원시코드의 메타 정보 관리를 위한 버전 제어 시스템의 설계와 구현

오 상 엽[†] · 장 덕 철[††]

요 약

빠른 컴퓨팅 환경과 응용 구조의 변화, 그리고 다양한 사용자 요구는 소프트웨어 개발에 대한 수요를 증대시키고 있다. 버전제어는 기존의 소프트웨어를 구축하는데 사용된 델타를 이용하여 소프트웨어의 생산성을 향상시킨다.

본 논문에서는 검색 시스템과 델타 관리 프로그램으로 구성된 객체지향 버전제어 시스템을 설계 및 구현하였다. 검색 시스템에서는 다양한 검색 방법을 제안한다. 이 방법은 파일이름, 내용, 크기, 그리고 작업일자를 가지고 처리하는 방법론을 제공한다. 다양한 검색 방법은 효율적인 델타 관리를 위해 중요하다. 이러한 방법을 이용하여 델타 관리를 위한 메타 데이터를 쉽게 구성할 수 있다.

구현된 버전 제어 시스템은 다른 시스템과 비교하여 다음과 같은 장점을 가진다. 첫째, 델타 관리를 위해 전향적 및 후향적 방법을 통합하여 버전의 유지보수를 처리한다. 둘째, 프로젝트내에서 델타 관리 부분은 전향적과 후향적 방법을 통합하여 관리의 효율성을 증대한다. 또한, 제안된 시스템은 프로젝트 저장소를 위해 파일과 데이터베이스를 사용하는 방법을 지원하여, 효율적인 버전 관리가 되도록 하였다.

Design and Implementation of Version Control System for Meta Information Management of Source Codes

Sang-Yeob Oh[†] · Duk-Chul Chang[††]

ABSTRACT

Rapid computing environment, change of the application structure, and various user demand will increase the demand of the software development. Version control is helpful to improve productivity using delta, and useful to establish component from existing data of source code.

This paper presents the design and implementation of the version control system, which is composed of retrieval system and delta management system. In retrieval system, various retrieve methods are proposed. This methods provides the process methodology with filename, content, size and date. Various retrieve methods are important for the effective delta management. Meta data can be easily composed for the delta management by these methods.

Compared with other systems, this implemented version control system has some advantage. First, for delta management, version maintenance for delta management becomes easier by integrating the forward and back-

† 정 회 원: 경원전문대학 교양과
†† 정 회 원: 광운대학교 전자계산학과
논문접수: 1997년 6월 11일, 심사완료: 1997년 10월 27일

ward methods. Second, delta managent part of a project is to unite the forward and backward method. The effieiency of this system is to increased in management. Also, this system supports a technique of using the database and files for project repository and makes the version management more effective.

1. 서 론

다양하고 복잡한 소프트웨어 시스템의 개발을 위해 많은 기술적 진보가 지난 20년동안 소프트웨어 분야에서 있어 왔으며, 빠른 컴퓨팅 환경과 응용 구조의 변화는 부분 또는 전체 시스템의 개발 확산을 가속화시키고 있다. 이를 위한 여러 가지 소프트웨어 개발 및 관리 기법에 대한 연구가 진행되고 있다. 근래에는 소프트웨어 개발 과정의 자동화, 새로운 소프트웨어 개발 방법론인 객체 지향 기법을 사용하는 방법 및 소프트웨어 유지 보수의 효율을 높이는 방법 등이 주로 이용되고 있다[1, 19]. 소프트웨어 개발 및 유지 보수에 도움을 주기 위한 방법으로 형상 관리(configuration management), 재사용(reusability), 버전 제어(version control), CASE 등이 있다.

이러한 새로운 방법론 및 자동화 도구들이 계속해서 개발되고 있는데, 그 중 하나가 버전 제어 시스템이다. 버전 제어 시스템은 오류 수정, 사용자 요구 사항의 변경, 소프트웨어 기능 향상 등의 이유로 시간이 지남에 따라 변화하는 소프트웨어 구성요소 및 소프트웨어 형상에 대한 메타 데이터(meta data)를 체계적으로 관리하기 위한 시스템을 말한다[2, 5, 8, 13, 14].

버전 제어를 위한 기반 시스템으로는 데이타베이스에 데이타를 어떻게 구성하여 저장, 관리할 것인가의 문제가 제시되며, 이를 위해 효율적인 검색 시스템이 제공되어져야 한다[7, 9, 10, 22, 23]. 또한 각 버전의 메타 데이터를 관리하기 위한 향상된 델타 관리가 지원되어야 한다.

기존의 델타 관리 방법은 전향적(forward)방법과 후향적(backward) 방법으로 나누어지며, 대부분의 버전 제어 시스템은 후향적 방법을 사용하고 있다[16]. 대표적인 버전 제어 시스템들 중에 CCC는 전향적 델타 관리 방법을, PVCS, SourceIntegrity, SourceSafe는 후향적인 델타 관리 방법만을 지원한다. 그러나 두가지 방법의 장점을 사용하면, 버전 제어에서 델타 관리의 효율성을 증대시킬 수 있다.

버전 제어는 시스템 진화와 개발에 따른 델타를 관리하며, 소프트웨어의 크기와 비용이 증가함에 따라 버전 제어의 중요성은 증대되고 있다. 버전 제어에서의 중요한 문제는 버전들의 일관성(consistency)을 유지하는 문제이며, 한 버전의 변경은 이와 관련된 다른 버전에 영향을 주게 된다.

소프트웨어 개발 프로젝트 동안에 구성요소의 버전 제어는 복잡한 작업이며, 이것은 구성요소가 계속적으로 변하기 때문이다. 또한 구성요소들은 서로 연관되어 있으며, 여러 사용자에 의해 공유되기 때문에 적절한 구성요소의 버전 제어가 균일한(uniform)한 방법으로 수행되어야 한다. 즉, 데이타 종속성과 데이타 중복성의 문제를 해결하고, 이를 위해 필요한 다양한 메타 데이타 관리를 필요로 한다.

본 논문에서의 구성요소는 특정 기능을 수행하는 단위로서 개발되며, 논리적으로 서로 독립된 기능을 수행하거나 분리되어 컴파일되었다가 후에 연결되어 사용되는 프로그램의 일부인 모듈을 구성요소로 한다. 기존의 방법중에는 순수한 객체지향 개념만을 도입하여 클래스를 구성요소로 처리한 논문도 있으나[21], 전체적인 응용 영역을 고려할 때 클래스를 포함하여 일반 프로그램을 구성하는 함수나 모듈도 구성요소로 처리할 수 있어야 한다.

본 논문에서는 Visual C++를 사용하여 버전 제어를 위한 검색 시스템과 델타 지원 프로그램을 설계 및 구현하였다. 검색 시스템과 델타 지원 프로그램에서는 효율적인 검색과 버전의 관리를 위해 전향적 및 후향적 델타 관리의 장점을 통합한 방법을 아이콘 자원에 의해 지원하여 버전 제어의 효율성을 증대하며 유지보수를 쉽게할 수 있는 장점를 가진다. 또한, 작업 영역을 위해 각 프로젝트는 데이터베이스 지원을 받고, 이를 파일로 처리하여 프로젝트의 관리를 용이하게 하였다. 이것은 각각 3장의 검색시스템과 델타 관리 부분에서 다루었다.

본 논문의 제 2장에서는 버전제어, 검색 시스템에 대한 관련 연구를, 제 3장에서는 버전 제어 시스템의 설계와 방법론, 제 4장에서는 구현 방법과 실행 예를 다루고, 본 논문에서 제시한 제어 관리 시스템을 다

른 시스템과 비교 및 분석을 하였으며, 5장에서는 결론을 기술하였다.

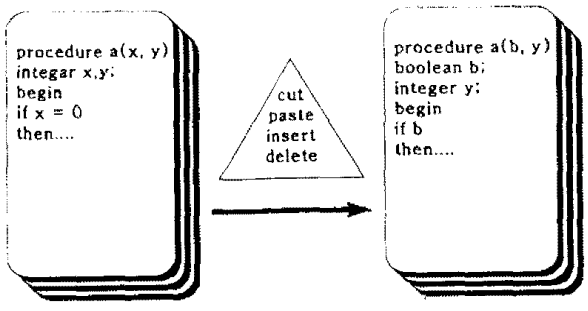
2. 관련 연구

2.1 버전 제어

시스템을 구성하는 기본 단위를 구성요소(component)라고 하며, 객체지향프로그램에서는 이것을 소프트웨어 객체가 된다. 소프트웨어 구성요소는 두가지 분류 방법이 제시되는데, 우선 생성 방법에 따라 분류하면, 원시 구성요소(source componet)와 파생된 구성요소(derived component)가 되고, 내부 구조에 따라 분류하면 더 이상 나눌 수 없는 원자 구성요소(atomic component)와 집합 구성요소(aggregate component)가 된다. 집합 구성요소는 하나의 시스템, 부시스템, 복합 모듈 등을 나타낸다.

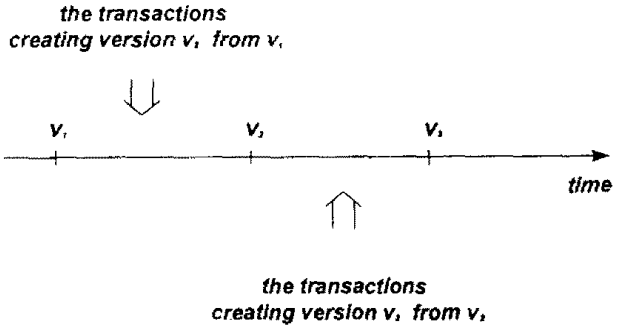
프로젝트 개발과 유지보수되는 동안 구성요소는 변경되며, 이러한 변경(changes)의 집합 즉, 대개 브라우저에 의해서 수행되는 텍스트상에 있는 오퍼레이션인 cut, paste, insert, delete 등의 작업으로 구성요소의 한 버전들을(연속적인) 다음 버전으로 변환시키는 이력 과정(history step)이 버전 제어이다[2, 11, 14].

버전은 변화하는 객체의 어느 순간의 상태를 말하며, 버전 제어는 이력 관리를 메타 데이터를 사용하여 프로그래밍의 생산성을 향상시키기 위한 방법이다. 기존의 버전 제어 방법은 각 버전을 모두 보관하는 전체 복사(full copy) 방법과 특정한 버전을 가지고 있으며, 나머지 버전들은 그 버전과의 차이로 관리하는 델타(delta) 방법으로 나눌 수 있다. 전체 복사 방법은 단순하지만 공간의 효율성 문제와 필요한 버전의 식별이 어렵다는 문제점을 가지고 있다. 델타 방법은 전체를 보관하는 비간의 선택에 따라, 최초의 버전에서 다른 버전들을 델타를 사용해서 구해내는 전향적(forward) 방법과 이와 반대로 최종 버전을 기준으로 하는 후향적(backward)인 방법으로 나누어지는데 최종 버전의 사용이 가장 빈번하므로 후향적인 델타 방법이 더 우수한 것으로 평가되고 있다[4]. (그림 1)은 한 버전을 새로운 버전으로 변환하는 이력 과정을 나타낸다.



(그림 1) 이력 과정
(Fig. 1) History step

변환하는 오퍼레이션들은 많은 시스템에서 이력 과정을 위한 델타(delta)의 형태를 제안하기 위해서 삼각형 박스 내에 들어있다. (그림 2)는 버전 제어 과정을 나타내며, 버전제어 과정중의 각 이력을 델타 형태로서 관리한다.

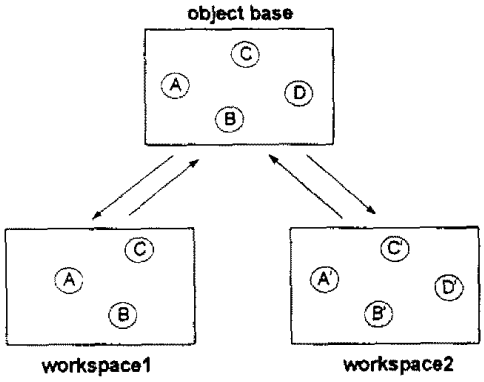


(그림 2) 버전 제어 과정
(Fig. 2) Version control process

버전 제어와 소프트형상 관리는 지난 20년 동안 많은 발전을 하였다. 버전 제어 시스템의 종류는 크게 3가지로 나누어 볼 수 있다[15, 16, 17]. 첫째는 파일 시스템을 기반으로 각 파일의 변경을 저장하고 관리하며, 각 버전을 위해 메타 데이타를 관리한다. 이러한 시스템으로는 SCCS, RCS, PVCS 등이 있다. 둘째는 데이타베이스에 모든 프로젝트와 파일에 관한 메타 데이타를 저장하는 방법으로서 모든 프로젝트에 관한 정보를 파일 시스템을 기반으로한 구조보다 효율적으로 관리할 수 있다. 이러한 시스템으로는 CCC, CMVC, PCMS 등이 있다. 셋째는 다른 사용자나 도구에 의해 버전 제어 시스템이 사용될 때, 파일 투명성(file transparency)를 제공하는 방법으로서 ClearCase 시스템에서는 캐시 디렉토리를 이용하여 해결하고 있다. 이러한 종류들은 각 시스템의 사용 여건에 따

라 각기 다른 특성들을 가지고 있다.

특정 버전에 대해 변경이 이루어지면, 사용자는 작업 공간(workspace)을 생성하고 요구되는 보존을 작업 공간에 복사한 다음 변경 관리를 수행한다. 작업 영역은 여러개의 버전이 생겼을 때 각 버전을 관리하기 위한 것으로 본 논문에서는 파일 단위로 처리된다. 그러므로 모든 변경은 각 작업 공간에서 이루어진다. 본 논문에서는 작업 공간의 처리를 위해 각 프로젝트마다 별도의 파일로 처리하여 해결하였으며, 이것은 검색시스템의 지원을 받는 데이터베이스와 델타 관리 프로그램의 지원에 의해 가능하다. 본 논문에서의 작업 공간에 대한 처리는 3장의 델타 관리 부분에서 다루기로 한다.



(그림 3) 분리된 작업 공간
(Fig. 3) Separated workspace

버전 제어는 프로젝트를 구성하여 프로젝트 컴포넌트와 변경 사항에 대한 관리, 그리고 시스템 개발을 지원해야 한다. 버전 제어 시스템은 다음과 같은 기능을 제공해야 한다[15].

- 파일에 대한 변경 관리 방법
- 프로젝트와 이들의 발전에 따른 제어
- 버전들의 관리
- 작업 영역 지원
- 사용자 인터페이스

버전 제어를 위해서는 기본적으로 표준화된 소프트웨어 구성요소의 작성과 이들을 데이타베이스에 체계적으로 저장하여 검색이 용이하도록 하여야 하며[5], 이것은 버전 제어를 위한 기반이 된다. 또한, 필요한 소프트웨어 구성요소를 명세하여 검색하고 다른 소프트웨어 구성요소들과 결합하여 새로운 소프트웨어를 작성하여 소프트웨어의 델타를 효율적으로 관리해야 한다[6, 12].

본 논문은 이 중에서 버전 제어를 위한 검색 시스템과 이의 지원을 받아 버전을 관리하기 위한 버전 제어 시스템을 설계 구현하였으며, 델타의 관리는 전향적 및 후향적 방법을 시각화된 아이콘에 의해 특정 프로젝트내에서 사용하여 버전 제어의 효율성을 증대하였다. 본 논문에서 제안한 시스템에서 사용하는 구성요소는 모두 원시 코드이면서 프로그램의 한 단위 기능을 가진 모듈이다.

2.2 검색 시스템

검색 시스템은 데이타베이스에서 소프트웨어의 구성요소를 카탈로깅(cataloging) 하고 검색(retrieving) 하기 위해 사용된다.

검색 시스템은 버전 제어에 필요한 구성요소를 식별하여 검색하며, 라이브러리 또는 데이타베이스에 구성요소를 등록시키고 새로운 구성요소를 작성할 수 있어야 한다[21, 22, 23].

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

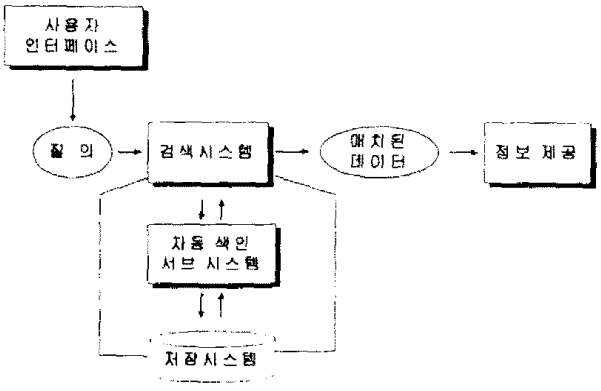
새로운 구성요소가 검색 시스템에 입력될 때, 사용자는 소프트웨어 구성요소와 연관된 정보를 명세(specifies)하여야 하며, 데이타베이스에 대한 관리도 이루어져야 한다. 소프트웨어 구성요소를 위한 검색 능력은 검색 시스템의 핵심이며, 검색된 소프트웨어의 카탈로그는 저장된 정보에 엑세스되지 못하면 소용이 없다. 사용자는 적절한 사용자 메뉴와 질의어를 통해 모든 정보를 처리할 수 있어야 하며, 데이타베이스 관리도 지원되어야 한다.

정보 검색 시스템의 사용은 원하는 정보에 대한 접근을 용이하게 함으로써 정보 수집에 대한 시간과 노력을 단축시키게 된다. 특히 관리할 정보의 양이 기하급수적으로 증가하고 있는 정보화 시대인 오늘날

에는 효율적인 정보 검색 시스템에 대한 요구는 증가하고 있다.

정보 검색 시스템은 몇몇의 기본적인 연산을 제공해야 한다[18]. 데이타베이스에 데이타들을 삽입하고, 데이타들을 변경하며, 필요에 따라 삭제할 수 있는 수단을 제공해야 한다. 또한, 원하는 문서를 찾을 수 있는 방법과 이 문서들을 사용자들에게 표시해 줄 수 있는 수단을 제공해야 한다. 이러한 정보 검색 시스템은 몇가지 요소들로 나뉘어질 수 있다. 사용자의 요구를 받고 결과를 제시하는 사용자 인터페이스 부분, 사용자의 요구를 해석하여 적절한 검색 방법들을 결정하며, 필요한 동작을 수행하는 검색 엔진 부분, 대용량의 데이타들을 저장 관리하는 저장 시스템 등으로 구분할 수 있다.

이에 따라 최근 정보 검색의 성능 및 유용성을 향상시키기 위해 정보 검색 시스템의 각 구성 부분에 대한 연구가 계속적으로 활발히 진행되고 있다. 그래픽 사용자 인터페이스와 같은 편리하고 인식도가 높은 수단을 개발하고, 멀티미디어를 이용한 검색 결과 제시 등을 고려하고 있으며, 보다 정확도가 높은 정보 검색을 위해 데이타 들의 저장 방법 및 검색 모델을 제안하고, 정보 검색 성능 향상을 위한 검색 방법 및 저장 시스템 구조, 인덱싱 방법 등을 제시하는 연구들이 계속되고 있다. 정보 검색 관련 연구는 정보 검색 시스템의 각 구성 요소 단위로 정보 검색 효율의 향상을 위한 방안들을 제시하고 있다.



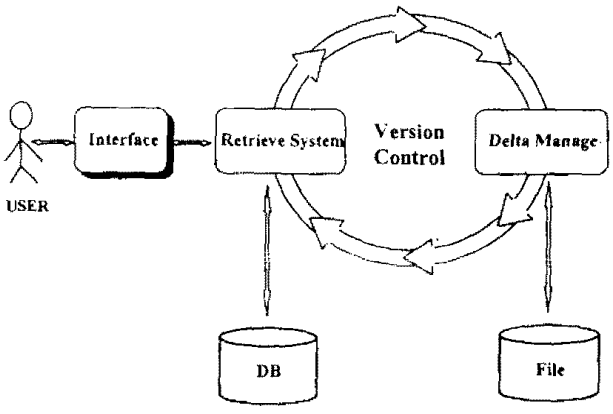
(그림 4) 일반적인 검색 시스템 구성
(Fig. 4) General Retrieval system

3. 버전 제어 시스템의 설계

3.1 시스템 모델

버전 제어를 위한 구성요소의 검색시에는 해당 구성요소에 관한 정보 뿐만아니라 해당 구성요소의 검색에 있어서 다양한 검색 방법을 지원하여 검색 능력을 향상시켜야 한다[12]. 그러므로 구성요소를 관리하기 위한 주기능으로 구성요소를 등록시킬 수 있는 데이타베이스와 이 데이타베이스에서 저장된 구성요소를 검색하고, 질의어를 이용하여 새로운 구성요소를 파악하기 위한 검색 시스템이 필요하며, 이 것은 델타 관리를 위해서도 중요하다. 또한, 각 작업영역을 처리하기 위해 본 논문에서는 하나의 파일에 각 작업영역에서 처리한 메타 정보들을 모아 처리하여 프로젝트의 관리를 처리하였다. 이를 위해 화면에서 제시하여 주는 도움말과 질의어에 따라 원하는 처리를 할 수 있도록 사용자 인터페이스를 설계한다.

본 논문에서 개발한 시스템 모델은 효율적인 버전 제어를 지원하기 위한 검색 시스템과 이와 연관된 델타 관리 도구로 구성한다. 이외에도 데이타 베이스, 관리 시스템, 브라우저, 사용자 질의어 및 인터페이스로 구성된다. 구성도는 다음 (그림 5)와 같으며, 각 구성은 다음 절에서 설명한다.



(그림 5) 시스템 모델의 구성
(Fig. 5) Structure of system model

3.2 시스템 모델의 기능

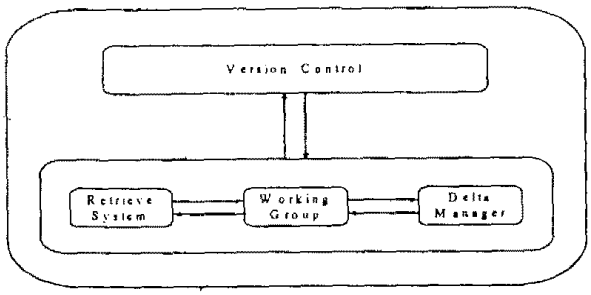
(1) 검색 시스템

검색 시스템은 시스템 모델의 전반적인 기능 중 가

[illegible] 중요한 [illegible] 능으로써, 버전 제어를 위한 중요한 기능을 제공한다.

구성요소의 검색 기능을 위하여 구성요소 자체 내에 구성요소에 대한 정보 또는 버전 제어 과정에 도움이 될 수 있는 주석을 포함 한다. 이것은 내용에 대한 검색을 지원한다. 즉, 구성요소의 소스, 구성요소의 복잡도, 버전 제어에 필요한 절차나 주의사항 그리고 관련 문서 등의 정보를 구성요소 내에 프로그램 작성자가 필요한 내용을 작성한다. 구성요소의 검색 시에는 이러한 주석 또는 구성요소에 기술된 특정 명령어를 가지고 찾도록 한다. 이를 위한 질의어는 사용자가 구성요소를 찾는데 필요한 단어를 사용하여 작성하도록 설계하고, 검색 기능에서 각 단어를 가지고 이에 관련된 구성요소를 찾도록 한다. 이러한 과정은 사용자가 입력한 질의어를 통해 자동적으로 수행되도록 하여 시스템을 관리하고 확장하는 부담을 줄일 수 있게 하였다.

본 논문의 검색 방법은 또한 파일 이름에 의한 검색을 지원하며, 이는 윈도내에서 와일드 카드 문자도 지원하도록 설계하였다. 지정된 파일의 크기와 같은 파일을 검색하는 기능도 추가하였으며, 다이얼로그 박스에서 제시된 크기보다 적거나 큰 파일도 검색하는 기능을 가진다. 이것은 특정 파일의 크기를 기준으로 사용자에게 파일에 상태에 관한 정보를 제공하여 사용자가 원하는 파일을 찾을 때 도움을 줄 수 있다. 다른 검색 방법은 년, 월, 일을 기준으로 사용자가 작업한 파일을 검색하도록 하였다.

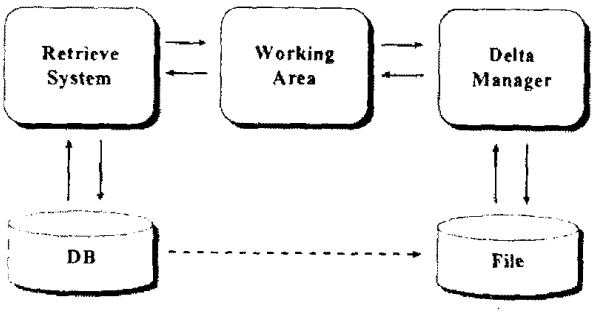


(그림 6) 버전 제어, 델타 관리, 그리고 검색 시스템과의 연관관계

(Fig. 6) Version control, Delta management, and Relation of the retrieval system

본 논문에서는 구성요소의 내용, 구성요소의 파일 이름, 구성요소의 크기, 작업일자 등의 다양한 검색 방법을 제안하므로 버전 제어 측면에서 구성요소를 선택하는 부담을 줄일 수 있으며, 버전 제어를 지원하는 시스템으로 사용될 수 있다. 본 검색 시스템을 이용하여 버전 제어 작업을 효율적으로 수행할 수 있으며, 이것은 검색 시스템이 [illegible] 제어와 델타 관리를 지원함으로써 가능하다.

검색 시스템을 사용하여 각 버전별 표시, 수정, 보고서, 각 버전 제어, 최근 작업 내용, 버전 부여 등의 버전 제어 작업을 수행할 수 있으며, 버전 제어 작업은 델타 관리 작업을 포함한다. 델타 관리 항목으로는 변동 사항에 대한 파일 관리 기능을 수행하며, 파일 이름, 작업자, 작업일자, 버전, 기술, 조건 사항, 하드웨어, 소프트웨어 등을 관리한다.



(그림 7) 검색 시스템의 사용

(Fig. 7) Using of the Retrieve system

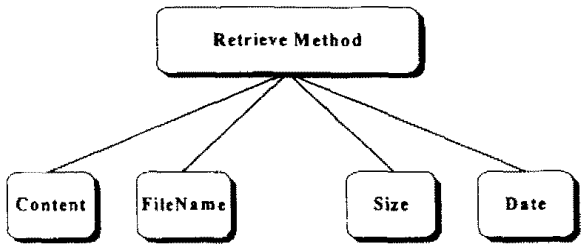
(그림 7)에서와 같이 버전 제어자와 델타 관리자의 작업은 검색 시스템을 기반으로 수행된다. (그림 7)에서 데이타베이스는 데이터 엑세스(Data Access)를 이용하여 데이타베이스의 프레임을 구성하고, 데이터베이스 처리를 위해 Visual C++상에서 ODBC를 DLL로 연결하여 처리하였다.

검색 시스템에 의해 사용자가 기본적인 파일을 찾아서 프로젝트에 필요한 내용을 데이타베이스에 구성하고, 이들 정보를 델타관리에서 응용하여 해당 프로젝트에 필요한 파일들에 대한 이력사항을 파일 형태로 관리하게된다.

버전 제어 프로그램을 수행하면, 초기 윈도상의 메뉴로서 〈File〉, 〈View〉, 그리고 〈Help〉 메뉴를 가지며, 〈File〉의 하위 메뉴로서 〈New〉, 〈Open〉, 〈Print Setup〉, 그리고 〈Exit〉가 있다. 〈New〉를 사용하여 델

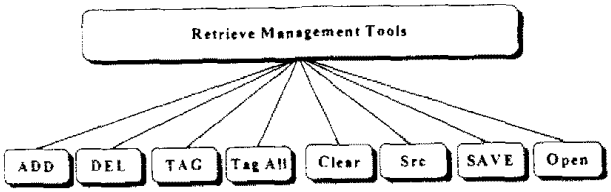
타 관리를 위한 윈도가 화면상에 나타나도록 설계하였다. 검색 시스템의 사용은 델타관리 윈도의 〈File〉 메뉴를 선택한 다음 〈Retrieve System〉을 수행한다.

다음은 검색된 파일들을 데이타베이스에서 저장하고 관리하기 위한 도구들로서, 본 논문에서는 버튼(button)으로 화면 설계를 하였다.



(그림 8) 검색 메소드
(Fig. 8) Retrieve method

(그림 9)의 검색 관리 도구들은 화면상에서 모두 버튼으로 처리된다. 버튼 ADD는 Tag나 Tag All로 선택한 파일들을 데이타베이스에 저장하기 위해서 사용하며, 이는 버전 제어에서 사용된다. DEL은 목록 리스트상에서 선택된 파일들을 삭제하기 위해 사용되도록 설계하였다. Clear는 선택된 태그들을 모두 제거하며, Src는 파일의 내용을 화면상에서 뷰(view)하여준다. SAVE는 데이타베이스에 선택된 파일을 데이타베이스에 저장한다.



(그림 9) 검색 관리 도구들
(Fig. 9) Retrieval management tools

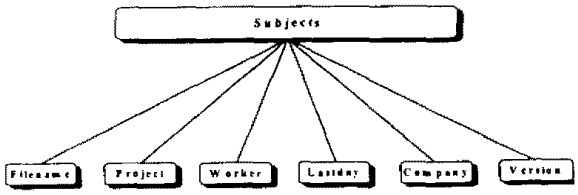
사용자 인터페이스에 있어서는, 질의어에 의한 검색과 해당 작업 항목을 마우스로 선택하여 사용자의 편의성을 도모한다. 이를 위해 각 검색 형태를 지원하는 대화상자에서 마우스로 처리하며, 각 검색은 해당 다이얼로그 박스에서 제어된다.

(2) 정보 추출

사용자가 각 검색 방법을 사용하여 찾아진 파일을 리스트하며, 이것은 사용자 인터페이스인 윈도와 메뉴바, 그리고 다이얼로그를 이용해서 사용자에게 정보를 추출하여 검색 정보를 제공한다. 검색 정보를 사용자에게 제공하기 위한 윈도 클래스를 생성하며, 본 논문에서 제시한 방법을 처리하기 위한 메소드와 다이얼로그를 가지고 처리된다.

(3) 델타 관리

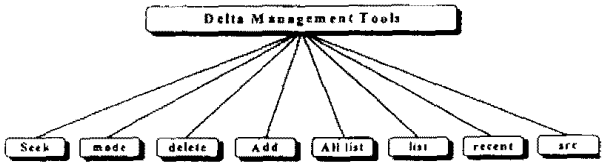
델타 관리는 검색 시스템의 데이타베이스에서 처리된 기본 정보들을 가지고 각 파일들에 대해서 델타 관리에 필요한 정보들을 처리하며, 프로젝트 관리를 위해 각 프로젝트별로 필요한 메타 데이타를 파일로 관리하여 작업 영역 문제를 처리하였다. 한 프로젝트를 구성하는데 여러 구성요소들을 가지고 한 작업 영역에서 처리할 수 있으며, 한 개의 구성요소만을 가지고 작업영역을 처리할 수 있다. 작업영역에서 사용되는 구성요소들을 파일로 관리하여 누가, 어떻게, 무슨 이유로 작업하였는가를 파악할 수 있다. 이 데이타에는 각 파일의 파일이름, 작업자, 작업일자, 버전, 조건 사항, 문서화 기술사항 등이 포함된다. 델타 관리는 (그림 10)과 같이 파일 이름, 프로젝트, 작업자, 최종 작업일, 회사, 그리고 버전별로 관리를 할수 있도록 하여 효율적인 델타 관리가 되도록 하였다.



(그림 10) 델타관리를 위한 주제 항목
(Fig. 10) Subject items for Delta management

(그림 10)과 같이 다양한 주제를 사용하여 프로젝트내에서 델타관리에 사용되며, 이들은 다음 (그림 11)에 의해서 버전제어 작업을 처리한다.

(그림 11)의 델타 관리 도구도 (그림 9)와 같이 윈도상에서 버튼으로 처리되도록 설계하였다. 전향적 및 후향적 방법으로 프로젝트내의 버전들을 관리하기 위해서 툴바(ToolBar) 자원을 가지고 아이콘을 설계하였다.



(그림 11) 델타 관리 도구
(Fig. 11) Delta management tools

(4) 사용자 인터페이스

사용자와 시스템 간의 상호 대화를 위한 경계이며, 이것은 자연스럽고 사용자에게 편리성을 제공해야 한다[20].

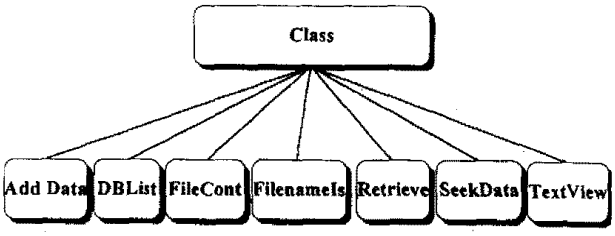
시스템에서 설명한 기본 인터페이스는 검색 시스템과 델타 관리 시스템을 질의어나 해당 항목을 마우스로 선택하여 처리할 수 있다. 이를 위한 사용자 인터페이스는 윈도즈(windows)를 이용하여 처리하며, 윈도는 관련된 다이얼로그와 메뉴바, 버튼, 리스트, 스크롤바 등의 컨트롤 자원을 사용하여 강력한 환경을 제공한다. 예를 들어 아이콘을 추가하는 경우, 아이콘의 크기나 모양, 색깔과 같은 정보들을 자원 파일로 관리한다. 모든 자원들은 텍스트 스트링이나 식별자를 사용하여 처리된다.

4. 버전 제어 시스템의 구현과 분석

4.1 클래스와 메소드의 구성

본 논문에서는 객체 지향 프로그래밍 언어인 Visual C++를 사용하여[3], 버전 제어를 위한 검색 시스템과 델타 관리 프로그램을 구현하고, 이를 위한 사용자 인터페이스를 제공한다. 이것은 구성요소의 등록과 검색을 지원하기 위한 사용자 환경을 구축하며, 버전 제어를 위한 기반 시스템으로 사용된다. 델타 관리 프로그램에서는 효율적인 버전의 이력 관리를 위해 전향적 및 후향적 델타 관리의 장점을 통합한 방법을 아이콘 자원에 의해 지원하여 버전 제어의 효율성을 갖도록 하였다. 시스템에서는 Visual C++에서 제공하는 강력한 사용자 환경(user environment)과 메시지 전송, 상속성, 동적 연결 등의 기본 특성을 이용하여 구성요소를 검색하고 카탈로깅한다. (그림 12)는 본 시스템에서 설계, 구현한 기본적인 클래스 구성도이다.

각 클래스는 응용 프로그램의 인스턴스, 윈도의 커서, 윈도의 아이콘, 윈도의 배경색, 윈도의 양식, 윈도의 메뉴 이름과 같은 정보들에 대한 핸들을 유지한다.



(그림 12) 시스템 클래스 구성
(Fig. 12) Structure of system class

(1) 검색 시스템

검색 시스템은 사용자로 하여금 관심있는 영역만을 탐색할 수 있고, 구성요소(component)에 대한 정보를 제공하여야 한다. 버전 제어를 위한 구성요소를 등록하고 저장하는 것은 버전 제어에서 기본이 되는 기능이며, 구성요소들의 집합이 크고, 그 구성요소들이 여러 응용 분야에 걸쳐 널리 사용되면서 우수한 모듈성을 갖는 환경에서는 필요한 구성요소를 검색하고 식별하는 것이 중요한 문제로 제기된다[11, 21, 24].

검색 클래스는 원문의 내용, 이름, 크기, 최종 작업일을 가지고 검색할 수 있는 정보 검색 시스템을 생성하고, 검색 클래스를 사용하기 위한 윈도와 메뉴바를 생성할 수 있도록 지원한다. 이를 위한 메뉴 설계 구축과 (그림 12)의 Retrieve 클래스를 이용하여 검색과 출력을 지원한다. (그림 8)의 검색 메소드와 버튼으로 구성한 (그림 9)의 검색 관리 도구를 사용하여 버전제어에 필요한 파일들을 데이타베이스에 저장하고, 프로젝트명, 기술사항(description), 문서(document) 등에 대한 정보도 데이타베이스에 저장되는데, 이것은 DBList 클래스와 AddData 클래스의 지원으로 처리된다. 이것은 델타 관리 프로그램에 의해 각 이력을 관리하고, 프로젝트 파일로 처리된다.

검색 시스템은 Visual C++ 상에서 구현한 Retrieve 클래스를 이용하여 처리하며, 내용에 의한 검색은 SeekData 클래스를 이용하며, 이 클래스내의 Seek Content( )를 이용하여 처리된다.

```
//파일안에 특정 문자로 찾을경우
case 0:
```

```
    {
    sprintf(file, "%s\\%s", list[2], list[0]);
    if(!SeekContent(file))
      doseek = FALSE;
    else
      doseek = TRUE;
      break;
    }
```

다음은 파일 크기에 의해 파일을 검색하기 위한 프로그램의 일부이다.

```
//파일크기로 찾을경우
case 2:
    {
    long fsize;
    sscanf(m_strSeekName. operator const char*(),
    "%ld", &fsize);
    switch(m_nCompSize)
    {
    case 1:
        if(c_file.size > fsize)
          doseek = TRUE;
        else
          doseek = FALSE;
        break;
    case 2:
        if(c_file.size> = fsize)
          doseek = TRUE;
        else
          doseek = FALSE;
        break;
    ......
    case 5:
        if(fsize == c_file.size)
          doseek = TRUE;
        else
          doseek = FALSE;
        break;
    default:
        doseek = FALSE;
        break;
    }
  }
```

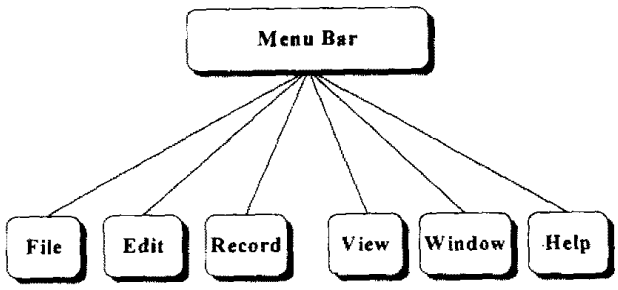
검색 시스템을 사용하여 데이터를 찾은 경우에는 원래의 파일에서 델타를 반영하여 관리해야 하며, 이를 위한 메타 데이터를 위한 클래스와 이의 처리를 위한 함수 선언 부분은 다음과 같다.

```
class AddDataDlg : public CDialog
{
// Construction
public:
    AddDataDlg(CWnd * pParent = NULL); // standard
    constructor
    AddDataDlg(HWND wnd);
// Dialog Data
      //{{AFX_DATA(AddDataDlg)
      enum {IDD = IDD_ADDDATA};
      CString m_filename;
      double m_version;
      CString m_lastdate;
      long m_size;
      CString m_worker;
      CString m_project;
      CString m_document;
      CString m_discription;
      CString m_condition;
      CString m_product;
      CString m_company;
      CString m_software;
      CString m_hardware;
      CString m_internet;
      CFileContView* pView;
      int m_nCurrentNum;
      int m_nTotalNum;
      HWND m_pWnd;
      //}}AFX_DATA

      void AddTotal(int num);
      void SetData(CString filename,CString date,
```

```
        CString size);
        void AddDb();

// Overrides
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(AddDataDlg)
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);
        // DDX/DDV support
        //}}AFX_VIRTUAL

// Implementation
protected:

        // Generated message map functions
        //{{AFX_MSG(AddDataDlg)
        virtual BOOL OnInitDialog();
        afx_msg void OnNext();
        afx_msg void OnPrev();
        afx_msg void OnAdd();
        afx_msg void OnAddall();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};
```

(2) 델타 관리

델타관리는 버전제어 작업의 주된 작업으로서 시스템에서 사용자는 대부분의 작업을 델타 관리에서 윈도와 메뉴바를 마우스와 단축키를 사용해서 처리하며, 메뉴바는 다음과 같은 메뉴로 구성한다.



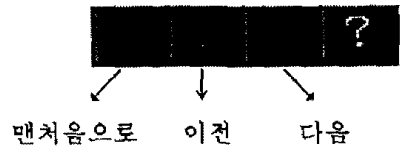
(그림 13) 델타관리를 위한 메뉴바의 구성

(Fig. 13) Structure of menu bar for Delta management

델타 관리를 위해서는 DBList, FileCont, TextView 등의 클래스 지원에 의해서 처리된다. FileCont 클래스는 검색 시스템의 지원으로 데이타베이스에 저장된 정보를 이용하여 버전을 부여하며, 사용자가 필요로 하는 최신의 정보를 입력된 메타 데이터를 이용하여 찾고, 이들을 작업영역에서 프로젝트 단위로 관리하기 위한 기능을 〈File〉 메뉴의 〈TextSave〉〈TextOpen〉 서브메뉴의 지원에 의해 파일 단위로 작업 여역을 관리한다. 메타 데이터는 3장에서 설명한 바와 같이 데이터베이스로 구축되어 사용되며, 이것은 델타 관리의 파일관리와 연계되어 정보를 공유한다. 델타 관리를 위해 3장에서 설계한 (그림 11)의 델타 관리 도구에 의해 지원되며, FileCont 클래스에 의해 생성되는 윈도에 의해 지원된다. TextView 클래스는 사용자가 델타 관리를 위해 작업한 파일들은 프로젝트 단위로 저장되며, 이것을 확인할 수 있도록 하는 대화상자로 사용자에게 제시되도록 하는 클래스이다. 다음은 TextView 클래스내에 작성한 프로그램의 일부분으로서, 화면상에 프로젝트로 구성한 파일들의 델타 관리 사항을 보여준다.

```
BOOL CTextViewDlg::OnInitDialog()
{
  CDialog::OnInitDialog();
  FILE *out=fopen(m_strFileName, "r");
  if(out==NULL)
    return TRUE;
  char buff[300];
  for(;;)
  {
  if(fgets(buff, 300, out)==NULL)
    break;
  buff[strlen(buff)-1]=NULL;
  m_strBuff=m_strBuff+buff;
  m_strBuff=m_strBuff+"\r\n";
  }
  fclose(out);
  UpdateData(FALSE);
  return TRUE;
  // return TRUE unless you set the focus to a control
}
```

다음 아이콘은 전향적 및 후향적 방법으로 프로젝트내에서 델타를 관리하기 위해 사용된다.



(그림 14) 델타 관리를 위한 아이콘
(Fig. 14) Icon for the Delta Management

본 논문의 전향적 및 후향적 방법은 한 프로젝트내에서만 전향적, 후향적 방법을 사용할 수 있다. 즉, 해당 프로젝트내에서만 전향적과 후향적 방법이 적용된다. 본 논문의 (그림 3)과 같이 여러 프로젝트가 생성될 수 있으며, 이 안에서 전향적 및 후향적 방법이 적용된다. 해당 프로젝트를 여는 것은 (그림 16)의 〈Text Open〉 메뉴를 가지고 수행하며, 해당 프로젝트내에서는 전후로 (그림 14)의 아이콘을 사용하여 이동할 수 있고, 내용도 새로이 수정할 수 있다. 이 때 (그림 10)과 (그림 11)의 도구를 가지고 버전 제어 작업을 수행한다. 프로젝트를 파일 단위로 만들고 추가하는 것은 〈TextSave〉에서 지원되며, 새로운 변경 정보를 가진 델타를 반영하여 새 버전을 작성할 수 있도록 지원한다.

(그림 14)의 델타 관리를 위해서는 (그림 11)의 델타관리도구에서 "All list" 버튼을 선택한 다음 사용해야 한다.

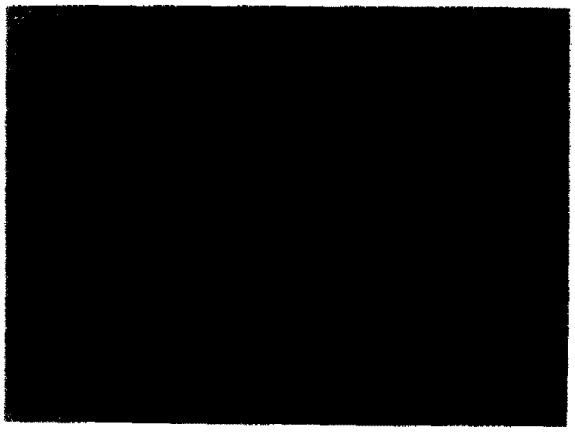
(3) 사용자 인터페이스

사용자 인터페이스를 위한 자원으로는 윈도, 버튼, 그리고 다이얼로그 박스를 사용한다. 윈도내에는 메뉴, 버튼, 그리고 컨트롤 등의 자원을 포함하며, 다이얼로그는 검색 처리 과정과 도움말에 대한 정보를 제공한다. 이를 위해 각 검색 방법에 대한 다이얼로그, 아이콘, 커서, 비트맵, 메뉴바, 버튼에 대한 정보를 각각 별도의 파일로 관리한다.

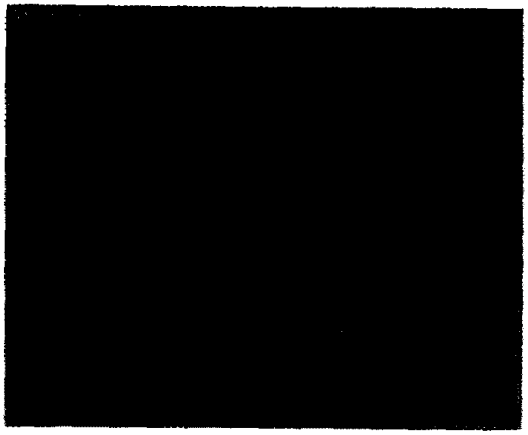
4.2 실행 및 결과 분석

(1) 실 행

논문상에서 제안한 시스템을 실행한 초기화면은 (그림 15)와 같으며, (그림 16)은 델타 관리를 위한 윈도에서 검색 메뉴를 선택한 예이다.



(그림 15) 초기화면
(Fig. 15) Primary screen

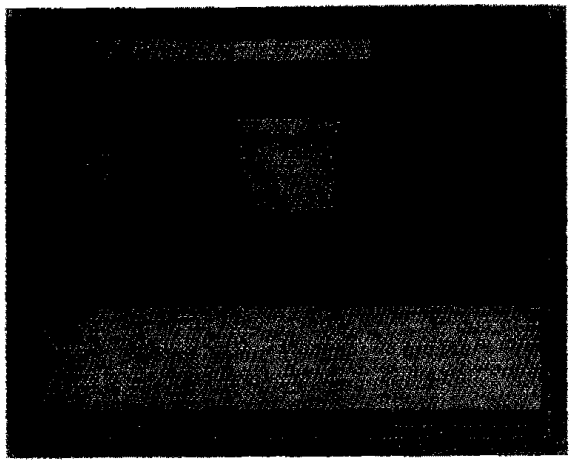


(그림 16) 검색 시스템의 선택
(Fig. 16) Select of Retrieve system

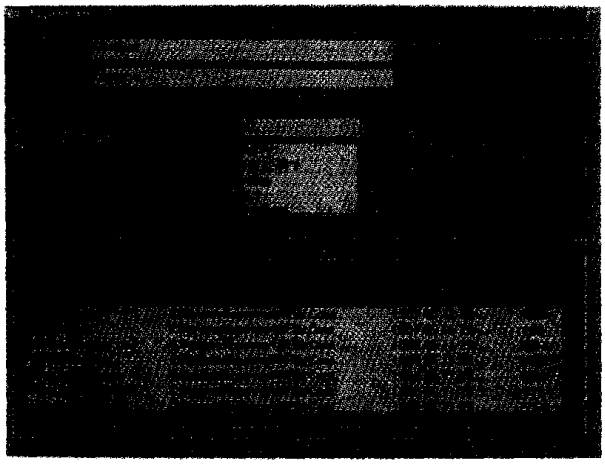
다음 (그림 17)은 검색을 위한 대화상자이며, 하단 영역에 검색된 데이터를 데이터베이스를 사용하여 관리하기 위한 버튼들을 보여주고 있다.

본 논문에서는 원시 코드와 메타 데이터를 모두 관리하고 있다. (그림 17)의 검색 화면에서 좌측 상단의 4가지 주제(Content, FileName, Size, Date)를 가지고 검색 작업을 하며, 이의 결과는 (그림 17)의 하단 부분의 대화 상자에 제시된다.

(그림 18)은 (그림 17)의 좌측 상단에서 내용에 의한 검색결과를 보여주는 화면이며, 다음의 (그림 19)는 하단의 [TAG] 버튼을 사용하여 사용자가 원하는



(그림 17) 검색 화면 1
(Fig. 17) Retrieve Screen 1



(그림 18) 검색 화면 2
(Fig. 18) Retrieve Screen 2

(그림 19) 검색 화면 3
(Fig. 19) Retrieve Screen 3

(그림 20) 원시파일 조회
(Fig. 20) Display of source file

원시 파일을 선택하는 것을 보여준다. [TAG] 버튼을 선택한 다음 [Src] 버튼으로 원시 파일의 내용을 (그림 20)과 같이 파악할 수 있다.

(그림 20)은 검색 시스템이 찾기전에 이미 하드디스크상에 저장된것이므로 프로그램상에서는 다시 저장할 필요는 없으며, 사용자에 의해 관리해 주면 된다.

델타 관리를 위한 메타 데이터에 대한 정보의 분석과 입력은 (그림 21)에서 이루어지며, 이 화면은 (그림 19)의 화면에서 [ADD] 버튼을 클릭한 경우이다.

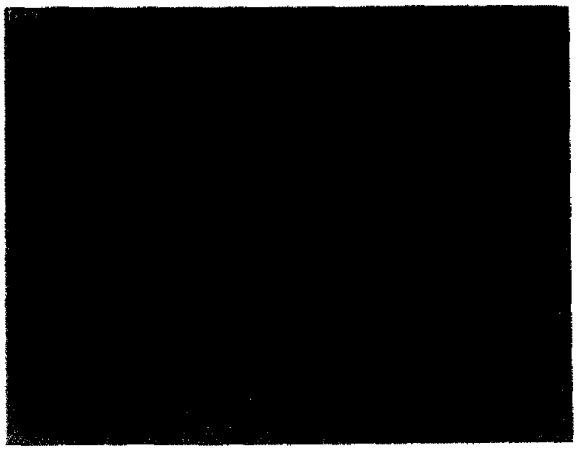
(그림 21)에 입력하는 정보는 사용자에 의해 DB에 저장되며, 이것은 (그림 23)과 (그림 25)에서 사용된다. (그림 22)에서 입력된 데이터는 새 버전 의 델타관리를 위한 메타 데이터를 Document, Discription, Condition등을 사용하여 파일에서 변경된 델타 관리를 위한 정보를 사용자가 만들어 낸다. 이것은 (그림 23)의 델타 관리 화면에서 연계되어 사용되며, 버전 관리중에 변경사항이 발생되면 (그림 16)에서의 〈Text Save〉 메뉴를 가지고 별개의 프로젝트로 구성하여 저장 및 관리한다. 이것은 버전들간의 변경사항인 이력을 관리하기 위한 방법이 되며, 원래의 파일에서 델타를 반영하여 새로 변경된 내용은 새로운 프로젝트로 구성할 수 있다.

(그림 23)에서 내용은 델타 관리 화면이며, 사용자가 필요한 메타 데이터는 검색 시스템의 지원에 의해 입력된 내용을 델타 관리에서 사용되도록 연계된다. (그림 23)에서 관리하는 구성요소를 우측 상단부분에 있는 버튼 "src"로 구성요소의 내용을 (그림 24)와 같

(그림 21) 메타 데이터의 입력 화면
(Fig. 21) Input Screen of Meta data

(그림 22) 입력 예
(Fig. 22) Example of input

(그림 23) 델타 관리 화면
(Fig. 23) Delta management screen



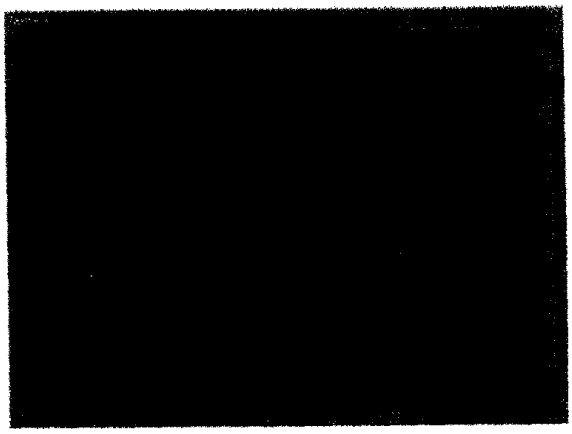
(그림 24) 구성요소 리스트
(Fig. 24) Component list

이 확인할 수 있으며, 이는 버전 제어 작업시 각 구성요소의 내용을 사용자가 파악하기 위해 필요하다.

(그림 25)의 좌측 상단에서 각 버전에 관한 변경 이력을 관리하기 위한 메뉴가 있으며, 이를 사용하여 원하는 버전에 관한 정보를 얻을 수 있다. 또한, 〈File〉 메뉴의 〈TextSave〉과 〈TextOpen〉의 서브 메뉴를 사용하여 사용자가 관리한 버전에 대한 내용을 확인할 수 있다. 또한, 3장의 델타 관리 부분에서 설명한 바와 같이 사용자가 관리한 여러 구성요소들은 하나의 파일 단위로 저장되어 관리될 수 있으며, 한 작업영역에서 사용된 구성요소들을 누가, 어떻게, 무슨 이유로 고쳤는가를 파악하도록 하여 버전 제어에 도움을 준다. 즉, (그림 25)의 화면에서 주제(subject)별로 관리한 내용을 가지고 1개의 독립된 프로젝트를 구성할 수 있으며(한 프로젝트내에 여러 파일 포함), (그림 24)에서 제시한 바와 같이 DB와 연계된 원시 파일 정보도 파악할 수 있다. 검색시스템과 버전 제어 부분에서 파악하는 이러한 원시 코드의 내용은 사용자에게 새로운 버전을 생성하는데 도움을 줄 수 있다. 한 프로젝트내에서 전향적 및 후향적 방법을 사용한다. 즉, 전향적 및 후향적 방법의 사용은 현재 열린 프로젝트내에서 사용되고, 다른 프로젝트에는 적용되지 않도록 하였다. 이것은 버전 제어에서 필요한 파일들이 프로젝트 단위로 관리되기 때문이며, 프로젝트는 파일 개념으로 관리된다.

(그림 26)은 작업 영역에서 사용자가 한 프로젝트

(그림 25) 버전 관리 위한 항목
(Fig. 25) Item for version control



(그림 26) 작업영역 리스트
(Fig. 26) Workspace list

를 구성하는데 필요한 구성요소들의 변경 이력에 관한 정보를 보여주며, 새로운 델타를 반영하여 만들어 내는데 사용된다.

(2) 비교 분석

본 논문에서 제안한 버전 제어 시스템은 Visual C++ 버전 4. 0 이상을 가지고 구현하였으며, 버전 제어에 필요한 정보들을 여러 시스템과 비교하여 분석하고, 이를 모델링하여 버전 제어 시스템을 구축하였다. 즉, 다른 시스템과의 프로젝트 관리, 사용자 인터페이스, 델타 관리, 저장소, 확장성을 고려하여 비교 분석하였으며, 이는 [16, 17]에서 제시된 평가 기준을 참조하였다.

3장과 4장에서 설계·구현한 바와 같이 본 논문에서 제안한 버전 제어 시스템의 사용자 인터페이스는 메뉴바, 버튼, 스크롤 바, 다이얼로그 박스 등을 사용하여 사용자에게 시각적인 인터페이스를 제공한다. 검색 방법은 파일의 이름, 내용, 크기, 작업날자 등의 다양한 방법을 사용하여 버전 제어를 위한 검색 시스템을 구현하였다. 또한, 작업역역을 프로젝트 단위로 관리하고, 이 프로젝트내에서는 전향적 및 후향적 방법을 지원하여 버전 관리의 효율성을 높이도록 하였으며, 파일과 데이터베이스의 적절한 지원으로 한 개의 저장소를 사용하는 시스템보다 데이터의 사고로부터 데이터를 복구할 수 있는 장점을 가진다.

5. 결 론

버전 제어 시스템은 기존의 시스템을 개발하는데

〈표 1〉 다른 시스템과의 비교 결과
〈Table 1〉 Compare result of another system

기준 / 시스템	사용자 인터페이스	프로젝트	델타 관리	저장소	Release & Variants
CCC	CLI	다중	전향적	프로젝트 저장소	virtual copy
PVCS	GUI(CLI)	단일	후향적	파일	braching & labeling
SourceIntegrity	CLI	다중	후향적	파일	braching & labeling
SourceSafe	GUI	다중	후향적	프로젝트 저장소	file sharing
제안 방법	GUI	다중	전향적 및 후향적	파일, 데이터베이스	database and file sharing

사용된 다양한 형태의 정보와 지식을 다른 시스템 개발에 재적용하여 생산성을 향상할 수 있고, 개발 비용을 감소하며, 유지보수를 쉽게할 수 있고, 소프트웨어의 작성 지식을 공유할 수 있는 장점이 있다.

본 논문은 Visual C++ 언어를 사용하여 소프트웨어 구성요소를 검색할 수 있는 검색 시스템과 델타 프로그램의 지원을 받는 버전 제어 시스템을 구현하였다.

본 논문에서는 첫째, 객체지향 프로그래밍 언어인 Visual C++를 이용하여 사용자가 원하는 구성요소를 파일 이름, 크기, 내용, 작업일자를 가지고 검색할 수 있는 검색 시스템을 제안하였다. 둘째, 전향적과 후향적 방법의 장점을 가지는 델타 관리 방법을 제안하였다. 이것은 프로젝트내에서 사용된다. 셋째, 델타 관리를 아이콘화하여 시각적인 면과 사용의 용이성을 증대하였다. 넷째, 데이터베이스는 ODBC를 DLL을 사용하여 처리하여 작업 영역 문제를 해결하였다. 검색 시스템에서 사용된 데이터 베이스를 델타관리에 지원하도록 하였다. 저장소를 파일과 데이터베이스로 분리하여 관리함으로써 검색시스템과 델타관리 시스템에서 필요한 데이터를 효율적으로 관리할 수 있으며, 이것은 하나의 파일이나 데이터베이스로 관리되는 시스템에 비해 불의의 사고에 대해서 안전하며, 검색 시간이 우수한 장점을 가진다. 다섯째, 시스템의 재사용성을 제공한다. 사용자가 데이타베이스에 등록한 구성요소들을 사용자가 원할 때마다 사용함으로써 사용자의 작업 시간 단축과 일관성 있는 작업을 유도하여 품질 향상을 가져올 수 있다.

참 고 문 헌

[1] Arthur E. Anderson, William J. Heinze, "C++ Programming and Funda-mental Concepts", Prentice Hall, Inc., 1992.

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

[3] David J. Kruglinski, "Inside Visual C++", MicroSoft Press, 1996.

[4] E. J. Choi, "Macroscopic and Microscopic Change Management of Software Objects", Korea Advanced Institute of Science and Technology, 1996.

[5] Feiler. P, Downey. G, "Transaction-Oriented Configuration Mana-gement", Carneige-Mellon University, November, 1990.

[6] G. Talens and C. Oussalah, M. F. Colinas, "Versions of Simple and Composite Objects", Proceedings of the 19th VLDB Conference Dublin, Irenland, 1993.

[7] Geoffrey M. Clemm, Evolutionary Software Inc., MA, "Replacing Version-Control with Jop-Control", ACM SIGSOFT SOFTWARE ENGINEERING NOTES, Vol. 14, No 7, pp. 96-105, 1989.

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

[9] Patrick Shicheng Chen, Rolf Hennicker and Matthias Jarke, "On the Retrieval of Reusable Software Components", Proceeding Advances in Software Reuse, pp. 89-98, 1993.

[10] R. Helm, Y. S. Maarek, "Integrating Information Retrieval and Domain Specific Approaches for Browsing and Retrieval in Object-Oriented Class Libraries", Proceeding of OOPSLA'91, pp 47-61, 1991

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

[12] 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.

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

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

[15] "PC-Based Version Control", http://www. silicom

/~alobba/pc/vc. html [16] "Team Development Overview", http://www. silicom/~alobba/overview. html.

[17] "Team Development Product Reviews", http://www. silicom/~alobba/review. html.

[18] 고영대, 유재수, 김병기, "효율적인 정보 검색 시스템 구축을 위한 새로운 프로세스 구조", 한국정보처리학회 논문지 제4권 1호, pp. 76-86, 1997. 1.

[19] 김덕현, 박성주, "확장된 객체지향 데이타 모형을 이용한 소프트웨어 변경 관리 시스템", 한국정보과학회 논문지 제 22권 2호, pp. 249-260, 1995.

[20] 김상근, 홍찬기, 이경환, "소프트웨어 재사용 시스템을 지원하는 사용자 인터페이스 구축기의 설계 및 구현", 한국정보처리학회논문지 제 2권 3호, pp. 324-334, 1995.

[21] 김정아, 문충렬, 김승태, "재사용 시스템 개발을 위한 객체 지향 검색 프레임워크", 한국정보처리학회 논문지 제 2권 5호, pp. 711-720, 1995.

[22] 오상엽, 김홍진, 장덕철, "버전 제어를 위한 소프트웨어 구성요소의 검색 시스템", 한국정보처리학회 논문지 제3권 5호, 1996.

[23] 장명섭, 정인상, 권용래, "재사용을 위한 소프트웨어의 분류 및 검색 방법", 한국정보과학회 논문지 제19권 6호, pp. 602-613, 1992.

장 덕 철

1974년 고려대학교 경영학과 졸업(경영정보학 석사)

1982년 고려대학교 경영학과 졸업(경영정보학 박사)

1981년~1982년 버클리 대학교 객원 교수

1976년~현재 광운대학교 전자계산학과 교수

관심분야: 소프트웨어 공학, 경영정보론, 버전 제어, 소프트웨어 형상 관리

오 상 엽

1989년 경원대학교 전자계산학과 졸업(공학사)

1991년 광운대학교 전자계산학과 졸업(이학 석사)

1992년~현재 광운대학교 전자계산과 박사과정 수료

1993년~현재 경원전문대학 교수

관심분야: 소프트웨어공학, 버전 제어