재사용 비즈니스 모델을 이용한 컴포넌트 버전 관리 설계

김 영 선† · 오 상 엽†† · 장 덕 철†††

요 약

소프트웨어의 재사용은 소프트웨어의 생산성을 향상시키기 위해 미리 만들어진 소프트웨어의 컴포넌트를 이용하는 것이다. 전자상거래의 발달은 비즈니스 모델의 변화를 신속하게 변경하여 반영할 수 있는 소프트웨어를 요구하게 된다. 이런 전자상거래의 변화하는 환경에 신속히 대응하기 위해 재사용은 필수적인 해결책이다. 이러한 재사용 비즈니스 모델은 버전 관리에서 제공하지 못하는 단점이 있다. 이를 보완하여 본 논문에서는 재사용 비즈니스 모델에 대한 신속한 변경 관리를 도입하여 컴포넌트를 재사용함으로써 소프트웨어 개발비용을 절감하고 개발기간을 단축시킬 수 있도록 한다. 새로운 재사용 비즈니스 모델에 의해 컴포넌트를 재사용함으로써 소프트웨어 설계에서 구현까지의 위험요소를 최소화시킬 수 있는 장점을 가진다. 소프트웨어의 재사용을 구현하기 위한 기술로 컴포넌트를 도입하여 컴포넌트에 대한 구성요소를 버전으로 관리함으로써 재사용의 효율성을 높일 수 있는 방법을 버전 관리를 이용하여 제시하고자 한다.

Design of Component Version Management using Reuse Business Model

Young Sun Kim† · Sang Yeob Oh†† · Deog Chul Jang†††

ABSTRACT

The reuse of software is to use the components of software to be made beforehand to improve the productivity of it. The development of electronic commerce requires it which can be shown from changing the change of business models. Reuse is the necessary solution to cope with a rapid change in the electronic commercial transaction. These reuse business models have the defects that they are not offered from version management. This paper has the purpose that by doing the supplementation of defects not to be offered from the version management and by introducing the quick change management about reuse business models, reusing the components and saving a development and reducing a development of period. The reuse of components by the new reuse business models has the advantages to minimize the danger elements from the design to the finish. We use the version management and try to present the method to make the efficiency of reuse by introducing components and managing the elements about components to the version in the technique to accomplish the reuse of software.

키워드 : 버전(Version), 비즈니스 모델(Business Model), 재사용(Reuse), 컴포넌트(Component), 소프트웨어 개발(Software Development)

1. 서 론

표준 개발 프로세스와 표준 부품을 통해서 제품을 생산하는 하드웨어 생산 공정과 같이 소프트웨어에서도 특정 프로그램 군에 대한 표준적인 프로세스와 표준 부품을 통해 효율적인 프로그램 개발을 목표로 하고 있다[1]. 이러한 프로그램 개발 목표를 달성하기 위한 것이 컴포넌트이다.

그 동안 연구되어온 컴포넌트 기반 개발(Component-Based Development) 방법은 주로 컴포넌트를 개발하고 이를 사용하는 방법으로서 컴포넌트의 개발 및 사용을 지원하는 연구가 이루어졌다. 또한, 컴포넌트의 상호동작을 지원하는 메카니즘으로 미들웨어 기술에 대한 연구들로 대규모 분산 소프트웨어 시스템 구축에 이용되고 있지만, 재사용 관점에서 대규모 재사용과 밀접한 관련이 되는 컴포넌트의 개발 및 사용을 소프트웨어 구조의 컴포넌트에 대한 효율적인 재사용에 대한 연구가 필요하게 되었다. 소프트웨어 개발에 필요한 부품인 컴포넌트를 재사용할 수 있도록 정형화 시켜 소프트웨어의 설계에서 구현까지의 위험요소를 최소화 시킬 수 있는 것이다.

최근에 소프트웨어 개발에 부품화와 조립의 특성을 갖는 컴포넌트 기술의 도입으로 컴포넌트를 기계 부품처럼 조립하고 기능이 개선된 새로운 부품을 재조립하는 재사용 방식을 통해서 소프트웨어 개발 생산성을 높일 수 있는 것이다[2].

컴포넌트를 이용한 소프트웨어 시스템의 구축은 이미 검

※ 이 논문은 2000년도 광운대학교 교내 학술연구비 지원에 의해 연구되었음.
† 정 회 원 : 대림대학 경영정보계열 교수
†† 정 회 원 : 경원전문대학 교양과 교수
††† 정 회 원 : 광운대학교 컴퓨터과학과 교수
논문접수 : 2002년 2월 18일, 심사완료 : 2002년 8월 28일

증된 컴포넌트를 합성 및 조립을 의미한다. 각 컴포넌트는 독립적인 활용의 단위로 각각의 생명주기와 각각의 버전을 갖는다. 효율적인 컴포넌트를 사용하기 위해서는 각 컴포넌트의 버전을 관리해야 한다[3]. 컴포넌트는 소프트웨어 구조에서 특정하게 정의된 기능을 수행하는 대체 가능한 소프트웨어 부품으로 객체지향형 시스템의 개발에 사용되는 독립적 배포의 단위로 동일한 인터페이스를 갖는 다른 컴포넌트와 대체가 가능하며 다른 컴포넌트와 상호 동작하도록 개발된다[4]. 컴포넌트는 소프트웨어 개발의 유연성을 주고 반복적인 개발을 감소 시켜 유지보수 비용도 절감된다[5]. 컴포넌트에 대한 정확한 정의는 컴포넌트의 사용자가 구축하려는 어플리케이션의 기능에 대한 중점적인 관심을 가질 수 있도록 해준다[6].

기존의 컴포넌트 관리 모델의 논문에서는 재사용 컴포넌트를 관리하는데 있어 라이브러리의 구축과 라이브러리에서 적절한 재사용 컴포넌트를 검색하는데 필요한 검색 시스템의 구현으로 컴포넌트의 관리에 역점을 두고 연구[7]되었으나 본 논문에서는 컴포넌트 관리시 컴포넌트 이력 관리를 제공하지 않는 것을 버전과 접목시켜 컴포넌트를 효율적인 버전관리를 통해 비즈니스 모델의 재사용을 정형적으로 나타내고, 컴포넌트 버전 관리를 설계하여 생산성이 향상되어 재사용의 활용으로 신속한 컴포넌트 제공과 재사용 비용의 극대화를 가져올 수 있도록 하였다. 소프트웨어 개발의 생산성 및 품질의 효율을 높이기 위해 컴포넌트를 도입하여 컴포넌트의 상호 동작 정보와 개선 및 변경 정보에 대한 명세를 버전으로 관리함으로써 기존의 컴포넌트의 변경 상황과 속성 변경을 재사용할 때 효율성을 높일 수 있는 방법을 버전 관리를 통해서 제시하고자 한다. 본 연구의 특징은 컴포넌트의 버전관리를 통해 소프트웨어 개발시 컴포넌트의 효율적인 재사용이 될 수 있도록 하는데 그 목적이 있다.

2. 관련 연구

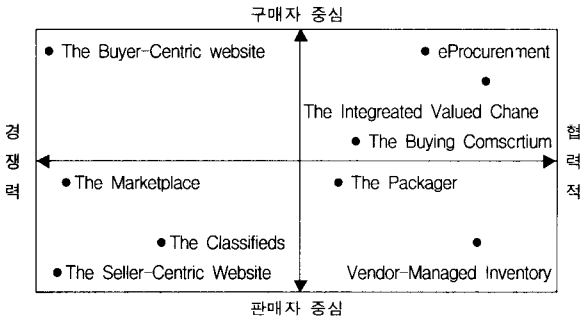
2.1 비즈니스 모델

비즈니스 모델은 근본적으로 기업이 어떻게 이윤을 추구하는지를 나타내는 것으로 단순한 사업모델부터 복잡한 사업모델까지 다양하다. 비즈니스 모델을 거래에 관여한 당사자들과 각각의 역할을 포함해 상품, 서비스와 정보의 흐름을 나타내는 구조와 거래에 참가하는 당사자들에게 주어지는 편익과 수입원에 대한 정확한 묘사가 표현되기도 한다[8]. 그래서, 비즈니스 모델은 사업을 시각적으로 보여주는 것으로 무엇을, 누구에게, 어떻게, 가치를 창출하고 전달하는 프로세스를 매핑하는 것이다.

비즈니스 모델은 새로운 사업의 아이디어로 거래를 어떻게 할 것인가를 나타내는 현물시장의 거래방법이나 경제법칙 자체 및 상거래방법 등을 나타내고, 기업이 어떻게 가치(Value)를 창출해서 소비자에게 전달하고 이를 기업의 이윤으로 만들어 내는 가하는 문제에 대한 모델이다.

비즈니스 모델은 가치사슬(Value Chain)의 분해(deconstruction)와 재결합(re-construction)을 통해 체계적인 접근으로 가치사슬상의 요소를 확인하고, 거래참여자의 상호활동(interaction) 패턴을 분석하여 가치사슬을 따라 정보를 통합하는 방법을 밝혀냄으로써 비즈니스 모델을 구분한다.

가치사슬 분해는 기본적활동과 지원활동에 속하는 것으로 가치사슬상의 요소를 찾는 것으로 상호활동에 관한 관계를 의미하고, 가치사슬 재결합은 가치사슬상의 여러 단계에 따라 정보처리과정을 통합을 통해 가치사슬 요소의 조합(combination of value chain elements)을 이룬다. 비즈니스 모델의 기술은 새로운 모델의 정의에서 기술개발에 대한 제안이 나오게 된다. 그리하여, 소비자의 욕구를 만족시키는 비즈니스 모델을 성공시키기 위해서는 우선적으로 사업분야를 선정하고 선정된 사업분야의 사업성 평가를 걸쳐 경쟁업체의 장단점 분석하고, 시장상황 분석하여 법적·정치적 환경을 고려하여 비즈니스 모델 제시한다. (그림 1)은 전자상거래에 대한 비즈니스 모델을 나타낸다.



(그림 1) 전자상거래 비즈니스 모델 (출처 : Andersen Consulting)

비즈니스 모델 분류를 참여 구분 모델, 수입 구분 모델, 상호 작용 모델, 사업 방식 모델, 이식 형태 모델로 나누고, 각각의 비즈니스 모델 분류에 따라 비즈니스 유형이 <표 1>과 같이 나눈다[13].

〈표 1〉 비즈니스 모델 분류

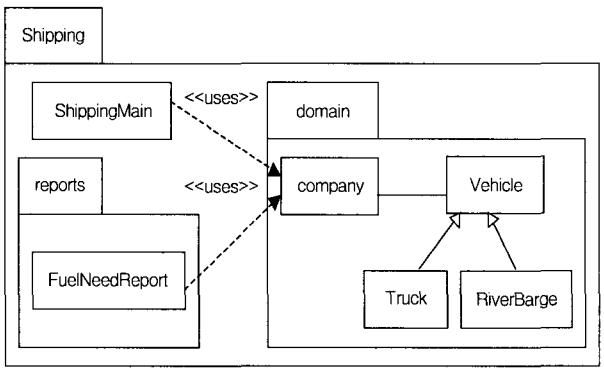
비즈니스 모델	비즈니스 유형
참여 구분 모델	기업과 소비자간(B2C) 비즈니스, 기업과 기업간(B2B) 비즈니스, 소비자와 소비자간(C2C) 비즈니스
수입 구분 모델	광고형 비즈니스, 수수료형 비즈니스, 이용료형 비즈니스, 회비형 비즈니스
상호 작용 모델	1대 1 (1 to 1) 비즈니스, 1대 다수(1 to n) 비즈니스, 다수 대 다수(n to n) 비즈니스
사업 방식 모델	소매형 비즈니스, 경매형 비즈니스, 역경매형 비즈니스, 포탈형 비즈니스, 카탈로그 판매형 비즈니스, 주문판매형 비즈니스
이식 형태 모델	순수한 인터넷 비즈니스, 인터넷에 이식된 비즈니스

2.2 컴포넌트

컴포넌트는 독립적으로 배포가 가능하며 잘 정의된 소프트웨어 구조에서 특정하게 정의된 기능을 수행하는 대체 가능한 소프트웨어 부품이다[9]. 컴포넌트는 단순한 하나의 클래스가 아닌 유스케이스(Use Case)를 가지는 상호작용 하는 클래스들의 집합으로 특정한 구조와 특정한 행위를 갖는다.

소프트웨어 컴포넌트는 재사용 가능한 소프트웨어 서비스를 독립적으로 전달 할 수 있도록 패키지화한 것이다. 일반적으로 컴포넌트는 명세, 구현 설계, 실행 모듈로 구성되어 있다. 명세는 컴포넌트의 의미를 설명하는 것으로 컴포넌트가 어떤 일을 수행하는지 어떻게 그 기능을 이용하는지 등을 설명한다. 구현 설계는 정의된 컴포넌트의 명세에 알맞게 설계하고 구축하는 방법을 설명한다. 실행 모듈은 정해진 플랫폼 사에서 컴포넌트의 기능을 전달할 수 있는 구현 결과물이다.

컴포넌트의 구성요소들은 컴포넌트 개념 모델로 표현할 수 있다[10]. 예를 들어, 컴포넌트 다이어그램을 나타내기 위해서 운송 시스템을 통하여 운송수단을 목록으로 보여주고 필요한 운행 경비를 알려주는 주간 보고서를 작성하여 보자. 이때, 운송 시스템이 ShippingMain 클래스를 갖고 있다고 할 때, 이 클래스는 회사의 운송수단 리스트를 작성하고 필요한 연료를 보고서로 만드는 컴포넌트 모델을 UML을 이용하여 나타내면 (그림 2)와 같다.



(그림 2) 컴포넌트 개념의 UML 모델(출처 : SUN의 Java programming)

비즈니스 모델에서 컴포넌트 기반의 소프트웨어가 독립적인 비즈니스 컴포넌트[14]를 이용해서 어플리케이션을 개발하기 위한 조립 기술 및 조립에 사용된 부품으로서의 컴포넌트를 관리가 필요하다. 컴포넌트 구조는 시스템을 구성하는 컴포넌트들과 비즈니스 컴포넌트의 변화를 적시에 수용하게 하는 핵심적인 개념인 인터페이스를 이용하여 구현된 것이다. 컴포넌트 기반 기술을 설계할 때 객체 지향적 방법과 UML 기반을 포함하고 있는 Catalysis 방법을 주로 사용한다. Catalysis 방법[9]은 컴포넌트 설계를 <표 2>과 같이 5단계로 나누어 설계한다.

<표 2> Catalysis의 기반 컴포넌트 설계 단계

단계	모 델	내 용
1단계	Domain Modeling	컴포넌트 스펙을 정하는 작업
2단계	System Context Modeling	인터페이스를 이용하여 시스템의 기능을 표현
3단계	Type interface Definition	오퍼레이션과 타입의 관계 및 오퍼레이션의 스펙을 확정
4단계	Component Interface Specification	실제적인 물리적 단위의 컴포넌트를 결정
5단계	Application user Interface Design	설계된 컴포넌트의 사용을 시험하는 단계

2.3 재사용 기술

소프트웨어의 재사용 기술은 구성 요소의 특성과 원칙에 따라 합성 방법과 패턴을 이용해 생성하는 방법으로 분류된다. 합성 방법은 재사용 컴포넌트의 원시 코드 형태 그대로 재사용 라이브러리에 저장되어 있어 새로운 시스템에 사용할 경우 저장된 형태를 변형시키지 않은 형태로 사용한다. 패턴을 이용한 생성 방법은 재사용 라이브러리에 재사용될 원시 코드의 일정한 규칙을 저장하여 새로운 시스템에 적용할 경우 주어진 제한 조건에 따라 규칙적으로부터 원시 코드를 생성한다. 합성 방법은 컴포넌트 검색 단계에서만 사용자의 개입이 일어나는 수동적인 방법이나 패턴을 이용한 생성 방법은 검색 단계는 물론, 패턴으로부터 원시 코드를 생성하는 단계에 파리미터를 주게 되어 사용자는 보다 능동적으로 개입하게 된다. 일반적으로 널리 이용되는 방법은 합성 방법으로 이를 기반으로 한 재사용 시스템이다.

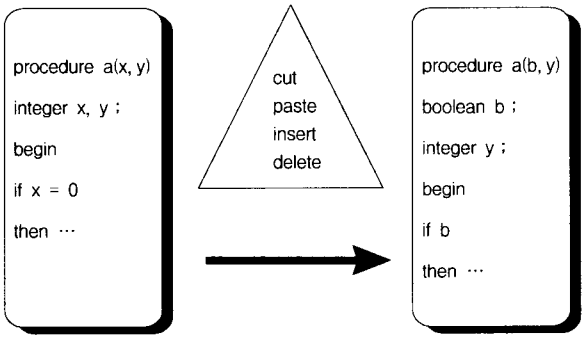
소프트웨어 공학적인 관점과 정보 공학적인 관점에 따라 재사용 컴포넌트의 특성을 분류하는 방법도 있다. 소프트웨어 공학적 관점은 재사용 컴포넌트의 공통성, 호환성, 모듈성, 유지 보수성, 접근 용이성, 이해 가능성, 재사용 가능성으로 분류하고, 정보 공학적인 관점으로는 기술적 타당성, 일반성, 수정 가능성을 고려하여 분류하기도 한다[10]. 이런 분류 방법은 각 요소들에 대한 정확한 측정이 쉽지 않고 개인에 따라 차이가 있어 명확한 분류가 쉽지 않다.

재사용 컴포넌트 라이브러리를 통해 재사용 컴포넌트을 검색하는 방법은 검색자의 검색 컴포넌트에 대한 정보와 지식의 정도에 따라 키워드에 의한 검색, 행위 나열에 의한 검색, 검색자와 상호 작용에 의한 검색등이 있다. 키워드에 의한 검색 방법은 검색 컴포넌트에 대한 이름 또는 파라미터 등 정보와 지식 정도가 많을 경우에 적합하고, 행위 나열에 의한 검색 방법은 검색 컴포넌트의 기능에 대한 정보만을 알경우에 사용된다. 검색 컴포넌트을 화면에 표시하거나 생성할 때 각 부품의 이해를 위해 필요한 설명, 매뉴얼 등을 같이 나타내 주어 컴포넌트을 이해하는데 도움을 준다[11]. 소프트웨어 재사용은 시스템을 개발하는데 이미 사용된 설

계, 원시 코드, 문서 등을 컴포넌트 형태의 정보와 지식을 체계적인 방법을 이용하여 유사한 영역이나 다른 시스템을 구축하는데 적용하는 기술이다.

2.4 버전 관리

소프트웨어의 개발 및 유지보수에 도움을 주기 위해 새로운 방법론 및 자동화 도구들이 계속해서 개발되고 있는데, 그 중 하나가 버전 관리이다. 버전 관리는 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 비즈니스 모델를 통해 컴포넌트가 수행되는 동안 구성요소를 연속된 다음 버전으로 변환시키는 이력과정(history step)을 버전 관리라 한다[12]. (그림 3)은 한 버전을 새로운 버전으로 변환하는 구성요소들의 이력과정을 위한 delta의 형태를 제안하기 위해서 삼각형 박스로 표시한다.



(그림 3) 이력 과정

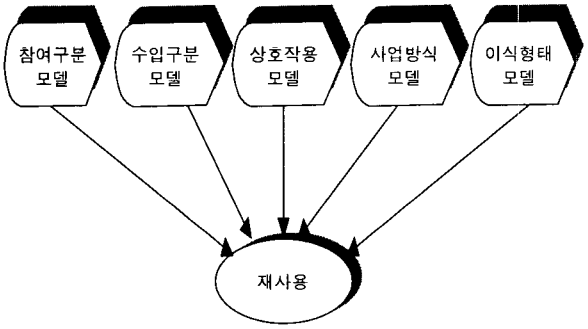
버전 관리 방법에는 full-copy 방식과 delta 방식으로 나누고 있는데, full-copy 방식은 모든 내용을 복사하여 변경 관리하기 때문에 손쉽게 모든 내용을 알 수가 있으나, 많은 공간을 차지하고 있어 공간(spaces) 활용에 있어서 효율적이지 못하다. delta 방식은 변경된 사항만을 관리하는 것으로 forward 방식과 backword 방식이 있는데, forward방식은 최초버전으로 부터 최근버전으로 검색할 수 있도록 포인터되고, backward방식은 가장 최근버전으로 부터 이전버전으로 검색하는 것으로 가장 이상적인 방식이다.

3. 재사용 비즈니스 모델의 컴포넌트 버전 관리 설계

소프트웨어 시스템 구축에 사용되는 컴포넌트 자체를 개발, 활용, 유지보수를 위해 미리 만들어진 컴포넌트를 재사용 한다는 것은 소프트웨어 개발 기간과 비용을 절감할 수 있어 생산성을 향상시킬 수 있다. 검증된 컴포넌트를 사용하여 위험 요소를 최소화 시키고, 컴포넌트 사용시 일관성을 확보할 수 있어 유지보수 비용도 절감할 수 있다.

3.1 재사용 기반의 비즈니스 모델 분류

재사용 기반의 비즈니스 모델 분류를 참여 구분 모델, 수입 구분 모델, 상호 작용 모델, 사업 방식 모델, 이식 형태 모델로 나누어진 비즈니스 모델에 따른 시스템 변화를 재사용 접근 방법을 사용하여 패키지로 구성되어 있는 것을 구분하고 객체로 세분화하여 객체로 구성되는 것을 (그림 4)과 같이 제시하여 객체의 변화를 최소화하는 것으로 시스템의 적응력을 높이고, 변화에 대하여 시스템에 영향이 적게 받도록 해야 한다.



(그림 4) 비즈니스 모델의 재사용

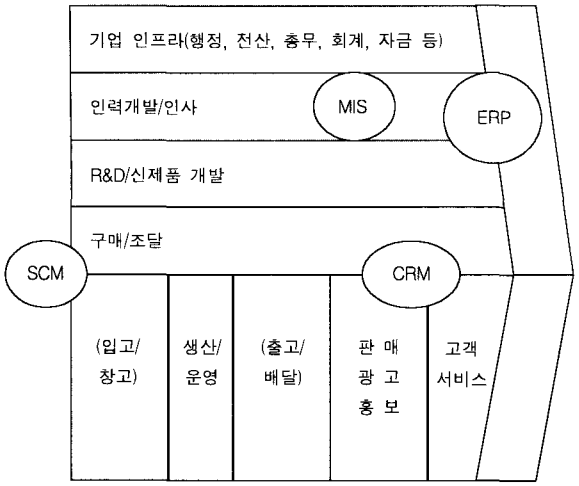
3.1.1 비즈니스 모델의 가치 사슬 컴포넌트 구조

비즈니스 모델은 기업이 어떻게 가치를 창조해서 고객에게 전달하고 이를 기업의 이윤으로 만들어 내는 가하는 문제에 대한 모델이다. 가치사슬이란 업체간 밀접한 제휴관계를 가능하게 하고, 특정 기능에 특성화하여 경제성을 추구하는 기업 또는 다양한 서비스를 제공할 수 있는 능력을 가진 기업이 경쟁상 유리하게 된다. 전자상거래 하에서는 고객이 필요한 제품을 낱낱이 탐색, 비교하는 기존 사업모델이 아닌 고객이 필요한 제품을 제시하면 업체들이 참여하는 방식의 사업모델이 가능하여 고객이 제품선택의 주도권을 갖게 되야 한다. 경영정보 시스템(MIS : Management Information System), 전사적 자원관리(ERP : Enterprise Resource Planning), 고객관계관리(CRM : Customer Relationship Management)와 공급망 관리(SCM : Supply Chain Management)로 구성된 비즈니스 모델의 가치 사슬[12]은 (그림 5)와 같이 표현할 수 있다.

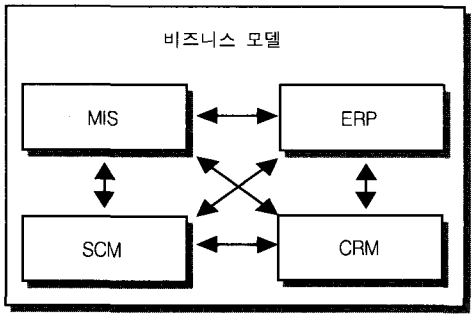
컴포넌트간의 독립적인 업무 영역을 갖도록 세분화된 프로세스별로 나누어 컴포넌트가 구성되도록 나눈다.

가치 사슬에 대한 예를 통해서 컴포넌트의 관계를 정립하여 보자. (그림 6)는 전형적인 비즈니스 모델의 연관관계로 다른 영역에 영향을 주어 독립적인 컴포넌트를 수행할

수 없으므로 좀 더 세분화된 프로세스로 독립적인 컴포넌트를 형성하여 재사용성과 적응력을 향상시킨다.

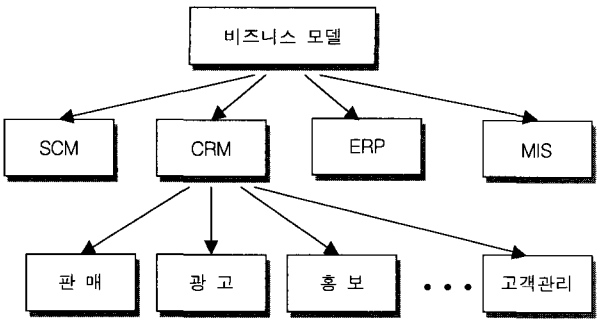


(그림 5) 비즈니스 모델의 가치 사슬(Value Chain)



(그림 6) 가치 사슬의 기본 구조

CRM의 클래스에서는 판매, 광고, 홍보 등의 컴포넌트로 나누어 독립적인 사용을 확보한다. 이것은 컴포넌트의 일관성을 유지하여 결과적으로 상호의존 관계가 제거된 (그림 7)과 같은 계층적인 구조를 갖는 컴포넌트를 구성하게 된다.



(그림 7) 가치 사슬의 컴포넌트 구조

3.1.2 비즈니스 모델의 의존관계 연산자

컴포넌트 의존관계는 제어 구조로 인터페이스에서 제시되는 연산들을 어떻게 사용할 것인가를 나타내는 것이다. 컴포넌트간의 연관관계가 해소된 제어 구조는 컴포넌트 소프트웨어 개발[14]의 구조 방식을 이용하여 집중 구조(Centralized Structure)에 따라 시퀀스 다이어그램을 통해서 구현한다. 집중 구조는 제어 구조의 수행 순서를 추상화할 수 있는 장점을 최대한 활용한다. 이것은 수행 순서를 변경하거나 새로운 연산을 추가하는 경우에 유리하다.

비즈니스 컴포넌트는 재사용의 효과를 높이기 위해서 재사용의 단위를 세분화시켜 개념적인 수준까지 컴포넌트의 크기를 확대시킬 때 재사용의 효과를 높일 수 있다.

컴포넌트의 특성상 컴포넌트는 개발 시점을 시작으로 계속해서 개선된다. 컴포넌트의 측면에서 새로운 요구를 만족하기 위한 새로운 서비스가 추가 될 수도 있고, 기존의 서비스가 삭제될 수도 있다.

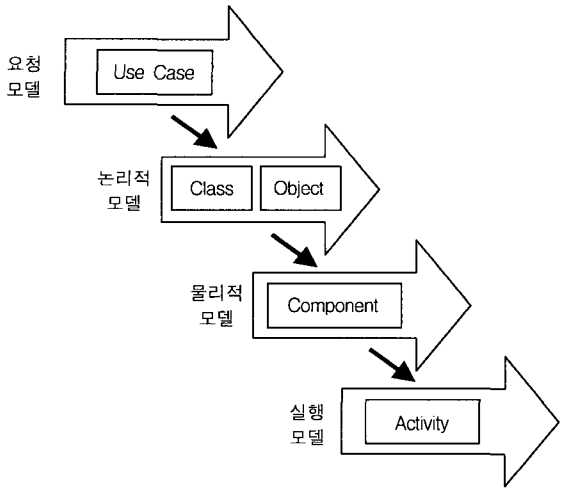
재사용 연산은 기본적인 재사용 컴포넌트를 제공하고 컴포넌트의 새로운 구성 요소의 추가하는 확장과 새로운 의존관계를 유지시켜 독립적인 구성요소를 <표 3>로 형성하게 된다.

〈표 3〉 재사용 컴포넌트의 연산

재사용 연산	의 미
확 장	컴포넌트의 기능의 확장
수 정	컴포넌트의 새로운 의존관계 추가 컴포넌트 간의 의존관계 삭제
삭 제	컴포넌트의 제거

3.2 비즈니스 모델을 이용한 컴포넌트 설계

본 논문에서는 컴포넌트가 시스템 구조에 많은 영향을 미치기 때문에 Catalysis의 기반 컴포넌트 설계을 이용하여 1단계는 요청 모델, 2단계와 3단계을 합쳐서 논리적 모델, 4단계는 물리적 모델, 5단계는 실행 모델의 개발 절차로 (그림 8)와 같이 제시하고자 한다.



(그림 8) 컴포넌트 개발 절차

3.2.1 요청 모델

전통적인 객체지향 개발 방법론과 유사한 유스케이스(Use Case)에 기반으로 한다. 사용자 요구사항을 유스케이스 다이어그램을 통해서 기술적으로 이용하여 사용자의 용어를 적용하여 사용자의 개발자간의 오해를 최소화하여 시스템의 구조적 이해도 반영한다.

3.2.2 논리적 모델

클래스 다이어그램을 통해서 시스템 구조를 기술하여 일관되게 반영하고, 시스템 내부에 존재하는 클래스들을 선별하여 나타낸다. 객체 다이어그램은 클래스 다이어그램에서 표현된 클래스가 실제 객체로 실체화되는 관계를 나타낸다.

3.2.3 물리적 모델

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

3.2.4 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.

3.3 컴포넌트 버전 관리 모델 설계

비즈니스 모델에 컴포넌트 이력 관리에 버전 개념을 도입하여 컴포넌트의 효율적인 관리를 위한 모델을 제시함으로 컴포넌트의 버전 관리를 수행할 수 있도록 하며 컴포넌트가 생성된 후 컴포넌트의 버전 변경에 따라 새로운 버전이 생성되며 각 버전의 구성 관리를 위해서 새로운 구성 정보가 생성되게 한다.

3.3.1 컴포넌트 3차원 데이터 버전 모델

본 연구에서 검토한 데이터 모델의 3차원 컴포넌트 데이터 모델은 3D-RDM (three-dimensional relation data model) 으로서 다음과 같이 정의[15]할 수 있다.

$$3\text{-}RDM = (hisR, \lambda, C)$$

hisR은 컴포넌트의 이력관계로서 이력속성들로 구성된 엔티티(entity)와 그의 어커런스(occurence)의 집합, λ는 컴포넌트 관계에 적용되는 연산자의 집합, C는 시스템 유지해야 할 무결성 제약 조건의 집합이다. 3차원 데이터 모델의 첫 번째 구성 요소인 컴포넌트의 이력관계(hisR)는 컴포넌트 이력관계의 이름은 hisV-name, m개 컴포넌트의 속성(VA)과 n개 버전 변수의 속성(TA)을 가질 때,

hisR-name (VA(1), VA(2), … VA(m), TA(1), TA(2), …, TA(n))로 정의 할 수 있다.

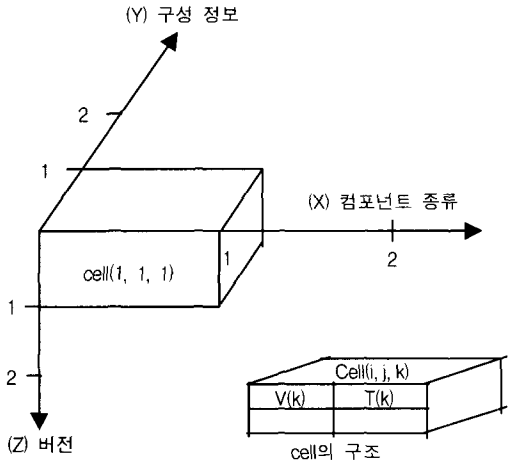
컴포넌트는 독립적인 요소로 수행이 단위가 될 수도 있고 조립과 합성을 통해서 소프트웨어 시스템을 구축할 수도 있다. 이때 합성, 조립된 컴포넌트는 기능의 개선이나 성능에 대한 요구 사항에 의해서 변경되며 이것은 컴포넌트의 변경으로 새로운 버전이 생성하게 된다.

3.3.2 컴포넌트 버전 관리 모델 구조

컴포넌트는 다른 컴포넌트와 상호 협력을 통해서 시스템을 구성하게 되는데 컴포넌트의 기능 및 인터페이스, 다른 컴포넌트의 관계를 명시하기 위해서 컴포넌트 버전 관리 모델은 (그림 9)와 같이 X, Y, Z 세 개의 축으로 표시되는 3차원 공간으로 나타낼 수 있다.

이를 구현하기 위하여 셀(cell) 변수를 사용한다. 셀변수는 X, Y, Z의 첨자를 가진다.

X축은 컴포넌트 종류, Y축은 구성정보, Z축은 버전 들을 나타내기 위한 축이다.



(그림 9) 컴포넌트 버전 관리 모델 구조

컴포넌트 버전 모델 구조는 X축의 컴포넌트의 종류로 컴포넌트의 개발 절차에 따른 모델의 종류가 나타나고 Y축은 컴포넌트에 구성 요소에 해당되는 내용에 따라 세분화된 구성 정보를 나타낸다. Z축은 컴포넌트에 대한 수정이나 추가 등이 있을 경우에 버전의 변경이 나타내어 컴포넌트를 효율적으로 사용할 수 있게 정보를 제공한다.

컴포넌트를 구현하기 위하여 셀(Cell) 변수를 사용한다. 셀변수는 i, j, k의 세 개의 첨자를 가진다.

i : 좌표상의 컴포넌트 위치
j : 구성정보의 위치
k : 버전의 위치

각 첨자는 양의 정수 값을 가지며, 셀은 버전 속성을 가지는 버전T(k)와 컴포넌트의 속성을 가지는 이력V(k)을 보관한다.

3.4 컴포넌트 버전 관리 시스템 설계

비즈니스 모델의 버전 관리 모델을 이용하여 재사용과 다형성의 역할을 수행할 수 있도록 버전 관리 시스템을 설계한다. 컴포넌트의 버전 관리를 위한 구성 요소와 시스템 구성을 제시한다.

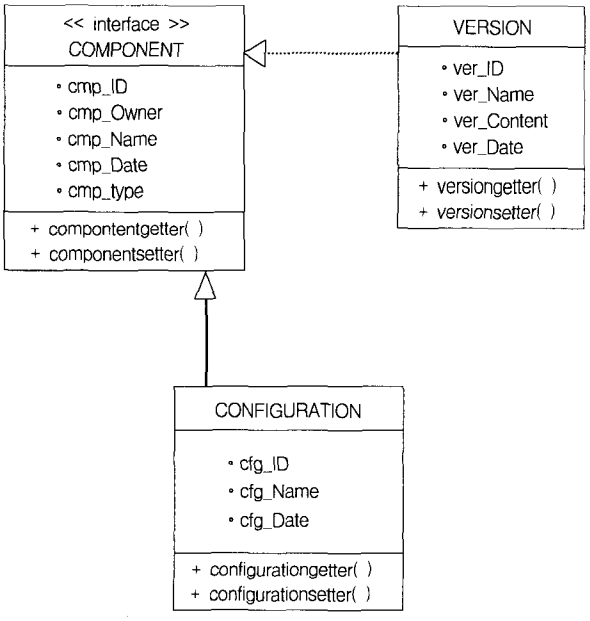
3.4.1 컴포넌트 버전 관리의 구성 요소

컴포넌트는 구조 및 기능, 성능의 변경에 따라 버전이 달라질 수 있는데, 각 버전은 버전별로 자신을 설명할 수 있는 구성 정보를 가질 수 있다. 컴포넌트 관리 모델의 구성요소는 <표 4>와 같다.

〈표 4〉 컴포넌트 관리 모델의 구성요소

구성 요소	항 목	내 용
컴포넌트	컴포넌트 ID	컴포넌트의 식별자
	컴포넌트 이름	컴포넌트의 이름
	컴포넌트 종류	컴포넌트의 분류
	컴포넌트 소유자	컴포넌트의 소유자
	컴포넌트 변경일자	컴포넌트의 가장 최근 변경된 일자
구성 정보	정보 ID	구성 정보의 식별자
	정보 이름	구성 정보의 이름
	확정 일자	컴포넌트가 확정된 일자
버 전	버전 ID	버전의 식별자
	버전 이름	버전의 이름
	버전 내용	버전의 대한 정보
	버전 변경일자	버전의 변경 일자

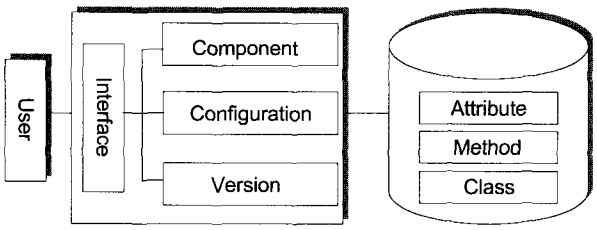
컴포넌트 관리 모델의 구성 요소는 여러 개의 구성 정보를 가지게 되는데 구성 정보는 하나의 컴포넌트가 구현되어 실행될 수 있는 단위로서 하나의 컴포넌트를 나타낸다.



(그림 10) 컴포넌트 클래스 구조

3.4.2 컴포넌트의 버전 관리 시스템 구성

컴포넌트 버전 관리 시스템은 사용자가 인터페이스를 통해서 시스템을 사용하여 컴포넌트의 변화에 따라 컴포넌트, 구성정보, 버전의 변경처리를 클래스 데이터베이스를 이용하여 관리한다.



(그림 11) 컴포넌트 버전 관리 시스템 구성

버전 관리는 컴포넌트의 삽입, 삭제 및 변경 모듈로 구성하고, 삽입 모듈에서는 속성(Attribute), 클래스(Class), 메소드(Method)의 삽입을 지원한다. 현재 클래스에서 같은 이름을 갖는 속성을 삽입하는 경우는 이름 중복 메시지를 발생하고 속성의 삽입을 허용하지 않는다. 클래스의 경우에는 현재 클래스의 하위 클래스로 삽입되며, 클래스의 삽입시에는 컴포넌트을 이용하여 생성한다.

삭제 모듈에서는 속성이나 메소드의 삭제는 현재 클래스의 컴포넌트 ID로 선택된 컴포넌트 ID를 삭제한다. 컴포넌트와 관련된 데이터는 삭제되는 것이 아니라 누락되는 것을 의미한다. 클래스의 삭제는 클래스 구조에 영향을 주므로 반드시 사용자에게 명시적으로 삭제하도록 한다.

4. 재사용 비즈니스 모델의 컴포넌트 버전 관리 알고리즘

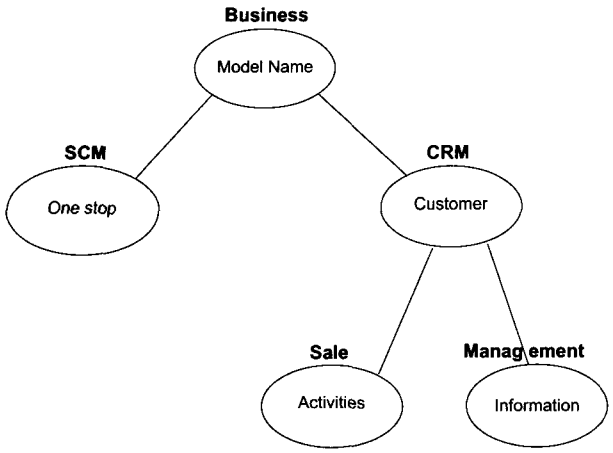
비즈니스 모델의 컴포넌트에 대한 버전 관리는 컴포넌트의 계속적인 변경은 소프트웨어에 적용된다. 컴포넌트는 이전 버전의 컴포넌트와 비교되기 때문에 반드시 버전이 부여되어 컴포넌트에 대한 유일한 식별자를 가지고 변경된 컴포넌트에 대한 새로운 버전을 부여한다.

4.1 비즈니스 모델의 컴포넌트 변경 클래스 그래프

비즈니스 모델의 컴포넌트 변경관리는 클래스의 속성에 때한 변경된 정보를 보관하고 있다가 컴포넌트의 효율적인 정보를 얻을 때 활용한다.

컴포넌트의 변경은 이전 버전의 컴포넌트와 비교하여 컴포넌트에 대한 유일한 식별자를 가지고 버전을 관리하는데, 이것은 변경에 따른 일관성을 유지하게 된다.

컴포넌트를 클래스 계층구조로 표현하여 클래스의 인스턴트들의 속성에 대한 값을 가진다. 기존에 컴포넌트에 변경이 될 때 컴포넌트의 종류에 따라 구성 정보와 버전에 영



(그림 12) 비즈니스 모델의 컴포넌트 그래프

향을 주어 상호 관계를 유지한다. 기존 속성은 delta로 보관하고 현재의 클래스에는 변경된 상태의 버전을 관리하고 있는 것을 볼 수 있다. 컴포넌트가 2002년 2월 1일에 만들어져서 3월 25일에 구성 정보의 변경이 있었고, 다시 6월 1일에 컴포넌트 종류가 변화된 경우의 컴포넌트의 변경 인스턴트 내용은 (그림 13)와 같다.

컴포넌트의 버전 관리는 비즈니스 모델에 관한 컴포넌트 클래스를 효율적으로 관리하여 소프트웨어 개발시 컴포넌트의 효율적인 재사용이 될 수 있도록 하고, 컴포넌트 검색을 신속하게 질의를 처리할 수 있게 한다.

4.2 컴포넌트 버전 관리 알고리즘

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

4.2.1 컴포넌트 버전 관리 삽입 모듈

새로운 컴포넌트의 발생은 유일하게 식별할 수 있는 식별자로써의 컴포넌트 ID를 부여하고, 새로운 컴포넌트의 발생은 클래스, 속성, 메소드 등을 삽입하는 기능을 제공한다. 특히 클래스의 삽입은 컴포넌트 생성 모듈을 이용하며 버전 관리 데이터베이스에 저장한다.

```
// Component Creation Module
begin
   call Component_inteface
       // Component_creation
        write(Component_Name, Component_Type,
              Component_Owner, Component_ Date)
       // Version_creation
        write(Version_Name, Version_Content,
              Version_Date)
       // Configuration_creation
        write(Configuration_Name, Configuration_Date)
end ;                    /* End of Insert Module */
```

(그림 14) 컴포넌트 버전 관리 생성 모듈

4.2.2 컴포넌트의 삭제 모듈

기존 컴포넌트의 삭제는 유일하게 식별할 수 있는 식별자인 컴포넌트 ID로 추출작업을 수행하여 구성 정보와 버전을 감소 시켜 삭제 처리하여 관리 한다. 삭제 컴포넌트의 발생은 클래스, 속성, 메소드 등을 삭제하는 기능을 제공한다.

```
// Component Delete Module
begin
  call Component_inteface
  search (Component_ID)        // call Component
       Delete(Component_ID)
  search (Version_ID)          // call Version
       Delete(Version_ID)
  search (Configuration_ID)    // call Configuration
       Delete(Configuration_ID)
end ;              /* End of Delete Module */
```

(그림 15) 컴포넌트 버전 관리 삭제 모듈

4.2.3 컴포넌트 버전 관리 변경 모듈

기존 컴포넌트에 대한 변경으로 구조적, 기능적 변화가 있을 때 발생하고, 컴포넌트의 구성 정보의 변화가 있을 때 생성된다. 컴포넌트에 대한 재사용을 위한 버전 관리에 저장하는 관리한다.

	Component ID	Component Name	Component Type	Component Owner	Component Date	Configuration ID	Configuration Name	Configuration Date
Version Date			02.06.01		02.06.01		02.3.25	02.02.01
Version Content			Sale2.1				Sale1.2	
Version Name			2		2		1	1
Version ID			2.1		2.1		1.2	1.0

(그림 13) 컴포넌트의 변경 인스턴트 내용

```
// Component Modify Module
begin
  call Component_inteface
  search (Component_ID)          // call Component
      Update(Component_Name, Component_Type,
            Component_Owner, Component_ Date)
  search (Version_ID)            // call Version
      Update(Version_Name, Version_Content,
            Version_Date)
  search (Configuration_ID)      // call Configuration
      Update(Configuration_Name, Configuration_Date)
end ;              /* End of Modify Module */
```

(그림 16) 컴포넌트 버전 관리 변경 모듈

5. 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.

재사용 비즈니스 모델은 기존의 작성된 재사용 컴포넌트를 관리하는데 있어 라이브러리의 구축과 이라이브러리에서 적절한 재사용 컴포넌트를 검색하는데 필요한 검색 시스템의 구현으로 컴포넌트를 관리한다. 이러한 재사용 비즈니스 모델에서는 컴포넌트 관리시 컴포넌트 이력 관리를 제공하지 않으므로 버전과 접목시켜 컴포넌트를 효율적으로 관리하기 위해서 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 실행 정보를 가시적으로 보여 줄 수 있는 실행 도구 개발에 대한 연구가 필요하다.

참 고 문 헌

[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 Component Model*," SIGS Publications, 1997.

[4] Clemens Szyperski, "*Componet Software : Beyond Object-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] Kangtae Kim, JeMin Bae, Jeong Ah Kim, Kying Whan Lee., "*Developing O-O Framework for Web Collaboration System*," 17th IASTED International Conference Proceeding, pp.165-167. 1999.

[8] Timmers, Paul, "*Business Models for electronic Markets*," Electoric Markets, Vol.8, No.2, 1998.

[9] Desmond F. D'souza and Alan C. Wills, "*Objects, Components and Frameworks with UML : The Catalysis Approach*," Addison Wesley, 1999.

[10] 김강태, "재사용 계약 기반의 컴포넌트 관리 모델을 적용한 소프트웨어 아키텍쳐 구축 방법론", 중앙대학교 박사학위논문, pp.1-9, 2000.

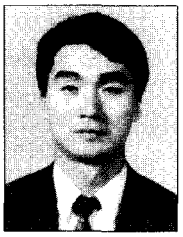
[11] 최은혁, 최은만, "역공학을 이용한 C 및 C++ 재사용 부품 추출 및 검색", 정보과학회논문지(C), 제2권 제2호, pp.197-205, 1996.

[12] 오상엽, "버전 제어에서 효율적인 형상 형성을 위한 혼합 검색 시스템의 모델링과 구현", 광운대학교 박사학위논문, pp.37-42, 1999.

[13] 삼성경제연구소, "전자상거래의 모델과 미국의 EC동향", 최종연구보고서, pp.6-8, 2000.

[14] 배두환, "e-Business를 위한 컴포넌트 소프트웨어 개발", 정보처리논문지, Vol.7, No.4, pp.27-32, 2000.

[15] 김영선, "객체지향 기법에 의한 스키마 버전제어 자동화 도구 설계 및 구현", 광운대학교 석사학위논문, pp.16-20, 1996.



김 영 선

e-mail : yskim306@daelim.ac.kr

1985년 광운대학교 전자계산기공학과 졸업(공학사)

1997년 광운대학교 전자계산학과 (이학석사)

2000년 광운대학교 컴퓨터과학과 (박사과정 수료)

1987년~1993년 (주) LG-CNS 근무

2000년 (주) 컴텍크 코리아 연구소장 역임

2000년~현재 대림대학 경영정보계열 전임강사

관심분야 : 소프트웨어공학, 데이터베이스, 버전제어, 모바일 인터넷, 컴포넌트, 보안

오 상 엽

e-mail : osy@kyungwon-c.ac.kr

1989년 경원대학교 전자계산학과 졸업 (이학학사)

1991년 광운대학교 대학원 전자계산학과 (이학석사)

1999년 광운대학교 대학원 전자계산학과 (이학박사)

1993년~현재 경원전문대학 교양과 부교수

관심분야 : 소프트웨어공학, 버전 제어, 컴포넌트, 멀티미디어, 객체지향프로그램

장 덕 철

e-mail : dcchang@cs.kwangwoon.ac.kr

1982년 고려대학교 대학원 경영정보학박사

1981년~1982년 버클리 대학교 객원 교수

1993년 광운대학교 전산사회교육 원장

1996년 광운대학교 전산대학 원장

1997년 광운대학교 이과대학 학장

1977년~현재 광운대학교 컴퓨터과학과 교수

관심분야 : 소프트웨어공학, 버전 제어, 컴포넌트, 객체지향설계 방법론