픽 사용자 인터페이스를 설계하는 기능을 제공한다.

- 편집 기능 : 사용자는 에디터 화면에서 응용 프로그램에 대한 원하는 그래픽 사용자 인터페이스를 더욱 쉽게 만들수 있도록 하며 라이브러리에 있는 사용자 인터페이스 컴포넌트를 사용자는 자신의 응용 프로그램에 복사함으로서 새로운 그래픽 사용자 인터페이스를 편집할 수 있도록 한다.
- 탐색 기능 : 라이브러리 관리 시스템의 정보를 이용하여 라이브러리내에 등록된 객체의 검색을 지원한다.

② API 개발 GUI 구축기를 이용하여 다음 그림 9와 같은 대화상자를 작성해 보자.

사용자는 이 대화상자를 작성하기 위해서, 본 시스템에서 제공하는 retrieve메뉴를 통하여, 기존에 작성한 컴포넌트들을 검색한 후 유사한 내용의 사용자 정의 객체를 작업 화면상에서 편집을 하는 재사용의 과정을 거쳐서 다음 그림 10과

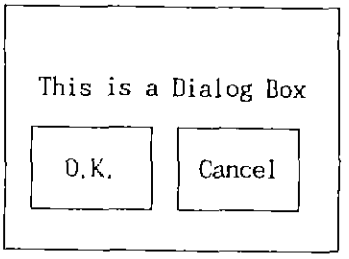

그림 9 대화 상자

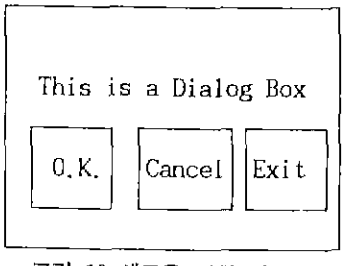

그림 10 새로운 대화 상자

같은 대화상자를 좀더 편리하게 작성할 수 있다.

이 대화상자는 4개의 윈도우 객체와 4가지의 간단한 text객체로서 구성되어 있으며, 사용자는 그림 10과 같은 대화상자를 작성하기 위해서, 기존의 라이브러리를 검색하여 작성하고자 하는 GUI와 유사한 것이 존재하는가를 알아보게 된다. 이 과정을 위해 사용자는 화면상의 retrieve메뉴를 선택하게 된다. 이는 객체들이 facet단위로 제공되기 때문에 등록되 있는 객체를 라이브러리에서 쉽게 제공 받을 수 있다.

4. 재사용 통합 도구의 개발

재사용 라이브러리를 확장하기 위해서는 이미 개발한 소프트웨어 부품으로 재구성하는 재구성 공학 (reengineering)이 필요하고, 보다 더 넓은 의미로는 역공학(reverse engineering)을 도입함으로서 개발된 소프트웨어의 자료문서와 프로세스를 정보저장소(repository)에 저장하고 새로운 코드를 생성할때 활용할 수 있다. 역공학에 의해서 재 구성하는 코드나 새롭게 생성하는 코드를 개발하기 위해서는 분석과 설계과정을 표준화한 객체모델링 방법을 도입해야 된다. 다시말해서 분석과 설계 도구를 개발하여 앞에서 언급한 CARS와 통합하여 각 개발과정의 생성물(개발정보)을 정보저장소에 의해서 지원받는 통합 도구를 개발해야 한다.

4.1 객체모델링과 도구 개발

객체를 이용한 모델링은 실세계를 그대로 반영한다는 장점을 지니고 있다. 구조적 기법에서 사용되어온 주된 모델링 도구는 DFD와 E-R다이아그램, STD이며 DFD는 분석 모델중 프로세스 모델의 특성을 표현하기위한 것이며, E-R 다이아그램은 자료 모델의 특성을 잘 나타내는 것이다. 이들 모델링 도구들은 시스템의 동적 특성을 기술하도록하고 있다. 그러나, 이들 모델링 도구들은 시스템을 구성하는 객체 단위로 표현하지 못하고 분석 모델이 갖는 특징대로 자료와 프로세스 관점을 분리하여 모델링 하도록 한다는 단점을 지니고 있다. 따라서 이와 같은 모델링 도구들의 단점을 보완하면서 원하는 모델을 이용할 수 있도록 하기위해 Beck[5]와 Cunningham[6]에 의해 제안된 CRC 카드 방법을 기반으로 하여, 사용자 모델, 시간적 상태/행위 모델과 관계모델에 필요한 사항을 추가하여 COM (Coherent Object Modeling)을 설명한다. COM은 CARS라이브러리의 부품을 생산하는 객체모델링의 방법론으로 개발 되었다.

4.1.1 COM방법론

먼저 제안한 COM의 특징을 살펴보면 다음과 같다.

1) 모델링 과정에서 필요한 다이아그램 뿐만 아니라 다이아그램에 대응되는 기술서를 제안한다.
2) 객체의 동적 특성은 확장된 STD를 이용하여 표현한다. 즉 시간적 상태/행위 모델의 객체 특성상 STD의 노드는 상태로, 아크는 상태 전이로 대응되게 되며, 상태전이를 이루는 여러 행동들은 아크에 연결된 것으로 간주한다. 이를 위해 본 논문에서는 기본적으로 Moore[7]가 제안한 형태의 STD를 사용한다.
3) 하나의 타입에 대한 모델링으로 여러 가지

다이아그램을 사용한다. 객체는 정적 특성과 동적 특성 모두를 포함하며 이를 하나의 다이아그램으로 분석, 표현하기는 어렵다. 그러므로 객체의 각 특성을 표현하는데 필요한 각각의 다이아그램을 제안하며, 각 다이아그램간의 연결 관계를 두어 단계별 이해 증진을 도모한다.

4) 객체의 정적 및 동적 구조 모두를 기술하는 타입 구조 카드는 객체의 자료 부분에 대한 정보들만 표현하는 E-R 다이아그램 대신 ADT모델을 충실히 표현할 수 있는 다이아그램을 사용한다.
5) 타입 구조 카드에 기술되는 모든 정보는 각 단계의 분석 산물들인 다이아그램들이 구성하는 정보들을 체계적으로 모은 것이다.

4.1.2 분석과 설계도구

일반적으로 분석의 첫단계는 시스템을 사용하게 될 사용자의 관점에서 분석하는 것이다. 이를 위해 시스템 모델링은 개발할 시스템이 받아들이거나, 만들어내야 하는 결과, 외부 신호등을 시스템의 외부사용자 관점에서 파악한 결과를 표현한다. 시스템 모델링으로 개발될 시스템의 범위를 정의할 수 있다. 이는 시스템을 사용하는 사용자를 사용 목적별로 구분하고 이들이 기대하는 시스템으로부터의 결과와 있을 수 있는 시스템 처리 오류의 경우를 파악한다. 제안한 COM을 통하여 먼저 시스템의 개발 범위를 결정하는 comtext diagram과 이에 대응되는 사용자 그룹 기술서를 작성한다.

COM은 분석 단계에 국한된 방법론으로 설계 단계에 대한 구체적인 수행방법을 제안하고 있지 않기 때문에 기존에 제안된 객체 지향 설계 방법론의 공통적인 부분을 근거로 수행하였다. 기존의 방법론들은 객체 지향 설계 단계에서는 분석단계에서 추출된 타입들을 기반으로 효율과 성능을 고려하여 서브 시스템으로 구분하고, 구현 환경에 적합한 자료 구조와 알고리즘을 설계하는 것을 기본으로 하고 있다. 또한 분석된 실세계의 객체를 구현 환경의 Motif widget화 하기 위해 서브 시스템 별로 widget 계층도를 구성하였다. 전체의 구조는 다이아그램 작성기, 기술서 작성기와 화일 편집기의 서브시스템으로 구성되므로 다이아그램 작성기와 기술서 작성기에 대한 widget 계층도와 시스템 기본 메뉴에 대한 widget계층도를 서브 widget계층도로 구성한다. 분석의 결과물로 나타난 각각의 타입이 하나의 복합 widget에 대응되었으며, sort는 복합 widget을 구성하는 요소로 기본 widget 으로 구성되었으며 메세지의 경우는 각 widget의 callback루틴으로 대응되었다. 이상과 같은 방법론을 자동화된 도구를 이용하여 분석과 설계의 CASE도구를 구축할 수 있다.

4.2 정보저장소

정보저장소는 소프트웨어를 개발하는 과정에서 사용자 요구, 분석과 설계, 그리고 코드의 재사용을 위해서 필요한 정보를 제공한다. 그림 11은 도구를 통합하는 정보저장소의 구조를 설명하고 있다.

그림 11에 의해서 소프트웨어 통합제어의 절

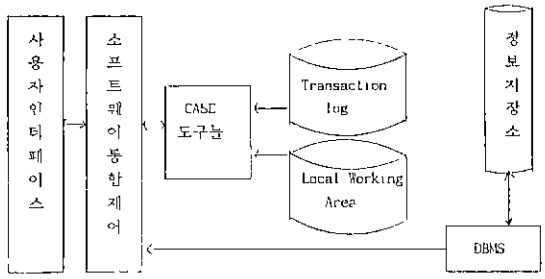

그림 11 통합 정보저장소

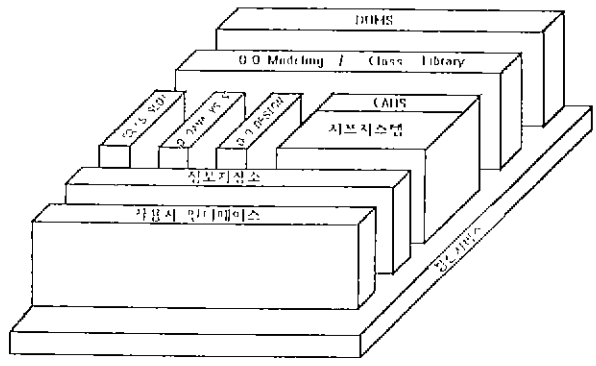

그림 12 객체 지향 워크 벤치

차는 다음과 같다.

① 도구들이 통합제어기를 통해서 정보 요청을 하고 데이타를 패스한다.

② 사용자를 체크하여 정보저장소를 엑세스하도록 허락하고 보안을 유지한다.

③ Local Working Area는 도구가 transaction 처리를 하는 동안에 필요한 자료를 임시 보관 한다.

④ 도구의 기능에 따라서 Local Working Area안에 있는 정보를 추가, 삭제, 그리고 변경할 수 있다.

⑤ Transaction log는 working area안에서 변경되는 모든 데이타를 기록한다. 그리고 변경된 사항을 정보저장소 안에 있는 부품에 수행될 사항으로 보관한다.

⑥ 사용자의 작업이 끝나면 정보저장소에 다음과 같은 메세지를 전달한다.
"COMMIT TO REPOSITORY"

⑦ Transaction log는 통합제어기에 전달된 메세지를 패스하고 local working ares는 사용자에게 성공적인 commit를 알려주고 정지한다.

⑧ 통합제어기는 도구에 메세지를 패스하고 도구는 메세지 내용을 사용자에게 패스한다.

CASE도구들과 통합제어기의 인터페이스 내용은 다음과 같다.

- 도구가 정보저장소에서 얻은 자료
- 정보저장소에서 빠져나간 자료와 도구에 패스된 자료
- 도구에 의해서 변경된 transaction log
- 변경된 사항을 밝히는 메세지

4.3 객체지향 워크벤치

CARS와 객체모델링 도구를 정보저장소에 의해서 통합된 그림 12와 같이 나타낼 수 있다. 그림 12는 객체모델링 도구와 클래스 라이브러리, 그리고 정보저장소에 의해서 도구조립틀을 구축하고 재사용을 지원한다. 통신 서비스는 통합된 통신기능을 가지고 다른 환경의 도구들과 사용자들간에 정보를 패스하고, 객체관리자를 통해서 부품의 재사용을 확대 지원한다. O-O DBMS는 CARS의 라이브러리 서브시스템은 지원했던 DB엔진을 확장하여 재사용을 지원할 수 있는 도구를 개발할 수 있다.

5. 결 론

소프트웨어의 재사용을 위한 CASE는 객체모델링에 의한 부품의 생산을 위한 분석과 설계 과정의 도구를 코드 재사용 도구와 정보저장소에 의해서 통합하여 구축할 수 있다.

본 논문에서는 중앙대학교가 과학 기술처의 지원을 받아서 CARS시스템을 객체 모델링 방법론을 연구 개발하여 COM이라 부르고 두가지를 통합하는 방법을 기술하였다.

CARS는 (주)유니온 시스템과 (주)한국 정보시스템에서 윈도우즈 버전으로 상품화하여 사용중에 있으며 정보저장소와 API의 기능을 추가하여 UNIX버젼을 개발중에 있다.

COM은 CARS의 라이브러리 부품을 분석설계하는데 적용하고 있다. 한편, 대수적 표현 기법을 이용하여 객체의 분류와 표현 기법을 추상화시키고, 위상수학의 개념을 도입하여 시간제약을 받은객체의 상변환에 관한 연구와, 그리고 카테고리 이론에 의한 클래스의 분류방법을 표준화시키려는 연구를 계속하고 있다.

참고문헌

[1] R. G. Lanergan and B. A Poynto "Reusable code : The Application Development Techniques of Future," In Proceeding on IBM Share/Guide Symposium, IBM Corp, Armont, N.K 1979.

[2] L. J. Arthure, B. Kapur, "Measuring Programmer Productivity and Software Quality," John Wiley and Sons Inc, 1985.

[3] T. C. Jones, "Reusablity in Progrmming : A Survey of the State of the Art," IEEE Trans on SE, September, 1984.

[4] Blair, D. C and Maron, M. E "An Evoluation of Retrieval Effectiveness for a Full-text Document-retrieval System," Communications of

the ACM, **28**(3) : March, pp. 289~299, 1985.
[5] K. Beck, and W. Cunningham, "A Laboratory for teaching Object-Oriented Thinking," SIGPAN Notices, **25**(10) 1989.
[6] W. Cunningham and K. Beck, "A diagram for Object-Oriented Programs," In Prpceedings on OOPSLA **86**, pp. 39~43, 1986.
[7] G. Moor, "Introduction to Morden Information retrieval," McG-Hill book Company, 1983.
[8] 중앙대학교, CARS 메뉴얼,1992.
[9] 중앙대학교, '소프트웨어 재이용에 관한 연구', 과기처 보고서, 1993, 10.
[10] 중앙대학교, 강원대학교, "객체지향, 파라다임에서의 품질평가 기준에 관한 연구", 과기처 보고서, 1993, 10.
[11] 이경환, SW재사용을 위한 객체모델링 기법.

이 경 환

1980 중앙대학교 대학원 응용수학 전공 이학박사
1982 미국 Auburn대학에서 소프트웨어 엔지니어링에 관하여 연구
1986 서독 Bonn대학 Institut Fuer Informatik과 공동연구
1986 ~현재 중앙대학교 전자계산학과 교수, 공과대학장

관심 분야 : 소프트웨어 엔지니어링, Object-Oriented Design, 소프트웨어 재사용