韓國 컴퓨터情報學會 論文誌
第13卷 第6號, 2008. 11.

2008-13-6-1-6

버전제어에서 컴포넌트의 형상형성 제어를 위한 객체지향 라이브러리

오 상 엽*, 최 우 승**

Object-Oriented Library System for Configuration Thread Control of the Component in Version Control

Oh, Sang Yeob*, Choi, Woo Seung**

요 약

버전 제어 시스템은 급속한 환경의 변화나 개발 환경이 복잡한 프로그래밍 환경에서 사용되고 있으며, 정의된 형상 규칙 정보를 정확하게 알지 못하는 경우나 미리 정의되지 않은 정보에 대해서는 형상 형성 정보 제공 문제가 발생한다.

본 논문에서는 사용자가 원하는 컴포넌트를 복합적으로 형상형성 제어 할 수 있는 라이브러리 시스템을 제안하고, 모델링하여 구현하였다. 제안한 형상형성 제어를 위해 사용되는 라이브러리는 확장 facet 분류를 응용하여 하부 표현 구조에 관계없이 라이브러리로부터 컴포넌트를 검색할 수 있도록 하였다. 이것은 본 논문에서 제안한 TreeSearch 클래스와 형상형성 제어 함수를 이용하여 관리된다. 본 논문의 라이브러리는 다른 언어와의 인터페이스를 통해 사용될 수 있으며, 사용자에 의해 facet이 확장되는 장점을 가진다.

Abstract

A version control system is used in a rapidly changed environment or a program which developed in a complicated environment. 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.

In this paper. Library system is suggested, modelled, and implemented so as to configuration thread control the components required by the user in many ways. As for the library used in the configuration thread control suggested in this paper, the components can be retrieved from the library regardless of the infrastructure, applying the extended facet classification. This retrieval framework is managed using TreeSearch class and the configuration thread control function. The library system of this paper can be used by the interface with other languages, and this system is to have a advantage to extend a facet by user.

▸ Keyword : Version control, Configuration thread, Software classification, Extended facet, Library system

• 제1저자 : 최우승 교신저자 : 오상엽
• 접수일 : 2008. 8. 29, 심사일 : 2008. 9. 19, 심사완료일 : 2008. 11. 26.
* 경원대학교 IT 대학 컴퓨터 소프트웨어 ** 경원대학교 교양대학

Ⅰ. 서 론

현대 정보화 사회에서 소프트웨어 수요는 증가하고 있는 반면 소프트웨어 생산성과 품질은 향상되지 못하고 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[1,2,3].

형상형성제어 시스템은 사용자로 하여금 관심 있는 영역만을 탐색할 수 있고, 컴포넌트(component)에 대한 정보를 제공하여야 하며, 이는 라이브러리 관리를 위한 기반 시스템이 된다. 버전 제어를 위한 컴포넌트를 등록하고 저장하는 것은 버전 제어에서 기본이 되는 기능이다. 사용자가 시스템을 구성할 때 적절한 버전을 선택하는 문제이며, 이를 형상 형성이라고 한다. 컴포넌트들의 집합이 크고, 그 컴포넌트들이 여러 응용 분야에 걸쳐 널리 사용되면서 우수한 모듈성을 갖는 환경에서는 필요한 컴포넌트를 형상형성제어하고 식별하는 것이 중요한 문제로 제기된다[4,5,6]. 기존의 형상형성제어 방법은 대부분 키워드를 기반으로 한 개념 등을 사용하여 파일을 형상형성 제어하였으며[7], 다른 시스템과의 인터페이스를 제공하지 못하고 있다. 이를 위해, 본 논문에서는 객체지향 프로그래밍 언어(OOP, Object Oriented Programming)인 C++ 언어를 이용하여 키워드, 파일 내용, 파일 크기 등을 사용하여 컴포넌트를 찾는 형상형성제어 시스템을 제안하며, 이는 다른 시스템과의 인터페이스를 통해 사용될 수 있다. 본 논문에서 제안한 형상형성제어 시스템은 버전 제어와 델타(delta) 관리를 지원한다.

본 논문의 형상형성 제어시스템에서는 첫째, 라이브러리 구성을 확장된 facet 분류를 응용하여 하부 표현 구조에 관계없이 라이브러리로부터 컴포넌트를 형상형성 제어할 수 있도록 하였다. 이것은 본 논문에서 제안한 TreeSearch 클래스와 형상형성 제어 함수를 이용하여 관리된다. 둘째, 형상 형성 정보를 효율적으로 제공하기 위해 부울리언 검색 모델과 벡터 검색 모델을 결합한 혼합 검색 모델을 제안하였다. 이를 위해 제안된 방법에서 라이브러리는 확장 facet 방법을 응용하여 설계하였다. 셋째, 기존에 연구된 형상형성 제어 시스템들이 유용하지만 다른 시스템과의 인터페이스를 제공하지 않으며, 일부 단위로만 관리하므로 포괄적이지 못하여 제한적으로 사용되고 있다. 본 논문에서는 객체 지향적인 특성을 반영한 효율적인 형상형성 제어 시스템을 구현하여 사용자가 원하는 컴포넌트를 효율적으로 관리할 수 있는 형상형성 제어 시스템을 제안하고, 이것은 버전 제어와 델타 관리 작업을 위한 기반 시스템이 된다. 넷째, 기존의 연구 방법 중에는 순수한 객체지향 개념만을 도입하여 클래스를 구성요소로 처리한 논문도 있으나[5], 전체적인 응용 영역을 고려할 때 클래스를 포함하여 일반 프로그램을 구성하는 함수나 모듈도 구성요소로 처리할 수 있도록 설계하였다.

성능 분석을 위한 연구 범위는 형상 형성으로 제한하여 비교하였다. 이것은 본 연구가 버전 제어에서 형상 형성 방법의 문제점을 해결하는 것을 목적으로 하기 때문이다. 본 논문의 제 2장에서는 버전제어, 라이브러리, 형상형성 관리에 대한 관련 연구를, 제 3장에서는 시스템 모델의 설계와 방법론, 제 4장에서는 구현 방법과 실행 예를 다루고, 본 논문에서 제시한 시스템을 [5, 16]에서 제시한 평가 기준을 가지고 분석을 하였으며, 5장에서는 결론 및 향후 연구 과제를 기술하였다.

Ⅱ. 관련 연구

2.1 버전 제어

프로젝트 개발과 유지보수되는 동안 컴포넌트는 변경되며, 이러한 변경(changes)의 집합인 cut, paste, insert, delete 등의 작업으로 컴포넌트의 한 버전들을(연속적인) 다음 버전으로 변환시키는 이력 과정(history step)이 버전 제어이다[8,9].

(그림 1)은 버전 제어 과정을 나타내며, 버전제어 과정중의 각 이력을 델타 형태로서 관리한다.

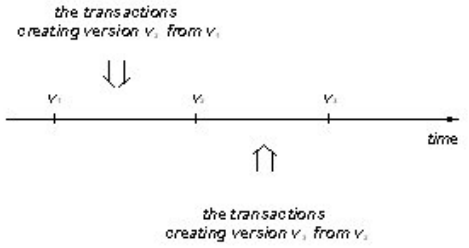

그림 1. 버전 제어 과정
Fig. 1. Version Control Process

버전 제어를 지원하는 주요한 도구들의 특성을 비교 분석하여 표 1에 나타내었다[3,9,10].

표 1. 버전제어 시스템의 비교 분석
Table 1. Comparison analysis of Version Control System

종류 / 구분	SCUMS	RAS	DREE	CRC	NAE
적용 대상	source code	source code	Ada source code, object code	Text	Text
동시성 관리	check-in check-out	check-in check-out	check-in check-out	check-in check-out	Copy - Modify- Merge
이력 관리	internal annotation	log message	modification request	log message	
버전 유형	revision variation	revision variation	revision variation	revision	object manage-cent system
버전 트리 디스 플레이	제공 안함	제공 안함	제공함	제공 안함	제공 안함

2.2 라이브러리

유사한 컴포넌트의 선택은 분류 문제이며, 유사성의 정도는 어떻게 집합(collection)을 조직하느냐에 따라 정해진다. 그러므로 집합의 조직과 선택은 이 모델에 있어서 중요하다 [11,12].

소프트웨어 컴포넌트의 관련성을 부여하는 방법에 따라 enumerative와 facet 방법[13], 어휘처리 수준을 포함하는 분류 방법[14], 그리고 구문 및 의미 분석을 포함하는 방법 [15]이 있다.

컴포넌트 라이브러리의 예로는 Raytheon Company에서 3200개의 코볼 코드 컴포넌트로 라이브러리를 구성하였으며, AT & T Pacific Bell사에서 250명의 개발자에게 공유된 약 1000개의 C언어 컴포넌트를 가지고 라이브러리를 구성하였고, 이외에도 Prieto Diaz의 라이브러리, EIFFEL, LaSSIE, AIRS, ROSE사 등에서 구축한 라이브러리가 있다.

2.3 형상형성 관리

형상형성 관리는 여러 리비젼이나 형상의 공존을 허용할 수 있어야 한다. 이런 지원을 위해서는 각 문서에 이름뿐만 아니라 리비젼이나 형상을 구별할 수 있는 레이블을 지원해야 하며, 또한, 형상관리는 형상 C가 문서 Di에 의존적인 모든 문서들이 새로운 리비젼을 갖게 되었을 때 Di의 새로운 리비젼을 생성할 수 있어야 한다. 이는 문서에 영향을 주는 모든 전위 문서들이 이 리비젼을 공유할 수 있는가에 대한 무결성을 점검함으로써 얻을 수 있다. 형상관리는 유도된 새로운 리비젼들이 같은 전위 리비젼으로부터 유도되었는가를 식별할 수 있어야 하며, 주어진 특정한 형상을 구성하기 위해서 사용되는 문서들 각각의 정확한 리비젼들을 찾을 수 있어야 한다.

형상형성 지원을 위해서 SourceSafe는 레이블, DSEE는 시스템 모형, ClearCase는 뷰, CCC/Harvest는 package와 package group, Sablime은 MR (Modification Request) 등을 사용한다.

III. 형상형성 제어를 위한 라이브러리 설계

3.1 시스템 모델

본 논문에서 개발한 시스템 모델은 형상형성 제어 시스템과 이의 지원을 받는 라이브러리, 컴포넌트를 관리하기 위한 관리 시스템, 브라우져, 사용자 질의어 및 인터페이스로 구성되며, 구성도는 다음 (그림 2)와 같다.

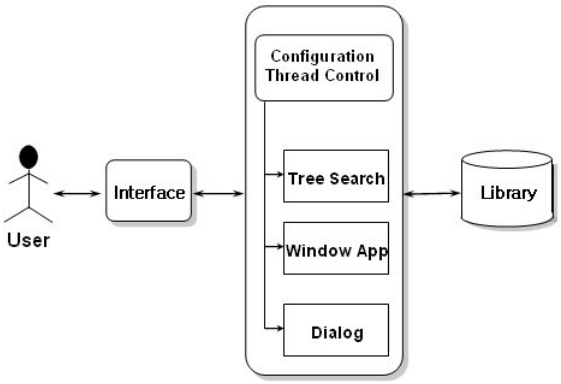

그림 2. 시스템 모델의 구성
Fig. 2. Construction of system model

3.2 시스템 모델의 기능

1) 형상형성 제어 시스템

컴포넌트의 형상형성 제어 시스템은 시스템 모델의 전반적인 기능 중 가장 중요한 기능으로써, 버전 제어를 위한 중요한 기능을 제공한다.

컴포넌트의 검색 기능을 위하여 컴포넌트 자체 내에 컴포

넌트에 대한 정보 또는 버전 제어 과정에 도움이 될 수 있는 주석을 포함한다. 즉, 컴포넌트의 소재, 컴포넌트의 복잡도, 버전 제어에 필요한 절차나 주의사항 그리고 관련 문서 등의 정보를 컴포넌트 내에 작성한다. 컴포넌트의 형상형성 제어에는 이러한 주석 또는 컴포넌트에 기술된 특정 명령어를 가지고 찾도록 한다. 이를 위한 질의어는 사용자가 컴포넌트를 찾는데 필요한 단어를 사용하여 작성하도록 설계하고, 검색 기능에서 각 단어를 가지고 이에 관련된 컴포넌트를 찾도록 한다. 이러한 과정은 사용자가 입력한 질의어를 통해 자동적으로 수행되도록 하여 시스템을 관리하고 확장하는 부담을 줄일 수 있게 하였다.

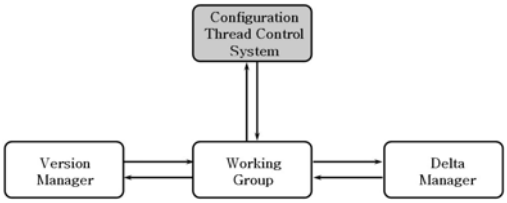

그림 3. 버전 관리, 델타 관리, 그리고 형상형성 제어 시스템과의 연관관계
Fig. 3. Version manager, delta manager, and relationship of configuration thread control system

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

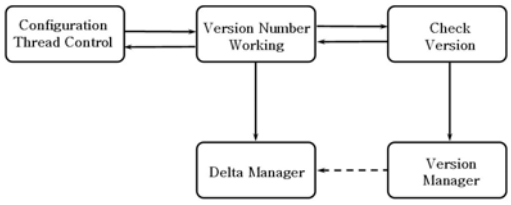

그림 4. 형상형성 제어 시스템의 사용
Fig. 4. Use of Configuration thread Control System

다음 (그림 5)는 버전 제어, 델타 관리, 그리고 형상형성 제어 시스템의 각 기본 도구를 보여준다.

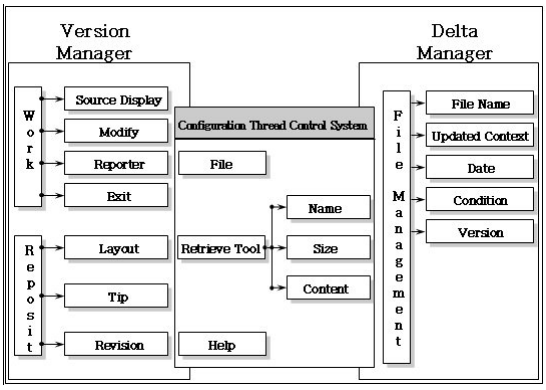

그림 5. 각 도구들의 기능 구성
Fig. 5. Function structure of each tools

2) 라이브러리

본 논문에서는 확장된 facet 방법을 사용하여 라이브러리를 구축한다. 이 방법은 새로운 객체를 추가하기 쉽고 라이브러리 확장에 쉽게 적응되지만 컴포넌트들의 계층적 관련성 표현이 어려운 단점이 있다. 또한, 현재 계층적 관련성 표현의 자동화는 시스템의 규모가 커지고, 소프트웨어의 규모가 커짐에 따라 복잡한 문제점을 가지고 있다. 이 문제를 해결하기 위해 본 논문에서는 사용자가 사용자 인터페이스를 통해 본 논문에서 제시한 형상형성 제어 시스템 사용할 때에 작업으로 각 컴포넌트들 간의 계층 관계를 관리하여 제한적으로 해결하였다. 컴포넌트가 점진적으로 증가하여 라이브러리의 규모가 증대되면, 이 방법은 문제가 발생될 수 있다. 이 문제 해결을 위해 컴포넌트를 검색하기 위한 find() 함수를 구현하여 컴포넌트를 검색하고, 다른 판단 기준을 만족하는 컴포넌트를 찾는 next() 함수를 호출한다. 이는 검색 메카니즘에서 이용되며, 내부에 search() 함수를 두어 재귀적으로 원하는 컴포넌트를 검색하도록 하였다.

Ⅳ. 형상형성 제어를 위한 라이브러리의 구현과 분석

4.1 클래스와 메소드의 구성

형상형성 제어 시스템에서는 C++에서 제공하는 강력한 사용자 환경(user environment)과 메시지 전송, 상속성, 동적 연결 등의 기본 특성을 이용하여 컴포넌트를 검색하고 카탈로깅한다. 이미 개발되어 저장된 컴포넌트가 컴포넌트

화가 잘 되어있고, 빌딩 블록(building block) 개념에 충실한 특성을 가지고 있으면, C++와의 명확한 인터페이스를 통하여 사용될 수 있다[2,11]. (그림 6)은 본 시스템에서 설계, 구현한 기본적인 클래스 구성도이다.

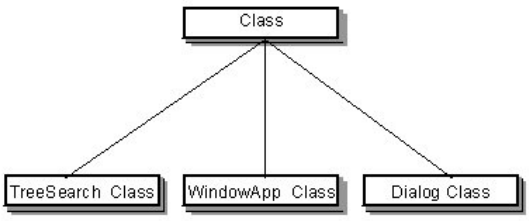

그림 6. 시스템의 클래스 구성
Fig. 6. Structure of system class

TreeSearch 클래스에서는 컴포넌트를 검색하기 위한 방법들을 지원하기 위한 클래스이며, WindowApp 클래스는 메뉴바와 서브메뉴 구축, 키보드와 컨트롤러를 지원하고, 이들을 사용해서 사용자가 윈도 클래스를 생성하여 컴포넌트의 형상형성 제어 시스템을 수행하는데 도움을 제공한다.

1) 검색 프레임워크

형상형성 제어 시스템은 구현한 TreeSearch 클래스를 이용하여 다양한 검색 기능을 수행하고, WindowApp 클래스와 Dialog 클래스를 사용하여 검색을 위한 환경을 지원한다. 형상형성 제어 시스템을 위한 TreeSearch 클래스는 다음과 같다.

```
class TreeSearch {
 public:
   TreeSearch(PTWindowsObject pParent, char drive);
   virtual ~TreeSearch();
   void Search();
 protected:
   virtual int checkFile(struct fsblk& fs) = 0;
   void print(struct fsblk& fs);
   char filePattern[13];
   int startDisk;
   char startPath[MAXPATH];
 };
```

형상형성 제어 시스템에서는 검색 메뉴를 위한 기본 메뉴를 설계 및 구축하고, TreeSearch 생성자내에서 디렉토리 구조를 트리로 보고 재귀적인 방법으로 각 디렉토리의 컴포넌트를 검색할 수 있도록 하였다. 다음은 Search() 함수에 대한 내용이다.

```
void TreeSearch::Search()
  {
    int done;
    .....
   // 루트디렉토리부터 하위 디렉토리를 탐색
   done = find("*.*", &fs, FA_DIREC);
   while (!done && !TCancelBox::IsDone()) {
       if (디렉토리 구조와 속성 비교) {
          if (strcmp() 함수를 이용해서 fs의 현재와
                상위디렉토리 컴포넌트 비교){
          .....
            Search();  // 재귀적 호출 . . .
            chdir("..");
           }
         }
         done = next(fs);
         // 메시지 루핑 수행
         .....
    }
```

fs는 찾고자 하는 컴포넌트 이름이며, 구조체 내에 정의되어 있다. 내부에서 find() 함수를 사용하여 각 서브 디렉토리를 트리 개념으로 검색한다. 크기에 의한 검색은 TreeSearchSize 클래스와 checkFile() 함수를 가지고 파일의 크기를 비교하였다. 다음은 TreeSearchSize 클래스이다.

```
class TreeSearchSize : public TreeSearch {
 public:
 enum matchType { lessThan, equalTo, moreThan
};
  TreeSearchSize(PTWindowsObject pParent, char
  drive, long siz, matchType t);
 protected:
    int checkFile(struct fsblk& fs);
    long siz;
    matchType match;
 };
```

내용에 의한 검색은 TreeSearchContent 클래스와 TreeSearch 클래스를 사용하여 재귀적으로 각 파일에서 사용된 내용인 단어를 가지고 검색하게 하였다.

2) 라이브러리

본 논문에서 제안한 라이브러리 구조는 루트 디렉토리에서 해당 디렉토리로 제한하면서 필요로 하는 소프트웨어 컴포넌트를 찾아 작업하므로 작업의 흐름을 파악할 수 있고, 다양한 검색방법을 지원하는 형상형성 제어 시스템으로 버전 제어 측

면에서 컴포넌트를 선택하는 부담을 줄일 수 있다. 또한 질의어에 의한 검색과 해당 작업 항목을 마우스로 선택하여 사용자의 편의성을 도모한다.

라이브러리는 지원되는 TreeSearch 클래스에 의해 관리되며, 버전 제어를 위한 컴포넌트를 관리할 수 있으며, 형상형성 제어 시스템을 사용하여 라이브러리에 컴포넌트를 등록, 검색할 수 있는 메소드와 함수를 지원한다.

4.2 실행 및 결과 분석

(그림 7)은 버전 제어에서 형상 형성 작업을 수행한 결과이며, 형상형성 정보를 제공하는 화면을 나타낸다. SourceSafe에서는 Label로 정의한 방법이외에는 형상 형성 정보를 제공하고 있지 않으며, CCC/Harvest에서도 Package로 정의되지 않은 버전에 대해서는 형상 형성을 제공하지 못하고 있다. 본 연구에서는 확장 부울리언 질의를 사용한 검색과 벡터 검색을 사용하여 다양한 형상 형성 방법을 제공한다. 기존의 도구들이 이와 같은 단점을 갖는 근본적인 이유는, 버전 선택을 어떻게 하면 개발자의 현재의 개발 환경을 쉽게 설정할 수 있게 할 것인가에 촛점이 맞추어져 있기 때문이다. 적합성 벡터에서는 한 리비젼을 유일하게 식별하였으나 본 연구에서는 다양한 형상 형성방법을 사용하므로 형상 형성 지원을 사용자에게 확대하는 결과를 제공한다. 이는 확장된 facet을 응용한 라이브러리와 TreeSearch 클래스, 그리고 형상형성 함수를 이용하여 관리된다.

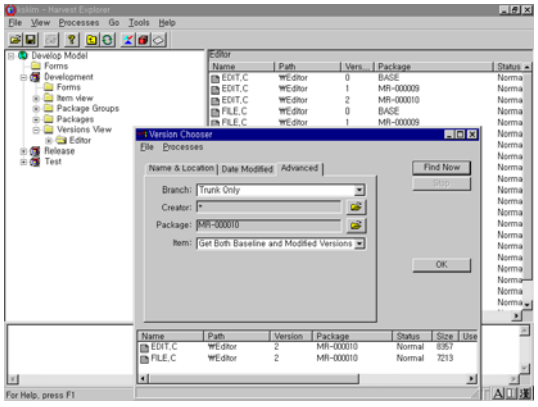

그림 7. 형상 형성 작업의 예
Fig. 7. Example of Configuration thread work

2) 비교 분석

본 논문에서 제안한 형상형성 제어 시스템은 다른 시스템과의 인터페이스 관계, 사용자 인터페이스, 검색 방법, 질의어, 분류구조, 확장성을 고려하여 다른 시스템과 비교 분석하였으며, 이는 [5,16]에서 제시한 평가 기준을 가지고 수행하였다.

3장과 4장에서 설계·구현한 바와 같이 본 논문에서 제안한 시스템은 version manager와 delta manager를 지원하며, 사용자 인터페이스에서 확장 facet 방법을 적용한 형상형성 정보를 이용한 검색 방법을 제공하여 검색의 효율을 증진시킬 수 있다. 또한, 본 논문은 확장 Facet 방법을 응용하여 디렉토리 구조를 트리 구조로 인식하여 이를 처리할 수 있는 find() 함수를 작성하였다. 본 논문의 검색 프레임워크는 C++의 확장성과 이식성을 고려하여 다른 시스템과의 인터페이스를 통해 사용될 수 있다. 이것은 기존에 연구된 형상형성 제어 시스템들이 유용하지만 다른 시스템과의 인터페이스를 제공하지 않으며, 일부 단위로만 관리하므로 포괄적이지 못하여 제한적으로 사용되고 있는 문제를 해결한다.

표 2. 평가 결과
Table 2. Test result.

시스템 / 기준	RSL	Diaz	[5]	제안 방법
사용자 인터페이스	메뉴사용	메뉴사용	시각적	시각적
검색 방법	키워드 매칭	키워드 매칭	시각적 추론	형상형성 정보이용
질의어	자연어	제한된 용어사용	시각화 제공	자연어, 시각화 제공
분류 구조	IR	Faceted	모두 지원	확장 Faceted 응용
확장성	새컴포넌트를 탬플리트에 추가	Faceted 확장	Cschem 확장	확장 Facet
인터페이스	불가	불가	가능	가능

V. 결론

본 논문은 소프트웨어 컴포넌트를 검색할 수 있는 형상형성 제어 시스템을 구현하였으며, 형상형성 제어 시스템의 지원으로 관리되는 라이브러리와 사용자 인터페이스를 제공하

는 시스템을 구현하였다. 이 시스템은 버전 제어를 위한 기반 라이브러리 시스템으로 사용된다.

본 논문에서는 제안한 검색프레임워크에서 사용하는 라이브러리는 확장된 facet 분류를 응용하여 하부 표현 구조에 관계없이 라이브러리로부터 컴포넌트를 사용자가 검색할 수 있도록 하였다. 또한, 시스템의 확장성을 제공한다. 시스템이 추가, 변경되는 경우 부분 수정으로 시스템을 변경할 수 있으며, 다른 시스템과 연계하여 인터페이스를 제공할 수 있다. 셋째, 시스템의 재사용성을 제공한다. 사용자가 라이브러리에 등록한 컴포넌트들을 사용자가 원할 때마다 사용함으로써 사용자의 작업 시간 단축과 일관성 있는 작업을 유도하여 품질 향상을 가져올 수 있으며, 본 논문의 형상형성 제어 시스템은 다른 언어와의 인터페이스를 통해 사용될 수 있다.

앞으로의 연구 과제는 현재 시스템의 제약 사항을 개선 및 보완해 나가며, 메타 정보에 대한 세부적인 다형성 관계, 클래스 관계에 대한 정보 제공 문제와 TCP/IP 기능 지원에 대한 연구가 보완되어야 한다.

참고문헌

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

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

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

[4] 김행곤, "소프트웨어 재사용 지원 정보 저장소 구축", 정보과학회·정보처리학회 공동 특집호, 제24권 제 11호, 2006. 11

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

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

[7] 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

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

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

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

[11] Keith E. Gorlen, "An Object-Oriented Class Library for C++ Program", Software -Practice and Experience. vol 17(12), pp.899-922, 1987.

[12] 박서영, 김갑수, 명선영, 신영길, 우치수, "객체지향 패러다임에서의 소프트웨어 컴포넌트 분류에 관한 연구", 한국정보과학회 논문지, vol. 20, no. 2, pp. 879-882, 1993.

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

[14] 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

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

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

저 자 소 개

오상엽(吳相燁)

1999년 광운대학교 대학원 전자계산학과(이학박사)

1993 ~현재 경원대학교 IT대학 컴퓨터소프트웨어 교수

<관심분야> 소프트웨어공학, 버전관리, 소프트웨어 재사용, 형상관리, 객체지향

최우승(崔宇勝)

1994년 동국대학교 공학박사(시스템공학)

1998년~2000년 (사) 한국컴퓨터정보학회 회장

2000년~현재 (사) 한국 컴퓨터정보학회 고문

1981년~2008년 현재 경원대학교 교양대학 교수

<관심분야> 비지도학습, 신경회로망, 퍼지논리시스템, 패턴인식, 소프트웨어 공학