규칙기반 분석 패턴과 UML Components을 사용한 가변적인 비즈니스 컴포넌트 개발과 적용 사례

이 용 환[†] · 민 덕 기[††]

요 약

컴포넌트 확장성 및 재사용성을 향상시키기 위해서는 분석단계에서부터 가변적인 것들을 규칙기반으로 분석해서 컴포넌트로 식별할 필요가 있다. 본 논문에서는 시스템 외부 이벤트에 대해 트랜잭션 처리를 규칙기반으로 처리해야 하는 도메인 상에서 객체 기반의 중요 개념을 규칙기반으로 효과적으로 추출해 UML Components 개발 프로세스 상에서 가변적인 컴포넌트를 개발할 수 있는 규칙 기반 분석 패턴을 제시한다. 업무 지식이나 경험이 다른 많은 분석가들이 서브 시스템 많은 규칙 기반의 가변적인 복잡한 비즈니스 업무 분석 시 제안한 분석 패턴을 사용할 경우 분석 산출물의 일관성이나 가독성을 좋게 하며 또한 UML Components 방법론상에서 효과적으로 가변적인 비즈니스 컴포넌트들을 식별할 수 있다. 이러한 분석 패턴의 타당성을 증명하기 위해 본 논문에서는 가변적인 규칙기반으로 업무를 처리하는 은행 수신과 수출입 업무 도메인에 적용한 결과 패턴에서 제시한 중요 개념을 기반으로 거의 유사한 비즈니스 개념 모델을 도출할 수 있었으며 또한 이들 중요 개념을 기반으로 UML Components 개발 프로세스 상에서 가변적인 비즈니스 컴포넌트를 효과적으로 식별할 수 있었다.

키워드 : 분석패턴, 객체지향 모델링, 개념적 모델링, 컴포넌트 기반 개발 프로세스, 규칙 기반

A Variable Business Component Development and Case Study Using a Rule Based Analysis Pattern and UML Components

Lee Yong Hwan[†] · Min Dug Ki[††]

ABSTRACT

In order to increase extensibility and reusability of business components, the variable things need to be analyzed from the analysis phase and identified as components. In this paper, we propose a rule-based analysis pattern, which can effectively extract object-based main concepts from a variable business process in the analysis phase and identify a variable business component by applying the pattern to the UML Components development process. It can make analysis artifacts consistent and readable for analysts with different level of knowledge and experience to apply the pattern to analysis of rule-based variable business processes. And also, variable business components can be easily identified by applying the pattern to the UML Components development process. In order to prove the feasibility of the pattern, we have applied the pattern the deposit and import/export subsystem of the banking domain. According to our experience, we can make the same business conceptual models between the deposit and import/export subsystem due to the main concepts suggested by the pattern and effectively identify a variable business components in the UML Components development process.

Key Words : Analysis Pattern, Object-Oriented Modeling, Conceptual Modeling, CBD Development Process, Rule-Based, UML Components

1. 서 론

비즈니스 전략이나 정책 변동에 따른 가변적인 상황에서 컴포넌트의 확장성 및 재사용성을 위해서는 컴포넌트의 독립성을 통한 변경의 파급효과를 최소화 하도록 해야 한다. 하지만 이러한 컴포넌트 독립성 만으로는 효과가 미비하며 컴포넌트 확장성이나 재사용성을 강화하기 위해서는 비즈니스 어플리케이션 개발 과정에서 발견된 가변성을 별도의 규칙으로 정의해서 규칙 컴포넌트와 같은 별도의 컴포넌트 형태로 개발해야 한다[1].

개발프로세스적인 측면에서 많은 기존 소프트웨어 개발 프로세스는 주로 분석 단계(Analysis Phases)를 피상적으로 다루고 있으며 설계나 구현 단계를 강조하고 있어 분석 단계에서 요구 사항이 충분히 반영되어 있는지 검토하지 않고 설계나 구현으로 넘어간다[2, 3]. 또한 많은 업무 분석가들이

† 정 회 원 : 건국대학교 연구교수
†† 종신회원 : 건국대학교 교수
논문접수 : 2005년 12월 15일, 심사완료 : 2006년 8월 25일

서로 다른 서브시스템들에 대해 분석 모델을 수행하는 경우 스타일과 추상화 수준이 다른 분석 산출물(예: 클래스 다이어그램 형태의 비즈니스 객체 모델)을 작성하게 된다. 시스템 분석을 위한 틀이 없이 분석, 설계 그리고 구현을 수행했을 때 각각 다른 수준의 산출물을 작성하게 되며 이는 산출물 가독성을 저하시키고 개발 생산성을 떨어뜨린다[4]. 특히 분석 단계에서의 서로 다른 개념들은 프로젝트 위험(Risk)을 증가시킬 수 있다.

최근 소프트웨어 엔지니어 분야 연구자들은 특정 도메인별 특성들에 대한 요구사항이나 개념들을 식별하기 위한 분석 패턴들을 연구하고 있다[4, 5, 6, 7, 8]. 하지만 외부 이벤트에 대해 내부 프로세스를 가변성을 수용할 수 있는 규칙 기반으로 처리하는 비즈니스 어플리케이션을 위한 분석 패턴은 존재하지 않는다. 또한 기존의 분석 패턴들에 대한 연구들은 분석 단계에 대해서만 강조를 하고 있지 분석 패턴 상의 주요 개념들이 컴포넌트 개발 프로세스 상의 분석, 설계 그리고 구현까지 어떻게 체계적으로 연결되어 적용되는지에 대한 연구는 없다.

본 논문에서는 시스템 외부 이벤트에 대해 비즈니스 어플리케이션 트랜잭션을 규칙 기반으로 처리해야 하는 도메인에서 객체 기반의 중요 개념을 추출해서 UML Components 개발 프로세스[9]상에서 효과적으로 가변적인 컴포넌트를 추출할 수 있는 규칙 기반 분석 패턴을 소개하고 은행의 수신과 수출입 업무에 적용한 예제를 제시한다. 제안한 규칙 기반 분석 패턴은 가변적인 규칙 기반으로 업무를 수행하는 도메인에서 업무 지식이나 경험이 상이한 많은 분석가들이 서로 다른 서브 시스템 분석 시 분석해서 UML Components 개발 프로세스 상에서 컴포넌트를 추출할 때 산출물의 일관성이나 가독성을 좋게 하며 효과적인 컴포넌트 식별이 가능하다. 또한 패턴이 제시한 몇 가지 핵심 개념을 사용해 분석 단계에서부터 가변적인 부분들을 별도로 관리해서 컴포넌트로 식별 함으로서 컴포넌트 재사용성이나 적응성을 높일 수 있다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 분석 패턴에 대한 관련 연구를 제시하고 3장에서는 본 논문에서 제시한 규칙 기반 분석 패턴에 대해서 기술한다. 다음으로 4장에서는 규칙 기반 분석 패턴이 UML Components 개발 프로세스상에 적용되는 방법을 은행 수신업무 적용 사례를 통해 기술하고 5장에서는 제시한 분석 패턴이 가지는 장점에 대한 타당성 설명을 4장의 예제를 통해 제시하며 마지막으로 6장에서 결론을 기술한다

2. 관련연구

2.1 분석 패턴 배경 및 관련 분석패턴

분석단계는 문제영역을 이해하는 것이 초점이기 때문에 문제영역을 잘 반영한 개념모델을 작성해야 한다. 이러한 개념모델에는 단지 요구사항 나열만이 아니라 요구사항 달성을 위한 내부 원리까지 포함한다[10]. 모델에 있어 옳고 그름은 없으며 모델 평가는 얼마나 더 유용하느냐에 의해서 결정된다. 분석 패턴은 실제 특정 도메인에 유용하고 다른 곳에서도 유용할 것으로 생각되는 아이디어로서 개념모델 작성 시 문제 영역 이해, 특정 도메인 상의 문제 해결을 위한 내부 원리 이해와 같은 목적을 위해 분석틀을 제시한다[11].

소프트웨어 개발 프로세스 초기 단계에서 패턴을 사용하고자 하는 많은 다른 접근 방법이 존재한다. 예를 들면, 최근에 Fowler는 회계, 무역, 조직 구성에 대한 추상화 같은 비즈니스 프로세스의 개념적인 모델링을 표현하기 위해 사용할 수 있는 몇 가지 패턴을 확인했다[8]. 또한 각 도메인별 특징을 반영하기 위한 패턴들을 연구하고 있는데 예를 들면 임베디드 시스템(Embedded System) 도메인 상의 어떤 특징을 반영하기 위한 분석 패턴인 객체 분석 패턴도 존재한다[4].

Gross와 Yu는 비기능적 요구사항과 설계패턴과의 관계에 대해 연구했고[10] Robertson은 요구사항 패턴을 식별, 정의 그리고 접근하기 위해 이벤트/유스케이스 모델링 사용에 대해 제안했다[11]. 또한 Sutchliffe는 서로 다른 어플리케이션들에 대한 일반적인 요구사항을 식별하기 위해 유스케이스 시나리오를 어떻게 조사해야 하는지에 대해서 기술했다[12].

Fernandez와 Yuan은 시멘틱 분석 패턴을 제시했다[7]. 시멘틱 분석 패턴은 몇 개의 유스케이스 혹은 소수의 요구사항 집합들을 가지고 있으며 상태 다이어그램과 시퀀스 다이어그램 관점에서 제안된 솔루션의 동적인 행위를 묘사하고 있다. 마지막으로 van Lamsweerde는 KAOS 방식을 개발했다[13]. 이 방식은 메타모델과 이를 통해 목적 기반의 초기 요구사항 획득을 어떻게 하는지에 대해서 기술했다. 이 밖에도 소프트웨어 아키텍처 패턴[14], 데이터베이스 접근 패턴[15], 무정지형 통신 시스템 패턴[16], 웹서비스에서 비 동기 호출을 위한 패턴[17], 목표지향 요구사항 정제를 위한 패턴[18], 항공 조정 시스템을 위한 설계 패턴[19], 실시간 설계 패턴[20], 보안 패턴[5] 등을 제안했다.

2.2 비즈니스 컴포넌트 재사용 기법

컴포넌트 기반 소프트웨어 재사용 기법은 수정, 조립 그리고 개조로 구분된다[21]. 컴포넌트 수정은 컴포넌트의 기본 특성 중의 하나인 특성(Property)에 의해 이루어지는 컴포넌트의 변경과정을 의미한다. 특성이란 컴포넌트의 범위와 행위를 결정짓는 특성으로 컴포넌트를 재 사용하여 어플리케이션을 설계할 때 그 값을 변경할 수 있다. 컴포넌트 수정을 통한 재사용은 컴포넌트 내부 구현 이해를 요구하기 때문에 재사용에 어려움이 많다. 컴포넌트 조립은 컴포넌트를 재사용하기 위한 가장 자연스러운 과정으로 컴포넌트들끼리 합성(Composition)하는 경우와 다른 어플리케이션들과 통합(Integration) 하는 것을 의미한다.

컴포넌트 개조는 컴포넌트가 어떤 이유로 미리 정의된 인터페이스나 행위를 변경해야 할 경우를 의미한다. 간단하게는 인터페이스 이름 변경이나 파라미터의 형식이 변경되는 것을 들 수 있으며 복잡하게는 새로운 인터페이스를 추가하

는 경우도 있다. 이러한 컴포넌트 개조를 위한 방법으로는 Adapter기법, Wrapper기법[22], Superimposition기법[23], Binary Adaptation기법[24] 등이 있다. 이러한 개조 기법은 컴포넌트 독립성을 이용하는 것으로 변경 영향 범위를 변경 원인이 되는 컴포넌트에게만 국한 시키는 방식이다. 하지만 컴포넌트 독립성만으로 소프트웨어 확장성 및 유연성을 보장하기에 미비하다는 제한 점을 가지고 있다.

비즈니스 요구사항은 시스템 개발 중에도 개발 완료 후에도 지속적으로 변할 수 있는 부분이다. 만일 가변적인 부분이 비즈니스 컴포넌트 안에 포함된다면 변경이 발생할 때마다 코드를 수정해야 한다. 따라서 가변적인 부분과 공통되는 부분을 분리하여 컴포넌트를 개발하고 가변적인 부분을 별도로 재정의 할 수 있도록 하는 것이 컴포넌트 재사용성을 강화할 수 있다.

규칙은 일반적으로 하나 또는 그 이상의 비즈니스 프로세스의 행위를 통제하는 규칙으로서 비즈니스 프로세스적인 규칙과 비즈니스 도메인 규칙로 구성된다. 비즈니스 도메인 규칙은 대상이 가지고 있는 가변적 특성과 특성을 해석하는 가변적 방법을 정의한다. 예를 들면 고객의 나이를 구하는 도메인 서비스는 문제의 특성에 따라 주민번호에 의한 법적 나이 일수도 있고 은행에서 사용하는 나이일 수도 있다. 비즈니스 프로세스 규칙은 하나의 업무를 처리하는데 필요한 작업종류, 순서, 처리 조건을 정의하는 규칙을 말한다.

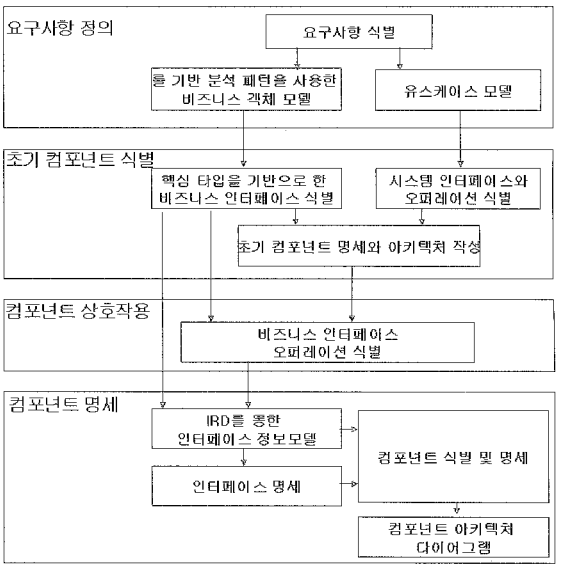
기존에 제시했던 분석 패턴과 다르게 본 논문에서 제시한 규칙 기반 분석 패턴을 사용해 컴포넌트를 개발할 경우에는 비즈니스 프로세스 컴포넌트와 비즈니스 도메인 컴포넌트와 별도로 규칙 컴포넌트를 통해 규칙의 해석 결과에 따라서 다음의 처리흐름 또는 처리결과를 만들어낼 수 있다.

2.3 UML Components개발 프로세스

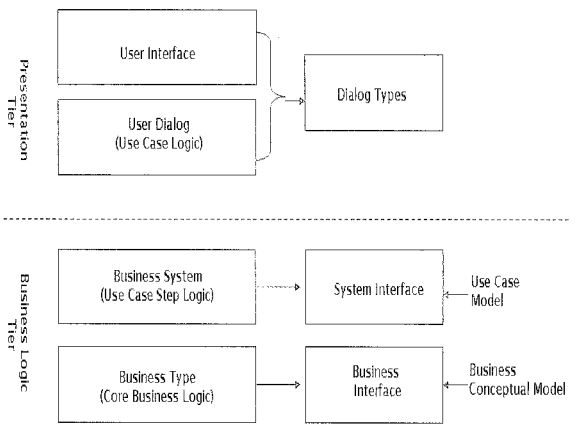
UML Components 개발 프로세스는 크게 요구사항 정의, 컴포넌트 명세, 컴포넌트 공급, 컴포넌트 조립, 테스트 그리고 배포의 과정으로 이루어져 있다[9]. 컴포넌트 명세 단계는 컴포넌트 식별을 위한 분석 단계로서 (그림 1)은 UML Components 개발 프로세스 중에서 컴포넌트 명세에 대한 주요 단계를 보여주고 있다.

요구사항 정의 단계는 요구사항을 식별한 후에 유스케이스 모델과 제시한 규칙 기반 분석패턴을 통해 비즈니스 객체 모델을 작성한다. 유스케이스 모델은 액터 관점에서 시스템의 기능이나 품질 요구사항을 분석한 것이고 비즈니스 객체 모델은 객체 관점에서 시스템을 분석한 것이다.

(그림 2)와 같이 UML Components 개발 프로세스에서 인터페이스는 시스템 인터페이스와 비즈니스 인터페이스로 구분된다. 시스템 인터페이스는 사용자와 시스템 사이의 상호작용을 위한 유스케이스 단계(Step) 로직을 제공하고 있으며 유스케이스에서 사용자와 시스템 사이의 상호작용 시나리오로부터 도출되며 주요 역할은 외부 이벤트를 처음으로 받아들이는 Facade와 다른 서브 시스템 내부에 존재하는 시스템 인터페이스와 상호작용을 조정한다.



(그림 1) UML Components 개발 프로세스



(그림 2) UML Components 어플리케이션 아키텍처

컴포넌트 식별 단계는 비즈니스 인터페이스와 시스템 인터페이스를 식별하며 시스템 인터페이스의 오퍼레이션을 식별해서 초기 컴포넌트 명세와 아키텍처를 작성한다. 컴포넌트 상호작용 단계는 식별된 인터페이스들과 초기 아키텍처를 기반으로 각 유스케이스 실현을 위한 인터페이스간 상호작용 명세를 작성한다. 마지막으로 컴포넌트 명세 단계는 각 컴포넌트 명세를 작성하는 단계로 식별된 인터페이스가 관리해야 할 정보모델을 통해 인터페이스 명세를 작성하고 이를 기반으로 컴포넌트 타입 식별, 각 컴포넌트 명세, 식별된 컴포넌트 타입을 기반으로 컴포넌트 아키텍처 다이어그램을 작성한다.

3. 규칙 기반 분석 패턴

3장에서는 외부 이벤트에 대해 비즈니스 업무를 규칙 기반으로 분석하기 방법인 규칙 기반 분석 패턴을 제안한다. 규칙 기반 분석 패턴도 설계 패턴과 마찬가지로 하나의 패턴이기 때문에 패턴이 가지는 포맷에 맞추어서 제시한다.

3.1 개요(Synopsis)

제시한 규칙 기반 분석 패턴의 용도는 먼저 많은 서브시스템들을 가진 복잡한 시스템을 서로 다른 분석가에 의해 분석 된 후 비즈니스 컴포넌트를 개발할 경우 산출물 일관성을 유지하고 시스템의 가독성을 좋게 함으로서 개발 생산성이나 유지보수를 향상 시키기 위해서 사용한다. 두 번째는 외부 액터에 의해 발생된 모든 이벤트에 대해 비즈니스 어플리케이션 트랜잭션을 규칙 기반으로 처리해야 할 경우 시스템 내부의 비즈니스 개념들을 규칙 기반으로 추출하고 그들 간의 상호작용 관계를 효과적으로 모델링 할 때 사용한다. 세 번째로 제시한 규칙 기반 분석패턴이 가지는 핵심 개념들을 사용해 효과적으로 비즈니스 컴포넌트를 개발할 수 있으며 식별된 컴포넌트 중에 규칙 컴포넌트를 사용해서 비즈니스 도메인 규칙과 비즈니스 프로세스 규칙을 처리함으로써 컴포넌트의 재사용성이나 적응성을 높일 수 있다.

3.2 배경(Context)

제시한 규칙 기반 분석패턴이 해결하고자 하는 문제는 두 가지 분류할 수 있다. 첫 번째는 서로 다른 업무 분석가들이 각 서브시스템을 분석한 후에 이를 기반으로 컴포넌트를 개발할 경우 각 업무 분석가의 경력이나 배경 지식에 따라 각 서브시스템 분석 내용이나 세부 수준이 다른 산출물을 작성하게 된다. 내용이나 수준이 다른 분석 산출물을 기반으로 컴포넌트를 개발할 경우 컴포넌트 개발이나 유지보수 측면에서 많은 비효율성이 발생하며 이는 프로젝트 관리에 많은 위험을 초래한다.

제시한 규칙 기반 분석패턴이 해결하고자 하는 두 번째 문제는 컴포넌트의 재사용성과 적응성이다. 기존 컴포넌트 재사용에 대한 연구는 환경과 요구사항 변경에 대한 파급효과를 최소화하기 위한 컴포넌트 독립성에 대한 것들이다. 하지만 컴포넌트 독립성 만으로는 만족할 만한 컴포넌트의 변경 용이성이나 재사용성을 달성할 수 없으며 규칙 컴포넌트와 같은 별도의 컴포넌트를 통해서 비즈니스 도메인 규칙과 비즈니스 프로세스 규칙을 처리해야 한다. 제시한 규칙 기반 분석 패턴은 가변적인 것들을 규칙 컴포넌트에서 관리한다는 측면에서 이러한 문제점에 대한 해결책이 될 수 있다.

3.3 고려사항들(Forces)

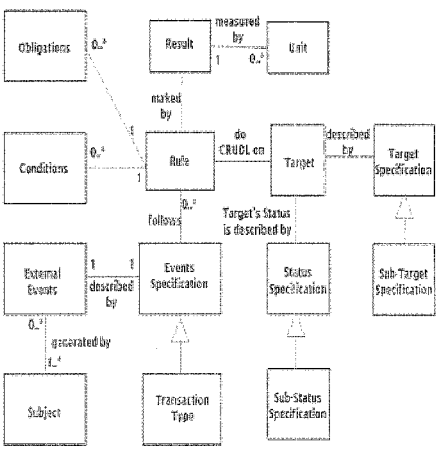
논문에서 제시한 분석 패턴이 제기한 문제에 대한 솔루션을 내기 위해서는 몇 가지 제약사항이 존재한다. 먼저 제시한 분석패턴이 적용될 수 있는 도메인은 은행 업무와 같이 모든 외부 이벤트에 대해 규칙 기반으로 거래를 처리해야 하는 경우이다. 두 번째로 분석 패턴상의 Target은 적용 영역에 따라 한 개 이상의 Target들이 존재할 수 있으며 각 Target별 액션을 위한 가변적인 규칙 들이 존재한다. 세 번째로 Target에 대한 상태 명세는 Target이 생명주기를 가지며 생명주기 동안 각 상태를 분별할 수 있을 때 가능하다.

제시한 규칙 기반 분석 패턴을 사용해 컴포넌트를 개발할 때 컴포넌트 재사용성을 강화하기 위해서는 비즈니스 프로세스적인 규칙이나 비즈니스 도메인 규칙에 대한 처리는 규칙 컴포넌트 내부에 포함시킬 수 있다. 이는 환경, 요구사항 그리고 기능 변경 발생 시 컴포넌트 재사용성과 적응성을 향상시키기 위함이다.

3.4 해결책(Solution)

제시한 분석 패턴이 가지는 주요 개념들에 대한 설명을 위해 은행 고객이 ATM을 사용해 돈을 예치(Deposit)한다고 가정하자. (그림 3)은 본 논문에서 제시한 분석 패턴에 대한 아키텍처 템플릿이다.

고객이 예치 서비스 요청을 보내면 ATM은 어떤 타입의 요청이 발생했는지를 식별하고 이 요청 이벤트 타입에 설정된 규칙들을 찾아서 해당 조건을 검사한 후에 규칙에 따라 고객의 계좌를 대상으로 액션을 수행한다. 액션 수행 후의 결과 값은 규칙에 따라 고객 계좌에 다시 재 반영된다. 다음은 각 개념들이 가지고 있는 주요 책임에 대한 설명이다.



(그림 3) 규칙 기반 분석 패턴 아키텍처 템플릿

3.4.1 Subject

(그림 3)에서 주체(Subject)는 서비스 사용을 위해 외부 이벤트를 발생시키는 액터를 말한다. 주체가 하는 책임은 시스템에 거래요청을 함으로서 이벤트를 발생시킨 후에 처리결과를 받는 것이다. ATM 예제에서 은행 고객이 바로 주체가 된다.

3.4.2 External Events

외부이벤트들은 시스템 외부 액터가 시스템을 사용 함으로서 발생하는 모든 이벤트를 말한다. ATM예제에서는 은행고객이 요청하는 입금, 출금, 계좌이체와 같은 거래들이 외부 이벤트가 된다.

3.4.3 Events Specification

이벤트 명세는 이벤트에 대한 메타 개념(Meta Concept)으로서 이벤트에 대한 표준 명세(Specification)을 담고 있다. 이벤트 명세에는 발생 가능한 이벤트 타입뿐만 아니라

각 이벤트 타입 별로 따라야 할 규칙을 가지고 있다.

3.4.4 Rule, Conditions, Obligations

외부 액터에 의해 발생한 모든 이벤트는 비즈니스 규칙에 적합한 지 검사된 후에 처리된다. 비즈니스 규칙은 이벤트 표준 명세상의 이벤트 타입(Event Type)과 관련해서 정의되며 규칙 표현은 조건(Condition), 의무(Obligation)사항 그리고 액션(Action)의 형태로 기술된다. 의무사항은 규칙 컨디션들이 만족되고 액션 수행 전에 해야 할 행위들로서 주로 액션수행과 관련된 규칙 관련 정보들을 얻어오는 것이다.

3.4.5 Result

결과 개념은 프로세스 관점에서 어떤 행위를 의미하며 정보 관점에서는 액션 수행 후의 상태 결과를 나타낸다. 액션 수행 여부와 방법은 비즈니스 규칙에 의해서 결정되며 액션 수행후의 상태 결과에는 소스에 해당하는 이벤트 타입 정보를 담고 있다. ATM예제에서 결과는 이자율, 세금 그리고 적수 계산과 같은 것들이다. 액션 수행 후 결과 값은 규칙을 통해 원래의 대상 자원에 다시 반영한다.

3.4.6 Unit

단위(Unit) 개념은 Folwer의 Quantity Pattern을 규칙 기반 분석 패턴에 적용한 것으로 대상 자원이나 결과를 양적으로 측정하기 위한 기본 단위를 나타낸다.

3.4.7 Target

외부 이벤트에 대해 규칙의 컨디션이 만족되면 규칙은 어떤 행위를 하게 된다. 이러한 행위는 대상이 존재하는데 Target은 행위의 근거가 되는 대상을 의미한다. Target은 프로세스 적인 것 보다는 정보를 표현하는 엔티티이며 CRUD(Create, Read, Update, Delete)를 수행한다.

3.4.8 Target Specification, Sub-Target Specification

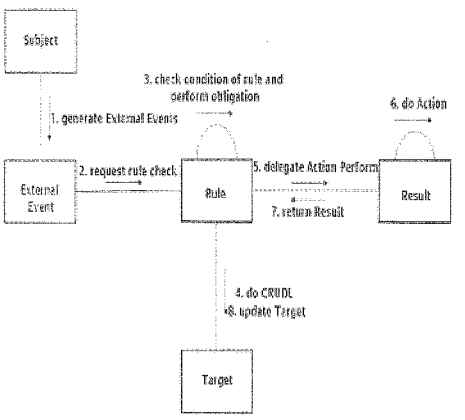
대상자원 명세는 대상자원이 가질 수 있는 다양한 타입들을 표현하며 그것들의 표준 명세를 제공한다. 명세에서 정의한 타입에 따라 다양한 Sub-Target Specification이 존재한다. 앞의 ATM 예제를 들어 설명하면 대상 자원인 계좌는 대상자원 명세에서 명시한 타입에 따라 다양한 계좌 상품(예: 요구 불 계좌, 정기예금 계좌, 당좌계좌)이 가능하다

3.4.9 Status Specification, Sub-Status Specification

대상자원 상태에 대한 명세는 Target이 일정한 생명주기를 가지면 생명주기상의 특정 상태가 구분(Discrete)이 가능한 경우에 적용된다. Status Specification은 Target의 상태에 대한 명세이며 Sub-Status Specification은 Status Specification에서 명시한 다양한 종류의 서브상태를 나타낸다. ATM예제에서 정상계좌, 비정상 계좌가 여기에 해당되면 비정상 계좌는 해지, 사고 그리고 세금면제 계좌와 같은 것들이 존재한다.

3.4.10 규칙 기반 이벤트 처리과정

(그림 4)는 규칙 기반 분석 패턴을 사용해 비즈니스 프로세스를 처리하는 단계이다. 먼저 주체는 이벤트 명세에 명시된 외부 이벤트를 발생한다. 이벤트를 처리하기 위해 규칙은 명시된 조건이 있는지 검사해서 만일 조건이 있다면 해당 조건을 검사한다. 조건검사 후에 규칙은 액션을 수행하기 전에 수행해야 할 의무사항이 있는지 검사한다. 만일 의무사항 내용이 액션수행에 필요한 규칙 관련 정보인 경우에 Target에 대해 CRUD를 요청한다. 규칙은 의무사항 수행 후에 규칙에 명시된 액션이 존재한 경우 Result에 액션을 위임한다. 규칙은 액션 수행 후 Result로부터 반환된 값을 받아 해석하고 조건 검사를 수행한 후에 원래의 대상 자원에 다시 반영하다. 규칙은 Result와 Target 간의 프로세스 흐름을 통제하는 조정자 역할을 수행하며 대상에 대한 가변적인 특성을 해석한다.



(그림 4) 규칙 기반 분석 패턴의 이벤트 처리 과정

3.4.11 규칙 기반 분석패턴을 사용한 컴포넌트 개발 방법

제시한 규칙 기반 분석 패턴은 분석 단계에서 주요 비즈니스 개념을 규칙 기반으로 추출하기 위한 분석틀이다. 본 절에서는 규칙 기반 분석패턴을 통해 식별된 몇 개의 개념들을 사용해 UML Components개발 프로세스에 적용하여 컴포넌트를 개발하는 방법을 기술한다.

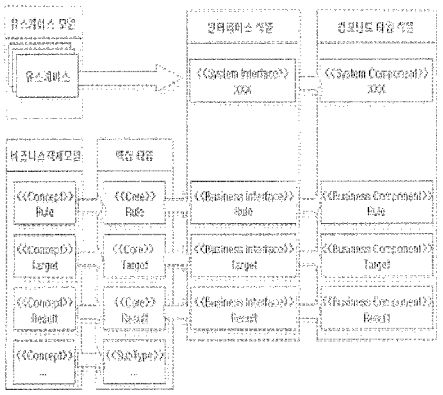
(그림 5)는 제시한 규칙 기반 분석 패턴을 사용해 비즈니스 객체 모델을 만들고 이를 기반으로 인터페이스와 컴포넌트를 식별하는 과정이다. 제시한 규칙 기반 분석 패턴을 사용해 작성된 비즈니스 객체모델에서 Rule, Target 그리고 Result가 정보간 의존성 관계에서 핵심타입이 되고 나머지 개념들은 이들 핵심 타입을 위한 Sub-Type이 된다. 세 개의 핵심타입은 각각 3개의 비즈니스 인터페이스가 되며 인터페이스와 컴포넌트간 매핑을 사용해서 컴포넌트를 식별한다.

(그림 6)은 인터페이스와 컴포넌트간 네 가지 매핑 종류를 나열한 것이다. 첫 번째 매핑은 컴포넌트 별로 하나의 인터페이스를 두는 방식이다. 두 번째는 하나의 컴포넌트 안에 여러 개의 인터페이스가 존재하는 경우이다. 세 번째는 여러 개의 컴포넌트가 한 개의 인터페이스를 공유하는 방식으로 인터페이스 구현은 각각 다르다. 네 번째는 외부

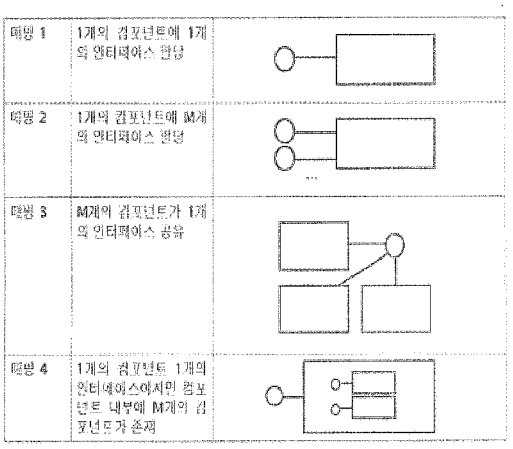
에서는 한 개의 컴포넌트가 한 개의 인터페이스를 가지고 있는 첫 번째와 유사하지만 내부에서만 사용할 수 있는 M개의 컴포넌트가 존재하는 형태이다.

초기의 3개의 비즈니스 인터페이스에 대해 (그림 6)의 매핑1에 해당하는 1:1 기본 매핑 방식을 사용해 앞의 (그림 5)와 같이 각각 Rule, Target, Result와 같은 3개의 컴포넌트가 된다. 하지만 Target은 이후의 정제과정에서 인터페이스나 컴포넌트가 가지는 규모, 특성 그리고 구현에 따라 여러 개의 인터페이스나 컴포넌트로 세분화 될 수 있다.

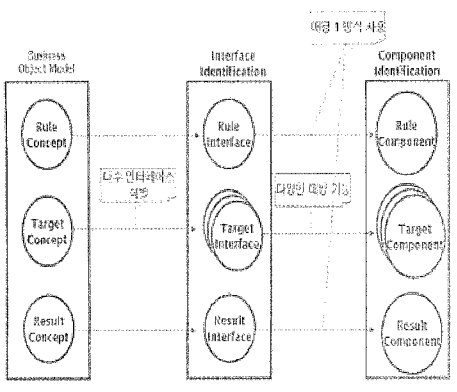
(그림 7)은 주요 개념이 인터페이스와 컴포넌트로 어떻게 매핑 되는지를 보여주고 있다.



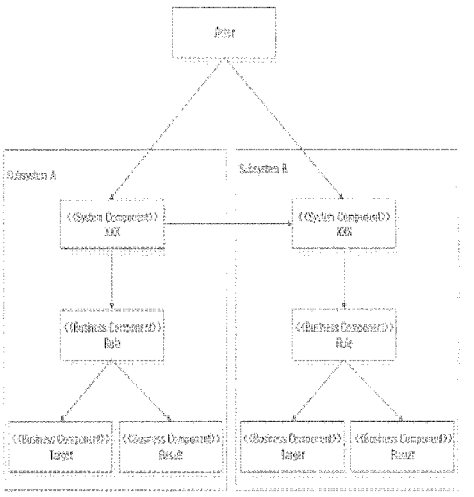
(그림 5) 규칙 기반 분석 패턴을 사용한 인터페이스 및 컴포넌트 식별



(그림 6)인터페이스와 컴포넌트간 매핑 종류



(그림 7) 분석 패턴을 사용한 인터페이스와 컴포넌트간의 매핑



(그림 8) 컴포넌트 아키텍처 다이어그램

Rule과 Result는 각각 한 개의 인터페이스, 컴포넌트로 식별되지만 Target은 여러 개의 인터페이스와 여러 개의 컴포넌트로 세분화 될 수 있다. 시스템 인터페이스는 유스케이스 모델상의 유스케이스들로부터 식별되어 인터페이스와 1:1매핑 방식을 사용해 한 개의 시스템 컴포넌트가 된다

(그림 8)은 앞에서 식별된 Rule, Target, Result 컴포넌트 타입을 기반으로 작성된 컴포넌트 아키텍처 다이어그램이다.

제시한 아키텍처의 장점은 먼저 시스템 컴포넌트가 가지는 시스템 인터페이스만 외부에 노출시키고 Rule, Target 그리고 Result 비즈니스 컴포넌트가 가지는 인터페이스를 외부에 노출 시키지 않음으로써 컴포넌트의 독립성을 강화하고 있으며 따라서 인터페이스 변경으로 인한 영향을 최소화 시킬 수 있다. 둘째 비즈니스 프로세스나 도메인에 관련된 규칙을 담고 있는 규칙 컴포넌트를 Target과 Result컴포넌트와 별도로 분리할 수 있기 때문에 가변적인 도메인 특성을 해석하고 Target과 Result 컴포넌트 타입간 처리 흐름을 변경을 용이하게 할 수 있다. 제시한 규칙 기반 분석패턴은 컴포넌트 독립성과 규칙 컴포넌트를 통한 가변성 관리를 통해 컴포넌트의 유연성과 적응성을 강화할 수 있다.

3.5 결과(Consequence)

본 논문에서 제시한 규칙 기반 분석 패턴은 다음과 같은 장점들을 제시한다. 첫 번째로 다른 분석 패턴과 같이 비즈니스 프로세스 분석을 위한 개념적 틀을 제시하지만 본 논문에서는 액터로부터 발생하는 외부 이벤트에 대해 규칙 기반 프로세스 분석을 위한 틀을 제시한다. 둘째로 명세(Specification)에 해당하는 지식 개념(Knowledge Concept)들과 운영수준에 해당하는 실제 개념(Real Concept)을 분리함으로써 각 도메인 별 혹은 환경 별로 실제 개념에 해당하는 추가적인 이벤트나 대상 자원의 타입들을 쉽게 추가할 수 있다.

제시한 분석 패턴이 가지는 세 번째 장점은 대상자원이나 결과 개념을 양적으로 측정하기 위해 필요한 단위(Unit) 개념을 추가함으로써 대상 자원을 다양한 단위로 측정할 수 있다. 네 번째로 대상 자원이 일정 생명주기를 가지고 각

생명주기 상태가 구분(은행 계좌 예: 정상 계좌 혹은 비정상 계좌)이 가능하고 미리 정의되어 있는 경우에는 대상자원의 상태를 상속을 이용해 서브 타입 형태로 상태의 개념을 확장해서 표현할 수 있다. 다섯째 규칙 기반 분석 패턴은 비즈니스 어플리케이션 프로세스를 규칙, 결과, 대상의 3가지 주요 개념 관계로 표현하기 때문에 개발자들이 쉽게 실제 모듈이나 컴포넌트가 가지는 작업(Task)의 복잡성을 단순화 시킬 수 있으며 개발 프로세스상의 주요 단계들을 효과적으로 수행할 수 있다. 특히 여러 서브시스템이 존재하는 상황에서 모델러들이 같은 관점에서 일관되고 효율적으로 컴포넌트 모델링을 수행 할 수 있다. 마지막으로 제시한 규칙 기반 분석 패턴을 사용해 컴포넌트 개발 시 비즈니스 프로세스 컴포넌트나 비즈니스 도메인 컴포넌트와 분리해서 규칙 컴포넌트에 프로세스나 정보상의 가변적인 부분들을 관리 하게 함으로서 컴포넌트 재사용성이나 적응성을 향상 시킬 수 있다.

4. 규칙 기반 분석패턴을 사용한 은행 수신 컴포넌트 개발 적용 사례

4장에서는 많은 서브시스템이 존재하는 복잡한 어플리케이션 개발 시 본 논문에서 제안한 규칙 기반 분석 패턴을 적용할 경우 비즈니스 객체 모델의 일관성, 가독성이 좋다는 것을 예시하기 위해 은행 수신과 수출 서브시스템 간의 비즈니스 객체 모델을 비교한다. 또한 효율적인 컴포넌트 개발 가능성을 제시하고 계좌 상품별로 가변성이 존재하는 특징을 가진 은행 수신업무에 대해 본 논문에서 제시한 규칙 기반 분석패턴을 사용해 컴포넌트를 개발한 적용 사례를 제시한다.

4.1 수신과 수출 업무 개요

은행 수신 업무(Deposit)는 모든 종류의 예금계좌 관리와 그에 따른 모든 거래를 처리하는 개인 고객을 상대로 한 은행 소매업의 한 부분으로서 타 업무 부분과 많은 인터페이스를 갖고 있다. 반면 은행 수출 업무는 수출기업을 상대로 수출 업무를 자동화를 위한 은행 도매업 부분중의 하나이다. (그림 9)는 수신과 수출 업무를 비즈니스 프로세스와 비즈니스 도메인 차원에서 분석한 것이다.

수신 업무 프로세스는 크게 계좌 관리, 계좌처리, 통장관리 업무로 구분되며 계좌관리는 계좌 생성, 수정, 해지 그리고 이자 관리로 구성되며 계좌처리는 입금, 출금, 조회 그리고 이체로 구성된다. 통장관리는 통장 발행과 통장 출력으로 구성된다.

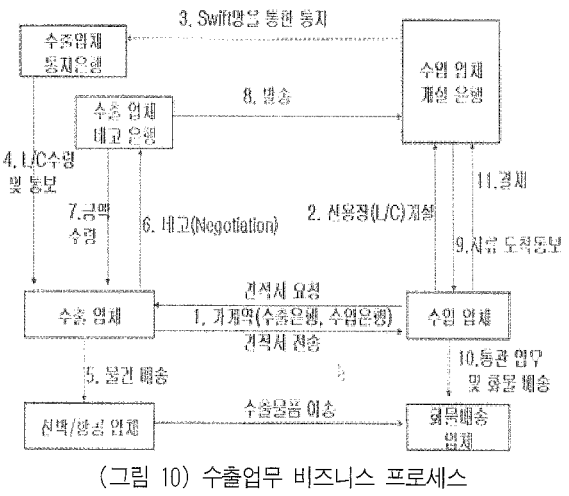
비즈니스 도메인 차원에서 수신 업무와 관련된 예금은 크게 분류해 보면 유동성 예금과 고정성 예금으로 분리된다. 유동성 예금에는 입출입이 자유롭지만 이자율이 낮은 보통예금, 이자는 없고 주로 수표발행과 관련하여 기업들이 사용하는 당좌예금, 보통 예금보다 이율은 조금 높고 주로 개인이 사용하는 계좌로서 저축예금 등이 있으며 고정식 예금에는 일정기간을 정해놓고 가입하며 이자가 가장 많은 정기예금, 일정 기간마다 부금을 납입하고 만기에 한번에 받는 정기적금, 정기예금이긴 하지만 통장을 발행하지 않고 증서를 발행하여 그 증서를 양도나 매매 할 수 있고 만기에 그 증서를 가진 사람이 은행에서 돈을 받을 수 있는 양도성 예금, 만기가 없는 정기예금인데 대신 해약하기 몇 일 전에 은행에 통보해야만 지급받을 수 있는 통지 예금 등이 존재한다. 이러한 계좌 개설은 예금 종류별로 이루어지며 이자율, 세율, 이전계좌 사용여부, 복수계좌 사용여부 등과 같은 규칙은 예금상품이나 은행 그리고 국가별로 상이하다.

(그림 10)은 수출업무에 대한 비즈니스 프로세스이다. 설명의 편의를 위해 미국기업이 구매할 물건의 종류와 수량을 적어 한국기업에 견적서(Offer Sheet)를 요청해 한국 수출기업이 미국의 수입 기업과 가계약을 맺고 거래 방식을 신용장(Letter Of Credit)으로 하기로 했다고 가정하자.

미국 기업은 먼저 신용장을 개설하기 위한 담보물을 가지고 자신의 주거래 은행을 찾아가 신용장을 개설한다. 그러면 미국 은행에서 한국 은행으로 국제 은행간 금융정보 통신망인 SWIFT를 통해 신용장을 전송된다. 한국의 통지 은

수신업무 (Deposit)	비즈니스 프로세스	계좌관리	계좌 생성, 수정, 해지, 이자율 계산...
		계좌처리	입금, 출금, 조회, 계좌이체...
		통장관리	통장발행, 출력...
	비즈니스 도메인	계좌	보통 예금, 당좌예금, 저축예금, 정기적금, 별단예금 ...
수출업무 (Export)	비즈니스 프로세스	신용장관리	신용장 등록, 통지, 전송, 수수료
		결제및 추심	신용장 혹은 DA/DP방식 결제, 신용장 혹은 DA/DP방식 추심
		유가증권관리	유가증권 수락통지, 결산처리, 이자, 만기일 연장, 부도 유가증권처리
	비즈니스 도메인	신용장	매입제한 신용장, 양도가능 신용장, 기한부 신용장
		유가증권	환 어음, 부도어음

(그림 9) 수신과 수출 업무 시스템



(그림 10) 수출업무 비즈니스 프로세스

행에서는 한국 수출기업에게 신용장도착 통지를 알린다. 한국 수출 기업은 신용장에 기재되어 있는 금액, 기간, 지급방법, 구비서류 등의 각종 조항에 맞추어 물건을 배나 항공기에 실어서 전송한다. 그런 후에 한국기업은 신용장에 기재되어 있는 상업송장, 포장명세서, 선하증권 등과 같은 구비서류들을 준비해 은행에 교환결제(Negotiation)를 요청하고 은행에서는 별다른 하자가 없으며 돈을 지급한다.

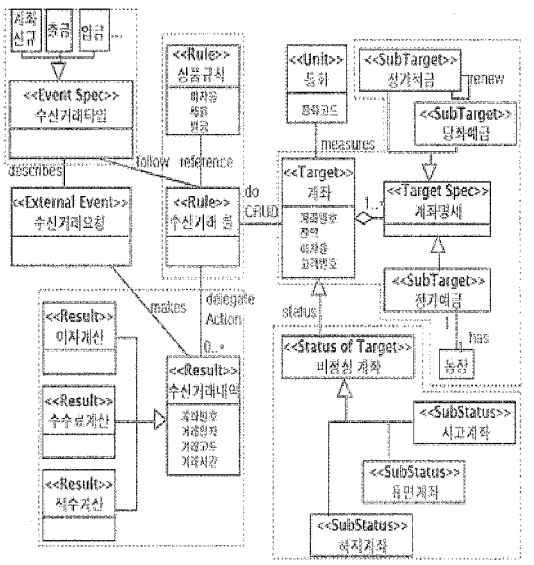
제출된 서류들은 한국의 은행에서 미국의 은행으로 전송되고 미국의 은행에서는 자신들에게 도착한 서류가 도착했음을 수입기업에게 통보하고 수입기업은 돈을 지불하고 서류를 찾아 공항이나 배 입항하는 곳에 가서 물건을 찾아 온다.

수출 업무 비즈니스 프로세스는 크게 신용장관리, 결제 및 추심, 유가증권 관리로 구성된다. 신용장관리는 수입업자 은행으로부터 전송된 신용장 등록, 수출 기업에게 통지, 수입업자 은행에게 신용장 전송, 신용장 수수료 등으로 구성된다. 교환 결제 및 추심은 수출 업자가 거래 은행으로부터 수출 대금을 지급 받는 행위를 말한다.

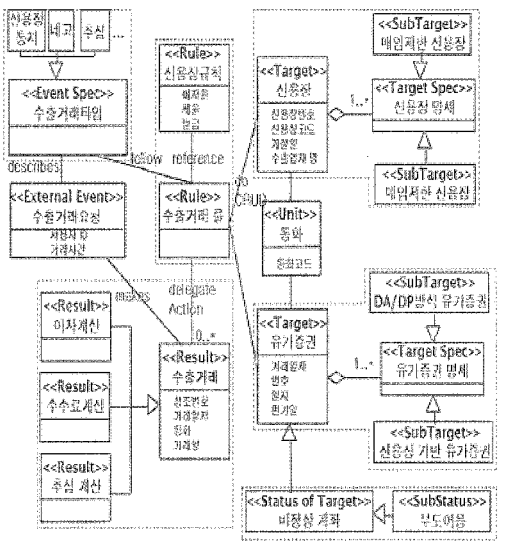
교환 결제 방식은 신용장 기반과 무신용장 기반이 존재한다. 신용장 기반은 은행이 지급을 보증한다는 것을 의미하는 신용장을 기반으로 한 결제 방식으로 수출업자가 신용장에 기술되어 있는 구비 서류 사항만 잘 갖추면 거래 은행으로부터 돈을 지급받는다. 무신용장 방식은 은행의 지급 보증 없이 수출입 업자간의 신용을 기반으로 한 방식으로서 주로 환어음을 기반으로 결제하는데 무신용장 지급인도 조건방식(D/P)과 무신용장 인수인도조건(D/A)이 있다. D/P방식의 대금 결제 방식은 수출상이 수출 물품을 선적하고 수입상을 지급인으로 하는 일람출금화 환어음을 발행하여 선적 서류와 함께 거래 은행을 통하여 수입국의 추심은행 앞으로 어음 대금을 추심하면 추심은행은 수입업자에게 어음을 제시하여 수입상의 대금지급과 동시에 선적서류를 인도하고 대금을 추심의뢰 은행에 송금하여 결제하는 방식이다. D/A방식은 D/P거래와 대금을 추심하는 과정은 같으나 일람출금화 환어음 대신 일람 후 정기 또는 확정일 출급환어음을 발행한다. 추심은행은 수입상에게 수출자발행 기한부 어음을 제시하여 수입자의 서명과 함께 선적서류를 건네 주고 만기일에 대금을 회수하는 방식이다. 수출 업무의 비즈니스 도메인은 신용장과 어음과 같은 유가증권이 존재한다. 수출 업무에 관련된 규칙들은 대부분 신용장 종류와 관련해서 결정된다.

4.2 비즈니스 개념모델 추출

(그림 11)과 (그림 12)는 본 논문에서 제시한 규칙 기반 분석 템플릿을 통해 추출된 수신과 수출업무 비즈니스 개념모델이다. (그림 13)은 규칙 기반 분석 패턴이 제시한 개념이 수신과 수출 도메인에서 어떠한 인스턴스로 매핑 되는지를 보여준다. 규칙 개념은 수신업무에서 주로 계좌 상품별로 수출업무에서는 신용장 종류별로 나타난다. 수신 업무에서 대상자원 개념에 대한 인스턴스는 계좌로 나타나고 수출업무는 신용장이나 유가증권으로 나타난다.



(그림 11) 수신업무 비즈니스 개념모델



(그림 12) 수출업무 비즈니스 개념모델

룰 기반 분석패턴	수신업무	수출업무
Subject	점원	수출 담당 점원
External Events	수신거래요청	수출거래요청
Transaction Type	계좌신규, 출금, 입금, 계좌이체...	신용장 통지, 신용장전송, 네고, 추심...
Rule	계좌 상품별 규칙(이자율, 세율, 벌금)	신용장 종류별 규칙, 유가증권 규칙...
Target	계좌	신용장, 유가증권
Sub Target	정기적금, 당좌예금, 정기예금...	매입제한 신용장, 기한부 신용장...
Sub Status	해지계좌, 사고계좌, 수수료면제 계좌, 휴면계좌...	유효기간 지난 신용장, 부도 어음...
Result	수신거래 내역(이자 계산, 수수료계산, 적수계산...)	수출거래 내역(이자 계산, 수수료계산, 추심...)

(그림 13) 수신업무에서 개념과 인스턴스간의 매핑

서브 대상 자원 개념은 수신업무의 경우에는 정기적금, 당좌예금, 정기적금 등과 같은 계좌상품 형태로 수출업무의 경우에는 기한부 신용장 등과 같은 신용장 종류로 나타난다. 서브 상태 개념은 수신업무에서는 비정상적인 계좌인 해지계좌, 사고계좌 등으로 나타나며 수출업무의 경우는 유효기간이 지난 신용장이나 부도 어음과 같은 형태로 나타난다.

수신업무와 수출업무 예제에서 보는 것처럼 분석 패턴을 통해 비즈니스 개념모델을 작성하게 되면 각 업무 도메인별로 개념에 대한 인스턴스를 쉽게 도출할 수 있으며 또한 많은 서브시스템상의 비즈니스 개념모델간의 통일성, 일관성, 가독성을 좋게 한다.

4.3 인터페이스 명세서 작성

UML Components 개발 프로세스에서 인터페이스는 시스템 인터페이스와 비즈니스 인터페이스로 구분된다. 시스템 인터페이스는 사용자와 시스템 사이의 상호작용 시나리오로부터 도출되며 수신 서브시스템의 모든 유스케이스를 위한 하나의 시스템 인터페이스를 도출한다. (그림 14)는 수신 서브 시스템에서 비즈니스 인터페이스를 도출하기 위한 핵심 타입과 비즈니스 인터페이스 식별을 나타내고 있다.

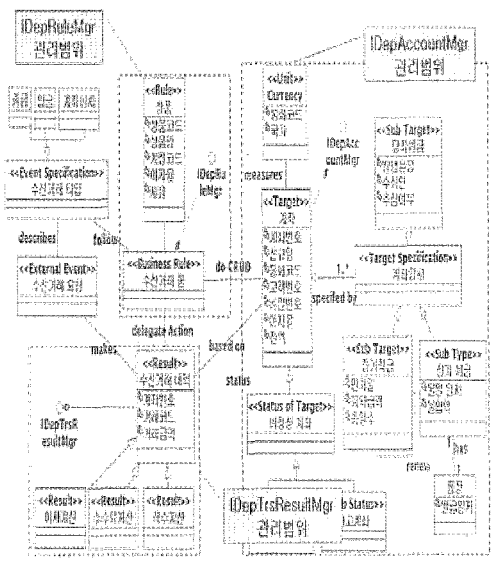
(그림 14)와 같은 비즈니스 인터페이스 식별을 위해서는 (그림 11)의 수신 비즈니스 개념 모델을 정제한 후에 비즈니스 타입 모델을 만들고 이를 기반으로 프로세스 처리와 관련해 다른 객체와의 의존성이 없이 독자적으로 존재할 수 있는 객체를 핵심 타입(Core Type)을 식별한다. 그리고 각 핵심 타입 별로 한 개의 비즈니스 인터페이스를 할당한다.

규칙 기반 분석 패턴 상의 규칙, 대상자원, 결과 개념만이 핵심 타입이기 때문에 (그림 14)와 같이 3개의 핵심 타입에 각각 한 개의 비즈니스 인터페이스를 할당했다. 또한 점선 형태의 블록은 이 인터페이스가 관리해야 할 정보모델을 나타낸다. 수신 서브 시스템에서 한 개의 시스템 인터페이스는 IDEPSIMgr이고 비즈니스 인터페이스는 (그림 14)와 같이 각각 IDepRuleMgr, IDepAccountMgr, IDepTrsResultMgr로 명명된다.

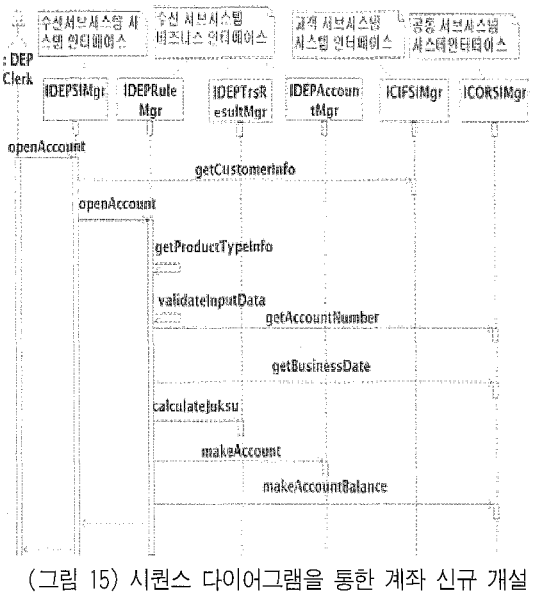
인터페이스가 식별되었으며 각 인터페이스에 대한 메소드를 식별해야 한다. (그림 15)는 수신 시스템의 신규 개좌 개설 유스케이스 실현을 위한 시스템 인터페이스와 비즈니스 인터페이스를 사용한 시퀀스 다이어그램이다.

신규 계좌 개설을 위한 시퀀스 다이어그램 작성시 상호작용 패턴은 본 논문에서 제시한 상호작용 패턴을 기반으로 작성된다. 계좌를 개설하기 위해 선행조건 형태로 필요한 고객 정보에 대한 검증과 정보를 제공하는 것은 고객정보 관리 시스템인 CIF에서 처리한다. 규칙 기반 분석 패턴 상에서 타 서브 시스템과 연동은 시스템 인터페이스에서만 수행할 수 있다. 따라서 IDEPSIMgr 시스템 인터페이스에서 고객 정보 관리 서브 시스템의 시스템 인터페이스인 ICIFSIMgr를 호출한다. 하지만 은행 공통 업무 서비스를 제공하는 ICORSIMgr 시스템 인터페이스를 호출할 때는 예외적으로 시스템 인터페이스가 아닌 규칙이나 대상자원 그리고 결과와 같은 비즈니스 인터페이스를 구현하는 부분에서 호출할 수 있다. 이는 공통 컴포넌트가 가지는 인터페이스가 안정성이 있고 또한 호출 빈도가 많다는 것을 고려한 것이다.대상자원과 결과의 조정 역할은 규칙 에서 하도록 제한하고 있는데 (그림 15)에서 보는 것처럼 IDEPRuleMgr 규칙 비즈니스 인터페이스가 대상 자원 개념인 IDEPAccountMgr 비즈니스 인터페이스와 결과 개념인 IDEPTransactionResultMgr 비즈니스 인터페이스를 조정하고 있다. 수신 서브시스템 상에 존재하는 각 유스케이스 별로 (그림 15)와 같은 시퀀스 다이어그램을 작성하게 되면 각 시스템 인터페이스와 비즈니스 인터페이스에 대한 메소드가 식별된다.

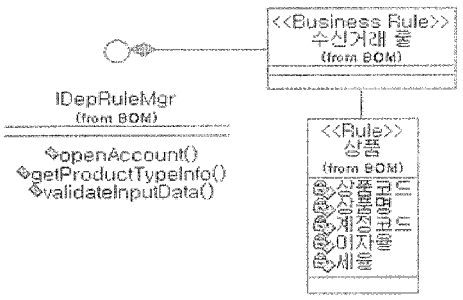
각 인터페이스에 대한 메소드가 식별되면 각 인터페이스 대한 책임할당 다이어그램을 작성하게 되며 이를 기반으로



(그림 14) 수신 서브시스템의 핵심 타입과 비즈니스 인터페이스 식별



(그림 15) 시퀀스 다이어그램을 통한 계좌 신규 개설 유스케이스 실현 방법

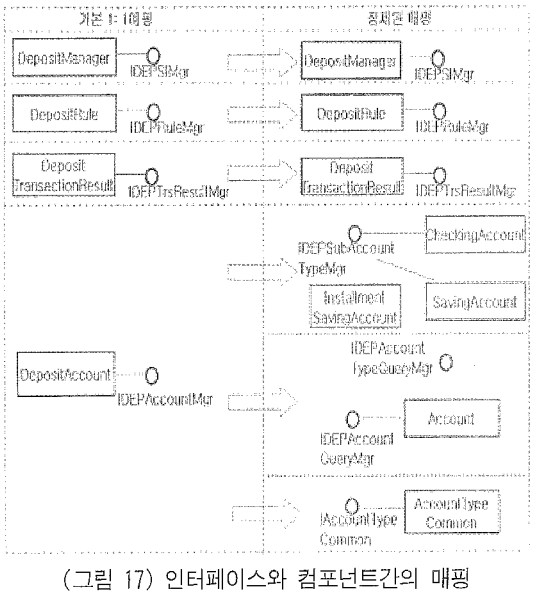


(그림 16) IDepRuleMgr 비즈니스 인터페이스 책임할당 다이어그램

인터페이스 명세서를 작성한다. (그림 16)은 IDepruleMgr 규칙 비즈니스 인터페이스에 대한 책임할당 다이어그램이다. (그림 16)과 같은 각 인터페이스 별 책임할당 다이어그램에는 인터페이스에 식별된 공개된 메소드 리스트, 인터페이스가 제공하는 서비스, 그 서비스를 제공하기 위해 관리해야 할 정보모델이 존재한다. 신규 개좌개설을 처리하기 위해 규칙 인터페이스가 가지는 메소드는 3개의 메소드가 존재하며 관리해야 할 정보는 상품 클래스에 존재하는 속성들이다.

4.4 컴포넌트 식별

인터페이스에 대한 명세를 작성한 후에 컴포넌트를 식별한다. (그림 17)은 수신 서브시스템에서 식별된 인터페이스와 컴포넌트간의 매핑을 보여주고 있다. 식별된 인터페이스를 기반으로 초기에는 1:1 기본 매핑을 수행한다. DepositManager시스템 컴포넌트에 IDEPSIMgr 시스템 인터페이스를 DepositRule규칙 비즈니스 컴포넌트에 IDEPRuleMgr비즈니스 인터페이스를 DepositAccount 대상자원 비즈니스 컴포넌트에 IDEPAccountMgr비즈니스 인터페이스를 DepositTransactionResult 결과 비즈니스 컴포넌트에 IDEPTrsResultMgr 비즈니스 인터페이스를 할당한다.

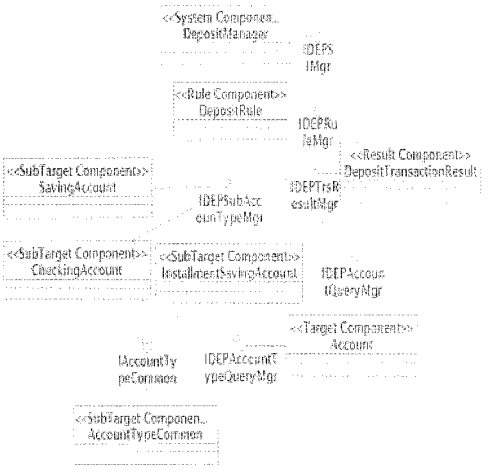


(그림 17) 인터페이스와 컴포넌트간의 매핑

기본 매핑이 완료된 후에 정제된 매핑을 수행하는데 시스템 컴포넌트, 규칙 비즈니스 컴포넌트, 결과 비즈니스 컴포넌트는 기본 매핑을 그대로 유지한다. 하지만 대상자원 비즈니스 컴포넌트의 경우에는 대상자원 명세에 따라 여러 종류의 서브 대상자원이 가능하기 때문에 여러 개의 인터페이스와 컴포넌트가 가능하며 이들간에 다양한 매핑 관계가 가능하다.

(그림 17)의 정제된 매핑을 보면 DepositAccount 대상자원 컴포넌트는 먼저 CheckingAccount, SavingAccount, InstallmentSavingAccount등과 같은 계좌 상품별 컴포넌트들로 세분화 되었으며 이들은 모두 하나의 IDEPAccountMgr 비즈니스 인터페이스를 공유한다. 이는 여러 개의 컴포넌트가 한 개의 인터페이스를 공유하는 1: M매핑을 보여주는 것으로서 각 계좌 상품들이 공통으로 제공해야 하는 인터페이스이지만 인터페이스를 구현할 때는 각 계좌 상품별로 다르게 구현되는 방식이다. 두 번째로 계좌 자체에 대한 서비스를 제공하는 Account비즈니스 컴포넌트로 세분화 되었다. Account컴포넌트는 계좌 자체에 대한 서비스를 제공하는 인터페이스와 계좌 상품별 컴포넌트에서 필요로 하는 서비스를 제공하는 인터페이스로 세분화 되며 각각 IDEPAccountQueryMgr, IDEPSubAccountTypeMgr로 명명된다. 이 매핑은 한 개의 컴포넌트가 여러 개의 인터페이스를 갖는 M:1 매핑 관계를 보여주고 있다. 마지막으로 모든 계좌 상품들의 공통로직을 제공하는 AccountTypeCommon 컴포넌트가 존재하며 IAccountTypeCommon로 명명된 인터페이스를 가진다. 이는 한 개의 컴포넌트가 한 개의 인터페이스를 갖는 형식이다.

(그림 18)은 식별된 컴포넌트와 인터페이스간의 매핑을 기반으로 수신 서브시스템의 컴포넌트 아키텍처 다이어그램이다. 수신 서브시스템에서 제공하는 서비스를 사용하는 모든 클라이언트는 시스템 컴포넌트 DepositManager가 가지는 IDEPSIMgr인터페이스를 호출함으로써 서비스를 제공받는다. DepositRule 비즈니스 규칙 컴포넌트는 대상자원 컴포넌트와 결과 컴포넌트를 조정한다. 따라서 대상자원 컴포넌트와 결과 컴포넌트간에는 직접적인 의존 관계가 존재하지



(그림 18) 수신 서브시스템 컴포넌트 아키텍처 다이어그램

않기 때문에 재 사용성이나 변경용이성이 뛰어나다. 계좌 상품별 컴포넌트들은 한 개의 비즈니스 인터페이스를 공유하고 있다.

계좌 상품들의 공통로직을 구현한 AccountTypeCommon 컴포넌트는 계좌 상품별 컴포넌트에서만 사용 가능하며 계좌 자체에 대한 서비스를 제공하는 Account컴포넌트는 2개의 인터페이스를 가지고 있는데 IDEPAccountQueryMgr는 규칙 컴포넌트에서 호출하며 계좌상품 컴포넌트들에서만 사용하는 IDEPAccountTypeQueryMgr가 존재한다.

5. 규칙 기반 분석 패턴 타당성 평가

기존에 제시했던 분석 패턴과 비교해 본 논문에서 제시한 분석 패턴은 먼저 비즈니스 컴포넌트 개발 시 액터로부터의 모든 외부 이벤트에 대해 규칙 기반의 비즈니스 프로세스를 위한 주요 개념과 그들 간의 관계를 효과적으로 표현할 수 있다는 것이다. 두 번째로 기존 분석 패턴이 개발 프로세스 상의 분석 단계에 중점을 두고 있는 반면에 본 논문에서 제시한 분석 패턴은 패턴에서 제시한 몇 개의 핵심 개념을 사용해 UML Components 상에서 어떻게 효과적인 컴포넌트 모델링을 수행하는지를 제시한다는 것이다.

본 논문에서는 제시한 분석 패턴이 가지는 장점에 대한 타당성 증명을 위해 규칙 기반으로 거래를 처리하는 은행 수신과 수출입 업무를 적용한 결과를 가지고 증명한다. (그림 19)는 제시한 분석 패턴이 가지는 타당성에 대한 요약이다. 먼저 제시한 분석 패턴은 (그림 13)과 같이 제시한 분석 패턴이 가지는 핵심 개념(예: Subject) 틀을 통한 수신 업무와 수출입 업무 중요 개념 인스턴스 (예: 점원, 수출입 점원)식별 할 수 있다. 두 번째로 (그림 12)와 (그림 13)과 같

규칙기반 분석 패턴 장점	타당성 증명 예시
규칙 기반 분석 틀	그림 13에서 분석 패턴에서 제시한 개념과 이를 수신 및 수출입 업무에 적용해 추출된 인스턴스 예제
명세개념과 실체 개념 분리를 통한 이벤트나 대상자원 확장 가능성	그림 11과 12에서 Event Specification 과 상속을 통한 실체 타입 확장예제
대상자원이나 결과 개념을 양적으로 측정하기 위한 단위 개념 추가	그림 11과 12의 동화 개념
대상자원의 상태를 표현할 수 있고 상속을 통한 타입 확장 가능성	그림 11 과 그림 12에서 계좌, 신용장, 유가증권의 종류에 대한 예제
UML Components 개발 프로세스 상에서 규칙, 결과, 그리고 대상의 3가지 주요 개념을 사용한 인터페이스 및 컴포넌트 식별 용이성	4.3과 4.4절의 은행 수신 업무를 통한 인터페이스 및 컴포넌트 식별 예제를 통한 타당성 증명
분석 패턴을 통한 일관된 산출물	규칙 기반 분석 패턴을 사용해 서로 다른 업무 분석가가 수신 및 수출 업무를 분석한 비즈니스 개념 모델 비교(그림 11과 12의 비교)
컴포넌트의 재사용성 및 적응성	가변성 관리를 위한 별도 컴포넌트

(그림 19) 규칙 기반 분석 패턴 타당성 증명 예시

이 수신 거래 혹은 수출거래 타입 명세(예: Event Specification)로부터 여러 종류의 수신 이벤트 타입들(예: 계좌신규, 출금, 송금) 혹은 수출 이벤트 타입들(예: 신용장 통지, 네고, 추심) 확장 가능하다는 것이다. 세 번째로 (그림 11)과 (그림 12)에서 계좌, 신용장, 유가증권을 측정하기 위한 통화 개념이 있다는 것이다.

네 번째로 (그림 11)과 (그림 12)와 같이 수신업무에서 계좌의 상태에 따라 정상계좌와 비정상 계좌로 구분 가능하고 또한 정상 계좌나 비정상 계좌 타입을 상속을 사용해 확장(예: 해지계좌, 사고계좌, 정기적금, 당좌예금)할 수 있으며 수출입 업무의 경우에도 신용장과 유가증권에 대한 상태 표현(정상 및 비정상 유가증권)과 상속을 통해 확장 가능(예: 매입제한 신용장, 기한부 신용장)하다.

다섯 번째로 4.3과 4.4에서 설명한 것처럼 UML Components 개발 프로세스 상에서 규칙, 결과, 그리고 대상의 3가지 주요 개념을 사용한 인터페이스 및 컴포넌트 식별을 용이하게 할 수 있다는 것이다. 여섯 번째로 (그림 11, 12)와 같이 규칙 기반 분석 패턴을 사용해 서로 다른 업무 분석가가 수신 및 수출 업무를 분석한 비즈니스 개념 모델 산출물을 비교해 보면 알 수 있듯이 분석 패턴을 사용하면 일관되고 가독성이 좋은 산출물을 얻을 수 있다. 마지막으로 가변성 관리를 위해 룰 컴포넌트를 식별함으로써 컴포넌트의 재사용성이나 적응성을 좋게 할 수 있다.

6. 결 론

본 논문에서는 규칙을 기반으로 비즈니스 내부 개념들과 이들 사이의 관계를 추출하기 위한 개념적인 분석틀을 제시했으며 또한 제시한 분석 패턴이 가지는 주요 개념들을 기반으로 UML Components 개발 프로세스 상에 적용하는 방법을 은행 수신업무 적용 사례를 통해 기술했다.

제시한 분석 패턴의 장점은 먼저 하나의 시스템 안에 여러 개의 서브시스템을 가지는 복잡한 비즈니스 어플리케이션 상황에서 각각의 분석가들이 규칙 기반 분석 패턴을 통해 표준화되고 일관된 비즈니스 개념들을 식별할 수 있다. 두 번째로 분석패턴이 제시한 주요 개념들을 사용해 UML Components와 같은 CBD개발 프로세스상에서 효과적으로 비즈니스 컴포넌트를 개발 할 수 있으며 산출물의 가독성과 일관성을 좋게 한다. 세 번째로 비즈니스 프로세스적인 규칙이나 비즈니스 도메인 규칙과 같은 가변적인 부분들을 별도의 규칙 컴포넌트에서 관리하도록 함으로서 컴포넌트 재사용성이나 적응성을 높일 수 있다.

참 고 문 헌

[1] Jeong Ah Kim, YoungTaek Jin, SunMyung Hwang: A Business Component Approach for Supporting the Variability of the Business Strategies and Rules. ICCSA (3)

2005: 846-857.
[2] M.Morisio and C.B.Seaman et al, Investigating and improving a COTS-based software development process, ICSE 2000, pp.31-40, 2000.
[3] Lars Geyer and Martin Becker, "On the influence of Variabilities on the Application-Engineering Process of a Product Family," Proceeding of SPLC2, 2002.
[4] S. Konrad, Betty H.C. Cheng, Laura A, Campbell, "Object Analysis Patterns for Embedded Systems," IEEE Transaction on software engineering, vol.30, no.12, December, 2004.
[5] S. Konard, B.H.C. Cheng, L.A. Campbell, and R. Wassermann, "Using Security Patterns to Model and Analyze Security Requirements,"Proc. Requirements for High Assurance Systems Workshop (RHAS ' 3), Sept., 2002.
[6] A. Geyer-Schulz and M Hashler, "Software Engineering with Analysis Patterns." 2001, http://wwwai.wu-wien.ac.at/~hahsler/research.
[7] E.B. Fernandez and X. Yuan, "Semantic Analysis Patterns." Proc. 19th Int'l Conf. Conceptual Modeling(ER 2000), PP. 183-195, 2000.
[8] M. Fowler, Analysis Patterns: Reusable Object Models. Addison-Wesley, 1997.
[9] Sterling Software Component-Based Development Method, http://www.sterling.com.
[10] D.Gross and E.S.K. Yu, "From Non-Functional Requirements to Design through Patterns," Requirements Eng., vol.6, no.1, pp.18-36, 2001.
[11] S. Robertson, "Requirements Patterns via Events/Use Cases," 1996,http://www.systemsguild.com/ GuildSite/SQR/Requirements_Patterns.html.
[12] A.G. Sutcliffe, N.A. Maiden, S. Minocha, and D. Manuel, "Supporting Scenario-Based Requirements Engineering," Software Eng., vol.24, no.12, pp.1072-1088, Dec., 1998.
[13] A.Dardenne,A.van Lamsweerde, and S. Fickas, "Goal-Directed Requirements Acquisition," Selected Papers Sixth Int'l Workshop Software Specification and Design, pp.3-50, 1993.
[14] M. Shaw, "Some Patterns for Software Architectures," Pattern Languages of Program Design vol.2, pp.255-269, 1996.
[15] W. Keller, "Object/Relational Access Layers-A Roadmap, Missing Links and More Patterns." Proc. EuroPLoP 1998 Conf, July, 1998.
[16] M. Adams, J. Coplien, R. Gamoke, R Hanmer, F. Keeve, and K. Nicodemus, "Fault-Tolerant Telecommunication System Patterns." Proc. Secon Conf, Pattern Language of Program, Sept., 1995.
[17] Uwe Zdun, Markus Völter, Michael Kircher: Pattern-Based Design of an Asynchronous Invocation Framework for Web Services. Int. J. Web Service Res. 1(3): 42-62 (2004).
[18] R.Darimont and A.van Lamsweerde,"Formal Refinement Patterns for Goal-Driven Requirements Elaboration," Proc. Fourth ACM SIGSOFT Symp. Foundations of Software Eng., pp.179-190, 1996.
[19] D. Lea, "Design Patterns for Avionics Control System," Technical Reports ADAGE-OSW-94-01, DSSA Adage Project, 1994.
[20] B.P. Douglass, Real-Time Design Patterns. Addison-Wesley, 2003.
[21] Nierstrasz Oscar, Meijler Theo Dirk, "Research Directions in Software Composition," ACM Computing Surveys, Vol.27, No.2, pp.262-264, June, 1995.
[22] Jim Q, Ning, "Component-Based Software Engineering," IEEE Software, 1997.
[23] Jan Bosch, Superimposition: A Component Adaptation Technique, Information and Software Technology, 41(5): 257-272, March, 1999.
[24] Urs Holzle. "Integration Independently-Developed Components In Object-Oriented Languages," Proceedings of ECOOP'93, Springer Verlag LNCS 512, 1993.

이 용 환

Email : yhlee@konkuk.ac.kr
1997년 건국대학교 행정학과 졸업(학사)
1999년 건국대학교 대학원 컴퓨터 공학부(공학석사)
2006년 건국대학교 대학원 컴퓨터 공학부(공학박사)
2003년~2005년 (주) 인터넷 커머스 코리아 연구소장
2005년~2006년 동덕여대 겸임교수
2006년~현재 건국대학교 연구교수로 재임
관심분야 : CBD, Ubiquitous Computing, Embedded Software

민 덕 기

Email : dkmin@konkuk.ac.kr
1986년 고려대학교 산업공학과 졸업
1991년 Michigan State 석사
1995년 Michigan State박사
1995년 건국대학교 교수로 부임
2006년 현재 건국대학교 교수로 재임
관심분야 : Distributed System, Ubiquitous Computing, Home Networking