

## UML를 이용한 컴포넌트 버전 제어 시스템 설계

김 홍 진\* 오 상 엽\*\* 김 영 선\*\*\*

## A Design Version Control System of Component using UML

Hong-jin Kim\* Sang-yeob Oh\*\* Young-sun Kim\*\*\*

### 요 약

소프트웨어의 개발은 생산성을 향상시키기 위해 미리 만들어진 소프트웨어의 컴포넌트를 이용한다. 컴포넌트의 재사용은 소프트웨어 개발비용을 절감하고 개발기간을 단축시킬 수 있다. 소프트웨어 개발에서 컴포넌트를 재사용함으로써 소프트웨어 설계에서 구현까지의 위험요소를 최소화시킬 수 있는 것이다. 소프트웨어를 구현하기 위한 기술로 UML을 도입하여 컴포넌트에 대한 명세를 버전으로 관리하여 재사용의 효율성을 높일 수 있는 방법을 버전 제어 시스템을 통해서 제시하고자 한다. 본 연구의 특징은 UML을 이용하여 컴포넌트의 버전 제어를 통해 소프트웨어 개발시 효율적인 재사용이 될 수 있도록 하는데 그 목적이 있다.

### Abstract

The Development of software puts component of software made in advance to use to improve productivity. Reuse of component can cut down on the development costs of software and reduce the development period. By reusing component needed for software development, it can minimize risk factors from the software design to implementation. We are going to proposal the method which can promote efficiency of reuse by introducing with skill to implement the software and managing the specification using UML with version through the version management systems.

The object of this study is that when the software is developed, we make software used effectively through the version management of component using UML.

\* 경원전문대학 컴퓨터정보과 교수

\*\* 경원전문대학 교양과 부교수

\*\*\* 대림대학 경영정보계열 전임강사

접수일자 : 2002.12.20

심사완료 : 2003. 2.23

## I. 서론

소프트웨어의 설계시 생산성, 품질, 효율성을 높이기 위하여 소프트웨어의 컴포넌트 재사용은 소프트웨어 개발 비용을 최대한 절감시킬 수 있다. 소프트웨어 개발에 정형화된 컴포넌트를 버전 관리하여 재사용함으로써 소프트웨어의 설계에서 구현까지의 위험요소를 최소화 시킬 수 있는 것이다. 컴포넌트는 시스템을 보다 유연하게 하고 개발 과정에서의 반복적인 작업을 줄이고 소프트웨어의 시스템 유지보수 비용을 절약할 수 있게 한다.

최근에 소프트웨어 개발에 부품화와 조립의 특성을 갖는 컴포넌트 기술의 도입으로 컴포넌트를 기계 부품처럼 조립하고 기능이 개선된 새로운 부품을 재조립하는 재사용 방식을 통한 소프트웨어 개발은 생산성 향상과 효율의 극대화를 가져온다[1].

소프트웨어 개발에 객체지향 분석과 설계 기술의 도입에 사용되는 언어인 UML을 이용한 컴포넌트 기술은 소프트웨어 구조에서 필요한 표준 부품에 대한 명세 및 구현에 적용될 수 있다. 컴포넌트는 독립적인 활용의 단위로 각각의 생명주기와 각각의 버전을 갖는다. 각각의 생명 주기를 갖는 컴포넌트는 서로 다른 버전 및 구성 형태를 가질 수 있다. 효율적인 컴포넌트를 사용하기 위해서는 각 컴포넌트의 버전을 관리해야 한다[2].

소프트웨어를 구현하기 위한 기술로 객체지향 분석과 설계하는 UML을 활용한 컴포넌트를 버전으로 관리하여 컴포넌트에 대한 재사용의 효율성을 높일 수 있는 방법을 버전 관리 시스템을 통해서 제시하고자 한다.

본 연구의 특징은 UML을 이용한 컴포넌트의 버전관리를 통해 소프트웨어 개발시 컴포넌트의 효율적인 재사용이 될 수 있도록 하는데 그 목적이 있다.

## II. 관련 연구

### 1. 객체 지향 언어 UML

현재는 소프트웨어에 대한 가치가 증가되어 소프트웨어 개발의 자동화, 소프트웨어의 개발 기간과 비용의 절감 및 소프트웨어의 질적인 향상이 요구되는데 특히, 시스템 개발의 범위와 규모가 커짐에 따른 시스템의 복잡성을 처리할 필요성을 느끼게 되었다. 시스템개발에 물리적인 시스템의 분산과 동시성, 반복성, 보안 및 시스템에 부하에 대한 균등화과 같은 문제 처리가 필요하게 되었다. 이러한 모든 필요성에 의해 소프트웨어 개발에 객체지향 분석과 설계 기술의 도입에 사용되는 언어가 UML이다. 모든 프로그래밍언어는 모델링 언어로 통합될 것인데 통합모델링언어(UML)와 같은 모델링 언어는 소프트웨어 개발을 위한 객체지향 분석과 설계의 결과를 표현하기 위한 그래픽언어[3]로 역할만 하고 있지만, 향후 실행언어의 기능까지 포함하는 자바나 C#과 같은 실행언어를 대체할 수 있을 것이다.

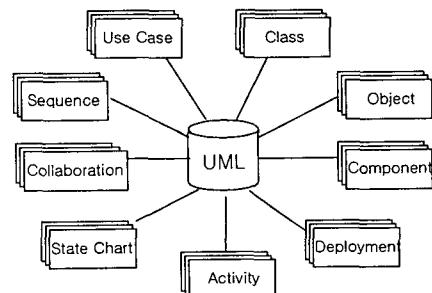


그림 1. UML 구성 다이어그램

- ① 유스케이스는 시스템 요구사항을 효과적으로 이용하여 신뢰성 있는 설계 모델을 개발하는데 필요한 개념과 기법을 제시하는 시스템의 요구 사양이다.
- ② 클래스 다이어그램은 시스템 내부에 존재하는 클래스들을 선별하여 나타내고 각 클래스들의 속성과 행위를 기입한다.
- ③ 객체 다이어그램은 클래스 다이어그램으로 표현한

- 클래스가 실제 객체로 실체화되는 관계를 나타낸다.
- ④ 시퀀스 다이어그램은 시스템이 실행시 생성되고 소멸되는 객체를 표기하고 객체들 사이에 주고 받는 메시지의 순서를 나타낸다.
  - ⑤ 콜레버레이션 다이어그램은 시퀀스 다이어그램과 함께 메시지의 흐름을 나타내는 시스템의 동적인 면을 나타내는 대표적인 다이어그램으로 객체와 객체들 사이의 관계를 나타낸다.
  - ⑥ 상태 다이어그램은 한 객체의 상태 변화를 다이어그램으로 나타낸 것으로 시스템의 실행시 객체의 상태는 메시지를 주고받음으로써 객체의 상태의 전이를 표현하는데 사용된다.
  - ⑦ 액티비티 다이어그램은 시스템 내부에 존재하는 여러 가지 행위 및 각 행위의 분기, 분기되는 조건 등을 모두 포함한 흐름을 나타내는 것을 말한다.
  - ⑧ 디플로이먼트 다이어그램은 시스템 동작시의 환경을 표현한 것으로 실제 하드웨어적인 배치와 연결 상태를 나타낸다.
  - ⑨ 컴포넌트 다이어그램은 소프트웨어의 물리적 단위(Exe, dll 등 기타 library)의 구성과 연결상태를 나타내게 된다.

## 2. 컴포넌트

컴포넌트는 독립적인 배포가 가능하며 잘 정의된 소프트웨어 구조에서 특정하게 정의된 기능을 수행하는 대체 가능한 소프트웨어 부품이다. 컴포넌트는 유스케이스(Use Case)가 정의되어 있고, 실행 인터페이스를 내부에 포함하고 있는 소프트웨어 시스템의 개발에 사용되는 부품으로 독립적 개발과 배포의 단위로 동일한 인터페이스를 갖는 다른 컴포넌트와 대체가 가능하며 다른 컴포넌트와 상호 동작하도록 개발된다[4]. 컴포넌트는 상호 작용하는 클래스의 집합으로 특정하고 명확한 기능과 구조를 갖는다. 컴포넌트는 동일한 인터페이스를 가지고 다른 컴포넌트와 대체될 수도 있다. 컴포넌트는 소프트웨어 개발의 유연성을 주고 반복적인 개발을 줄 일 수 있어 유지보수 비용도 절감된다[5].

소프트웨어 컴포넌트는 재사용 가능한 소프트웨어 서비스를 독립적으로 전달 할 수 있도록 패키지화된 것이다. 컴포넌트는 다른 컴포넌트와 상호동작을 통해 특정한 소프트웨어 구조에 필요한 기능을 제공한다. 컴포넌트의 상호동작은 명시적으로 정의되는 인터페이스를 이용한다[6]

인터페이스는 컴포넌트의 서비스를 제공하는 오퍼레이션의 집합에 대한 이름 부여와 서비스의 호출 기능을 수행한다.

컴포넌트의 구성요소들은 컴포넌트 개념 다이어그램으로 표현할 수 있다.

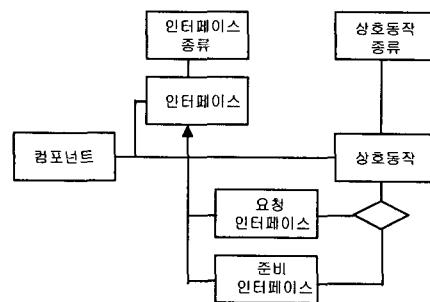


그림 2. 컴포넌트 개념 다이어그램

컴포넌트의 가장 중요한 것은 컴포넌트 자신의 기능과 그 기능을 이용하는 방법에 대한 정의를 갖는 것이다[10]. 컴포넌트에 대한 정확한 정의는 컴포넌트의 사용자가 구축하려는 어플리케이션의 기능에 대한 중점적인 관심을 가질 수 있도록 해준다. 컴포넌트가 제공하는 서비스는 프로그래머나 설계자에 의해서 제공되며 그 형태는 코드 혹은 데이터이다[7]. 이러한 코드와 데이터는 컴포넌트의 명세를 완벽하게 만족시켜야 한다.

## 3. 버전 제어

소프트웨어의 개발 및 유지보수에 도움을 주기 위해 새로운 방법론 및 재사용이 계속해서 개발[8]되고 있는데, 그 중 하나가 버전제어이다. 버전 제어는 오류 수정, 사용자 요구사항의 변경, 소프트웨어 기능 향상 등의 이유로 시간이 지남에 따라 변화하는 소프트웨어 구성요소 및 소프트웨어 형상을 체계적으로 관리를 하는 것을 말한다.

프로젝트 개발 및 유지보수되는 동안 구성요소는 변경되며, 이러한 변경(changes)의 집합을 비즈니스모델을 통해 컴포넌트가 수행되는 동안 구성요소를 연속된 다음 버전으로 변환시키는 이력과정(history step)을 버전 제어라 한다.

일반적인 소프트웨어 구성요소의 버전도 수정버전과 변형버전으로 나눌 수 있다. 수정 버전은 형상의 수정과 비슷한 개념이나 변형버전은 이와는 달리 두 가지 정의가 가능하다. 그 하나는 소프트웨어 객체의 추상(abstraction)에 있어서 서로 구별할 수 없는 두 객체 사이에 존재하는 것과 이전의 버전(old version)에 대한 변경이 있을 때 생성되는

버전을 말한다. 실제에 있어서 변형 버전은 이 두 가지 개념을 모두 만족할 수도 있고, 하나만 만족할 수도 있다[9].

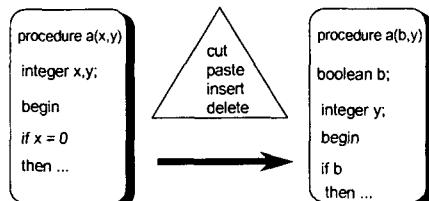


그림 3. 이력 과정

(그림 3)은 한 버전을 새로운 버전으로 변환하는 구성 요소들의 이력과정을 위한 delta의 형태를 제안하기 위해서 삼각형 박스로 표시한다.

### III. 컴포넌트 제어 시스템 설계

현재의 소프트웨어의 개발 방법은 컴포넌트를 어떻게 조립하고 관리하느냐에 따라 효율적인 소프트웨어 시스템을 구축할 수 있느냐를 판가름하게 된다. 소프트웨어 시스템 구축에 사용되는 컴포넌트 자체를 개발, 활용, 유지 보수를 위해 미리 만들어진 컴포넌트를 재사용 한다는 것은 소프트웨어 개발 기간과 비용을 절감할 수 있어 생산성을 향상시킬 수 있다. 검증된 컴포넌트를 사용하여 위험 요소를 최소화시키고, 컴포넌트 사용시 일관성을 확보할 수 있어 유지보수 비용도 절감할 수 있다. 이와 같은 장점을 얻기 위해 컴포넌트의 관리 모델이 필요한 것이다.

#### 1. 컴포넌트 모델의 연산 관계

본 논문에서 제시하는 모델은 소프트웨어 객체를 저장하는 COMPONENT 클래스는 세 종류의 부클래스들을 갖는다.

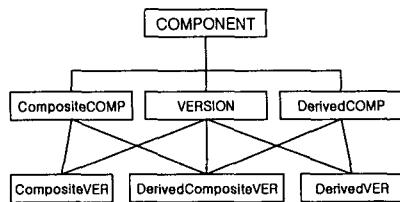


그림 4. COMPONENT 클래스 계층 구조

COMPONENT 클래스 계층 구조에서 첫 번째 부클래스 CompositeCOMP은 다른 소프트웨어 객체들로 구성된 소프트웨어 집합 객체인 인스턴스를 갖는다. 두 번째 부클래스 VERSION은 버전이 주체가 되는 소프트웨어 객체의 추상적인 정의를 나타낸 것이다.

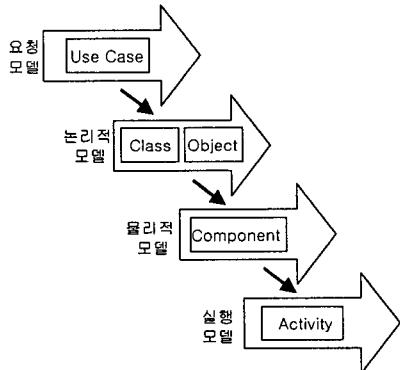
컴포넌트는 다른 컴포넌트와의 상호 협력을 통해서 시스템을 구성하게 되는데 이때 컴포넌트의 기능은 다른 컴포넌트 혹은 시스템에 영향을 미친다. 재사용 연산은 기본적인 재사용 컴포넌트를 제공하고 컴포넌트의 새로운 구성 요소의 추가하는 확장과 새로운 의존관계를 유지시켜 독립적인 구성요소를 형성하게 된다.

표 1. 재사용 컴포넌트의 연산

재사용연산	의 미
확장	컴포넌트의 기능의 확장
수정	컴포넌트의 새로운 의존관계 추가 컴포넌트 간의 의존관계 삭제
삭제	컴포넌트의 제거

#### 2. UML를 이용한 컴포넌트 설계

비즈니스 모델에서 컴포넌트의 기반의 소프트웨어가 독립적인 비즈니스 컴포넌트를 이용해서 어플리케이션을 개발하기 위한 조립 기술 및 조립에 사용된 부품으로서의 컴포넌트를 관리가 필요하다. 본 논문에서는 컴포넌트가 시스템 구조에 많은 영향을 미치기 때문에 Catalysis[7]을 기반으로 하여 개발 절차를 제시하고자 한다.



컴포넌트 관리 모델은 (그림 6) 와 같이 X, Y, Z 세 개의 축으로 표시되는 3차원 공간으로 나타낼 수 있다.

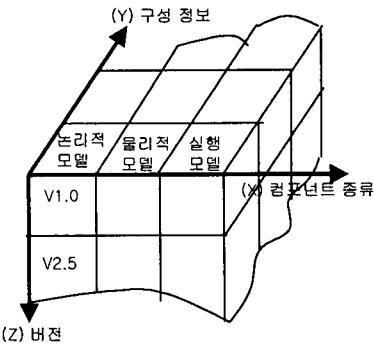


그림 6. 컴포넌트 제어 모델 구조

#### (1) 요청 모델

사용자 요구사항을 유스케이스 다이어그램을 통해서 기술적으로 적용하여 사용자의 개발자간의 오해를 최소화 하여 시스템의 구조적 이해도 반영한다.

#### (2) 논리적 모델

클래스 다이어그램을 통해서 시스템 구조를 기술하여 일관되게 반영하고, 시스템 내부에 존재하는 클래스들을 선별하여 나타낸다.

#### (3) 물리적 모델

컴포넌트 다이어그램은 소프트웨어의 물리적 구성요소로 서로 밀접하게 연관된 클래스와 객체를 묶어서 실제적인 연결상태를 나타내는 컴포넌트를 구성한다.

#### (4) 실행 모델

시스템에 적용되는 영역에서 컴포넌트들을 비즈니스 행위 컴포넌트로 정확하게 재사용 가능한 연산들을 지원하여 시스템 내부에 존재하는 여러 가지 행위 및 각 행위의 분기, 분기되는 조건 등을 모두 포함한 흐름을 나타낸다.

### 3. 컴포넌트 버전 제어 모델 설계

비즈니스 모델의 컴포넌트 관리에서 발생할 수 있는 컴포넌트 관리 모델을 재사용에 대한 버전 제어로 전체 시스템의 형상관리를 지원한다.

컴포넌트는 독립적인 요소로 수행이 단위가 될 수도 있고 조립과 합성을 통해서 소프트웨어 시스템을 구축할 수도 있다. 이때 합성, 조립된 컴포넌트는 기능의 개선이나 성능에 대한 요구 사항에 의해서 변경되며 이것은 컴포넌트의 변경으로 새로운 버전이 생성하게 된다. 컴포넌트를 효율적으로 사용하기 위해서는 각 컴포넌트의 종류 별 버전 및 구성 정보를 체계적으로 관리할 필요가 있다.

## IV. 컴포넌트 버전 제어 알고리즘

컴포넌트에 대한 버전 제어는 컴포넌트의 계속적인 변경은 소프트웨어에 적용된다. 컴포넌트는 이전 버전의 컴포넌트와 비교되기 때문에 반드시 버전이 부여되어 컴포넌트에 대한 유일한 식별자를 가지고 변경된 컴포넌트에 대한 새로운 버전을 부여한다.

### 1. 컴포넌트 버전 제어의 구성 요소

컴포넌트는 여러 개의 버전을 가지며 각 버전별로 여러 개의 구성 정보를 갖는다.

#### (1) 컴포넌트

- 컴포넌트 ID : 컴포넌트의 식별자
- 컴포넌트 이름 : 컴포넌트의 이름
- 컴포넌트 종류 : 컴포넌트의 분류

- 컴포넌트 소유자 : 컴포넌트의 소유자
- 컴포넌트 변경일자 : 컴포넌트의 가장 최근 변경된 일자

### (2) 구성 정보

- 정보 ID : 구성 정보의 식별자
- 정보 이름 : 구성 정보의 이름
- 확정 일자 : 컴포넌트가 확정된 일자

### (3) 버전

- 버전 ID : 버전의 식별자
- 버전 이름 : 버전의 이름
- 버전 내용 : 버전의 대한 정보
- 버전 변경일자 : 버전의 변경 일자

## 2. 컴포넌트 버전 제어의 모듈

컴포넌트 변경사항이 발생하였을 때는 이전 버전의 컴포넌트와 비교하여 반드시 컴포넌트에 대한 유일한 식별자를 가지고 변경된 컴포넌트에 대한 새로운 버전을 부여 한다.

### (1) 컴포넌트 생성 모듈

새로운 컴포넌트의 발생은 유일하게 식별할 수 있는 식별자로써의 컴포넌트 ID를 부여하고, 새로운 컴포넌트의 발생은 발생시점의 버전을 기록관리 한다. 이에 대한 알고리즘은 (그림 7)과 같다.

```
Begin
Get_obj_name(Component_Name, Exist)
If Exist Then Return
    Get_New_Table(Component)
    Read_Db_Class(Component_ID, Exist)
    If Exist then
        version = version + 1
    endif
    write(Component)
    write(Version)
    write(Configuration)
endif
end
```

그림 7. 컴포넌트 생성 모듈

### (2) 컴포넌트의 변경 모듈

기존 컴포넌트에 대한 변화으로 구조적, 기능적 변화가 있을 때 발생하고, 컴포넌트의 구성 정보의 변화가 있

을 때 생성된다. 컴포넌트에 대한 재사용을 위한 버전제어에 저장하는 알고리즘은 (그림8)과 같다.

```
Begin
Read_Class(Component_ID, Exist)
if Find then
    Modify(Component)
    Modify(Version)
    Modify(Information)
else
    write(Component)
    write(Version)
    write(Configuration)
endif
```

그림 8. 컴포넌트 변경 모듈

## VI. 결 론

소프트웨어 개발의 생산성 향상은 소스 코드의 재사용에서부터 시작하여 소프트웨어 모든 요소들을 재사용하는데 있다. 소프트웨어 개발 과정에서 개발비용이 많이 차지하는 분석과 설계에 대한 재사용에 대한 지원하기 위해서 표준화된 부품과 표준화된 개발 공정, 소프트웨어 실행 환경을 제공하는 모든 것을 컴포넌트 모델로 구성하여 재사용 함으로써 소프트웨어의 생산성 향상을 도모할 수 있다.

본 논문에서는 컴포넌트의 정형화시킨 모델로 구성하여 컴포넌트의 재사용에 대한 버전을 관리함으로써 컴포넌트를 사용하여 어플리케이션을 개발할 때 코드와 설계 및 분석 등을 재사용 함으로써 개발비용의 대한 절감과 소프트웨어에 대한 생산성 향상을 가져올 것으로 예측된다. 앞으로 향후 연구과제는 재사용에 대한 익숙하지 못한 개발자를 위해서 재사용 컴포넌트에 대한 자동화 도구를 개발하여 컴포넌트의 효율적인 버전관리가 이루어 질 수 있도록 실행 도구 개발에 대한 연구가 계속되어야 할 것이다.

## 참고문헌

- [1] David M. Weiss, Chi Tau Rober Lai., "Software Product-Line Engineering : A Family-Based Software Development Process", Addison Wesley, 1999.
- [2] Desmond F. D'Souza Alan C. Wills., "Objects, Components and Frameworks with UML", Addison Wesley, 1997.
- [3] Carmichael A. R, "Seeking A Unified Componet Model", SIGS Publications, 1997.
- [4] Clemens Szyperski, "Componet Software : Beyond Objet-Oriented Programming", Addison Wesley, 1998.
- [5] Peter Herzum, Oliver Sims., "Business Component Factory", Wiley Computer Publishing, 2000.
- [6] Keith Short., "Component Based Development and Object Modeling", Tech. report of Texas Instruments, 1997.
- [7] Desmond F. D'souza and Alan C. Wills., "Objects, Component, and Frameworks with UML: The Catalysis Approach", Addison Wesly, 1999.
- [8] E. A. Stohr and K. Youngbeom, "Software Reuse: Survey and Resarch Directions", Journal of Management Information Systems, pp. 33-37, Spring 1998.
- [9] 오상엽, "버전 제어에서 효율적인 형상 형성을 위한 혼합 검색 시스템의 모델링과 구현", 광운대학교 박사학위논문, 1999. 2.
- [10] 김강태, "재사용 계약 기반의 컴포넌트 관리 모델을 적용한 소프트웨어 아키텍쳐 구축 방법론", 중앙대학교 박사학위논문, 2000. 12.

## 저자 소개



김 흥 진

광운대학교 전자계산학과 이학석사 및 이학박사  
현재 한국 컴퓨터정보학회 회장  
현재 경원전문대학 컴퓨터정보과 교수



오 상 엽

광운대학교 대학원 전자계산학과 이학석사 및 이학박사  
현재 경원전문대학 교양과 부교수



김 영 선

광운대학교 전자계산학과 이학석사  
광운대학교 컴퓨터과학과 박사과정 수료  
현재 대림대학 경영정보계열 전임강사