특집

컴포넌트 기반 개발(CBD)을 위한 도메인 공학

문 미 경[1] 염 근 혁[2]

목 차

1. 서 론

요즘 기업 환경의 시스템은 지속적으로 변화하는 외부환경에 잘 적응하며 그때그때 시스템 자체가 변하거나 확장하는데 있어서 뛰어난 능력을 발휘해야 한다. 특히, 분산환경과 large-scale, n-tier, web 기반의 e-Commerce로의 확장이 용이해야 한다. 이에 대한 해결책으로 소프트웨어 재사용과 컴포넌트 패러다임이 공통적인 접근 방식으로 인식되고 있다. 컴포넌트와 재사용의 개념을 함께 가지는 컴포넌트 기반 소프트웨어 개발 방법은 소프트웨어 개발 시간과 비용을 줄이고, 생산성을 향상시키는 등의 장점을 가진다.

그러나 하드웨어의 재사용성과는 달리 소프트웨어는 많은 다양성을 가지고 있어 이를 구성하는 컴포넌트를 식별하고 표준화시키는 것은 어려운 작업이다. 뿐만 아니라, 컴포넌트를 여러 다른 환경에서 재사용 가능하도록 유연성(flexibility)있게 구현하기 위해서는 추가적인 비용이 요구된다. 이러한 어려운 작업에 대한 추가적인 노력과 비용은 컴포넌트가 충분히 재사용 될 수 있는 경우에만 지불할 만한 것이 된다.

그럼 과연, 어떻게 컴포넌트가 충분히 재사용 될 수 있을 지 예측할 수 있을까? 또한, 여러 다양성에도 불구하고 어떻게 컴포넌트를 체계적으로 재사용할 수 있을까?

이러한 목적을 위해 컴포넌트가 놓일 도메인에 대한 정확한 분석이 요구된다. 소프트웨어 시스템 세계에서 '도메인'이란 용어는 특정 분야에서 공통의 기능을 가지는 일련의 시스템의 집합을 나타내는 것이다. 특정 도메인 내에서 공통성과 다양성을 찾아내고 이들을 다시 재사용 될 수 있는 형태로 만들 수 있다면 그 도메인 내에 속하는 또 다른 새로운 시스템을 만들 때 이를 유용한 정보로 이용할 수 있게 된다. 즉, 관련된 시스템들을 일관성 있게 관리함으로써 증명된 핵심 자산들(core asset)을 확보할 수 있고, 개발의 정합성을 이룰 수 있게 된다. 이에 대한 연구들이 도메인 분석 및 도메인 공학에서 이루어지고 있다.

도메인 공학 프로세스에서 나오는 산출물들은 새로운 시스템 개발에 재사용될 수 있는 형태를 갖

1) 부산대학교 컴퓨터공학과 박사과정 수료
2) 부산대학교 컴퓨터공학과 부교수
부산대학교 컴퓨터 및 정보통신 연구소 연구원

춤으로써 효율적인 시스템 개발을 지원해 줄 수 있다. 그러므로 컴포넌트 기반 소프트웨어 개발 프로세스를 통해 시스템이 개발될 시에는 이에 적합하게 맞추어진 도메인 공학 프로세스가 필요하다. 예를 들어, 컴포넌트 기반의 도메인 공학에서는 컴포넌트 인식, 컴포넌트간의 구성(configuration), 컴포넌트 인터페이스, 컴포넌트 명세서 등에 대한 정보가 산출되도록 요구된다. 본 글에서는 도메인 분석 및 도메인 공학에 대한 관련 연구들을 살펴본다. 또한 이를 바탕으로, 컴포넌트 기반 도메인 공학에서 고려되어야 할 이슈들을 프로세스와 산출물들을 중심으로 논의한다.

2. 관련 연구(Related works)

2.1 도메인 분석 및 도메인 공학(Domain Analysis and Domain Engineering)

'도메인 분석'과 '도메인 공학'이란 용어는 종종 서로 일치하지는 않지만 호완성 있게 사용된다. 비록 '도메인 분석'이란 용어가 '도메인 공학'보다 앞서 나왔지만, '도메인 공학'이 좀 더 포괄적인 용어다[1].

도메인 분석은 시스템을 효율적으로 개발하고 유지하기 위해 체계적이고 대규모적인 재사용을 지원하는데 초점을 두고 있다. 이는 도메인 내에 있는 시스템에서 공통성과 다양성을 찾아냄으로써 이루어지며, 도메인 모델로 그 결과를 산출하게 된다. 도메인 공학에서는 도메인 분석, 아키텍쳐 개발과 재사용 가능한 컴포넌트 개발 단계들을 포함하며 이 과정에서 도메인 모델, 도메인 아키텍쳐, 재사용 도메인 컴포넌트 등을 산출한다. 소프트웨어 개발자들이 새로운 시스템을 개발하는 데 재사용 할 수 있는 부품들 —모델, 코드, 문서, 테스트 플랜 등— 을 식별, 개발, 보급하는 것이 바로 도메인 공학의 목적이다[2].

2.2 관련 연구

2.2.1 Feature를 이용한 도메인 분석

1990년대 초 이후 시스템의 집합 중에서 주도적인 또는 독특한 피쳐(feature)를 인식하는 것에 기초해 도메인을 분석하는 방법들이 나왔다. 대표적인 방법으로는 Feature Oriented Domain Analysis(FODA)[3]가 있다.

FODA에서 'feature'라는 것은 구현되고 테스트되고 배포, 유지되어져야 하는 기능적 추상화를 뜻하는 것으로 요구사항이나 기능들을 의미한다. 이러한 피쳐들로 이루어진 피쳐 그래프는 도메인 내에서 공통성과 다양성을 추출하기 위한 도구로서 사용된다. FODA는 후에 소프트웨어 설계와 구현 단계 부분까지 확장되어 FORM(Feature Oriented Reused Method) 방법론[4]으로 발전한다. FORM은 재사용 가능한 아키텍쳐와 컴포넌트 개발, 그리고 도메인 공학에서 생성된 결과들을 이용한 어플리케이션 개발을 지원한다.

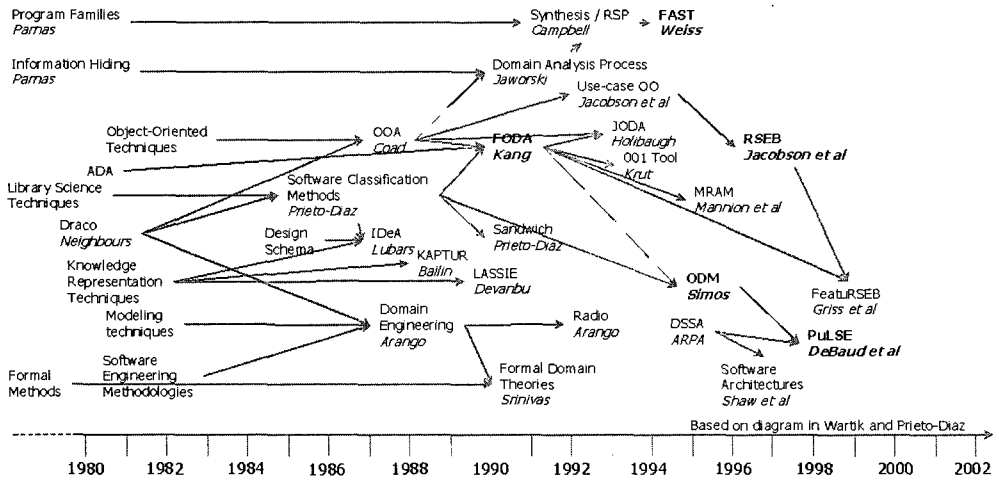
1990년대 후반 FODA의 피쳐 모델링 방법에 Reuse-Driven Software Engineering Business(RSEB)의 프로세스를 통합한 FeatuRSEB(the Featured RSEB)[5] 방법이 나왔다. 이 방법은 피쳐 모델을 만드는 활동과 병행하여 유즈케이스 모델을 만든다. 이는 도메인 내의 어플리케이션들의 일반적인 능력들을 캡쳐하기 위하여 객체 지향 요구사항 분석 방법인 유즈케이스 모델링 방법을 추가한 것이 특징이다. 이 유즈케이스 모델은 도메인 내의 각각의 유즈케이스 모델을 합하여 도메인 유즈케이스 모델을 만들어 낸다.

2000년도 이후에 소프트웨어 프로덕트 라인 개발에서 다양성을 표현하기 위한 연구에 피쳐가 사용되었다[6]. 여기서 사용되는 피쳐는 프로덕트들 사이에 나타날 수 있는 차이성을 기술하기 위한 용어로서 사용된다. 이는 소프트웨어 프로덕트 라인의 목적이 생산되는 시스템의 변경을 허용하

는 것이기 때문에 아키텍쳐들 사이에 동일한 요소들보다 변할 수 있는 요소들을 인식하는 것에 중요성을 두고 있기 때문이다.
이처럼 도메인을 분석하는 분야에서 피쳐는 다방면에서 사용되어졌다. 그러나 피쳐를 분류하는 과정에 있어서 구체적이고 객관적인 기준을 따르기보다는 단지, 도메인 분석가의 경험이나 직관에 의존하고 있었다[7].

2.2.2 그 외 도메인 공학 방법론

ODM (Organization Domain Modeling)[8]은 형식적이고 반복적인 도메인 공학 방법을 제공하며, 크게 Plan Domain단계, Model Domain 단계와 Engineer Asset Base단계로 진행된다. 이 방법은 프로젝트와 도메인에 대한 관심을 두고 있고, 다양한 관점과 경험을 가지고 있으며, 다양한 용어를 사용하는 stakeholder에 초점을 두었다. 또한 도메인 예(examples)의 명시적인 집합인 exemplar를 기반으로 모델링하였다. 그러나 ODM은 도메인 아키텍쳐와 구현을 개발하기 위한 방법에 대해서는 구체적으로 규정하지 않았다. 또한 하위 레벨 도메인 객체를 분석하는데 중점을 잘못 두고 있다는 평가가 있기도 하다[5].
ODM을 기반으로 해서 도메인 아키텍쳐를 개발하고 개발된 도메인 자산들을 어플리케이션 개발에 적용시키는 단계까지 확장한 DAGAR 프로세스[2]가 있다. DAGAR은 Ada에 기반을 두고 도메인 자산(asset)을 구현하기 때문에 도메인 아키텍쳐를 만들기 위한 2가지 기본 요소도 Ada의 용어인 '영역(realm)'과 '컴포넌트(component)'로 제시한다. 영역(realm)은 도메인 자산에 의해 제공되어지는 핵심 서비스를 계층화된 도메인 아키텍쳐 내에서 나타내는 것이다. 그러나 DAGAR은 하나의 특정 프로그램 언어 Ada를 기반으로 하고 있어 그 사용에 한계가 있다. 독일의 Frauenhofer IESE(Institut fr Experimentelles Software Engineering)에서 개발한 KobrA[10]는 크게 프레임웍 엔지니어링과 애플리케이션 엔지니어링으로 구성된다. 프레임웍 엔지니어링 단계에서는 시스템 패밀리의 공통적인 부분은 물론 가변성을 추가하여 프레임웍을 만들게 되고, 애플리케이션 엔지니어링 단계에서는 이러한 가변성을 제거하여 특정 멤버의 요구사항에 맞는 애플리케이션을 개발하게 된다. 그러



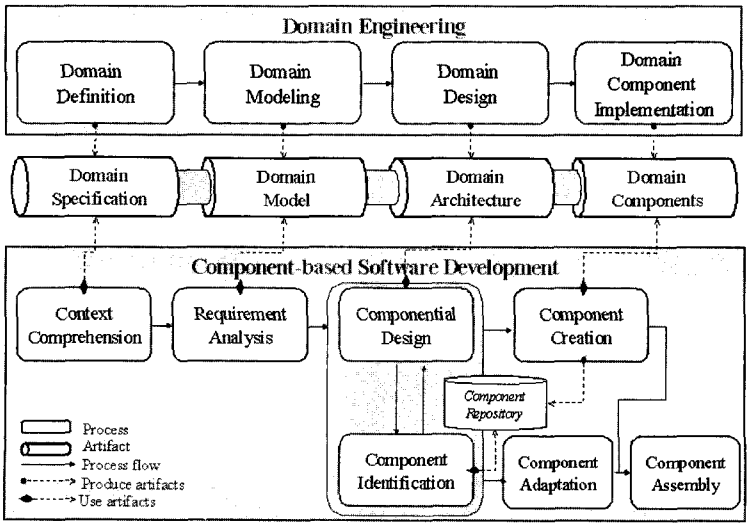
(그림 1)도메인 공학 관련 연구 계보[9]

나 IESE에서 개발한 KobrA 방법은 재사용 가능한 자산을 개발하고 그 자산을 기반으로 프로덕트를 개발하는 프로세스와 자산의 표현 형태에 대해 구체적으로 제시하고 있으나, 아키텍쳐의 중요성에 대한 인식이 부족하여 단순히 컴포넌트를 기반으로 전체 프로세스가 진행되고 있다.
(그림 1)은 도메인 분석 및 도메인 공학 관련 연구들의 계보를 보여준다.

3. 컴포넌트 기반 도메인 공학

3.1 컴포넌트 기반 도메인 공학 프로세스

컴포넌트 기반 도메인 공학은 컴포넌트 기반 소프트웨어 개발 프로세스에 대해서 유용한 정보들을 산출해야 한다. 이를 위해, 먼저 CBSD 프로세스를 정의하고 각 단계에서 요구되는 재사용 정보를 인식한다. 이를 바탕으로 도메인 분석과 설계 프로세스가 정립이 되고 CBSD 프로세스와 병행해서 진행된다. 이들 간의 프로세스는 각각의 산출물을 중심으로 상호 작용하며, 독립된 두 개의 프로세스는 선순환 관계를 이루며 더 이상 개별적이지 않은 하나의 성숙된 프로세스가 된다.
컴포넌트 기반 소프트웨어 개발 프로세스는 먼저, 개발할 어플리케이션이 놓일 문맥을 이해하고 요구사항을 분석하는 과정을 거친다. 시스템의 전체 구조와 구성 요소들을 식별해주고 그들의 동적 행위들을 결정해주며 컴포넌트들의 계층적 상태, 시스템의 수평적 분할(horizontal partitioning)과 수직적 분할(vertical partitioning) 등을 보여주는 컴포넌트화 설계(Componential design) 단계를 거친 후, 이를 통해 요구사항에 맞는 컴포넌트를 인식하고 어플리케이션의 특성에 맞게 컴포넌트를 적응시킨다. 이 과정에서 인식되어진 컴포넌트가 존재하지 않을 경우에는 컴포넌트를 생성한다. 마지막으로 컴포넌트를 조립하여 어플리케이션을 만들게 된다.
컴포넌트 기반 도메인 공학의 목적은 컴포넌트 기반 소프트웨어 개발을 지원하는 것이므로 프로세스가 이에 의존적이게 된다. 도메인 공학프로세스는 먼저, 도메인 정의 단계에서는 도메인의 목적을 정하고 도메인의 범위를 확정한다. 도메인 모델링 단계에서는 도메인 내에 존재하는 여러 어플



(그림 2)컴포넌트 기반 도메인 공학 프로세스

리케이션의 요구사항들을 추출하여 모델링 시키게 된다. 이때 요구사항들의 공통성과 다양성을 분리하여 구분시킨다. 이를 바탕으로 도메인 컴포넌트를 인식하게 되고 도메인 아키텍쳐를 만든다. 이 과정에서 생성되는 산출물들은 서로의 연관성을 유지하며 저장된다. 이는 컴포넌트 기반 소프트웨어 개발 시 유용한 정보로써 재사용 된다. (그림 2)는 컴포넌트 기반 소프트웨어 개발 프로세스와 이를 지원하는 도메인 공학 프로세스를 나타내며 이 두 프로세스간의 상호작용도 함께 보여준다.

3.2 도메인 산출물

3.2.1 도메인 요구사항

도메인 요구사항들은 도메인 내의 여러 시스템의 요구사항들에 관심을 두고 분석해야 함으로 산출되는 형태와 수가 일반 시스템 요구사항보다 훨씬 복잡하다. 그러나 도메인 요구사항은 도메인에 속하는 새로운 어플리케이션 개발 시뿐만 아니라, 다음 번 도메인 재분석 시나 하위 도메인 요구사항 분석 시에도 재사용 될 수 있기 때문에 그 형태를 일반화(generalization)시킬 필요가 있다. 도메인 요구사항 일반화가 의미하는 바는 두 가지다. 첫째, 도메인 요구사항의 속성 — 공통성, 다양성, 선택성 — 을 객관적으로 구분 짓는 일이다. 여기서 중요한 의미를 가지는 '객관적 구분'은 과거의 경험 — 즉, 레거시 시스템의 요구사항들 — 에서 얻은 자료를 수집하고 이를 바탕으로 평가하여 지표(indicator)를 만듦으로써 그 방법을 찾을 수 있다. 둘째, 도메인 요구사항들을 속성에 따라 그 수와 형태를 줄이는 것이다. 이는 나누어진 속성에 따라 도메인 요구사항들을 일정 수준 추상화시키는 방법을 사용할 수 있다.

일반화된 도메인 요구사항들은 지금까지 도메인 전문가의 경험과 본능적인 감각(heuristic)에만 의존해서 인식하게 되는 공통성과 다양성의 추출에 좀 더 객관화 된 정보로서 사용할 수 있게 되고, 이와 더불어 도메인 특성의 복잡성을 줄임으로써 재사용의 기회를 늘이게 된다.

3.2.2 도메인 모델

도메인 모델은 도메인 분석 과정의 결과물로서 도메인의 모든 기능적인 요구사항들을 모델링 한 것이다. 이를 위해 유즈케이스 모델링 기법을 이용할 수 있다. 유즈케이스는 시스템이 해야 할 행동을 가시화하고, 명세화하고, 구축하고, 문서화하는데 유용하게 사용할 수 있는 방법이다[11]. 도메인 모델을 산출하기 위해 들어올 수 있는 정보는 일반화 된 도메인 요구사항들과 기존의 레거시 시스템에 대한 분석 모델들이다. 도메인 모델로서 산출되는 요소에는 도메인 유즈케이스 다이어그램, 도메인 유즈케이스 서술서, 도메인 타입 모델, 도메인 시퀀스 다이어그램 등이 있다. 도메인 모델에서 명시되어야 하는 공통성, 다양성 속성들은 기존의 자료에 대한 수집으로 만들어진 지표와, 이전 단계의 도메인 요구사항에 의해 구분되어질 수 있다. 이러한 과정을 우리는 '도메인 모델에 대한 일반화 과정'이라고 부를 수 있고, 이를 통해 도메인 모델을 추상화시킴으로써 재사용 형태로 만들 수 있다.

3.2.3 도메인 컴포넌트

도메인 컴포넌트는 소프트웨어 개발 시 바로 배치(deploy)될 수 있는 물리적 컴포넌트와는 달리, 플랫폼 독립적인 논리적 수준의 서비스 중심 단위 패키지로 정의한다. 이는 서로 다른 단일 어플리케이션 개발 시 시스템의 요구사항으로부터 서로 다른 granularity의 배치 가능한 컴포넌트가 추출될 수 있는데, 이러한 컴포넌트에 대한 granularity의 가변성을 서비스 기반으로 추출된 도메인 컴포넌트를 통해 제공해 줄 수 있게 된다. 그러므로 도메인 컴포넌트는 비즈니스 프로세스

에 대한 기능성을 중심으로 분석된 도메인 유즈케이스를 기반으로 추출할 수 있다. 또한 도메인 컴포넌트는 도메인 모델에서 표현된 공통성과 다양성, 선택적 속성들을 반영하도록 추출되어져야 한다.

3.2.4 도메인 아키텍쳐

도메인 컴포넌트들 간의 관계를 인식하고 표현하여 도메인 아키텍쳐를 생성한다. 도메인 아키텍쳐에서는 논리적인 측면에서의 컴포넌트간의 상호 작용 모습과 전체적인 컴포넌트의 모습들을 표현하며 도메인 컴포넌트에 대한 명세를 보여준다. 도메인 아키텍쳐는 특정 개발 기술이나 개발 도구, 개발 언어에 독립적이어야 한다. 또한 도메인 아키텍쳐의 모습에는 요구사항 분석 단계로부터 생성되어 각 단계를 거치면서 정제, 유지되어 온 공통성, 다양성, 선택적 속성을 반영하여야 한다. 이러한 모습은 컴포넌트 기반 소프트웨어 개발 시 도메인 아키텍쳐에 표현되어 있는 컴포넌트 속성에 따라 아키텍쳐의 일부분이 분리, 대치될 수 있도록 해 주기 때문에 결국, 유연한(malleable) 아키텍쳐를 생성할 수 있도록 해준다. 이를 위해, 도메인의 복잡성을 줄이고 호환성을 증진시키는 고수준의 추상화(abstract)를 이용한다.

4. 결 론

소프트웨어 재사용은 time-to-market을 줄이기 위한 한 가지 방법으로 여겨지며, 잠재적으로 소프트웨어 개발의 생산성과 품질을 향상시키는 해결책이 된다. 재사용은 지난 십여 년 동안 계속 활발히 연구되고 있는 분야이며, 이와 더불어 재사용을 좀 더 체계적으로 지원하기 위한 방법으로 도메인 분석과 도메인 공학에 대한 연구가 진행되어 오고 있다. 그러나, 재사용에 있어서 가장 괄목할 만한 성과는 이전의 프로시저나 객체와 같은 소규모 단위 재사용의 한계를 넘어 프레임 워크 단위의 재사용을 가능케 하는 컴포넌트의 등장이다. 현재 국내외적으로 컴포넌트 기반 개발 기술에 대한 연구가 활발하게 진행되고 있다. 이러한 점에 발맞추어 도메인 공학 방법도 CBD 개발 환경에서 재사용을 잘 지원 할 수 있도록 그에 맞추어질(tailor) 필요가 있다. 즉, 기존의 연구에서 얻을 수 없었던, 컴포넌트를 인식하고 그들 간의 관계를 고려하여 컴포넌트를 조립해 내는 과정에 대한 정보들을 컴포넌트 기반 도메인 공학을 통해 얻을 수 있도록 할 필요가 있다. 컴포넌트 기반 도메인 공학에서 산출되는 주요 결과물에는 도메인 요구사항, 도메인 모델, 도메인 컴포넌트, 도메인 아키텍쳐 등이 있고, 여기에는 도메인의 공통성 요소, 다양성 요소가 적절한 형식으로 명확하게 표현되어져야 한다. 컴포넌트 기반 재사용의 흐름에 병행하여 앞으로 이들에 대한 계속적인 연구가 필요하다.

참고문헌

[1] SEI in Carnegie Mellon University, "Domain Engineering and Domain Analysis", URL:http://www.sei.cmu.edu/str/descriptions/deda_body.html

[2] Klingler, C.D., "DAGAR: A Process for Domain Architecture Definition and Asset Implementation." , In: Proceedings of ACM TriAda, 1996.

[3] Kang, K. C., "Feature-Oriented Domain Analysis for Software Reuse", Joint Conference on Software Engineering, pp. 389-395, 1993.

[4] Kang, K. C., Kim, S., Lee J.,and Kim, K., "FORM: A Feature-Oriented Reuse Method with Domain Specific Reference

Architectures", Pohang University of Science and Technology(POSTECH), 1998.

[5] Griss, M. L., Favaro, J., and d' Alessandro, M., "Integrating Feature Modeling with the RSEB" , in Proceedings of 5th International Conference on Software Reuse, Victoria Canada, June, IEEE, pp. 76-85, 1998.

[6] van Gurp, J., Bosch, J., and Svahnberg, M., "On the notion of variability in software product lines ", Proceedings on Working IEEE/IFIP Conference on Software Architecture, pp. 45 -54, 2001.

[7] Frakes, W., Prieto-Diaz, R., and Fox, C., "DARE-COTS: A Domain Analysis Support Tool" , Proceedings on XVII International Conference of the Chilean Computer Science Society (Valparaiso, Nov, 1997), pp. 73 -77, 1997.

[8] Creps D., Klingler, C., Levine, L., and Allemang, D., "Organization Domain Modeling(ODM) Guidebook Version 2.0" , Software Technology for Adaptable, Reliable Systems (STARS), 1996.

[9] Lewis, O., Buchanan, W.J., "Performance Issues of Variability Design in Embedded System Application Families", Distributed Systems and Mobile Agents Research Page, Resources, chapter 2

[10] Atkinson, C., Bayer, J., Bunse, C., etc, "Component-Based Product Line Engineering with UML", Addison Wesley, 2001.

[11] Booch, G., Rumbaugh, J.,and Jacobson, I., "The Unified Modeling Language User Guide", Addison-Wesley, January 1999.

저자약력

문 미 경

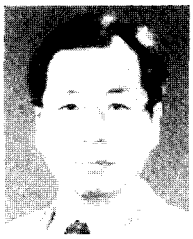
1990년 이화 여자 대학교 전자계산학과 (이학사)

1992년 이화 여자 대학교 전자계산학과 (이학석사)

2002년 부산대학교 컴퓨터공학과 박사과정 수료

관심분야 : 도메인 공학, 소프트웨어 아키텍쳐, 컴포넌트 기반 소프트웨어 개발

이 메 일 : mkmoon@pusan.ac.kr



염 근 혁

1985년 서울대학교 계산통계학과(학사)

1992년 Univ. of Florida 컴퓨터공학과(석사)

1995년 Univ. of Florida 컴퓨터공학과(박사)

1985년 1월 ~ 1988년 2월 금성반도체 컴퓨터연구실 연구원

1988년 3월 ~ 1990년 6월 금성사 정보기기연구소 주임연구원

1995년 9월 ~ 1996년 8월 삼성SDS 정보기술연구소 책임연구원

1996년 8월 ~ 현 재 부산대학교 컴퓨터공학과 부교수

부산대학교 컴퓨터 및 정보통신 연구소 연구원

관심분야 : 컴포넌트 기반 소프트웨어 개발, 도메인 공학, 소프트웨어 아키텍처, 객체지향 소프트웨어 개발방법론, 요구 검증, 분산 컴포넌트, 소프트웨어 재사용 등임.

이 메 일 : yeom@pusan.ac.kr