韓國OA學會誌
第2卷 第2號, 1996. 2

소프트웨어 재사용율에 근거한 소프트웨어의 구성요소의 저장과 검색 시스템의 설계와 구현

The Design and Implementation of Storage and Retrleval System of Software Components based on Software reuse rates

오 상 엽*, 최 우 승*, 김 홍 진*

요 약

본 논문에서는 소프트웨어 구성 요소(component)를 저장, 검색할 수 있는 라이브러리를 제안하고, 이를 관리하고 제어하기 위한 사용자 인터페이스를 설계 및 구현하였다. 이 라이브러리에 버전 제어를 위한 구성 요소를 적용하고, 구성 요소를 정량적으로 테스팅하여 구성 요소를 사용자가 선택할 때에 정량적 비율이 높은 구성 요소를 선택할 수 있다. 이 시스템은 버전 제어를 위한 기반 시스템으로 사용되며, 이러한 사용자 인터페이스와 라이브러리를 Smalltalk를 사용하여 구현하였다.

ABSTRACT

In this paper library is proposed to store and retrieve the software components, in addition to that designed and implemented the user interface for management and control of this library.

Applying components to this library for version control, component's quantitative testing are processed. In the case of user selection of component's, a high quantative rate components can be selected.

This system is based on version control, it's user interface and library are implemented using Smalltalk.

Ⅰ. 서 론

컴퓨터를 일반적인 정보 처리에 이용하면서 컴퓨터의 이용 범위가 확대되었으며, 보다 복잡한 데이터를 빠르고 쉽게 처리할 수 있도록 발전하고 있다. 초기에는 컴퓨터를 운용 비용 중에서 하드웨어 부분이 70 ~ 80% 정도이었으나 현재는 소프트웨어의 비율이 오히려 더

* 경원전문대학

커지게 되었다.

버전 제어는 변경 이력(history)의 유지가 필수적인 경우에 효과적인 방식이며, 버전을 만드는 작업은 간단한 반면에 동일한 개체에 대한 여러 버전들에 대한 효과적인 표현, 저장, 선택 수단을 필요로 한다[14,15].

버전 제어를 위한 구성 요소를 식별하는 것은 버전 제어에서 기본이 되는 기능이다. 구성 요소들의 집합이 크고, 그 구성 요소들이 여러 응용 분야에 걸쳐 널리 사용되면서 우수한 모듈성을 갖는 환경에서는 필요한 구성 요소를 검색하고 식별하는 것이 중요한 문제로 제기된다[3,10,13].

라이브러리는 구성 요소의 접근, 탐색, 제어, 보안에 용이한 포괄적인 DBMS이며, 사용자가 구성 요소를 생성, 편집, 검증, 합성하기 위한 기능을 제공하며, 확장이 용이해야 한다.

또한, 버전 제어를 위한 검색 시스템은 사용자로 하여금 관심있는 영역만을 탐색할 수 있고, 구성 요소에 대한 정보를 제공하여야 한다.

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 구성 요소를 라이브러리에 저장할 때는 facet 분류 방식을 사용한다.

본 논문에서는 버전 제어를 위한 라이브러리를 개발하고, 이를 사용할 수 있는 사용자 인터페이스를 설계 및 구현한 라이브러리 관리 시스템을 질의어와 사용자 메뉴와 마우스를 사용하여 객체 지향 프로그래밍(OOP, Objcet Oriented Programming) 언어인 Smalltalk로 구현한다.

Ⅱ. 관련 연구

2.1 버전 제어

프로젝트 개발과 유지보수 동안 구성 요소는 변경되며, 이러한 변경(changes)의 집합 즉, 대개 브라우저에 의해서 수행되는 텍스트 상에 있는 오퍼레이션들, cut, paste, insert, delete 등의 작업으로 구성 요소의 한 버전들을(연속적인) 한 버전으로 변환시키는 이력 과정(history step)이 버전 제어이다.

하나의 구성 요소는 시간이 지나고 프로젝트가 계속되는 동안 evolve되며, 이 결과의 구성 요소들을 version, revision, variant, 또는 deviation이라고 한다[2,12,15].

그림 2-1은 한 버전을 새로운 버전으로 변환하는 이력 과정(history step)을 나타낸다.

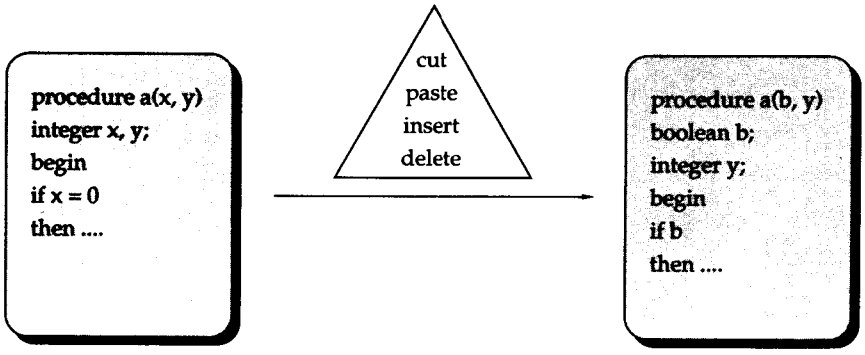


그림 2-1. 이력 과정

변환하는 오퍼레이션들은 많은 시스템에서 이력 과정을 위한 delta의 형태를 제안하기 위해서 삼각형 박스 내에 들어있다.

버전 제어 시스템을 위해서는 다음과 요건을 충족해야 한다.

(1) 가능한 다양한 개체들을 포괄할 수 있어야 한다.
(2) 가능한 다양한 의미론적 관계성을 추적할 수 있어야 한다.
(3) 버전 제어 규칙은 설계자, 사용자, 응용 프로그램등이 공유할 수 있어야 한다.
(4) 버전 제어 규칙은 상황 변동시 탄력적으로 재정의할 수 있어야 한다.

SCCS[9], Make, RCS[4] 등의 버전 제어 시스템들은 프로그램의 변경 이력으로 버전들로 유지하고 필요시, 기존의 버전들을 이용해서 새로운 프로그램 작성을 돕기 위한 도구로 개발되었다.

버전 제어를 위해서는 기본적으로 표준화된 소프트웨어 구성 요소의 작성과 이들을 라이브러리에 체계적으로 저장하여 검색이 용이하도록 하여야 하며, 각 구성 요소들의 분류가 체계적으로 이루어져야 한다[5]. 이것은 버전 제어를 위한 기반이 된다. 또한, 필요한 소프트웨어 구성 요소를 명세하여 검색하고 다른 소프트웨어 구성 요소들과 결합하여 새로운 소프트웨어를 작성하여 소프트웨어의 이력을 효율적으로 관리해야 한다[18].

소프트웨어 구성 요소는 버전 제어를 위하여 작성한 후 라이브러리에 기억시켜 놓은 프로그램 구성 요소이다.

2.2 라이브러리

라이브러리는 구성 요소의 접근, 탐색, 제어, 보안에 용이한 포괄적인 DBMS이며, 사용자가 구성 요소를 생성, 편집, 검증, 합성하기 위한 기능을 제공하며, 확장이 용이해야 한다.

유사한 구성 요소의 선택은 분류 문제이며, 유사성의 정도는 어떻게 집합(collection)을 조직하느냐에 따라 정해진다. 그러므로 집합의 조직과 선택은 이 모델에 있어서 중요하며, 분류 구조는 버전 제어를 위한 라이브러리를 위해서 중요하다.

분류는 한정된 영역안에서 비슷한 성질의 구성 요소들을 하나의 집단(group)으로 묶고, 그 집단과 집단들 사이의 관련성을 표현하는 과정으로 정의한다[3].

소프트웨어 구성 요소의 관련성을 부여하는 방법에 따라 Enumerative와 Facet의 2가지 방법이 있다[10].

Enumerative 분류는 구성 요소들의 집합을 좁은 클래스로 분할해가면서 그들 사이의 계층성을 표현한다. Enumerative 분류 방법은 구성 요소들 간의 관련성을 표현하기 좋으며, 검색 시간이 빠르지만 새로운 구성 요소를 추가하기가 어렵다. Facet 분류에서는 대상들의 공통적인 측면의 값들을 모아 facet를 구성한 뒤 이러한 Facet 들에서 해당 소프트웨어에 알맞는 구성 요소들을 선택하여 합성함으로써 특정 프로그램을 구성한다. Facet 방법은 공통적 성질의 객체들을 모아 Facet을 구성하여 기억시켜놓은 재사용 라이브러리에서 알맞는 요소의 소프트웨어 구성 요소를 선택하여 결합할 수 있도록 한다. 이 방법은 새로운 객체를 추가하기 쉽고 라이브러리 확장에 쉽게 적응되지만 객체들의 계층적 관련성 표현이 어려운 단점이 있다.

또한, 라이브러리는 구성 요소와 연계된 정보를 관리하고, 구성 요소의 등록과 삭제, 구성 요소에 필요한 사항의 수집과 관리 등에 관한 사항도 포함해야 한다.

구성 요소 라이브러리의 예로는 Raytheon Company에서 3200개의 코볼 코드 구성 요소로 라이브러리를 구성하였으며, AT & T Pacific Bell사에서 250명의 개발자에게 공유된 약 1000개의 C 언어 구성 요소를 가지고 라이브러리를 구성하였으며, 이외에도 prieto Diaz의 라이브러리, EIFFEL, LaSSIE, AIRS, ROSE사 등에서도 라이브러리를 구축하였다.

2.3 정보 검색 시스템

검색 시스템은 라이브러리에서 소프트웨어의 구성 요소를 카탈로깅(cataloging)하고 검색(retrieving)하기 위해 사용된다.

버전 제어에 필요한 소프트웨어 구성 요소를 잘 식별하기 위해서는 이들을 보관·관리하고, 유지하며, 필요한 구성 요소를 찾아내는 검색 메카니즘의 성능이 좋아야 한다. 구성 요소의 검색시에는 해당 구성 요소에 관한 정보뿐만아니라 다른 구성 요소와의 관계를 표현하여 검색 능력을 향상시켜야 한다[13]. 그러므로 시스템의 주 기능으로는 구성 요소를 등록시킬 수 있는 라이브러리와 이 라이브러리에서 저장된 구성 요소를 검색하고, 질의어를 이용하여 새로운 구성 요소를 작성하기 위한 도구가 필요하다.

검색 시스템에서의 각 사용자 메뉴(menu)는 일반적으로 키워드(keyword)의 리스트(list)로 구성되어 있고, 이것은 소프트웨어 조직의 특수한 필요성을 반영하기 위하여 정의된다. 많은 사용자 메뉴는 많은 수의 정의된 구성 요소 분류와 직접 관련되어 요구된다. 새로운 구성 요소 형태(type)가 추가되면, 반드시 새로운 사용자 메뉴가 생성 되어야 한다.

검색 시스템은 다양한 사용자 사이에서 같은 의미를 가지는 용어(terminology) 차이를 해결하고자 할때 라이브러리에서 각 텍스트의 유사 단어를 검색하고, 저장된 정보와 사용자의 질의어 사이의 용어 차이 또는 모호성(ambiguitious)를 제거하는데 도움을 주기위하여 검색 과정 동안 사용된다.

라이브러리는 관련된 동의어(synonyms)를 가진 단어나 구를 디렉토리(directory)에 저장하며, 이 구조에 저장 되어있는 기본 단어(primary word)는 모든 메뉴 키워드와 전반적인 조직에 대해 중요 하거나 공통적인 것으로 간주되는 비키워드를 포함한다.

검색 시스템의 전반적인 원리를 유지하기 위하여, 현존 정보 시스템의 능력과 개념은 이용 되어야 한다. 시스템에 의해 요구된 저장(storage)과 검색 메카니즘은 오늘날 이용 가능하다. 현 기술은 정보 시스템의 영역에서 많은 대안을 제공한다.

◎ 정보 검색 시스템(IRS)
◎ 데이타베이스 관리 시스템(DBMS)
◎ 경영 정보 시시템(MIS)
◎ 의사 결정지원 시스템(DSS)

검색 시스템은 낮은 수준(low level)의 지원을 위하여 필요하고, 기본적인 저장과 검색 능력을 위하여 요구된다. MIS, DSS 시스템에서 이용 가능한 부가적인 처리 능력은 검색 시스템 자체가 부가적인 처리를 제공하므로 필요하지 않다.

다른 주요한 고려 사항은 처리되는 데이타의 형태(type)이다. 검색 시스템에 의해 처리되는 정보는 사실상 원문(textual)이며, 정보검색(IR, Information Retrieval) 시스템은 텍스트 정보의 처리를 위주로 하며, 대부분의 다른 시스템은 실제의 데이타를 다룬다. 이러한 고려하에 IR 시스템이 검색 시스템을 위해 선택된다. 일반적인 기억 구조와 인덱싱, IR 시스템의 질의어 과정도 고려되어야 한다[14,16].

IR 시스템에 있는 각 항목(entry)은 하나의 문서(document)로서 식별된다. 각 document는 특유의 단어(words)들로 구성 되어있고, 이것은 분리된 소절이나 문장으로 나누어 질 수도 있고, 안 나누어 질수도 있다. IR 시스템의 능력(capabilities)을 이용하기 위하여 검색 시스템 안에서 소프트웨어 구성 요소 정보는 IR 시스템의 구조와 일치해야 한다. 대부분의 상업적으로 이용 가능한 IR 시스템은 데이타 항목 조직을 위해 역 화일(inverted file) 구조를 사용한다. 이 접근은 개개의 저장된 항목(items)에 대해 빠른 접근(access)를 제공한다. 이의 향상된 버전은 부가적인 단어의 위치를 ㅊ는 정보를 제공한다. 이 정보는 문서, 절, 소절, 단어에 의해 정확한 문서를 찾을 수 있다.

역 인덱스의 향상된 버전은 검색 시스템에서 두가지 명백한 문제를 제거하는 데 도움을 준다. ㅊ째는, 검색을 위한 단어와 구(phrases)의 사용과 관련된다. 검색 시스템은 반드시 IR 시스템에 대한 질의어 요구(request)에서 단어와 구를 명시하여야 한다.

향상된 인덱스에 의해 제거되는 두번째 문제는 검색 시스템의 자료구조(data structure)를 IR 시스템의 문서 형태에 사상(mapping)시키는 것이다. 사용자 메뉴는 IR 시스템에서 하나의 문서로서 저장된다. 각 사용자 메뉴를 적절히 식별하고 정의된 구성 요소의 형태를 열거(enumerative)하기 위하여, 각 문서의 첫번째 소절은 특정 메뉴와 관련된 구성 요소 분류로 구성되어야 한다. 구성 요소의 형태와 사용자 메뉴는 구성 요소 분류 방법에 의해 접근된다.

라이브러리 항목(entries)은 기본 단어와 유사어(synonym)의 목록(list)도 찾을 수 있어야 하며, 완전한 소프트웨어 구성 요소는 IR 시스템에서 하나의 텍스트 문서로서 저장된다. 검색 시스템의 일반 사용자에 의해 수행되는 네 가지 기본적인 기능은 다음 항목이 지원 되어야 한다.

◎ 새로운 구성 요소의 추가
◎ 현 구성 요소의 수정
◎ 현 구성 요소의 삭제
◎ 현 구성 요소의 검색

새로운 구성 요소가 검색 시스템에 입력될 때, 사용자는 소프트웨어 구성 요소와 연관된 정보를 명세(specifies) 하여야 하며, 라이브러리에 대한 관리도 이루어 져야 한다. 소프트웨어 구성 요소 위한 탐색 능력은 검색 시스템의 핵심이며, 검색된 소프트웨어의 카탈로그는 저장된 정보가 엑세스되지 못하면 소용이 없다. 사용자는 적절한 사용자 메뉴와 질의어를 통해 모든 정보를 처리할 수 있어야 하며, 라이브러리 관리도 지원되어야 한다.

Ⅲ. 라이브러리와 사용자 인터페이스

버전 제어에 필요한 소프트웨어 구성 요소를 잘 식별하기 위해서는 이들을 저장·관리하고, 유지하며, 필요한 구성 요소를 찾아내는 검색 메카니즘의 성능이 좋아야 한다. 버전 제어를 위한 구성 요소의 검색시에는 해당 구성 요소에 관한 정보 뿐만아니라 다른 구성 요소와의 관계를 표현하여 검색 능력을 향상시켜야 한다. 그러므로 구성 요소를 관리하기 위한 시스템의 주 기능으로는 구성 요소를 등록시킬 수 있는 라이브러리와 이 라이브러리에서 저장된 구성 요소를 검색하고, 질의어를 이용하여 새로운 구성 요소를 작성하기 위한 도구가 필요하다. 그러므로 화면에서 제시하여 주는 도움말과 질의어에 따라 원하는 처리를 할 수 있도록 사용자 인터페이스를 설계한다.

3.1 시스템 모델

본 논문에서 개발한 시스템 모델의 구성도는 그림 3-1과 같다.

3.2 시스템 모델의 기능

1) 검색 시스템

구성 요소 검색 시스템은 시스템 모델의 전반적인 기능 중 가장 중요한 기능으로써, 버전 관리를 위한 중요한 기능을 제공한다. 검색 시스템은 구성 요소의 라이브러리를 초기화 하여 주는 기능과 라이브러리에 구성 요소의 추가, 삭제 및 검색할 수 있는 기능을 지원하도록 한다. 이 기능에 의해서 처리되는 구성 요소는 실제의 원문(text)이 된다. 그러므로 데이타의 중복을 최소화하여 실제의 데이타를 저장한 데이타베이스의 성격보다도 질의어를 이용하여 원문속의 핵심 단어를 가지고 원하는 구성 요소를 검색하는 정보 검색 시스템의 성격을 가진다.

구성 요소의 검색 기능을 위하여 구성 요소 자체 내에 구성 요소에 대한 정보 또는 버전 제어 과정에 도움이 될 수 있는 주석을 포함한다. 즉, 구성 요소의 소재, 구성 요소의 복잡도, 버전 제어에 필요한 절차나 주의 사항 그리고 관련 문서 등의 정보를 구성 요소 내에 작성한다. 구성 요소의 검색시에는 이러한 주석 또는 구성 요소에 기술된 특정 명령어를 가지고 찾도록 한다. 이를 위한 질의어는 사용자가 구성 요소를 찾는데 필요한 단어(word)를 사용하여 작성하도록 설계하고, 검색 기능에서 각 단어를 가지고 이에 관련된 구성 요소를 찾도록 한다. 이러한 과정을 사용자가 입력란 질의어를 가지고 자동적으로 수행하도록 하여 시스템을 관리하고 확장하는 부담을 줄일 수 있도록 하였다.

구성 요소 검색 시스템은 라이브러리를 초기화 하기 위한 initialize, 구성 요소를 추가 기능을 지원하는 addModule, 구성 요소 검색 기능을 지원하는 searchModules, 구성 요소 삭제 기능을 지원하는 removeModule과 같은 서비스를 구현한다.

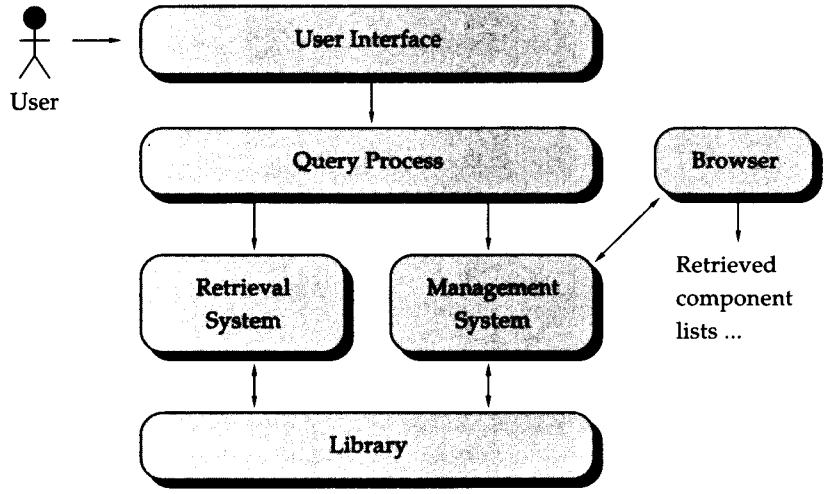


그림 3-1. 시스템 모델의 구성도

구성 요소를 위한 검색 시스템의 기능을 지원하기 위해 다음과 같은 3가지 구체화된 facet를 두어 구현한다.

(1) 대상 facet

라이브러리에서 사용자가 질의어로 탐색한 연관된 구성 요소들을 모아 놓은 facet이다.

(2) 언어 facet

사용하고자 하는 언어의 종류별로 facet를 구성한다.

(3) 특성 facet

구성 요소 중에서 특성이 비슷한 것들을 모아 놓은 것이다.

전체 facet에서 사용자가 질의어로 조건에 맞는 구성 요소를 대상 facet에 가져오면, 이 중에서 사용자가 언어와 특성을 선택하여 해당 facet에서 작업을 처리할 수 있도록 하였다. 언어 facet는 일반적으로 다음과 같은 체계로 제한하였으며, 해당 언어 facet에서 작업을 수행한다.

〈언어 facet〉
A. COBOL
B. C
C. C++
D. Lisp
E. Smalltalk

그림 3-2. 언어 facet의 체계

전체 facet에서 해당 facet로 제한하면서 필요로 하는 소프트웨어 구성 요소를 찾아서 작업하므로 작업의 흐름을 파악할 수 있고 버전 제어 측면에서 구성 요소를 선택하는 부담을 줄일 수 있다. 또한 질의어에 의한 검색과 해당 작업 항목을 마우스로 선택하여 사용자의 편의성을 도모한다.

2) 구성 요소 관리 시스템

이 분에서는 라이브러리에서 검색 시스템으로 탐색한 구성 요소들이 해당 facet에 존재할 때, 이들의 수정, 추가, 삭제, 결합, 저장, 읽기 작업을 처리할 수 기능과 이를 위해 브라우져를 포함한다.

3) 브라우저

일반적인 화면 편집기에서 제공하는 기능을 가진다. 화면 편집기의 기능은 라이브러리 관리 기능이나 새로운 구성 요소 관리에 이용된다.

화면 편집기는 Smalltalk 시스템이 가지고 있는 filbrwsr 클래스를 이용한다. 편집용 화면 크기 정의, 종료는 사용자 질의어를 이용하여 사용자가 직접 작성하도록 설계한다.

4) 사용자 인터페이스

사용자와 시스템 간의 상호 대화를 위한 경계이며, 이것은 자연스럽고 사용자에게 편리성을 제공해야 한다.

본 논문에서 사용자 인터페이스는 주 인터페이스와 서브 인터페이스로 분류한다. 검색 시스템에서 설명한 라이브러리를 관리하고 사용하기 위한 단어의 인덱스와 질의어를 생성할 수 있도록 하며, 이 것은 사용자 인터페이스에서 서브 인터페아스를 사용하여 생성·처리한다. 기본 인터페이스는 검색 시스템과 구성 요소 관리 시스템을 질의어나 해당 항목을 마우스로 선택하여 처리할 수 있으며, 4장의 구현 예에서 살펴보기로 한다.

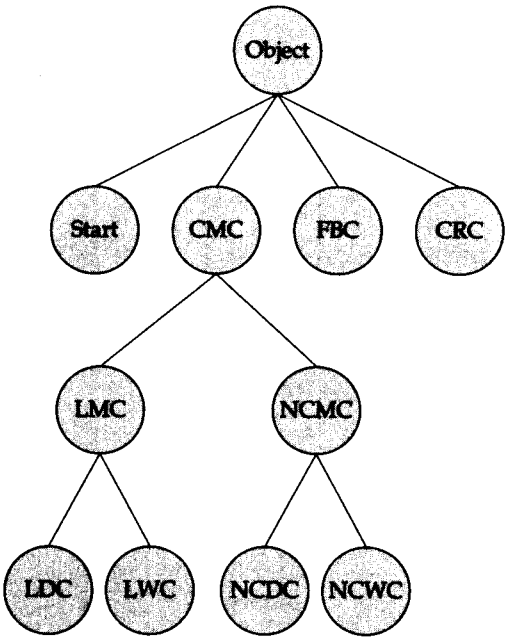
Ⅳ. 라이브러리와 사용자 인터페이스의 구현과 분석

4.1 클래스와 메소드의 구성

본 논문에서는 객체 지향 프로그래밍 언어인 Smalltalk를 사용하여, 버전 제어를 지원하는 라이브러리와 이의 사용자 인터페이스를 구현하였다. 이 것은 버전 제어를 위한 라이브러리를 구성하고, 구성 요소의 등록과 검색을 지원하기 위한 사용자 환경을 구축하며, 버전 제어를 위한 기반 시스템으로 사용된다. Smalltalk는 구성 요소를 발견하고, 이 구성 요소를 변형하여 수정하기 위한 환경을 가지고 있다[1,4,6,7,8,11].

라이브러리는 Smalltalk에서 제공하는 강력한 사용자 환경(user environment)과 메시지 전송, 상속성, 동적 연결 등의 기본 특성을 이용하여 구성 요소를 검색하고 카탈로깅한다. 이미 개발되어 저장된 구성 요소가 구성 요소화가 잘 되어있고, 빌딩 블록 개념에 충실한 특성을 가지고 있으면, Smalltalk와의 명확한 인터페이스를 통하여 사용될 수 있다. 또한, 자료 추상화를 이용한 자료구조에서의 알고리즘과 view/controller 등을 통한 응용(application)에서의 프레임워크(framework)를 사용할 수 있다.

라이브러리에서는 새로운 구성 요소의 추가, 삭제, 수정 작업을 할 수 있으며, Smalltalk에서는 객체가 클래스의 인스탄스이고, 클래스는 인스탄스와 특정한 종류의 객체를 사용하고 구축하는데 필요한 모든 정보를 제공한다. 클래스 내의 메소드(methods)은 프로시듀어로서, 클래스에 메시지를 전송하면 메소드를 호출하여 처리하고, 발생될 처리 방식의 동적(dynamic) 결정에 도움을 주는 상속성을 이용하여 구성 요소를 구성한다. 클래스 상의 메소드는 그 실행 중에 사용할 임시 변수를 할당한다.



- CMC : Component Management Class
- LMC : Library Management Class
- CRC : Component Retrieval Class
- NCMC : New Component Management Class
- LDC : Library Definition Class
- LWC : Library Working Class
- NCDC : New Component Definition Class
- NCWC : New Component Working Class
- FBC : File Browser Class

그림 4-1. 시스템의 클래스 구성

1) 라이브러리

라이브러리 클래스는 버전 제어를 위한 소프트웨어를 등록할 수 있는 라이브러리를 생성하여 초기화하여 주고, 이 라이브러리에 구성 요소를 추가, 검색, 삭제할 수 있는 메소드를 가진다.

라이브러리의 메소드는 구성 요소 관리 프로그램에서의 사용자 화면과 질의어 처리를 통한 구성 요소의 추가, 삭제, 검색, 재명명, 수정 등의 작업에 사용된다. 3장에서 설명한 라이브러리 시스템의 각 기능을 Smalltalk에서의 메소드로 구현하였다.

라이브러리 클래스는 원문을 검색할 수 있는 정보 검색 시스템을 생성하고, 이 원문에서 포함한 단어를 가지고 원문의 이름인 구성 요소명을 찾아 낼 수 있다.

또한, 라이브러리 클래스를 사용하기 위한 단어의 인덱스와 질의어를 생성할 수 있도록 하며, 사용자 인터페이스에서 생성·처리한다. 인덱스와 질의어를 생성하기 위한 메시지로 initialize를 사용하여 원문과 구성 요소명을 저장할 set과 dictionary를 초기화 시킨다.

구현한 라이브러리는 Smalltalk 상의 Set 클래스를 이용하여 원문을 저장하고, Dictionary 클래스를 사용하여 구성 요소명을 기록한다. 즉, 본 논문에서의 라이브러리를 위해서는 Set과 Dictionary 클래스를 사용한다. Set 클래스는 중복을 제거하여 객체를 저장하고, Dictionary 클래스는 키 값에 의해서 객체를 저장하고 검색하는 기능을 가진다.

2) 검색 시스템

구성 요소의 검색에서는 asSet 메시지를 사용하여 검색하고자 하는 구성 요소명을 얻는다. 구성 요소의 추가는 add:, addWord: 메시지를 사용하고, 추가시에는 각 단어의 대소문자에 대한 검색시의 오류 방지를 위해 asLowerCase to: 메시지를 사용한다. 검색에서는 occurrencesOf: 메시지를 사용하며, 삭제에서는 remove: 메시지를 이용하여 처리한다.

3) 관리 시스템

구성 요소 관리 프로그램은 Object 클래스의 하위 클래스로서 구성 요소 관리 프로그램 처리 클래스를 가진다.

이 클래스의 하위 클래스로서 라이브러리 관리 클래스와 새로운 구성 요소 작성 클래스가 구성되며, 라이브러리 관리 클래스는 라이브러리 정의 클래스와 라이브러리 작업 클래스를 가진다. 새로운 구성 요소 작성 클래스의 하위 클래스로는 새로운 구성 요소 정의 클래스와 새로운 구성 요소 작업 클래스가 있다.

전체 시스템의 시작은 Object 클래스와 상위 클래스의 속성을 상속 받은 라이브러리 작업 클래스와 새로운 구성 요소 작업 클래스로 실행한다.

4) 질의어 처리

각 처리 방식에서는 Smalltalk 윈도우(window) 중의 하나인 Prompter와 menu, Display를 사용하여 사용자가 필요로 하는 질의사항을 처리하며, 사용자에 의해 작성된 질의어의 오류 발생시에는 오류 메시지를 출력한다.

4.2 알고리즘 및 실행

검색 시스템의 초기화와 구성 요소 추가 알고리즘은 그림 4-2, 그림 4-3과 같다.

```
initialize
    Set class for text storage
    Dictionary class for module name
```

그림 4-2. 구성 요소 초기화

```
addModule: pathName
    |word wordStream|
    move pathName to wordStream
    write documents using add: message
    check wordStream is nil
    whileFalse:
    convert lowercase documents
    close
```

그림 4-3. 구성 요소 추가

초기 실행 화면은 구성 요소 관리 프로그램시작 클래스를 실행함으로써 화면상에서 Smalltalk 시스템에 실행 메시지를 전송하여 처리된다.

그림 4-4의 각 메뉴에서는 화면에 출력되는 질의어 메시지와 마우스를 사용하여 사용자는 라이브러리에서 소프트웨어 구성 요소를 관리할 수 있다.

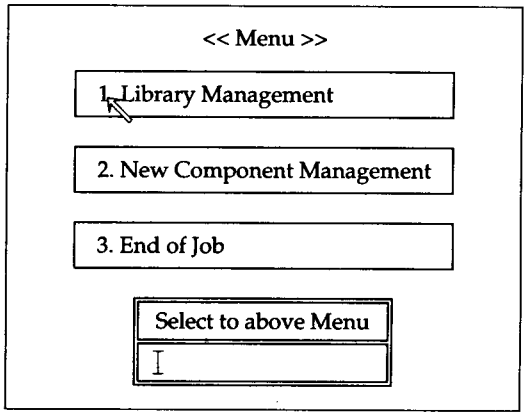


그림 4-4. 라이브러리 관리 시스템의 초기 화면

4.3 성능 분석

본 논문에서 구현한 소프트웨어 라이브러리에는 C로 작성된 50 개의 구성 요소를 사용하여 8개 프로그램으로 구성된 샘플 프로그램을 개발하였다. 개발된 각 프로그램의 라인 수는 메뉴 프로그램 121 라인, 입력 프로그램 80 라인, 조회 프로그램 150 라인, 삭제 프로그램 140 라인, 수정 프로그램의 260 라인과 두개의 출력 프로그램이 116 라인과 185 라인으로 작성한 후 이들을 재사용율을 분석하여 라이브러리의 효율성을 입증하였다. 사용된 구성 요소의 종류는 표 1과 같다.

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

본 논문에서 사용된 구성 요소의 재사용율은 다음과 같이 계산한다.

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

재사용된 구성 요소 중의 일부는 수정없이 사용된 것도 있으나, 일부는 수정되어서 사용된다. 재사용된 구성 요소내에서 수정없이 사용된 순수 재사용율은 다음과 같이 계산한다.

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

재사용된 구성 요소내에서 수정되어 사용된 부분 수정율은 다음과 같다.

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

본 논문에서 사용한 모델을 실행하여 분석한 결과 재사용율은 54 %, 재사용된 구성 요소에서 수정없이 사용된 순수 재사용율은 37%, 수정된 후 된 부분 수정율은 71%로 나타났다.

표 1. 구성 요소의 종류

종 류	메 뉴	입 력	조 회	삭 제	수 정	출 력	검 색	백 업	기 타	합 계
구성 요소	6	6	4	6	6	6	4	2	6	50
라인 수 평균	132	88	140	125	154	102	149	59	31	

Ⅴ. 결 론

본 논문은 Smalltalk 언어를 이용하여 라이브러리와 이의 사용자 인터페이스를 제공하는 시스템을 구현하여 소프트웨어 구성 요소를 저장 및 검색할 수 있는 검색 시스템과 질의어를 구현하였으며, 이 시스템은 버전 관리를 위한 라이브러리로 사용될 수 있다.

버전 제어의 효율은 라이브러리에 기억된 소프트웨어 구성 요소의 질과 양, 새롭게 개발하려는 프로그램과 라이브러리에 기억된 구성 요소의 관련도에 따라 효율이 달라지며, 보다 다양하고 질 좋은 구성 요소가 많이 있고, 이들과 새로 개발하려는 프로그램과의 관련도가 높을수록 효율은 향상된다.

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 효율적으로 구성 요소를 정량적으로 측정하여 사용 빈도수가 높은 구성 요소를 파악 및 관리하기 위한 방법 등이 연구되어야 한다. 또한, 이들의 시각 프로그래밍과 연관된 아이콘화 및 사용자 인터페이스에 대한 추가 연구가 필요하며, 개발된 라이브러리를 기반으로 한 버전 제어 시스템을 구축해야 한다.

참 고 문 헌

[1] Adele Goldberg and David Robson, "Smalltalk-80 : The Language and Its Implementation", Addison-Wesley, 1988

[2] Bernhard Westfechtel, "Revision Control in an Integrated Software Development Environment", ACM, pp.96-105, 1989

[3] 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

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

[5] Ian Thomas, "Version and Configuration Management on a Software Engineeing Database", ACM, pp.23-25, 1989

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

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

[8] J. Diederich and J. Milton, "Experimental Prototyping in Small-talk, IEEE Software, pp.50-64, May 1987

[9] MARC J. ROCHKIND, "The Source Code Control System", IEEE trans. on software eng., vol. 1, no. 4, pp.364-370, DECEMBER 1975

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

[11] Smalltalk/V, digitalk inc., 1986

[12] Scott A. Kramer "History Management System", ACM, pp.140-143. 1991

[13] 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

[14] Tichy, W. F., "RCS - A System for Version Control", Software Practice & Experience, Vol. 15, No. 7, pp.637-654, 1985

[15] Vincenzo Ambriola, Lars Bendix, Paolo Ciancarini, "The evolution of configuration management and version control", Software Engineering Journal, pp.303-310, November 1990

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

□ 著者紹介

오 상 엽(吳 相 燁)

광운대학교 대학원 전자계산학과(이학석사)
광운대학교 대학원 전자계산학과 박사과정 수료
경원전문대학 교양과(전산교육담당) 조교수

※ 관심 분야 : 소프트웨어 재사용, 컴파일러

최 우 승

1977년 동국대학교 전자공학과 졸업
1981년 동국대학교 대학원 전자공학과 졸업(공학석사)
1994년 동국대학교 대학원 전자공학과 졸업(공학박사)
1978년 ~ 1980년 동국대학교 전자계산원 교수
1981년 ~ 현재 경원전문대학 전자계산과,사무자동화과 교수
1994년 ~ 현재 한국 OA 학회 부회장

김 홍 진

광운대학교 대학원 전자계산학과(이학석사)
광운대학교 대학원 전자계산학과(이학박사)
경원전문대학 전자계산과 부교수
경원전문대학 전자계산소장

※ 관심 분야 : 소프트웨어 재사용, 컴파일러