유연한 금융 수수료를 위한 업무 규칙 기반 컴포넌트 설계

홍성우*, 김영갑**
*고려대학교 컴퓨터과학기술대학원 디지털정보공학과
**고려대학교 컴퓨터학과 소프트웨어시스템 연구실
e-mail : hongsw@paran.com, ygkim@software.korea.ac.kr

Design of Business Rule-Based Component for Flexible Financial Charge

Sung-Woo Hong*, Young-Gab Kim**
*Dept. of Digital Information,
Graduate school of Computer Science & Technology, Korea University
**Software System Lab. Dept. of Computer Science & Engineering, Korea University

요 약

최근 금융권의 수익 기반이 되고 있는 수수료는 다양한 형태의 규칙을 내포하고 있으며, 복잡성이 증가하고 있어 유연하고 동적인 수수료 구조가 요구된다. 이러한 요구 사항을 충족시키기 위해서 업무 규칙(business rule)이 활용될 수 있다. 본 논문에서는 은행권의 수수료를 분석하여, 수수료 부과 기준을 업무 규칙으로 정의하고, 이를 파라미터 드리븐(parameter driven) 방식의 룰 데이터베이스(rule database)로 설계하였다. 이를 통하여 복합 수수료를 즉시 적용할 수 있는 유연한 설계로 어플리케이션 구조를 단순화 할 수 있는 업무 규칙 기반 수수료 처리 컴포넌트를 설계하였다.

1. 서론

선진 은행들은 대부분 금융 산업의 통합(financial convergence)이라는 시대적 조류에 편승, 전통적인 예대마진(예금과 대출의 이자율 차) 중심에서 수수료 기반의 상품·서비스 등으로 수익 구조를 다변화하기 시작했다[1]. 고부가가치 서비스를 제공함으로 수수료 증대 중심의 수익 구조 창출 필요성이 대두 되고 있으며, 이를 통해 비이자 수익 중심의 안정적인 수익 기반을 확대하고 있다.

수수료는 고객에게 서비스를 제공하거나, 공공 기관 등을 대신해 공과금을 징수할 때와 같이 은행이 고객으로부터 부과하는 금액이다. 이는 고객이 온라인 거래와 같은 이벤트에 부과되거나 은행이 정한 주기에 따라 부과될 수 있다. 대량의 온라인 거래를 수반하는 금융 시스템에서는 다양한 유형의 수수료 개발과 실시간 반영은 은행의 수익 기반에 핵심적인 역할을 한다.

수수료 처리 기능은 유연하고 동적인 수수료 구조로 설계되어야 하며, 다양한 유형의 조건을 지원하여 그 결과로 부과될 최종 수수료를 결정할 수 있도록 하여야 한다. 동시에 수수료 계산은 빈번히 호출되는 부분이므로 높은 수준의 성능이 필요하다. 이러한 요구 사항을 충족하기 위하여 프로그램으로부터 수수료 기능 처리를 위한 기준을 분리하여, 규칙의 신설 및 변경 시에도 프로그램의 변경 없이 적용 할 수 있는 업무 규칙(business rule)이 활용되고 있다[2].

기존의 수수료 처리 기능은 각 수수료를 개별적으로 설계·구현하여 통합 관리의 어려움이 많고, 새로운 유형의 수수료를 즉시 적용하는 것에 어려움이 있다. 또한, 고객 정보와 같이 외부 컴포넌트와의 인터페이스를 통한 업무 규칙은 프로그램 내의 업무 로직(logic)으로 구현되므로 업무 규칙의 변경이 필요하면 프로그램의 변경이 수반되는 문제점이 상존 한다.

본 논문에서는 수수료 부과 기준을 업무 규칙으로 정의하고, 이를 파라미터 드리븐(parameter driven) 방식의 룰 데이터베이스(rule database)로 설계하였다.

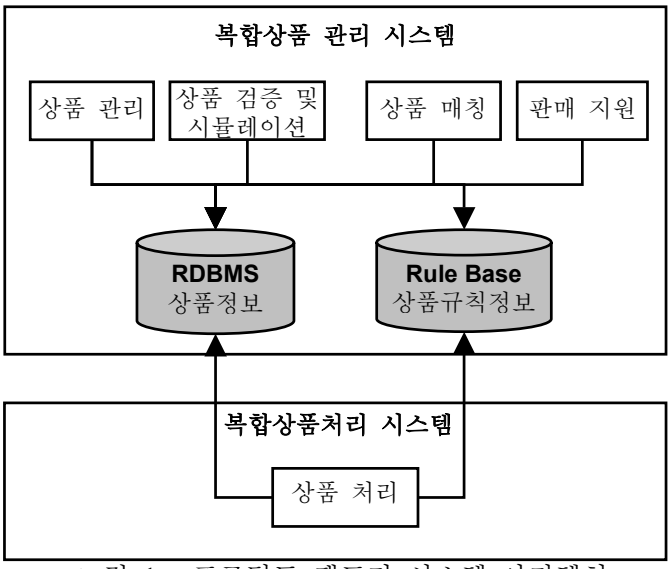
이를 통해서 다양한 수수료 기준을 업무 규칙으로 재정의하고, 업무 규칙을 프로그램에서 분리하여 규칙의 구조가 변경되더라도 메인 어플리케이션의 변경이 없도록 하여 수수료 컴포넌트의 유연성을 획기적으로 개선하고자 한다.

2. 관련연구

업무 규칙이란 비즈니스의 일부 국면을 제어하거나 정의하는 규약을 말한다[3]. 업무 규칙은 비즈니스의 변화가 요구되나 시스템의 변경이 어려운 경우, 새로운 법규와 지침으로 인하여 비즈니스의 지속성을 필요로 하는 경우등에 사용될 수 있다[4][5]. 최근에는 업무 규칙을 효과적으로 관리하고, 추론기능(inference engine)을 지원하는 BRE(Business Rule Engine) 솔루션인 Computer Associate's Aion, Fair Isaac's Blaze Advisor, ILog's JRules 등의 사용이 확대되고 있다[6].

<그림 1>과 같이 업무 규칙이 적용된 대표적인 금융 시스템으로서 프로덕트 팩토리(product factory) 연구가 있다[7][8]. 이 연구에서는 상품을 속성들의 집합으로 정의하고, 속성의 값 뿐만 아니라 속성의 집합에 변화, 즉 상품 규칙 정보를 이용하여 복합 상품을 설계한다.

복합상품을 설계할 수 있는 프로덕트 팩토리 시스템을 CA(Computer Associates)사의 CleverPath Aion Business Rules Expert 를 이용하여 구현 하였다.



<그림 1> 프로덕트 팩토리 시스템 아키텍처

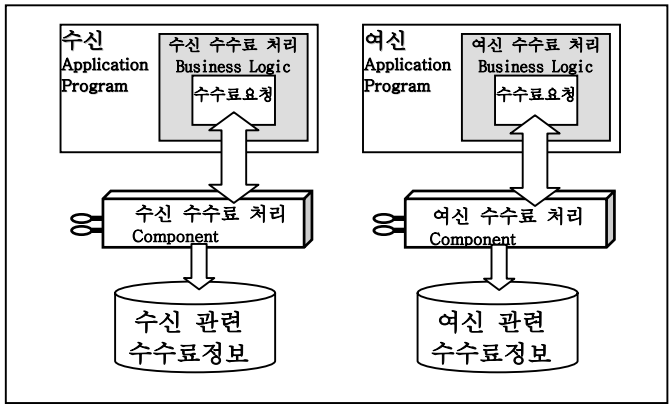
하지만, 업무 규칙 구현을 위해서 솔루션을 도입할 경우 솔루션 도입에 따른 비용의 문제, 솔루션에 종속적인 특화된 업무 규칙의 적용, 벤더(vendor) 종속적인 BRE 솔루션 적용에 따른 위험이 존재한다[9].

본 논문에서 제시할 수수료 처리 기능은 업무 규칙의 정의가 매우 구체적이며, 획일적인 업무 기반 시스템이 아니라 수수료에 특화된 업무 규칙을 가지고 있다. 또한, 추론 기능이 필요하지 않으며, 솔루션의 업무 규칙 시스템과 어플리케이션 통합에 어려움이 상존하므로 업무 규칙을 룰 데이터베이스에 구현하는 파라미터 드리븐 방식을 선택하였다.

3. 업무 규칙 기반 수수료 분석

3.1 수수료의 기능 요건

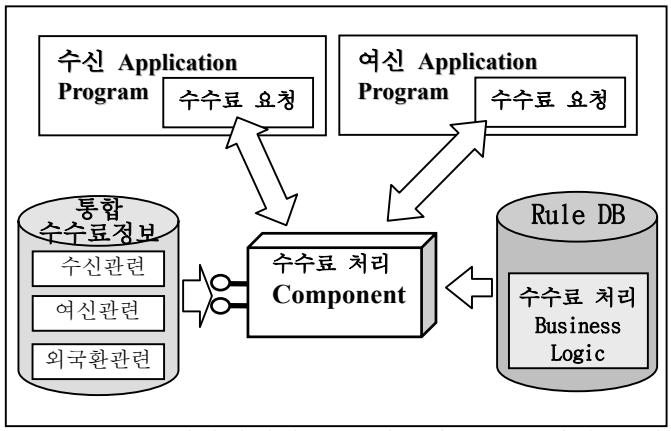
기존의 수수료 처리 기능은 <그림 2>와 같이 단위 업무별로 구성된 수수료의 기본 속성을 참조하여, 각 어플리케이션 내의 업무 규칙이 적용되어 수수료를 산출하는 방식이다.



<그림 2> 현행 수수료 처리 구성

개선된 수수료 처리 기능은 <그림 3>와 같이 각 업무의 업무 규칙을 룰 데이터베이스로 통합 관리하여, 어플리케이션에 내의 업무 규칙을 제거하여 어플리케이션에서는 수수료 요청만으로 수수료 산출이 가능하도록 설계 하였다.

관리자는 수수료 추가 필요에 따른 수수료만 정의하여 이를 온라인 거래와 연동하면 된다. 또한 사용자가 정의하는 업무 규칙을 활용해 다양한 고객의 등급에 따른 우대 수수료 정의 기능도 제공 하고, 동일한 수수료를 은행이 취급하는 복수의 통화로 정의할 수 있도록 하였다.



<그림 3> 룰 데이터베이스를 적용한 수수료처리 구성

3.2 수수료 유형 설계 분류

일반 은행에서의 수수료의 종류는 [표 1] 과 같이 분류할 수 있다[10]. 각 항목별로 수수료 부과 기준을 분석하여 수수료 컴포넌트 설계를 위한 수수료의 유형을 정의할 수 있다.

[표 1] 일반 은행의 수수료 유형

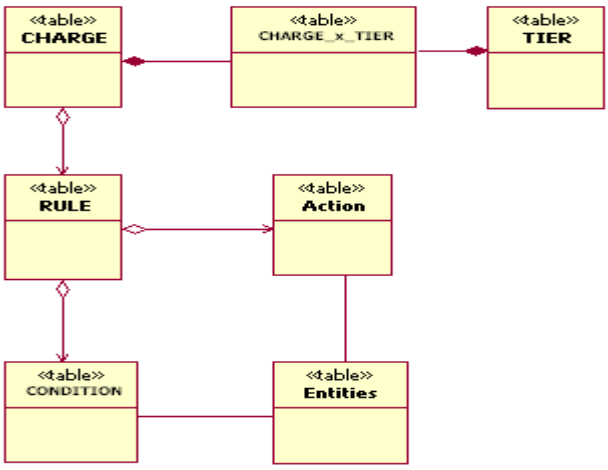
분 류	종 류
수신관련	- 자기앞수표 발행수수료 - 제 증명서 발급 수수료 - 부도처리 수수료 - 결제연장 수수료 등
폰/인터넷 뱅킹	- 폰/인터넷 뱅킹 서비스 수수료
환관련	- 송금수수료 - 타행 수표 추심 수수료 - CD,ATM 수수료 등
여신관련	- 지급보증서 발급 수수료 - 담보조사 수수료 - 담보변경 수수료 - 임대차 수수료 등
보호예수	- 보호예수 수수료
대여금고	- 대여금고 수수료
신용카드	- 신용카드 수수료
외국환	- L/C 발행 수수료 - 환가료 등 - 당· 타발 송금 수수료 등

기존에 구현된 방식인 프로그램 내에 업무 규칙이 존재하는 방식에서는, 각 수수료 항목별로 일률적인 수수료를 제외한 수수료를 산출하는 업무 규칙을 프로그램 로직으로 구현되었다.

본 연구에서는 [표 1]에서 제시된 수수료의 유형을 파라미터 드리븐 방식의 업무 규칙을 기반으로 설계하여 프로그램에서 수수료 산출을 위한 로직을 분리하여, 제시된 모든 수수료 산출 기능을 업무 규칙을 활용하여 설계할 수 있다.

3.3 업무 규칙 기반 수수료 유형 설계

부과될 수수료는 은행이 정한 일정한 업무 규칙과 수수료에 의해 결정된다. <그림 4>와 같이 업무 규칙은 평가 대상 조건(condition)과 수행될 실행(action), 조건과 실행에서 참조하는 연산자와 피연산자를 정의하는 엔티티(entities)로 구성된다. 또한, 거래 금액이나 거래 건수에 따라 수수료를 단계별로 적용할 수 있는 단계별 수수료(tiered charge)를 정의하고 있고, 각 단계별(tier)로 정액 또는 최소/최대 금액 범위를 정한 비율 수수료 방식을 가질 수 있다.



<그림 4> 업무 규칙 기반 수수료 모델 다이어그램

4. 업무 규칙 기반 수수료 컴포넌트 클래스 설계

특정 수수료 계산이 필요한 이벤트는 모두 수수료 처리 컴포넌트를 호출하여 수수료 금액을 구하게 된다. 업무 규칙은 IF <조건 1>…<조건 n> THEN <실행 1>…<실행 n> 의 형태로 기술 되며[11], 수수료 계산의 업무 규칙도 이를 따른다.

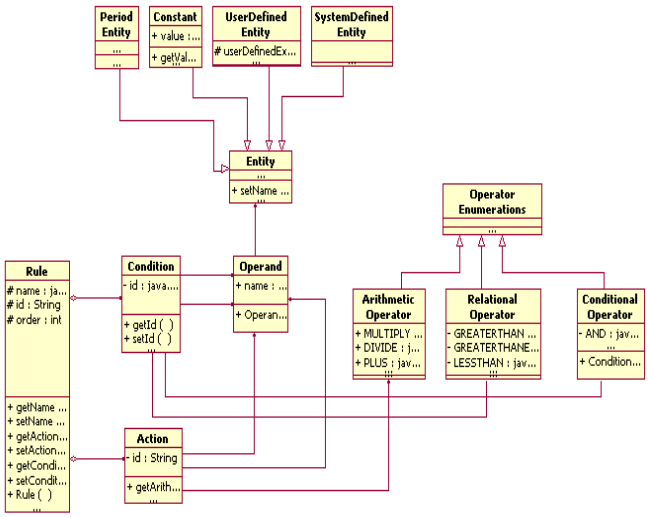
<그림 5>의 의사코드(pseudo-code)에서는 업무 규칙이 실행되는 과정을 나타낸다. 업무 규칙을 활용하는 수수료인 경우, 업무 규칙이 포함된 IF <조건> THEN <실행> 형식의 명령어가 생성된다. 조건은 왼쪽 피연산자와 오른쪽 피연산자를 비교연산자로 비교하여 수수료를 산출하는 실행문을 호출하도록 한다.

```
protected void calculateCharge(Id , Amount , cus_No,) {
/* Id : 수수료 번호,  Amount : 수수료 산출 금액,
cus_No: 고객번호, Arg_Id : 계좌 */

Get   Charge by Id;   /* 수수료 Id 로 수수료 정보 획득 */
Get   Type of Charge; /* 수수료의 종류 */
If      Type is Rule
Then   /* 업무 규칙을 사용하는 수수료 계산 */
        calculateRuleCharge(ruleId);
Else
        Get Value();
End     }
protected void calculateRuleCharge (ruleId, chargeAmount ) {
/* 룰 Id 로 수수료 업무 규칙의 조건 정보 획득 */
Get    Condition_ Table   by   ruleId;
/* 룰 Id 로 수수료 업무 규칙의 실행 정보 획득 */
Get    Action_ Table   by   ruleId;

/* 룰의 조건을 표현하는 If 문을 조건의 수 만큼 생성 */
/* 예) If (잔고>10000) AND (고객등급 = VIP) */
If ((leftOperands[i] Function rightOperands[i] )
    booleanOperators[i] (leftOperands[i+1]Function
    rightOperands[i+1]);
Then
/* 룰의 조건이 참(true)일 경우 수행되는 산술 연산하여
수수료 산출 */
/* 예) 수수료 = (잔고*0.3)+(거래금액*0.1) */
chargeAmount = (leftOperand[i] arithmeticOperator[i]
    rightOperand[i]) linkingOperator[i] (leftOperand[i+1]
    arithmeticOperator[i+1] rightOperand[i+1]);
 End     }
```

<그림 5> 수수료 업무 규칙 실행의 의사코드

<그림 6>의 업무 규칙 기반 클래스의 구성에서와 같이 업무 규칙은 조건과 실행으로 구성된다. 조건은 정의된 특정 엔티티와 상수 및 관련 연산자로 구성되어, 조건 연산자와 비교 연산자의 수행 결과에 따라 실행이 수행되거나 무시된다.

실행에서는 정의된 특정 엔티티와 산술 연산자로 구성되어, 실행이 수행되면 부과될 수수료가 산출된다. 피연산자는 잔액, 고객등급과 같은 시스템정의 엔티티(system-defined entities), 사용자 정의 엔티티 (user-defined entities), 고객 등급인 VIP 와 같은 상수(constant), 일자와 같은 기간 엔티티(period entites) 로 구성 된다.



<그림 6> 업무 규칙 기반 수수료 클래스 다이어그램

5. 기존 구현과의 장단점 비교

[표 2] 에서는 기존 방식인 수수료의 업무 규칙을 프로그램 내에 정의하는 방식과 본 논문에서 제시한 수수료의 업무 규칙을 룰 데이터베이스에 정의하는 방식을 비교하였다.

[표 2] 구현 방식에 따른 비교

구분	프로그램 방식	업무 규칙 방식
확장성	프로그램 내에 업무 규칙이 구현되어 있어, 추가요건의 수용이 제한적으로 적용 가능하다.	업무 규칙을 룰 데이터베이스에 통합 관리하므로 추가 요건의 적용이 간단하다.
생산성	수수료의 업무 규칙이 프로그램 내에 존재하므로 개발자의 기술력에 의한 편차가 크다.	수수료의 업무 규칙을 룰 데이터베이스에 정의하므로, 개발 및 변경의 단순화로 개발자에 의한 편차가 작다.
유지 보수성	프로그램 내의 업무 규칙관리로 수수료의 업무 규칙 관리에 어려움이 크다	신규 수수료 추가 및 변경 시에도 개별적인 룰 데이터베이스에 데이터 적용만으로 적용 가능
성능	개발자의 기술 역량에 따라 수수료 처리 성능의 편차가 발생한다.	수수료 업무 규칙이 룰 데이터베이스에 등록되므로 개발자에 따른 성능 편차가 적다.
재사용성	수수료의 업무 규칙이 통합적이고, 체계적으로 관리되지 못하여 재사용이 제한된다.	수수료의 업무 규칙이 통합되어 관리되며, 업무 규칙의 재활용 가능하다.

6. 결론 및 향후 과제

금융권의 수익 기반이 되는 대부분의 수수료는 고객이 온라인 거래 시 부과되므로 안정성과 신뢰성이 중요하며, 점차 복잡하고 다양한 유형의 수수료가 개발되고 있다. 이를 위해서 업무 규칙을 분석하여 구조화하는 업무 규칙 시스템의 사용이 확산되고 있다. 하지만, 업무 규칙 시스템의 솔루션을 이용하여 수수료 처리 기능을 적용하는 것에는 솔루션 도입의 비용 문제, 수수료 처리 업무 규칙이 제한적이고 특화되어 있다는 사실에 비추어 볼 때 제한적으로 사용 될 것이다.

본 논문에서는 솔루션을 적용하지 않고, 업무 규칙을 룰 데이터베이스에 구현하는 파라미터 드리븐 방식을 적용하여 금융 수수료를 데이터와 업무 규칙으로 정의하고, 수수료의 유형을 정액 수수료, 비율 수수료, 단계별 수수료, 업무 규칙 기반 수수료로 정의하였다. 이를 통해 은행권에서 취급되는 유형의 수수료를 산출하는 컴포넌트를 설계할 수 있었다.

특히, 수수료 연산에 필요한 정보를 룰 데이터베이스에 저장하여 수수료의 추가 및 변경 시 프로그램을 변경하는 대신 업무 규칙의 파라미터를 변경하거나, 룰 데이터베이스에 업무 규칙을 재정의하는 것으로 처리 할 수 있었다.

향후 연구 과제로는 업무 규칙을 적용하는 업무별로 생성되는 룰 데이터베이스를 통합적으로 관리하는 방안과 룰을 실행할 때 유형에 따라 여러 피연산자와 연산자 들을 호출 함에 따른 성능 향상 방안에 대한 추가 연구가 필요하다.

참고문헌

[1] IBM Business Consulting Services Korea, “금융기관의 on-demand 혁신전략”, 한국경제신문출판사, 2004.
[2] Barbara von Halle, “Business Rules Applied—Business Better Systems Using the Business Rules Approach”, Wiley & Sons, 2002.
[3] 김동수, “능동적 업무 규칙 관리를 통한 효과적인 워크프로우 관리”, 한국경영과학회/ 대한산업공학회 춘계공동학술대회 논문집, 2001. 4.27 ~ 28, 관동대학교.
[4] Ronald G.Ross, “Principles of the Business Rule Approach”, Addison-Wesley, 2003.
[5] Gartner Research, “The Business Rule Engine 2003 Magic Quadrant”, 2003.
[6] Barbara von Halle, “Business Rules Applied”, Wlley Computer Publishing, 2002.
[7] 이성하, “금융 프로덕트 팩토리를 위한 복합상품설계시스템의 개발 ” , 고려대학교 석사학위논문, 2003.
[8] 최성철, “맞춤형 온라인 금융상품 추천/설계시스템의 개발에 관한 연구” , 고려대학교 석사학위논문, 2003.
[9] Malcolm Chisholm, “How to Build a Business Rules Engine: Extending Application Functionality through Metadata Engineering”, Morgan Kaufmann, 2003.
[10] 서영만, “국내 은행의 비이자 수익 구조 개선 방향”, 금융 시스템 리뷰, 2005.
[11] Ernest Friedman-Hill, “Jess In Action Rule Based Systems In Java”, Manning, 2003.