룰 기반 컴포넌트 개발 기법 및 사례

김 정 아[†] · 황 선 명[††] · 진 영 택[†††]

요 약

컴포넌트 설계 시에 컴포넌트의 확장성 및 재사용성을 확대하기 위해서는 비즈니스 어플리케이션 개발 과정에서 발견된 가변성을 별도의 룰로 정의할 필요가 있다. 인터페이스의 래핑이나 구현 클래스의 재정의를 통한 컴포넌트의 개조 기법은 컴포넌트의 재사용을 지원하는데 한계가 있기 때문이다. 따라서, 컴포넌트 개발 과정에서 향후 컴포넌트의 재사용성을 고려한 설계가 필수적이다. 본 연구에서는 컴포넌트로 부터 가변적 특성을 분리하여 룰 컴포넌트를 포함하도록 기존의 컴포넌트 아키텍처를 확장하였으며, 룰 정의에 필요한 구문을 정의하였다. 또한 보험 판매 시스템에 적용하여 룰의 재정의를 통한 컴포넌트의 재사용성을 검증해 보았다.

Rule based Component Development Technique and Case study

Jeong Ah Kim[†] · Sun Myung Hwang[††] · Young Taek Jin[†††]

ABSTRACT

In order to increase extendibility and reusability of components during component design, the variability discovered in a business application development needs to be defined to separate rules. That is because component adaptation techniques through redefinition of implementation classes and interface wrapping have limits to support the component reusability. Therefore, It's essential to design the component which takes into account the future reusability in the component development. In this paper, we extended the existing component architecture to incorporate rule components by separating variable properties from the components and defined the necessary syntax for the rule definition. In the case study, we built the business components for an insurance sales application and verified the component reusability through the rule redefinition.

키워드 : 컴포넌트(Component), 가변성(Variability), 룰 기반(Rule-based), 제품 공학(Product Line Engineering)

1. 서 론

소프트웨어 산업이 급속하고 다양한 형태로 발전해가고 기업간 경쟁이 더욱 심화되면서 소프트웨어 재사용성, 적시성, 유지 보수성 등이 중요한 요인으로 대두되면서 컴포넌트 소프트웨어 기술이 점차 각광을 받고 있다. 컴포넌트를 기반으로 하는 개발(CBD: Component Based Development)과 제품 계열 소프트웨어 개발(Product Line Software Development)은 표준 컴포넌트 기술을 이용하여 개발된 컴포넌트를 획득하여 조립하므로서 생산성과 품질을 높이면서 소프트웨어 제품을 만들 수 있는 유망한 방법이다. 이러한 개발은 기존의 방법과 달리 많은 노력이 요구 사항, 테스팅 및 컴포넌트의 조립과 통합에 관련한 아키텍처를 기반으로 개발하게 되므로, 상대적으로 적은 노력이 코드 설계에 들어가는 다른 관점을 제시한다[1, 17]. 비즈니스 측면에서 볼 때 비즈니스 조직은 계속적으로 변경되는 비즈니스 요구 사항 및 환경 그리고 프로세스 또는 워크플로우에 적절하게 대응해야 하며 새로운 비즈니스 요구에 신속하게 적응하기 위한 비즈니스 프로세스의 재사용을 위한 필요성이 존재한다[2]. 아울러 컴포넌트는 비즈니스 어플리케이션 또한 기존 프로세스의 변경, 새로운 프로세스를 생성하기 위한 기존 프로세스의 재배열 및 새로운 비즈니스 프로세스에 즉각 적응할 수 있도록 설계되어야 한다. 이러한 소프트웨어와 비즈니스 두 측면의 요구사항을 반영하기 위해서는 컴포넌트의 재사용성 뿐만아니라 컴포넌트의 적응성 또는 확장성의 특성 역시 중요한 요소이다[3]. 컴포넌트는 특정한 기능을 수행하기 위해 컴포넌트 모델에 의거 독립적으로 개발, 보급되고 잘 정의된 인터페이스를 가지며 다른 부품과 조립되어 응용시스템을 구축하기 위해 사용되는 소프트웨어의 단위라고 정의될 수 있다[4]. 인터페이스 기반의 재사용이 가능한 컴포넌트는 재사용 될 애플리케이션의 비즈니스 요구사항의 상이성으로 인해 인터페이스의 수정 및 구현 방식의 변경이 필요한 경우가 많다. 컴포넌트는 바이너리 형태의 재사용을 목표로 하기 때문에 구현 내부의 수정을 통해 재사용이 어렵기 마련이다. 그러므로 컴포넌트의 설계시 컴포

† 종신회원 : 관동대학교 컴퓨터교육과 교수
†† 종신회원 : 대전대학교 컴퓨터공학과 교수
††† 종신회원 : 한발대학교 정보통신 · 컴퓨터공학부 교수
논문접수 : 2004년 3월 26일, 심사완료 : 2005년 1월 31일

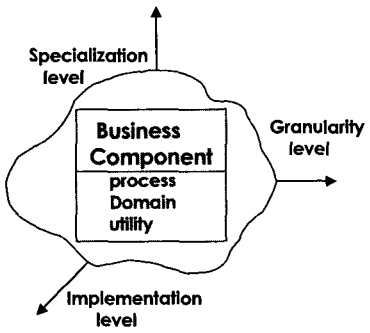
넌트의 확장성 및 컴포넌트 개발 당시와 상이한 요구 사항에 적응할 수 있는 적응형 컴포넌트의 개발이 필수적이다. 지금까지는 컴포넌트의 재사용 과정에서 일어나는 인터페이스의 불일치, 구현 방식의 문제를 해결하기 위해서 래핑(Wrapping) 기법이나 BCA(Binary Component Adaptation) 기법을[5] 적용해왔다. 이 기법은 서비스 중의 컴포넌트 변경 즉 동적 컴포넌트의 개조를 지원하지 못한다는 단점과 소스 코드 수준의 변경 없이는 추가 요구 사항을 완전하게 수용할 수 없다는 제한 점을 갖는다. 컴포넌트 자체의 재사용성을 높이기 위해서는 설계 단계 부터 컴포넌트의 가변성을 고려한 설계가 필수적이다[17]. 본 논문에서는 도출한 가변성을 단순하게 또 다른 클래스로 표현하는 방식은 한계가 있다고 보고, 가변성을 클래스 코드와 불리된 비지스니 룰로 표현하여 해결하고자 하였다. 즉, 여러 애플리케이션 마다 달라 질 수 있는 가변적 요소를 비즈니스 룰의 가변성으로 본 것이다. 이로써 상이한 비즈니스 요구 사항이 생기면 룰의 변경을 통해서 처리하고자 하였다. 본 논문에서는 (1) 있는 그대로의 재사용을 가능하게 하는 컴포넌트와 가변적 요소를 담고 있는 룰 컴포넌트를 분리한 컴포넌트 아키텍처를 제안하고 (2) 룰을 정의하는데 필요한 룰 구문을 정의하며, (3) 룰을 처리하기 위해 필요한 룰 엔진을 소개 하며, (4) 룰 기반의 컴포넌트 개발 개념을 적용한 사례를 소개한다. 본 논문의 구성은 2장에서는 일반적인 CBD방법과 컴포넌트 재사용 시 적용할 수 있는 기존의 컴포넌트 개조 기법을 정리하고 3장에서는 룰 기반의 컴포넌트 아키텍처와 룰 정의 방법을 제시한다. 4장에서는 보험 판매 시스템 개발에 적용한 사례와 효과를 정의하고 5장에서 결론을 맺는다.

2. 컴포넌트 기반 개발 방법 및 개조 기법

2.1 일반적인 CBD 방법

컴포넌트는 크케 비즈니스 컴포넌트와 애플리케이션 컴포넌트로 구분할 수 있는데, 비즈니스 컴포넌트는 비즈니스 도메인에 공통의 컴포넌트로 그 재사용성의 가치가 높다.

비즈니스 컴포넌트의 정의는 제품, 방법 및 기본 기술에 따라 가변적이지만 (그림 1)에서 제시된 바와 같이 다음 세 가지 관점으로 일반적으로 분류될 수 있다[6].



(그림 1) 비즈니스 컴포넌트의 분류

2.1.1 컴포넌트의 규모(granularity)

컴포넌트의 규모는 단위 언어 클래스에서 비즈니스 컴포넌트 시스템, 비즈니스 컴포넌트 시스템의 연합까지 다양하다.

2.1.2 컴포넌트의 특성

컴포넌트는 특정한 도메인 및 조직에서 사용될 수 있는 것에서부터 도메인간 또는 조직간에서 공통적으로 사용될 수 있는 컴포넌트가 있다.

2.1.3 컴포넌트의 구현

비즈니스 컴포넌트는 컴포넌트를 구성하는 비즈니스 객체의 집합으로서 실세계의 비즈니스 개념을 구현하며 자치적이고 독립적이며 대규모 크기이다. 또한 비즈니스 엔티티 컴포넌트, 비즈니스 프로세스 및 비즈니스 유틸리티 컴포넌트로 구성되며 각각 다른 특성을 갖는다[7, 8, 9].

<표 1>은 일반적인 CBD의 컴포넌트 아키텍처를 나타낸다.

〈표 1〉 일반적 CDB 아키텍처

컴포넌트	역할
User Interface Component	사용자에게 보여지는 화면 요소
User Dialog Component	사용자와 시스템 사이 또는 사용자 인터페이스 요소 간의 일반적 상호 작용을 담당
Application Component	애플리케이션 별 차별화된 비즈니스를 처리를 담당하는 컴포넌트로 애플리케이션의 유지 보수성의 증대에 기여하는 목적이 강함
Business Process Component	비즈니스 트랜잭션을 처리하는 컴포넌트로 단위 비즈니스 서비스간의 업무 흐름과 처리 조건을 정의
Business Domain Component	비즈니스 도메인별 단위 서비스를 제공
Data Component	비즈니스 데이터 처리에 관련된 서비스를 제공

2.2 컴포넌트의 재사용 시 적용 가능한 개조 기법

컴포넌트 기반 소프트웨어 재사용의 기법은 개조, 수정, 조립으로 구분한다[10, 11]. 필요한 컴포넌트 개조, 수정 및 조립의 개념 및 방법을 정의한다.

1) 컴포넌트 수정 : 수정은 컴포넌트의 기본 특성 중의 하나인 프라퍼티(Property)에 의해 이루어지는 컴포넌트의 변경 과정을 의미한다. 프라퍼티란 컴포넌트의 범위와 행위를 결정짓는 특성으로 컴포넌트를 재 사용하여 어플리케이션을 설계 할 때 그 값을 변경할 수 있다.
2) 컴포넌트 조립 : 컴포넌트를 재 사용하기 위한 가장 자연스러운 과정으로 컴포넌트들끼리 합성(Composition)하는 경우와 다른 어플리케이션들과 통합(Integration)을 의미한다.

3) 컴포넌트 개조(Adaptation) : 컴포넌트가 어떤 이유로든 미리 정의된 인터페이스나 행위를 변경해야 할 경우를 의미한다. 간단하게는 인터페이스의 이름이 변경되거나 파라미터의 형식이 변경되는 것을 들 수 있다. 복잡하게는 새로운 인터페이스를 추가하는 경우도 있을 수 있다.

컴포넌트 개조는 일반적으로 다음의 방법으로 가능하다.

① 개조기(Adaptor) 기법 : 인터페이스의 차이를 보이는 두 컴포넌트 사이에 개조기 패턴을 활용하여 개조기 컴포넌트를 정의하여 해결한다. 이 개조기 컴포넌트가 차이나는 두 컴포넌트 사이의 인터페이스 불일치를 해결한다.

② Wrapper 기법 : 이는 인터페이스의 차이를 없애는 새로운 컴포넌트 정의하고 이 컴포넌트가 기존의 컴포넌트를 포함하여 관리하는 방식이다[12,13]. 지금의 바이너리 컴포넌트 기술에 있어서 가장 많이 사용되는 기술로서 Wrapping 컴포넌트는 아주 적은 변경이 필요한 경우에 생성되고, 기존의 컴포넌트에 정의된 기능이 필요하면 요구를 전달하기만 한다. 즉, Wrapping은 이름의 변경 또는 파라미터의 변경 등의 간단한 변경에 있어서 적용 가능하다. 그러나 빈번한 개조가 일어날 경우 개조된 컴포넌트의 크기가 기하 급수적으로 커질 수 있다는 단점 또한 있다.

③ Superimposition에 의한 개조 : 기본 개념은 개조를 해야 하는 컴포넌트와 개조를 처리하는 컴포넌트를 두 개로 분리하되, 둘을 하나로 묶어서 긴밀하게 동작시키도록 하려는 것이다[14]. 이는 개조의 대상이 되는 컴포넌트와 개조를 처리하는 컴포넌트 모두 독립적으로 재 사용하려는 목적을 갖는다. 이 방법은 컴포넌트의 개조 유형이 다양하지 않고 정해진 범위 내에서 이루어진 다는 전제 하에 가능한 방법이다. 이를 위해 언어를 처리하는 새로운 모델을 제시하였으며, 이 기법의 지원을 위해서는 기존의 언어에서 이루어지는 메소드 호출을 별도로 처리하는 기반 시스템을 구축해야 하기 때문에 일반적이지 못하다.

④ Binary Adaptation 기법 : UCSB(University of California at Santa Barbara)에서 개발한 방법으로 동적 로더(Dynamic Loader)를 개발하여, 로딩된 바이너리 컴포넌트를 직접 수정하는 방식이다[15]. 이를 위하여 Java의 클래스 로더를 수정하여 UC Santa Barbara 대학에서 개발한 Delta file 컴파일러를 사용해야 한다는 제약점이 있다. 즉, 일반적인 환경에서는 사용 불가능하다. 이러한 개념이 자바에서는 가능하나, EJB (Enterprise JavaBeans) 컴포넌트의 경우는 로더를 변경하는 것이 명세에 금지되어 있으므로 EJB에 바로 적용하기는 어려운 기법이다.

이러한 개조 기법은 모든 컴포넌트를 대상(물론 구현 언어별 차이는 있을 수 있음)으로 적용하는 기법이므로 어떤 컴포넌트에도 적용할 수 있다는 적용성을 갖기는 한다. 그러나 현실적으로는 그 효과가 미비하다는 제한점을 갖고 있다. 이에 본 논문에서는 컴포넌트 설계 부터 컴포넌트의 가변성을 고려한 설계가 필요하며 이를 실현 가능한 룰 기반의 컴포넌트 개발 기법과 룰의 변경을 통한 컴포넌트의 동적 개조를 제안한다.

3. 룰 기반 컴포넌트 개발

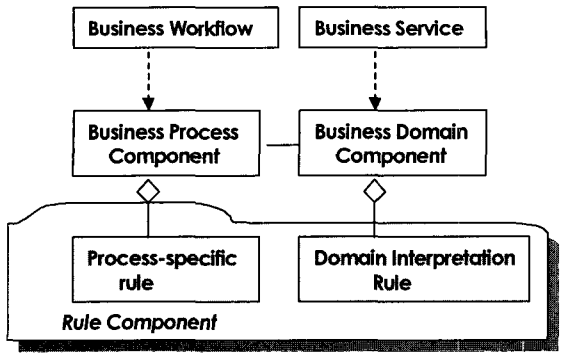
3.1 룰 기반 컴포넌트 아키텍처

컴포넌트 기반 개발을 수행할 때 어떤 단위로 컴포넌트를 개발해야 하는가의 결정은 매우 중요하고 어려운 일이다. 너무 범용적인 내용을 담는 다면 컴포넌트의 재사용성이 떨어질 것이고 너무 구체적 내용을 구현한다면 컴포넌트의 재사용의 범위가 적어질 것이다. 컴포넌트를 있는 그대로 재사용하는 것은 현실적으로 어려운 일이다. 컴포넌트의 수정을 통해서 재사용을 한다는 것 또한 컴포넌트 내부 구현의 이해를 요구하는 일이기 때문에 또다시 일반 모듈의 유지보수와 동일하게 어려워진다.

비즈니스 요구 사항은 시스템 개발중에도 개발 완료 후에도 지속적으로 변할 수 있는 부분이다. 가변적인 부분이 비즈니스 컴포넌트 안에 포함되어 있다면 변경이 발생할 때마다 코드를 수정해야 한다. 기본적으로는 가변적인 부분과 공통되는 부분을 분리하여 컴포넌트를 개발하고 가변적 부분을 별도로 재정의할 수 있도록 하는 것이 컴포넌트 재사용의 향상에 기여하는 길이다. 어떤 경우는 어떤 것이 가변적인 부분일지 조차 판단하기 어려울 때가 있기도 하다. 일반적으로 소프트웨어 있어서의 가변적인 부분은 모델의 변경과 모델 해석 방법의 변경이다. 구체적으로 정의한다면 속성의 추가, 오퍼레이션 이름의 변경 또는 추가, 오퍼레이션 구현 방법의 변경 등을 들 수 있다. 그러나 어떤 컴포넌트의 어떤 속성이 추가 될지, 그 이유는 무엇인지, 어떤 오퍼레이션이 새로운 이름으로 변경될 가능성이 있는 것인지를 모두 예측한다는 것은 너무 어려운 일이다. 이를 일반화하면 모델의 변경과 이의 해석 방법의 변경이라고 간주할 수 있다. 즉 어떤 컴포넌트이든지 변화 가능한 부분은 컴포넌트의 모델이 변경될 수 있고 모델을 해석하는 방법이 바뀔 수 있다는 것이다. 따라서 비즈니스 전략과 규칙의 가변성에 대해 유연하게 대응할 수 있도록 룰 컴포넌트를 비즈니스 프로세스 컴포넌트와 비즈니스 도메인 컴포넌트로부터 분리할 필요성이 제기된다. (그림 2)룰 컴포넌트의 역할을 나타낸다.

룰은 하나 또는 그 이상의 비즈니스 프로세스의 행위를 통제하는 규칙으로서 비즈니스 프로세스 룰과 비즈니스 도메인 룰로서 구성이 되며 룰 컴포넌트를 구성하는 요소가 된다.

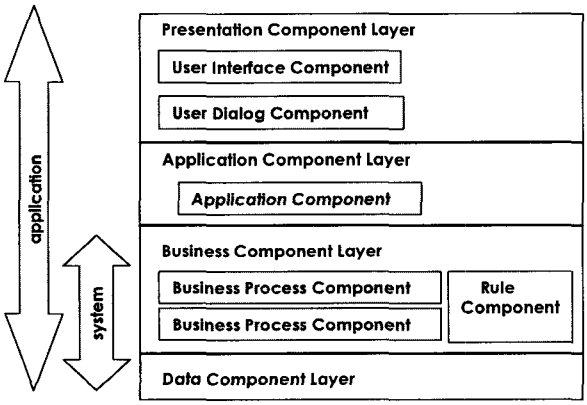
비즈니스 도메인룰은 대상이 가지고 있는 가변적 특성과 특성을 해석하는 가변적 방법을 정의한다. 고객의 나이를 구



(그림 2) 룰 컴포넌트의 역할 및 관계

하는 도메인 서비스는 문제의 특성에 따라서 보험 나이로 구할 수도 있고, 주민번호에 의한 법적 나이를 구할 수도 있다. 이는 동일한 고객 컴포넌트가 어떤 비즈니스 영역에 적용되는가에 따라 적용할 비즈니스 룰이 달라지는 예이다. 비즈니스 프로세스 룰은 하나의 업무를 처리하는데 필요한 작업 종류, 순서, 처리 조건을 정의하는 룰이다. 예를 들면, 보험 업무에서 가입 설계는 가입조건 심사, 보험료 구하기, 연금 구하기 순서로 진행한다는 것을 명시할 수 있다. 가입 조건 심사가 완료되야 다음 단계로 넘어가며 연금 상품일 때는 연금 구하기 절차를 따라야 한다. 이러한 모델 요소 중에서도 새로운 비즈니스 요구에 신속하게 적응하며 비즈니스 프로세스의 재사용 및 빠른 진화를 관리하기 위한 필요성이 존재한다. 동일한 보험 도메인에서도 각 애플리케이션의 성격에 따라서 심사의 순서와 심사의 진행 조건이 달라 질 수 있다. 이러한 작업 흐름의 가변적 규칙을 비즈니스 프로세스 룰로 정의할 수 있다.

이러한 가변적 비즈니스 처리 부분을 별도로 정의하여 안정된 모델을 구현한 비즈니스 컴포넌트와 가변적 특성을 처리하는 룰 컴포넌트를 분리한다. 이로써 가변적 특성의 룰의 변경을 통해 비즈니스 컴포넌트의 재사용성을 증대시킬 수 있다.



(그림 3) 룰 기반 컴포넌트 아키텍처

(그림 3)은 룰 컴포넌트가 추가된 룰 기반 컴포넌트 아키텍처를 정의한 것이다.

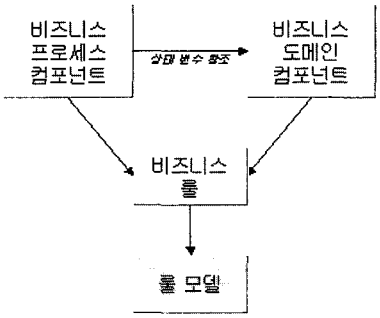
3.2 룰 컴포넌트

그림 3에서 보는 바와 같이 비즈니스 전략과 규칙의 가변적 내용을 유연하게 대응할 수 있도록 룰 컴포넌트를 비즈니스 프로세스 컴포넌트와 비즈니스 도메인 컴포넌트로부터 분리하였다. 룰 컴포넌트의 역할은 <표 2>에 정의하였다.

〈표 2〉 룰 컴포넌트의 역할

컴포넌트	역 할
Rule Component	컴포넌트의 서비스 부분 중 가변성을 갖는 비즈니스 규칙 및 전략을 분리한 컴포넌트로 컴포넌트의 확장성 및 유연성 증대를 보장함

각 컴포넌트간의 역할 및 책임을 정의해보자. 먼저 그림으로 컴포넌트간의 상호 연관성을 정의해보면 (그림 4)와 같다.



(그림 4) 컴포넌트와 룰 간의 관계

기존의 컴포넌트 아키텍처 상에서의 컴포넌트간의 상호작용은 비즈니스 프로세스 컴포넌트와 비즈니스 도메인 컴포넌트 사이의 상호 작용이였다. 프로세스 수행중에 필요한 도메인 서비스를 제공 받기 위해서 비즈니스 프로세스 컴포넌트는 비즈니스 도메인 컴포넌트를 활용하는 구조였다. 본 논문에서 추가한 룰 컴포넌트를 활용하면, 컴포넌트의 처리 과정에서 가변적 요소의 해석을 위해서는 비즈니스 룰을 담고 있는 룰 컴포넌트를 호출하여 룰의 해석 결과에 따라서 다음의 처리 흐름 또는 처리 결과를 만들어내게 된다. 또한 이러한 룰의 해석을 위해서는 룰 정의에 필요한 룰 모델을 참조하게 된다.

룰은 다음과 같은 기준으로 정의할 수 있다.

3.2.1 룰의 기본 구문

룰도 하나의 객체와 동일하게 속성과 오퍼레이션을 갖도록 정의하였다. 속성은 XML을 기반을 정의하도록 하였으며, 오퍼레이션은 JSR94[16]의 기본 명세를 만족하도록 정의하였다. 단 JSR94의 명세와 다른 점은 ELSE와 Additional Info 구문을 추가하였다는 것이다. 실제 구현에 활용해 본 결과 룰의 해석 결과만 반환하여서는 그 결과에 대한 원인을 제공할 수 없다는 한계점 때문에, 반환 결과값에 대

한 부가적 정보를 제공하는 것이 효과적이였기 때문이다.

```
반환타입 룰 헤딩 {
        IF ( 조건)
        THEN
        ELSE
        ADDITIONALINFO
}
```

3.2.2 룰 헤딩

반환타입 룰이름(파라미터)

룰 헤딩은 함수 헤딩과 동일한 구조를 갖는다. 룰이름은 반드시 하나의 도메인에서는 식별성을 가져야 한다.

3.2.3 조건식

조건식은 ( ) 안에 정의해야 하며, 참과 거짓을 판단할 수 있는 구문으로 사용가능한 오퍼레이터로는 다음과 같다.

① 논리 연산자 : and, or, !, ture, false,IsExist,
② 관계 연산자 : <, >, =, <=, >=, ~=
③ 기타 사용 가능한 함수 :
- CALL(함수이름) : 내장함수 및 사용자 함수를 호출
- FIRE(룰이름) : 다른 룰 호출

3.2.4 처리문

IF 문에서 조건이 true 일때는 THEN 구문이, false 일때는 ELSE 구문을 처리한다. THEN 구문이나, ELSE 구문의 내용이 처리된 후에는 그 리턴값을 반환하게 된다. THEN이나 ELSE의 리턴값은 룰 Heading에서 정한 리턴값과 동일해야 한다.

3.2.5 부가 정보 제공

ADDITIONALINFO 구문은 조건에 따른 처리와 상관없이 사용자에게 보여줄수 있는 추가적인 정보를 정의하기 위해 사용할 수 있다. "ADDITIONALINFO THEN" 구문은 THEN이 처리 될때 사용자에게 정보를 제공하기 위하여 사용한다. "ADDITIONALINFO ELSE" 구문은 ELSE 가 처리될때 사용자에게 정보를 제공하기 위하여 사용한다. "ADDITIONALINFO BOTH" 구문은 두 경우를 구분하지 않고 정보를 제공한다.

이렇게 정의한 룰은 자바 기반의 룰 엔진[17,18,19]을 통해 실시간으로 해석될 수 있다.

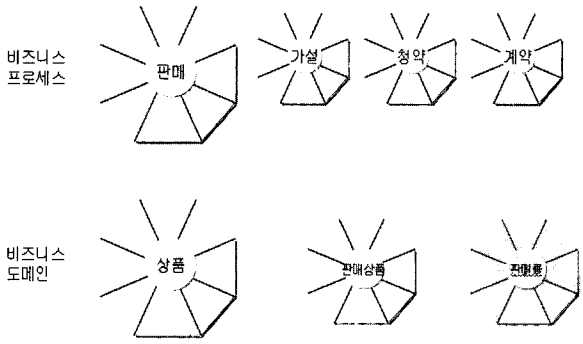
4. 룰 기반 컴포넌트 개발 사례

본 논문에서는 제안한 룰 컴포넌트의 개념 및 룰 컴포넌트를 통한 가변성 처리의 가능성을 보이기 위하여 보험 판매 시스템 도메인 영역에 룰 컴포넌트 개발 개념을 적용해 보았다.

4.1 보험 판매 시스템의 개요

보험 판매 시스템은 상품 등록, 가입 설계, 가입 설계의 결과 처리, 청약서 발행, 순으로 이루어진 업무로서 보험 판매를 위해서 보험을 가입할 경우 받을 수 있는 혜택에 대한 다양한 정보를 제공 받아서 상품끼리, 상품에서 가입조건끼리 비교할 수 있도록 해주며 보험 판매를 위해서 가입 설계, 청약과 같은 업무에서 새로운 보험 상품의 등록에서 관리에 이르기까지 다양하다.

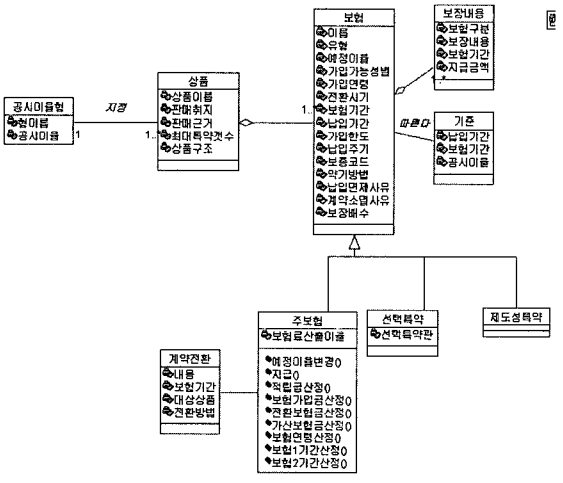
(그림 5)에서 정의한 것 처럼 보험 판매 시스템은 판매 프로세스와 상품 도메인으로 크게 구성되어 있다. 판매 프로세스를 좀더 세분화 하면 가설, 청약, 계약으로 구분할 수 있다. 본 논문에서는 프로세스와 도메인에 가변적 특성을 모아서 판매 룰을 정의하고 이를 룰 컴포넌트화 하여 프로세스의 가변성과 도메인의 가변성을 처리하도록 시도하였다.



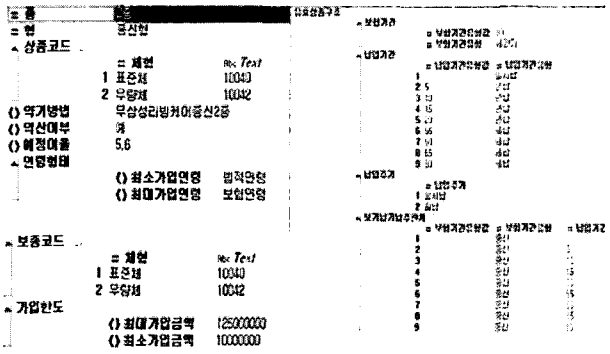
(그림 5) 보험 판매 시스템

4.2 룰 정의

보험 판매 시스템의 분석 과정을 통해 상품의 가변적이 특성과 가설, 청약, 계약 프로세스의 가변성을 룰로 정의하였다. 가변적 속성을 XML로 정의하였으며, 이의 해석의 가변적 특성을 룰 오퍼레이션을 정의하였다. (그림 6)은 가변적 속성을 모델로 정의한 것이며, (그림 7)은 모델에 대한 실제 룰 인스턴스를 정의한 것이다.



(그림 6) 가변적 속성에 대한 객체 모델



(그림 7) 룰의 속성 값이 채워진 인스턴스

(그림 6, 7)에 정의한 룰 속성을 해석하기 위해 정의한 룰 오퍼레이션의 예는 (그림 8)과 같다.

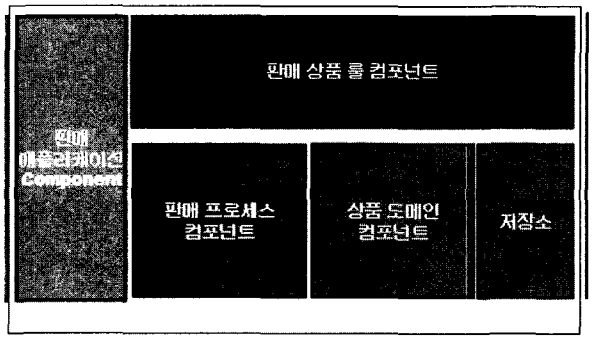
```
Boolean 가설_상품_선택특약_가입금액만원단위체크룰()
{
        IF( #상품.선택특약(*).계약금액 % 10000 == 0)
        THEN (Boolean true)
        ELSE (Boolean false)
        ADDITIONALINFO ELSE "선택특약 가입금액이 만원단위가 아닙니다."
}
Boolean 가설_상품_주보험_최대가입금액체크룰()
{
        IF( #상품.주보험.계약금액 <= 상품.주보험.가입한도.최대가입금액 )
        THEN (Boolean true)
        ELSE (Boolean false)
        ADDITIONALINFO ELSE "고객이 입력한 주보험의 가입금액(또는 보험료) "
                + #상품.주보험.계약금액 + "은 주보험의 최대가입금액인 "
                + 상품.주보험.가입한도.최대가입금액 + " 보다 큽니다."
}
Boolean 가설_상품_주보험_납기납후심사룰()
{
        IF ( (#상품.주보험.납입주기 == "일시납" and #상품.주보험.납입기간.유형 == "일시납") or
             (#상품.주보험.납입주기 != "일시납" and #상품.주보험.납입기간.유형 != "일시납") )
        THEN (Boolean true)
        ELSE (Boolean false)
        ADDITIONALINFO ELSE "납입주기나 납입기간 중 하나가 일시납이면 다른 하나도 일시납이어야 합니다."
}
```

(그림 8) 룰 오퍼레이션 예

4.3 보험 판매 시스템의 컴포넌트 아키텍처

(그림 5)에 정의한 범위의 보험 판매 시스템을 룰 기반으로 구현한 컴포넌트 아키텍처는 (그림 9)와 <표 3>에 정의하였다.

본 논문에서는 비즈니스 도메인 컴포넌트를Java클래스를 포함하는 package 구조로서 J2EE를 이용하여 개발하였다. 각 오퍼레이션의 파라메터는 XML을 기반으로 하고 명령에 따라서 파라메터 개수를 조정할 수 있다. EJB 컴포넌트로의 변환은 session bean만 추가되며 캡슐화를 제공하기 위해 jar 또는 war 파일 형태로 패키징 되어 배포될 수 있다.



(그림 9) 룰 컴포넌트가 적용된 아키텍처

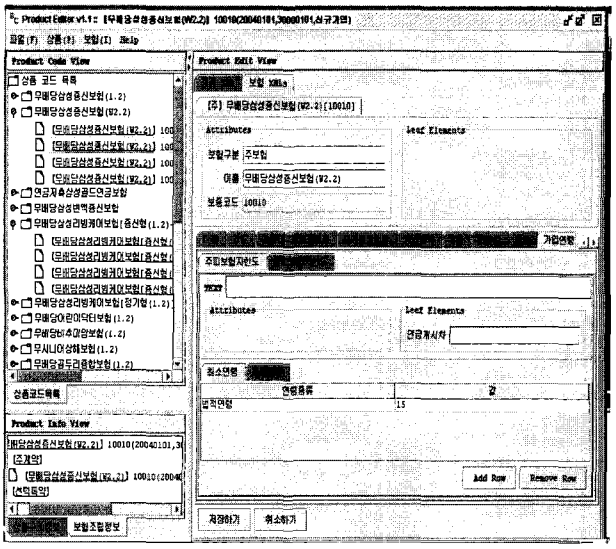
〈표 3〉 컴포넌트 아키텍처 상의 컴포넌트

비즈니스 프로세스 컴포넌트	판매	판매 프로세스에서 이루어지는 채널별 공통의 프로세스를 담당한다. 주로 가입조건 심사, 보험료 구하기, 보장내역 예시, 연금 및 해약환급금 구하기, 기타 부가 정보 구하기등의 워크플로우를 처리한다.
비즈니스 도메인 컴포넌트	상품	판매의 단위가 되는 상품별 기본 정보를 관리하는 컴포넌트로 판매를 위해 필요한 프로세스의 각 단계별 처리에 필요한 정보와 해석 방법을 제공한다.
룰 컴포넌트	판매룰	상품이 판매되기 위해 필요한 가입, 청약, 계약의 규칙과 판매 개념으로 상품을 해석하는 방법을 정의한다

4.4 룰 컴포넌트의 활용

새로운 비즈니스 요구에 신속하게 적응하고 비즈니스 프로세스의 변경, 재사용 및 빠른 진화를 관리하기 위한 필요성에 따른 룰 컴포넌트를 활용할 수 있다.

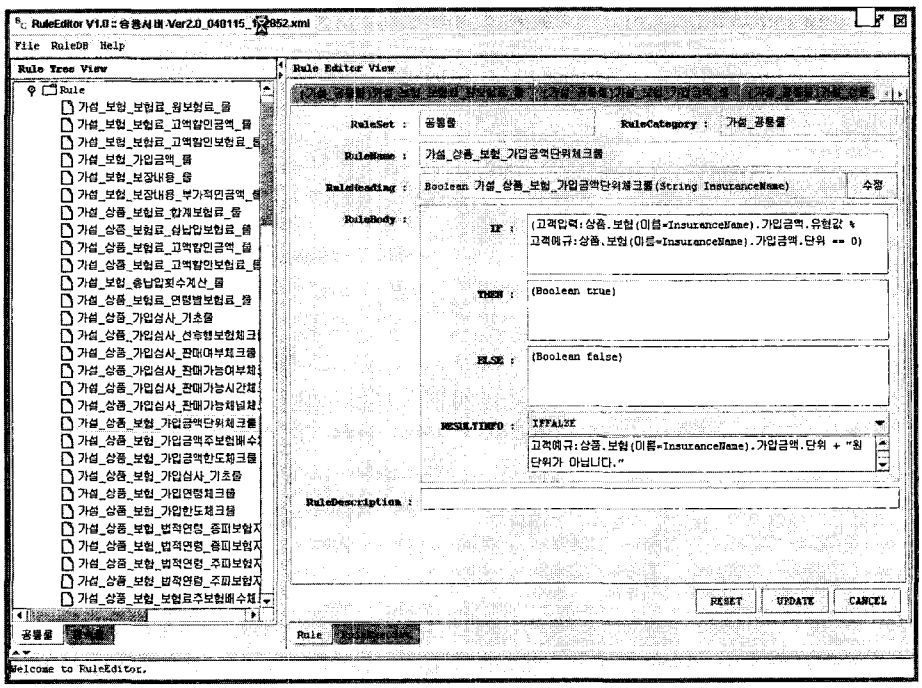
예를 들어서, 도메인 룰인 보험 가입 연령에서 법정 최소 가입 연령이 변경되었을 때 도메인 컴포넌트를 변경하지 않고 (그림 10)과 같이 룰 속성 값만 변경하면, 변경된 내용이 컴포넌트의 재 배포 없이 즉각적으로 반영될 수 있다.



(그림 10) 룰 속성 변경 화면

보험 가입 심사에서 보험 가입 금액 한도를 확인해야 하는데 한도를 확인하는 방법이 기계약 심사까지 포함해야 하는 경우라면 (그림 11)과 같이 해당 룰을 검색하여 룰 오퍼레이션을 변경하면 추후 컴포넌트 실행 과정에서 변경된 룰이 실행된다. 이는 애플리케이션의 변경된 요구 사항이 반영 된 결과이다.

비즈니스 가변성을 컴포넌트와 분리하여 별도의 룰로 정의하므로써 요구 사항의 변경이 발생할 경우, 컴포넌트의 변경 없이 룰의 변경을 통해 요구 사항을 수용할 수 있다.



(그림 11) 룰의 변경

룰로써 가변성을 처리하므로써 동적 컴포넌트의 수정이 가능하다는 장점이 있는 반면, 아직 까지는 룰 기반의 컴포넌트 개발의 적용 범위를 확정할 수 있지 못하다는 제한점을 갖고 있다. 본 논문에서 적용한 사례와 같이 가변성을 정형화된 룰 모델로 표현이 가능하거나 가변성이 명확한 경우에는 적용 가능한 것으로 판단할 수 있다. 그러나 룰로 표현할 수 있는 범위에 대한 추가 검증은 필요하다.

또한 룰을 적용하기 위한 가변성을 분석하고 모델링 하는 프로세스의 정립도 필요하다.

5. 결 론

비즈니스 요구 사항의 변경은 소프트웨어 개발과 유지 보수를 매우 어렵게 하는 원인이지만, 비즈니스의 목표와 환경이 변하기 때문에 일어날 수 밖에 없는 것이 현실이다. 개발 중에도 소프트웨어 시스템에 대한 요구 사항은 지속적으로 변화할 수 밖에 없다. 소프트웨어 시스템에 대한 지속적 요구 사항의 변화의 특성은 소프트웨어 개발 당시에 재구성 가능하고 확장 가능한 소프트웨어로의 개발을 요구 한다. 컴포넌트의 독립성으로 컴포넌트의 변경은 원인이 되는 컴포넌트에게만 국한시킬 수 있게 된다. 그러나 컴포넌트의 독립성 만으로는 소프트웨어의 확장성 및 유연성을 보장할 수 있는 것은 아니다. 컴포넌트를 설계함에 있어 가장 중요한 부분은 컴포넌트 내에 가변적인 부분을 담지 않는 것이다. 즉, 가변적인 부분을 별도로 분리하므로써 컴포넌트 운영 중에 코드의 수정이 아니라 분리된 가변적 부분만을 변경하여 변화된 요구 사항에 대처할 수 있도록 해야 한다. 룰 컴포넌트화 통해서 시스템 설계에 근본적으로 영향을 주지 않고 기존 프로세스의 많은 부분을 재사용하며 대부분의 변경 또는 재구성(configurability)요구가 단일 비즈니스 컴포넌트 또는 비즈니스 컴포넌트 조립에만 영향을 주도록 하는 이점을 취할 수 있다. 또한 프로세스의 변경이 미리 정의된 룰 및 룰 실행 집합을 통해서 미리 정의된 방법으로 관리될 수 있다. 따라서 재사용성과 확장성을 높이기 위해 비즈니스 프로세스 및 도메인의 가변적 특성을 룰로 명세하여 별도로 관리한다. 사례 연구로서 보험 업무의 상품, 청약 및 가입 설계에 대한 비즈니스 컴포넌트가 구축되었으며 이를 통하여 새로운 상품의 등록 및 관리와 같은 유사한 비즈니스 업무 개발에서 효율성이 입증되었다.

참 고 문 헌

[1] M.Morisio and C.B.Seaman et al, Investigating and improving a COTS-based software development process, ICSE 2000, pp.31-40, 2000.

[2] Herbert Weber, Asuman Sunbul and Julia Padberg, Evolutionary Development of Business Process Centered Architectures Using Component Technologies, Technical University Berlin.

[3] Greg Baster,Prabhudev Konana, and Judy E.Scott , Business Components:A case study of Bankers Trust Australia Limited, CACM, Vol.44, No.5, pp.92-98, 2001.

[4] Clemens Szyperski, Component Software-Beyond Object-Oriented Programming, Addision Wesley, 1998.

[5] Ralph Keller and Urs Hlzle, Binary Component Adaptation, Lecture Notes in Computer Science, Vol.1445, 1998.

[6] Jun Ginbayashi, Rieko Yamamoto and Keiji Hasimoto, Business Component Framework and Modeling method for Component-based Application Architecture, EDOC'00, pp. 184-193, 2000.
[7] Peter Herzum and Oliver Sims, The Business Component Approach, OOPSLA'98 Business Object Workshop IV, 1998.
[8] Peter Herzum and Oliver Sims, Business Component Factory, OMG Press, 2000.
[9] Hans Albrecht Scmid, Business Entity and Process Components, Business Object Design and Implementation III, OOPSLA'99 Workshop Proceedings, D. Patel, J. Sutherland and J.Miller(Eds).pp.131-145, 1999.
[10] Nierstrasz Oscar, Meijler Theo Dirk, "Research Directions in Software Composition", ACM Computing Surveys, Vol. 27, No.2, pp.262-264, June, 1995.
[11] Johannes Sametinger, "Classification of Composition and Interoperation", OOPSLA'96 Poster Presentation.
[12] Jim Q. Ning, "Component-Based Software Engineering" IEEE Software, 1997.
[13] Jim Q. Ning, "A Component-Based Software Development Model", in Proceedings of 21th Annual International Computer Software and Application Conference, 1996.
[14] Jan Bosch. Superimposition: A Component Adaptation Technique. Information and Software Technology, 41(5): 257-273, March, 1999.
[15] Urs Holzle. Integrating Independently-Developed Components in Object-Oriented Languages. Proceedings of ECOOP'93, Springer Verlag LNCS 512, 1993.
[16] JSR94, http://www.jcp.org/aboutJava/communityprocess/review/jsr094/
[17] Lars Geyer and Martin Becker, "On the influence of Varaibilities on the Application-Engineering Process of a Product Famliy", Proceedings of SPLC2, 2002
[18] Ahmed Abulsorour and Siva Visveswaran, "Business process automation made easy with Java", http://www.javaworld. com/javaworld/jw-09-2002/jw-0906-process_p.html
[19] Ernest Friedman-Hill, Jess in Action, Manning Publications Company, 2003

김 정 아

e-mail : clara@kd.ac.kr
1990년 중앙대학교 전자계산학과 (공학석사)
1994년 중앙대학교 컴퓨터공학과 (공학박사)
1996년 관동대학교 컴퓨터교육과 조교수
현재 관동대학교 컴퓨터교육과 교수
관심분야 : 재사용, CBD, Product Line, MDA 기술, 프로젝트 관리 및 프로세스 개선 등

황 선 명

e-mail : sunhwang@dju.ac.kr
1982년 중앙대학교 전자계산학과(학사)
1984년 중앙대학교 대학원 전자계산학과 (이학석사)
1987년 중앙대학교 대학원 전자계산학과 (이학박사)
2000년~현재 한국 S/W프로세스 심사인협회(KASPA) 이사
2000년~현재 한국정보처리학회 논문지편집위원
1997년~현재 ISO/IEC JTC7/WG10 한국운영위원
1998년~현재 한국정보통신기술협회 TTA 특별위원
1989년~현재 대전대학교 컴퓨터공학과 교수
관심분야 : 소프트웨어 프로세스 모델, 품질 메트릭스, 소프트웨어공학 표준화, 컴포넌트 품질측정, 테스팅방법론 등

진 영 택

e-mail : ytjin@hanbat.ac.kr
1981년 중앙대학교 전자계산학과(학사)
1983년 중앙대학교 대학원 전자계산학과 (이학석사)
1992년 중앙대학교 대학원 컴퓨터공학과 (공학박사)
1983년~1990년 한국 에너지기술연구소 연구원
1999년~2000년 호주 UTS대학교 컴퓨터공학부 교환 교수
1990년~현재 한밭대학교 정보통신·컴퓨터공학부 교수
관심분야 : 소프트웨어공학(객체지향 분석/설계, 설계패턴, 역공학, CBD, CBSE, Product Line)