2005 한국경영과학회/대한산업공학회 춘계공동학술대회
2005년 5월 13일~14일, 충북대학교

BPEL 실행을 위한 사건 기반 규칙 관리 시스템 구조 설계

Designing an Event based Rule Management System Architecture for Executing BPEL

한재준(연세대학교 정보산업공학과 석사과정)
김선일(연세대학교 정보산업공학과 석사과정)
김창욱(연세대학교 정보산업공학과 교수)

Abstract

Nowadays, the web service was based on composing the e-business and distributed computing environment. But existing workflow management systems have many problems with the interoperability between companies because they are made before most web services. For these problems, the workflow based composition language such as the Business Process Execute Language for Web Services (BPEL4WS or BPEL) appeared in expressing the web service composition. In this paper, we present an event based rule management system architecture that executes automatically the BPEL using the Event-Condition-Action (ECA) rules. The event based rule management system consists of the event manager, rule manager, and Jess. The Jess is the part of executing and deciding the condition.

1. 서론

지난 수 십 년 동안, 기업들은 비즈니스 프로세스에 대한 관심을 가져왔다. 이러한 관심은 프로세스가 조직 내에서 비즈니스 가치의 기본 단위라는 사실에서 비즈니스 운영, 조직 통합 및 비용 절약을 합리화하기 위한 필요에서 나왔다.

최근 기업들은 비즈니스 환경에서 많은 변화를 겪고 있다. 첫째는 급변하는 시장에서 다양한 고객의 요구 증가이다. 이를 만족시키기 위해 유연하고 다양한 기업 내부의 비즈니스 프로세스 관리가 필요하게 되었다. 둘째는 기업 외부의 변화로, 인터넷의 발전과 함께 나타난 e-비즈니스의 증가이다. 이는 다른 회사의 비즈니스 프로세스들간에 연계를 요구한다[17].

전자로 인해 많은 기업은 비즈니스 프로세스를 지원하고 기업 업무의 흐름을 통제하는 워크플로우 관리 시스템(Workflow Management Systems)을 도입하였다[4]. 하지만 e-비즈니스의 증가로 인해 기업들간 프로세스 연계가 빈번하게 필요하게 되었다. 기존 워크플로우 관리 시스템은 대부분 클라이언트/서버 구조로 구성되었기 때문에 상호운용성에 많은 문제점을 가지고 있다. 이러한 문제점을 해결하기 위해 등장한 것이 웹 서비스(Web Services)[19]이다.

웹 서비스는 프로세스간 느슨한 결합(loosely coupling)을 지원하며 표준적인 방법을 통하여 분산 컴퓨팅을 구현한다. 웹 서비스에 대한 관심의 증가와 수많은 기업들의 웹 서비스 수용은 이전에 CORBA나 DCOM이 가지고 있던 문제점을 해결해 주었다. 즉, 웹 서비스란 공통 프로토콜의 사용으로 인해 기업간 상호운용성을 높였다.

웹 서비스의 가장 큰 이점은 웹 서비스들이 서로 결합되어 새로운 비즈니스 프로세스가 나타날 수 있다는 점이다. 일반적으로 비즈니스 운영은 여러 가지 다양한 비즈니스 프로세스들의 사용을 필요로 하기 때문에, 이런 프로세스들 사이의 적절한 조정(Coordination)이 필요하다. 비즈니스 프로세스는 여러 참여자들과 관련된 작업의 흐름을 정의한다. 그러나 웹 서비스는 프로세스간 통합을 지원하지만 기업의 비즈니스 프로세스를 지원하는데 문제점을 가지고 있다. 이러한 웹 서비스와 비즈니스 프로세스간 문제점을 해결하기 위한 웹 기반 워크플로우 관리 시스템 연구가 진행되어 왔다[6][8][11][12][14][18].

비즈니스 프로세스 내에서 여러 웹 서비스를 통합하고 구성하기 위한 능력은 매우 중요하다. 더 중요한 것은 공통적으로 인정된 언어를 사용하는 표준화된 방법으로 비즈니스 프로세스를 표현하는 능력이다. 이러한 이유로 BPEL4WS(Business Process Execute Language for Web Services, also BPEL)[2], ebXML BPSS(Business Process Specification Schema)[7], WSCI(Web Services Choreography Interface)[20], BPML(Business Process Markup Language)[3] 같은 표준화된 워크플로우 기반 조합(composition) 언어가 등장하게 되었다. 이러한 언어들은 웹 서비스들 사이에서 논리적인 종속성을 결정하는 비즈니스 프로세스를 정의한다[5].

본 논문에서는 기업의 공급망(Supply Chain)상에서 사건-조건-실행(Event-Condition-Action) 규칙을 통해 BPEL을 자동으로 실행하는 사건 기반 규칙 관리 시스템 구조를 제안한다.

사건-조건-실행 규칙은 능동 데이터베이스 관리 시스템(Active Database Management System)[16]을 구성하기 위해 제안된 방법이다. 능동 데이터베이스 관리 시스템은 사건-조건-실행 규칙에서 사건, 조건, 실행과 이들의 집합을 명세하기 위한 수단으로서 규칙 정의 언어를 제공한다. 사건이 발생하면 이를 판단하고 이 결과에 따라 해당 동작을 수행한다. 하지만 데이터베이스 중심의 시스템은 고정적이며 유

연성이 부족하다[13][1]. 또한 데이터베이스 스키마 변경은 전체 정보시스템의 변경을 초래하여 시스템 전반에 걸친 정보의 일관성에 문제를 일으킨다.

본 논문은 비즈니스 프로세스를 수행하기 위한 사건-조건-실행 규칙을 일관적인 방법으로 생성, 갱신, 삭제하는 시스템 구조를 제시하는 것이 의의라 할 수 있다.

본 논문의 구성은 다음과 같다. 제2장에서는 시스템을 구성하는 요소에 대한 설명을 한다. 제3장에서는 사건 기반 규칙 관리 시스템의 클래스에 대해서 설명한다. 제4장에서는 본 연구의 결론을 맺고 추후 연구 과제에 대해서 설명한다.

2. BPEL 실행을 위한 사건 기반 규칙 관리 시스템 구조

<그림 1>은 사건-조건-실행 규칙을 적용한 사건 기반 규칙 관리 시스템 구조이다. 사건 기반 규칙 관리 시스템은 Jess 규칙 엔진(Rule Engine), 규칙 관리자(Rule Manager), 사건 관리자, 데이터베이스 연결자(DB Connector) 및 BPEL이 있다. 시스템에서 발생하는 사건은 사건 관리자를 통해 감지되고, 이는 Jess 규칙 엔진을 통해 판단되어 규칙 관리자를 통해서 실행된다.

이 구조의 특징은 사건, 조건과 실행 부분을 분리한 점이다. 데이터베이스 연결자와 사건 관리자의 연결을 분리하여 종속성을 줄였다. 이는 데이터베이스 연결자가 데이터베이스 장치를 관리하는데 유연성을 주며, 사건 관리자는 데이터베이스의 구조가 변경되어도 해당 부분을 담당하는 객체(Object)의 변경을 통해 문제를 해결한다.

Jess 규칙 엔진과 규칙 관리자는 조건을 판단하고 실행하는 부분으로써 프로그래밍 코드와 규칙을 분리하였다. 이를 통해 회사의 정책이 바뀌거나 실행 방법이 변경되었을 때 유연하게 대응할 수 있다. 또한 기존의 적은 부분을 수정하여 웹 서비스를 실행 할 수 있다.

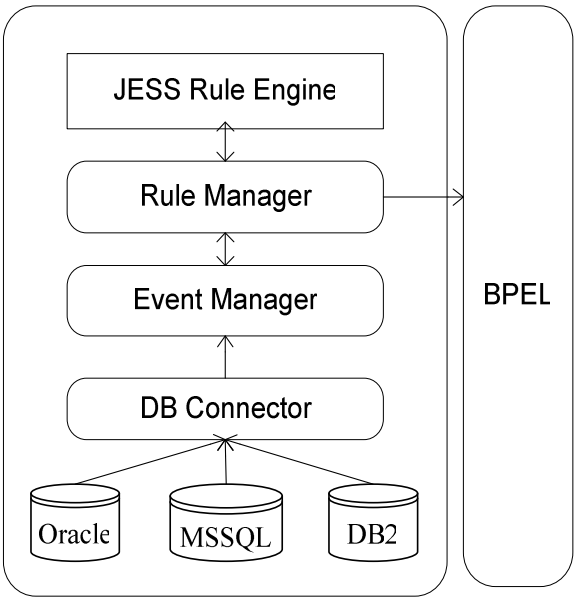

그림 1. Event Based Rule Management System Architecture

각 부분의 내용을 살펴보면 다음과 같다.

1) 데이터베이스 연결자

데이터베이스 연결자는 회사에 존재하는 다양한 데이터베이스 연결을 관리하는 부분으로 데이터베이스에 적합한 드라이버를 가지고 있다. 데이터베이스 연결자는 다양한 데이터베이스 장치를 관리하는데 유연성을 준다. 시스템을 구현할 때 장치에 상관없이 Façade 패턴을 이용하여 복잡한 데이터베이스 접속을 단순화 시켜준다

2) 사건 관리자

사건 관리자는 데이터베이스에 새롭게 입력, 삭제 및 갱신된 사건을 감지하는 부분으로써 Observer패턴을 이용하여 작업을 수행한다. 또한 사건 관리자는 새로운 사건을 정의할 수 있는 인터페이스(Interface)를 제공하며 <그림 2>의 사건 정의 규칙에 따라 정의된다.

<그림 3>은 규칙-조건-실행 규칙의 모습으로 사건 이름-Order_Product_LCD-을 <그림 2의>의 사건 정의 규칙에 따라 생성한 것이다. <그림 3>의 첫 번째 예는 고객이 주문을 요구했을 때 구매자(Buyer)의 재고 수량보다 주문 수

<EVENT>	→	<EVENT-NAME> <EVENT-ARGUMENT>
<EVENT-ARGUMENT>	→	<EVENT-ARGUMENT> | <ACTUAL_DB-FIELD-NAME> ',' <ACTUAL_DB-FIELD-VALUE> | ε
<EVENT-NAME>	→	<DB-TABLE-LIST> '_' < DB-FIELD_LIST>

```
<DB-TABLE-LIST>        →    <DB-TABLE-LIST> '_' <ACTUAL_DB-TABLE-NAME>
                              |<ACTUAL_DB-TABLE-NAME>

<DB-FIELD_LIST>        →    <DB_FIELD_LIST>   |<ACTUAL_DB-FIELD-NAME> '_'
                              <ACTUAL_DB-FIELD-VLAUE>| '_'   |ε
```

그림 2. Event Definition Rules (BNF Rule)

량이 적으면 주문을 실행하는 프로세스를 실행한다. 이 프로세서를 실행 한 후 재고 정보가 갱신되는 사건이 발생한다. 이 사건은 사건 관리자에 의해 감지된다. 두 번째는 이 때 발생된 사건에 예 이다. 만약 현재고가 안전재고 수량 보다 적게 되면, BPEL을 실행하여 주문 프로세스가 수행된다.

Example1 (Order)
Event: Order_Product_LCD
Condition: Order Quantity < Inventory Quantity
Action: Function Call(conducts the order)

Exampel2 (Safety Stock)
Event: SafetyStock_Product_LCD
Condition: On hand < Safety Stock
Action: Function Call(executes BPEL in order to order the product)

그림 3. Example of ECA Rules

3) 규칙 관리자

규칙 관리자의 역할은 크게 3가지로 구분된다. 사건 관리자로부터 감지된 사건을 Jess 규칙 엔진으로 보내는 역할, 규칙 자체를 정의하는 역할과 Jess 규칙 엔진에서 처리한 결과를 실행하는 부분으로 구성된다.

사건 관리자를 통해 관심 사건을 등록하면 규칙 관리자는 이 사건과 관련된 규칙을 생성할 수 있다.

4) Jess 규칙 엔진(Jess Rules Engine)

규칙 기반 언어는 절차 기반 언어(Procedural Language)와는 달리 선언적 언어(Declarative Language)이다[9]. 절차 기반 언어는 문제에 대한 자료 형을 미리 선언하고 개발자가 정한 순서에 따라 진행한다. 하지만 규칙 기반 언어는 문제에 대한 규칙만을 미리 선언하고 실행 방법은 실시간에 따라 규칙을 통해 정해진다. 절차 기반 언어는 규칙이 프로그래밍 언어의 코드에 포함되어 변경이 용이하지 않는 반면, 선언적 언어는 코드와 규칙이 서로 분리되어 있어 규칙의 변경이 용이하다[15]. JESS는 규칙 기반 언어방식을 따르며 Java언어를 지원한다. JESS 규칙 엔진에는 기업의 정책과 제한사항 등의 규칙을 가지고 있다. 이러한 규칙은 입력된 사실(fact)을 바탕으로 실시간에 규칙들이 연계되어 문제를 해결한다.

Jess 언어는 이전에 설명한 것처럼 선언적인 언어이므로 시스템을 운영하기 위해서는 규칙, 사실 및 함수(Function)가 미리 구현되어야 한다. Jess가 실행되기 위해서는 규칙에 적합한 사실이 시스템에 입력되어야 한다. 즉 사실의 입력을 통해 해당되는 규칙이 엮여서 하나의 결과를 도출해낸다.

```
(deftemplate order
        (slot production-name (type STRING))
        (slot order-quantity (type INTEGER))
        (slot due-date (type INTEGER))
)
```

그림 4. Jess template

<그림 4>는 Jess에서 사실을 입력하는 부분이다. 이 부분은 프로그램 언어에서 데이터를 저장하는 변수(variable)라고 생각하면 이해하기 쉽다. <그림 4>는 사건 관리자를 통해 등록한 하나의 사건에 대한 사실을 입력 받는 부분이다. 이 부분은 <그림 3>의 예 1의 사건에 대한 것으로 고객이 주문을 하면, 고객이 주문한 내용이 데이터베이스에 기록됨과 동시에 사건 관리자에 의해 사건을 규칙 관리자로 전송한다. 그러면 규칙 관리자는 Jess에 데이터를 넘겨주고 Jess는 이 데이터를 선언(assert)하여 사실화 한다.

<그림 5>는 <그림 3>에 있는 조건과 실행 부분을 Jess 언어로 나타낸 규칙이다. 이 규칙은 고객으로부터 온 주문을 처리하는 부분과 현재의 재고 수준이 안전 재고 이하로 내려갈 때 BPEL을 실행하여 공급업체에 해당 제품을 주문하는 규칙으로 구성되어 있다.

앞에서 언급했듯이, 위 언어에서는 규칙만을 기술하고 있어, 비즈니스 정책의 변경시 <그림 5>의 내용만을 변경함으로써 변경된 정책을 쉽게 적용 가능하다는 장점을 가지고 있다.

```
(defrule order-check
(order (order-quantity ?x))
(inventory (inventory-quantity ?y))
(test ( < ?x ?y)
```

```
=>
(assert (order-execute true))
(call Java or Jess function)
)

(defrle safety-stock-check
(inventory-check (inventory-quantity ?x)
(safety-stock (safety-stock-vlaue ?y)
(test ( < ?x ?y)
=>
(assert (under-safety-stock true))
(call Java or Jess function)
)
```

그림 5. Jess Language

5) BPEL

BPEL는 웹 서비스를 구성하여 비즈니스 프로세스를 정의하는 XML 기반 언어이다. 기업 내에서, BPEL은 기업 어플리케이션 통합과 비즈니스 파트너 사이에서 기존 시스템들을 통합하기 위해 사용된다. 또한 기업 간에서, BPEL은 비즈니스 파트너들끼리 더 쉽고 효율적인 비즈니스 프로세스의 통합을 가능하게 한다. <그림 6>은 BPEL을 구성하는 요소들의 개요도이다. 우리 시스템에서는 규칙의 최종 결과로 BPEL을 실행하게 된다.

비즈니스 프로세스가 BPEL에서 어떻게 기술되는지 설명하기 위해, 공급망상에서 구매자가 공급업자에게 주문을 하는 간단한 비즈니스 프로세스를 <그림 6>에서와 같이 나타냈다. 구매자가 주문을 하기 위해 비즈니스 프로세스를 호출하면, BPEL은 첫째로 현재의 재고 상태를 파악한다. 만약 재고가 있다면, 공급업자는 고객에게 주문 확인을 하기 위한 프로세스를 실행한다. 그리고 나서, 공급업자는 구매자에게 제품을 배송하기 위해 운송업체에게 배송 요청을 한다. 그러면, 운송업체는 공급업자에게 송장을 보내고, 공급업자는 다시 구매자에게 송장을 보낸다. 이 시나리오는 공급망상에서 데이터의 흐름처럼 비동기적인 BPEL 프로세스로 구현했다.

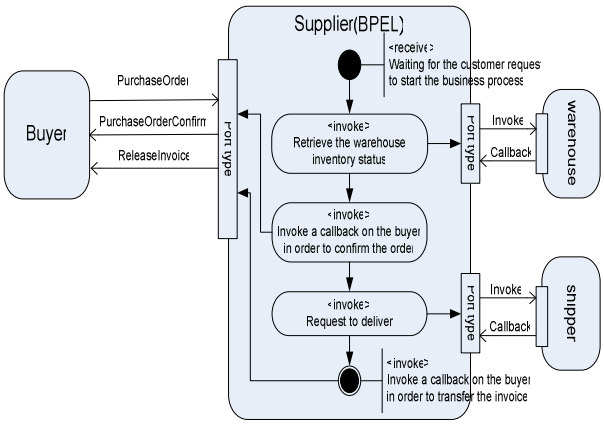

그림 6. Example of BPEL Process for Purchasing Order

3. 사건 기반 규칙 관리 시스템의 클래스 다이어그램

1) 데이터베이스 연결자

데이터베이스 연결자는 각각의 데이터베이스와 연결하는 데이터베이스 드라이버 클래스의 집합이다. Façade 패턴은 복잡한 서브시스템에 대한 단순한 인터페이스를 제공함으로 시스템의 확장성과 클래스 간 종속성을 줄여주며 클래스의 재사용성을 높여준다[10]. <그림 > 에서는 데이터베이스 연결자는 인터페이스 역할을 제공하며 이를 통해 실제 데이터베이스인 OracleDriver, MSSqlDriver, DB2Driver와의 연결을 제공한다. 각각의 드라이버는 데이터베이스에 입력, 삭제 및 갱신 사건을 처리하는 규칙이 구현되어있으며 인터페이스를 통해 일관된 방법으로 접속할 수 있다.

2) 사건 관리자

사건 관리자는 사건 처리자(Event Handler)와 사건 추출자(Event Extractor)로 구성되어 있다. 사건 처리자는 사건을 생성, 삭제 및 갱신하는 역할 즉 사건 자체를 관리한다. 사건 추출자는 Observer패턴을 적용하여 사건의 상태를 관리한다. 시스템에서 발생하는 사건을 감시하여 데이터베이스의 변경이 발생하면 그에 해당하는 사건을 규칙관리자에게 전달한다. Observer 패턴은 객체간 일대다의 관계를 가질 때 객체들 간 일관성을 유지하기 하고 재사용성을 높일 필요가 있을 때 자주 사용되는 패턴이다. 즉 어떤 객체의 변경을 다른 객체에 반영할 때 사용되며 게시-구독(Publish-Subscribe)관계라고 한다.

사건 처리자는 EvnetSpec, EventInterface,

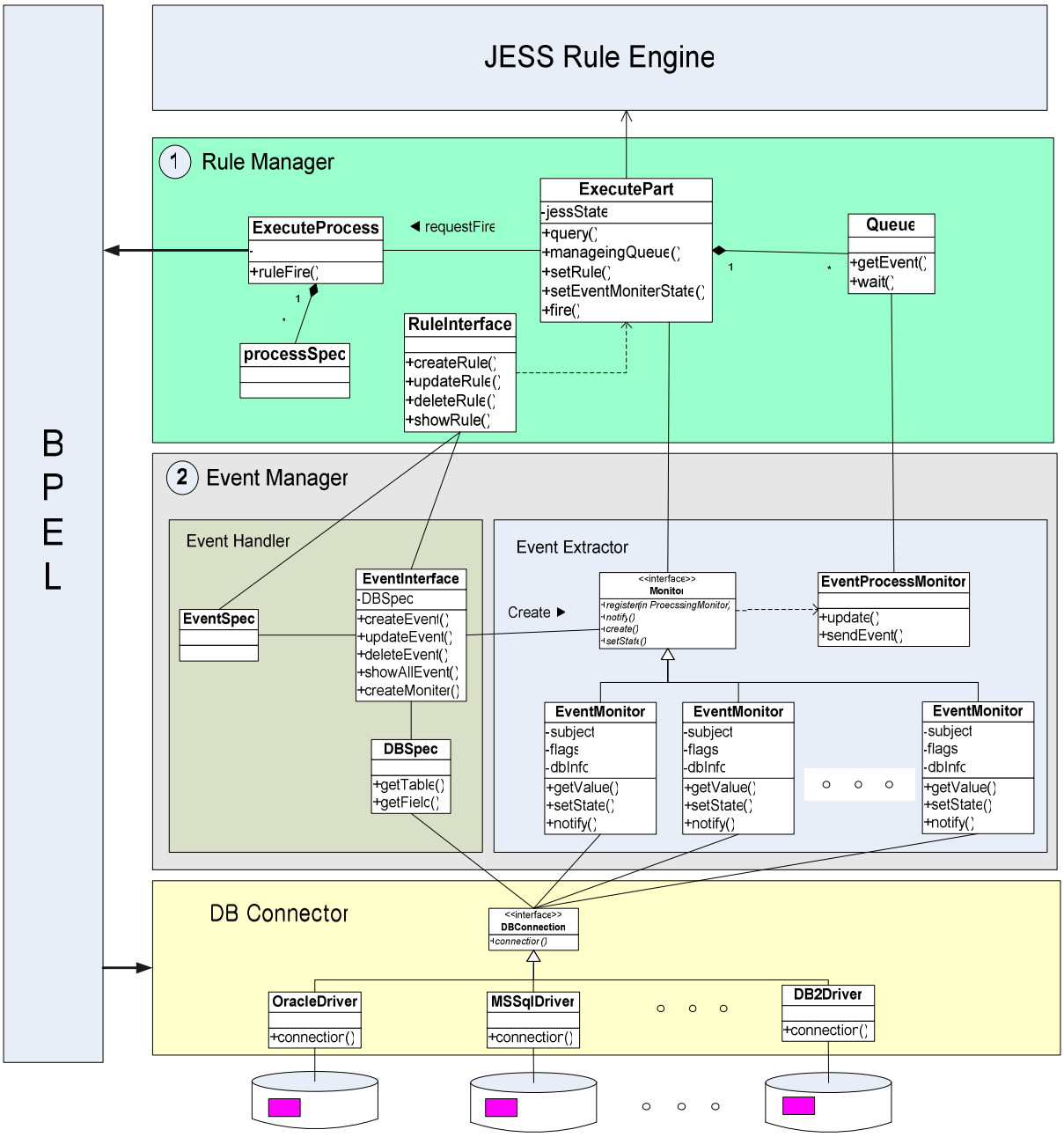

그림 7. Class Diagram of Event Based Rule Management System

DBSpec클래스를 가지며 사건 추출자는 다수의 EventMonitor, EventProcessMonitor, Monitor 클래스를 가진다. EventSpec클래스는 기존에 정의된 사건의 정보를 보관하고 있는 클래스이다. EvnetInterface클래스는 사용자에게 GUI(Graphic User Interface)환경을 제공하여 편리하게 사건을 생성할 수 있다. 또한 사건이 규칙과 연관이 있는 경우가 빈번하므로 RuleInterface와 연관되어 있다. DBSpec은 사용자가 사건을 생성할 때 <그림 2>와 같이 생성하므로 관련된 데이터베이스 정보를 관리하는 DBSpec클래스가 필요하다. EventMonitor는 데이터베이스 값을 감시하는 클래스이다. Monitor클래스를 상속받으며 해당 사건에 대한 다양한 플래그(Flag) 값을 가지고 있는데 그 중에 하나가 현재 해당 이벤트를 발생하여 Jess Rule엔진에서 처리 여부이다. 즉 해당 이벤트가 규칙엔진의 처리를 위해 이미 발생하여 Queue에 대기 중이라면 후에 입력된 같은 이벤트는 발생하지 않게 된다. Monitor클래스는 EventMonitor의 부모 클래스로 일관된 방법으로 EventMonitor에 접근하는 방법을 제공한다. Monitor의 역할은 EventInterface에서

사건 생성 요청이 들어오면 EventMonitor를 생성하는 역할과 ExecutePart에 해당 사건이 Jess 규칙 엔진에서 넘겨지면 플래그를 변경하는 역할을 한다. EventProcessMonitor는 Event가 입력되면 이를 처리하는 클래스로 EventMonitor의 플래그를 체크하여 해당 사건을 Queue에 전달 여부를 결정한다.

3) 규칙 관리자

규칙 관리자는 5개의 클래스로 구성된다. RuleInterface 클래스는 사용자가 규칙을 정의할 수 있는 GUI환경을 제공하며 EventInterface 클래스와 연관되어 규칙과 사건을 같이 정의할 수 있다.

Queue 클래스는 발생된 사건을 순서대로 보관하는 역할을 한다. ExecutePart 클래스는 Queue에 저장된 사건을 순서대로 Jess 규칙 엔진에 전달하는 역할을 한다. 또한 RuleInterface를 통해 정의된 규칙을 Jess 규칙 엔진에게 전달한다. 그리고 Jess 규칙 엔진에서 처리한 결과를 ExecuteProcess클래스에 전달하여 BPEL 또는 내부 함수의 호출을 요청한다. ExecuteProcess 클래스는 BPEL 또는 내부 함수의 목록을 가지고 있는 ProcessSpec클래스와 연관되어 ExecutePart에서 요청하는 작업을 수행하는 역할을 한다.

Jess 규칙 엔진은 사건에 대한 판단 결과를 엔진 내부에서 호출하지 않고 인덱스만을 넘겨준다. 그 이유는 구현과 판단을 분리하여 유연성을 부여하기 위해서이다. 즉 구현이 변경되어도 규칙은 변경되지 않음을 보장되기 때문에 시스템에 대한 유연성이 증가한다.

4) 시스템 이벤트 상태

시스템 상에서 사건의 상태는 <그림 8>과 같다. idle 상태는 시스템 상에 새로운 사건 발생 전의 대기 상태이다. 만약 고객이 LCD에 대한 주문을 요청하면 데이터베이스에 주문이 입력된다. 그리면 데이터베이스를 감시하는 EventMonitor를 통해 Order_product_LCD 사건이 발생하고 사건의 상태는 Occurred상태가 된다.

EventMonitor에서 발생된 사건은 Monitor의 update()함수를 호출함으로써 EventProcessMonitor 전달되고 플래그 검사를 한다. 만약 같은 사건이 Jess 규칙 엔진에서 처리 중이면, Reject 상태가 된다. 그렇지 않으면 Queue로 전달되어 Waiting 상태로 변경된다. Waiting 상태인 사건은 Jess 규칙 엔진으로 전달 된다. 만약 규칙 엔진이 처리 중이면, Processing 상태로 변경된다. 그런 후에, 규칙 엔진의 처리 결과로 BPEL이 실행되면 사건은 BPEL Processing 상태로 변경된다. BPEL Processing의 결과로 사건이 처리되거나 예외가 발생하면 사건은 Terminated상태로 변경된다. 시스템은 다시 사건이 발생 될 때까지 idle 상태로 있으며, 사건이 발생되면 위의 상태가 반복 순환된다

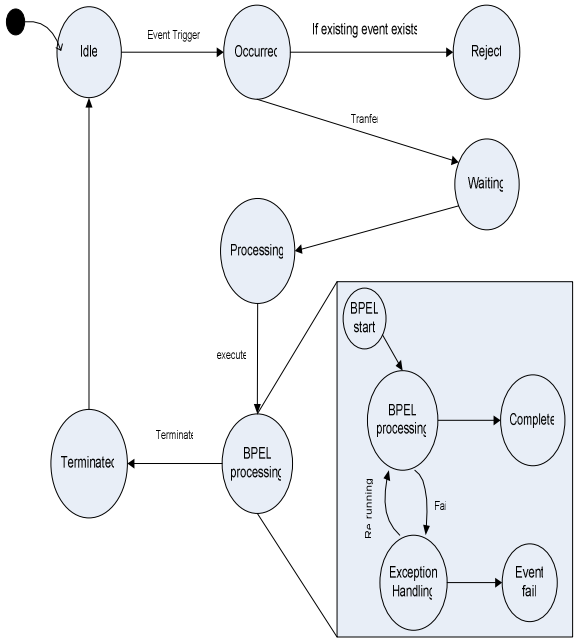

그림 8. System Event State Diagram

4. 결론

본 논문에서는 사건-조선-실행 이론을 바탕으로 사건 기반 규칙 관리 시스템 구조를 제안하고 이를 통해 비즈니스 프로세스를 구성하는 BPEL을 자동으로 실행 하였다. 본 논문의 의의는 사건-조건-실행 부분을 독립함으로써 재사용성과 유연성을 높일 수 있으며, 기존 워크플로우 시스템을 많이 수정하지 않고 웹 서비스를 정의 할 수 있다.

추후 연구 과제로는 첫째로, 본 시스템의 구조를 공급망 전체에 적용할 수 있도록 확장하는 것이다. 둘째로, 기업간 거래에서 협상(Negotiation)과 예외 사항을 처리할 수 있는 방안을 도출해 내는 것이다.

Reference

[1] Alonso, G., Gunthor, R., Agrawal, D., El Abbadi, A. and Mohan, C. (1996), Advanced Transaction Models in Workflow Contexts, *Proceedings of the Twelfth International Conference on Data Engineering*, 574-581.

[2] BPEL4WS Available at http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/
[3] BPML Available at http://www.bpmi.org
[4] Casati, F., Castano, S., Fugini, M., Mirbel, L., and Pernici, B. (2000), Using Patterns to Design Rules in Workflows, IEEE Transaction on Software Engineering, **26**(8), 760-785.
[5] Charfi, A. and Mezini, M., (2004), Hybrid Web service Composition: Business Processes Meet Business Rules, *Proceedings of the 2nd international conference on Service oriented computing*, 30-38.
[6] Chiu, Dickson K.W., Karlapalem, K., Li, Q., and Kafeza, E. (2002), Workflow View Based E-Contracts in a Cross-Organizational E-Services Environment, *an international journal of Distributed and parallel databases*, **12**(2/3), 193-216.
[7] ebXML BPSS Available at http://www.ebxml.org/specs/ebBPSS.pdf
[8] Espinosa, J. and Pulido, A. (2002), IB (Integrated Business): A Workflow-Based Integration Approach, *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, 2697-2702.
[9] Friedman-Hill, E. (2003), Jess in Action: Rule-Based Systems in Java, MANNING
[10] Gamma, E., Helm, R. Johnson, R. and Vlissides, J. (1995), *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley.
[11] Grnesarajah, D. and Lupu, E. (2002), Workflow-based composition of Web-services: a business model or a programming paradigm?, *Proceedings of the Sixth International Enterprise Distributed Object Computing conference*, 273-284.
[12] Jung, J. Y., Hur, W., Kang, S. H. and K, H. T. (2004), Business Process Choreography for B2B Collaboration, IEEE Internet Computing, **8**(1), 37-45.
[13] Mohan, C. (1994), A Survey and Critique of Advanced Transaction Models, *Proceedings of the 1994 ACM SIGMOD international conference on Management of data*, **23**(2), 521.
[14] Maurer, F., Dellen, B., Bendeck, F., Goldmann, S., Holz, H., Kotting, B. and Schaaf M., (2000), Merging Project Planning and Web-Enabled Dynamic Workflow Technologies*, IEEE Internet Computing*, **4**(3), 65-74.
[15] Russell, S. and Norving, P. (2003), Artificial Intelligence: A Modern Approach, Prentice Hall
[16] The ACT-NET consortium, The active database management system manifesto: A rulebase of ADBMS features, *ACM SIGMOD RECORD*, **25**(3) (1996) 40-49
[17] van der Aalst, W.M.P. (1999), Process-Oriented Architectures for Electronic Commerce and Interorganizational Workflow, *Information Systems*, **24**(8), 639-671.
[18] Kim, Y. H., Kang, S. H., Kim, D. S., Bae, J. S. and Ju, K. J. (2000), WW-FLOW: Web-Based Workflow Management with Runtime Encapsulation, *IEEE Internet Computing*, **4**(3), 55-64.
[19] Web Services Available at http://www.w3.org/2002/ws
[20] WSCI Available at http://www.w3.org/TR/wsci