韓國 컴퓨터情報學會 論文誌
第10卷 第4號, 2005. 9.

2005-10-4-1-13

객체지향 시스템을 이용한 버전제어에서 효율적인 형상 형성 제어 모델링에 관한 연구

오 상 엽*

A Study on the Efficient Configuration Thread Control Modeling in Version Control using Object Oriented System

Sang-Yeob Oh*

요 약

버전 제어 시스템은 급속한 환경의 변화나 개발 환경이 복잡한 프로그래밍 환경에서 사용되고 있으며, 형상 형성(configuration thread) 정보의 제공과 이의 처리 방법은 버전 제어에서 중요한 역할을 하고 있다. DSEE에서의 시스템 모형, ClearCase의 뷰, SourceSafe의 레이블, CCC/Harvest의 패키지 등의 형상 형성 도구들은 정형화된 형상 규칙을 사용자가 적용하여 필요한 버전에 대한 형상 정보를 얻고 있다. 그러나, 이들 방법에서는 정의된 형상 규칙 정보를 정확하게 알지 못하는 경우나 미리 정의되지 않은 정보에 대해서는 형상 형성 정보 제공 문제가 발생한다. 또한, 이들 정보는 세부적으로 연계된 정의되지 않은 관련 버전이나 메타 정보를 제공하지 못하는 단점을 가지고 있다. 본 논문에서는 이러한 문제를 해결하고 효율적인 형상 형성제어를 위한 시스템을 모델링하고 구현하였다. 이 시스템에서는 형상 형성 정보를 효율적으로 제공하기 위해 부울리언 검색 모델과 벡터 검색 모델을 결합한 혼합 검색 모델을 제안하였으며, 라이브러리는 확장 facet 방법을 응용하여 설계하였다.

Abstract

A version control system is used in a rapidly changed environment or a program which developed in a complicated environment. And configuration thread information supporting and it's processing method has an important part in version control. Configuration thread tool such as a system model of DSEE, a view of ClearCase, a label of SourceSafe, and the package of CCC/Harvest have applied to formalized configuration rule by user and obtained a desired configuration information of the version. But it is a problem of configuration thread in supporting information that we, in this method, can't know a exactly well-defined configuration rule information and a predefined information. And these information have a demerit that can't supported the close connection along with undefined version and a meta-information. In this paper, we have modeling a system for these probelms to solve and a efficiently configuration thread supported. We also proposed a mixed retrieval model included a boolean retrieval model and a vector retrieval model for support efficiently configuration thread information. We applied and designed the libraries using extended facet method.

▸ Keyword : Version control, Configuration thread, Boolean and Vector retrieve, facet, library

• 제1저자 : 오상엽
• 접수일 : 2005.07.03, 심사완료일 : 2005.08.30
* 경원전문대학 교양과 부교수

Ⅰ. 서 론

소프트웨어 개발자는 소프트웨어 개발 중에 한 구성요소에 대해 여러 개의 버전들을 발생하며, 이의 유지 보수시에 에러에 대한 수정, 작업 효율을 높이기 위한 성능의 향상, 하드웨어에 따른 기능의 변화 등 다양한 이유에 의해 많은 수의 버전들을 생성한다. 이러한 버전의 중복은 다음과 같은 문제점을 가진다.

첫째, 버전들 간의 관계에 대한 데이터 관리가 필요하다. 만일 한 버전의 기능에 이상이 있어 그 앞 버전을 찾고자 할 때에는 자신의 앞 버전을 가리키는 메커니즘이 존재하여야 한다.

둘째, 버전에 대한 설명의 부족이다. 새로운 버전이 언제, 그리고 어떻게 생성되었는가에 대한 설명이 있어야 한다. 이러한 설명은 프로젝트 관리에 중요한 문서가 되기 때문이다.

셋째, 사용자가 시스템을 구성할 때 적절한 버전을 선택하는 문제이며, 이를 형상 형성(configuration thread)이라고 한다. 기존의 버전 제어 시스템은 초기 버전이 구성되고 버전이 어느 정도 진행되면, 이를 기반으로 델타를 사용하여 사용자에게 필요한 버전의 생성 문제를 해결한다. 그러나 기존의 버전들을 가지고 새로운 버전을 생성해내기 위해서는 저장된 버전들에 대한 형상 형성 정보를 사용자에게 제공해야 하며, 이것은 버전의 생성에 있어 중요하다.

기존의 버전 제어 시스템들로는 전진법을 사용한 SCCS[1]와 CCC/Harvest[2, 3], 역진법을 사용한 RCS[4], Clear Case[5], SourceSafe[6], 역진법을 트리구조에 적용한 버전 제어 방법[7], 포인터 저장 알고리즘을 이용한 버전 제어 방법[8], 컴포넌트 연결 과정과 버전관리 명세서를 제공한 방법[9] 등이 있다. 버전 제어에서 형상 형성 지원을 위해서 SourceSafe는 레이블, DSEE는 시스템 모형, Clear Case는 뷰, CCC/Harvest는 package와 package group, Sablime은 MR(Modification Request) 등을 사용한다. 이들 형상 형성 도구들은 미리 정의한 형상 규칙이나 미리 정의된 정보에 의해서만 형상 형성 정보를 제공한다. 즉, 이들 방법에서는 정의된 형상규칙 정보를 정확하게 알지 못하는 경우나 미리 정의되지 않은 정보에 대해서는 형상 형성 정보 제공 문제가 발생한다. 또한, 이들 정보는 세부적으로 연계된 정의되지 않은 관련 버전이나 메타 정보를 제공하지 않는 단점을 가진다. 이 문제를 해결하기 위해서는 형상 형성 방법으로서 개발중에 현재 작업에서 필요로 하는 정의된 버전 뿐만 아니라 비정의된 버전들도 쉽게 접근할 수 있는 방법을 제공해야 한다. 그리고 역동적인 개발 환경의 특성을 반영시키기 위해서는 문맥상으로 버전을 선택할 수 있는 보다 유연한 버전 선택 방법이 요구되고 있다.

이러한 문제를 해결하고, 효율적으로 형상 형성을 지원하기 위해 형상 형성 정보를 제공하기 위한 시스템을 모델링하고 구현하여 비정의된 정보와 세부 정보를 제공함으로써 제안한 방법이 기존 형상 형성 방법들에 비해 효율적임을 입증하는데 그 목적이 있다.

Ⅱ. 관련 연구

2.1 버전 제어에서의 형상 형성

형상 형성의 초기 형태라고 할 수 있는 것으로는 파일 하나의 이력을 관리할 수 있는 버전 제어 시스템인 SCCS와 RCS가 있고, 형상 형성을 위한 보조도구로는 make 프로그램이 있다. 이 2가지 시스템을 결합하면 기본적인 형상 형성 기능을 수행할 수 있다. Make 프로그램은 본래 시스템 구성(build)에 사용되는 도구이지만, 의존 관계를 표현하는 규칙에 버전 선택 명령어를 사용하면 특정 버전이나 가장 최근의 버전을 자동으로 획득할 수 있는 방법을 제공한다. 그러나 make도구에서 파일들간의 관계 설정에 사용되는 makefile의 관계설정 방법과 버전 선택 방법은 복잡하고, 여러 개의 makefile을 유지하는 경우 이들 간의 정규적인 매개변수 전달 방법이 존재하지 않을 뿐만 아니라 단순한 버전 록(vesion lock)으로만 동시성 제어를 하고 있기 때문에 이 도구를 대규모 환경이나 분산 환경에 사용하는 것은 불가능하다는 단점을 갖는다.

이에 비해, Apollo를 기반으로 하는 DSEE는 시스템 모형(system models)과 BCT(Bound Configuration Threads)라는 높은 수준의 시스템 형성형성 방법과 버전 선택 방법을 제공한다. Clearcase는 DSEE의 기능을 확장하고 Apollo와 같은 특정 운영체제나 하드웨어에 종속되지 않고 여러 플랫폼, 특히 UNIX상에서 작동할 수 있도록 개선하였다. 그러나 시스템 모형과 BCT를 이용한 방법은 시스템 형성정보가 개발 생산성에만 초점이 맞추어져 있다는 점과 버전을 선택하는 방법이 경직되어 있다는 점에서 형상 형성에 대한 관리 기능 측면에서 단점을 가진다. 최근에 PC상에서 사용되는 버전 제어 도구로 MS사의 SourceSafe가 있다. 이 도구에서는 형상 형성과 파일 공유를 위해 레이블과 핀이라는 방법을 사용한다. CCC/Harvest는 앞에서 설명한 Clearcase나 SourceSafe와는 다른 방법으로 형상 형성을 지원한다. 다른 형상 형성 방법과는 달리 프로세스 지원을 중요하게 생각하는 철학 위에서 개발되었기 때문에 다른 시스템에서는 지원하지 않는 package와 package group 이라는 개념을 사용하고, 상태(state)를 옮겨 다니면서 작업을 하도록 지원한다. Sablime은 CCC/Harvest의 package와 유사한 개념인 MR(Modification Request)을 사용한다. MR에서는 버전 변경 요구 내용을 서술하며, 이를 이용하여 버전을 인출하고 저장한다. Sablime에서는 생명 주기 모형이 고정되어 있어서 사용자의 요구에 맞게 고칠 수 없는 문제를 가진다.

2.2 검색 시스템과 라이브러리

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 오늘날에는 효율적인 정보 검색 시스템에 대한 요구는 증가하고 있다. 정보 검색 시스템은 몇몇의 기본적인 연산을 제공해야 한다[10]. 데이터베이스에 데이터들을 삽입하고, 데이터들을 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.

검색 시스템은 표면적인 특성에 따라 매우 다양할 수 있지만, 일반적으로 3가지 모델, 즉 부울리언 검색 모델(boolean retrieval model), 벡터 검색 모델(vector retrieval model), 확률 검색 모델(probabilistic retrieval model)로 분류할 수 있다.

기존의 이러한 검색 방법들은 전통적으로 텍스트 검색 방법을 채택하고 있기 때문에 각 버전을 표현하는데 필요한 정보를 제공하기 어렵고, 사용자의 요구사항과 일치하는 버전의 검색에 실패했을 경우 다양한 검색 방법을 지원하여 검색 효율을 향상시켜야 한다. 또한, 검색한 버전은 적절한 수정을 거쳐서 개발 중인 시스템에 새로이 추가되거나 업버전되므로 버전에 대한 내용을 사용자가 이해하기 쉽게 제공해야한다.

라이브러리는 구성요소의 접근, 탐색, 제어, 보안에 용이한 포괄적인 DBMS이며, 사용자가 구성요소를 생성, 편집, 검증, 합성하기 위한 기능을 제공하며, 확장이 용이해야 한다. 소프트웨어 구성요소의 관련성을 부여하는 방법에 따라 라이브러리는 enumerative와 facet 방법[11], 어휘처리 수준을 포함하는 분류 방법[12], 그리고 구문 및 의미 분석을 포함하는 방법[13]이 있다.

III. 시스템의 설계

3.1 시스템 모델

소프트웨어 버전의 분류를 위한 라이브러리를 위해 제안한 혼합 검색 모델에서는 확장 facet 방법을 사용하였으며, 각 facet별로 가중치가 할당된다. 그리고 이 분류 방식을 이용하여 소프트웨어 버전들을 분류한 결과 버전들의 분류가 용이하고, 유사한 버전을 쉽게 식별할 수 있었다. 또한, facet 분류 방식을 사용하므로 검색을 수행한 다음 사용자로 하여금 관심 있는 영역만을 탐색할 수 있으며, 이는 사용자 인터페이스에 의해 메타 정보와 자동으로 연계되므로 소프트웨어에 대한 자세한 정보를 제공하며, 유사한 버전을 효과적으로 구분할 수 있는 분류체계를 지원하도록 하였다.

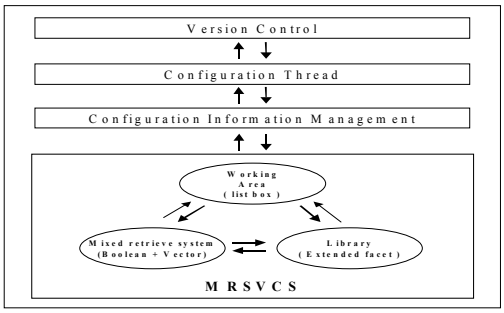

그림 1. 시스템 모델
Fig. 1. System model

3.2 검색 시스템

형상 형성 지원을 위한 모델링 시스템에서 부울리언 검색은 시스템 모델의 전반적인 기능 중 가장 중요한 기능으로써, 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 포함한다. 이 검색 항목은 내용에 대한 검색을 지원한다. 내용에 의한 검색에서는 확장 부울리언 질의를 지원한다. 그러므로 논리합, 논리곱, 부정을 포함하는 질의를 작성할 수 있으며, 이것은 형상 형성 도구를 사용하지 않고 세부적으로 연계된 정보를 찾는데 도움을 준다. 논리합을 위해 +, 논리곱을 위해 *, 부정을 위해 - 기호를 사용하여 확장 부울리언 질의를 작성하도록 설계하였다. 그리고 이들 확장 부울리언 질의를 위한 연산은 혼합해서 사용할 수 있으므로 특정 버전과 연관된 파일을 찾을 때 효율적으로 이용할 수 있다.

파일 이름에 의한 검색에서는 윈도내에서 와일드 카드 문자를 지원하며, 파일 이름에 의한 검색에서는 논리합 연산을 확장 부울리언 질의로 작성할 수 있도록 설계하였으며, 이를 위해 ; 기호를 사용한다. 지정된 파일의 크기와 같은 파일을 검색하는 기능도 추가하였으며, 콤보 상자에서 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 항목으로는 년, 월, 일을 기준으로 사용자가 작업한 파일을 검색하도록 하였다.

벡터 검색 모델은 사용자의 요구 사항을 표현하기 위해 질의 항목의 중요도를 나타내는 가중치를 할당하며, 검색시 가중치에 의해 소프트웨어 버전의 순위를 조절할 수 있다. 즉, 벡터 검색에서는 중요한 항목에 가중치를 부여할 수 있고, 출력될 객체의 수와 등급을 정할 수 있다. 부울리언 질의와 벡터 질의의 단점을 보완하기 위해 두가지 방법의 장점을 결합하여 버전 제어에 적합한 혼합 검색 모델을 제안한다. 자동화된 검색 시스템은 사용자가 작성한 질의 항목의 중요성을 반영해야 한다. 질의 항목의 중요성을 반영하기 위한 방법은 각 라이브러리 내의 버전들을 화면상에 표시할 때 재배열하는 것이다. 라이브러리의 버전들의 순서가 고정되어 있다면, 효율적으로 사용자들의 요구 사항을 만족시켜 주지 못한다. 본 논문에서는 이를 해결하기 위해 검색된 결과를 가중치에 따라 순서를 조정하여 사용자가 유사한 버전을 효율적으로 찾을 때 도움이 되도록 하였다.

3.3 형상형성 지원

Facet 분류 방식은 버전들의 공통 특성을 모아 facet를 구성한 다음, 각 facet에서 알맞는 항목을 선택하고 이들을 조합하여 버전들을 분류할 수 있으므로 버전 제어에 적합하다. 본 논문에서는 각 facet에 가중치를 할당한 확장 facet 방법을 사용하여 사용자가 빈번이 참조되는 facet에 높은 가중치를 할당하여 사용자가 특정 facet를 검색할 때, 도움이 되도록 하였다.

라이브러리에서 제안된 확장 facet 분류 방식에 의해 분류 작업이 완료되면, 다음 단계로 facet에 대해 가중치를 할당한다. 확장 facet 분류 방식에서 가중치 부여는 기본 facet이 사용자 인터페이스에 의해 참조될 때 자동으로 가중치 값이 증가되어, 많이 참조되는 facet은 가중치가 자동적으로 많이 부여되도록 설계하였다. 또한, 각 버전에 대해서는 앞에서 설명한 바와 같이 혼합 검색 모델에 의해 가중치가 할당된다. 이 가중치들은 검색 과정에서 질의 작성을 쉽게 하고, 유사한 버전을 검색하기 위해 사용한다. (그림 2)는 형상형성 지원 관리 도구들과 정보들을 나타내고 있다.

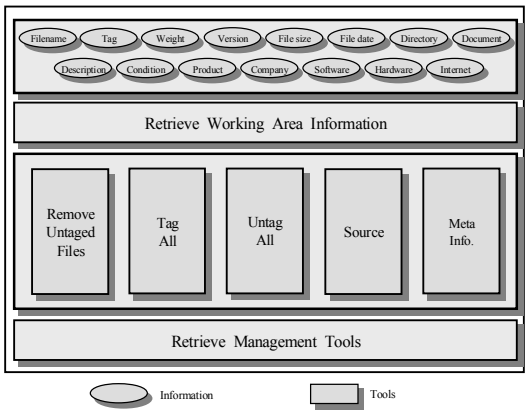

그림 2. 형상형성 관리 도구들과 작업정보
Fig. 2. Configuration thread management tools and work informations

일반적인 버전 제어 시스템에서 메타 정보는 각 버전들을 설명해 주는 데이터로서 시스템마다 메타 정보를 설계한 방법이 각각 다르게 사용되고 있다[14, 15, 16]. 이 메타 정보를 사용하여 각 버전에 대한 설명을 얻을 수 있으나 각 버전을 일일이 파악하고 확인한 다음 메타 정보를 얻을 수 있는 문제를 가지고 있으므로 특정 버전을 선택할 때 형상 형성 도구들을 사용한다. 또한, 각 시스템은 각 버전에 대한 메타 정보를 시스템에 맞게 특성화하여 정의·사용한다. 본 논문의 혼합 검색 모델을 사용하여 사용자가 라이브러리에 새로운 facet를 구성한 경우에는 혼합 검색 시스템과 연계하여 메타 정보를 형상 형성 정보로서 직접 입력할 수 있도록 설계하였다. 혼합 검색 모델을 사용하여 기존의 facet에서 특정 버전을 검색할 경우에는 사용자가 메타 정보를 변경할 수 있다. 또한, 이들 정보는 버전 제어에서 형상 형성을 위한 정보로서 제공될 수 있도록 통합 화면으로 사용자 인터페이스를 설계하였다.

두 개의 버전 간의 차이를 제공해 주기 위해 본 연구에서는 Visual C++상의 MFC에서 사용되는 windiff와 diffexec를 사용하여 두 버전간의 차이를 위한 프로그램을 구현하고, 이 결과를 메타 정보로서 저장하도록 하였다.

3.4 사용자 인터페이스

다음은 본 논문에서 제안한 형상 형성 지원을 위한 혼합 검색 시스템을 수행할 때 화면상에 나타나는 버튼들을 나타낸다. 다음 (그림 3)은 사용자 인터페이스를 위해 버튼들을 가지고 설계한 것이며, (그림 4)는 통합화면에서 사용되는 주요 메뉴들을 나타낸다.

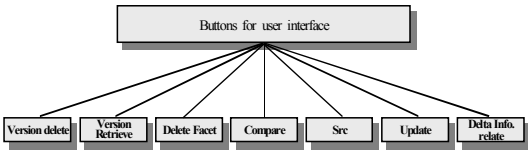

그림 3. 사용자 인터페이스를 위한 버튼들
Fig. 3. Buttons for User information

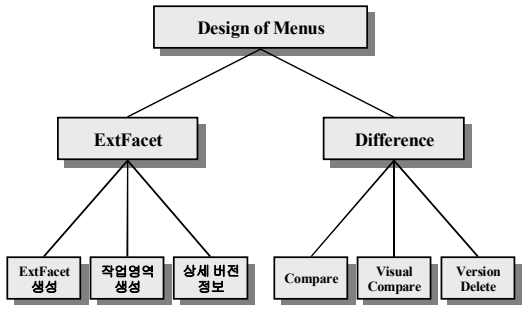

그림 4. 주요 메뉴의 설계
Fig. 4. Design of Main menu

Ⅳ. 시스템의 구현

4.1 시스템 구성

본 논문에서는 소프트웨어의 버전 제어를 위해 소프트웨어의 구성요소를 라이브러리에 저장 및 생성하고, 사용자의 요구사항과 일치하는 버전을 검색하기 위해 부울리언 검색 모델과 벡터 검색 모델을 장점을 가지고 버전 제어에 적합하게 구성한 시스템과 이에 대한 사용자 인터페이스를 제공하는 형상 형성 관리 프로그램을 구현하였다. 그리고 이에 대한 시스템을 설계하여 Visual C++ 언어[17, 18, 19]로 구현하였다. 그리고 이 시스템을 다른 형상 형성 방법과 비교 분석하여 제안한 방법이 다른 형상 형성 방법에 비해 효율적인 것을 평가하는데 그 목적이 있다. 시스템 구성은 (그림 5)와 같다.

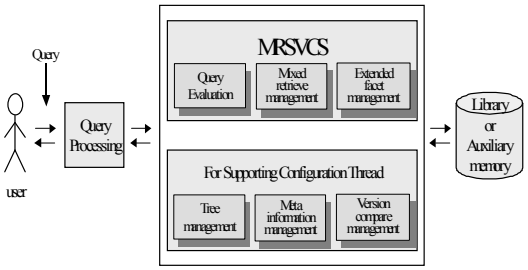

그림 5. 시스템 구성
Fig. 5. System structure

4.2 모델링 시스템

CompVersion 클래스는 대화상자에서 두 개의 버전을 입력받아서 이를 비교하여 처리하기 위한 클래스이다. CopyProject 클래스는 특정 facet를 사용자가 작업 영역에 복사해 주는 작업을 처리한다. 작업 영역에서 변경된 사항은 모델링 시스템을 사용하거나 사용자에 의해 라이브러리로부터 제공된 facet을 후에 수정하여 새로운 facet을 생성할 수 있도록 한다. VersionCont는 주로 facet과 연관된 작업으로서 기본 사용자 인터페이스 화면에서 처리해 주는 작업을 수행한다. 이 작업들은 다음과 같은 작업을 포함한다. 임의의 facet을 읽어서 facet들의 리스트를 나타내며, 각 facet에 포함된 파일들을 사용자 인터페이스에 의해 표시할 수 있도록 수행한다. 그리고 사용자에 의해 facet이 참조되면 해당 facet에 대한 가중치를 증가한다. 또한, 사용자 인터페이스 부분에 선택된 facet의 메타 정보도 나타내 주는 역할을 한다.

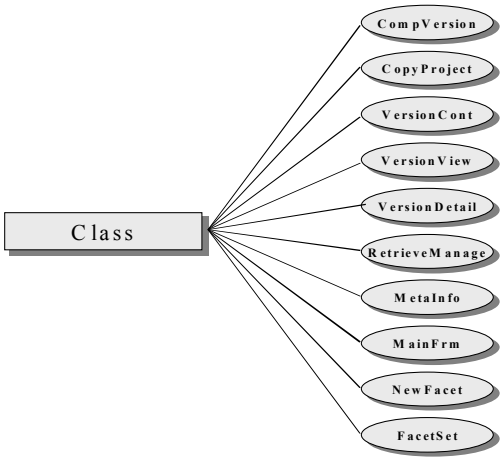

그림 6. 시스템 클래스의 구성
Fig. 6. Structure of system class

VersionDeatil 클래스는 facet의 생성과 삭제, facet에 대한 설명을 추가할 수 있는 기능을 지원한다. VersionView 클래스는 트리 관리를 위한 클래스로서 라이브러리인 facet 구조에 저장된 버전 번호를 트리 구조를 이용하여 표시해 주는 역할을 수행한다. RetrieveManage 클래스는 혼합 검색 시스템을 위한 함수와 동작 모듈을 포함하고 있다. 즉, 벡터 검색과 부울리언 검색 방법들에 대한 알고리즘들을 포함하며, 확장된 질의를 지원하기 위한 파싱 프로그램을 포함한다. MetaInfo 클래스는 메타 정보를 정의하며, NewFacet 클래스는 facet과 연관된 facet을 관리하기 위해 필요한 클래스이다. 이 클래스에서는 또한 facet에 대한 가중치 초기값을 설정한다. MainFrm 클래스는 CompVersion 클래스와 연계되어 있으며, 실제 두 개의 버전 간의 비교 작업은 이 클래스에서 수행된다. FacetSet 클래스는 facet에 대한 가중치를 처리한다.

4.3 형상 형성 정보 관리

형상 형성 정보 관리는 (그림 5)에서 설명한 바와 같이 라이브러리 관리를 포함하여 트리 관리, 메타 정보 관리, 버전 비교 결과를 메타 정보에 연계하기 위한 부분으로 구성된다. 그리고 트리 관리, 메타 정보 관리는 3장에서 설계한 바와 같이 한 화면상에서 바로 형상 형성 정보를 확인할 수 있도록 통합환경에서 구축함으로써 향상된 추적 관리 기능을 제공한다.

트리 구조를 지원하기 위해 Visual C++의 리소스 에디터에서 CTreeCtrl 클래스를 사용하여 사용자가 변수를 설정하여 리스트 컨트롤 상자를 가지고 구현하고, 이를 DDX_Control을 사용하여 사용자가 지원 변수와 클래스를 연결시킨다. 다음은 형상 형성 정보 관리 부분에서 특정 facet을 선택한 경우에 해당 버전을 트리 형태로 표시하고 관리하기 위한 동작 모듈이다.

트리 정보 처리는 시스템에 의해 새로운 버전이 생성되거나 변경되는 경우에 변경된 버전에 대한 트리 구조를 새로이 나타내 준다. 이 트리는 새로운 리비전들이 같은 전위 리비전으로부터 유도되었는가를 식별해 주는 역할을 하며, 유도된 리비전의 식별에 대한 서비스를 제공한다.

다음 (그림 7)은 라이브러리를 구성하는 facet 관리를 위해 본 논문에서 제안하여 구현한 내용이며, facet들에 대한 작업을 나타낸다. “Extende Facet Make”는 사용자가 새로운 facet을 생성하며, 이것은 VersionView 클래스의 지원과 RetrieveManage 클래스에 의해 처리된다. 사용자가 새로운 facet을 생성하기 위해서는 사용자 인터페이스 화면에서 “ExtFacet” 다음의 콤보 상자에서 새로운 facet 이름을 입력한 다음 “Version Retrieve” 버튼을 이용한다.

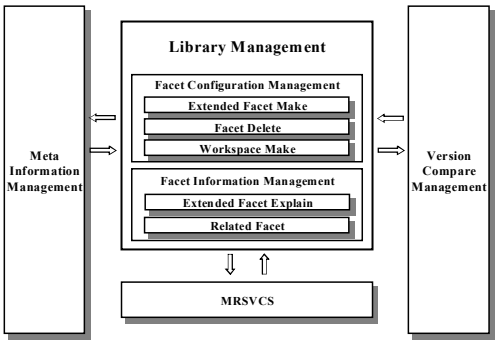

그림 7. 확장 Facet 관리 작업
Fig. 7. Extended facet management work

또한, 각 작업영역을 처리하기 위해 본 논문에서는 하나의 파일에 각 작업영역에서 처리한 메타 정보들을 모아 facet 단위로 관리하며, facet은 확장 facet 개념을 지원하여 facet 자체에 가중치를 할당하여 많이 참조되는 facet을 사용자가 선택할 때 도움이 되도록 하였다. 이 것은 3장에서 설명한 바와 같이 모델링 시스템에서 사용하는 벡터 검색을 이용하여 중요한 항목에 가중치를 부여하고, 출력될 객체의 수와 등급을 처리할 수 있도록 사용자 인터페이스를 구현하였다.

본 논문에서는 버전 제어 시스템에서 델타 정보의 일부인 두 버전간의 비교 결과를 메타정보에 자동으로 연결하여, 후에 특정 메타 정보와 관련된 버전 정보의 파악이 용이하도록 하였다. 이를 위해 버전 비교 작업은 MFC 라이브러리에 있는 프로그램을 사용하여 구현하였으며, 앞에서 설명한 향상된 추적 관리 기능을 제공하기 위하여 통합 화면으로 사용자 인터페이스를 구성하였다. 두 버전 간의 비교 결과는 메타 정보에 포함되며, 새로운 리비전에 의한 영향 판단과 특정 버전에 대한 형상 형성 정보를 얻을 수 있으며, 효율적인 형상 형성을 지원할 수 있다.

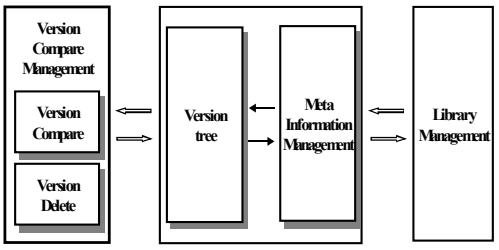

그림 8. 버전 비교 작업
Fig. 8. Version compare work

버전 비교 결과는 라이브러리의 해당 버전이 속하는 facet에 저장되며, 이 정보는 (그림 8)에서와 같이 메타 정보에 포함되어 처리된다. 또한, 버전 트리, 확장 facet 처리, 버전 처리, 메타 정보 등을 통합화면에서 처리하였으며, 본 논문의 부울리언 검색과 벡터 검색을 가지고 제안한 혼합 검색 시스템을 사용하여 형상 형성을 효율적으로 지원할 수 있다.

V. 실증적 적용 및 평가

5.1 적용 사례

이 적용 사례에서는 본 논문에서 제안한 형상 형성 지원을 위한 모델링 시스템이 다른 시스템에 비해 효율적임을 제시한다. 그리고 본 논문에서 구현한 시스템과 다른 시스템과의 실증적 비교를 위해 기존 국내 모 연구소에서 CCC/Harvest에 적용하여 사용하고 있는 일부 버전 제어 데이터들을 가지고 형상 형성 측면에서 분석 비교하여 성능을 평가하였다.

비교 평가로 사용되는 시스템으로는 SourceSafe, CCC/Harvest, Sablime을 가지고 수행하였으며, 다음 5.2절에서는 본 논문에서 제안한 시스템과 동일 데이터를 적용하여 문제점을 분석하고, 제안한 시스템의 타당성을 제시하였다. 본 논문에서 구현한 시스템과 다른 시스템에 적용할 데이터와 처리 방법에 대한 제약 사항은 다음과 같다.

1. 모델링 시스템이 다른 시스템과의 비교는 형상 형성으로 제한하여 비교하였다. 이것은 본 연구가 버전 제어에서

형상 형성 방법의 문제점을 해결하기 위해 모델링 시스템을 제안하고 구현하여 평가하는 것을 목적으로 하기 때문이다. 기존의 버전 제어 시스템에서 형상 형성 방법은 정의된 한가지 방법만을 사용하여 적절한 버전의 선택에 있어서 문제점을 가지고 있다. 즉, 형상 변이의 공존성에서는 리비젼이나 형상을 구별할 수 있는 방법을 효율적으로 지원해야 한다.

2. 형상 형성 방법과의 실증적 비교를 위한 데이터는 국내 모 연구소에서 CCC/Harvest에 적용하여 사용하고 있는 일부 버전 제어 데이터들을 가지고 여러 버전을 만들어 적용한다.

3. 2의 데이터를 가지고 제안한 모델링 시스템, SourceSafe의 Label, CCC/Harvest의 Package 등과 비교 분석하여 평가한다. 비교 평가의 핵심 주제는 다음과 같다.

   ① 미리 정의한 정보인 형상 규칙이나 정의된 정보 등에 의해서만 시스템에 대한 형상 형성 정보를 제공하는 문제이다. SourceSafe는 오직 레이블을 통해서만 필요한 버전을 선택한 다. 개발 당시에 미리 의미 있는 레이블을 정의하고, 명칭을 붙이지 않으면 추후에 다시 정의하는 것이 불가능하다. ClearCase는 각 버전들의 시스템 구조, 특성 등을 미리 입력하게 하고, 이 정보를 이용하는 형상 규칙을 적용하여 필요한 버전을 선택한다. 그러나 이 방법도 앞의 경우와 같이 규칙에서 사용하는 메타 정보가 미리 사용자에 의해서 정의되어 있어야 하며, 또한 이 정보를 사용하는 규칙 자체가 복잡한 단점을 가진다. CCC/Harvest는package나 package group을 이동시킴으로써 필요한 버전들을 선택할 수 있다. 상태를 사용함으로써 복잡한 규칙을 알아야 하는 필요는 없지만, 이 역시 미리 package로 정의된 관계 이외에는 여러 파일들의 버전 선택이 매우 어렵다는 단점을 가진다.

   ② 시스템 전체에 대해서 사용자가 정의한 정보를 제공하지만, 정의되지 않은 세부적인 정보는 제공하지 않는 문제이다. 예를 들어, 특정 파일의 특정 버전과 관계있는 다른 파일의 버전을 선택하는 경우를 가정하자. ClearCase의 경우에는 임의로 hyperlink라는 관계를 이용하거나 컴파일 등을 통한 유도된 관계의 경우에는 자동적으로 "BOM"을 작성하여 이를 지원하고 있다.

4. 기타 다른 시스템과의 장단점을 분석하기 위해 사용자 인터페이스, 형상 형성 방법, 세부 정보 제공 여부, TCP/IP 지원, Visual Difference, Life cycle 지원, 저장소, 확장성을 가지고 분석한다.

본 논문에서 제안한 시스템과 이의 형상 형성 지원 프로그램을 위한 화면은 (그림 9)와 같으며, 형상 형성을 위한 버전 검색 작업은 (그림 10)과 같다.

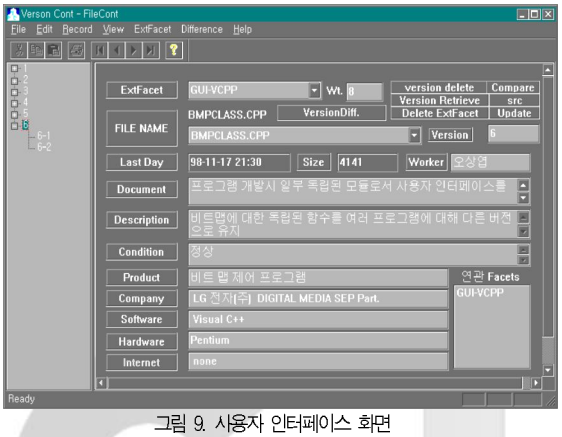

그림 9. 사용자 인터페이스 화면
Fig. 9. User interface screen

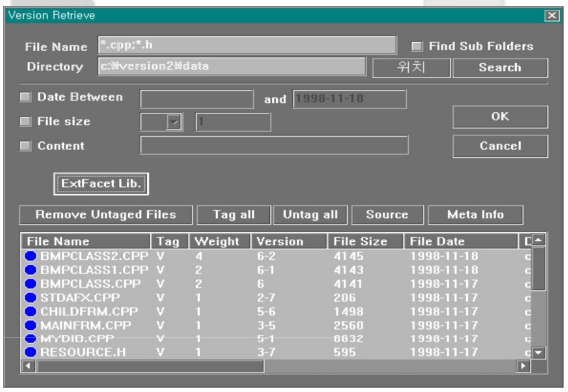

그림 10. 형상 형성을 위한 버전 검색 작업
Fig. 10. Version retrieve work for configuration thread

(그림 10)은 형상 형성을 위해 확장 부울리언 질의를 사용하여 검색하고, 벡터 검색에 의해 가중치가 부여된 내용을 나타낸다. 부울리언 검색은 유도된 리비전과 문서의 정확한 리비전 판단에 사용된다. 벡터 검색은 새로운 리비전에 대한 영향 판단을 가중치에 의해 제공된다. 본 연구에서

구현한 시스템에서는 다양한 형상 형성방법을 사용하므로 형상 형성 지원을 사용자에게 확대하는 결과를 제공한다. (그림 10)에서는 확장 부울리언 질의를 적용한 데이터를 가지고 검색한 결과를 제시하였다. 기존 facet들에 정의되지 않은 버전들의 파일을 파악할 수 있으며, 현재 작업중인 facet에 없는 버전일지라도 다른 facet에 해당 버전이 있으면 그 결과를 사용자에게 검색하여 바로 제시할 수 있는 장점이 있다.

5.2 비교 및 평가

본 논문에서 제안한 버전 제어 시스템은 사용자 인터페이스, 형상 형성 방법, 세부 정보 제공 여부, TCP/IP 지원, Visual Difference, Life cycle 지원, 저장소, 확장성을 고려하여 비교 분석하였다.

표 1. 다른 시스템과의 비교 결과
Table. 1. Compare results of Other systems

시스템 / 기준	SourceSafe	CCC	Sablime	제안 방법
사용자 인터페이스	GUI	CUI	GUI	GUI
형상 형성 방법	정의된 정보 (레이블과 핀)	정의된 정보 (패키지)	정의된 정보 (MR)	정의 및 비정의된 정보
세부 정보 제공	×	×	×	○
TCP/IP	×	○	○	×
Visual Difference	×	○	○	○
Life cycle	×	○	○	○
저장소	프로젝트 저장소	프로젝트 저장소	파일	라이브러리와 보조기억 매체
Release & Variants	File sharing	Virtual Copy	Branching & Labeling	Branching & Labeling

※ ○ : 가능, × : 불가

VI. 결론

본 논문은 Visual C++ 언어를 사용하여 소프트웨어 버전 제어에서 형상 형성을 지원하기 위한 모델링 시스템을 설계 및 구현하였다. 형상 형성 정보를 효율적으로 제공하기 위해 부울리언 검색 모델과 벡터 검색 모델을 결합한 혼합 검색 모델을 제안하였으며, 라이브러리는 확장된 facet 방법을 응용하여 설계하였다. 본 연구에서 제안한 시스템은 기존 버전 제어 방법에서의 형상 형성 방법에 비해 다음과 같은 장점을 가지고 있다.

첫째, 본 논문에서 제안하는 시스템에서는 사용자는 복잡한 형상 규칙을 사용하는 대신 간단하고 유연성 있는 질의 구성을 할 수 있으며, 대규모 라이브러리에 효율적으로 적용할 수 있다. 그러므로 기존의 형상 규칙 사용에 비해 효율적인 형상 형성 정보를 제공해 준다.

둘째, 사용자의 의도를 정확히 표현할 수 있는 부울리언 질의로 검색 성능을 높여주므로 버전 제어 작업의 효율을 높일 수 있다. 그리고 이를 이용하여 세부적인 정보도 사용자에게 제공할 수 있다.

셋째, 제안한 시스템을 기반으로 질의 항목의 중요도를 나타내는 가중치를 사용하여 출력되는 버전과 facet들의 등급을 나타내어 사용자가 적절한 버전 선택에 도움이 되도록 하였다.

넷째, 기존의 버전 제어에서는 메타 정보와 형상 형성 도구를 별도로 관리하고 있으나, 본 논문에서는 메타 정보를 통합환경에서 구축함으로써 향상된 추적 관리 기능을 제공한다.

다섯째, 일반적으로 메타 정보는 특정 버전을 탐색한 경우에만 참조할 수 있다. 그러나 본 논문에서 제안한 시스템을 사용하여 메타 정보를 가지고 버전을 탐색할 수 있도록 하여 형상 형성 측면에서 이용할 수 있도록 하였다.

앞으로는 소스뿐만 아니라 설계문서, 사용자 매뉴얼에 대한 버전 관리와 통계 보고 기능을 제공하는 시스템에 대한 연구가 추가되면 더 효율적일 것이다.

참고문헌

[1] Marc J. Rochkind, "The Source Code Control System", IEEE transactions on Software Engineering., vol. 1, no. 4, pp. 364-370, 1975

[2] Team Development Overview, URL : http://www.silicom/~alobba/overview.html

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

[4] W. F. Tickey, RCS - A System for Version Control, Software Practice and Experience,Vol. 15, 637-654, 1985

[5] 이태흔, 우치수, "포인터 저장 알고리즘을 사용한 소스 버전 관리 도구의 설계 및 구현" 한국 정보과학회 논문지 제21권 9호, 1994

[6] 오상엽, 김홍진, 김영선, "UML을 이용한 컴포넌트 버전 제어 시스템 설계", 한국 컴퓨터 정보학회 논문지, 제 8권 제 1호, 2003. 3

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

[8] 이태흔, "분산 소프트웨어 개발에서 관계관리 지원을 위한 형상 관리 기법", 서울대학교 계산 통계학과 박사학위 청구논문, 1997.2

[9] 채은주, 한정수, 김귀정, "컴포넌트 형상관리를 위한 인터페이스 설계에 관한 연구" 제20회 한국정보처리학회 추계학술발표대회 논문집, 제10권 제20호, 2003. 11

[10] 오상엽, 장덕철, "구성요소의 버전 제어를 위한 객체지향 검색프레임 워크", 한국정보과학회 논문지(C) 제3권 4호, 1997. 8

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

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

[13] P. Devanbu, et. al, "LaSSIE : A Knowledge Based Software Information System", CACM Vol. 34, No. 5, pp. 34 - 49, 1989

[14] C. V. Ramamoorrthy, "The Evolution Support Environment System", IEEE Transactions on Software Engineering, vol. 16, no. 11, pp. 1225 - 1234, 1990

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

[16] 오상엽, 장덕철, "원시 코드의 메타 정보 관리를 위한 버전 제어 시스템의 설계와 구현", 한국 정보처리학회 논문지 제 5권 3호, 1998. 3

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

[18] Kate Gregory, Special Edition Using Visual C++ 4.2, 1996

[19] 이상엽, Visual C++ 5, 영진출판사, 1998

저 자 소 개

오 상 엽

1999년 광운대학교 대학원 전자계산학과(이학박사)

1993~현재 경원전문대학 교양과 부교수

<관심분야> 소프트웨어공학, 형상관리, 객체지향