제26회 한국정보처리학회 추계학술발표대회 논문집 제13권 제2호 (2006. 11)

분산 비즈니스 룰 시스템의 충돌 검출에 관한 연구

박재현*, 류성열**
*숭실대학교 컴퓨터학과
**숭실대학교 컴퓨터학과 교수
e-mail: *pjh@selab.ssu.ac.kr

A Study about Conflict Detection of Distributed Business Rules System

Jae-Hyun Park*, Sung-Yul Rhew**
*Dept of Computer Science, Soong-Sil University
**Dept of Computer Science Prof, Soong-Sil University

요 약

단일의 BRE를 사용하는 비즈니스 룰 시스템에 새로운 룰이 추가 또는 수정될 경우, 기존의 비즈니스 룰과 의미적인 충돌이 발생할 수 있으며, 이는 잘못된 비즈니스 수행결과를 초래하는 원인이 된다. 이를 위해 비즈니스 룰 시스템은 내부적으로 발생하는 비즈니스 룰 간의 충돌을 검출하고 해결할 수 있는 기능을 제공한다.

그러나 서로 다르게 구현된 다수의 BRE를 이용하는 분산 비즈니스 룰 시스템에서는 하나의 BRE에 추가된 새로운 비즈니스 룰이 다른 BRE의 룰과 충돌이 발생하는지 여부를 자동적으로 확인할 수 있는 방법이 제공되지 않아서, 사용자가 직접 모든 BRE에서 룰의 충돌 여부를 확인해야 하는 어려움을 가지고 있다. 기존의 분산 비즈니스 룰 시스템에 관한 연구에서도 서로 다른 BRE들을 분산 배치하고 비즈니스 룰을 실행하는 방법에 대한 연구만을 수행하였을 뿐, 분산된 비즈니스 룰 간의 충돌을 해결하는 것에 대한 연구는 미흡한 실정이다.

본 논문은 이러한 분산 비즈니스 룰 시스템에서 비즈니스 룰의 추가 및 변경 발생 시, 룰 간의 충돌을 검출하고 해결하기 위하여 기존 연구의 한계점을 분석하고 새로운 시스템 요구사항들을 도출하였다. 이를 기반으로 기존에 제시된 비즈니스 룰 브로커 구조를 확장, 보완한 분산 비즈니스 룰 충돌 검출 시스템의 아키텍처를 제안하고 이를 설계하였다.

1. 서론

BRE(Business Rules Engine)은 룰 기반 시스템의 한 종류로서 비즈니스 룰을 관리하고 실행하는 시스템을 말한다[1]. 핵심적인 비즈니스 로직에서 룰들을 분리하는 것은 비즈니스 프로세스와 응용 프로그램들과는 관계없이 비즈니스 룰의 재사용을 가능하게 하고, 응용프로그램과 프로세스 로직의 결합도를 낮추고, 비즈니스 룰의 빠른 유지보수를 할 수 있으며, 프로그램 유지보수 비용 감소시키는 등 많은 이점을 가지고 있다.

비즈니스 세계는 빠르게 변화하고 있으며, 기업들은 세계 각지에 널리 퍼져 활동하고 있다. 이를 위해 다수의 서로 다른 BRE를 사용하여 업무를 처리하는 분산 시스템에 관한 연구들이 수행되었다[2][3]. 그러나 분산 비즈니스 룰 시스템에 새로운 룰이 추가 또는 수정될 경우, 기존의 비즈니스 룰과의 충돌이 발생할 수 있으며, 이것은 잘못된 비즈니스 수행결과의 원인이 된다. 단일의 BRE에서 발생하는 비즈니스 룰 충돌을 검출하고 수정하는 기능은 각 BRE에 의해 제공되고 있으나, 서로 다른 종류의 BRE를 사용하는 분산 비즈니스 룰 시스템에서 비즈니스 룰 간의 충돌을 검출하고 해결하는 방안에 대한 연구는 미흡하다[3].

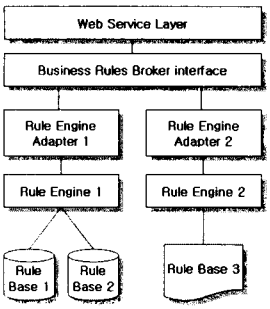
이 논문에서는 기 연구된 분산 비즈니스 룰 시스템의 한계점을 분석하여 새로운 시스템에 대한 요구사항을 도출하고, 기존의 비즈니스 룰 브로커 구조를 확장, 보완하여 룰 간의 충돌을 검출할 수 있는 아키텍처를 제안하고 이를 설계하였다.

* 본 연구는 숭실대학교 교내연구비 지원으로 이루어졌음.

2. 관련연구

2.1 비즈니스 룰 브로커(Business Rule Broker)

비즈니스 룰 영역에서 표준의 부재로 인해, 서로 다르게 구현된 엔진들이 생겨났고 다양한 룰 표현 형식들이 만들어 졌다. 더욱이 전사적 기업 환경에서 서로 다른 이종의 비즈니스 룰 기반의 응용프로그램들은 통합을 더욱 어렵게 하고 있다. 이에 대한 해결 방안으로 이종의 BRE들을 통합하기 위하여 비즈니스 룰 브로커 아키텍처를 이용한 서비스 지향 접근적인 설계와 구현이 제안되었다[2].



(그림 1) 비즈니스 룰 브로커 아키텍처

제안된 시스템의 아키텍처는 (그림 1)과 같다. 서로 다르게 구현된 Rule Engine들은 Rule Base를 가지고 있으며 이곳에 룰을 저장하고 관리한다. Rule Engine은 Rule Engine Adapter를 통하여 Business Rule Broker와 연결된다. Business Rule Broker는 각 룰에 대한 위치 정보를 가지고 있으며 웹 서비스를 이용하여 접근할 수 있다. 사용자가 특정 룰의 실행을 Business Rule Broker에게 요청하면 해당 룰을 가지고 있는 Rule Engine에게 실행을 요청하고 Rule Engine은 Rule Base를 이용하여 비즈니스 룰을 실행한다.

2.2 비즈니스 룰 충돌 해결

비즈니스 룰의 충돌이란 비즈니스 조건들이 동시에 하나 이상의 룰을 만족시킬 때 발생하며, 이 충돌의 해결 전략은 룰의 실행 순서를 조절함으로서 가능해 진다. 예를 들어, 자동차 렌트 시스템의 경우, "항상 VIP 회원은 렌트비의 10%를 할인해 준다."라는 룰과 "특별기간 동안 VIP 회원은 렌트비의 30%를 할인해 준다."라는 룰이 있다면 "항상"이라는 항목과 "특별기간" 사이에 충돌이 발생하게 된다. 시스템에 기간에 대한 우선순위를 따로 지정하여 주지 않는다면 시스템은 두 룰 사이에 충돌이 있다고 판단하게 된다.

이러한 충돌을 해결하기 위해서는 하나 또는 그 이상의 전략이 필요하다. 충돌이 발생하고 나면 충돌이 나는 룰들의 순서를 해결 전략에 따라 재조정하여야 하고, 이것은 자동화가 아닌 비즈니스 분석가에 의해서 결정되고 프로그래머에 의해 직접 수정된다[1][4].

2.3 기존 연구의 한계점

비즈니스 룰의 충돌은 업무수행에 있어서 의도하지 않은 결과를 만들어내며, 시스템의 중대한 오류를 발생시킨다. 특히 작은 결함에도 민감한 금융관련 시스템이나 의사결정 시스템에 있어서 치명적인 문제로 발전할 수 있다.

동일한 BRE내에서 발생하는 비즈니스 룰 충돌을 검출하는 기능은 이미 각 BRE에서 제공하고 있다. 그러나 분산되어 있는 서로 다른 BRE에서의 비즈니스 룰 충돌을 검출하는 기능은 제공되지 않고 있다. 기존의 비즈니스 룰 브로커와 분산 비즈니스 룰 시스템 연구에서도 룰의 배치와 실행에 관한 연구만 수행되었을 뿐, 시스템에 새로운 룰이 추가되거나 변경이 발생하였을 경우, 각 BRE 시스템의 비즈니스 룰 간의 충돌을 해결하는 연구는 미흡하였다[2][3]. 그러므로 분산 비즈니스 룰 시스템에서 하나의 BRE에서 비즈니스 룰의 추가 및 변경이 일어났을 경우, 다른 BRE에게 이를 알려주고 룰의 충돌 여부를 확인할 수 있는 방법에 대한 연구가 필요하다.

3. 새로운 시스템의 요구사항 분석

분산 비즈니스 룰 시스템의 충돌 검출을 위해서는 각 BRE와 룰에 대한 정보를 주고받으며, 충돌을 검출하기 위한 컴포넌트가 필요하고, 기존의 비즈니스 룰 브로커[2]를 확장한 새로운 아키텍처가 요구된다.

비즈니스 룰은 BRE가 인식하기 위해 Markup Language 형태의 명세 언어로 작성되어 진다. 그러나 비즈니스 룰 영역에서 표준의 부재로 인해 서로 다르게 구현된 엔진들이 생겨났고 다양한 룰 표현 형식들이 만들어 졌다[2]. 분산 비즈니스 룰 시스템에서 서로 다른 형태로 작성된 룰을 주고받기 위한 통일된 형식의 명세언어와 변환기가 필요하며, 이를 위해 현재 가장 주목받고 있는 The Rule Markup Initiative에서 발표한 비즈니스 룰을 위한 명세 언어인 RuleML(Rule Markup Language)을 사용한다[5].

BRE는 비즈니스 룰 처리를 위해 룰 변수(Rule

Variable)를 사용하며, 이들은 각 BRE마다 다르다. 하나의 BRE에서 생성된 비즈니스 룰을 다른 BRE가 이해하기 위해서는 동일한 의미로 사용된 변수들 간의 변환이 이루어져야 한다. 이를 위해, 각 엔진에서 사용하는 룰 변수들 간의 맵핑을 수행하는 컴포넌트와 맵핑 정보를 저장하는 저장소(Repository)가 필요하다.

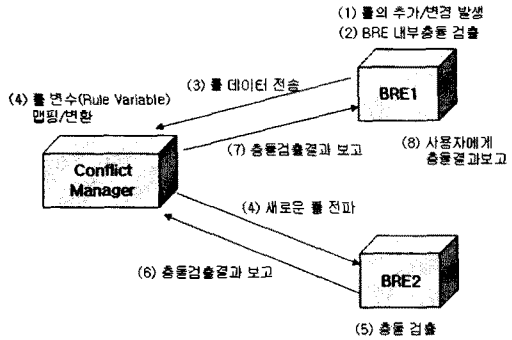
각 BRE간의 비즈니스 룰 교환을 위해서는 충돌과 관련된 정보를 주고받기 위한 메시지 구조가 정의되어야 한다. 새로운 룰이 추가되거나 변경됨을 알리는 메시지와 충돌 검출결과를 보고하는 메시지에 대한 설계가 이루어져야 한다.

4. 분산 비즈니스 룰 충돌 검출 시스템 설계

3장에서 제시된 기존 연구의 한계점과 새로운 시스템 요구사항 분석을 기반으로 분산 비즈니스 룰 충돌 검출 절차를 정의하였으며, 분산 비즈니스 룰 충돌 검출을 위한 새로운 아키텍처를 제안하고 이를 설계하였다.

4.1 분산 비즈니스 룰 충돌 검출 절차

분산 비즈니스 룰 시스템의 충돌을 검출하기 위한 절차를 (그림 2)와 같이 도식화하였다.

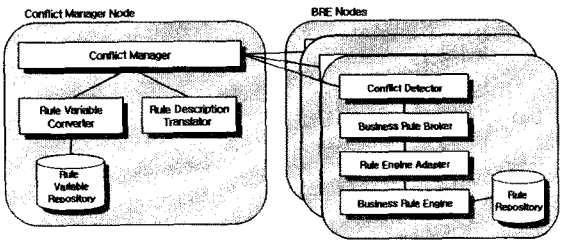


(그림 2) 분산 비즈니스 룰 충돌 검출 절차

룰의 추가 및 변경이 발생하면 해당 BRE1 자체 내에서 충돌 여부를 검사하고 충돌이 발생하지 않았을 경우, CM에게 변경된 룰의 정보를 전송한다. CM은 전달받은 룰을 다른 BRE2에게 전송하기 위한 룰 변수 맵핑을 수행 후 정보를 전달한다. BRE2는 전달받은 룰이 충돌이 발생하는지 확인하고 그 결과를 CM을 통해서 처음 룰이 생성된 BRE1으로 전송한다. BRE1은 CM으로부터 전달받은 룰에 대한 충돌결과를 사용자에게 보고한다.

4.2 시스템 아키텍처

충돌을 검출하기 위한 시스템 아키텍처는 (그림 3)와 같고, 각 컴포넌트들을 다음과 같은 역할을 수행한다.

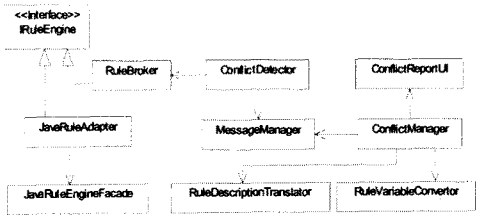


(그림 3) 분산 비즈니스 룰 충돌 검출 시스템 아키텍처

CM(Conflict Manager)은 CD(Conflict Detector)에서 보고된 새로운 룰을 RVC(Rule Variable Converter)를 이용하여 다른 BRE에서 이해할 수 있도록 룰 변수를 변환하고 각 CD로 전송하며, CD는 CM에게서 전달받은 새로운 룰에 대한 충돌을 검사하고 그 결과를 다시 CM에게 보고하는 역할을 한다. RDT(Rule Description Translator)는 표준으로 사용되는 명세언어(RuleML)과 BRE에서 사용하는 명세언어간의 상호 변환을 담당하고, RVC는 RVC Repository를 이용하여 각 BRE에서 사용하는 룰 변수를 맵핑하고 관리한다. BRB(Business Rule Broker)는 서로 다른 BRE를 동일한 방법으로 사용하기 위한 통합된 인터페이스를 제공하며, REA(Rule Engine Adapter)는 서로 다르게 구현되어 있는 BRE를 하나의 동일한 인터페이스로 연결하기 위한 아답터의 역할을 수행한다.

4.3 상세 설계

룰 충돌 검출을 위한 클래스와 그들 간의 관계를 나타낸 클래스 다이어그램을 (그림 4)와 같이 작성하였다.

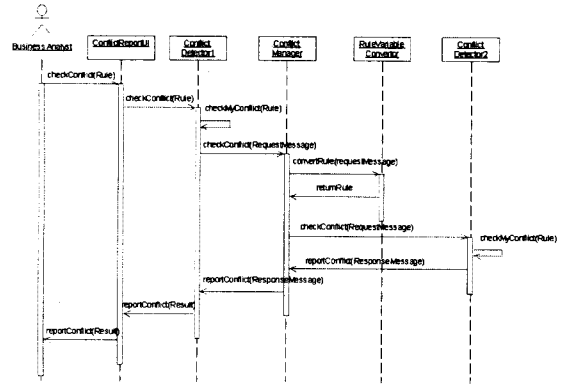


(그림 4) 비즈니스 룰 충돌 검출에 참여하는 클래스

CM과 CD 클래스는 각 BRE와 통신을 위해 메시

지 구조를 생성을 담당하는 MessageManager 클래스를 사용한다. CM은 룰 변수의 변환을 담당하는 RVC 클래스와 RuleML과 다른 룰 명세간의 상호변환을 위한 RDT 클래스를 이용하고, CD는 Rule Broker 클래스를 통해 BRE를 제어하고 룰의 충돌을 검출한다.

(그림 5)는 룰 충돌 발생시 이를 각 시스템에 알려주기 위한 시퀀스 다이어그램이다. 사용자가 UI를 통해 비즈니스 룰을 수정 또는 변경 후 CD1에 충돌 검출 요청을 하면 CD1는 자신의 BRE 충돌 여부를 검사하고 이상이 없음을 확인한다. 그 후, CM에 룰 변경알림 메시지를 전송하고, CM은 RVC를 이용하여 CD2를 위한 룰 변수를 변경하고, 각 CD2에 변경된 비즈니스 룰 정보를 전송한다. CD2는 자신이 담당하는 BRE의 충돌 여부를 검사 후, 그 결과를 다시 CM과 CD1을 통해서 사용자에게 보고한다.



(그림 5) 비즈니스 룰 충돌 검출 시퀀스 다이어그램

각 BRE의 룰 정보와 충돌검출 결과를 주고받기 위해서 2가지의 메시지에 대한 정의가 필요하며 이는 XML로 작성된다. 메시지의 세부항목은 <표 1>과 같다.

<표 1> 충돌 검출을 위한 메시지 구조

구 분	룰 변경알림 메시지	룰 충돌검출 메시지
항 목	- 룰 변경알림 ID - 수정일자 - 수정노드(BRE) - 룰 카테고리 - 룰 세부내용	- 충돌검출 결과 ID - 검사일자 - 룰 변경알림 ID - 룰 카테고리 - 검사결과

룰 변경알림 메시지는 분산 비즈니스 룰 충돌 검출 절차의 (3),(4)에서 사용되며, 한 BRE에서 룰이 추가되거나 변경되었을 경우에 이를 다른 노드에 알리기 위해서 사용한다. 반면, 룰 충돌검출 메시지는 충돌 검출 절차의 (6),(7)단계에서 사용되며, 새로운 룰의 충돌 여부 검사에 대한 결과 보고를 위해 사용된다.

5. 결론

비즈니스 룰의 충돌은 업무수행에 있어서 의도하지 않은 결과를 만들어내며, 시스템의 중대한 오류를 발생시킨다. 단일의 BRE를 사용하는 비즈니스 룰 시스템에 새로운 룰이 추가 또는 수정될 경우, 기존의 비즈니스 룰과 의미적인 충돌이 발생할 수 있으며, 이를 위해 비즈니스 룰 시스템은 내부적으로 발생하는 비즈니스 룰 간의 충돌을 검출하고 해결할 수 있는 기능을 제공한다. 그러나 다수의 BRE를 사용하는 분산 비즈니스 룰 시스템에서는 비즈니스 룰 간의 충돌을 검출하고 해결할 수 있는 구조가 제안되지 않아서 직접 사용자가 모든 BRE의 룰을 체크해야만 하는 번거로움이 발생한다.

본 논문에서는 이러한 기존 연구의 한계점을 분석하고 요구사항을 도출하여 서로 다른 BRE 간의 룰 충돌을 검출하고 이를 자동적으로 사용자에게 보고하여 해결할 수 있는 분산 비즈니스 룰 충돌 검출 시스템의 아키텍처를 제안하였으며, 이를 바탕으로 시스템의 설계를 수행하였다. 향후계획으로 본 논문에서 제안한 분산 비즈니스 룰 충돌 검출 시스템을 구현하고 그 유용성을 보일 것이다.

참고문헌

[1] The Business Rules Group. Defining Business Rules - What Are They Really? http://www.businessrulesgroup.org/first_paper/br01c0.htm, 2000. 7.

[2] F. Rosenberg and s. Dustdar, Design and Implementation of a Service-Oriented Business Rules Broker, In Proceeding of the 2005 Seventh IEEE International Conference on E-Comemerce Technology Workshops(CECW'05), 2005.

[3] F. Rosenberg and s. Dustdar, Towords a Distributed Service-Oriented Business Rules System, In Proceedings of the Third European Conference on Web Services(ECOWS'05), 2005.

[4] JBOSS, JBOSS Rules User Guide, http://labs.jboss.com/file-access/default/members/jbossrules/freezone/docs/3.0.4/html/index.html, 2006. 8.

[5] The Rule Markup Initiative, http://www.ruleml.org