컴포넌트 형상관리를 위한 요구명세에 관한 연구

A Study on Requirement Specification for Component Configuration Management

정대성, 한정수
천안대학교 정보통신학부
Dae-Sung Jung(f15cc@naver.com), Jung-Soo Han(jshan@cheonan.ac.kr)

요약

소프트웨어 개발방법이 컴포넌트기반 개발방법으로 진화하고 최근 들어 형상관리 시스템의 도입이 대두되고 있다. 본 논문은 컴포넌트기반 개발방법론에서 발전한 컴포넌트 형상관리를 다루고 컴포넌트 명세는 정규화된 문서인 XML을 기반으로 하고 있다. 컴포넌트 버전관리와 의존관리는 컴포넌트 이름으로 수행하였다. 컴포넌트 의존정보는 컴포넌트 명세가 합성될 때 합성에 사용된 컴포넌트 이름을 사용하여 이를 반영하고 의존정보로 사용하여 명세서를 통한 버전관리와 의존관리의 가능을 확인하였다.

■ 중심어 : | 컴포넌트 | 형상관리 | 명세서 | 버전관리 |

Abstract

Software development process evolves by component based development(CBD) process and introduction to configuration management system is risen recently. In this paper, we explained component configuration management system evolved out of CBD methodology and focused on the specification management in component configuration management. The component specification is being based on XML that is a normalization document. Also, component version management and dependency management are achieved by a component name. And component dependency information is just the integration information using a component specification name used in the composition when component specification is composed. Thus we confirmed possibility of version management and dependency management using the specification.

■ keyword : | Component | Configuration Management | Specification | Version Management |

I. 서 론

소프트웨어 개발방법이 진화함에 따라서 보다 빠르고 효율적인 개발이 필요하게 되면서 컴포넌트 기반 개발방법론(CBD : Component Based Development)이 제시되고 현재 대부분의 소프트웨어 개발에 컴포넌트기반 개발방법론이 적용되고 있다. 하지만 CBD로도 부족한 고객의 요구 및 시간으로 말미암아 형상관리 시스템을 도입한 개발방법에 이목이 집중되고 있다. 형상관리 시스템 SCM(Software Configuration Management)을 이용하게 되면 기존의 CBD 방식보다 빠른 개발환경의 구축이 가능하다. 형상관리를 이

접수번호 : #050701-001
접수일자 : 2005년 05년 07일
심사완료일 : 2005년 08년 01일
교신저자 : e-mail : 한정수, jshan@cheonan.ac.kr

용한 개발방법은 컴포넌트 재사용 이라는 형상관리 시스템의 최대 장점을 이용한 개발이 이루어지기 때문에 보다 빠르고 쉬운 개발이 가능하게 된다[1]. 또한 시스템 개발에 있어서 가장 중요한 부분은 요구사항 으로써, 요구사항은 개발기간 동안에 변경될 수 있기 때문에 불완전하고 일치하지 않는 정의는 프로세스 요구 설명이 완료되지 않거나, 요구사항 명세서가 잘 정의될 수 없다. 요구사항 명세서는 컴포넌트 조립 과정에서 가장 우선적으로 실행되는 부분으로서 이 명세서를 기반으로 하여 구현이 이루어지고 이를 통하여 컴포넌트를 생성하거나 새로운 버전의 컴포넌트를 얻게 된다. 이때 생성되는 명세서는 신규 명세와 기존 명세에서 업그레이드된 명세가 생성되는데, 업그레이드된 명세의 경우 버전관리가 이루어 져야 기존 시스템에 미칠 수 있는 악영향을 사전방지 할 수 있으며, 이후 컴포넌트 선택에 있어 효율적인 선택이 가능하게 된다. 본 논문의 구성은 다음과 같다. 2장에서는 관련연구로 컴포넌트기반 개발에 대해 간단히 소개하고, 3장에서 요구사항 명세서에 대해 기술하고 4장에서 이를 토대로 한 시뮬레이션을 하고 5장 추후 방향으로 끝을 맺는다.

II. 관련 연구

2.1 컴포넌트기반 개발

컴포넌트 기반 개발(Component Based Development)은 객체지향 개발방법론 이후에 재사용 목적을 위해 등장한 개발 방법론이다. CBD는 재사용 가능한 소프트웨어 모듈 컴포넌트를 생성 및 조립 생산, 선택, 평가 및 통합으로 구성하여 더 큰 컴포넌트를 생성하거나 완성된 애플리케이션 소프트웨어를 구축하는 개발기법이다. 컴포넌트 기반 개발 방법론은 개발된 컴포넌트를 조립하는 방식을 사용하며, 이미 개발된 컴포넌트를 재사용함으로서 빠른 시간 안에 새로운 애플리케이션(application)을 개발할 수 있다[2][3]. 따라서 시간과 비용의 절감과 더불어 새로운 시스템 구축에 있어 효율성을 극대화 할 수 있다. 이러한 장점 때문에 컴포넌트 기반 소프트웨어 개발기법이 대두되었으며, 이 컴포넌트 기반 개발은 소프트웨어개발 뿐만 아니라 건축산업에서도 컴포넌트를 기반으로 한 건축방법이 사용되어지고 있으며, 현재 건축방식은 컴포넌트를 기반으로 하여 수행되어 지는 것이 대부분으로 기존 방식보다 시간과 비용을 대폭 줄이는 효과를 보인다.

2.2 형상관리

소프트웨어 형상관리의 목적은 복합체를 구성하는 각 구성원(entities)을 관리하는 것이지만 대부분의 소프트웨어형상관리 기능은 개발과정에서 사용되고 런타임시에는 잘 활용되지 못하고 있다. 소프트웨어 형상관리의 주요 원칙은 버전관리(Version Management), 형상관리(Configuration Management), 변경관리(Change Management) 그리고 컴포넌트 관리를 위한 의존관리(Dependency Management)가 필요하다[4].

III. 요구사항 명세서

정규화된 형식으로 작성된 컴포넌트 명세서는 컴포넌트 형상관리에 있어서 가장 중요한 역할을 맡는다. 이 명세서는 컴포넌트의 정보를 가지며, 합성되어 새로 생성된 컴포넌트의 경우는 연결된 컴포넌트 정보를 포함하고 있어야 한다.

요구명세(RS : Requirements Specification) 아이템은 특정한 요구를 기술하게 되며, 최상위층의 구조는 기본적인 요구를 담고 있고 상세 요구는 하위 층에 담고 있다. RS아이템은 자연어로 기술하여 이해하기 쉽도록 해야 한다. 또한 각각의 요구명세는 정형화된 문서로 작성해야 한다. 만약 각기 다른 규칙을 가지는 요구명세를 사용하게 되면, 형상관리에 있어서 오류를 범하거나 예외발생이 생길 수 있다.

3.1 변경요청

변경 요청은 기존의 시스템의 업그레이드 혹은 컴포넌트 모듈간의 조합으로 새로운 시스템을 구축할 때 이루어지는 작업이다. 새로운 시스템 버전을 만들기 위해

서는 기존 시스템 버전에서 변경할 파일을 선택하여 만든다. 변경할 파일은 직접 선택되지 않고 변경요청(CR: Change Request)시 새로운 시스템 버전을 만들기 이전에 선택된다. 이 방식의 이점은 새로운 소프트웨어 버전과 논리적인 변화사이의 맵핑이 계획된 논리적인 교환과 직접적으로 이루어진다[3, 4].

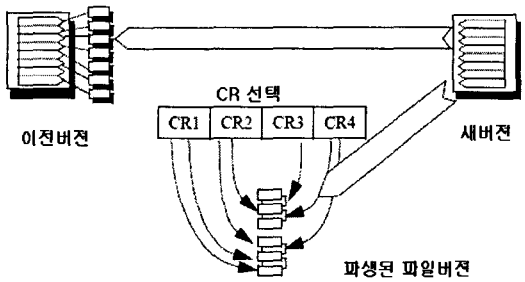

그림 1. CR을 선택하여 나온 새로운 버전

[그림 1]은 CR을 선택하여 나온 새로운 버전의 파일을 보여주고 있다. 먼저 이전버전의 파일에서 변경되는 부분을 선택하고, 이때 단수가 아닌 복수의 파일의 선택이 가능하게 된다. 변경요청을 받아 정상적으로 수립이 될 때 새로운 버전의 컴포넌트가 생성되게 된다. 생성되는 새로운 컴포넌트는 버전관리가 함께 수행되어져야 하는데 이때 버전관리는 컴포넌트 이름으로 할 수 있으며, 이미 동일한 컴포넌트 이름이 존재하면 생성되는 파일이름을 변경하여 생성시켜 버전관리를 수행케 한다. 만일 이러한 버전관리가 이루어지지 않고 변경시킨 컴포넌트를 접목시킬 경우 원치 않는 결과를 초래하거나 시스템의 다운을 초래할 수 있다.

3.2 시스템버전 일관성

CR(Change Request)은 파일버전과 일치하게 만들어야 하며, 다른 버전을 제외한 같은 파일에 언급될 수 있다. 이것은 몇몇 변경 요구로 인하여 파일이 변경될 때 일어나며, 이 경우 작업 공간은 1개의 파일에 복수 버전을 포함할 수 있으며, 사용되어지는 컴포넌트는 마지막으로 선택된 버전을 사용한다. 이때 최신 버전이 다른 CR에 몇몇 변경요소를 포함할 수 있으며, 코드가 변경된 최신버전의 파일을 사용하면, 새로운 시스템 버전도 마찬가지로 변경된다. 만약 이것이 다른 CR의 유일한 부분일 때, 예를 들어 수정될 파일과 다른 파일의 CR이 같게 된다면 일치하지 않는 결합이 발생될 수 있다.

새로운 버전을 지향하는 형상관리 에서는 다른 방법으로 이 문제를 처리한다. 어떤 것은 그 문제를 무시하지만 또 다른 것은 일치하지 않는 시스템에 경고를 발생시킨다.

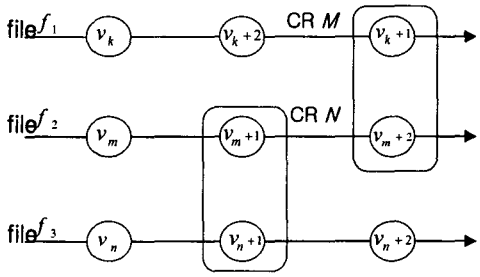

그림 2. CR 연결

[그림 2]는 CR 연결을 보여주고 있다. CR A를 파일버전으로 설정하면 $A = \{f_1 \cdot v_1, f_2 \cdot v_2, \ldots\ldots f_k \cdot v_k\}$ 는 v_i 로 표시하고 f_i는 파일버전을 표시하고 CR A에 포함시킨다. CR은 두 개 M과 N 이 있다고 가정한다. 만약 파일 f_i가 있을 때 파일버전은 $f_i \cdot v_m \in M$ 과 $f_i \cdot v_n \in N$ 으로 나타낸다. 다음으로 CR N 의 연결은 CR M if $n \le m$ 으로 나타낸다. CR M이 선택되면, CR N은 CR M에게 연결된다. 파일버전은 $f_1 \cdot v_{k+2}, f_2 \cdot v_{m+2}$ 그리고 $f_3 \cdot v_{n+1}$ 으로 새로운 시스템 버전에 통합된다. 만약 CR A가 CR B와 연결되고 CR B가 CR C와 연결된다면 CR A는 CR C와 연결된다. 이것은 연결된 CR들의 선택이 재귀 프로세스(반복처리)인 것을 의미한다.

3.3 명세서 변경

명세서 변경은 CR(Change Request)을 통하여 이루어지고 컴포넌트 모듈간의 합성 이전에 명세서를 통하여 명세 정보를 확인하고 이를 기반으로 하여 우선적으

로 합성 가능여부를 우선적으로 판별할 수 있으며, 합성 통합되어 나오는 명세서를 기반으로 실질적인 컴포넌트 합성을 수행한다[5].

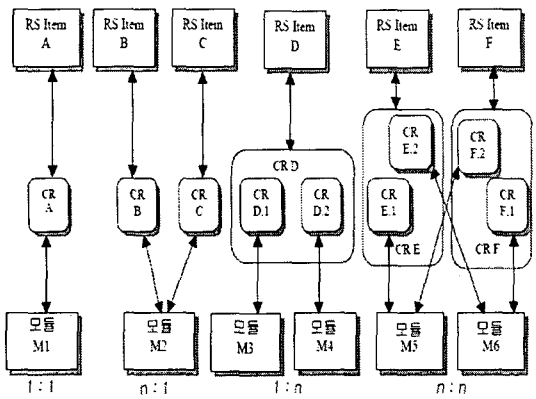

그림 3. 요구명세 아이템과 시스템모듈 사이의 변경요청 관계

[그림 3]의 목적은 구현 프로세스 동안에 필요한 요구사항을 시각적으로 표시되게 함으로써 개발자들이 인식하기 편하게 하는 것을 목적으로 두고 있다. 컴포넌트 모듈의 생성은 RS 아이템에 CR을 접목시켜 새로운 모듈을 생성하는데 1:1, n:1, 1:n, n:n의 연관을 가지고 있고 그에 따른 여러 가지 모듈이 생성된다. CR처리는 CR의 체크인/아웃을 시행하여 개발자는 RS항목에 접근할 수 있는데, 형상관리는 이런 실행 가능한 수단을 제공해야 한다. 또한 특정 RS 아이템에서의 CR은 유사한 다른 아이템(소스코드, 문서)으로 취급할 수 있는데 이렇게 하면 CR에 관련된 항목의 모든 버전들의 접근 가능성이 용이하게 된다.

IV. 컴포넌트 명세서 조합

컴포넌트 명세서에는 컴포넌트와 관련된 정보를 가지고 있기 때문에, 형상관리 시스템에서 중요한 요인이 된다. 명세서 조합의 과정은 이미 XML 형식으로 만들어진 컴포넌트 명세서를 DOM을 이용한 파싱으로 트리구조를 만들고 이 구조를 이용하여 각기 다른 컴포넌트 명세서의 조합을 하였으며, 조합과정에서 컴포넌트 버전관리를 수행하였다. 제안한 시뮬레이션 프로그램의 구현시스템 환경은 Intel Pentium4 프로세서 시스템과 마이크로소프트 윈도우XP SP1 시스템에서 수행되어 졌으며 사용된 프로그래밍 언어는 자바 언어를 사용하였고 버전은 j2sdk1.4.0_02 버전을 사용하였다.

4.1 컴포넌트명세구조

컴포넌트 명세서는 XML을 이용하였고 태그와 내용으로 이루어진 구성을 가지고 있다. 때문에 시작 구문이 있으면 종료 구문이 항시 존재해야 한다. <A>라는 요소가 시작되면 </A>로 요소의 끝을 나타내야 한다.

표 1. XML 명세서 구조

컴포넌트	〈Components〉 〈/Components〉
컴포넌트이름	〈C_name〉〈/C_name〉
오퍼레이션	〈Operation〉〈/Operation〉
인터페이스	〈interfaces_name〉〈/interfaces_name〉
인터페이스타입	〈interfaces_type〉〈interfaces_type〉
파라메타	〈parameter〉〈/parameter〉

[표 1]은 명세서 구조를 보여주고 있다. 루트 요소인 컴포넌트는 <component></component>로 요소를 지정하였고 상세 정보인 컴포넌트이름은 <c_name></c_name>로 표현하였다. 또한 컴포넌트에 관한 오퍼레이션은 <Operation></Operation>으로 선언 하였고 인터페이스는 <interface></interface>로 인터페이스타입은 <interface_type></interface_type> 그리고 파라메타는 <parameter></parameter>로 각각 태그(Tag)를 지정 하였다. 태그는 명세 정보를 동시에 수반하기 때문에 최대한 자연어로 기술하여 태그 이름으로 어떠한 정보를 담고 있는지를 나타내어 준다. 태그 안에는 각각의 요소가 선언된다.

[그림 4]는 두 컴포넌트 명세서의 합성 후 생성되는 새로운 컴포넌트 명세서로 합성된 컴포넌트 정보를 포함하고 있으며, XML 문서로서 웹브라우저에서의 뷰(view)를 보여주고 있다. 컴포넌트를 조합하여 새로 만들어진 컴포넌트 명세서에서는 조합 후 새로 만들어지

는 과정에서 자동적으로 라인(line) 단위의 공백이 제거되어 정렬된 후 생성된다. 이때 저장되는 정보는 이전에 정의한 태그를 토대로 정보가 저장되며, 컴포넌트 이름인 <c_name> 안에 조합된 컴포넌트 모듈의 이름을 담게 된다. 또한 이 명세는 각각의 명세 합성을 나타내고 구성이 XML 로 이루어지기 때문에 XML 규칙을 준수해야만 한다.

각각의 컴포넌트 명세에는 <components>태그로 시작하고 </components> 태그로 종결되어 지는데, 새로운 컴포넌트 명세 작성에 있어서 이 <components> 태그는 한번만 존재 해야만 한다. 단순히 파일입출력의 라인바이 라인의 읽기, 쓰기를 할 경우 <components> 태그가 다수 선언될 수 있고 실질적으로 라인수가 다른 명세서 조합에 다수 선언되게 되고 이 루트 요소뿐만 아니라 각각 자식요소도 마찬가지로 라인이 맞지 않을 경우 잘못된 태그 이름이 들어가게 되기 때문에 쓰레기 값이 들어가게 된다. 이렇게 되면 XML 규칙을 벗어나기 때문에 루트 요소는 한번만 선언되어져야 한다.

```
<?xml version="1.0" encoding="EUC-KR"?>
<components>
  <c_name>A01</c_name>
     <operation>a.dll</operation>
     <operation>b.dll</operation>
     <operation>c.dll</operation>

<interfaces_name>provider</interfaces_name
>
  <interfaces_type>String</interfaces_type>
  <parameter>Sum, subtraction</parameter>
<c_name>B01</c_name>
    <operation>a.dll</operation>
    <operation>b.dll</operation>

<interfaces_name>provider</interfaces_name
>
  <interfaces_type>String</interfaces_type>
  <parameter>division, multiply</parameter>
</components>
```

그림 4. 합성된 XML 명세서

4.2 컴포넌트명세 요소 추출

XML을 이용하면 텍스트 파일입출력 보다 효율적인 요소 검출이 가능하게 된다. 이때 추출할 요소를 선언해야 하는데, 구현된 프로그램에서는 요소 태그네임을 이용하여 추출하였다. 태그 이름을 사용하게 되면 동일한 태그 이름으로 된 요소의 값을 효율적으로 추출할 수 있고 복수의 동일한 태그에 복수 요소가 존재할 경우 효율성이 증가하게 된다. [그림 5]는 XML로 이루어진 명세서에서 요소를 추출하는 과정을 슈도코드를 통하여 보여주고 있다. 컴포넌트 이름을 받아오기 위해서 getElementsByTagName() 함수를 사용하여 노드가 아닌 태그의 이름으로 컴포넌트이름을 가져오고, 동일한 태그 값의 길이만큼 작업을 반복하며 요소 정보를 수집한다. 이때 하나의 노드는 하나의 아이템으로 선언하고, 첫 번째 Child 값부터 동일한 값을 가지는 요소가 더 이상 없을 때까지 반복하여 수행하여 값을 찾고, 동일한 태그의 존재가 종료될 때까지 문서를 파싱 후 기억하게 된다.

①에서 NodeList Lst_1 은 노드 리스트를 뜻하며, 여기서의 Lst_1....Lst_n 은 컴포넌트 명세서를 의미한다. 예제 슈도코드 에서는 명세 1번을 보여주고 있으며, 명세 2...n도 같은 구조를 가지게 된다.

②는 요소 추출을 담당하고 있다. 명세 아이템은 item으로 선언하고 이 아이템은 하나가 아닌 다수의 아이템이 존재하는 것이 일반적 이므로 루프를 돌면서 요소 아이템 개수만큼 반복하여 추출하고 이 정보를 가져오게 된다. 이때 노드는 첫 번째 노드를 찾아 요소를 가져온 다음 자식노드를 순회하면서 요소를 가져온다.

```
③  Text txt = Document out and
   createTextNode(ch1.getNodeValue();
        elm1 for appendChild(txt);
        root for appendChild(elm1);
```

③번 구문에서는 명세서 문서 출력을 담당하고 추출된 요소의 정보를 elm1 에 담고 있으며 이를 루트요소로 지정한다. 이러한 DOM을 이용한 요소 추출을 하게

되면, 필요 없는 정보는 제거하고 필요한 요소만을 사용할 수 있게 되고 일반적인 파일 입출력 방식에서의 문제점을 해결할 수 있는 장점을 가지게 된다.

```
① NodeList Lst_1 = Document
   getElementsByTagName("name");
② For( i=0; To Element Length Value();
 i Increment); {
        Node n1 = lst_1.item(i);
        for(Node ch1 = n1.getFirstChild();
        ch1 ! to null;
        ch1 = ch1.getNextSibling())
{
Element elm1 = outDoc.createElement("name");
③ Text txt = Document out and
   createTextNode(ch1.getNodeValue();
elm1 for appendChild(txt);
root for appendChild(elm1);
break;
  }
}
```

그림 5. 명세서 요소추출

4.3 버전관리

버전관리는 형상관리 시스템에 있어서 상당히 중요한 부분을 차지하는 요소이다. 잘 연결된 컴포넌트의 조합이라도 버전관리가 잘못되어 원하지 않는 컴포넌트로 분류되면 의미 없기 때문이다. 또한 새로 생성된 컴포넌트 정보관리가 잘못되어 이전 시스템에 악영향을 미칠 때 최악의 경우 시스템이 정지되는 결과를 가져오게 되기 때문에 서로 다른 컴포넌트 조합에서 생성되는 새로운 컴포넌트는 이전버전과의 분류를 필요로 하게 된다. 이때 분류 방법 중 하나가 컴포넌트 이름으로 분류하는 방법이다. 즉 이전에 만들어진 컴포넌트들의 조합으로 새로운 컴포넌트 창출에 있어서, 이전 컴포넌트의 이름을 검사하고 컴포넌트 이름이 일치할 경우 이후버전의 이름으로 생성시키게 되면 시스템의 문제를 미연에 방지할 수 있고 개발자나 관리자로 하여금 컴포넌트 분류 및 검색에 있어서 보다 정확한 선택을 가능하게 한다.

```
If Check Specification == (Existence Specification){
Display File check Result="Check Specification" File
for-Existence
For( i is from 0 to 99; i increment)          {
Result Specification    = (New DecimalFormat(Result
Specification).format(i));
//Existence Specification = True Loop          If
NewSpecification(Result Specification).exists())
  {
// Compare To Next
      Continue;          }
          Break ;    }
Display New SpecificationName = +Result
Specification+Change Display Output file has been
creates : +Result Specification          }
BufferedWriter bufw = new BufferedWriter( new
SpecificationWriter(Result Specification));
Print SpecificationName = new Print(bufw);
```

그림 6. 버전관리

[그림 6]은 컴포넌트 명세서 버전관리를 슈도코드를 이용하여 보여주고 있다. 먼저 새로운 컴포넌트 명세서 버전으로의 출력에 앞서 컴포넌트 저장소에 동일한 컴포넌트가 존재하는지의 여부를 검사하게 된다. 이때 동일한 컴포넌트가 존재할 경우 그 내용을 Display 하고, 자동으로 출력되는 컴포넌트 파일명을 바꾸도록 하고 바뀐 컴포넌트 이름도 존재할 경우 다시 다음버전의 이름으로 검색하여 컴포넌트 파일명 변경을 유도하고, 동일한 파일명이 존재하지 않을 경우에는 변경작업 절차 없이 지정한 파일명으로 출력하게 된다. [그림 7]은 구현된 프로그램을 통한 컴포넌트 명세서 버전관리 모습을 보여주고 있다. 컴포넌

트 명세서 출력작업에 있어서 동일한 컴포넌트가 존재하게 되면, 새로운 파일명으로 자동으로 변경하여 출력하게 되고 만일 컴포넌트 명세 조합에 있어 A01 컴포넌트와 B01 컴포넌트 명세서 조합할 때 A01 이라는 컴포넌트를 재 호출하게 되면 A01 컴포넌트의 이름을 A01 이 아닌 A02와 같은 이름으로 새로운 명세파일을 생성하고, 이 변경된 명세파일을 사용하고 A01과 A02 명세 내용은 동일한 내용을 사용하게 된다. 이렇게 시스템 버전관리와 의존관리를 동시에 수행함으로써 시스템의 일치하지 않는 오류를 사전에 대비할 수 있는 효과가 있다.

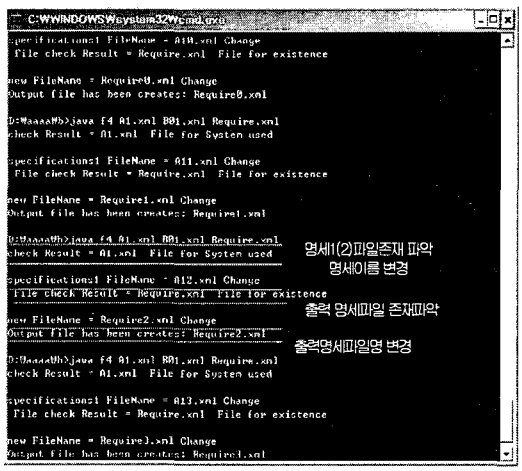

그림 7. 버전관리 화면

V. 결 론

본 논문은 형상관리 시스템에 있어 컴포넌트 조합 이전에 수행되어져야할 컴포넌트 명세서에 대해서 기술하였다. 컴포넌트 명세서에는 형상관리 시스템에서의 컴포넌트의 정보를 가지고 있으며, 이 정보에는 합쳐진 컴포넌트 정보와 사용되는 <operation> 등의 정보를 필수적으로 담고 있어야 한다. 컴포넌트 명세서 합성에 있어서 시뮬레이션 프로그램을 통해 정보의 합성 가능성을 확인하고. 컴포넌트 조합과 더불어 새로운 컴포넌트 명세서 생성시 이전 컴포넌트 정보를 수집하여 요구된 컴포넌트와 비교하여 기존에 존재하는 컴포넌트 명세서로 판정될 경우 파일명을 자동으로 변경하여 사용하게 된다. 또한 컴포넌트 버전관리를 행함으로 인해 사전에 시스템 안정화를 유도하였으며, 컴포넌트 명세이름을 이용하여 합성된 컴포넌트 이름을 모두 포함시켜 새로운 컴포넌트 명세를 만들어 컴포넌트 의존관리를 동시에 수행하게 하였다. 이 시뮬레이션을 통하여 명세서를 이용한 컴포넌트 버전관리가 가능함을 확인하였다. 향후 연구로는 컴포넌트 명세서 혹은 컴포넌트 조립에 있어 명세서를 통하여 자동으로 업그레이드 여부를 판단하고 조립에 선택된 컴포넌트가 조립 가능하지 않은 경우 가장 적절한 컴포넌트를 자동으로 선정해 주는 연구가 필요하다.

참 고 문 헌

[1] 최성, 윤태권, "CBD(Component Based Development)현황과 전망", 정보처리학회지, 제10권 제3호, pp.16-17, 2003.

[2] I. Crnkovic, "Experience with Change-oriented SCM Tools," In Proceedings Software Configuration Management SCM-7 Boston, MA, USA, pp.2-6, May, 1997.

[3] I. Crnkovic, "A Change Process Model in an SCM Tool," In Euromicro 98, proceedings of the 25th EUROMICRO conference, Sweden, Sep, IEEE, Computer society, p.4 1998.

[4] I. Crnkovic, "Processing Requirements by Software Configuration Management," 25th Euromicro Conference (EUROMICRO '99), Vol.2, pp.2-5, Sep.8-10, 1999.

[5] 한정수, 정대성, "컴포넌트 형상관리를 위한 프로세스 요구사항", 추계학술발표 논문집, Vol.2, No.1, 2003.

저 자 소 개

정 대 성 (Dae-Sung Jung) 준회원

- 2003년 2월 : 천안대학교 정보통신학부(공학사)
- 2005년 8월 : 천안대학교 정보기술대학원(공학석사)

<관심분야> : CBD, 형상관리, 소프트웨어 아키텍처

한 정 수(Jung-Soo Han) 종신회원

- 1990년 : 경희대학교 전자계산공학과(공학사)
- 1992년 : 경희대학교 전자계산공학과(공학석사)
- 2000년 : 경희대학교 전자계산공학과(공학박사)
- 2001년~현재 : 천안대학교 정보통신학부 조교수

<관심분야> : 소프트웨어 공학, CBD, OOP, S/W 개발 방법론