# 분산환경에서 동적 메시지 교환을 위한 룰 기반 JMS 메시지 라우팅 시스템

## A Rule-based JMS Message Routing System for Dynamic Message Communication in based Distributed Systems

조풍연(Poongyoun Cho)*, 최재현(Jaehyun Choi)*, 박제원(Jaewon Park)*,
이남용(Namyong Lee)**

### 초 록

오늘날 컴퓨팅 환경은 과거에 비해 매우 분산되어 있으며 다양한 시스템과 연결되고 통합되는 동적인 구성을 필요로 한다. 대부분의 기업들은 이를 위해 객체지향적인 메시지 송수신 체계인 MOM(Message Oriented Middleware) 시스템을 사용하여 원격의 시스템과의 통신 채널을 구성하고 송수신 정보를 XML 메시지로 주고받고 있다. 그러나 대부분의 MOM 시스템들은 송수신 메시지로 사용되는 XML을 처리함에 있어 XML 메시지의 동적 변환 및 라우팅을 지원하지 않아 기업 시스템의 효율성과 유지보수성을 저하 시키고 있다. 이에 본 논문에서는 가장 널리 사용되고 있는 MOM 시스템인 JMS를 기반으로 동적 XML 메시지 변환 및 라우팅을 지원하기 위한 룰 기반의 라우팅 시스템인 RMRS (Rule-based Message Routing System)를 제안한다. RMRS는 룰 기반 이벤트 처리를 위해 사용되는 ECA(Event- Condition-Action)룰을 확장하여 분산 시스템 구현 시에 빈번하게 발생하는 라우팅 처리를 XML 변환과 함께 동적으로 처리할 수 있도록 한다. 이것은 내부적으로 메시지가 라우팅 되기 전에 수신한 메시지의 성격에 따라 XML 메시지의 변환을 수행할 수 있도록 설계되어 있어, 이를 바탕으로 MOM 시스템을 구축할 경우, XML 메시지의 동적 변환 및 라우팅을 효과적으로 구현할 수 있다.

### ABSTRACT

Today's computing environment which is getting distributed to communicate with various systems needs dynamic inter-connectivity of the systems. MOM(Message Oriented Middleware) is popularly used for transmitting XML messages among the distributed systems for the inter-connectivity. But, they do not support event-based message routing functionalities with XML transformation for processing effective message routing, which is essential to inter-connectivity, and there is no integrated platform to cope with these requirements. Although event-based message routing and XML transformation have been studied in a wide range of computer science areas, development of message routing systems is considered as a tough job due to the technological difficulties. In order to address these requirements, we proposed a novel system, named RMRS(Rule-based Message Routing System), which supports event-based message routing as well as XML message transformation. To make the proposed system easy to use, we also redesigned ECA(Event- Condition-Action) rule to fit in our system and developed a tool to map source XML structure into target XML structure.

**키워드** : 자바 메시지 서비스, 분산 메시지 처리, 룰 기반 메시지 라우팅
MS(Java Message Service), Distributed Message Processing, Rule-based Message Routing

---

본 연구는 숭실대학교 교내연구비 지원으로 이루어졌음.

* 숭실대학교 대학원 컴퓨터학과 박사과정

** 교신저자, 숭실대학교 컴퓨터학과 교수

2007년 12월 13일 접수, 2008년 05월 19일 심사완료 후 2008년 06월 09일 게재확정.

# 1. 서 론

기업의 활동범위가 넓어지고 다수의 파트너와 업무를 수행하는 오늘날의 기업 시스템들은 기존과는 달리 분산된 시스템 형태를 가진다. 이러한 분산시스템에서는 기업 내 또는 기업 간의 긴밀한 협업을 위해 분산된 기업 시스템들은 상호 실시간으로 정보를 주고받으며 동작하게 되므로 교환되는 정보의 신뢰성은 성공적인 기업 시스템 운영과 직접적으로 관련이 있다. 따라서 이러한 정보의 신뢰성과 같은 몇몇 운영조건을 보장하는 것은 성공적인 기업 시스템 운영을 위한 필수조건이며, 이에 대한 많은 연구가 이루어지고 있다. 특히 그러한 노력의 결과로 등장한 MOM(Message Oriented Middleware)은 실시간 정보의 여러 운영 건들을 만족시키기 위한 도구로써 활용되고 있다[5].

JMS(Java Message Service)는 대표적인 MOM 으로 자바 기반으로 메시지를 처리하며 기업 시스템 내부의 메시지 교환에 있어 널리 활용되고 있다. 이것은 보내고자 하는 데이터를 메시지 단위로 포장하고 송수신에 필요한 커넥션관리, 트랜잭션관리 등을 위한 표준화된 API를 제공하여 시스템의 개발 및 유지보수의 시간과 비용을 획기적으로 줄여준다. 그러나 JMS를 사용하여 시스템 간에 메시지를 송수신할 때 입력 메시지의 타입과 내용에 따라 원격 시스템으로 재송신하거나 변환 등의 작업을 주로 수행하게 되는데(이것을 메시지 라우팅(Message Routing)이라고 부른다) 대부분의 기업들이 이러한 과정을 별도의 처리 모듈을 개발함으로써 해결하고 있어 시스템의 재구축이나 유지보수에 있어 많은 어려움을 겪고 있다. 또한 이 과정에서 필요한 JMS 프로그래밍이나 XML 메시지 변환은 기술적으로도 어려운 작업이기 때문에 많은 비용과 시간을 요구하고 있다.

따라서 본 연구에서는 위와 같이 메시지 처리 시에 빈번하게 요구되는 메시지 라우팅과 XML 메시지 변환에 필요한 노력을 줄이기 위하여 JMS 환경에 적합한 JMS 라우팅 룰을 ECA 룰을 응용하여 개발하고 필요에 의해 수신 XML 메시지를 변환할 수 있는 XSLT(XSL Transformation)를 적용하는 방안에 대해 연구한다. 나아가서, RMRS(Rule-base Message Routing System)라는 시스템을 구현하여 연구의 결과를 실현하며 실제 환경에 적용한 시나리오를 통해 연구의 성과와 효율성을 나타내고자 한다.

본 논문의 구성은 살펴보면, 제 2장에서는 본 연구의 배경이 되는 JMS 플랫폼과 XML 변환 및 ECA 룰에 대한 관련연구를 살펴본다. 제 3장에서는 본 연구에서 제안하고자 하는 시스템을 소개하고 사용할 이벤트 기반의 메시지 라우팅 룰을 정의하며 구현 결과를 설명한다. 다음으로 제 4장에서는 구현한 RMRS를 사용하여 B2B(Business-to-Business)환경의 업무에 적용하여 제안한 방안과 구현 시스템이 실제적으로 의미가 있음을 보이고, 마지막으로 제 6장에서는 본 연구의 중요성 및 향후 과제와 함께 결론을 제시한다.

# 2. 관련 연구

## 2.1 JMS 및 메시지 라우팅

JMS는 오늘날 가장 많이 사용되는 메시

지 송수신 플랫폼으로서 실시간/비실시간 및 동기/비동기 통신을 모두 지원한다. JMS 시스템의 구성요소는 JMS 프로바이더(JMS Provider), JMS 클라이언트(JMS Client), 메시지(JMS Message)로 구성되며 <그림 1>과 같다.

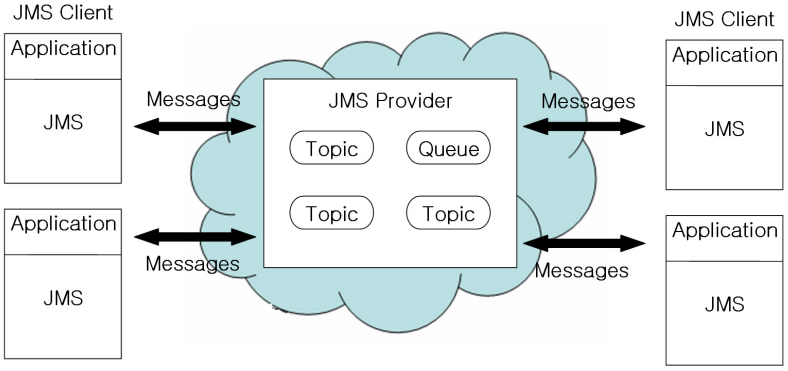
우선 JMS 프로바이더는 JMS 인터페이스를 구현한 메시지 서버이다. 메시징 자체를 제공하는 것 외에도 관리, 통제 기능을 제공한다. JMS 클라이언트는 Java언어로 작성된 JMS 프로바이더를 통해 메시지를 생성하고 소비하는 프로그램이나 컴포넌트를 의미한다.

메시지는 서로 다른 시스템 간에 주고받을 정보를 송수신을 위한 논리적인 단위로 표현한 구조체로서 시스템간의 약속에 따라 요청이나 보고 또는 이벤트 발생 등의 의미를 가질 수 있다.

JMS 프로바이더와 JMS 클라이언트는 JMS에서 제공하는 API를 통해 다양한 형태의 메시지를 표준화된 인터페이스를 이용하여 송수신 할 수 있다. 또한 분산 환경에서 원격의 메시지를 송수신할 수 있을 뿐 아니라 메시지 전송 보장기능이 지원되어 매우 신뢰할 수 있는 수준의 메시지 처리를 가능하게 한다.

송수신 시스템은 전송할 데이터를 메시지 형태의 객체에 담아 JMS 프로바이더를 통해 정해진 메시지 큐(Queue) 또는 메시지 토픽(Topic)에 데이터를 담게 된다. 메시지 큐는 JMS 프로바이더가 제공하는 메시지 중간 저장소 역할을 하며 송신 시스템에서 보낸 메시지를 일정시간동안 임시 저장하며 수신 시스템에서의 요청이 있을 경우 조건에 따라 메시지 큐에서 해당 메시지를 수신 시스템으로 전송하고 메시지 큐에서는 삭제한다. 메시지 토픽은 메시지 큐와 유사한 역할을 하지만 동일한 메시지를 다수의 구독자에 의해 수신될 경우 활용한다.

JMS에서 제공하는 메시지 전송 보장기능은 일련의 JMS 송수신 트랜잭션 중 발생한 오류를 감지하여 오류가 발생한 메시지를 rollback 시킴으로서 전송을 취소하고 해당 메시지를 자동으로 재전송하도록 하고 있다. 따라서 메시지 전송 시에 오류가 발생한 경우에 송수신 시스템은 이에 영향을 받지 않게 되며 메시지 생성 및 처리를 위한 비즈니스 로직에




〈그림 1〉 JMS (Java Message Service) 시스템 구성

만 집중할 수 있게 된다.

JMS 어플리케이션과 컴포넌트 사이에 일어나는 메시징 시스템은 점대점 연결 (Point-to-Point) 메시징 모델과 발행/구독(Publish-Subscribe) 메시징 모델의 두 가지가 존재한다. 메시징 모델인 점대점 연결 메시징 모델은 송신자가 보낸 메시지를 JMS 프로바이더의 메시지 큐를 통해 메시지를 교환 하는 모델을 의미하며, 발생/구독 메시징 모델은 송신자가 보낸 메시지를 JMS 프로바이더의 메시지 토픽을 통해 메시지를 다수의 구독자에게 발행하는 모델을 의미한다.

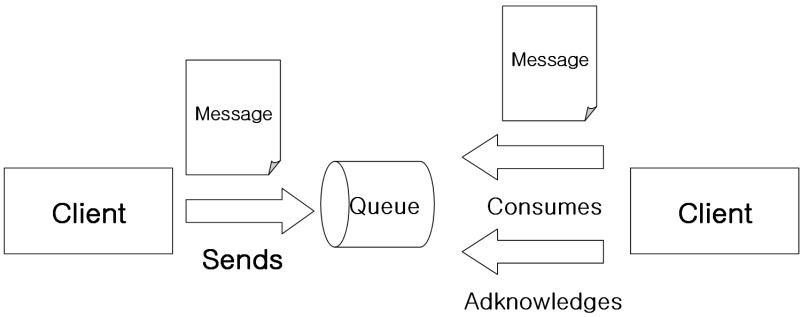
메시지는 수신자가 메시지를 읽어갈 때까지 혹은 유효시간이 경과되기 전까지 메시지 큐에 저장되어 있다. 다수의 수신자가 주어진 하나의 큐를 통하여 메시지를 수신 받을 수 있지만 메시지는 오직 한 수신자에게만 허락된다. 즉 하나의 메시지는 하나의 송신자와 수진자의 쌍 간에만 유효하다.

## 2.2 XML 변환

XML은 문서의 구조와 의미를 나타내는 정보들만 있고, 사용자에게 문서의 내용을 어떻게 표현할 것인가에 대한 정보를 가지고 갖고 있지 않다. HTML(Hyper Text Markup Language)의 경우 각각의 태그가 문서의 내용을 어떻게 출력할지를 결정하지만 XML의 경우 태그는 문서의 구조를 나타낼 뿐, 표현(presentation)과는 완전히 분리되어 있다. 따라서 XML 문서를 일반 사용자에게 제공하기 위한 별도의 언어가 필요한데, 이 언어가 바로 XSLT(XSL Transformations)이다[9, 11, 16].

XSLT는 질의 결과의 구조를 변환(structural transformation) 시키고, 포맷팅(formatting) 해서 새로운 구조의 XML 문서 혹은 다른 형식의 문서로 변환시켜서 사용자가 원하는 방식으로 표현가능하게 해 준다. <그림 3>은 XSLT를 사용하여 XML을 다양한 형태로 변환하는 기능을 도식화하고 있다. 핸드폰과 같은 기기에서 문서의 내용을 보기 위해서는 무선 인터넷에서 사용하는 WML (Wireless Markup Language) 스타일을 적용할 수 있고, HTML 스타일을 적용하면 웹 브라우저를 통해 읽을 수 있다.

XSLT는 XML 문법을 따라 작성된 문서를 다른 문법을 따르는 문서로 변환시키는




〈그림 2〉 JMS에서의 메시지 송수신 모델

기술을 기술한 언어이다. 이는 여러개의 템플릿(template)들로 구성되어 있고, 각각의 템플릿들은 패턴 정보를 가지고 있다. 여기서 패턴이란 XML 문서 내의 경로정보를 나타내며 XML 문서의 고유한 노드를 참조하기 위한 표준인 XPath로 표현된다[8].

## 2.3 ECA 룰

ECA 룰은 이벤트(event)가 발생하면 조건(condition)이 이를 분류하고 동작(action)을 실행하는 순으로 이루어진 동적인 규칙이다. 시스템은 각 규칙(rule)과 관련된 이벤트가 발생하게 되면 해당 규칙의 조건을 조사하고, 만약 조건을 만족한다면 규칙의 동작 부분을 실행한다[13].

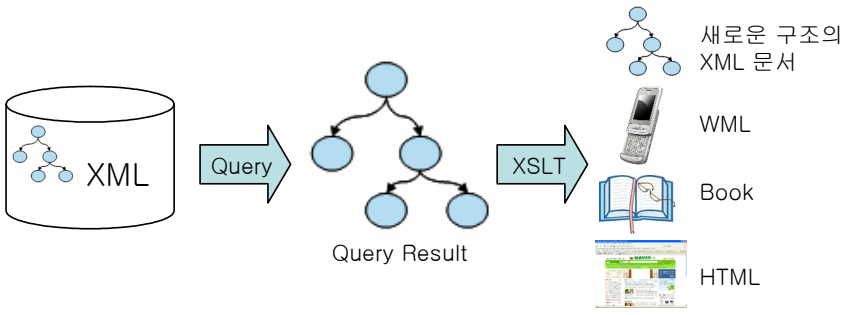
사용자가 이벤트와 조건에 해당하는 동작을 정의할 수 있으며, 사용자 또는 규칙 결정자는 프로그래밍과 같은 레벨의 지식에 대해선 알 필요가 없다.

이벤트가 발생하는 것은 규칙의 원인이 되고 조건은 규칙을 발생 시킬 때 필터 역할을 한다. 즉, 조건에 참일 때만 동적이 실행될 수 있다. 또한 ECA 룰은 동적으로 추가 및 변화가 가능하여 본 연구에서 추구하는 목적에 부합된다[15]. ECA룰은 이벤트와 주어진 조건에 의한 행동의 기술을 위해 룰 형태의 언어로서 아래와 같이 *on-if-do* 형태로 구성된다[14].

**on** *이벤트(event)*
**if** *조건(conditions)*
**do** *실행(actions)*

*on*은 ECA 룰이 호출 되어질 이벤트를 정의하는 부분이며 if는 호출 된 후 이벤트를 분석하여 실행을 할 것인지를 파악하는 부분이다. 마지막으로 do는 만일 if 부분의 조건이 모두 만족되었다면 실행되는 부분이다. ECA 룰은 근래에 들어 기업 정보시스템의 이벤트 프로세스 및 유비쿼터스 환경에서의 이벤트 처리 등의 여러 분야에서 현재 활발히 응용되고 있다[1, 2, 3].

ECA 룰안에서는 이벤트와 조건 그리고 행동을 명시하기 위하여 XML을 위한 다른 질의 언어를 사용하기 보단 XPath와 XQuery를 사용한다. XPath는 XML문서의 특정부분을 선택하고 조합하기위하여 XPointer나 XSLT, XQuery같은 W3C 권고에서 사용되고 ECA 룰의 요구사항에도 적합하다[6]. XQuery는 ECA룰에서 XML의 새로운 조각을 구성해야




〈그림 3〉 XSLT의 기능

할 필요가 있을 경우에만 사용된다.

### 2.4 JMS에서의 XML응용

JMS 메시지 송수신 시 실제 송수신 내용을 일반적인 텍스트 또는 바이너리 형태로 보내기 보다는 구조화된 정보 표현 형태인 XML을 사용함으로써 손쉽게 프로그램적으로 컨트롤하거나 변환할 수 있는 장점을 갖게된다. 이와 관련하여 XML 메시지를 동적으로 바인딩, 변환 또는 재전송을 위한 연구가 이루어져 왔다[16]. 이를 통해 JMS의 XML 메시지를 Java와 같은 언어로 바인딩하여 고급 프로그래밍을 쉽게 가능 하도록 지원하고 있다.

하지만 이 같은 연구는 JMS에 의해 송수신되는 XML 메시지의 컨트롤에 대한 연구이며 실제 JMS 메시지 자체의 라우팅이나 XSLT를 이용한 변환에 대한 통합된 플랫폼을 만족하지는 못하고 있는 실정이다.

나아가 JMS 응용기술은 본 연구에서 추구하고 있는 이벤트 기반의 메시지 라우팅을 실현하기 위한 이벤트, 라우팅 및 XML 변환에 대한 동적측면을 고려하고 있지 않고 있기 때문에 실제 프로젝트에서 필요한 메시지 송수신과정에서의 다양성 충족하고, 메시지교환을 효율적으로 지원하기에는 다소 어려운 점이 있다.

## 3. RMRS 구성 및 이벤트 룰

본 연구에서는 JMS 환경에서 응용될 수 있는 이벤트 룰과 XML 변환 룰을 개발하고 이를 구현한 RMRS (Rule-based Message Routing System) 시스템을 구현하였다. RMRS는 JMS 환경에서의 룰 기반의 메시지 라우팅 및 XML 메시지 변환을 자동화하고 통합하며, 수신 메시지의 성격에 따라 XML 변환을 통해 타 JMS 시스템 등으로 재전송하는 기능을 제공한다. 이는 JMS 응용 프로그램 개발 시에 간단한 라우팅 룰 및 변환 룰 작성만으로도 이를 가능하게 하여 시스템 구축을 효율적으로 지원할 수 있게 된다.

### 3.1 RMRS 소개

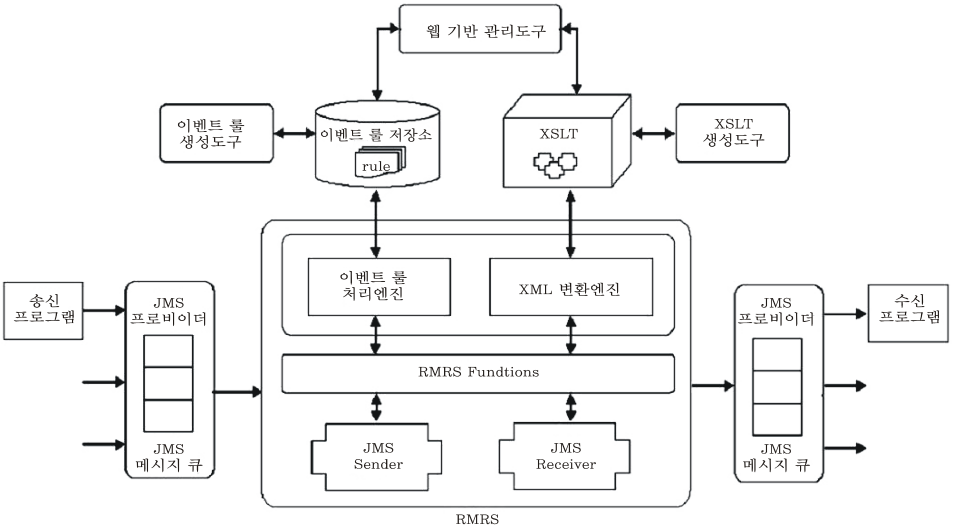
RMRS는 JMS에서 제공되는 JMS 프로바이더를 응용하고 JMS Receiver 및 JMS Sender를 활용하여 이벤트 룰을 처리하고 XML 변환을 수행하게 된다.

이벤트 룰은 ECA 룰을 JMS 메시지 처리에 적합하도록 응용한 규칙이며 XML 변환을 위해서는 XML 변환 표준규약인 XSLT를 활용한다. RMRS의 아키텍처는 <그림 4>와 같다.

시스템은 크게 서버와 클라이언트 도구로 구분되어진다. 또한 서버에는 크게 이벤트 룰 처리엔진, XML 변환엔진, RMRS Functions의 세 가지의 구성요소로 이루어지는데 각각의 역할과 기능은 다음과 같다.

- **서버 측 컴포넌트**

1) 이벤트 룰 처리엔진 : JMS 프로바이더를 통해 수신한 메시지가 적용되어질 이벤트 룰을 이벤트 룰 저장소에서 검색한 후 해당 이벤트 룰을 JMS 메시지에 적용한다.




〈그림 4〉 RMRS 아키텍처

2) XML 변환엔진 : 수신 JMS 메시지의 XML 문서를 재 송신할 XML 문서로 변환한다.
3) RMRS Functions : RMRS에서 사용될 행동을 정의하는 인터페이스로서 이벤트 룰 처리엔진, XML 변환엔진 등의 모든 기능이 이에 포함된다. RMRS Functions 자체는 이벤트 룰 처리엔진이 동적으로 호출하여 사용할 수 있는 함수의 개념적 명칭이며, 실제 역할을 수행하는 물리적 함수들은 사전에 정의된 함수를 이용하거나 사용자의 필요에 의해 새로운 함수를 작성하여 적용할 수 있다. 본 연구의 구현에서는 가장 자주 사용되는 XML 메시지 변환 함수인 *Xml Transformation* 함수를 사전에 구현하였다.

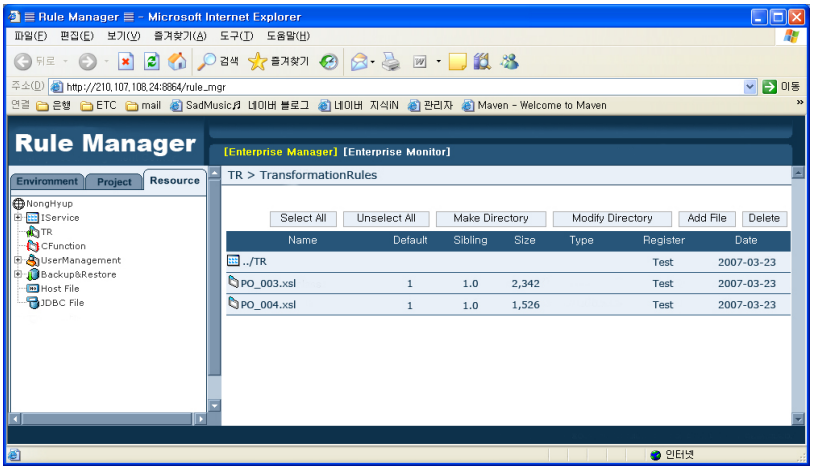
클라이언트는 다음의 웹 기반 관리도구, 이벤트 룰 생성 도구, XSLT(XML 변환 룰) 편집도구의 세 가지 도구로 이루어진다.

- **클라이언트 측 도구**

1) 웹 기반 관리도구 : 이벤트 룰과 XML 변환 룰을 등록 및 관리한다. 또한 로그 및 설정 등을 관리한다.
2) 이벤트 룰 생성도구 : 이벤트 룰을 생성하고 변경한다.
3) XSLT(XML 변환 룰) 편집도구 : XML 변환 룰을 생성하고 변경하고 새로운 변환 함수를 작성하고 관리할 수 있다.

## 3.2 이벤트 룰 구성

이벤트 룰은 수신된 JMS 메시지에 조건이 만족되었을 때 정해진 기능을 자동적으로 수행하기 위해 사용된다. 또한 향후 업무의 변경으로 인해 처리 규칙이 변경되었을 때




〈그림 5〉 웹 기반 관리도구

시스템의 수정 없이 해당 이벤트 룰만 변경하거나 생성함으로써 변경된 업무가 적용 될 수 있도록 한다.

본 연구에서 개발한 이벤트 룰은 ECA 룰을 JMS 환경에 적합하도록 응용한 JMS 이벤트의 처리 규칙이다. JMS 메시지는 보통 XML 문서 형태로 구성되며 반복되는 구조를 포함하게 된다. 각 반복되는 구조는 상황에 따라 개별적인 변환이나 재송신이 필요하게 되는데 ECA 룰은 이러한 XML 문서 형태의 반복적인 처리작업에 부적합하다. 또한 일반적인 ECA 룰은 XML 구조를 인식하지 못하는 문제가 있기 때문에 JMS 메시지에 포함되는 XML 문서의 특정 값을 참조하여 조건을 설정하기 위해서는 XPath 기반의 참조가 필요하다.

이를 위해 XPath 기반의 ECA 룰인 XML ECA 룰을 사용하였다. 그리고 JMS 메시지에 적합하도록 확장하였다.

이벤트 룰의 형태는 기존의 ECA 룰과 유사하지만 XML 처리에 적합하도록 *for-each, if else* 등의 항목을 추가하였다. 또한 JMS 메시지 헤더(JMS Message Header)에 JMS_EVENT_TYPE 속성을 정의하고 해당 값이 이벤트 룰의 on 부분과 동기화 될 수 있도록 구성함으로서 이벤트 룰 구현 시 약간의 명명 규칙(Naming Convention)을 활용하여 손쉽게 JMS 메시지와 연관 시킬 수 있도록 하였다. 이벤트 룰의 형태는 다음과 같다.

**on** *JMS_EVENT_TYPE*
**for-each** *Loop condition*
**if** *XPath-based conditions*
**do** *RMRS Functions*
***if else*** *XPath-based conditions*
**do** *RMRS Functions*

다음 <표 1>은 이벤트 룰의 구성항목에 대한 세부 설명과 필부여부이다.

수신된 JMS 메시지는 약속된 JMS_EVENT_TYPE 속성을 JMS 메시지 헤더에 포함하게 된다. 수신된 JMS 메시지와 XML 문서 내

〈표 1〉 이벤트 룰의 구성항목

| 항 목 | 설 명 | 필수여부 |
|---|---|---|
| on | 수신 JMS 메시지에 따라 적용할 룰을 지정 | 필수 |
| for-each | XML 문서의 반복내용 처리를 위한 조건 설정 | 선택 |
| if | 이벤트 실행 여부를 결정하기 위한 조건 설정 | 필수 |
| if else | 상위의 if문 또는 if else문을 만족하지 못하였을 경우 이벤트 실행 여부를 결정하기 위한 조건 설정 | 선택 |
| do | 조건이 만족 되었을 때 적용할 함수 설정 | 선택 |

용이 아래 <표 2>와 같을 때 <표 3>과 같은 이벤트 룰이 적용 될 수 있겠다.

〈표 2〉 JMS 메시지의 예시

```
JMS_EVENT_TYPE
PurchaseOrder

XML Message Body
<PurchaseOrder id = "073098_KI_ASD3982">
  <Sender type = "partner" source_id = "SED : 39d : S56">
    <CompanyName>Company_A</CompanyName>
    <Address>... Seoul, SouthKorea ... </Address>
    ...
  </Sender>
  <OrderItems t_count = "5">
    <OrderItem oi_id = "1">
      <ProductId>AOP19923</ProductId>
      <ProductType>CF</ProductType>
      <OrderCount>15</OrderCount>
      <ArrivalDate>2007-02-17</ArrivalDate>
      ...
    </OrderItem>
    ...
  </OrderItems>
</PurchaseOrder>
```

<표 2>는 PurchaseOrder 이벤트 타입을 가진 JMS 메시지로써 회사이름과 주소 등의 발신자 정보와 주문할 제품의 수량, ID, 타입, 예상 도착일 등의 내용을 가지고 있다. 한편 <표 3>의 이벤트 룰은 PurchaseOrder 이벤트에 대해서 "/PurchaseOrder/OrderItems/OrderItem" 엘리먼트 각각에 대하여 ProductId가 AOP19923이고 ProductType은 CF인 조건이 맞을 경우 변환과 메시지 전송(라우팅)을 수행한다. 따라서 <표 2>와 같은 JMS 메시지 수신 시 <표 3>의 이벤트 룰이 적용되어 'P04_P0'라는 XML 변환 룰에 따라 JMS 메시지 본문에 변환되고 변환된 메시지는 destination_A로 재전송 되게 된다.

〈표 3〉 이벤트 룰 예시

```
on
PurchaseOrder
for-each
($xml_body/PurchaseOrder/OrderItems/OrderItem)
if
ProductId[. = 'AOP19923'] and ProductType[. = 'CF']
do
XmlTransformation('P04_PO'),
SendMessage('system_A', 'destination_A')
```

### 3.3 룰 기반 메시지 변환

이벤트 룰의 *if* 항목에서 정의한 조건에 맞는 메시지가 수신되었을 경우 *do* 이하에 정

의된 함수를 실행하게 된다. 이때, JMS 메시지에 포함된 XML 메시지의 변환이 필요할 경우 *XmlTransformation* 함수를 호출하여 변환을 처리하게 된다.

RMRS에서는 이러한 XML 변환 룰 작성을 보다 쉽게 하고 수작업 시 발생하는 오류를 최소화하고 XML 변환 룰이 수정될 때 사용자로 하여금 빠르고 손쉬운 작업을 위하여 다음 <그림 6>과 같은 XSLT 편집도구를 사용한다.

XSLT 편집도구는 사전에 정의된 원본 XML의 구조와 목적 XML의 구조를 변환 각각 왼쪽과 오른쪽 화면에 트리형태로 로드하여 원본 XML 구조의 XML 원소를 목적 XML 구조의 XML 노드에 사상(mapping)하는 방식으로 XML 변환 룰(XSLT)을 정의한다. 이때 단순 사상뿐만 아니라 XSLT 표준에 정의되어 있는 사전 정의 함수와 사용자가 직접 구현한 사용자 정의 함수를 사용하여 복잡한 구조를 가지는 변환 룰도 작성이 가능하다.

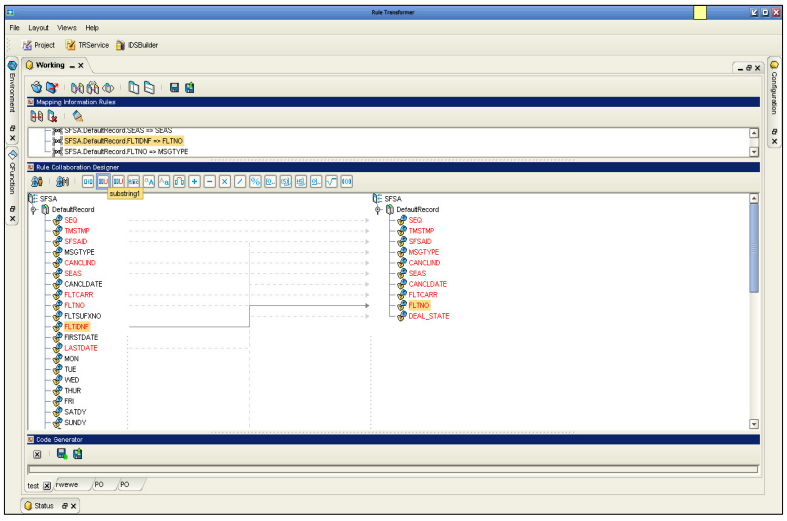
위와 같은 XSLT 편집도구를 활용함으로써 기존의 XML 메시지 변환을 위한 다수의 반복적인 프로그램 작업을 없애며, 향후의 시스템 변경 시에도 프로그램의 수정 없이 도구를 통해 만들어 져 있던 사상의 간단한 수정으로 빠르고 효율적으로 대처할 수 있다.

### 3.4 RMRS 처리 알고리즘

RMRS의 처리 알고리즘은 메시지 수신, 메시지 분석, 이벤트 룰 로딩 및 이벤트 룰 실행의 4가지 단계으로 구성된다. <그림 7>는 RMRS의 알고리즘 순서도 이다.

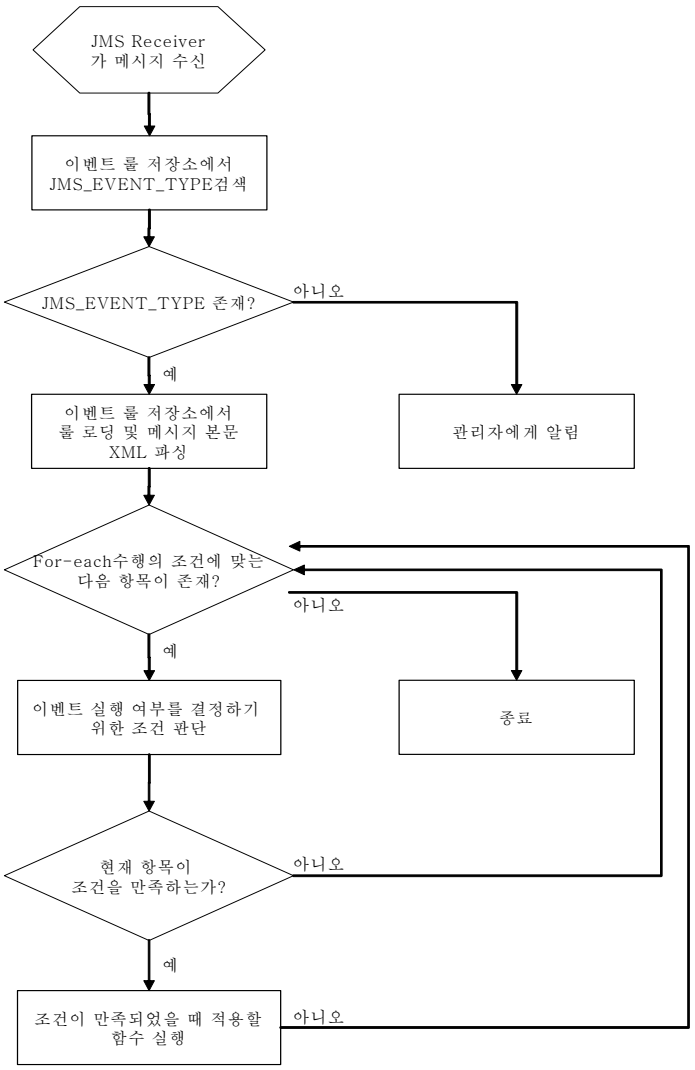
처음 메시지 수신 단계에서는 JMS 프로바이더의 메시지 큐로 송신측 프로그램에서 송신한 원본 메시지가 들어오게 된다. 이때 전송되는 메시지의 헤더에는 메시지를 처리할 이벤트의 타입과 송신자의 정보, 메시지 유효 시간 같은 정보가 명시되어 있다.

둘째로 메시지 분석 단계에서는 JMS Re-




〈그림 6〉 XML 변환 룰 생성 도구

ceiver가 메시지 큐로부터 메시지를 수신한 후 메시지 헤더를 분석하여 수신된 메시지를 처리할 이벤트 타입을 이벤트 룰 저장소에서 검색한다. 검색된 이벤트 룰은 이벤트 룰 저장소로부터 이벤트 룰 처리엔진으로 로딩 되는 이벤트 룰 로딩단계를 거치게 된다. 이때 수신 메시지에 명시된 이벤트 룰이 이벤트 룰 저장소에 없을 경우 메시지는 롤백 되고 UndefinedEvent라는 오류 사항이 관리자에게 통보되어 관리자가 해당 이벤트에 적절한 이벤트 룰을 정의할 수 있도록 한다. 이벤트 룰이 성공적으로 로딩 되면 다음의 룰 실행 과정을 위하여 메시지 본문 XML을 파싱(parsing)한다.




〈그림 7〉 알고리즘 순서도

마지막으로 이벤트 룰 실행단계에서는 로딩 된 이벤트 룰의 조건문을 통해 파싱된 XML 본문을 검사하고 조건에 맞을 경우 해당 조건문에 명시된 명령을 실행 한다. 이 명령들은 RMRS Functions의 조합으로 구성되고 이를 통하여 변환 및 라우팅을 할 수 있다.

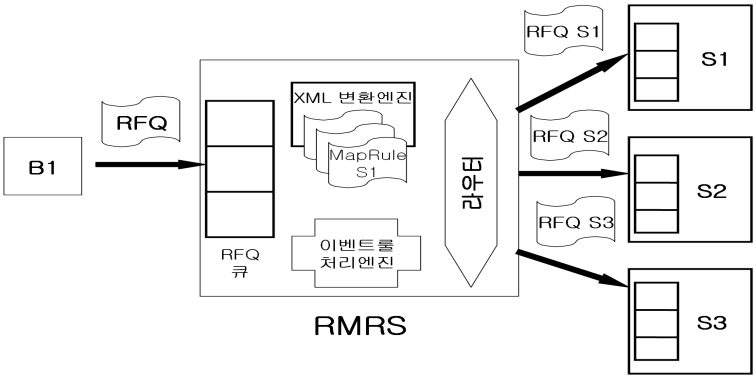
## 4. 사례연구 : RMRS를 활용한 의류 생산 업체의 B2B 거래 시나리오

본 장에서는 일반적인 비즈니스 환경에서 파트너와의 협업 시 RMRS가 어떻게 적용될 수 있는지 시나리오를 통해 적용해봄으로써 RMRS의 효용성을 평가하고 검증하도록 한다.

현실에서의 비즈니스는 매우 분산된 환경에 수요자와 공급자가 얽혀있는 상황에서 이루어진다. 업무 수행을 위하여 각 주체들 간에 여러 가지의 요구사항이 발생하고 이를 전산 화 하기 위하여 수많은 다른 시스템이 존재하기 때문이다. 이러한 시스템들 간의 연계를 위하여 다시 새로운 시스템이 도입되는 과정에서 새로운 프로그램의 개발이 필요해진다. 이때 각 시스템 별 환경은 대부분 다르므로 모든 요구사항들을 만족 시키는 것은 상당한 작업을 필요로 하게 된다. 메시징 표준인 JMS가 도입되고 문서표준인 XML의 일반적인 사용으로 인하여 이러한 어려움은 많은 점에서 개선되었으나 근본적인 해결책이 되지는 못하고 있다.

RMRS를 적용해 볼 시나리오는 의류 생산업체의 재료 견적요청 과정이다. 재료를 주문하여 의류를 생산하는 기업 B1과 각 재료를 공급하는 기업 S1, S2, S3이 있다고 가정했을 때 B1이 견적 요청서 (RFQ, Request for Quotation)를 통하여 의류 생산에 필요한 재료의 견적을 각 공급업체에 문의하는 과정을 구현해 보았다.

1대 다의 견적요청 업무 구현을 위해 일반적인 메시지 전달 방식을 사용할 경우 보통 다음과 같은 절차를 따르게 된다. 우선 견적요청을 하고자 하는 재료에 대한 견적 요청서를 각 공급자의 요구사항에 적합한 양식으로 작성한다. 이때 각 공급자가 다른 형태의 양식을 사용할 경우 개별적인 변환 프로그램




〈그림 8〉 시나리오 시스템 구성도

〈표 4〉 견적 요청 JMS 메시지

```
JMS_EVENT_TYPE
RequestForQuotation

XML Message Body
<RequestForQuotation id = "384791_RQ_GJS9483">
  <Buyer type = "partner" source_id = "PAR : 59g : A10">
    <CompanyName>B1</CompanyName>
    <Address>SinrimDong, KwanakGu, Seoul, SouthKorea</Address>
    <Phone>02-3252-5920</Phone>
  </Buyer>
  <Suppliers>
    <Supplier name = "S1">
      <OrderItems count = "2">
        <OrderItem oi_id = "1">
          <ProductId>FS235</ProductId>
          <ProductType>안감1</ProductType>
          <OrderCount>500</OrderCount>
        </OrderItem>
        <OrderItem oi_id = "2">
          <ProductId>FS243</ProductId>
          <ProductType>안감2</ProductType>
          <OrderCount>250</OrderCount>
        </OrderItem>
      </OrderItems>
    </Supplier>
    <Supplier name = "S2">
      <OrderItems count = "1">
        <OrderItem oi_id = "1">
          <ProductId>T93</ProductId>
           <ProductType>가죽</ProductType>
<OrderCount>350</OrderCount>
        </OrderItem>
      </OrderItems>
    </Supplier>
    <Supplier name = "S3">
      <OrderItems count = "1">
        <OrderItem oi_id = "1">
          <ProductId>AGH4</ProductId>
<ProductType>바깥감</ProductType>
<OrderCount>475</OrderCount>
        </OrderItem>
      </OrderItems>
    </Supplier>
  </Suppliers>
</RequestForQuotation>
```

을 개발 하여야 한다. 그리고 생성된 견적 요청서를 전송하는 프로그램을 작성한다. 즉 수행하는 전 과정에 걸쳐 프로그램의 개발이 필요하므로 구현이 매우 어려워진다.

이 절차를 RMRS를 통하여 처리할 경우 B1은 RMRS 도구를 통해 XML 변환 룰과 이벤트 룰을 정의하고 이를 배포한 후 RMRS의 JMS 큐로 XML로 작성된 견적 요청서를 전송하면 배포된 ECA룰을 통해 XML 변환 룰 변환과 각 공급자로의 라우팅이 이루어진다. RMRS를 적용한 시나리오 시스템 구성도는 <그림 8>과 같다.

B1이 견적요청을 위하여 RMRS로 전송할 JMS 메시지의 내용은 <표 4>와 같다.

위 견적 요청서는 공급자 S1로 부터는 안감 1과 안감 2를, S2로 부터는 가죽을 S3으로 부터는 바깥 감에 대한 견적요청을 담고 있다.

위와 같은 JMS 메시지가가 수신되었을 때 본문 검색을 통하여 각 공급자가 원하는 양식에 맞는 변환 룰을 지정하고, 공급자들에게 라우팅을 하기 위한 이벤트 룰을 작성하여야 한다.

위 <표 5>에 나타나 있는 이벤트 룰은 메시지 본문의 "/PurchaseOrder/OrderItems/Supplier" 엘리먼트를 기준으로 하여 3가지의 조건문을 수행하는 견적 요청을 위한 이벤트 룰의 예시이다. 각 조건문은 제조사의 name 애트리뷰트에 따라 각 제조사에 맞는 XML 변환 룰을 사용하여 변환 함수를 호출하고 각 제조사의 수신 큐를 목적지로 하여 메시지를 전송한다. 이 과정에서 앞서 <그림 6>에 제시된 XML 변환 룰 편집도구를 이용하면, <표 4>에 제시된 견적요청메시지의 일부분을 각 공급자에게 전달하기에 앞서 필요한 XML 변환 룰(Supplier 엘리먼트 내의

〈표 5〉 견적 요청을 위한 이벤트 룰

```
on
RequestForQuotation
for-each
($xml_body/PurchaseOrder/OrderItems/Supplier)
if
Supplier[@name. = 'S1']
do
XmlTransformation('P01_RFQ'), SendMessage('B1', 'S1')

if else
Supplier[@name. = 'S2']
do
XmlTransformation('P02_RFQ'), SendMessage('B1', 'S2')

if else
Supplier[@name. = 'S3']
do
XmlTransformation('P03_RFQ'), SendMessage('B1', 'S3')
```

각 엘리먼트들을 공급자에게 적합한 형태의 메시지로 변환하기 위한 룰)을 작성을 손쉽게 작성할 수 있다. <표 5>의 이벤트 룰을 적용하여 견적 요청서를 RMRS를 통해 전송하면 각 공급자에게 <표 6>, <표 7>, <표 8>과 같은 견적 요청서가 전달된다.

이상 RMRS를 사용하여 시나리오를 적용해 본 결과 XML 변환 룰 및 이벤트 룰의 설정만으로 새로운 프로그램 작성 없이 견적 요청서 전송을 처리 할 수 있었다. 특히 재료의 수량 변경이나 제조사 변동 등 요구사항이 바뀌는 경우가 발생 하여도 해당 룰만 변경하여 재적용하는 작업만 거치면 바로 서비스에 적용할 수 있으므로 매우 효율적이다. 적용한 시나리오의 메시지 변환이나 라우팅은 비교적 간단하지만 XML 변환도구의 응용으로 보다 복잡한 변환도 손쉽게 처리 할 수 있고, 이벤트 룰의 간단한 수정으로 추가

〈표 6〉 S1이 수신하는 견적 요청서

```
<RequestForQuotation id = "384791_RQ_GJS9483">
  <Buyer type = "partner" source_id = "PAR : 59g : A10">
    <CompanyName>B1</CompanyName>
    <Address>SinrimDong, KwanakGu, Seoul, SouthKorea</Address>
    <Phone>02-3252-5920</Phone>
  </Buyer>
  <OrderItems count = "1">
    <OrderItem oi_id = "1">
      <ProductId>FS235</ProductId>
      <ProductType>안감1</ProductType>
      <OrderCount>500</OrderCount>
    </OrderItem>
  </OrderItems>
</RequestForQuotation>
```

〈표 7〉 S2이 수신하는 견적 요청서

```
<RequestForQuotation id = "384791_RQ_GJS9483">
  <Buyer type = "partner" source_id = "PAR : 59g : A10">
    <CompanyName>B1</CompanyName>
    <Address>SinrimDong, KwanakGu, Seoul, SouthKorea</Address>
    <Phone>02-3252-5920</Phone>
  </Buyer>
  <OrderItems count = "1">
    <OrderItem oi_id = "1">
      <ProductId>T93</ProductId>
      <ProductType>가죽</ProductType>
      <OrderCount>350</OrderCount>
    </OrderItem>
  </OrderItems>
</RequestForQuotation>
```

〈표 8〉 S3이 수신하는 견적 요청서

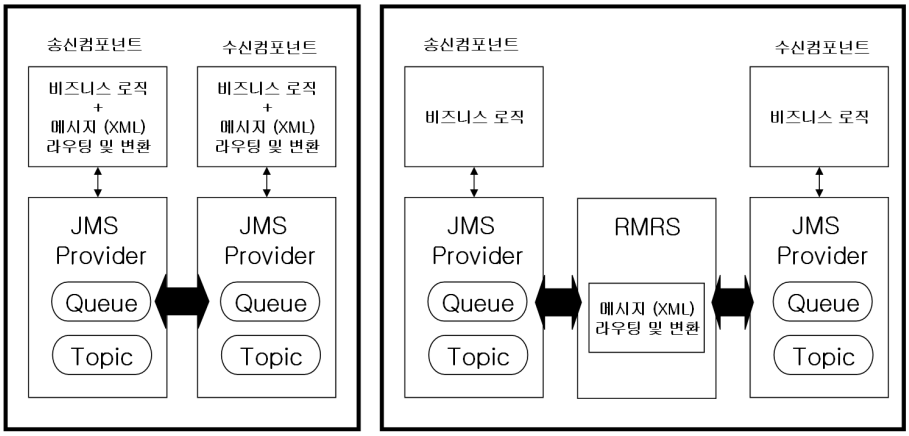
```
<RequestForQuotation id = "384791_RQ_GJS9483">
  <Buyer type = "partner" source_id = "PAR : 59g : A10">
    <CompanyName>B1</CompanyName>
    <Address>SinrimDong, KwanakGu, Seoul, SouthKorea</Address>
    <Phone>02-3252-5920</Phone>
  </Buyer>
  <OrderItems count = "1">
    <OrderItem oi_id = "1">
      <ProductId>AGH4</ProductId>
      <ProductType>바깥감</ProductType>
      <OrderCount>475</OrderCount>
    </OrderItem>
  </OrderItems>
</RequestForQuotation>
```

적인 라우팅을 지원 할 수 있다.

## 5. RMRS의 효율성 및 활용성에 대한 평가

MOM(Message Oriented Middleware)는 다양한 시스템을 연결·통합하기 위해 사용하는 미들웨어로 XML 메시지를 기반으로 송수신지 간의 정보의 교환을 지원한다. 하지만 이러한 MOM을 기반으로 시스템을 통합할 경우, 필요한 메시지의 송수신이나 트랜잭션 관리는 MOM이 제공하는 표준 API를 통해 해결할 수 있으나, 비즈니스 업무에 따른 실제적인 메시지변환이나 라우팅은 별도의 모듈이나 로직을 개발하여 해결하여야 한다. 일반적으로 이러한 모듈은 특정 비즈니스 컴포넌트의 내부로직에 포함되거나 별로의 컴포넌트로 되는데 문제는 이러한 것들을 통합적으로 관리 지원할 수 있는 체계가 없어 시스템의 재구축이나 유지보수가 어렵다는 것이다.

RMRS는 이러한 메시지 변환 및 라우팅을 통합관리 지원하기 위한 일종의 미들웨어 시스템이다. 즉, RMRS는 MOM과 같은 메시지 큐를 사용해 메시지를 주고받는 과정에서 메시지의 변환과 라우팅을 위한 룰의 생성과 관리를 통합적으로 관리 및 지원하기 위한 시스템이다. RMRS는 송신측 메시지큐와 수신측 메시지큐로 구성되는 2-tier 메시지 전달 구조를 송신측 메시지큐, RMRS, 수신측 메시지 큐의 3-tier 메시지 구조로 재구성하게 된다. 메시지 전달은 상용벤더들이 제공하는 MQ 시리즈들이 담당하게 되며, 메시지의 변환 및 라우팅은 RMRS가 담당하게 되는 형태이다. 따라서 RMRS를 이용할 경우, 비즈니스 컴포넌트의 설계 시 메시지 전달과 변환 및 라우팅의 과정을 MQ와 RMRS에 위임할 수 있어 컴포넌트 구조의 단순화와 독립성을 확보할 수 있고 이를 통해 전체 시스템의 유지보수성을 증대시킬 수 있다. 또한 룰을 통한 체계적인 메시지 변환 및 라우팅의 독립적




(a) RMRS 를 적용하기 전의 2-tier 구조 (b) RMRS 를 적용한 후의 3-tier 구조

〈그림 11〉 RMRS 적용 전과 적용 후의 송수신컴포넌트 구조의 비교

관리는 메시지 변환의 효율성과 로직의 재사용이라는 측면에서 전체 시스템의 성능향상 및 통합비용 감소에 기여할 수 있다.

## 6. 결 론

JMS 플랫폼과 XML 메시지는 분산된 이질적인 시스템간의 효율적인 정보 송수신을 위해 다양한 분야와 응용을 통해 적용되고 있다. 특히 JMS 플랫폼이 갖는 큐 기반의 메시지 송수신 특징을 활용하여 단순 일회성 전송이 아니라, 필요에 의해 정보를 재가공하면서 여러 차례의 재전송을 수행함으로서 분산된 이질적인 시스템 간의 데이터, 서비스를 통합하고 나아가 비즈니스 프로세스를 연결시켜 주는 핵심적인 도구로 활용되고 있는 실정이다.

이때 XML 문서는 다른 구조를 갖는 XML 문서 또는 다른 형태의 포맷으로 변환하기 쉽기 때문에 송수신되는 JMS 메시지에는 주로 XML 문서가 사용된다. 또한 JMS를 이용한 메시지 송수신은 많은 경우에 다양한 조건에 따라 동적으로 라우팅 되어야 하고 필요에 의해 변환 전후에 재전송 되어 여러 시스템 간에 연속적으로 송수신이 수반된다. 하지만, 이러한 필요성에도 불구하고 메시지 변환 및 라우팅 플랫폼의 부재로 특정 JMS 메시지 형태에 적합한 재전송 및 변환 모듈 및 프로그램을 변환 및 라우팅 시 마다 별도로 개발하는 문제점이 있었다.

이러한 문제의 해결을 위해 본 연구에서는 ECA 룰을 응용하여 JMS 환경에 적합한 이벤트 룰을 개발하였다. 이벤트 룰은 수신 JMS 메시지의 XML 문서 내용에 따라 동적으로 다른 구조의 XML 메시지로 변환하거나 또는 JMS 메시지를 구성하여 재송신을 위한 규칙 담는다. 또한 사용자는 필요에 의해 다양한 함수를 작성하여 이벤트 룰에서 응용할 수 있다. 또한 이러한 이벤트룰을 바탕

으로 한 동적 메시지 변환 및 라우팅을 위한 RMRS 시스템을 구현하고 실제 환경에 적용시켜 봄으로서 본 연구의 결과를 검증하고 그 효율성을 입증하였다.

그러나 본 연구는 JMS 플랫폼 환경에서 이벤트 룰과 XML 변환 룰을 적용하였기 때문에 그 이외의 메시지 미들웨어 및 MOM (Message Oriented Middleware)에는 적용하기 어려점이 있다. 따라서 향후 연구에서는 범용적인 메시지 라우팅 플랫폼을 기반으로 동적 메시지 변환 및 라우팅을 위한 아키텍처를 제시하고 이를 구현하여 제시하도록 할 것이다.

## 참 고 문 헌


[1] Diego Lopez de Ipina, Eleftheria Katsin, "An ECA Rule-Matching Service for Simpler Development of Reactive Applications," IEEE Distributed Systems Online 2001.

[2] J. Bailey, G. Papamarkos, A. Poulovassilis, P. Wood, "An Event Condition Action Language for XML," Web Dynamics 2004.

[3] T. Heimrich, G. Specht, "Enhancing ECA Rules for Distributed Active Database Systems," NODe 2002.

[4] James Bailey, Alexandra Poulovassilis, Peter T. Wood, "An Event-Condition-Action Language for XML," WWW 2002, 2002, pp. 486-295,

[5] Coulouris, G., Dollimore, J, Kindberg, T., "Distributed Systems : Concepts and Design," Addison Wisley, 1994.

[6] James Bailey, Alexandra Poulovassilis, Peter T. Wood. "An event-condition-action language for XML," Proceedings of the 11th international conference on World Wide Web 2002.

[7] G Papamarkos, A Poulovassilis, PT Wood. "Event-Condition-Action Rule Languages for the Semantic Web," Workshop on Semantic Web and Databases 2003.

[8] World Wide Web Consortium. XML Path Language (XPath), Version 1.0. See http://www.w3.org/TR/xpath, November 1999. W3C Recommendation.

[9] James Clark, XML Transformation (XSLT), version 1.0. W3C Recommendation, http://www.w3c.org/TR/xslt20/, 1999.

[10] Charles F. Goldfarb, Paul Prescod, The XML handbook, Prentice Hall PTR, Upper Saddle River, NJ, 1999.

[11] Apache XML project, "Xalan," http://xml.apache.org/xalan/index.html, 2000.

[12] Lionel Villard, Nabil Layaida. An Incremental XSLT Transformation Processor for XML Document Manipulation. In Proceedings of the 11th International Conference on the World Wide Web, 2002.

[13] Y. Jin, “A Design Pattern for Active Rule-Based System Architectures,” Proceedings of the IEEE International Conference on Information Reuse and Integration, 2005.

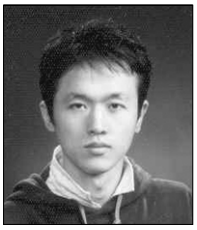
[14] H. Ma, H. Johansson, K.Orsborn, “Distribution and synchronisation of engineering information using active database technology,” Advances in Engineering Software, Vol. 36, No. 11/12, 2005, pp. 720-728.

[15] Aberdeen Group, “The Power of Rules-Driven Processing,” Internal Report, Aberdeen Group, 2000.

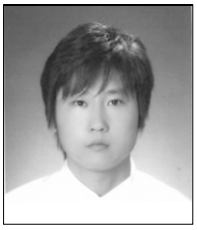
[16] Galip Aydin, Ali Kaplan, Ahmet E. Topcu, Beytullah Yildiz, Marlon Pierce, and Geoffrey Fox, “An XML Based System for Dynamic Message Content Creation, Delivery, and Control,” Information and Knowledge Sharing IKS 2002 Conference. St. Thomas, Virgin Islands, USA. November 2002, pp. 18-20.

## 저 자 소 개

조풍연 (E-mail : cpy@metabuild.co.kr)
1996년 동국대학교 정보산업대학원 전자계산과 (공학석사)
2005년 숭실대학교 일반대학원 컴퓨터학과
관심분야 EAI/BPM, 웹서비스, SOA/ESB, RFID , XMLDB



최재현 (E-mail : uniker80@gmail.com)
2006년 숭실대학교 일반대학원 컴퓨터학과 졸업 (공학석사)
2006년~현재 숭실대학교 일반대학원 컴퓨터학과 박사과정
관심분야 소프트웨어 개발방법론. 소프트웨어프로젝트 관리,
소프트웨어 아키텍처, SOA/ESB, 웹서비스



박제원 (E-mail : jwpark5656@hotmail.com)
2006년 숭실대학교 컴퓨터학과 (석사)
2006년~현재 숭실대학교 컴퓨터학과 박사과정
관심분야 소프트웨어테스팅, 소프트웨어 프로세스, 웹서비스,
SOA/SB

이남용 (E-mail : nylee@ssu.ac.kr)
1983년 고려대학교 경영정보학과 (석사)
1993년 미시시피주립대학 경영정보학과 (경영학박사)
1979년~1983년 국군정보사령부 정보처 정보시스템분석 장교
1983년~1999년 한국국방연구원 군수체계 및 정보체계연구부장
2000년 한국전자거래학회 논문편집위원장
2004년 한국정보통신기술사협회 회장
1999년~현재 숭실대학교 컴퓨터학과 교수
관심분야 소프트웨어테스팅, 시스템엔지니어링 등