한국경영과학회/대한산업공학회 2003 춘계공동학술대회
2003 년 5 월 16 일-17 일 한동대학교(포항)

상태 기반 협상 모델을 이용한 동적 비즈니스 프로세스 통합
Dynamic business process integration using state-driven brokering models

윤장혁*, 이순재*, 김광수*
*정보시스템 연구실, 포항공대 산업공학과

국문요약

웹서비스 기술이 발전함에 따라 분산환경의 웹기반 서비스들을 조합한 복잡한 비즈니스 프로세스를 구현할 수 있게 되었다. 이러한 비즈니스 프로세스는 일반적으로 비즈니스 활동(activity)들의 순서와 로직을 포함한 이벤트 기반 모델(event-driven model)로 표현되고 있다. 하지만 비즈니스 활동을 수행하기 위해서는 불특정 다수의 대안 서비스들 중 하나를 런타임-바인딩(runtime-binding)하여 해당 비즈니스 활동을 수행해야 하는 경우가 빈번히 일어나는데, 이는 기존의 이벤트 기반 모델링 기법만을 이용해서 해결하기 어렵다. 따라서 본 논문에서는 비즈니스 프로세스를 동적으로 통합할 수 있도록 상태 기반 협상 모델링(state-driven brokering modeling : SDBM)을 기존 모델링 기법에 확장한 아키텍쳐를 제안한다. SDBM은 이벤트 기반 모델링에서 표현하기 힘든 대안 서비스들의 동적 런타임-바인딩을 가능하게 하는데, 이는 대화 정책(conversation policy)을 구현함으로써 가능해 진다. 본 논문에서는 핵심 비즈니스 프로세스에 독립적이며 웹기반의 분산 비즈니스 프로세스 간의 유연한 통합과 자동화된 협업을 수행할 수 있는 확장된 메커니즘을 웹서비스 기술로 구현하였다.

Keywords : 비즈니스 프로세스, 동적 통합, 협상 모델, 상태 기반, 웹서비스, e-비즈니스

1. 서문

웹을 통해 제공되는 표준화된 웹기반 서비스, 즉 웹서비스 기술의 발전은 표준 인터넷 프로토콜을 사용하여 비즈니스 내,외부 애플리케이션으로의 접근을 자유롭게 하였다.[9] 이와 같은 자유로운 애플리케이션으로의 표준화된 접근 방법은 애플리케이션 구축 방식을 컴포넌트 기반의 개발로 바꾸어 가고 있다. 이러한 웹서비스들 중 비즈니스 트랜잭션에서 일어날 수 있는 개별 활동(activity)으로 볼 수 있는 것들을 조합하여 하나의 복잡한 웹서비스 즉, 비즈니스 프로세스를 표현하고 구동할 수 있도록 하기 위해서 BPEL4WS(Business Process Execution Language for WebServices)가 발표되었다.[4] 이는 단순한 모듈 이라는 개념을 가진 기존 웹서비스를 애플리케이션으로서의 기능을 넘어서 커다란 비즈니스를 구성하는 개별 활동(activity)으로 볼 수 있도록 하였다. 이와 관련하여 웹서비스들을 이용하여 구조적으로 표현된 비즈니스 프로세스의 실행 시에 결함 없이 정상적 수행을 할 수 있도록 비즈니스 트랜잭션 프로토콜(WS-Transaction)[6]과 이를 배치(deploy)할 수 있는 프레임워크(WS-Coordination)[5]에 대한 명세서(specification)들은 웹서비스를 비즈니스 단위 활동으로 이용할 수 있다는 정당성을 가중시키고 있다.

웹기반 서비스들을 이용하여 비즈니스 프로세스 모델링을 할 수 있도록 하는 현재의 모델링 언어들은 이벤트 기반의 모델링(event-driven modeling)을 바탕으로 하고 있다. 이벤트 기반의 모델링이란 '특정 비즈니스 활동의 수행'이라는 사건(event)이 일어나는 것을 프로세스 전개의 기준으로 보고 모델링함을 의미한다. 이러한 프로세스 표현 기법은 프로세스에 다양한 로직을 포함하여 구조적인 프로세스를 표현하는 데 좋기 때문에 널리 사용되고 있다.

하지만 이벤트 기반의 비즈니스 프로세스 구성은 프로세스 하위 활동(activity)들의 동적 특성을 제한하여 버린다. 실제로 하나의 비즈니스 프로세스가 구동될 때, 특정 비즈니스 활동을 수행하기 위해서는 불특정 다수의 대안 웹서비스 또는 비즈니스 프로세스 중 하나를 런타임-바인딩(runtime-binding)하여 해당 비즈니스 활동과 연결하고 이 때에 수행해야 하는 경우가 빈번히 일어난다. 하지만 이벤트 기반 비즈니스 프로세스 모델링은 비즈니스 활동들과 웹기반 서비스들 간의 연결관계를 프로세스 내부에 명시적으로 표현하므로 런타임 시의 동적 바인딩을 효과적으로 표현하기 어렵다.

따라서 본 논문에서는 비즈니스 프로세스 구동시에 대안 웹기반 서비스를 중심 비즈니스

프로세스에 동적으로 통합할 수 있도록 상태 기반 협상 모델(state-driven brokering model)이 구현된 브로커링 어댑터를 기존 프로세스 모델링에 확장한 아키텍쳐를 제시한다. 상태 기반 협상 모델은 이벤트 기반 모델에서 표현하기 힘든 대안 서비스들의 동적 런타임-바인딩을 가능하게 할 수 있으며, 이는 대화 정책(conversation policy)을 구현함으로써 가능해 진다.

본 논문에서는 핵심 비즈니스 프로세스에 독립적이며 웹기반의 분산 비즈니스 프로세스 간의 유연한 통합과 자동화된 협업을 수행할 수 있는 확장된 메커니즘을 웹서비스 기술을 이용하여 구현하였다.

논문의 2장에서는 이벤트 기반의 비즈니스 프로세스 모델링에 대한 검토와 이를 이용할 때의 비즈니스 동적 통합에의 문제점을 살펴보고, 3장에서는 확장된 메커니즘인 상태 기반 협상 모델링의 구체적인 이용방안을 살펴본다. 그리고 4장에서는 확장된 아키텍쳐의 구현, 5장에서는 런타임-바인딩 시에 가정했던 전제들과 좀더 동적인 비즈니스 프로세스의 통합과 협업을 위해서 차후에 연구, 개발되어야 할 사항에 대해서 논의하였다.

2. 이벤트 기반 비즈니스 프로세스 모델링 (event-driven business process modeling)

2.1 BPEL4WS[4]

오늘날 웹기반 서비스들은 상호간 통신할 수 있고 스스로를 광고할 수 있으며 산업계 표준을 사용하여 발견 및 호출될 수 있다. 이러한 서비스들을 비즈니스 프로세스로 함께 연결 또는 조합하기 위해서 Business Process Execution Language for Web Services (BPEL4WS, 이하 BPEL)가 탄생하였다. BPEL은 WSFL[14]과 XLANG[13]의 장점들을 패키지로 묶어 자연스러운 방식으로 여러 종류의 비즈니스 프로세스를 구현할 수 있도록 지원한다. BPEL은 구현 언어일 뿐만 아니라 추상 프로세스 개념을 사용하여 비즈니스 프로세스의 인터페이스를 설명하는데 사용될 수 있다.

또한, BPEL 프로세스는 기본적으로 알고리즘 표현 같은 플로우 차트로도 볼 수도 있다. 프로세스의 각 단계를 활동(activity)이라고 하는데 이들의 역할은 다음과 같다

① 웹 서비스 작동 호출(<invoke>)
② 외부의 메시지 기다리기(<receive>)
③ 인풋/아웃풋 작동의 응답 (<reply>)
④ 얼마 동안 기다리기 (<wait>)
⑤ 데이터 복사 (<assign>)
⑥ 잘못된 부분 지적(<throw>)
⑦ 서비스 인스턴스 종료 (<terminate>)
⑧ 아무것도 하지않기(<empty>)

이러한 활동들은 BPEL에서 제공하는 다음과 같은 플로우 로직들을 사용하여 좀더 복잡한 알고리즘으로 조합될 수 있다.

① 주문된 순서를 정의 (<sequence>)
② "case 문" 브랜칭 (<switch>)
③ 루프 (<while>)
④ 선택적 경로 중 하나 실행 (<pick>)
⑤ 병렬 실행 기능(<flow>)
⑥ 링크를 통한 실행 순서 제한

BPEL은 <throw>와 <catch> 구문를 통해 프로세스 실행 시에 발생할 수 있는 예외에 대한 해결책을 제시하고 있다. BPEL에 대한 오류 개념은 WSDL의 오류 개념과 관련이 있고 그 기반 위에 구현된다. 또한, BPEL은 보상(compensation) 개념도 지원하는데, 이는 프로세스 디자이너가 되돌릴 수 없는 특정 활동에 대한 보상 활동을 구현할 수 있도록 한다.

이러한 BPEL 문서는 IBM의 BPEL4J와 같은 BPEL 엔진에 의해서 이용되어 실제 프로세스를 구동할 수 있고 프로그래밍적인 성격이 강한 BPEL Scenarion™[11]로 구현되어서 Web Logic 애플리케이션 서버 위에서 구동될 수도 있다.

2.2 이벤트 기반 프로세스 모델의 구현에서 프로세스 동적 통합에의 문제점

기본적으로 BPEL에서 표현되는 정보는 WSDL[15]과 연동되어 특정 이벤트가 일어난다는 것을 알고 있음을 가정하고 있다. 그러므로 BPEL로 표현된 비즈니스 프로세스 안에 해당 비즈니스 활동을 수행하기 위해서 그림 1과 같이 실제 사용될 웹서비스에 대한 레퍼런스(reference)를 명시적으로 표현해야 한다.

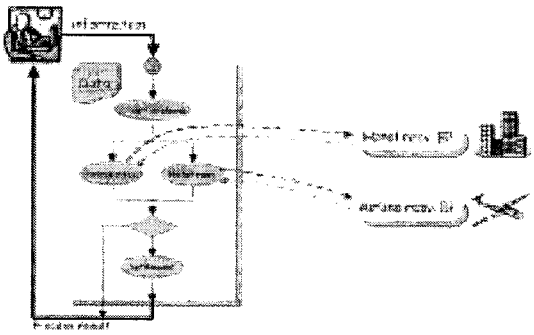


그림 1. BPEL에서 웹기반 서비스와의 명시적인 연결

하지만, 비즈니스 프로세스를 구성하는 활동은 추상적으로 명시되어야 할 경우가 있을

수 있다. 다시 말해, 비즈니스 활동을 실제로 수행할 웹서비스에 대한 명확한 바인딩이 이루어지지 않은 경우가 그렇다. 그림 2는 여행 패키지 예약 대행 회사의 비즈니스 프로세스의 예를 보여주고 있다.

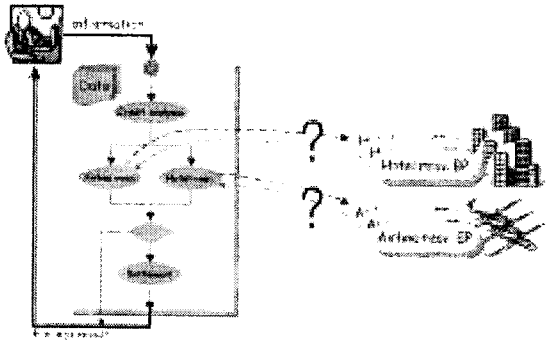


그림 2. 비즈니스 프로세스의 동적 바인딩이 필요한 경우

그림 2와 같이 다양한 비즈니스 파트너의 비즈니스 프로세스와 핵심 비즈니스 프로세스(여행 패키지 예약 대행회사의 비즈니스 프로세스)가 통합되어야 할 경우가 있다고 하자. 그러면 핵심 비즈니스 프로세스에서 첫 번째로 해결해야 하는 것은 하나의 비행기 예약(airline reservation) 활동을 수행하기 위해 이용 가능한 불특정 다수의 비행기 예약 비즈니스 프로세스가 참여할 수 있도록 하는 것이다.

두 번째로는 비행기 예약 비즈니스 활동과 이를 실제 수행할 수 있는 비즈니스 프로세스들의 동적 연결을 가능하게 해야 한다는 것이다.

예를 들어 고객이 시카고로 xx월 xx일에 여행을 가고자 여행 예약 대행사에 자신의 정보를 보내었을 때에 고정적으로 이용되는 비즈니스 활동(신용 조회 (credit analysis) 서비스 : 이것은 항상 정해진 곳에 서비스를 요청하는 경우로 가정)과 비행기 예약을 실제로 수행하는 A라는 비즈니스 프로세스가 바인딩 될 수 있겠다. 하지만, 고객이 LA로 여행을 가고자 여행 대행 비즈니스 프로세스에 자신의 정보를 보내면 B라는 비행기 예약 비즈니스 프로세스를 이용할 수도 있다는 것이다. 이와 같이 다른 이용 가능한 호텔, 자동차 예약 활동 들에 대해서도 동일한 과정을 가진다고 생각할 수 있다.

하지만 이와 같은 프로세스 수행시점에서 불특정 다수의 외부 비즈니스 프로세스들과의 동적 바인딩은 핵심 비즈니스 프로세스에서 중심사항이 아니다. 그러므로 핵심 프로세스에 의해서 이용될 수 있는 비즈니스 프로세스들을 명시적으로 모두 나열하여 비즈니스 컴포지션(composition)을 하는 것은 핵심 비즈니스 프로세스를 복잡하게 한다.

뿐만 아니라 기존의 이벤트 기반 모델링(event-driven modeling)으로는 앞서 설명한 대안 비즈니스 프로세스들과의 동적인 연결을 표현하는 것은 힘들다. 왜냐하면 비즈니스 파트너(개별 예약 웹기반 서비스를 제공해주는 비즈니스 프로세스)들은 불특정 다수일 뿐만 아니라 예약 대행 비즈니스 프로세스와의 거래에 있어서 참가, 탈퇴가 비교적 자유롭기 때문이다. 또한 구동 시에도 핵심 비즈니스 프로세스는 실제로 어떤 대안 비즈니스 프로세스와 연결되어 사용될 지를 알 수 없기 때문에 고객으로부터 전송되는 모든 정보에 따라 BPEL과 같은 비즈니스 프로세스 모델링 언어 또는 BPEL과 같은 내용을 수행하는 프로그래밍 로직으로 핵심 비즈니스 프로세스가 실제 구동시에 생길 수 있는 동적 바인딩에 대한 내용을 모두 표현할 수는 없는 것이다.

결국 여행 예약 대행 회사에서도 현재 핵심 비즈니스 프로세스에 통합되고자 참가하는 파트너가 누구인지는 관심의 대상이 되지 않는다. 단지, 여행자의 목적에 어긋나지 않고 최소한의 비용으로 예약을 할 수 있는 비행기 예약 비즈니스 프로세스를 찾아서 예약을 대신 해주고 기업의 중개 수수료를 극대화 시키는 것이 목적일 뿐이다. 그러므로 핵심 비즈니스 프로세스에 해당하지 않는 부분은 애써 핵심 비즈니스 프로세스에 표현할 필요가 없다.

위에서 보였던 예와 같이 다른 비즈니스 프로세스들을 동적으로 통합하고자 하는 핵심 비즈니스 프로세스들은 기존의 이벤트 기반 모델링을 바탕으로 구현될 경우 프로세스 구동시의 동적 바인딩 문제를 해결할 수가 없다.

이에 대해 다음 장에서 이벤트 기반 모델이 해결하지 못하는 프로세스 간의 동적 바인딩 문제를 해결하기 위해 기존의 이벤트 기반 프로세스 모델에 상태 기반 협상 모델을 확장 구현한 시스템 아키텍쳐를 제시한다.

3. 상태 기반 협상 모델링(state-driven brokering modeling)

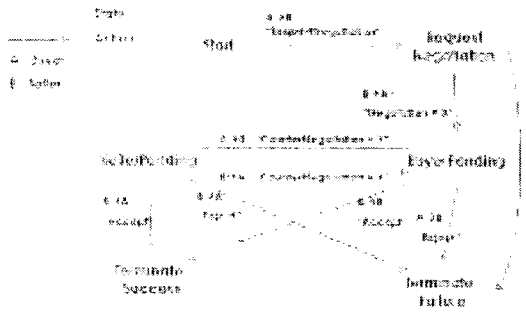
상태 기반 협상 모델링이란 사건, 즉 이벤트가 일어나는 것을 중심으로 프로세스를 구성하는 것이 아니라 상태를 중심으로 프로세스가 흘러가는 것을 기술하는 기법이다. 이는 특정 상태에서 주어진 종류의 사건을 수행할 수 있고 수행된 사건으로 인하여 다른 상태로 전환되는 것을 표현한 것으로 스테이트 머신(state machine)으로 표현할 수 있다.

상태 기반 협상 모델링은 대화 정책(conversation policy)을 이용한다. 기존의 대화 정책은 비즈니스 파트너들(고객, 기업 등) 간에 생길 수 있는 롱타임 트랜잭션(long-time transaction)을 효과적으로 처리하거나, 기업간 비즈니스 프로토콜의 합의를 자동으로 이끌어 내는 비즈니스 컴포넌트 통합의 방법으로 제시되었다.[1-2]

이 장에서는 보다 발전된 비즈니스 프로세스 간의 통합을 위한 브로커링 어댑터를 제시한다. 대화 정책을 구현한 브로커링 어댑터(brokering adaptor)는 핵심 비즈니스 프로세스에 독립적이며, 실제 수행될 불특정 다수의 대안 비즈니스 프로세스들과 핵심 비즈니스 활동 사이에서 동적 바인딩을 가능하게 한다.

3.1 대화 정책(conversation policy)

본 논문에서는 동적으로 비즈니스 프로세스를 중재(brokering)하기 위한 방법으로 대화 정책을 이용한다. 대화 정책이란 소프트웨어 커뮤니티에서 자동화된 머신-to-머신(machine-to-machine) 대화를 가능하게 하기 위해서 미리 정의된 패턴을 상태 기반 모델링 기법(state-driven modeling)을 이용하여 기술해 놓은 것이



다.[2] 그림 3은 간단한 대화 정책의 예이다.

그림 3. 구매 협상을 위한 대화 정책

그림 3의 예는 물건을 사고자 하는 파트너(A)와 물건을 팔고자 하는 파트너(B) 간의 대화 정책을 나타내고 있다. 위의 그림에서 보는 바와 같이 대화 정책은 1:1로 주어져 있는데, 자유식 대화(free dialogue)가 아닌 정형화된 대화(formal dialogue) 이다. 이와 같이 두 상대가 대화를 하는 과정에서 특정 액션(action)을 취하면 둘 간의 상태는 이 액션에 연결된 상태(state)로 변화하고, 다음에는 변화한 상태에서 취할 수 있는 액션을 수행한다. 위의 예는 이와 같은 방법으로 두 파트너의 대화가 종료 상태에 이를 때까지 협상을 하는 시나리오를 취하고 있다. 이처럼 특정 상태에 따른 액션들이 지정되어 있으므로 실제로 주고 받는 메시지의 포맷만 서로 동의하고 있다면 개별 비즈니스 파트너는 대화 정책을 수행할 수 있다. 이때 자동으로 대화를 수행하기 위해서는 교환하는 메시지와 상태에 따른 액션을 정의하는 비즈니스 로직을 대화 정책에 삽입하면 된다.

이러한 대화 정책은 XML기반의 cpXML(conversation policy XML)[8]로 표현할 수 있다. cpXML은 두 비즈니스 파트너 간의 정형화된 대화, 즉 상태와 상태에 따른 액션들을 표현할 뿐만 아니라, 애플리케이션 레벨에서 다루어 질 수 있는 비즈니스 파트너들 간에 주고 받는 메시지 포맷을 지정한다.

3.2 상태 기반 협상 모델이 확장 구현된 아키텍처

대화 정책이 비즈니스 프로세스들을 동적으로 연결해 주는 역할을 수행하기 위해 사용되려면 비즈니스 활동과 이를 실제로 수행하기 위해서 바인딩될 가능성이 있는 대안 비즈니스 프로세스들이 불특정 다수 존재한다는 가정을 한다. 이러한 가정은 외부의 웹기반 서비스를 이용하여 비즈니스 프로세스를 만들 시에 실제로 빈번히 발생할 수 있는 예라고 할 수 있다.

비즈니스 상황에 따라서 여러 종류의 스테이트 머신이 이용될 수도 있겠지만, 여행 예약 대행 비즈니스 프로세스의 활동과 대안 비즈니스 프로세스의 관계를 옥션 모델(auction model)이라고 가정하자. 그러면 비즈니스 활동과 이를 실제로 수행하기 위한 대안 비즈니스 프로세스들은 1:N의 관계를 가지게 된다. 이러한 상황에서는 다자간의 협상을 통해 하나의 비즈니스 프로세스를 최종적으로 선택할 수 있음을 표현하여야 하므로 그림 4와 같은 다자간 대화 정책(multi-party conversation policy)이 표현될 수 있다.

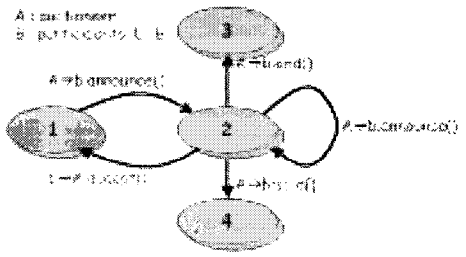


그림 4. 다자간 대화 정책

그림 4와 같은 옥션 모델의 대화 정책[3]을 XML로 표현한 cpXML은 기본적으로 비즈니스 참가자들이 개별적으로 가지고 있다. 우선 경매를 시작하는 시점에 경매 주관자는 자신과 거래를 하겠다는 파트너에게 경매가 시작되었음을 알린다.(1: A→b: announce, state 2) [이때 주관자 A는 거래할 비즈니스 파트너들이 A가 보내는 메시지를 받을 메시지 서버의 주소를 알고 있다고 가정한다.] 메시지를 받은 각 기업들은 경매의 조건을 자신의 비즈니스 로직에 맞추어 보고 A에게 경매 가격에 대한 메시지를 보내거나 보내지 않을 수 있다.(b→A:accept, state 1 or 2) 다음으로 A는 가장 큰 가격을 제시한 b의 제시 가격을 참가 파트너들에게 공지한다.(A→b:announce, state 2) 이와 같은 과정을 반복하다가 b가 얼마시간 동안 경매 가격을 제시하지 않으면 가장 최근에 경매 가격을 제시한 b에게 물건을 팔고(A→b:sold, state 4) 다른 기업들에게는 경매가 끝났음을 공지 한다.(A→b:end, state 3)

위와 같은 다자간의 대화 프로토콜을 이용할 경우 핵심 비즈니스 프로세스와 대안 비즈니스 프로세스들 마다 메시지를 주고 받은 결과에 따른 비즈니스 로직을 미리 삽입해 놓으면 자동화된 협상을 할 수 있게 된다. 뿐만 아니라 핵심 비즈니스 프로세스 관점에서 보더라도 최고의 솔루션(solution)을 가지는 웹기반 서비스를 이용할 수 있다는 장점을 가진다.

이러한 관점에서 비즈니스 프로세스를 모델링하고 구동하는 아키텍쳐에 실제 비즈니스 활동 수행을 위한 대안 비즈니스 프로세스들의 동적인 연동을 가능하게 하기 위한 협상 모델(brokering model)은 그림 5와 같은 아키텍쳐를 가진다.

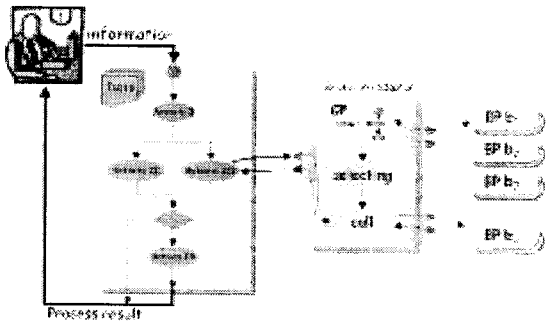


그림 5. 협상 모델의 아키텍쳐

그림 5와 같이 상태 기반 비즈니스 프로세스와 이들의 하위 활동을 실제로 수행하는 대안 비즈니스 프로세스들을 동적으로 이어주는 브로커링 어댑터(brokering adaptor)가 존재한다. 핵심 비즈니스의 활동을 수행하기 위해서, 우선 비즈니스 활동은 브로커링 어댑터에 서비스 메시지를 보낸다. 그리고 이 메시지를 받은 브로커링 어댑터는 주어진 활동을 수행하기 위해서 불특정 다수의 비즈니스 프로세스들과 협상을 하게 된다. 이때 협상을 하는 과정은 앞에서 말했던 미리 정의된 상태 기반의 협상 모델들을 이용한다. 즉, 앞에서 보여준 다자간 대화 정책에서 브로커링 어댑터는 경매 주관자의 역할을 수행하고 개별 대안 비즈니스 프로세스들은 경매 참가자의 역할을 수행하는 셈이다.

이와 같이 다자간 협상을 위해 각 개별 비즈니스 프로세스들은 대화 정책에 자신의 비즈니스 로직을 구현해 놓는다. 이 과정을 거치면 자동적으로 다자간 협상을 수행할 수 있게 된다.

협상이 끝나게 되면 최종적으로 비즈니스 활동을 수행할 개별 비즈니스 프로세스를 알 수 있게 된다. 이때 브로커링 어댑터는 실제로 이용되는 개별 비즈니스 프로세스에게 메시지를 보내어 서비스를 수행한다. 그리고 되돌려 받은 서비스 수행 결과를 다시 핵심 비즈니스 프로세스로 보내 준다.

이러한 아키텍쳐에서는 핵심 비즈니스 프로세스가 실제로 이용되어야 할 대안 비즈니스 프로세스가 어떤 것인지에 대해서 고려할 필요가 없어진다. 왜냐하면 핵심 비즈니스 프로세스가 단지 메시지를 보내어줄 곳은 브로커링 어댑터라는 정적인(static) 곳이기 때문이다. 그러므로 기업은 경쟁력의 중심인 핵심 비즈니스 프로세스를 상위 레벨에서 유연하게 만들 수가 있다.

4. 구현

BEA systems에서 개발된 웹서비스 개발 도구인 weblogic workshop을 이용하여 핵심 비즈니스 프로세스와 개별 웹기반 서비스들을 만들었다. 이 때에 사용된 프로그래밍 언어는 BPEL 시나리오(BPEL Scenario™)[11]라는 BPEL 구현 언어이다. 그리고 실제 비즈니스 활동을 수행하기 위해서 브로커링 어댑터(brokering adaptor)와 실제 이용되는 웹서비스들을 각각 구현하고 개별 웹서비스들은 핵심 비즈니스 프로세스에 참가할 수 있도록 자신의 웹서비스 인터페이스(interface)와 메시지 서버의 정보를 브로커링 어댑터에 등록할 수 있도록 한다. 그림 6과 7은 실제 수행을 위한 핵심 비즈니스 프로세스와 브로커링 어댑터의 웹서비스 인터페이스를 보여준다.

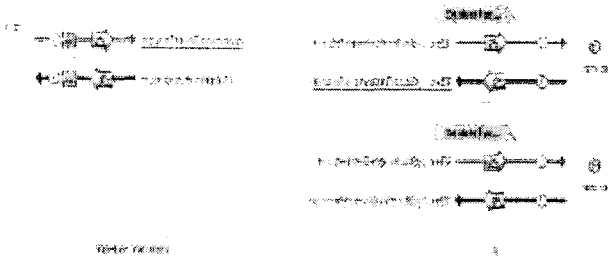


그림 6. 핵심 비즈니스 프로세스와 브로커링 어댑터 간의 웹서비스 인터페이스

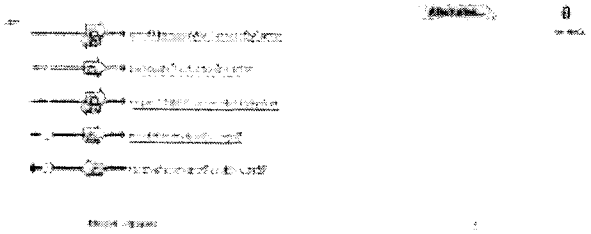


그림 7. 브로커링 어댑터 웹서비스 인터페이스

5. 결론 및 토의

본 논문에서는 이벤트 기반 비즈니스 프로세스 모델이 실제로 수행될 때, 비즈니스 활동과 실제 사용될 웹기반 서비스들의 바인딩을 위해 대화 정책을 구현한 브로커링 어댑터를 제시하였다

본 논문에서 제시한 아키텍쳐의 협상모델 구현을 위해서 기본적으로 다음과 같은 가정을 하였는데 이들은 웹기반 서비스를 이용한 동적 비즈니스 프로세스 모델링을 위해서 차후에 반드시 해결되어야 할 과제이다.

① **비즈니스 간의 온톨로지(ontology)에 대한 사전 합의와 서비스 인터페이스 변환(converting) 문제 배제**: 통합되고자 하는 비즈니스 프로세스 사이에서 서로 다른 온톨로지를 참조하고 있다면 비즈니스 협업 수행은 불가능 하다. 그러므로 협업을 수행하고자 하는 비즈니스 파트너들 간에는 같은 도메인 온톨로지(domain ontology)를 이용하고 있다고 가정 하였다. 온톨로지의 통일이 이루어지면 비즈니스 프로세스들 간에 주고 받을 메시지에 대한 웹서비스 인터페이스의 변환(converting)이 가능해 진다. 자유롭게 특정 프로세스들과의 통합을 원할 경우에는 메시지를 주고 받기 위한 인터페이스가 규정되어 있어야 하는데, 이를 위해서 각 대안 비즈니스 프로세스 앞에 인터페이스 변환기(converter)를 두면 될 것이다. 각 개별 웹기반 서비스 측에 인터페이스 변환기를 두는 이유는 거래에 있어서 진출과 탈퇴가 핵심 비즈니스 프로세스보다 자유롭다고 볼 수 있기 때문이다. 그러므로 서비스를 제공하고자 하는 비즈니스 파트너는 웹서비스 인터페이스를 변환하여 내부 프로시져와 연결할 수 있는 모듈을 구현해야 할 것이다. 이와 같은 인터페이스 변환을 위해서 DAML+OIL[10] 같은 온톨로지 마크업 언어가 사용될 수 있으나 온톨로지 마크업이라는 것이 워낙 복잡할 뿐만 아니라 머신 레벨에서의 자동적인 인터페이스 변환은 현재의 기술로는 매우 힘들다. 가능하다고 하더라도 오류의 위험이 다분히 있을 것이다. 그러므로 사람이 중간에 개입하여 서비스 인터페이스를 변환하는 과정이 필요하다. 이때에 브로커링 어댑터가 보낼 웹서비스 메소드 시그너쳐(method signature)와 메시지 파라미터(message parameter)로 이용될 수 있는 메시지의 데이터 각각에 대해서 SI(subject indicator)[12]를 지정하면 보다 정확한 변환을 할 수 있을 것이다.

② **서비스 질(Quality of Service : QoS) 문제의 배제** : 실제로 서비스를 수행해 주는 비즈니스 프로세스의 서비스 질(service quality)을 알 수가 없기 때문에 단순히 비즈니스 로직에 의해서 선택된 서비스를 이용하는 것에 대한 신뢰성의 문제 또한 연구되어야 할 것이다.

③ **웹서비스 결과 조합의 한계점 배제** : 핵심 비즈니스 프로세스를 위한 서비스들을 정상적으로 이용한다 하더라도 수행된 웹서비스 결과들이 조합되었을 경우에는 최적의 결과가 되지 않을 수 있다. 예를 들어 비행기 예약을 하고 탑승하기 위한 공항은 시카고이고 렌트카를 빌리기 위한 곳은 LA이면 개별 비즈니스 활동은 성공적으로 수행이 되었을 지 모르지만 예약을 맡긴 고객의 입장은 난감해 질 수 있다. 개별 비즈니스 활동들의 최적화가 아닌 이들이 조합되었을 때에 전체 프로세스의 최적 결과가 나올 수 있도록 하는 연구가 필요하다.

본 논문에서 제시한 브로커링 어댑터는 핵심 비즈니스 프로세스에 독립적이며, 대안 비즈니스 프로세스과 대화 정책을 수행하여 비즈

니스 프로세스를 동적으로 구동할 수 있게 한다.

이러한 아키텍쳐를 이용하면 우선 핵심 비즈니스 프로세스 개발을 유연하게 할 수 있다. 즉, 브로커링 어댑터가 개별 웹기반 서비스와의 동적 바인딩을 해결해 주므로 프로세스 디자이너는 하이 레벨(abstract level)의 비즈니스 프로세스 모델링을 할 수 있게 된다. 또한 비즈니스 파트너의 참가/탈퇴에 따라 핵심 비즈니스 프로세스의 내용을 수정해야 하는 기존의 이벤트 기반 프로세스 모델을 보완할 수 있다.

본 논문에서 제시한 아키텍쳐는 다양한 웹기반 서비스들이 자유롭게 임의의 비즈니스 프로세스에 참가할 수 있는 기반을 마련했다는 의의를 가진다. 이벤트 기반의 프로세스 모델링에서는 프로세스 내부에 명시된 웹기반 서비스들만 연결되어 사용될 수 있었는데 반해, 확장된 아키텍쳐에서는 개별 웹기반 서비스를 제공하는 기업에서 자신이 서비스를 제공해 줄 수 있는 비즈니스 프로세스 활동을 발견했을 경우 브로커링 어댑터에 자신의 정보를 등록하고 브로커링 어댑터와의 대화 정책을 위한 비즈니스 로직만 삽입하면 된다. 그러면 차후에 브로커링 어댑터와의 대화 정책을 통하여 자동적인 거래를 할 수 있는 자발적 비즈니스 협업(pro-active business collaboration)이 가능해 질 것이다

6. 참고 문헌

1. Conversation Support for Business Process Integration, James E. Hanson, Prabir Nandi, and Santhosh Kumaran, *The 6th International Enterprise Distributed Object Computing* (EDOC'02 - Sep 17-20, 2002, Ecole Polytechnic, Switzerland)
2. Conversation-enabled Web Services for Agents and e-business, James E. Hanson, Prabir Nandi, and David W. Levine, *Proceedings of the International Conference on Internet Computing (IC-02)*, CSREA Press, 2002, pp.791-796.
3. Protocols and Intentional Specifications of Multi-Party Agent Conversations for Brokerage and Auctions, V. Pitt, F. Guerin and C. Stergiou, *Proceedings Autonomous Agents 2000*
4. Business Process Execution Language for Web Services Specification. Version 1.0, *http://www-106.ibm.com/developerworks/library/ws-bpel/*, 31 July 2002
5. Web Services Coordination Specification, *http://www-106.ibm.com/developerworks/library/ws-coor/*, 9 August 2002
6. Web Services Transaction Specification, *http://www-106.ibm.com/developerworks/library/ws-transpec/*, 9 August 2002
7. Dynamic e-Business Using BPEL4WS, WS-Coordination, WS-Transaction, and Conversation Support for Web Services, Santhosh Kumaran, Prabir Nandi. IBM T. J. Watson Research Center, *http://www.research.ibm.com/convsupport/papers/BPEL%20&%20Conversations.htm*
8. cpXML, e-Commerce and Autonomic Computing Department. IBM T.J. Watson Research Center, *http://www.research.ibm.com/convsupport/papers/cpXML-v1.htm*
9. Web Services, *http://www.w3.org/2002/ws/*
10. DAML + OIL, *http://www.daml.org/2001/03/daml+oil-index.html*
11. BPEL Scenario™, Collaxa Inc., *http://www.collaxa.com/devpack.samples.WorkshopFlow.html*
12. XML Topic Maps(XTM) 1.0, TopicMaps.Org, *http://www.topicmaps.org/xtm/1.0/*, 6 August 2001
13. XLANG, Microsoft Corporation, *http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm*
14. WSFL Version 1.0, IBM, *http://www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf*
15. WSDL Version 1.2, *http://www.w3.org/TR/wsdl12/*