SOA 관점의 Rule Engine 활용에 관한 연구

이재만
*포스코 ICT 기술연구소 SW 융합기술팀
e-mail : jaemanii.lee@poscoict.com

A Study on Usage of Rule Engine for SOA

Jae-Man Lee*
*SW Convergence Technology Team, POSCO ICT

요 약

최근의 SW 개발환경은 네트워크 환경이 발달하고 복잡한 비즈니스를 구현하기 위해 분산 컴퓨팅 환경으로 옮겨진 상황이다. 따라서, 시스템간의 쉬운 연결성을 보장하여 자원의 낭비를 줄이고 재사용하며 활성화 할 수 있는 플랫폼을 갖출 수 있는 SOA(Service Oriendted Architecture)의 활용이 필요한 시점이다. 이런 복잡한 니즈의 비즈니스를 구현함에 있어 업무 규칙을 별도로 관리하고 룰엔진에 의해 판단을 할 수 있다면, 프로그램 코드로부터 업무를 완전히 분리해낼 수 있다. 본 논문에서는 이러한 진일보한 사상을 조합하여 비즈니스 구현에 집중하고 결합성은 낮출 수 있는 SW 개발 아키텍처를 제시해본다.

1. 서론

SW 를 개발함에 있어 방법론이 추구하는 궁극적인 목적은 재사용성과 유연성을 높이는 데 있는데, 이를 위해서 의존성을 최대한 낮추려는 시도가 계속되고 있다. 특히 인터넷을 기반으로 한 네트워크 환경이 진화하고 시스템의 규모가 확장되면서 복잡도 역시 증가함에 따라 발생할 수 있는 위험을 감소하고 SW 제품의 품질을 향상시킬 수 있는 방법들이 고안되었다. 분산 컴퓨팅 환경에서 시스템간의 쉬운 연결을 통해 자원의 낭비를 줄이며 재사용할 수 있도록 하는 서비스 지향 아키텍처(SOA) 로의 진화는 최근의 이러한 요구사항을 반영한 트렌드라고 할 수 있다. SOA 에서 서비스로 구성된 시스템의 경우 각 서비스들은 다른 서비스에 독립적이고 쉬운 연결성을 보장한다.

시스템 개발의 목적이라고 할 수 있는 비즈니스 로직의 복잡도는 지속적으로 증가하고 있다. 웹환경에서는 이를 효과적으로 구현하기 위해 MVC 모델이나 프레임워크 등을 활용하여 각 구현 계층의 분할을 통해 비즈니스 로직 구현의 집중도를 높일 수 있었다. J2EE 의 JSR-94 스펙에서 제안하고 있는 룰 엔진은 이러한 복잡한 비즈니스 컴포넌트를 프로그램 코드에서 분할하여 기존 방식보다 집중도를 높이고 검증하는데 매우 용이한 아키텍처라고 할 수 있다.

본 논문에서는 SOA 관점에서 룰엔진을 활용할 수 있는 방법을 기술하고 사례를 제안하고자 한다.

2. REST 기반의 SOA 와 ESB

SOA(Service Oriented Architecture)는 기업의 IT 환경을 기존 기술 중심에서 실제 업무 중심으로 전환하여 기업이 역동적인 시장 요구에 민첩하게 대응할 수 있는 인프라를 구축한다는 개념으로 기업이 제공하는 서비스에 IT 를 효율적으로 활용한다는 목적에서 출발하고 있다. 그런데 이기종간 상호 운용성을 지원하는 웹서비스의 출현과 확산으로 인해 SOA 가 단지 논의가 아닌 실제 SW 개발환경에서 사용되고 있다. 보통 Open API 라고 하는 형태로 서비스에 대한 재사용이 이루어지고 있는데 웹서비스 구현 기술 중 하나인 RESTfull 웹서비스는 REST 기반의 웹서비스를 의미하고 HTTP 의 기본 기능만으로 원격 정보에 접근하는 서버 응용 기술이다. REST 에서 리소스(resource)란 REST 아키텍처의 핵심 요소로서 웹 사이트, 블로그, 이미지, 음악, 이용자, 지도, 검색 결과 등 웹에서 다른 이들과 공유하고자 개방된 모든 자원을 의미한다. REST 구조에서의 리소스는 그들의 고유한 URI 를 가지며, HTTP 의 기본 메소드인 GET / PUT / POST / DELETE 만으로 접근할 수 있는 단순한 구조를 지니고 있다. 따라서 개발을 위해 스펙을 숙지해야 하는 등의 문제가 없으므로 비교적 쉽게 프로그래밍을 할 수 있으므로, 공유 또는 제공하고자 하는 서비스를 다른 시스템에서 쉽게 이용이 가능하다.

ESB(Enterprise Service Bus) 는 SOA 플랫폼을 유연하고 확장성 있게 구현하기 위해 REST 등과 같은 기반 기술들을 연결할 수 있는 환경을 제공하고 있는 프레임워크이다. J2EE 에서 제공하는 기술 스펙들에 대해 비교적 자유롭게 서비스 믹스(Service Mix)가 가능하도록 관련 컴포넌트들을 내장하고 있다. 따라서, REST 아키텍처의 구현 플랫폼으로써 ESB 는 훌륭한 환경을 제공할 수 있다.

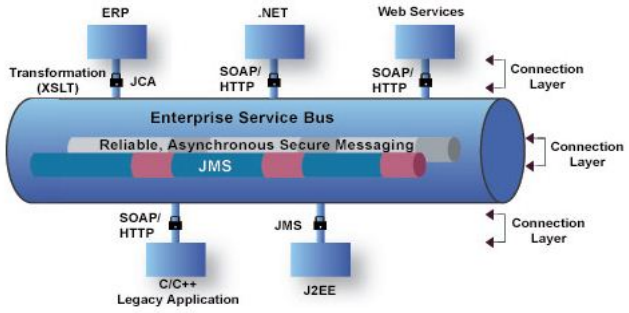

그림 1. Enterprise Service Bus 구성도

3. 룰엔진 아키텍쳐

시스템을 구축하는 작업은 어떤 비즈니스에 대한 요구가 발생하고 이를 구현할 방법 또는 아키텍쳐를 설계하여 프로그램 코드로 구현하는 프로그래밍 작업을 통해 완성하는 것이 일반적인 패턴이라고 할 수 있다. 예를 들어, 로그인 처리, 사용자 권한 인증, 신용 심사 처리 등 모든 것이 로직 즉 비즈니스와 관련된 것이다. 그런데 실제로 시스템을 구축하는 경우에 이런 로직은 매우 빈번하게 변경되는 것이 일반적인 상황이다. 예를 들어, 금융 업무를 생각해보면, 대출 상품은 매년 다양하게 출시되고 이를 심사하는 로직은 나이, 소득수준, 부동산평가, 신용등급에 따라 대출 가능 여부를 결정하게 된다. 또한 대출을 신청하는 고객의 등급에 따라 다른 기준 조건이 작용할 수 있으므로 로직 변경이 매우 빈번할 수 밖에 없는 상황이다. 이런 잦은 변경은 개발자들로 하여금 직접 소스코드를 수정하여야 하고 다양한 테스트까지 수행해야 새로운 로직이 적용될 수 있다. 심지어 아주 사소한 변화가 생기더라도 마찬가지인데, 해당 부분을 직접 수정하고 이로 인해 발생할 수 있는 모든 경우의 수에 대한 테스트를 마친후 배포하는 작업을 수행해야 한다. 로직과 프로그램 코드를 철저히 분리할 수 있다면 위와 같은 변경에 대해 기민하게 대응할 수 있고 컴포넌트의 품질 향상을 기대할 수 있다. 룰엔진(Rule Engine)은 비즈니스에 대한 규칙(Rule)을 별도의 저장소에 관리하고 변경에 대해 유연하게 대응할 수 있는 기술이다.

룰엔진(Rule Engine)은 정해진 규칙에 따라 입력 데이터에 대한 처리를 수행하는 플랫폼이며, 최소한의 지식(Knowledge)만으로도 결론을 추론(inferencing) 해낼 수 있는 도구이다. 이런 룰엔진을 기반으로 시스템을 구성한다면 복잡한 업무 규칙을 정의/운영/관리하는데 있어 매우 효과적일 수 있다. 그리고 이러한 업무 규칙의 정의는 가독성이 높은 특화된 별도의 언어를 사용함으로써 개발자외에는 시스템을 이해하기 힘들었던 기존의 시스템 개발 구조에서 비즈니스 전문가가 직접 시스템 개발에 참여할 수 있는 일종의 전문가 시스템(Expert System) 구조로 진화가 가능하다.

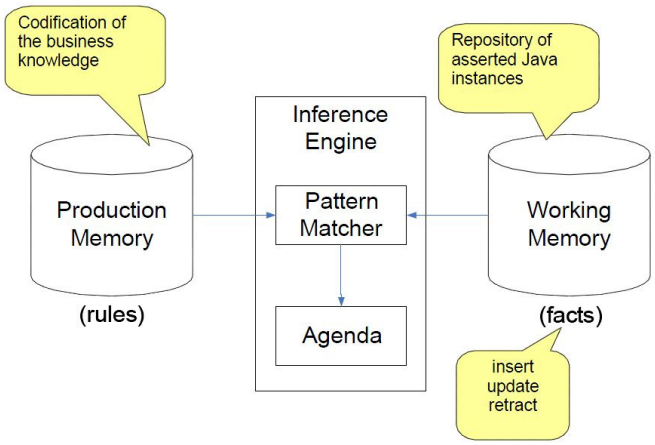

그림 2. Rule Engine 구조

룰엔진 기반의 시스템은 다음과 같은 장점을 지닌다.

1) 기민함(agility) : 업무 규칙을 별도로 관리하기 때문에 비즈니스 변경에 대해 빠르게 대응이 가능하다.
2) 표현성(representation) : 룰 기반 시스템은 업무 규칙을 전문적으로 정의하도록 설계된 룰 언어를 사용하기 때문에 프로그래머가 아닌 사람도 업무 규칙을 쉽게 적용할 수 있다.
3) 간결성(simplicity) : 업무 규칙을 룰엔진이 수행하도록 되어 있어서 복잡한 판단로직을 구현할 필요가 없다.
4) 성능(performance) : 로직을 판단할 수 있는 최적의 패턴 매치 알고리즘을 사용하고 있으므로 안정적인 성능을 제공한다.

4. 서비스 지향(SOA)의 룰엔진 수행 모델

SOA 에 기반한 룰엔진 수행 모델은 SOA 구현 프레임워크인 ESB 를 활용하여 RESTfull 방식의 웹서비스를 통해 룰수행 결과를 서비스로 제공한다.

본 수행 모델을 구성하기 위해 크게 Rule 관리/Rule 엔진/Service Prover 이렇게 3 가지의 형태로 구성된다. Service Provider(서비스 제공자)는 업무를 규칙에 따라 결정하기 위한 사실 데이터를 받아서 룰엔진에 전달하고 그 수행결과를 RESTfull 한 웹서비스로 응답하는 컴포넌트이다. 이런 컴포넌트는 프로토콜 변환이 자유롭고 서비스의 재사용성이 용이한 ESB 기반에서 구현한다. 룰엔진 API 는 서비스 제공자가 룰 엔진으로부터 결과를 얻기 위한 프로그램 인터페이스로써 Rule 과 ESB 를 연결해 주는 역할을 담당한다. 룰엔진은 사전 정의된 룰과 기준자료(사실, Facts)를 바탕으로 룰을 적용하고 평가하여 이후 수행될 작업을 결정하는 컴포넌트이다. 룰엔진은 기본적으로 룰언어로 기술된 업무 규칙과 애플리케이션에서 발생하는 정보인 팩트를 메모리에 올리고 팩트에 룰을 순차적으로 적용시키는 작업을 수행한다.

이와 같은 SOA 기반의 룰엔진 수행 모델을 바탕으로 금융 업무 처리 시스템을 구축의 예를 들어보자.
IT 서비스에 있어서 금융 업무 시스템은 대표적으로 복잡한 비즈니스 로직을 갖고 있는 대표적인 시스템이라고 할 수 있다. 특정 고객의 대출 가능 여부와 금액을 평가하기 위한 로직을 구현한다고 했을 때 HTTP 를 통한 XML 의 요청과 응답이 이루어지는데 ESB 와 REST 아키텍처가 이를 담당하게 되고, 대출심사 평가라는 비즈니스 로직의 구현은 룰엔진이 담당하게 된다.

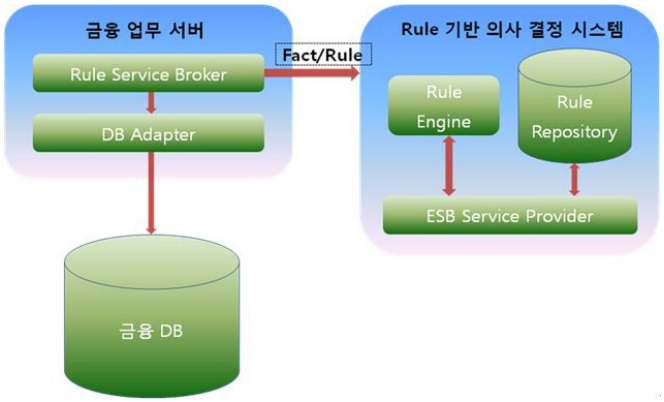

그림 3. 금융업무처리를 위한 시스템 구성도

위의 시스템 구성도에서 SOA 기반 룰엔진 시스템에 전달하기 위한 Fact 는 약속된 XML 형식을 사용한다.

- **URI** : GET http://{host}:{port}/load/verify
- **Header**
  Accept : application/xml
  RuleId : loan2012
- **Body**

```
<loanApp>
   <loanAmount>700000</loanAmount>
   <incomeRatio>0.9</incomeRatio>
   <user>
     <id>test</id>
     <age>39</age>
     <income>600000</income>
     <creditGrade>5</creditGrade>
   </user>
   …
</loanApp>
```

<대출심사요청 XML>

loanApp 엘리먼트 내부에 대출심사에 필요한 Fact 정보가 들어가게 되는데, 대출신청금액, 담보인정비율, 나이, 수입, 신용등급 등이 구조화된 XML 로 생성되어 시스템에 요청하게 된다.

그리고 다음과 같은 업무 규칙이 존재한다고 한다면 대출심사결과에 대한 응답을 반환한다.

rule1	규칙	나이검증
	조건	신청자의 나이가 20 세보다 작다
	결과	대출이 불가능하고, '대출신청자의 최소나이는 20 세 이상이어야 한다.' 라는 메시지 출력
rule2	규칙	신용등급
	조건	신용등급이 10 등급 이상이다.
	결과	대출이 불가능하고, '대출신청자의 신용등급은 10 등급 이상이어야 한다.' 라는 피드백 메시지 저장
rule3	규칙	최대대출가능금액계산
	조건	대출신청자의 연소득 인정비율만큼 최대대출금액을 계산한다.
	결과	최대대출가능 금액을 계산하여 저장한다.
rule4	규칙	대출가능금액점검
	조건	신청금액이 최대대출가능금액보다 크다
	결과	대출이 불가능하고 최대대출가능금액과 '최대대출가능금액은 XXX 원입니다' 라는 피드백 메시지 저장

<대출심사업무규칙>

```
STATUS : 200 OK
Content-Type : application/xml
<examineResult>
   <availableFlag>true</availableFlag>
   <amount>50000</amount>
   <maxAmount>50000</maxAmount>
   <feedbacks>
     <message></message>
     <message></message>
   </feedbacks>
</examineResult>
```

<대출심사응답결과 XML>

5. 결론

본 논문에서는 SOA 의 기술적 바탕인 ESB, RESTfull 웹서비스와 Rule 을 활용하여 금융 업무의 일부 비즈니스에 대한 시스템을 간략히 설계하였다.
SOA 에 기반한 시스템을 개발하는 목적은 재사용가능한 비즈니스 중심의 서비스를 생산하고 활성하려는데 그 목적이 있다. 또한 룰엔진(Rule Engine)을 결합함으로써 비즈니스를 프로그램 코드에서 분리하여 비즈니스 계층의 결합도를 낮추고 응집도를 높여 시스템의 품질을 높이는 효과적인 방법이 될 수 있다. 앞으로 위의 상호보완적인 기술의 조합을 진화하고 발전시킴으로써 SW 개발에 있어 비즈니스 구현의 수준을 한단계 올릴수 있는 아키텍처가 정립될 수 있기를 바란다.

참고문헌

[1] Robert B. Doorenbos 외 4 명, CMU-CS-95-113, “Production Matching for Large Learning Systems”
[2] 이용환, 정보과학회논문지[2006.12], “가변적인 컴포넌트 방식의 룰 엔진”
[3] 안윤애, 콘텐츠학회[2010], “유해가스 모니터링을 위한 JESS 기반 추론 규칙의 설계”
[4] 구중억 외 1 명,
한국도서관. 정보학회지(제 38 권 제 3 호), “SOA 기반 웹서비스의 Library 2.0 적용방안에 관한 연구”