기업간 상호운영성을 위한 BSI 엔진의 설계

* 오동근[O] 이정훈 홍정선 정재우 김광훈 ** 최성환 황재각 이용준
* 경기대학교 전자계산학과 워크플로우 기술 연구실
** 한국전자통신연구원 정보화기술연구소
dkoh@kyonggi.ac.kr

Design of BSI Engine for Interoperability

* Dong-Keun Oh[O] Jung-Hoon Lee Jung-Sun Hong Jae-Woo Jung Kwang-Hoon Kim
** Sung-Hwan Choi Jae-Gak Hwang Yong-Jun Lee
* Dept. of Computer Science, Kyonggi University
** Electronics and Telecommunications Research Institute

요 약

ebXML은 단일한 방법으로 기업간 비즈니스 프로세스의 자동화를 지원할 수 있는 표준이다. 본 논문은 ebXML의 CPA(Collaboration Protocol Agreement) 및 BPS(Business Process Specification)에 정의된 내용에 따라 프로세스 자동화를 관리하는 BSI(Business Process Interface) 엔진의 주요 설계 내용을 기술한다. 고려 사항으로는 BPSS 1.05, CPA 2.0을 지원하였다.

1. 서 론

최근 국내의 산업현장에서는 비즈니스 프로세스 기술의 중요성에 대하여 인식의 폭이 과거와는 달리 상당히 향상되었음을 쉽게 알 수 있다. 이는 업무자체의 소요 시간보다 업무간의 전이 시간을 줄이는 것이 업무처리에 대한 월등한 생산성 향상에 있음을 아는 것으로서 바람직한 현상이라 말할 수 있다. 또한 전통적인 산업자원분야의 급속한 디지털화 및 XML, 인터넷 기술의 발달은 전자상거래 및 전자무역의 활성화와 더불어 전자정부 등에서 조직간의 프로세스 연동성 및 통합의 필요성이 강조되고 있기 때문에, 이를 구현하기 위한 프로세스 중심 정보기술의 연구 개발에 많은 투자와 관심이 집중되고 있다. 이의 핵심 기술은 워크플로우(비즈니스 프로세스) 관리 기술로서, 조직내의 정보기술 인프라와 조직간의 정보관리 및 통합을 위한 기술 인프라 두 측면에서 빠르게 진행되고 있다. 본 논문에서 제안하는 BSI 엔진은 위의 두 흐름 중에서 기업간의 상호운영성을 담당하는 것으로, ebXML의 CPA 및 BPS에 정의된 내용에 따라 프로세스 자동화를 관리해 주는 컴포넌트이다. 다음 2장에서는 BSI의 정의를 살펴보고, 3장에서는 BSI의 주요 설계에 대한 내용을 기술하고 4장에서 결론을 맺는다.

2. BSI(Business Service Interface)의 정의

최근의 몇 년 동안 프로세스의 관리를 나타내는 비즈니스 프로세스 관리(BPM)와 기존의 워크플로우 관리(WfM) 기술과의 차이점 또는 상호 개념적 정의상의 혼란이 국내외 연구자들간에 논쟁의 이슈가 되었다. 즉, BPM은 주로 BPMI 표준화 기구측에서, 그리고 WfM은 WfMC 측에서 주장하는 용어로 설명이 가능하다. 그런데 최근 두 기구간에 Joint Member Meeting을 여러 차례 개최하여 워크플로우 및 비즈니스 프로세스 관련 단일의 표준안을 개발하고자 활발히 협력방안을 모색중에 있으며, 지난 2003년 6월에 있었던 Joint Member Meeting에서는 워크플로우라는 용어와 비즈니스 프로세스라는 용어를 상호 동일한 개념으로 사용하기로 합의하였다. 이는 그 동안 WfM (Workflow Management)과 BPM(Business Process Management) 간의 개념상의 혼란을 종식시켰다는 면에서 상당한 의미를 갖는다고 할 수 있다.[1]

한편 BSI의 정의는 ebXML 스펙에서 언급되는 용어로서 위의 비즈니스 프로세스 관리의 또다른 용어로 설명할 수 있다. 그런데 BSI는 BSI 만을 위한 스펙 문서가 따로 존재하지 않고 BPSS, CPA 스펙 문서등에서 간헐적으로 언급되는 특징이 있다. 이러한 언급들을 종합해 보면 ebXML BPSS 규약에 따라 거래 파트너들이 표준화된 형태로 전자문서를 주고 받으며 기업간 협업을 수행할 수 있도록 해주는 상호운영성에 초점을 맞춘 시스템이라고 간단히 정의할 수 있다. 또한 BSI의 기본 기능은 1) CPA에 정의된 알고리즘에 따라 비즈니스 프로세스의 수행 2) 모든 프로세스 과정에서 처리되는 상황의 로깅(Logging) 3) 두 파트너간의 협업, 비즈니스 트랜잭션의 관리 및 라우팅(Routing) 4) 에러 핸들링등의 기본적인 기능을 제공해야 한다.

3. BSI의 설계

본 BSI 엔진의 설계는 [2]의 구현 결과물의 연장선에 있는 과제로서, [2]는 기본적으로 BPSS 1.01 버전에 기초를 두었고, 1.05 버전을 이번 설계에 확장하였다. BPSS의 1.05는 그 이전 버전과의 역호환이 안되는 점 및 다자협업등 많은 부분들이 논쟁의 여지와 확정되지 않은 내용들이 아직까지 많이 남아 있지만[3], 2.0에서는 많은 부분들이 안정화 될 것으로 예상한다. 다음 절부터는 BSI의 주요 설계 내용을 기술한다.

3.1 BSI 전체 시스템 구성도

그림 1은 본 엔진이 적용되는 BSI 전체 시스템의 구성 요소를 보여주고 있다. 레거시 애플리케이션 또는 모니터링 애플리케이션은 비즈니스 거래에 필요한 수행 환경 및 모니터링 정보를 제공하는 유저 인터페이스이다. 레거시 애플리케이션은 문서 작성을 한 후 본 논문에서 설계한 BSI 엔진에게 RMI 호출을 통하여

XML 문서로 전송한다. 이때 전송되는 인터페이스는 향후 확장성(SOAP)을 고려하여 이동되는 데이터를 XML 문서로 설계하였다. 그리고 BSI 엔진은 애플리케이션으로부터 받은 문서를 ebXML 메시지를 생성하여 MTS에게 전달, 상대편의 MTS에게 전달한다. 이때 BSI 엔진과 MTS간의 통신은 JMS를 사용하는 특징이 있다.

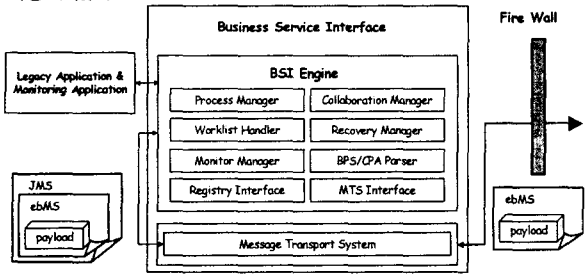

그림 1. BSI 전체 시스템 구성도

3.2 비즈니스 트랜잭션 엑티비티 처리 다이어그램

그림 2는 협업에 있어서 거래의 최소 단위인 비즈니스 트랜잭션 엑티비티를 처리하는 엑티비티의 다이어그램을 추상화한 그림이다. 왼쪽은 거래를 요청하는 부분의 흐름을 나타내고 오른쪽은 이에 응답하는 거래의 순서적인 흐름을 묘사하고 있다.

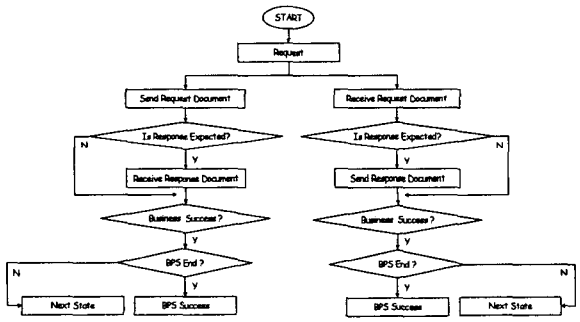

그림 2. 엑티비티 처리 다이어그램

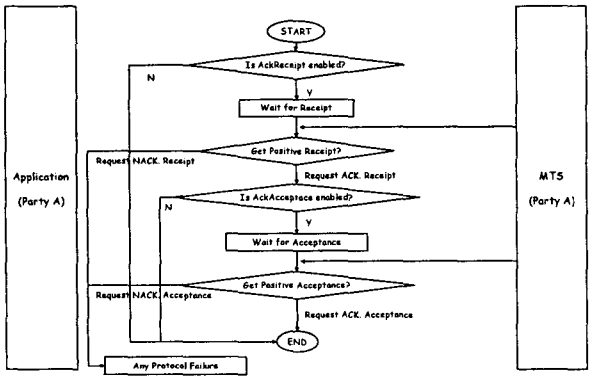

그림 3. Get And Check Request Signal

그림 3은 요청 엑티비티에서 상대 참여자에게 보낸 문서에 대한 시그널을 체크하는 그림을 보여주고 있다. 그림에서 두 시그널에 대해서 체크되어 있고 부정(Negative)의 신호가 온다면 모두 AnyProtocolFailure의 상태로 옮겨지는데, 이때 BSI 엔진은 트랜지션의 "Condition Guard"에 따라서 그 다음의 상태로 전이시킨다. 그림 4는 AnyProtocolFailure와 같은 예외 사항을 분류한 그림이다.

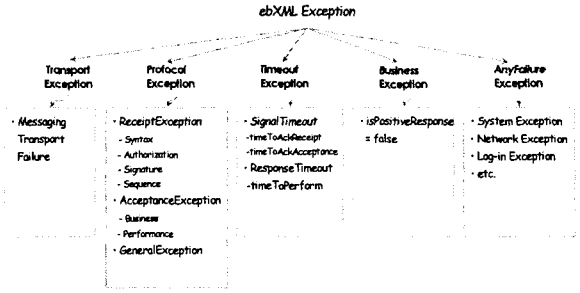

그림 4. ebXML 예외사항 분류

3.3 재시도(Retry) 메커니즘

BPSS 1.05 버전에서 추가된 "Retry Count"는 UMM에서 나온 속성으로 비즈니스 레벨의 재시도를 뜻한다. 즉, 비즈니스 트랜잭션 엑티비티, 그리고 각각의 시그널에 대한 "Time To Perform"의 속성을 체크하여 그 시간을 넘겼을 경우 그림 4의 "Timeout Exception"으로 전이되고, "Retry Count" 수만큼 트랜잭션의 롤백을 처리한다. 이는 MSH(Message Service Handler)의 재시도와는 독립적이다. 이때 CPA에 MTS의 Reliable Message 속성이 체크되었을 경우 트랜스포트단에서 메시지의 전송을 보장시키고, 만약 그 설정된 시간동안 처리가 안되었을 경우 MTS는 BSI 엔진에게 통보, 엔진은 이에 대한 재시도를 안하고 바로 Failure로 전이 시킨다.

그런데 거래를 요청하는 쪽에서 실패가 났을 경우, 응답하는 참여자에게 Failure Exception을 보내는 시그널이 BPSS 1.01 버전에서는 존재했으나, 1.05 버전에서는 Exception의 시그널 전송을 없앤 것이 특징이다. 따라서 거래를 요청하는 쪽에 Failure가 발생 또는 재시도를 시행했을 경우, 비즈니스 거래에 응답하는 참여자는 그러한 사실을 판독할 수 있는 알고리즘이 필요하다.

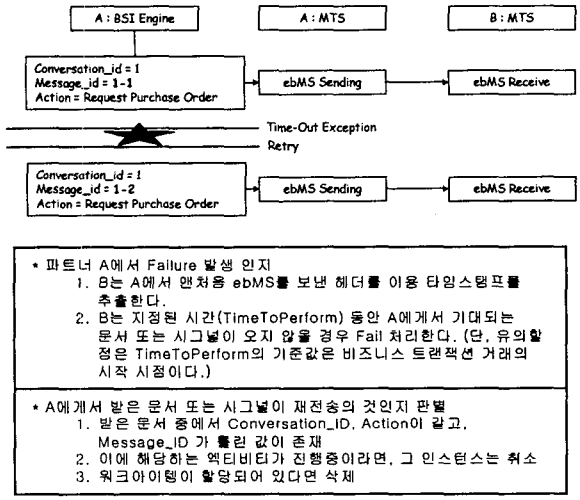

그림 5. Failure 인지 알고리즘

그림 5는 요청 문서를 A에서 발생시켜 B에게 전달하는 모습을 도식화한 것과 Failure 발생 및 재전송의 유무를 판단하는 간단

한 알고리즘을 나타낸다. Conversation ID는 협업(양자협업)을 식별할 수 있는 ID이고, Action은 비즈니스 트랜잭션 엑티비티에서 요청 혹은 응답의 엑티비티의 이름 또는 비즈니스 시그널의 이름을 나타내는데, 그 이외의 것은 MSH 자체에서 쓰는 시그널이라 가정한다. 그리고 Retry는 응답 엑티비티에는 권한이 없음을 주의하여야 한다.

3.4 Recovery Handling

다수의 프로세스가 처리되는 가운데 시스템 장애가 발생했을 경우, 이러한 문제들을 복구시키는 문제는 BSI 엔진 뿐만 아니라 여타 시스템에서도 제일 중요한 요소로 고려된다. 본 엔진의 에러 복구 메커니즘의 기본 아이디어는 협업 프로세스 및 비즈니스 트랜잭션 엑티비티의 상태 변화에 따라 그 상황을 DB로 저장하는데, 중요한 것은 비즈니스 트랜잭션 엑티비티의 상태를 효율적으로 정의하는데 있다. 여기서 고려할 사항은 다음과 같다. ebXML 거래의 기본은 두 파트너 간의 문서의 흐름과 Receipt/AcceptanceAcknowledgement 시그널 두개의 흐름이 존재한다. 참고로 BPSS 1.01에서는 거래의 응답에서 Acceptance Acknowledgement의 시그널이 존재하지 않고, 1.05 버전에서는 요청과 응답 두거래에 모두 존재한다. 또한 이러한 거래는 시스템이 자동으로 수행하거나 사용자가 일을 할당 받아서 처리하는 두 형태가 모두 가능하다. 따라서 이러한 작업을 처리하는 워크아이템의 개념이 필수적이다. 이러한 분류 작업에 근거한 비즈니스 프로세스 엑티비티의 상태는 표1에서 보는 바와 같이 분류하였다. (참고로 현재 이 부분에 대한 연구는 EJB 기술과의 연동을 고려해서 진행 중에 있다.)

표 1. 비즈니스 프로세스 엑티비티의 상태 상수

01	STATE_START
02	STATE_END
03	STATE_RECEIVE_REQUEST_DOCUMENT_FROM_APPLICATION
04	STATE_RECEIVE_REQUEST_DOCUMENT_FROM_MTS
05	STATE_SEND_REQUEST_RECEIPT_ACK_TO_MTS
06	STATE_RECEIVE_REQUEST_RECEIPT_ACK_FROM_MTS
07	STATE_SEND_REQUEST_RECEIPT_NACK_TO_MTS
08	STATE_RECEIVE_REQUEST_RECEIPT_NACK_FROM_MTS
09	STATE_SEND_REQUEST_ACCECPTANCE_ACK_TO_MTS
10	STATE_RECEIVE_REQUEST_ACCECPTANCE_ACK_FROM_MTS
11	STATE_SEND_REQUEST_ACCECPTANCE_NACK_TO_MTS
12	STATE_RECEIVE_REQUEST_ACCECPTANCE_NACK_FROM_MTS
13	STATE_SEND_RESPONSE_DOCUMENT_TO_MTS
14	STATE_RECEIVE_RESPONSE_DOCUMENT_FROM_MTS
15	STATE_SEND_REPONSE_RECEIPT_ACK_TO_MTS
16	STATE_RECEIVE_REPONSE_RECEIPT_ACK_FROM_MTS
17	STATE_SEND_REPONSE_RECEIPT_NACK_TO_MTS
18	STATE_RECEIVE_REPONSE_RECEIPT_NACK_FROM_MTS
19	STATE_SEND_REPONSE_ACCECPTANCE_ACK_TO_MTS
20	STATE_RECEIVE_REPONSE_ACCECPTANCE_ACK_FROM_MTS
21	STATE_SEND_REPONSE_ACCECPTANCE_NACK_TO_MTS
22	STATE_RECEIVE_REPONSE_ACCECPTANCE_NACK_FROM_MTS
23	STATE_PROTOCAL_SUCCESS
24	STATE_PROTOCAL_FAILURE
25	STATE_BUSINESS_SUCCESS
26	STATE_BUSINESS_FAILURE

3.5 BSI 엔진의 클래스 및 시퀀스 다이어그램

본 엔진의 주요 클래스는 그림 6과 같다. 레거시 애플리케이션과 직접적인 관련이 있는 클래스는 WorklistHandlerBean (해야할 작업의 목록을 관리)과 MonitorBean(모니터링 정보를 제공), LoginBean(사용자 인증)등이 있는데, 이러한 빈(Bean)들과의 단일한 창구를 사용하기 위하여 ClientInterfaceBean을 앞단에 둔 특징이 있다. 또한 MTSBean은 MTS에서 보내는 메시지를 수신하기 위한 리스너 부분이 구현되어 있고, 또한 특징적인 것은 중간 단에 MessageBean을 통하여 내부적으로 EJB의 메시지 드리븐 빈을 사용한다. MessageBean의 도입으로 내부 큐를 사용하여 대용량 처리에 있어 보다 나은 효율을 기대할 수 있을 것으로 기대한다. 또한 커널 부분은 CollaborationManager Bean을 통해서 협업 인스턴스 및 그에 속한 비즈니스 트랜잭션 엑티비티, 시그널 처리등을 관리한다.

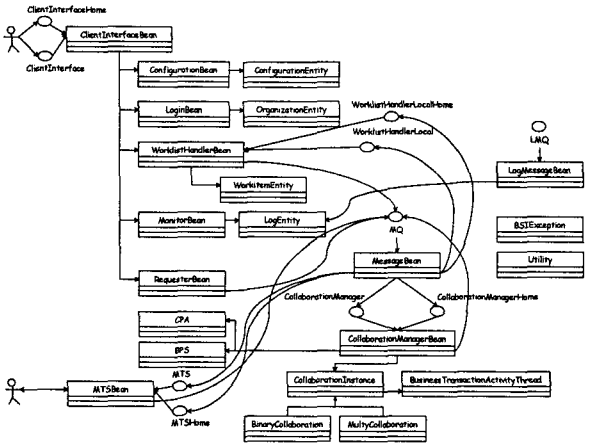

그림 6. BSI 엔진의 주요 클래스 다이어그램

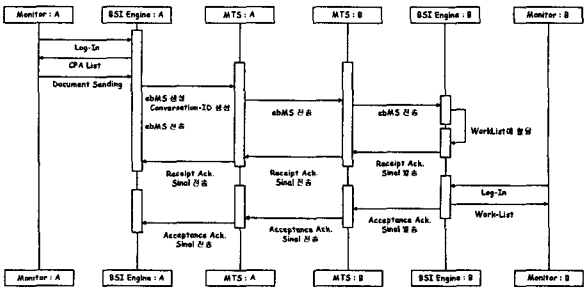

그림 7. 인스턴스 생성 시퀀스 다이어그램

4. 결 론

ebXML은 기업간 비즈니스 거래의 자동화를 제공해 주는 주목받는 표준중의 하나이고, 표준 스펙은 현재 기본 뼈대가 완성이 되었다. 하지만 그 내용이 현재 진행중인 관계로 실제 구현함에 있어 다른 스펙과(BPSS, CPA등)의 개념상 모호함, 추상적인 부분이 많아, ebXML 솔루션 개발자들에게 많은 어려움이 따른다. 이런 시점에서 본 논문은 스펙의 내용을 구체화하여 BSI 엔진의 설계 및 내용을 기술하였다. 향후 연구로는 EJB 프레임워크 기술을 적용, BSI의 대용량 처리에 대해서 진행할 예정이다.

참고문헌

[1] 김광훈, "워크플로우 기술 II", TTA 저널, 88호, pp.120-133, 2003.05

[2] 오동근 외 6명, "ebXML 기반의 전자물류 비즈니스 프로세스 관리 시스템 엔진", 한국정보과학회 춘계학술발표논문집, 30권1호, pp.608-682, 2003.04

[3] Birgit Hofreiter, Christian Huemer, Wolfgang Klas, "ebXML:Status, Research Issues, and Obstacles", RIDE'02, 2002