컴포넌트의 재사용과 확장성을 위한 개발 방법

이 은 서[†] · 이 경 환[††]

요 약

컴포넌트 개발 시에 속성과 행위에 대한 공용성과 가변성을 분석하여 재사용과 확장성을 제공할 수 있다. 그러므로 공용성과 가변성이 분석된 컴포넌트는 기능과 적합성의 불일치를 줄이고자하는 목적에서 사용하게 된다. 본 논문에서는 영역에 의하여 존재하는 컴포넌트의 공용성과 가변성을 분석해서, 컴포넌트의 기능을 행위분석에 의하여 추출하였다. 이와 같은 분석에 의하여 컴포넌트의 재사용과 확장성을 연구하고자 한다.

A Study of Development Method for Component Reuse & Extension

Eun Seo Lee[†] · Kyung Whan Lee[††]

ABSTRACT

When you develop component, you can offer reusability and extension by analyzing commonality and variability about attribute and behavior. Therefore Component that analyzing commonality and variability are use for the purpose of reducing discord of function and adaptedness. I wish to research reusability and extension of component by analysis commonality and variability of component that exist by domain in this paper, extract function of component by behavior analysis.

키워드 : 컴포넌트 개발 방법(component development methodology), 컴포넌트 재사용(component reuse), 컴포넌트 공용성(component commonability), 컴포넌트 가변성(component variability)

1. 서 론

소프트웨어의 개발에서 신뢰성의 증가와 생산성, 품질, 효율성을 높이고, 보다 용이하게 하기 위하여 객체 지향 개념이 나타나게 되었다[1]. 객체지향 개념은 표준 인터페이스와 상속에 의해 코드의 재사용을 증가시키는 방법을 제공하고, 추상 클래스와 템플리트 기법은 설계 정보에 대한 재사용과 코드의 재사용을 동시에 지원함으로써 소프트웨어 개발의 막대한 부분을 차지하는 분석 및 설계 정보에 대한 재사용을 가능하게 하였다. 그리고 이를 더욱 발전시켜 유사 영역의 프로그램 군(群)의 일반적이고 공통적인 설계 정보 및 프로그램 구조를 제공한다[2, 3]. 따라서 부품화와 조립의 특성을 갖는 컴포넌트 기술의 사용으로 컴포넌트 기반 개발 방법이 대두되었다[4]. 그러나 컴포넌트를 사용하는 과정에서 개발영역의 특성을 잘못 분석하여 예상하는 것과는 다른 결과를 산출하는 오류가 발생하곤 한다. 즉, 하나의 컴포넌트가 여러 개발영역에 적용할 수 없게 된다. 이러한 오류를 줄이기 위하여 컴포넌트 개발 방법과 연속적인 기술발전이 병행되어야 한다. 컴포넌트 개발 방법과 연속적인 기술발전을 위한 요소로는 융통성, 적응성, 유지보수, 재사용, 속한 통합, 상호작용, 확장성이 있으며, 이와 같은 요소의 기능을 이용하여 손쉽게 적은 비용으로 재사용 가능한 컴포넌트를 개발할 수 있다[5].

컴포넌트 재사용 시에 하나의 컴포넌트를 같은 개발영역에서 적용을 하더라도 여러 가지 영역의 상황에 의하여 가변성이 존재할 수 있다. 따라서 본 논문에서는 컴포넌트의 재사용 과정에서 발생할 수 있는 확장성을 효과적으로 분석 및 설계하기 위하여 공용성과 가변성을 추출하는 방법을 연구하고자 한다.

2. 기반 연구

2.1 컴포넌트 특성

컴포넌트는 인터페이스를 통하여 서비스를 제공하고 요구할 수 있는 것으로 독립적이고, 작은 단위로 나누어질 수 있는 시스템의 요소이며, 시스템의 필요에 의하여 다른 요소로 대체 가능한 것이다. 그리고 미리 개발된 응용코드의 일부분이다[6]. 재시된 컴포넌트의 정의를 기초로 하여, 컴포넌트의 특성을 나타낼 수 있다. 특성으로는 식별가능성, 추적가능성, 교체 가능성, 인터페이스에 의한 접근 가능성, 인터페이스가 제공하는 서비스 고정성, 문서로 정확히 기록되는

† 준 회 원 : 중앙대학교 대학원 컴퓨터공학과
†† 정 회 원 : ISO/SC7/WG10 SPICE 한국 위원장
논문접수 : 2001년 8월 7일, 심사완료 : 2002년 6월 24일

서비스, 물리적 구현의 은닉, 독립성, 언어 및 개발 툴에 독립적인 재사용성, 플랫폼과 무관한 재사용성, 동적 재사용을 제시할 수 있다. 또한 컴포넌트는 일반적으로 논리적인 관점과 물리적인 관점의 컴포넌트로 나눌 수 있다[5]. 컴포넌트의 특성을 기반으로 하여 논리적인 컴포넌트와 물리적인 컴포넌트의 개념을 제시하면 다음과 같다. 논리적인 컴포넌트는 비즈니스 컴포넌트를 의미하며, 비즈니스 영역에서 실세계의 개념을 모델링한 것을 나타낸다. 반면에 물리적인 컴포넌트는 비즈니스 컴포넌트를 독립적인 소프트웨어로 나누어서 공학적인 관점으로 구축을 하는 것을 의미한다. 그리고 컴포넌트의 개발 방향은 위에서 제시된 정의와 특성을 기반으로 재사용성과 확장성이 주된 요소가 된다. 이 외에도 재사용성과 확장성에 연관된 컴포넌트 특성중의 하나가 패밀리요소이다.

2.2 컴포넌트 패밀리

패밀리의 정의는 개발영역에서 요구사항의 명세를 포함한 것으로서, 공용적인 면과 가변성 측면의 아이템 집합이다[7]. 패밀리를 구성하는 것은 패밀리 멤버이며, 패밀리 멤버는 공통적인 측면과 가변적인 측면을 갖게 된다. 패밀리 멤버의 설명은 다음과 같다. 공용성은 패밀리의 모든 멤버에 대하여 항상 공통적으로 수용 가능한 것이다[8]. 그리고 공용성 요소의 식별은 공용성 분석이라는 패밀리의 정의 프로세스를 기반으로 이루어진다[7]. 공용성의 식별은 컴포넌트에서 재사용 될 수 있는 부분을 구별하는 것과 같은 의미가 되고, 결과적으로 재사용성을 향상시킬 수 있게 된다. 공용성 부분이 식별되고 이를 기반으로 컴포넌트 설계가 이루어진다면, 사용자는 컴포넌트 스펙에 따라서 재사용 되는 부분을 쉽게 식별할 수 있게 된다. 그러므로 재사용성을 향상시키기 위해서 공용성의 식별은 필수적인 요소가 된다.

가변성은 하나의 패밀리, 또는 멤버에서 수용 가능 하지만 다른 패밀리멤버에서는 수용 불가능한 것을 말한다[9]. 따라서 가변성은 컴포넌트를 재사용하고 확장하는 경우에 예기치 못한 상황을 발생시킬 수 있다. 예기치 못한 상황으로는 인터페이스를 추가하거나 필요 없는 부분을 삭제하는 경우에, 인터페이스 본래의 기능을 바꿀 수가 있다. 이로 인해서 전혀 원하지 않는 작업을 수행하여 예상치 않는 결과가 도출되는 상황이 발생할 수 있다. 그러므로 컴포넌트 개발 시 재사용 할 수 있는 부분과, 개발영역의 특성과 사용자의 요구사항에 맞게 수정을 해야 하는 부분으로 명시해줄 필요성이 요구된다. 이를 위하여 공용성과 가변성의 분석과 추출은 컴포넌트의 재사용과 확장성에 필수적으로 요구되는 단계로서 본 논문에서는 컴포넌트 분석 및 설계 시에, 공용성과 가변성을 추출하여 재사용을 효과적으로 수행할 수 있는 설계 방법을 제시한다.

2.3 기존 컴포넌트의 개발 형태

컴포넌트를 설계하기 위한 개발 과정의 관리와 컴포넌트 기능상의 오류를 줄이고, 컴포넌트의 특성인 재사용성을 제공하기 위하여 개발 방법론의 적용이 필요하다. 다양한 개발 방법론 중에서 컴포넌트의 개발자가 방법론을 선택하게 되는데, 개발될 컴포넌트의 특성과 환경을 고려해서 개발 방법론을 적용하게 된다.

선택될 방법론은 컴포넌트 개발과 어플리케이션 관점으로 컴포넌트 설계를 하게 되는데, 어플리케이션의 개발영역을 중심으로 영역 분석을 한 후, 컴포넌트 설계와 구현이 이루어진다. 그리고 구현된 컴포넌트는 어플리케이션에서 컴포지션(composition)에 의하여 어플리케이션 시스템에서 사용되게 된다[10]. 다른 방법으로는 비즈니스 모델링과 계획, 관리, 진화(evolution)을 포함하는 비즈니스 프로세스와 컨텍스트내에 컴포넌트 분석, 설계, 개발을 포함하는 아키텍처 생명주기(Architecture Life Cycle)가 있다[12].

CBSE 개발 방법론에서 컴포넌트 기반 설계 방법론으로는 여러 가지가 존재한다. 많은 방법론 중에서도 카타르시스(Catalysis)를 많이 참조하는데, 이 방법은 UML(Unified Modeling Language)로 표현하며, 크게 세 부분으로 구성이 되어있다[13].

2.4 컴포넌트 재사용 시의 문제점

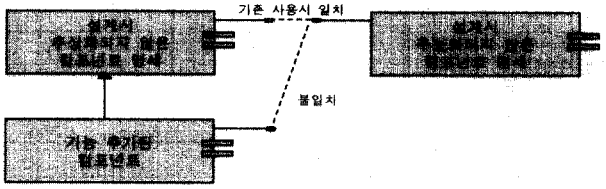
여러 컴포넌트는 개발 영역에 적합하도록 개발이 되어지고 있다. 그러나 같은 영역이라도 컴포넌트가 적용 및 배치될 환경에 의하여 오류가 발생하게 된다. 이와 같은 컴포넌트는 다른 영역과 어플리케이션에서 재사용 할 수 없게 되고, 특정 영역과 어플리케이션에서도 커스터마이징(customizing)이 불가능하기 때문에 재사용성과 확장성을 기대할 수 없게 된다. 따라서 변화 요소를 개발시점에서 분석하지 않는다면 재사용과 확장성이 제공할 수 없게 되며, 변화요소가 발생될 때마다 개발초기단계에서부터 모든 과정을 수행을 해야하므로 개발비용과 노력의 절약을 얻을 수 없게 된다. 본 절에서는 앞에서 제시한 내용의 문제점을 세분화하여 세 가지 측면에서 제시하였다.

2.4.1 컴포넌트 설계 시 명세의 추상화 문제

전통적으로 객체지향 모델링은 클래스와 같이 프로그래밍 언어의 단위에서 시스템의 명세를 구조화 하지만 컴포넌트의 경우는 인터페이스를 통하여 사용되게 된다. 따라서 컴포넌트에 상호 연결하기 위하여 컴포넌트의 인터페이스가 식별되고, 컴포넌트는 완전한 어플리케이션 시스템으로서 다양한 크기와 복잡도를 가진 객체가 된다. 그러므로 레가시(legacy) 어플리케이션은 잘 정의된 인터페이스를 통해서 하나의 객체와 같이 독립적인 행위를 할 수 있다. 또한 클래스의 연결 생성/명세, 집합, 또는 합성에 의하여 컴포넌

트를 연결하기 위하는 경우에는 일반적인 방법으로 명세를 나타내어서 이를 연결 시에 사용할 수 있도록 한다. 예를 들면 UML을 이용한 명세를 뜻한다.

컴포넌트 명세에 추상화가 적용되지 않은 컴포넌트는 새로운 개발영역에서 요구하는 기능을 수용할 경우에, 컴포넌트의 연결과 합성의 예측이 불가능하게 되므로 협력(collaboration)이 불가능하게 된다[14]. (그림 1)은 컴포넌트 설계 시 명세의 추상화가 이루어지지 않았기 때문에 새로운 영역이나 어플리케이션의 요구사항을 만족할 수 없는 경우를 나타낸 것이다. 그러므로 (그림 1)과 같은 형태의 컴포넌트는 협력(collaboration)이 불가능하게 된다.

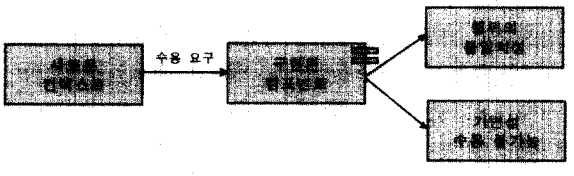


(그림 1) 컴포넌트 설계 시 명세의 추상화의 중요성

2.4.2 컴포넌트의 새로운 컨텍스트 수용 문제

재사용 가능한 컴포넌트 개발의 목적 가운데 하나가 많은 제품과 영역에서 재사용 가능하도록 하는데 있다. 또한 새로운 컨텍스트가 발생한 경우에, 이전의 컨텍스트는 새로운 컨텍스트를 적절히 수용해야 할 수 있어야 한다.

컴포넌트의 새로운 컨텍스트 문제에서 주요한 사항은 새로운 컨텍스트를 찾기 위한 문제이다. 이와 같은 사항에는 두 가지 측면으로 접근할 수 있다. 첫째는 컴포넌트는 특정 영역과 프로덕트의 특성, 기능 컨텍스트와 요구사항의 집합을 기반으로 하여 개발되어진다. 이와 같은 요소를 기반으로 개발된 컴포넌트는 영역이나 프로덕트, 또는 기능 컨텍스트에서 하나라도 변화가 발생하는 경우에 새로운 내용을 찾아서 적용하는 것이 어렵게 된다. 둘째는, 요구되어지는 가변성의 디자인이다. 재사용 가능한 컴포넌트는 초기 요구사항 스펙에서 가변성을 수용하는 것이 가장 좋은 형태가 된다. 그러므로 새로운 컨텍스트에서는 발생하면, 컴포넌트에서는 이를 새로운 가변성으로 인식하게 된다[15].



(그림 2) 컴포넌트의 새로운 컨텍스트 수용 문제

2.4.3 컴포넌트간의 전개성

컴포넌트는 재사용을 위하여 전개 시에 의존성과 확장성의 특성과 연관되어진다. 이와 같은 특성은 재사용을 위하여 필수적이지만, 시스템의 모듈화와 디자인에 의하여 만족하기는 어렵게 된다. 그러므로, 컴포넌트가 전개되는 과정에서 의존성과 확장성이 제공되지 않은 문제점이 발생한다.

컴포넌트 전개시의 문제점은 유지보수 관점에서 문제가 발생된다. 즉, 컴포넌트가 추가되는 기능을 수용하기 위하여 확장이 가능해야 한다[16]. 확장되는 기능이 여러 컴포넌트에 의존성을 갖는 경우, 확장 시에 여러 컴포넌트에게 기능상의 오류를 발생시켜서는 안된다. 따라서 컴포넌트 설계 시에 확장성을 고려해야 한다.

본 논문에서는 재사용 될 수 있는 컴포넌트를 분석 및 설계하는데 초점을 맞추었으며, 이를 달성하기 위하여 컴포넌트의 요소 중에서 공용성과 가변성의 추출을 통하여 재사용성과 확장성을 연구하고자 한다.

3. 컴포넌트의 공용성과 가변성 추출 과정

컴포넌트의 공용성과 가변성을 추출하기 위한 환경으로는 이벤트에 의하여 서비스가 요구되는 영역을 대상으로 하였다. 컴포넌트의 공용성과 가변성은 세 단계로 추출을 하게되며, 각 단계는 요구사항 추출, 공용성과 가변성 식별, 컴포넌트 식별과 인터페이스 추출 단계로 나누어서 추출을 하게 된다. 각 단계는 순차적이며, 전 단계의 결과물은 다음단계의 입력이 된다.

본 논문에서는 조소(address)를 중심으로 하여 다른 영역에서 컴포넌트가 재사용 될 때 발생하는 가변성을 추출하게된다. 따라서 주소와 연관된 컴포넌트의 설계시 공용성과 가변성을 분석하여 설계를 수행하게 되고, 이를 통하여 발생될 수 있는 위험요소를 제거하고자 한다.

3.1 요구사항 추출

요구사항을 추출하기 위하여 고려해야 할 사항으로 크게 시스템과 사용자 측면의 요구사항으로 구분을 할 수 있다. 요구사항 추출은 시스템의 내·외부적 요소와 연관성이 존재하는가를 식별하고, 사용자의 요구사항간에 어떤 연관성이 있는지의 사항을 추출하기 위함이다. 이는 이벤트가 발생되는 경우를 기반으로 하여 제시한 것이다. 사용자의 요구사항을 시스템측면과 요구사항간의 연관성 측면에서 구분을 한 표는 다음과 같다.

〈표 1〉 사용자의 요구사항 분석표

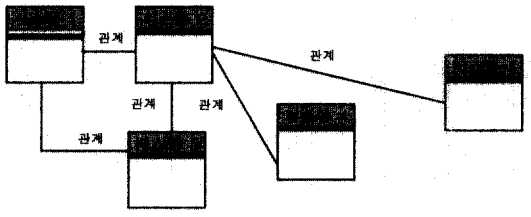
분석 요인 / 사용자의 요구사항	필요한 외부 환경	요구 사항간의 연관성	시스템과의 연관성	시스템 수용 가능성
요구사항 1				
요구사항 1				
⋮				
요구사항 n				

3.2 공용성과 가변성의 식별

현 단계는 요구사항 분석결과를 기반으로 하여 개발하려는 컴포넌트의 공용성과 가변성 요소를 추출하기 위하여 수행을 한다. 구성 단계로는 시스템 구조인자 식별, 시스템의 행위와 상태추출 단계로 제시하였다. 각 단계를 상세히 살펴보면 다음과 같다.

3.2.1 시스템 구조인자 식별

시스템 구조인자 식별 단계는 시스템간의 연관성을 추출하기 위한 단계이다. 시스템간의 연관성을 추출해야 서로 간의 상관관계를 얻을 수 있다. 추출된 상관 관계를 기반으로 공용성과 가변성의 인자를 추출한다. 시스템 구조도에서 시스템간의 관계를 명시함으로서, 연관성을 쉽게 인식할 수 있다. 시스템 구조도는 다음과 같다.



(그림 3) 시스템 구조도

시스템의 구조가 분석된 이후, 서브 시스템 요소를 추출한다. 서브 시스템의 추출 이유는 요구하상이 시스템에서 완성되기 위하여 시스템관점의 연관성이 요구되게 된다. 따라서 이를 위하여 caller와 callee를 판별하여 의존성을 식별하였다. 이것은 서로의 하부 인자를 call하는 경우에, 연관성을 식별하기 위함이다. 이와 같은 시스템 구성인자의 연관성은 <표 2>와 같이 나타낼 수 있다. 여기서 call의 의미는 식별된 시스템 구성 인자간의 연관성을 나타낸 것이다. 그리고 call의 식별은 요구사항 추출을 기반으로 한다.

〈표 2〉 시스템 구성인자의 연관성

caller element \ callee element	A	B	C	D
A		call		
B			call	
C				
D			call	

<표 2>에서 A, B, C, D는 시스템을 구성하고 있는 인자를 나타낸다. 여기서는 A가 B를 호출하고, B가 C, 그리고 D가 C를 호출하는 경우가 된다.

3.2.2 시스템의 행위와 상태추출

시스템 구성 인자간의 연관성을 식별한 후에는 내부적으로 메시지를 이용하여 서로를 호출하는 구조가 형성된다. 이를 가능하게 해주는 것이 행위라는 관점에서 가능하다. 행위는 수행 시에 시간에 의존적이기보다는, 시점을 기반으로 표현되는 행동들의 합성으로 구성되게 된다. 행위를 이용하여 공용성과 가변성을 추출하게 되는데, 가변성 식별이 가능한 이유는 존재하는 행위가 원상태대로 사용되는 것이 아니라, 상태조건에 의존하여 변화가 있을 수 있다라는 점에서 가능하게 된다. 따라서 상태가 행위를 결정할 수 있는 중요한 요소가 된다. 이것은 컴포넌트의 공용성과 가변성의 측면에서 본다면, 행위가 상태변화의 필요성이 요구되는가의 유·무로 공용성과 가변성을 식별할 수 있다. <표 3>은 시스템의 구성인자 중에서 A라는 인자가 가지는 행위와 상태에 의한 공용성과 가변성의 관계를 나타낸 것이다.

〈표 3〉 행위호출에 의하여 요구되는 상태표

A'behavior \ callee	A	B	C	D
b_1	s_1	s_2	s_3	s_4
b_2	s_5	s_5	s_5	s_5
b_3	s_6	s_6		
b_4	s_8	s_8	s_9	

<표 3>에서는 행위를 b_i, 상태는 s_i로 나타낸다. <표 3>에서 행위b_1의 경우, 호출될 때마다 상태가 s_1, s_2, s_3, s_4로 다른 상태를 요구한다. 따라서 행위 b_1은 가변성을 포함하고 있다. 그러나 b_2는 호출될 때마다 같은 상태를 요구하므로 공용성으로 구분된다.

추출된 행위의 공용성과 가변성을 기반으로 구분 표를 작성한다. 표에서는 공용성과 가변성의 조건이 존재하는 경우를 기입해서 행위의 상태를 명세 한다.

〈표 4〉 행위의 공용성과 가변성 조건 표

A' behavior	공용성과 상태 조건	가변성과 상태 조건	비 고
b_1	condition1		
b_2		condition	
b_3		condition	
b_4	condition		

<표 2>는 컴포넌트의 연관성을 정의할 때에 사용이 되고, <표 3>과 <표 4>는 컴포넌트 명세 정의 시에 사용된다.

3.3 컴포넌트 식별과 인터페이스 추출

컴포넌트 식별을 위하여 시스템의 Use Case를 작성하고, 이를 기반으로 객체를 추출한다. 객체의 추출은 일반적인 시퀀스다이어그램을 사용한다. 그 이유는 객체의 동적인 면을 나타내기 위함이다. 컴포넌트 식별은 하나의 Use Case를 기반으로 이루어진다. 이때 식별된 행위의 공용성과 가변성을 고려해야 한다. 그리고 식별된 컴포넌트의 책임을 부여하기 위하여 식별된 객체를 기반으로 객체간의 이벤트 흐름을 분석하여 속성과 오퍼레이션, 그리고 인터페이스를 결정한다. Use Case에 의하여 기능중심으로 컴포넌트를 식별하

고, 여기에 공용성과 가변성을 적용하여 컴포넌트 구축시에 분석 및 설계단계에서 오류가 될 수 있는 가변성을 명시할 수 있다. 그리고 이러한 가변성을 명시함으로서, 사용자가 가변성에 해당되는 부분을 커스터마이징과 테일러링이 가능하게 하여서 같은 영역내에서 뿐만 아니라 다른 개발 영역에서도 재사용성과 확장성을 높일 수 있게 된다.

〈표 5〉 인터페이스 추출 형식

구성 형태	내 용
Interface	식별된 인터페이스의 이름을 입력한다.
Preconditions	현재의 인터페이스의 선행조건을 기술한다.
Postconditions	현재의 인터페이스의 후행조건을 기술한다.
Input Parameter	현재 인터페이스의 입력 파라미터를 나열한다.
Output Parameter	현재 인터페이스의 출력될 파라미터를 나열한다.
Use Case	현재 인터페이스와 연관된 유즈 케이스를 기술한다.

앞에서 컴포넌트의 식별이 이루어졌고, 각 컴포넌트를 구성하는 인터페이스는 <표 5>와 같은 형태로 명세 할 수 있다. 따라서 인터페이스를 정의한다는 것은 컴포넌트간의 책임과 관계를 명시한다는 것이므로. <표 5>를 기반으로 하여 컴포넌트의 책임은 interface를 정의함으로서 부여하게 된다. 따라서 <표 5>에서는 컴포넌트의 인터페이스에 해당되는 행위의 연관성을 명시하기 위한 인터페이스 추출 형식을 정의한 것이다. 결론적으로, 인터페이스와 컴포넌트의 관계는 클래스 다이어그램으로 나타내게 된다.

4. 적용 사례

제 3장에서 제시한 컴포넌트의 공용성과 가변성의 추출 방법을 적용하기 위하여, 제 4장에서는 영역을 설정하고 그 영역에서 컴포넌트의 공용성과 가변성을 추출하고자 한다. 우선 해당 영역에서의 컴포넌트에 대하여 분석을 하고, 이를 기반으로 실제 컴포넌트의 재사용 시에 사용할 수 있는 확장 타입을 추출한다.

4.1 영역 설정

적용할 영역은 전자 고지서 발급 시스템과 전사적인 자원 관리 시스템에서 사용할 수 있는 주소(address) 부분이다. 또한 사례 적용될 컴포넌트는 전자 고지서 발급 시스템뿐만 아니라, 개인 정보를 요구하려는 시스템에서 이용할 수 있다.

본 논문에서 개발될 주소 컴포넌트는 C2R(Cyber to Reality)라고 명시하였다.

4.2 컴포넌트 개발을 하기 위한 선행 작업

컴포넌트 개발을 하기 위한 선행 작업으로는 개발 영역 선정, 사용자의 요구사항 분석, 운영 시나리오 작성이 요구된다. 개발 영역 선정은 영역 설정에서 제시를 하였고, 사용자의 요구 사항 분석은 다음과 같은 표를 기반으로 수행한다.

〈표 6〉 주소 컴포넌트의 사용자 요구사항 분석

분석 요인 / 사용자의 요구사항	필요한 외부 환경	요구 사항간의 연관성	시스템과의 연관성	시스템 수용 가능성
고지서 발송	mail server 구축	사용자 확인, 공공성 부여	EJB	가능
사용자 확인	인증 서버 구축, 사용자 개인 정보 DB 구축	사용자에게 전송		가능
사용자에게 전송	mail server 구축	공공성 부여		가능
공공성 부여	사용자 개인 정보 DB 구축	사용자 확인	EJB	가능

현 시점에서의 운영 시나리오는 개괄적인 내용이 된다. 운영 시나리오는 사용자의 요구사항을 기반으로 하고, 시스템 분석이 이루어지기 전에 사용자의 요구사항에 의한 시나리오를 구축하게 된다. 운영 시나리오는 Use Case에서 사용되는 시나리오를 확장하고 정확하게 명세 할 수 있는 기초 자료가 되며, <표 7>과 같은 절차로 나타낼 수 있다.

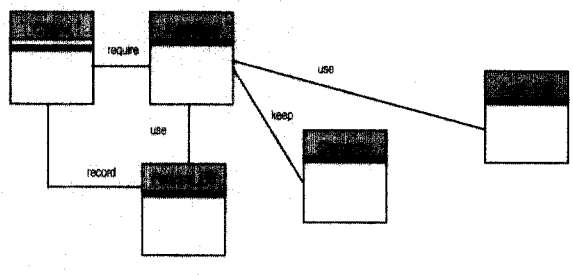
〈표 7〉 운영 시나리오

단계	운영 시나리오
1	가상 주소(cyber address)에 의해서 고지서 발송 업체의 고지서 발송을 요청
2	발송 받을 사용자를 확인
3	고지서 내용 확인
4	사용자의 실제 위치(Reality) C2R 체계 적용 변환
5	고지서 내용 전송 (발송지역의 인쇄 센터로 전송)
6	고지서 발송(문서 발송 POOL 시스템) 체계 (고지서별 인쇄, 배달을 위한 분류, 배달)

4.3 공용성과 가변성의 식별

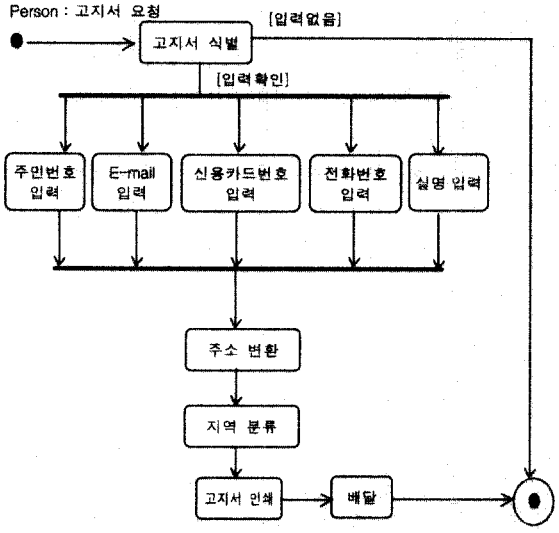
4.3.1 시스템 구조인자 식별

시스템 구조인자 식별 단계에서는 개발영역 전문가의 지원이 요구되며, 사용자 요구사항을 기반으로 하여 비즈니스의 특성을 분석하게 된다. 개발영역의 업무 분석을 위한 공통적인 요구사항에 의해서 초기 비즈니스 모델링을 수행한다. 그리고 현재 개발하고자 하는 시스템은 주소(address) 컴포넌트로서 사용자의 정보를 요구하는 시스템에서 사용하고자 한다.



(그림 4) C2R 컴포넌트 시스템 구조도

현재 개발 과정을 나타내는 컴포넌트는 주소(address)가 중심요소가 되어서 이를 분석하여 다른 영역에서 사용될 때, 공용성과 가변성의 대상을 식별하고자 한다. (그림 4)의 시나리오는 다음과 같다. 클라이언트는 주소부분에 고지서 발송을 요청한다. 주소는 개인 정보 데이터베이스(Personal_DB)를 사용하여서 추가적인 내용을 저장(Add DB)한다. 그리고 보안(Security)도 유지를 하게된다. 분류 및 전송은 고지대상자를 거주별로 분류를 하고 전송을 한다. <표 7>의 개괄적인 운영 시나리오와 (그림 4)을 기반으로 하여서 구체적인 시나리오를 작성한다.

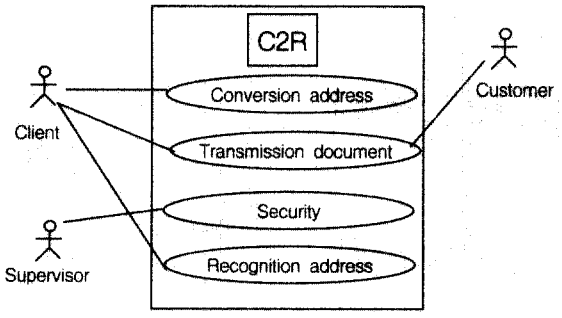


(그림 5) 주소 체계활용을 위한 구체화된 시나리오

(그림 5)에서 배달은 프린터를 소유한 고객과 소유하지 않은 고객으로 나누어서 배달을 한다. 프린터를 소유한 고객에게는 전송을 바로 하지만, 미 소유 고객은 집배지에서 프린트하여 전송을 한다.

4.3.2 서브 시스템 식별과 연관성 정의

시스템 구조도를 기본으로 하여 식별된 인자에 존재하는 서브인자를 추출한다. 추출은 인식을 용이하게 하기 위하여 UseCase를 사용한다. 추출된 UseCase는 (그림 6)과 같다.



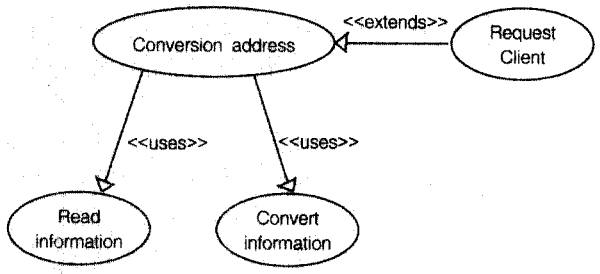
(그림 6) C2R UseCase 모델링

(그림 6)에서 추출된 각 UseCase는 <표 8>과 같은 기능으로 액터와 시스템간에 상호 작용을 한다.

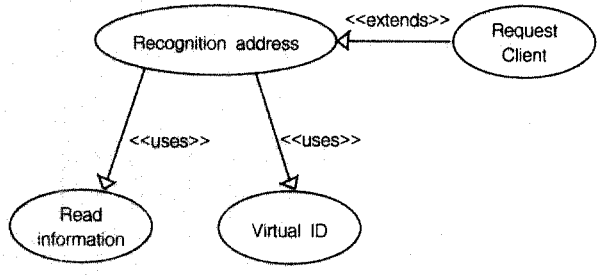
<표 8> UseCase의 내용

UseCase	UseCase의 내용
Conversion Address	고객의 주소를 가상 주소로 변환하는 UseCase
Transmission Document	고지서 발송 업체의 요청에 의하여 발송 될 문서(고지서)를 전송
Security	고객이 정보 유출을 방지(외부의 서버로 운영)
Recognition Address	수령할 고객의 주소를 인식하여 확인

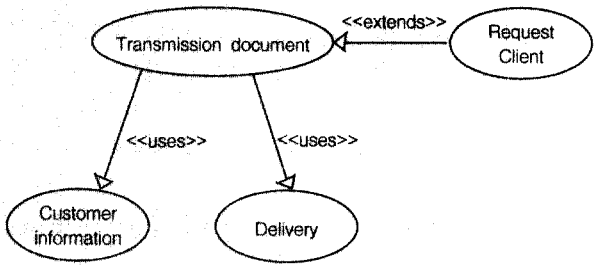
서브 시스템 인자는 (그림 7), (그림 8), 및 (그림 9)의 UseCase와 같다.



(그림 7) C2R 주소 변환 UseCase 연관도



(그림 8) C2R 문서 전송 UseCase 연관도



(그림 9) C2R 주소 인식 UseCase 연관도

주소 컴포넌트를 개발하기 위하여 시스템 내부적으로 연관성을 식별해야 한다. 따라서 분석된 UseCase와 서브 시스템 인자간의 연관성을 분석한 것이 <표 9>와 같다.

〈표 9〉 시스템 구성인자의 연관성

callee element / caller element	Conversion Address	Transmission Document	Recognition Address
Legacy DB	call	call	call
Update DB	call		
Request Client		call	call
E-mail Server		call	
Post		call	
Deliver	call	call	
Virtual ID			call

서브시스템 인자와의 연관성은 상위 시스템인자와 요구사항을 기반으로 하여 추출한다.

4.3.2 시스템 행위와 상태 추출

주소 컴포넌트 시스템의 구성인자 관계를 명시함과 동시에 구성인자간의 행위를 추출할 수 있다. 추출된 행위는 <표 10>과 같다.

〈표 10〉 주소 컴포넌트 행위의 설명

행위 네임	행위 네임의 의미
CusNum	고객번호
CusAddr	고객 주소
CusInfo	고객 신상 명세
C2RAddr	공공성이 부여된 주소
Docu	문서 발송

주소 컴포넌트를 추출하기 위해서는 추출된 행위와 Use Case간의 연관성 정의에 의하여 컴포넌트에서 필요로 하는 메소드를 추출할 수 있다. 따라서 행위과 UseCase간의 관계는 <표 11>과 같다.

〈표 11〉 행위와 UseCase간의 연관성

callee element / caller element	Conversion Address	Transmission Document	Recognition Address
CusNum	call		
CusAddr		call	
CusInfo	call		call
C2RAdddr			call
Docu		call	

추출된 행위의 상태조건을 기반으로 공용성과 가변성을 추출하게 된다. 여기서는 행위가 동작을 하는 것을 내·외부적으로 이벤트를 받는 경우와 예외적인 상황을 처리하는 경우로 제시하였다. 따라서 상태 조건의 추출도 세 가지 경우에 초점을 맞추어서 추출이 된다.

내·외부적인 이벤트와 예외 상황에 따른 메소드 추출은 행위의 상태 조건에 의하여 구분을 한 것으로서 <표 12>와 같다. <표 12>에서 메소드는 시스템 구조 인자와 연관된 UseCase를 기반으로 추출했다.

〈표 12〉 행위의 상태 조건과 메소드 추출표

행위 이름	행위의 상태 조건	추출된 메소드
CusNum	<Event> 고지서 발송 요청, 주소 변환 요청 <Exception> 없음	CusNumSend( ) CusNumReceive( )
CusAddr	<Event> 고지서 발송 요청 주소 변환 요청 <Exception> 없음	CusAddrSend( ) CusAddrAdd( ) CusAddrDelete( ) CusAddrCreate( ) CusAddrAppend( ) CusAddrDestroy( )
CusInfo	<Event> 고객 정보 갱신 <Exception> 영역에 따라서 사원 정보내용이 달라짐	CusInfoAdd( ) CusInfoSend( ) CusInfoDelete( ) CusInfoCreate( ) CusInfoAppend( ) CusInfoDestroy( )
C2RAddr	<Event> 주소 변환 요청, 변환된 주소 전송 <Exception> 없음	C2RAddrSend( )
Docu	<Event> 발송될 내용 전달 <Exception> 없음	DocuSend( )

추출된 메소드의 의미는 <표 13>과 같다.

〈표 13〉 C2R 컴포넌트 메소드의 설명

메소드	메소드의 의미
CusNumSend( )	고객 번호 전송
CusNumReceive( )	고객 번호 수신
CusAddrSend( )	고객 주소 전송
CusInfoSend( )	고객 정보 전송
DocuSend( )	문서(고지서) 전송
C2RAddrSend( )	C2R로 주소 전송
CusAddrAdd( )	고객 주소에 새로운 내용 추가
CusAddrDelete( )	고객 주소 삭제(부분)
CusAddrCreate( )	고객 주소 생성
CusAddrAppend( )	고객 주소가 변경된 내용 추가
CusAddrDestroy( )	고객 주소 삭제(전체)
CusInfoAdd( )	고객 정보 새로운 내용 추가
CusInfoDelete( )	고객 정보 삭제(부분)
CusInfoCreate( )	고객 정보 생성
CusInfoAppend( )	고객 정보에 변경된 내용 추가
CusInfoDestroy( )	고객 정보 삭제(전체)

<표 13>에서 CusInfo에서 가변성의 요소를 찾을 수가 있는데, 그 이유는 현재 설계하고자하는 주소 컴포넌트를 다른 영역(예를 들면 ERP)에서 사용하려고 한다면, 사원의 정보를 추가해야 한다. 그 내용으로는 입사일자, 퇴사 일자,

개인정보(가족 관계) 등이 가변성 부분으로 존재하게 된다.

4.4 컴포넌트 식별과 인터페이스 추출

컴포넌트 식별과 인터페이스 추출 단계에서는 (그림 7), (그림 8)와 (그림 9)을 기반으로 하여 컴포넌트를 구성하는 인터페이스들을 명세 한다. 이와 같은 인터페이스 명세를 기반으로 컴포넌트를 구성한다. 인터페이스 추출은 <표 14> ~<표 18>과 같다.

〈표 14〉 C2R 고지서 발송 요청 인터페이스 명세

인터페이스 네임	인터페이스 추출
고지서 발송 요청	Interfrel : (IN) "고지서 발송 요청" [From] 고지서 발송 Preconditions : 고객 정보에서 고지서를 네트워크로 받지 않을 고객이 구별되어야 한다. Postconditions : 고객정보에서 고지서를 받을 고객이 설정되어야 한다. Input Parameter : 고객번호 Output Parameter : 고객번호 Use Case : Conversion address

〈표 15〉 C2R 고객 정보 조회 인터페이스 명세

인터페이스 네임	인터페이스 추출
고객 정보 조회	Interfrel : (OUT) "고객정보 조회" [To] C2R Preconditions : 고객선정이 완료되어야 한다. Postconditions : 선택된 고객의 정보가 저장되어야 한다. Input Parameter : 고객번호 Output Parameter : 고객번호, 고객 주소, 고객의 신상 명세 Use Case : Security, Address Recognition

〈표 16〉 C2R 고객 정보 update 인터페이스 명세

인터페이스 네임	인터페이스 추출
고객정보 update	Interfrel : (OUT) "고객정보 update", "고지서 발송" [To] C2R Preconditions : 조회한 고객의 정보가 식별되어야 한다. Postcoditions : 고지서가 발급될 고객 명단을 저장해야 한다. Input Parameter : 고객번호, 고객주소, 고객의 신상명세 Output Parameter : 고객번호, 고객주소, 고객의 신상명세 Use Case : Conversion address

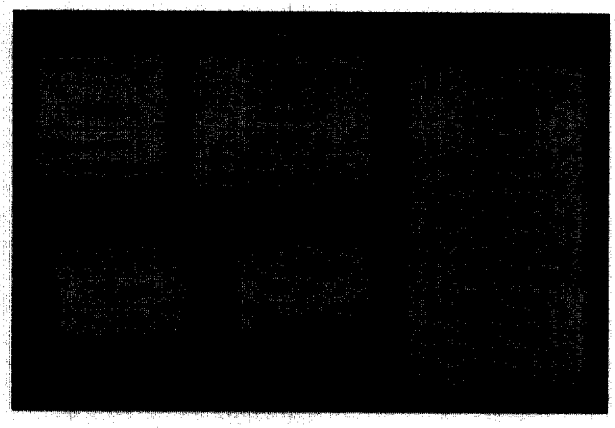
〈표 17〉 C2R 고지서 내용 전송 인터페이스 명세

인터페이스 네임	인터페이스 추출
고지서 내용 전송	Interfrel : (IN) "고지서 내용 전송" [From] 고지서 발송 Preconditions : 고지서가 발송될 고객의 정보가 확보되어야 한다. Postconditions : 발송될 위치에 공공성을 부여한다. Input Parameter : 고객번호, 고객주소, 고객의 신상명세, 고지서 내용 Output Parameter : 고객번호, 고객주소, 고객의 신상명세, 공공성이 부여된 주소, 고지서 내용 Use Case : Conversion address, Transmission document

〈표 18〉 R2C 발송될 내용 전달 인터페이스 명세

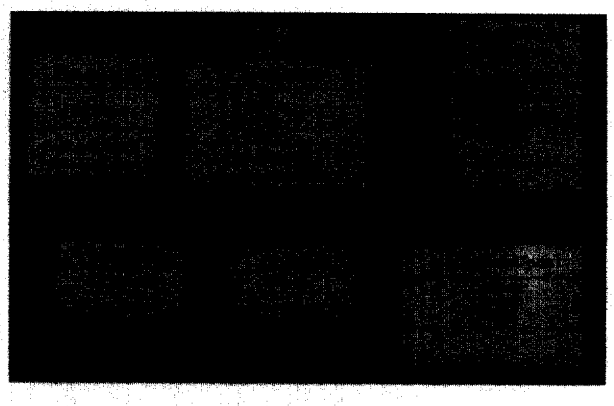
인터페이스 네임	인터페이스 추출
발송될 내용 전달	Interfrel : (OUT) "발송될 내용 전달" [To] R2C Preconditions : 고객위치의 공공성 부여와 전달될 내용이 완료되어야 한다. Postconditions : 고객이 내용을 전달받을 수 있는 환경이 구축되어야 한다. Input : 공공성이 부여된 주소, 고지서 내용 Output : 전자 우편 Use Case : Conversion address, Address Recognition

컴포넌트 인터페이스 추출이 완료된 후에, 컴포넌트에 존재하는 전반적인 인터페이스 구조를 작성한다. 컴포넌트는 다음과 같은 형태의 인터페이스를 갖는다.



(그림 10) C2R 컴포넌트 인터페이스 구조

(그림 10)에서는 예외 사항이 반영되지 않은 상태이다. 여기서 ERP 영역에 적용하기 위하여 가변성이 추가된 컴포넌트 인터페이스 구조는 (그림 11)와 같다.



(그림 11) 확장된 C2R 컴포넌트 인터페이스 구성도

(그림 11)에서 CusInfoUpda는 (그림 10)과 동일하다. 공용성과 가변성을 반영한 형태의 컴포넌트 구조를 유지함으로서 재사용성과 확장성을 제공할 수 있다.

추출된 인터페이스와 확장된 가변성의 메소드 설명은 <표 19>, <표 20>과 같다.

〈표 19〉 C2R 컴포넌트 인터페이스의 설명

인터페이스 네임	인터페이스 네임의 의미
DocSendReq	고지서 발송 요청, 주소 변환 요청
CusInfoRefe	고객정보조회
CusInfoUpda	고객정보update
AddrTransDeli	변환된 주소 전송
ContDeli	발송내용 전달

〈표 20〉 C2R 관점에서 ERP 시스템의 메소드

영역 \ 도메인	ERP 시스템
Addr	PrivAddr : 주소 PrivCertNum : 주민번호 Name : 이름 PrivEntDate : 입사일자 PrivRetDate : 퇴사일자 PrivInfo : 개인정보(가족관계등)

4.5 평 가

본 논문에서는 컴포넌트 재사용과 확장성을 제공하기 위하여 기능상의 불일치를 해결하고자 기본적으로 사용자의 요구사항을 수렴하여 정규화 한다. 그리고 요구사항을 기반으로 하여 공용성과 가변성을 분석한다. 분석된 자료는 컴포넌트를 구성하는 인터페이스로 만들어지고, 인터페이스는 오퍼레이션을 제공하여서 기능중심의 컴포넌트를 제공하게 된다. 이러한 기능중심의 컴포넌트는 서비스를 처리하기에 적합한 형태가 된다. 본 논문의 개발 방법을 적용한 컴포넌트는 다음과 같은 사항으로 평가할 수 있다.

〈표 21〉 공용성과 가변성을 적용한 컴포넌트 개발 방법의 특징

비교 항목	영역 중심의 컴포넌트 공용성과 가변성 분석
명세 단위	행위를 기반으로 한 기능 단위
공용성 분석	인터페이스의 오퍼레이션 단위로 추출한다.
가변성 분석	공용성 중심의 클래스 화된 구조를 기반으로 가변적인 오퍼레이션을 추출한다.
확장성	가변성 부분을 커스터마이징 할 수 있도록 스테레오 타입으로 정의한다.
재사용성	여러 도메인에 적용 가능하며 컴포넌트 재사용시의 기능상의 오류를 방지할 수 있다.

본 논문에서는 개발 영역 중심으로 컴포넌트 개발이 이루어진다. 도메인 중심으로 개발을 수행하면 영역 중심적인 특성을 가진 컴포넌트 개발에 좀더 적합하기 때문이다. 그러나 공용성과 가변성을 분석하는 시간이 할당되어야 하므로, 개발 기간이 길어질 수 있다. 그러나 공용성과 가변성의 분석에 의하여 사용자의 요구 사항의 변화하는 부분을 식별하여서 융통성 있게 변화를 수용할 수 있다.

5. 결론 및 향후 연구과제

본 논문에서는 재사용을 위한 컴포넌트의 설계 시에 공용성과 가변성을 추출하기 위해서 요구사항 추출, 공용성과 가변성의 식별, 컴포넌트 식별과 인터페이스 추출의 단계를 제시하였다. 이와 같은 단계를 제시한 것은 컴포넌트의 동작의 구성을 행위에서 유추하여 상태 조건에 따른 공용성과 가변성을 분석하고자 함이다. 공용성과 가변성을 구분하여 컴포넌트를 설계함으로서, 다른 도메인이나 새로운 요구사항의 변화를 수용해서 컴포넌트의 재사용성과 확장성을 제공받을 수 있다.

향후 본 논문에서는 제시된 컴포넌트 공용성과 가변성을 추출하는 자동화 툴의 개발과 이를 시각화할 수 있도록 하기 위한 연구를 요구된다. 이와 같은 연구는 컴포넌트 개발 시에 가변적인 요소를 빠른 시간 내에 적은 노력으로 추출하여서 컴포넌트 설계의 정확성과 빠른 생산성을 얻을 수 있을 것이다.

참 고 문 헌

[1] 이경환, "최신 소프트웨어 공학", 청문각, 1998.

[2] Darren Govoni, "Java Application Framework," Wiley Computing Publishing, 1999.

[3] Patrick Steyaert. "Object-Oriented Framework," Vrije Universiteit Brussel. tech. report, "http://progwww.vub.ac.be/prog/pools/Frameworks/Frame works.html," 1998.

[4] Desmond F. D'Souza, Alan C. Wills. "Objects, Components and Frameworks with UML," Addison Wesley, 1997.

[5] Chris Frye, "Understanding Components," Andersen Consulting Kowledge Xchange, 1998.

[6] Grady Booch, "a primary contributor to the UML and Rational Software's Objectory process," December, 1997.

[7] David M. Weiss, Chi Tau Robert Lai, "Software Produce-Line Engineering," Addison-Wesley, 1999.

[8] Cleaveland, J. Craig. "Building Application Generators," IEEE Soft, July, 1988.

[9] 이은서, "컴포넌트 재사용을 위한 공통성과 가변성 추출에 관한 연구", 중앙대학교 석사학위논문, 2001.

[10] 전진옥, "컴포넌트 기반 소프트웨어 개발기술 동향", Software Technology Conference Korea'99, 한국정보산업연합회, 1999.

[11] Lana Kuzmanov, "CBSE in the Realm of Computing /Information System Life Cycle," ICSE proceedings, 1999.

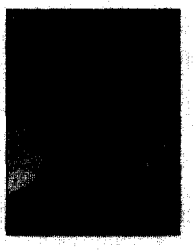
[12] COCOMOII 2000, "object.cau.ac.kr/selab/index.html," 중앙대학교 selab semina notes, 2001.

[13] Rational Software, Microsoft, "Hewlett-Packard, UML Notation Guide UML Summary," 1997.

[14] Stefan Tai, "A Connector Model for Object-Oriented Component Integration," International Workshop, 1998.

[15] Jan Bosch, PO Bengtsson, Component Evolution in Product-Line Architectures, International Workshop, 1999.

[16] Pearl Brereton, "Evolution of Component-Based Systems," International Workshop, 1999.



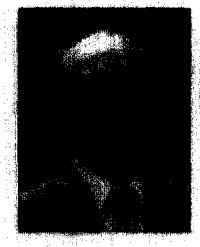
이 은 서

e-mail : esleee@object.cau.ac.kr

1999년 중앙대학교 컴퓨터공학과(석사)

2001년~현재 중앙대학교 컴퓨터공학과 (박사과정)

관심분야 : CBD, Formal method, Quality model, SPI(Defect Analysis)



이 경 환

e-mail : kwlee@object.cau.ac.kr

1964년 중앙대학교 이과대학 수학과 졸업 (학사)

1966년 중앙대학교 이과대학 수학과 졸업 (석사)

1980년 중앙대학교 대학원 수학과 졸업 (박사)

1992년~현재 ISO/SC7/WG10 SPICE 한국 위원장

1992년~현재 한국표준 소프트웨어 공학 (SC7) 위원

1992년~JCSE '92 Conference Chairman 서울

1993년~1995년 중앙대학교 공과대학 학장

1994년~1995년 중앙대학교 정보산업대학원 원장

1998년~2000년 한국 온라인 가상 대학 운영 위원장

1998년~2000년 중앙대학교 정보통신연구소 소장

1999년~2000년 한국 정보과학회 회장

2000년~2001년 중앙대학교 총무처장

1982년~1983년 미국 AUBURN 대학교 객원교수

1986년~1986년 독일 BONN 대학교 객원교수

2001년~2002년 미국 USC 대학교 객원교수

관심분야 : CBD Architecture, SPI(Defect Analysis), MBASE/CeBASE