경영정보학연구
제 2 권 1 호
1992년 6 월

Smalltalk를 이용한 원시 모듈 구현에 관한 연구

오 상 엽*, 김 홍 진**, 장 덕 철***

A Study on the Source Module Implementation using Smalltalk

We presents a source module implementation system using building blocks as one of the software reusing approach.

Based on module text retrieval system, system is composed of library management program and new module making program for module management.

We build a software module by virtue of the reuser's customization and by applying a practical module to this model and then proved that program development through reusing approach is better than that of writing out a source code directly.

I. 서 론

소프트웨어의 생산성과 품질 향상을 위한 방법으로 기존의 소프트웨어에 대한 재사용 기법이 많이 연구되고 있다. 기존의 소프트웨어를 재사용함으로써 소프트웨어의 개발 기간을 단축하며, 신뢰도를 향상시킬 수 있고, 프로그래머의 능력을 증진시킬 수 있다[Manfred Lenz외 2인, 1987; Samuel T. 외 1인, 1988; Will Tracz, 1987; 이영곤, 1988].

소프트웨어 재사용의 개념은 1949년 캠브리지 대학에서 개발한 프로그램 내장형 컴퓨터인 에드삭에서 서브루틴 라이브러리가 제안된 이후 프로그래밍에서 발전되었다[Will Tracz, 1988]. 1967년 McIlroy는 소프트웨어 구성요소(component)의 등록을 계획하였으며, 이의 응용이 Lanergan과 Poynton에 의해 제한된 영역안에서 수행되었다[Ruben Prieto-Diaz and Peter Freeman, 1987].

소프트웨어 재사용을 위한 접근 방법에는 사용하는 구성 요소 성격에 따라 빌딩 블록(building block) 방법[Gerald Jones, 1987; Susan P.

* 광운대 전산과 박사과정
** 경원전문대 전산과 교수
*** 광운대 전산과 교수

Arnold외 1인, 1987; Ted Biggerstaff외 1 인, 1987]과 패턴(pattern)재사용 방법이 있다[Gerald Jones, 1987; Peter Freeman외 1인, 1987].

재사용 가능한 구성 요소를 식별하는 것은 소프트웨어 재사용에서 가장 기본이 되는 기능이다. 특히, 재사용 가능한 구성 요소들의 집합이 크고, 그 구성 요소들이 여러 응용 분야에 걸쳐 널리 사용되면서 우수한 모듈성을 갖는 재사용 환경에서는 필요한 구성 요소를 검색하고 식별하는 것이 중요한 문제로 제기되며[Bruce A. Burton외 4인, 1987; Samuel T.외 1인, 1988; Scott N. Woodfield외 2인, 1987; Victor R. Basili, 1990], Susan P. Arnold, Stephen L. Stepoway, W. B. Frakes, B. A. Nejmeh 등에 의해 등록과 검색을 위한 재사용 시스템이 발전되어 왔다.

검색된 구성 요소의 선택은 분류(classification)가 기본적인 작업이다. Ruben Prieto-Diaze와 Peter Freeman은 enumerative 분류와 facet 분류를 처음으로 제시하였으며, facet 분류를 이용한 소프트웨어 라이브러리 구성을 제안하였다[Gerald Jones, 1987; Ruben Prieto-Diaz and Peter Freeman, 1987].

본 논문에서는 재사용 가능한 모듈의 저장과 검색을 지원하는 모듈 텍스트 검색 시스템(MTRS, Module Text Retrieval System)을 구현하고, 이 MTRS를 기반으로 새로운 모듈을 구현하기 위한 재사용성 모듈 구현 시스템(RMIS, Reusable Module Implementation System)의 구현을 목적으로 한다. RMIS는 라이브러리 관리 프로그램과 새로운 모듈 작성 프로그램으로 구성된다.

이를 위한 방법으로서 구성 요소는 특정 기능을 수행하는 단위로서 개발되며, 논리적으로 서로 독립된 기능을 수행하거나 분리 번역되었다가 후에 연결되어 사용되는 프로그램의 일부인 모듈로 구성한다. 해당 모듈을 재사용 목적에 적합한 빌딩 블록 방법으로 처리하고, 라이브러리에 저장할 때는 facet 분류 방식을 사용한다.

MTRS와 연관된 재사용 라이브러리에서 사용자가 필요로 하는 모듈을 찾기 위한 질의어와 사용자 메뉴를 설계한다. 그리고 질의어 중심으로 프로그램을 작성하여 사용자가 필요로 하는 프로그램의 주문을 받아서 이를 통합하는 메카니즘을 갖는 실험적 모델을 객체 지향 프로그래밍(OOP, Objcet Oriented Programming) 언어인 Smalltalk로 구현한다.

Ⅱ. 빌딩 블록 방법에 의한 소프트웨어 개발

1. 재사용환경에서의 소프트웨어 개발

재사용 가능한 소프트웨어에 대한 접근은 대상 구성 요소의 성격에 따라 빌딩 블록 방법과 패턴 재사용 방법으로 분류한다.

빌딩 블록 방법은 소프트웨어 구성 요소들을 라이브러리에 저장하여 놓고 새로운 개발에 필요한 구성 요소들을 검색하여서 적절히 수정한 후 다른 구성 요소들과 결합하여 목적 프로그램을 개발하는 방법이다. 빌딩 블록은 전자 부품(electronic chips)과 같은 특성을 가지며, 기능은 캡슐화(encapsulated)되어 있다. 빌딩 블록 방법은 잘 정의된 인터페이스를 필요로 하며, 전통적인 소프트웨어 개발 방법과 비교하여 볼 때, 질적으로 우수하며, 재사용 비율에 정비례하여 생산성이 향상되고, 짧은개발 기간을 가진다[Manfred Lenz외 2인, 1987]. 이 경우에는 재사용 가능한 모듈의 저장과 검색을 지원하는 시설의 개발과 재사용 모듈의 라이브러리를 검색하는 질의어의 개발이 중요하다. 〈표 1〉은 빌딩 블록 방법과 패턴 재사용 방법을 비교하여 제시한 내용이다[Gerald Jones, 1987; Ted Biggerstaff외 1인, 1987; Ted J. Biggerstaff, 1987].

〈표 1〉 빌딩 블록 방법과 패턴 재사용 방법의 비교

구성 요소 재사용	빌딩 블록		패턴		
구성 요소 성 격	수 동 적		능 동 적		
재 사 용 원 칙	구성(compositon)		생성(generation)		
대 상	응 용 구성 요소 라이브러리	조직, 구성원칙	언어 기반 생성기	응용 발생기	변환 시스템
대표적인 시스템	서브루틴의 라이브러리	객체지향, 파이프 구조	VHLLs POLs	CRT Fmtrs File Mgmt	언어 변환기

패턴 재사용 방법은 구성 요소 자체가 능동적 요소이며, 목적 프로그램을 생성하는 방법은 언어 기반 생성기(language-based generator), 응용 발생기(application generator), 변환 시스템(transformation system)과 같은 세 가지로 분류된다[Thomas A. Standish, 1984; Victor R. Basili, 1990]. 재사용 가능한 구성 요소가 응용 발생기와 같은 도구 또는 언어 속에 구축되어 선택(selecting), 주문 제작(customizing), 구성 작업의 대부분이 자동화 되어 있다[Gerald Jones, 1987].

빌딩 블록 방법은 재사용자의 목적에 적합하게 하기 위하여 목적 프로그램의 구성 요소가 개조 결합되므로 수동적 방법이라고 한다.

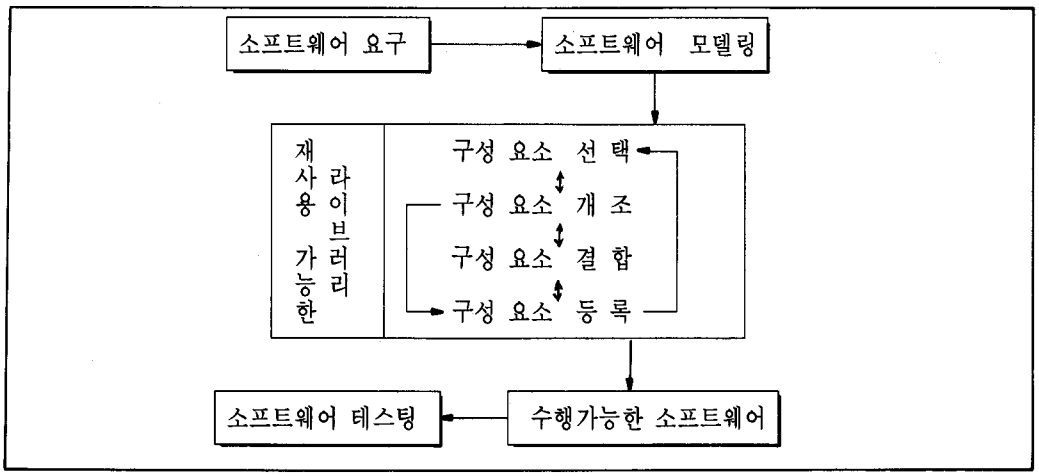
빌딩 블록 방법의 주 목적은 재사용 가능한 소프트웨어의 검색과 등록을 위한 라이브러리를 제공하고, 라이브러리로 부터 재사용 구성 요소를 검색하여 새로운 소프트웨어를 생성하기 위한 환경을 제공하여야 한다. 또한, 재사용자가 반드시 존재하는 구성 요소를 인식하고, 이들이 사용되는 방법을 알아야 한다.

재사용 구성 요소로서 빌딩 블록은 잘 정의된 규칙과 체계적으로 결합된다. 빌딩 블록 방법으로 부터 새로운 프로그램을 얻는 것은 구성의 문제이다. UNIX의 파이프 메카니즘은 한 프로그램의 출력을 다른 프로그램의 입력으로 연결하는 방법에 의해 간단한 프로그램으로 복잡한 프로그램을 구성한다. Smalltalk와 같은 언어의 객체는 일반 프로그래밍 언어의 함수 호출(function call)기능인 메시지 전송과 발생될 처리 방식의 동적(dynamic) 결정에 도움을 주는 상속성을 이용하여 구성 요소를 구성한다.

빌딩 블록 방법의 장점은 새로운 모듈을 구축하는 데 필요한 시간과 노력이 감소되고, 프로그래머의 능력을 증폭시킬 수 있다. 재사용자가 재사용을 위한 도구(tool)를 사용하여 재사용 가능한 모든 모듈을 검색하므로 수작업에 비하여 효율적이며, 시스템의 유지보수가 용이하다. 단점으로는 소프트웨어를 재사용하기 위한 도구를 구축하는 데 필요한 자본, 시간, 인력 등이 소요되며, 소프트웨어의 검색과 명세가 자동화 되어 있지 않으면, 재사용 가능한 소프트웨어를 채기 위한 시간의 양이 증가된다.

2. 빌딩 블록 방법에 의한 소프트웨어 개발 절차

빌딩 블록 재사용 방법에 의한 소프트웨어 개발 절차는 다음과 같이 수행된다[Ruben Prieto-Diaz and Peter Freeman, 1987; Samuel T.외 1인, 1988; Ted Biggerstaff외 1인, 1987].



[그림 1] 빌딩 블록 재사용을 이용한 소프트웨어 개발

(1) 선택(Selection)

선택은 잠재적으로 응용 가능한 구성 요소를 질의어를 기반으로 한 검색이나 브라우징(browsing)으로 식별하는데, 이것은 재사용 가능한 구성 요소들을 라이브러리에 저장하기 위함이다.

(2) 개조(Adaptation)

선택된 구성 요소는 현 프로젝트의 요구 사항에 부합되도록 수정할 필요가 있다. 개조는 현 프로젝트의 요구에 부합시키기 위하여 소프트웨어 개발자의 요구에 따라 만들거나(customization), 조정(tuning), 확장(extension)하는 작업을 의미하며, 개조 작업의 대부분은 재사용자에 의해 수행된다.

(3) 결합(Assembly)

구성 요소가 검색, 생성되면, 이들은 완전한 시스템 기술에 따라서 구성되어야 한다. 결합은 편집(editing), 링킹(linking) 뿐만 아니라 통합된 시스템을 생산하기 위해 필요한 부수적인 구성요소 개발을 포함한다.

(4) 등록(Cataloging)

Component와 결합된 시스템은 앞으로의 프로젝트에서 재사용 가능하므로 재사용 라이브러리에 등록되어야 한다. 등록은 분류(classification), 새 구성 요소의 저장, 라이브러리 접근 메카니즘을 통한 정보 검색을 하기 위해 필요한 라이브러리 유지보수 운영을 포함한다.

(5) 평가(Assessment)

평가는 다른 행위를 안내하고 한 곳에 모으는 것을 위해 필요한 타당성(validation)과 검증(verification)을 포함한다.

성공적인 결합은 부수적인 개조를 요구할 수도 있으며, 선택은 구성 요소의 개조와 결합성의 실현 정도에 의해 영향을 받는다. 개조를 위해 시험적인 결합과 평가를 요구할 수도 있다. 기본적인 선택, 개조, 결합, 등록이 반복된다.

본 논문에서의 구성 요소는 비 절차적(non-procedural) 언어인 dbaseⅢ 프로그램을 모듈로 구성한다. 비 절차적 언어는 프로그램 생산성에서 절차적 언어보다 각 프로그램 작성의 실행 순서의 지정이 쉽고, 이해가 쉬우며, 프로그램의 개조가 절차적 언어보다 효율적이다.

유사한 모듈의 선택은 분류 문제이며, 유사성의 정도는 어떻게 집합(collection)을 조직하느냐에 따라 정해진다. 그러므로 집합의 조직과 선택은 이 모델에 있어서 중요하며, 분류 구조는 재사용 과정에 있어서 매우 중요하다.

3. 빌딩 블록 방법에서의 모듈 분류

소프트웨어를 재사용하는 데 필요한 개조와 이해에 요구된 노력을 감소시키는 방법은 모듈의 집합을 분류하는 것이다. 만일 집합이 소프트웨어 요구에 정의된 속성에 의해 조직되면, 상관없는 모듈을 검색할 가능성은 줄어든다. 그러므로 잘 정의된 분류 구조는 효율적인 검색 시스템에서 매우 중요하다.

분류는 한정된 영역안에서 비슷한 성질의 모듈들을 하나의 그룹으로 묶고, 그 모듈과 그룹 혹은 그룹들 사이의 관련성을 표현하는 과정이다[Ruben Prieto-Diaz and Peter Freeman, 1987].

분류 구조에는 두가지 방법이 있다. Enumerative 분류는 모듈들의 집합을 좁은 클래스로 분할해가면서 그들 사이의 계층성을 표현하고, Facet 분류에서는 대상들의 공통적인 측면의 값들을 모아 facet를 구성한 뒤 이러한 Facet 들에서 해당 소프트웨어에 알맞는 모듈들을 선택하여 합성함으로써 특정 프로그램을 구성한다.

Enumerative 분류 방법은 모듈들 간의 관련성을 표현하기에는 좋으나 새로운 모듈을 추가하기가 어렵다.

Facet 분류 방법은 모듈들의 집합이 지속적으로 확장되는 경우에도 쉽게 적응할 수 있으나 모듈들 간의 계층성을 효과적으로 표현하지 못하는 단점을 가지고 있다. Facet 구조는 재사용 모듈의 집합이 상당히 크고, 계속 증가하는 경우에 매우 유용하며, 각 그룹에 유사한 모듈이 많을 때 그 활용도가 높다. 본 논문에서는 소프트웨어 집합을 등록하고 선택하기 위해 facet 분류에 기반을 둔 라이브러리 시스템을 개발한다.

Ⅲ. 재사용 모듈의 구성을 위한 시스템의 설계

1. 시스템 모델

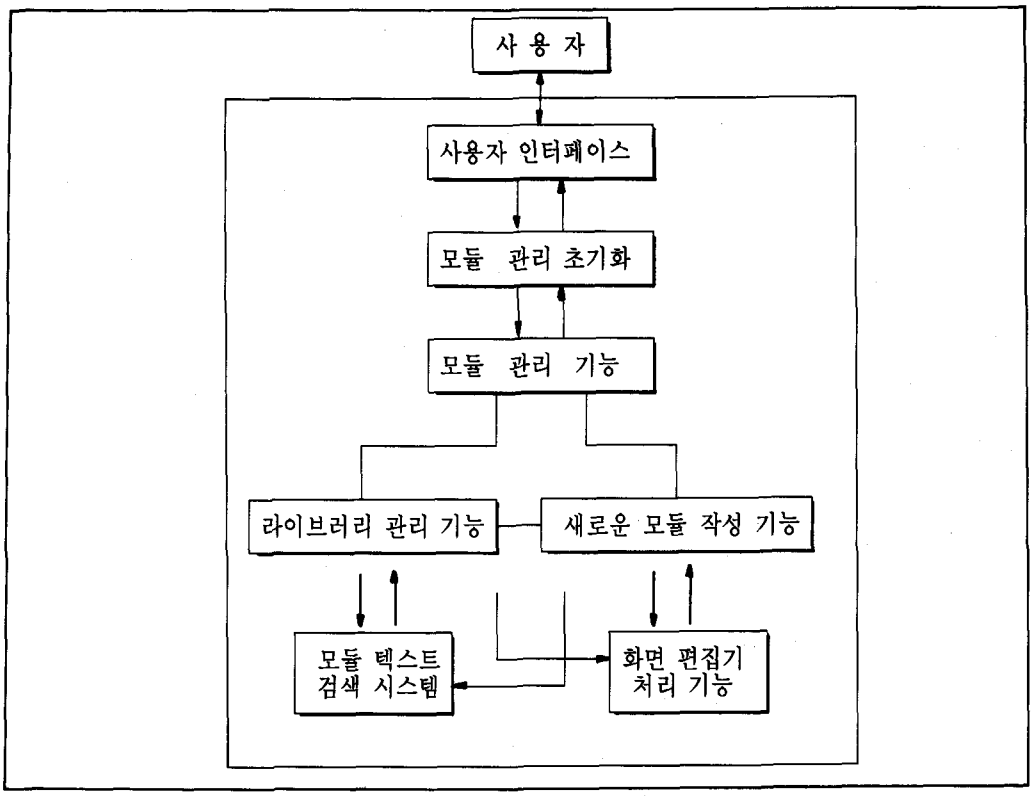
본 논문에서 개발한 시스템 모델의 구성도는 다음 [그림 2]와 같다.

2. 시스템 모델의 기능

2.1 모듈 텍스트 검색 시스템

모듈 텍스트 검색 시스템은 시스템 모델의 전반적인 기능 중 가장 중요한 기능으로써, 모듈의 라이브러리를 초기화 하여 주는 기능과 라이브러리에 모듈의 추가, 삭제 및 검색할 수 있는 기능을 지원하도록 한다. 이 기능에 의해서 처리되는 모듈은 실제의 원문(text)이 된다. 그러므로 데이타의 중복을 최소화하여 실제의 데이타를 저장한 데이타베이스의 성격보다도 질의어를 이용하여 원문속의 핵심 단어를 가지고 원하는 모듈을 검색하는 정보 검색 시스템의 성격을 가진다[Susan P. Arnold외 1인, 1987; W. B. Frakes외 1인, 1987].

모듈의 검색 기능을 위하여 모듈 자체 내에 모듈에 대한 정보 또는 재사용 과정에 도움이 될 수 있는 주석을 포함한다. 즉, 모듈의 소재, 모듈의 복잡도, 재사용에 필요한 절차나 주의 사항 그리고 관련 문서 등의 정보를 모듈 내에 작성한다. 모듈의 검색시에는 이러한 주석 또는 모듈에 기술된 특정 명령어를 가지고 찾도록 한다. 이를 위한 질의어는 사용자가 모듈을 찾는데 필요한 단어(word)를 사용하여 배열로 작성하도록 설계하고, 검색 기능에서 각 단어를 가지고 이에 관련된 모듈을 찾도록 한다. 모듈의 이름은 12자 이내에서 사용자가 임의로 설정하여 주면된다. 모듈 텍스트 검색 시스템은 다음과 같은 서비스 기능을 가진다.



[그림 2] 시스템 모델의 구성도

- initialize : 라이브러리를 초기화 시킨다.
- addModule : 모듈을 추가 기능을 지원한다.
- searchModules : 모듈을 검색 기능을 지원한다.
- removeModule : 모듈을 삭제 기능을 지원한다.

2.2 화면 편집기 기능

이 기능은 일반적인 화면 편집기에서 제공하는 기능을 가진다. 화면 편집기의 기능은 다음에 설명할 라이브러리 관리 기능이나 새로운 모듈 작성 기능에 의한 모듈 수정시에 이용된다.

편집용 화면 크기 정의, 종료는 사용자 질의어를 이용하여 사용자가 직접 작성하도록 설계한다.

2.3 모듈 관리 초기화

라이브러리 관리 기능과 새로운 모듈 작성 기능을 시작하기 위한 처리절차와 화면을 제공한다.

2.4 모듈 관리 기능

1) 라이브러리 관리 기능

라이브러리에 새로운 모듈을 등록하고, 이의 복사, 재명명, 편집, 조회 작업을 하기 위한 기능을 지원한다. 이러한 기능을 위한 질의어, 질의 사항, 화면 등도 아래의 각 기능에 추가시키며, 모듈 텍스트 검색 시스템의 기능을 이용하여 처리한다. 라이브러리 관리 기능은 다음과 같은 서비스 기능을 이용할 수 있도록 지원한다.

- copyLib : 모듈을 라이브러리에 복사한다.

- inputLib : 모듈을 라이브러리에 등록한다.
- deleteLib : 모듈을 라이브러리에서 삭제한다.
- renameLib : 라이브러리에 있는 모듈명을 수정한다.
- updateLib : 라이브러리에 있는 모듈을 수정한다.

2) 새로운 모듈 작성 기능

라이브러리에 등록된 모듈을 재사용자의 필요에 따라 검색하고 수정하여 결합하는 기능을 주 기능으로 한다. 이외에 모듈 재명명, 결합된 모듈의 등록, 모듈 결합을 하기 위한 사용자 질의어에 대한 주석, 모듈을 결합하는 데 필요한 화면과 질의어, 이의 처리절차 등을 설계하며, 모듈 텍스트 검색 시스템의 기능을 이용하여 처리한다. 새로운 모듈 작성 기능은 다음과 같은 기능을 이용할 수 있도록 지원한다.

- nameLib : 라이브러리에 있는 모듈 이름을 수정한다.
- retriveLib : 질의어를 사용하여 라이브러리에서 조건에 맞는 모듈을 검색한다.
- registLib : 새로이 수정된 모듈을 라이브러리에 등록한다.
- coneditLib : 모듈의 결합을 위하여 필요한 질의어를 이용하여 모듈을 결합시키기 위한 명령어를 재사용자가 작성하며 주석은 화면에서 제공된다.
- conexecLib : 모듈을 결합시키기 위한 명령어를 실행시킨다.
- editLib : 검색과 결합 작업에 의하여 변형된 모듈의 내용을 재사용 목적에 맞도록 수정 조회한다.

Ⅳ. 재사용 모듈 구성을 위한 시스템의 구현과 분석

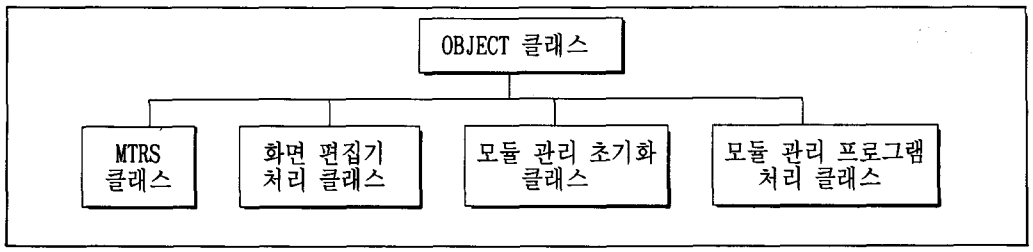
1. RAMIS의 구성

본 논문에서는 객체 지향 프로그래밍 언어인 Smalltalk를 사용하여, 재사용성 모듈을 구현하는 시스템(RMIS, Reusable Module Implementation System)을 구현한다. Smalltalk는 재사용 구성 요소를 발견하고, 이 구성 요소를 변형하여 수정하기 위한 환경을 가지고 있다[Gerald Jones, 1987].

RMIS는 Smalltalk에서 제공하는 강력한 사용자 환경(user environment)과 메시지 전송, 상속성, 동적 연결 등의 기본 특성을 이용하여 재사용성 모듈을 구현한다. 이미 개발되어 저장된 모듈이 모듈화가 잘 되어있고, 빌딩 블록 재사용 개념에 충실한 특성을 가지고 있으면, Smalltalk와의 명확한 인터페이스를 통하여 사용될 수 있다. 또한, 자료추상화를 이용한 자료구조에서의 알고리즘 재사용과 view/controller 등을 통한 응용(application)에서의 프레임워크(framework)를 재사용할 수 있다.

RMIS의 구성은 장에서 설명한 모듈 텍스트 검색 시스템(MTRS, Module Text Retrieval System)과 이를 기반으로 새로운 모듈의 추가, 재명명, 복사, 삭제, 수정 작업을 할 수 있는 라이브러리 관리 클래스를 포함한다. 또한, 기존의 모듈을 검색하고 모듈의 내용을 변경, 결합하여 새로운 모듈을 작성한 후 라이브러리에 등록할 수 있는 새로운 모듈 작성 클래스를 구현한다.

RMIS의 목적은 재사용자의 주문에 따라 소프트웨어를 작성하고, 이의 등록과 검색을 하기 위한 환경을 구축하는 것이다. Smalltalk에서는 객체가 클래스의 인스탄스이고, 클래스는 인스탄스와 특정한 종류의 객체를 사용하고 구축하는데 필요한 모든 정보를 제공한다. 클래스 내의 처리 방식(methods)은 프로시듀어로서, 클래스에 메시지를 전송



[그림 3] RMIS의 클래스 구성도

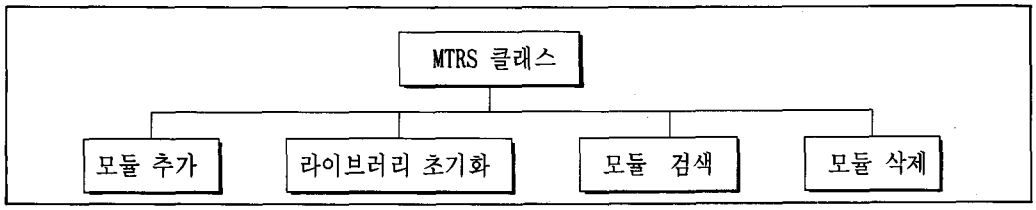
하면 처리 방식을 호출하여 처리한다. 클래스 상의 처리 방식은 그 실행 중에 사용할 임시 변수를 할당한다. [그림 3]는 RMIS의 구성을 나타낸다.

RMIS의 구성은 위에서 설명한 기본적인 구성이외에 모듈을 수정하거나 간단한 질의어를 사용하기 위한 화면 편집기를 [그림 3]에서와 같이 포함한다. 화면 편집기는 Smalltalk 시스템이 가지고 있는 filbrwsr 클래스를 이용한다. Set과 Dictionary 클래스는 Smalltalk 시스템 상의 Collection 클래스의 하위 클래스로서 라이브러리의 구축에 사용된다.

1.1 MTRS 클래스

MTRS 클래스는 재사용 소프트웨어를 등록할 수 있는 라이브러리를 생성하여 초기화하여 주고, 이 라이브러리에 모듈을 추가, 검색, 삭제할 수 있는 처리 방식을 택하게 된다.

MTRS 클래스를 사용하기 위한 단어의 인덱스



[그림 4] MTRS의 처리방식

와 질의어를 생성할 수 있도록 하며, 이는 모듈관리 프로그램에서 생성 처리한다. 인덱스와 질의어를 생성하기 위한 메시지로 initialize를 사용하여 원문과 모듈명을 저장할 set과 dictionary를 초기화시킨다. MTRS 클래스에서 구현한 라이브러리는 Smalltalk 상의 Set 클래스와 Dictionary 클래스를 사용하여 원문과 모듈명을 기록한다. Set 클래스는 중복을 제거하여 객체를 저장하고, Dictionary 클래스는 키 값에 의해서 객체를 저장하고 검색하는 기능을 가진다. 모듈의 검색에서는 asSet 메시지를 사용하여 검색하고자 하는 모듈명을 얻는다. 모듈의 추가는 add:, addWord: 메시지, 검색은 occurrencesOf: 메시지, 삭제는 remove: 메시지를 이용하여 처리한다.

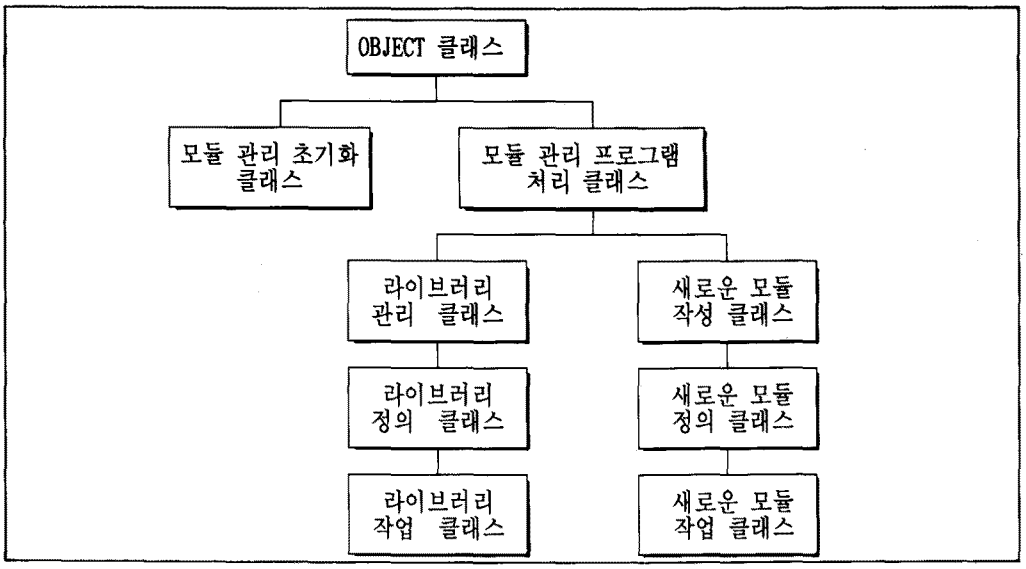
1.2. 모듈 관리 프로그램

모듈 관리 프로그램은 Object 클래스의 하위 클래스로서 모듈 관리 프로그램 처리 클래스를 가진다.

1.2.1 모듈 관리프로그램의 처리 클래스

1) 라이브러리 관리 클래스

라이브러리의 관리를 위한 실제적인 모듈 복사, 추가, 삭제, 재명명,편집과 조회 작업을 할 수 있



[그림 5] 모듈 관리 프로그램을 위한 클래스 계층 구성도

는 수정 작업에 대한 사용자 화면과 도움말, 질의어에 대한 처리 방식과 지역 변수를 가진 클래스이다. 각 처리 방식에서는 Smalltalk 윈도우(window) 중의 하나인 Prompter를 사용하여 재사용자가 필요로 하는 질의 사항을 처리한다. Ⅲ장에서 설명한 라이브러리 관리 기능은 라이브러리 관리 클래스의 처리 방식으로 구현된다.

(1) 모듈 복사

모듈명을 입력 받고, 이 모듈명에 대한 모듈을 새로운 이름으로 라이브러리에 등록하기 위한 화면과 질의어를 처리한다.

(2) 모듈 등록

라이브러리에 새로운 모듈을 추가 시키며, MTRS의 모듈 추가 처리 방식을 이용한다.

(3) 모듈 삭제

라이브러리에 있는 기존의 모듈을 삭제한다. 이 처리 방식은 MTRS의 모듈 삭제 처리 방식을 이용하여 처리한다.

(4) 모듈 재명명

모듈 복사 처리 방식을 이용하여 새로운 모듈을 라이브러리에 복사하여 저장하고, 기존의 모듈을 라이브러리에서 삭제한다. 새로이 저장된 모듈 이름은 사용자가 질의어를 통하여 제공한다.

(5) 모듈 수정

Single Pane Window를 사용하여 모듈 수정에 필요한 도움말과 재사용을 위한 라이브러리를 효과적으로 이용할 수 있는 질의어를 제공한다. 모듈의 조회나 수정은 fil-brwsr 클래스를 이용한다.

○ 라이브러리 정의 클래스

라이브러리 관리 클래스를 실행하기 위해 필요한 사용자 메뉴 화면과 재사용자가 선택한 작업을 수행하기 위한 처리 방식과 지역변수를 가진다.

```
copyLib
    |input output stringi stringo|
    input  output := '                    '.
    { Move space to input output }
    Rmis reScreen. { Update screen_color and screen_form}
    input := Prompter
    prompt: 'Old Module Name ?'.
    { Do ask a question and input accept }
    output := Prompter
    prompt: 'New Module Name ?'.
    { Do ask a question and output accept }
    stringi := File pathname: input.
    stringo := File pathname: output.
    [ stringi atEnd]
        whileFalse: [stringo  nextPut: stringi next].
    Index addModule: output.
    stringo close.
        stringi close.
```

[그림 6] 모듈 복사를 위한 알고리즘

```
updateLib
    |string|
    string := '                    '. { Move space to string }
    Rmis reScreen. { Update screen_color and screen_form }
    LearnDispatcher := TextEditor windowLabeled:
      'Comment Message' frame: (170 @ 280 extent 400 @ 500).
    LearnDispatcher := nextPutAll:
      'Editing or Displaying Job end.'; cr.
    LearnDispatcher := nextPutAll: '=> "Start new" execute'; cr.
    string Prompter { Do ask a question and string accept }
    prompt: 'Updating or displaying Module name ?'.
    FileBrowser new openOn: (File pathName: string).
    close string.
```

[그림 7] 모듈 수정을 위한 알고리즘

ㅇ라이브러리 작업 클래스

라이브러리 정의 클래스 처리를 위한 작업을 수행하는 클래스로서, 라이브러리 관리 클래스와 라이브러리 정의 클래스 특성을 상속 받아 라이브러리를 관리하기 위한 처리 방식과 지역 변수를 가진다.

2) 새로운 모듈 작성 클래스

라이브러리에 등록되어 있는 모듈을 검색하고, 검색된 모듈을 개발하려는 소프트웨어의 요건에 맞도록 결합하고, 이에 대한 등록, 편집, 조회, 재명명 작업을 수행한다. 재사용자가 필요로 하는 모듈을 결합하기 위한 질의어 작성은 filbrwsr 클래스를 사용하여 처리한다. 모듈의 결합은 Smalltalk 시스템의 ScreenDispatcher 클래스에 있는 free: toExecute: withPause 메시지를 사용한다.

ㅇ새로운 모듈 정의 클래스

새로운 모듈 작성 클래스를 실행하기 위해 필요한 사용자 메뉴 화면과 재사용자가 선택한 작업을 수행하기 위한 처리 방식과 지역 변수를 가진다.

ㅇ새로운 모듈 작업 클래스

새로운 모듈 정의 클래스 처리를 위한 작업을 수행하는 클래스로서, 새로운 모듈 작성 클래스와 새로운 모듈 정의 클래스의 특성을 상속 받아 라이브러리의 모듈을 관리하기 위한 처리 방식과 지역 변수를 가진다.

1.2.2 모듈 관리 초기화 클래스

라이브러리 관리 클래스와 새로운 모듈 작성 클래스를 관리하는 사용자 메뉴와 이의 처리에 필요한 클래스이다.

```
select
  |s|
  Rmis reScreen. { Update screen_color and screen_form }
  Rmis menuScreen. { Menu screen display }
  s := Prompter
  prompt: 'Select to above menu'.
  { Do aska question and s accept }
  (s = '1')
      ifTrue: [Linit new]. { Do  Library_mamagement }
  (s = '2')
      ifTrue: [Ninit new]. { Do New_module_making }
  Display end_message.
  Scheduler systemDispatcher redraw.
```

[그림 8] 모듈 관리 초기화 클래스의 알고리즘

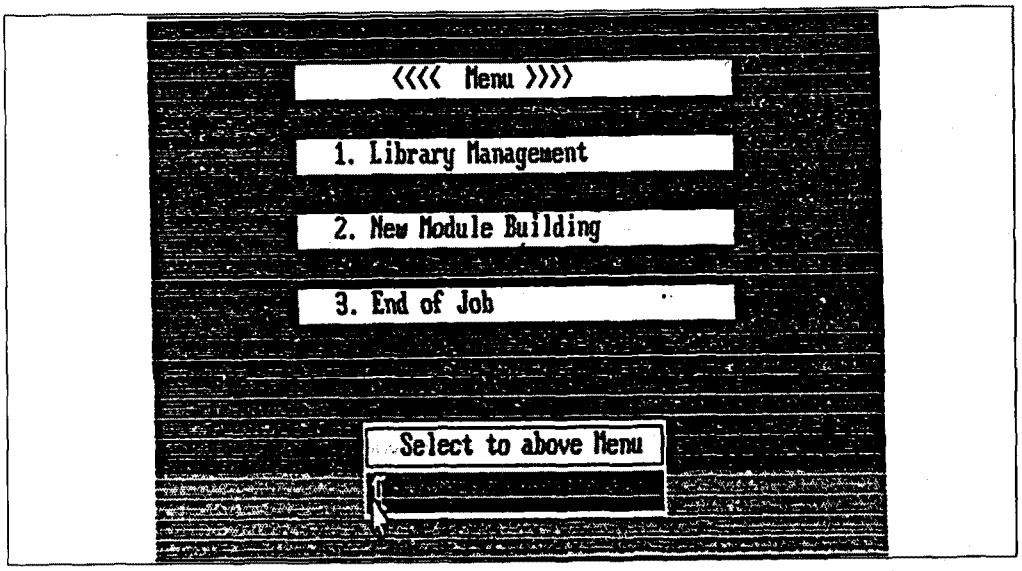
```
Start new
```

[그림 9] 모듈 관리 프로그램 시작 클래스의 처리 명령

2. RAMIS의 구현

본 논문에서 구현된 RMIS의 초기 실행 화면은 [그림 10]과 같으며, 모듈 관리 프로그램시작 클래스를 실행함으로써 화면상에서 처리된다. 이의 실행은 [그림 9]와 같이 Smalltalk 시스템에 실행 메시지를 전송한다.

"Start"는 모듈 관리 프로그램 시작 클래스의 이



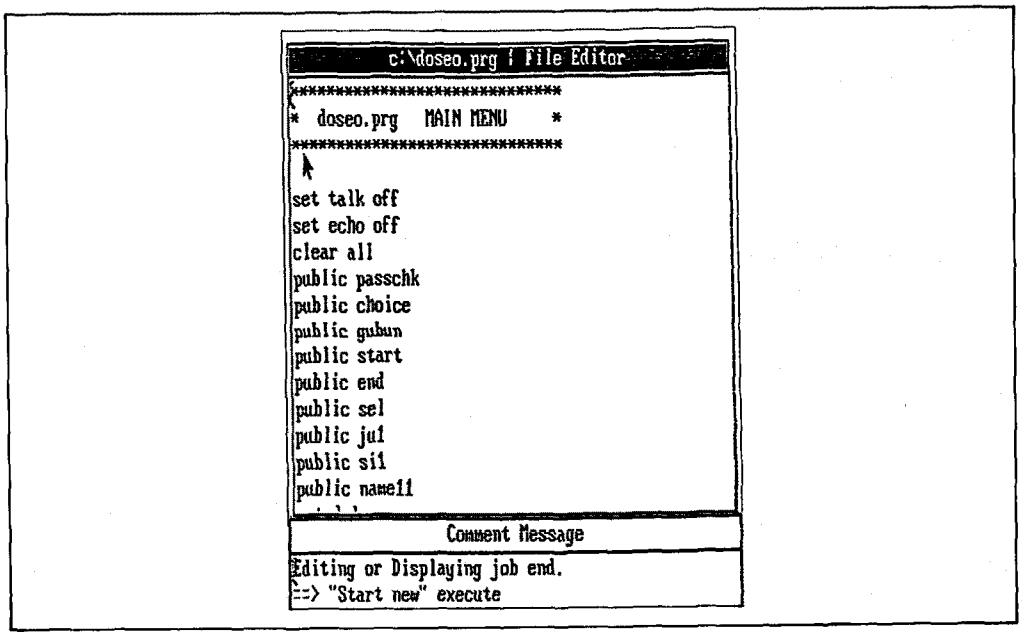
[그림 10] 초기화면

름이며, "new"는 클래스에 대하여 새로운 객체를 생성시키기 위한 메시지이다.

초기 화면에서 1번 라이브러리 관리를 선택하여 화면에서 모듈 수정 작업을 선택하면, 화면에 수정할 모듈 이름을 입력하기 위한 윈도우가 설정된다. 수정할 모듈 이름이 입력되면, 모듈을 수정하기 위한 화면과 이를 위한 질의어 주석이 [그림 11]과 같이 표시된다.

3. 결과 분석

본 논문에서 구현한 소프트웨어 라이브러리에는 dBASEⅢ로 작성된 도서 관리 프로그램에 관계되는



[그림 11] 모듈수정작업의 화면

50개의 모듈을 사용하여 8개 프로그램으로 구성된 성적 프로그램을 개발하였다. 개발된 각 프로그램의 라인 수는 메뉴 프로그램 171 라인, 입력 프로그램 325 라인, 조회 프로그램 231 라인, 삭제 프로그램 261 라인, 수정 프로그램의 434 라인과 두개의 출력 프로그램이 216 라인과 285 라인으로 작성한 후 이들에 대한 재사용율을 분석하여 효율성을 입증하였다. 사용된 모듈의 종류는 〈표 2〉와 같다.

〈표 2〉 모듈의 종류

종 류	메뉴	입력	조회	삭제	수정	출력	검색	백업	기타	합계
모 듈	3	3	2	3	3	6	2	2	26	50
라인수 평 균	132	88	240	205	254	202	149	59	31	

이들 모델의 원시 프로그램들을 빌딩 블록 방법으로 작성하여 원시 코드를 직접 코딩한 라인(line) 수와 재사용된 라인 수로 재사용 비율을 계산하였다.

본 논문에서 사용된 모듈의 재사용율은 다음과 같이 계산한다.

$$\text{재사용율} = \frac{\text{재사용된 라인 수}}{\text{전체 라인 수}} \cdots\cdots (1)$$

재사용된 모듈 중의 일부는 수정없이 사용된 것도 있으나, 일부는 수정되어서 사용된다. 재사용된 모듈내에서 수정없이 사용된 순수 재사용율은 다음과 같이 계산한다.

$$\text{순수 재사용율} = \frac{\text{수정안된 라인 수}}{\text{재사용된 라인 수}} \cdots\cdots (2)$$

재사용된 모듈내에서 수정되어 사용된 부분 수정율은 다음과 같다.

$$\text{부분 수정율} = \frac{\text{수정된 라인 수}}{\text{재사용된 라인 수}} \cdots\cdots (3)$$

본 논문에서 사용한 모델을 실행하여 분석한 결과 재사용율은 54 %, 재사용된 모듈에서 수정없이 사용된 순수 재사용율은 37 %, 수정된 후 사용된 부분 수정율은 71 %로 나타났다. 수작업에 의한 방법과 RMIS에 의한 방법의 결과가 (1), (2), (3) 식의 평가에 의하면 비슷한 결과이지만 라이브러리에 모듈의 수가 많아지고, 시간이 지날수록 재사용자의 측면에서는 수작업을 통한 방법보다 시스템 개발, 유지보수 등의 작업을 효율적으로 할 수 있는 환경을 지원하는 도구를 사용하는 것이 라이브러리에 저장된 모듈을 효율적으로 재사용할 수 있었으며, 원시프로그램을 새로이 작성하는 것 보다 본 논문에서 구현된 시스템을 사용하는 것이 모듈의 검색과 개조 등의 작업과 유지보수를 효율적으로 할 수 있음이 입증되었다. 원시프로그램을 작성할 때 본 모델을 적용하여 분석해 본 결과 라이브러리에 저장된 재사용 가능한 모듈의 양과 종류, 새로 개발하려는 모듈의 특성, 라이브러리에 기억된 모듈과 새로 개발하려는 소프트웨어의 관련성에 따라 비율은 달라지겠지만, 재사용 방법에 의한 소프트웨어 개발이 원시 코드로 직접 표기하는 것보다 효과적으로 나타났다.

V. 결 론

본 논문은 빌딩 블록 방식에 의한 재사용 기법

을 사용하고, Smalltalk 언어를 이용하여 라이브러리에 저장된 소프트웨어를 재사용하여 새로운 원시 프로그램을 작성하는 도구인 RMIS 모델을 구현하였다. 본 논문에서는 50 개의 실험적 프로그램을 사용하여 RMIS에 의한 원시 프로그램을 작성한 후 분석한 결과 재사용율 54 %, 순수 재사용율 37 %, 부분 수정율 71 %인 것으로 나타났다. 이는 논문 [Robert G. Lanergan and Charles A. Grasso, 1984]에서의 60 %, [T. Capers Jones, 1984] 논문에서의 결과인 75 %와 근접한 결과이며, 원시코드를 직접 작성하는 것보다 빌딩 블록 재사용 방법을 이용한 프로그램 작성 방법이 더 효과적인 것으로 입증되었다.

앞으로의 연구 과제는 현재 시스템의 제약 사항을 개선 및 보완해 나가며, 소프트웨어 개발 과정의 생산성을 더욱 향상시키기 위한 질 좋은 라이브러리의 구축과 이들을 효과적으로 사용할 수 있는 도구에 대한 추가적인 연구와 재사용 시스템을 평가하는 방법에 관한 연구가 필요하다.

참 고 문 헌

이영곤, "재사용을 위한 소프트웨어 COMPONENT의 식별에 관한 연구", 과기원 석사학위논문, 35p, 1988.

Adele Goldberg and David Robson, "*Smalltalk-80 : The Language and Its Implementation*", Addison-Wesley, 1988.

B. Meyer, "Reusability : The case for Object-Oriented Design", *IEEE Software*, pp. 50-64, March 1987.

Bruce A. Burton, Rhonda Wienk Aragon, Stephen A. Baily, Kenneth D. Koehler, and Lauren A. Mayes, "The Reusable Softwar Library" , *IEEE Software*, pp. 25-33, July 1987.

Bruce Barnes, Thomas Durek, John Gaffney, and Arthur Pyster, *Proceedings of the Workshop on Software Reusability and Maintainability*, October 1987.

George Copeland and David Maier, "Making Smalltalk Database System", *ACM SIGMOD*, pp. 316-325, 1985.

Gerald Jones, "Methodelogy/Environment Support for Reusability", *Procedings of the Workshop on Software Reusability and Maintainability*, October 1987.

Jakob Nielsen and John T. Richards, "The Experience of Learning and Using Smalltalk", *IEEE Software*, pp. 73-77, May 1989.

Kyo C. Kang, "A Reuse-Based Software Development Methodology", *Proceedings of the Workshop on Software Reusability and Maintainability*, October 1987.

L. Peter Deutsch, "Reusability in the Smalltalk-80 Programming System", *ITT Proceedings of the Workshop on Reusability in Programming*, pp. 72-76, 1983.

Lewis J. Pinson and Richard S. Wiener, "*An Introduction to Object-Oriented Programming and Smalltalk*", Addison-Wesley, 1988.

J. Diederich and J. Milton, "Experimental Prototyping in Smalltalk", *IEEE Software*, pp. 50-64, May 1987.

Manfred Lenz, Hans Albrecht, and Peter F. Wolf, "Software Reusethrough Building Blocks", *IEEE Software*, pp. 34-42, July 1987.

Patrick A V Hall, "Software components and reuse-getting more out of your code", *Information and Software Technology*, Vol. 29, No. 1, January/ February 1987.

Peter Freeman, "Reusable Software Engineering : Concepts and Research directions", *ITT Proceedings of the Workshop on Reusability in Programming*, pp. 129-137, 1983.

Robert G. Lanergan and Charles A. Grasso, "Software Engineering with Reusable Designs and Code", *IEEE Trans. Software Eng.*, Vol. SE-10, No. 5, pp. 498-501, September 1984.

Ruben Prieto-Diaz and Peter Freeman, "Classifying Software for Reusability", *IEEE Software*, pp. 6-16, January 1987.

Samuel T. Redwine Jr. and William E. Riddle, "*Software Reuse Process*", *Software Productivity Consortium Reston*, Virginia, USA.

Scott N. Woodfield, David W. Embly, and Del T. Scott,"CanProgrammers Reuse Software ?", *IEEE Software*, pp. 52-59, July 1987.

Smalltalk/V, digitalk inc., 1986.

Susan P. Arnold and Stephen L. Stepoway, "THE REUSE SYSTEM : CATALOGING AND RETRIEVAL OF REUSABLE SOFTWARE", *Proceedings of COMPCON S'87*, pp. 376-379, 1987.

T. Capers Jones, "Reusability in Programming : A Survey of the State of the Art", *IEEE Trans. Software Eng.*, Vol. SE-10, No, 5, pp. 488-493, September 1984.

Ted Biggerstaff and Charles Richer, "Reusability Framework, Assessment, and Directions", *IEEE Software*, pp. 41-48, March 1987.

Ted J. Biggerstaff, "Foreword", *IEEE Trans. Software Eng.*, Vol. SE-10, No. 5, pp. 474-476, September 1984.

The Xerox Learning Research Group, "The Smalltalk-80 System", *Byte*, Vol. 6, No. 8, pp. 36-48, August 1981.

Thomas A. Standish, "An Essay on Software Reuse", *IEEE Trans. Software Eng.*, Vol. SE-10, No. 5, pp. 494-497, September 1984.

Victor R. Basili, "Viewing Maintenance as Reuse-Oriented Software Development", *IEEE Software*, pp. 19-25, January 1990.

W. B. Frakes and B. A. Nejmeh, "An Information System for Software Reuse", *Proceedings of the Tenth Minnowbook Workshop on Software Reuse*, 1987.

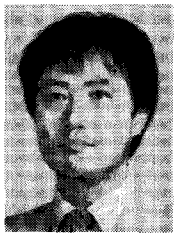
Will Tracz, "Software Reuse : Motivators and Inhibitors", *Proceedings of COMPCON S'87*, pp. 358-363, 1987.

Will Tracz, "Software Reuse Myths", *ACM SIGSOFT Software Engineering Notes*, Vol. 13, No. 1, pp. 17-21, January 1988.

◇ 저자소개 ◇

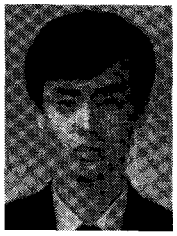
공동저자 장덕철은 고려대학교 대학원에서 경영정보 시스템론 전공 석박사 학위를 취득하고 미국 캘리포니아 버클리 대학 객원 교수로 재직하였으며, 현재는 광운대학교 전자계산학과 교수, 전산원장으로 재직하고 있다.

연구 관심분야는 Expert Decision Support System, 소프트웨어 공학(특히, 소프트웨어 재사용, 소프트웨어 구성 관리)이다.



공동저자 김홍진은 광운대학교 전자계산학과에서 학사와 석사과정을 수료하고 박사 과정중이며, 현재 경원 전문 대학에 재직하고 있다.

연구관심 분야는 소프트웨어 재사용, 시스템 프로그램 중 입출력 제어이다.



공동저자 오상엽은 광운대학교 대학원 전자계산학과 대학원을 졸업(이학 석사)하고, 현재는 동 대학원 박사과정에 재학하고 있다. 연구 관심분야는 소프트웨어 재사용, 객체 지향 프로그래밍, 소프트웨어 구성 관리이다.