XML기반의 기업간 비즈니스 프로세스 모델링 기술

김 형 도

요 약

전통적으로 기업간 전자거래에서는 비즈니스 문서만을 표준화하여 거래를 실행하는 것이 일반적이지만, 비즈니스 프로세스에 대한 충분한 고려가 선행되지 않을 경우, 문서구조가 비효율적이고, 동적인 비즈니스 환경에서 재활용되기 어렵다. 따라서, ebXML이나 웹 서비스와 같이 새롭게 부상하고 있는 전자상거래 프레임워크에서는 기업간 비즈니스 프로세스를 분석하여 명세를 작성하고, 이것을 기본으로 거래를 실행하여, 재활용을 향상시키는 방안을 제시하고 있다. 그러나, 비즈니스 프로세스를 명세하기 위한 언어들은 아직까지 충분히 표준화가 진전되지 못하고 있으며, 기본적인 가정이나 표현 능력에서 큰 차이를 보이고 있다. 이 논문에서는 기업간 비즈니스 프로세스를 모델링하기 위한 XML기반 명세 언어들을 비교분석해 보면서, 이와 관련된 주요 이슈와 바람직한 대응방안에 대하여 논한다.

Key word *: XML, Business process Modeling, Electronic Commerce., ebXML, Web Service*

1. 서론

어떤 기업이 거래 기업과의 상호작용을 편리하게 하기 위하여 공유된 역할(Role), 관계(Relationship), 의무사항(Responsibility) 등을 어떻게 수행할 것인지를 상세히 정의한 것이 기업간 거래에서의 업무 프로세스다. 전통적인 기업간 전자상거래에서 많이 사용된 EDI는 비즈니스 프로세스를 고려하지 않고 단순히 문서만을 표준화하여 사용하였기 때문에 문서를 합리화하고 단순화하는 것이 불가능하였다. 최근에는 xCBL, UBL, OAGIS BOD 등과 같이 XML을 기반으로 하는 문서 표준화 활동도 활발한데, 재활용의 수준을 문서 수준에서 시나리오 (비즈니스 프로세스) 수준으로 확대하여, 시나리오를 정의하는 과정에서 문서에 대한 검토가 자연스럽게 이루어질 수 있도록 지원하며, 시나리오 수준에서 기업간 거래를 실행하고 관리하는 것이 중요하다.

ebXML에서는 비즈니스 프로세스를 모델링 하여 거래 시나리오를 작성하고, 이 시나리오에 따라서 기업간 업무를 자동화하여 실행한다. 시나리오와 문서를 정의하기 위해서는 체계적인 분석이 필요한데, 이러한 분석 작업을 지원하기 위해서 수 십 년에 걸친 EDI분야의 경험과 지식이 반영된 UMM(UN/CEFACT Modeling Methodology)의 사용이 권장된다. ebXML을 사용하여 거래하고자 하는 기업은 자신의 거래 시나리오를 모델링 하여 등록저장소에 등록하고, 거래상대방과 이 시나리오를 사용하여 거래를 수행할 수 있다.

필요에 따라서는 자신의 상황에 적합한 시나리오를 등록저장소로부터 선택하여 그대로 활용하거나 일부분을 변경하여 거래를 수행할 수도 있다. 주문관리 등과 같이 공통적으로 사용될 수 있는 거래를 비즈니스 프로세스로 모델링 하여 재활용하는 것도 매우 중요하다. 국가적인 차원이나 특정한 산업분야 차원에서 기업간 시나리오들을 체계적으로 정의하여 관리할 수 있다면 거래를 수립하기 위한 시간과 비용을 절감하는 효과를 거둘 수 있게 될 것이다. XML로 명확히 정의된 시나리오에 따라서 기업간 거래를 실행하고 관리하는 기능이 지원될 수 있기 때문에, 비즈니스 시나리오를 프로그램 코드에 구현하여 수행하는 기존의 B2B 시스템들 (EDI시스템, XML/EDI시스템 등등) 이 변화에 유연하게 대응하기 어려웠던 점을 극복할 수 있다.

웹 서비스는 표준 웹 프로토콜들을 사용하여 접근 가능한 응용 프로그램 구성요소로 정의될 수 있으며, XML을 사용한다는 특성에 중점을 두어 XML 웹 서비스라고 표현하기도 한다. 최근 웹 서비스를 단순한 것과 복잡한 것으로 구분하는 경우가 많아지고 있다. 단순한 웹 서비스란 RPC형식의 통신과 같이 동기식으로 작동되면서, 컴포넌트들이 강력히 결합되어 사용되며, 작동 상태가 관리되지 않는 경우를 지칭한다. 복잡한 웹 서비스란 메시지 기반의 웹 서비스로도 불리는데, 컴포넌트들이 약하게 결합되어 사용되며, 비동기 방식으로 메시지를 교환하는 경우를 의미한다. 복잡한 웹 서비스의 경우 SOAP, UDDI, WSDL 이외에도 기업간의 협력을 안정적인 수준에서 제공하기 위한 다양한 요소들이 고려되고 있다. 특히, 산업 분야, 기업 규모, 또는 지역/국가에 무관하게 XML을 사용하여 업무 데이터를 교환하기 위한 통합 프레임워크인 ebXML과는 표준화 내용이나 목표 면에서 많은 공통점을 가지고 있음이 지적되고 있다[김형도, 2002].

웹 서비스에서의 비즈니스 프로세스는 다수의 웹 서비스를 연결하여 복합적인 서비스를 제공하기 위한 시나리오를 의미하는데, 이러한 시나리오를 정의하기 위한 언어로는 WSFL [IBM, 2001], WSCL [HP, 2002], XLANG [Microsoft, 2001], WSCI [BEA, Intalio, SAP, Sun, 2002], BPEL4WS [BEA, IBM, Microsoft, 2002] 등 여러 가지가 제안되어 혼란을 가중시키고 있다. 최근에 제안된 WSCI와 BPEL4WS가 웹 서비스에서의 비즈니스 프로세스 명세의 특성을 잘 보여주고 있다. BPEL4WS는 IBM의 WSFL과 MS의 XLANG을 결합하여 확장한 것으로, 웹 서비스를 강력하게 추진하고 있는 IBM과 MS가 참여하고 있기 때문에 웹 서비스에서의 비즈니스 프로세스 명세 표준하의 희망을 높여주고 있다. WSCI 는 Intalio의 BPML(Business Process Modeling Language) [Intalio, 2002]을 기반으로 하고 있으며, 웹 서비스의 또 다른 축인 Sun이 참여하고 있다. 일반적으로 이러한 언어들은 웹 서비스의 정적인 인터페이스를 기술하기 위한 WSDL과 함께 동적인 인터페이스를 정의하는 역할을 한다.

이 논문은 웹 서비스에서의 비즈니스프로세스 표준화 노력을 ebXML의 BPSS와 비교분석해 보면서 이와 관련된 주요 이슈와 바람직한 대응방안을 논한다. 이 논문은 다음과 같이 구성되어 있다. 먼저 2장에서는 ebXML에서 비즈니스 프로세스를 명세하기 위하여 제정된 스키마 언어 (BPSS)의 주된 개념을 정리하고, 3장에서는 웹 서비스에서의 대표적인 WSCI와 BPEL4WS의 특징을 사례와 함께 소개한다. 4장에서는 의미와 규칙 등 메타 데이터를 이용한 비즈니스 프로세스 표현 방법에 대하여 사례와 함께 검토한다. 마지막으로, 5장에서는 XML을 이용한 비즈니스 프로세스 명세와 관련된 주요 이슈, 그리고 앞으로의 방향에 대하여 논한다.

2. ebXML BPSS (Business Process Specification Schema)

어떤 기업이 다른 거래 기업과의 상호작용을 편리하게 하기 위하여 공유된 역할 (Role), 관계 (Relationship), 의무사항(Responsibility)을 어떻게 수행할 것인지를 상세히 정의한 것이 ebXML 업무 프로세스이며, 이러한 역할간의 상호작용은 잘 설계된 일련의 업무 거래(Business Transaction)들로 구성된다. 명세 스키마는 업무 거래의 정의와 이들을 사용한 업무 협력 (Business Collaboration)을 규정하고 있다. 각각의 업무 거래는 전자 업무 문서(Electronic Business Document)의 교환으로 표현되는데, 이들의 교환 순서는 업무 프로세스에 의하여 결정된다. 다수의 표준 패턴을 사용하여 업무 거래를 구현하는 것이 필요한데, 이러한 패턴은 거래 기업간의 메시지와 업무 신호 (Business Signal)들을 교환하는 형태를 정의할 수 있다. 이러한 패턴을 정의할 수 있도록 명세 스키마에서는 여러 패턴에 공통적인 일련의 모형화 엘리먼트를 제공한다. 유의할 점은 명세 스키마에서 업무 문서의 구조를 자체적으로 정의티어 있지 않으므로, 기존의 업무 문서 표준을 사용하거나, ebXML 핵심 구성요소 규격에 포함된 업무문서 모델을 사용해야 된다는 점이다. 따라서, 거래 기업간에 합의가 된다면, EDI나 OAGIS, RosettaNet 등에서 정의된 문서들을 그대로 사용할 수 있다.

업무 거래란 두 기업가의 거래에서 가장 작은 단위 업무를 지칭한다. 하나의 업무 거래는 하나의 요청 업무 행위(Requesting Business Activity)와 하나의 응답 업무 행위(Responding Business Activity)로 구성된다. 요청 업무 행위에는 반드시 하나의 문서 흐름 (Document Flow)을 동반하게 되며, 응답 업무 행위에는 조건적으로 하나의 문서 흐름을 동반할 수 있다. 계약이나 합의와 같은 경우에는 응답 업무 행위에도 하나의 문서 흐름이 동반될 수 있지만, 일방적인 통지의 경우에는 문서 흐름 없이 응답 업무 행위가 수행된다. 유의할 점은 하나의 응답 업무 행위에 대하여 다수의 문서 흐름이 연관될 수 있으나, 실행시에는 조건을 만족하는 단 하나의 문서흐름만이 동반될 수 있다는 점이다. 추상적인 유형인 Business Action은 요청 업무 행위와 응답 업무 행위에 공통적인 속성들을 관리하기 위한 것이다.

업무 거래에서 업무 문서와 업무 신호의 가능한 조합은 [그림 1]과 같이 표현할 수 있다. 모든 업무 거래에서는 요청 역할 측의 요청 업무 행위부터 작동되어 문서흐름을 응답 역할 측으로 송신하게 된다. 하나의 업무거래를 구성하는데 있어서 이 문서흐름만이 반드시 필요한 요소이며, 나머지는 선택적으로 사용될 수 있는 것들이다. 이 문서흐름을 수신한 응답 역할 측은 수신확인 (Receipt Acknowledgment) 신호와 수락확인 (Acceptance Acknowledgment) 신호요청 역할 측에 전송할 수 있다. 이후에 응답 업무 행위를 작동시켜서 업무 처리가 모두 종료되면, 하나의 응답 문서 흐름을 요청 역할 측으로 전송할 수 있다. 만약 응답 문서흐름을 전송하는 경우에는 다시 요청 업무행위가 작동되며, 수신확인 (Receipt Acknowledgment) 신호와 수락확인(Acceptance Acknowledgment) 신호를 응답 역할 측으로 송신할 수 있다. 모든 업무 거래는 성공하거나 실패하는 것으로 종결되는데, 실패하는 경우에는 업무 거래가 시작되기 이전의 상태로 복원되어야 한다.

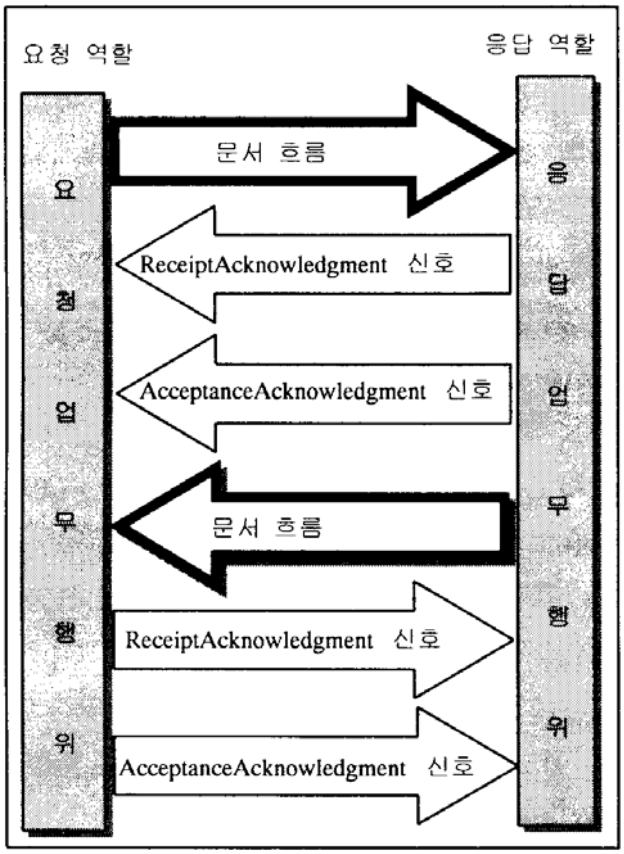

그림 1. 업무 문서와 업무 신호

확인(Acknowledgment) 업무 신호는 업무 거래의 상태를 통지하는 응용 수준의 문서를 나타낸다. 어떤 업무 신호가 발송되어야 하는가는 업무 거래에 대한 상호작용 패턴에 의하여 결정된다. 수신확인(Receipt Acknowledgment) 신호는 문서를 수신 받은 측이 문서 내용을 읽을 수 있는 시점에 발송되는 신호로서, 거래 기업간에는 이 시점에 대한 합의가 사전에 이루어져야 한다. 수락확인 (Acceptance Acknowledgment) 신호는 문서의 내용을 업무 규칙을 사용하여 검증을 받은 이후의 시점으로서 마찬가지로 거래 기업간에는 이 시점에 대한 합의가 사전에 이루어져야 한다.

[그림2]는 2개의 문서 흐름과 3개의 업무 신호로 구성된 주문생성 업무 거래를 DTD 명세 스키마를 준수하여 기술한 것이다. 요청 업무 행위의 수신 확인 시간 (timeToAcknowledge Receipt)은 2일 (P2D : Period = 2 Days), 수락 확인 시간 (timeToAcknowledge Receipt Acceptance) 는 3일로 설정되어 있어서, 수신확인과 수락확인이 모두 요구됨을 알 수 있다.

```
<BusinessTransaction name="Create Order">
    <RequestingBusinessActivity name=""
        isNonRepudiationRequired="true"
        timeToAcknowledgeReceipt="P2D"
        timeToAcknowledgeAcceptance="P3D">
        <DocumentEnvelope
          BusinessDocument="Purchase Order"/>
    </RequestingBusinessActivity>
    <RespondingBusinessActivity name=""
        isNonRepudiationRequired="true"
        timeToAcknowledgeReceipt="P5D">
        <DocumentEnvelope isPositiveResponse="true"
            BusinessDocument="PO Acknowledgement"/>
        </DocumentEnvelope>
    </RespondingBusinessActivity>
</BusinessTransaction>
```

그림 2. 업무 거래 모델 사례

요청 문서 흐름과 응답 문서 흐름은 업무 거래에 필요한 업무 문서를 포함하게 된다. 업무 문서는 문서의 유형에 따라서 구조를 달리하는 반면에, 업무 신호는 모두 동일한 구조를 갖는다. 따라서 업무 신호는 한번 정의하여 여러 번 사용할 수 있는 공통 모형화 엘리먼트로서 사용된다. 업무 흐름은 직접적으로 정의되는 대신에 두 역할 간에 전송되는 문서 봉투(Document Envelope)를 사용하여 간접적으로 정의 된다. 요청하는 업무 행위에는 반드시 하나의 문서 봉투가 수반되어야 하며, 응답하는 업무 행위에는 문서 봉투가 없어도 되나, 하나의 문서 봉투가 수반될 수도 있다. 유의할 점은 모델링 단계에서는 응답 업무 행위와 다수의 문서 봉투들을 연결하여 관계를 설정할 수 있으나, 수행 시에는 하나의 문서 봉투만을 수반할 수 있음을 유의해야 된다. 하나의 문서 봉투에는 하나의 업무 문서만이 포함되며, 이 문서와 관련된 다수의 첨부물이 동반되어 포함될 수 있다.

업무 협력이란 거래 기업간에 이루어지는 업무 거래의 집합을 가리키는 용어로서, 각각의 기업은 업무 협력에서 하나 이상의 역할을 수행하게 된다. 명세 스키마에서는 2가지 수준의 업무 협력을 지원하는데, 2개의 역할 간에 이루어지는 양자간의 업무 협력과 3개 이상의 역할 간에 이루어지는 다자간의 업무 협력이 바로 그것이다. 양자간 업무 협력은 두 역할 간에 이루어지는 업무 행위(Business Activity)들로 표현되는데, 업무 행위는 업무 협력의 상태를 반영하는 것으로 업무 거래를 수행하는 행위나 다른 업무 협력을 수행하는 복합적인 행위를 나타낸다. 양자간 협력에서 다른 양자간 협력을 수행하는 복합 행위를 표현할 수 있는 능력은 양자간 협력을 반복적으로 결합하여 재사용할 수 있도록 허용한다. [그림 3]은 업무 협력을 구성하는 요소들 간의 관계를 그림으로 보여주고 있다.

특정한 양자간 업무 협력에 포함된 업무 행위들 또는 특정한 다수 간 업무 협력에 통합된 양자간 업무 협력에 걸친 업무 행위들을 정렬시키고 순서화 시키는 일을 안무라 한다. 안무는 업무 상태(Business State)와 이들 간의 전이(Transition)를 사용하여 정의된다. 업무 행위가 업무 상태의 일종으로 표현되어 전체적인 안무가 작성되며, 보조적으로 시작(Start), 성공(Success), 실패(failure), 분기(Fork), 결합(Join) 등 다수의 업무 상태가 제공된다.

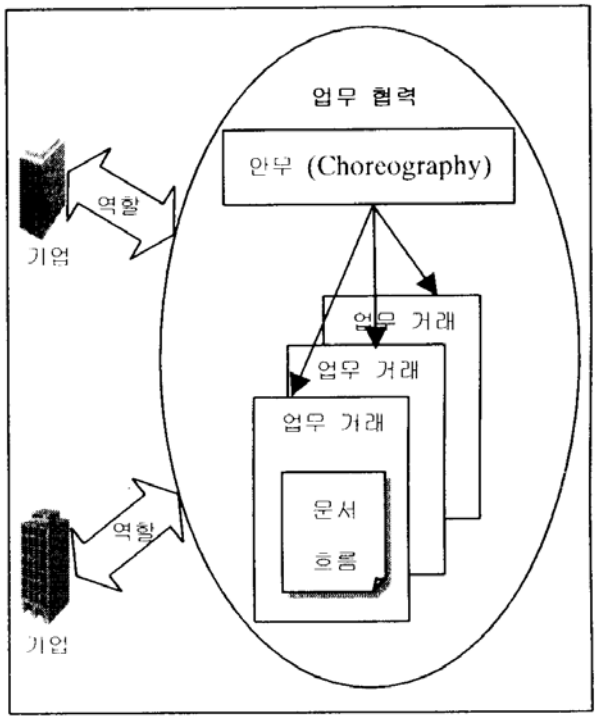

그림 3. 업무 협력의 구성요소

3. 웹 서비스와 비즈니스 프로세스

웹 서비스에서의 비즈니스 프로세스는 다수의 웹 서비스를 연결하여 복합적인 서비스를 제공하기 위한 시나리오를 의미하는데, 이러한 시나리오를 정의하기 위한 언어로는 WSFL [IBM.2001], WSCL [HP, 2002], XLANG [Microsoft, 2001], WSCI [BEA, Intalio, SAP, Sun, 2002], BPEL4WS[2002] 등이 제안되어 혼란을 가중시키고 있다. 최근에 제시된 WSCI와 BPEL4WS에 대하여 정리하고, 웹 서비스에서의 비즈니스 프로세스 표현방법에 대하여 정리한다.

3.1 WSCI

WSCI(Web Service Choreography Interface)는 BPML(Business Process Modeling Language)[2002]을 기반으로 하고 있는데, 웹 서비스의 정적인 인터페이스를 기술하기 위한WSDL과 같은 언어들과 함께 동적인 인터페이스를 정의하는 역할을 한다. WSCI에서는 웹 서비스의 동적인 인터페이스를 액티비티(Activity) 간의 상호작용으로 표현한다. 이러한 액티비티는 또 다른 액티비디로 구성될 수 있으며, 최종적으로 메시지를 송신하거나 수신하는 것과 같이 가장 기초적인 단위의 행위를 의미하는 액션 (Action) 으로 구성된다. 복합적인 액티비티에 대해서는 순차적인 수행이나 병렬적인 수행, 조건적인 수행, 반복 등과 같은 구성 액티비티 간의 상호작용을 기술할 수 있다. [그림4]와 같이 WSCI는 WSDL에서 정의된 다수의 오퍼레이션간의 복잡한 상호작용을 액션을 사용하여 기술할 수 있도록 지원한다.

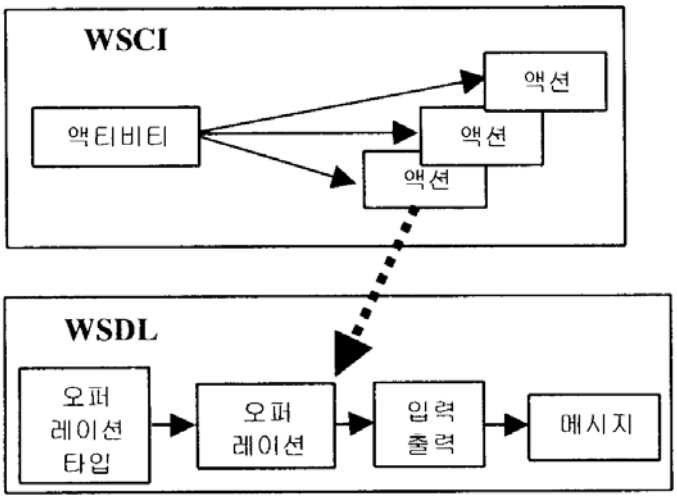

그림 4. WSCI와 WSDL의 관계

WSCI에서 프로세스란 자신의 수행문맥을 수립할 수 있어서 실행 가능한 특별한 종류의 액티비티를 의미한다. 이러한 프로세스는 새로운 메시지가 서비스에 도착하면서 실행될 수도 있으며, 다른 프로세스에 의해서 종속적으로 또는 병렬적으로 실행될 수 있다. 메시지의 특정한 부분의 값에 따라서 웹 서비스의 행위에 영향을 미칠 수 있도록 설정하는 것도 가능하다. 오퍼레이션들이 트랜잭션 방식으로 처리되는 범위, 그리고 이 트랜잭션 수행시 문제가 발생하면 실행될 보상 행위를 정의할 수도 있다. 또한 예외적인 상황이 발생할 경우에 웹 서비스가 대응하는 방법을 세부적으로 규정할 수 있도록 지원한다. 전체적인 관점에서 다수의 WSCI 모델간의 연결 관계를 동일한 메시지를 교환하는 관점에서 기술하는 것도 가능하다.

간단한 사례를 통해서 WSCI의 특성을 알아보자. 이 사례는 여행을 하고자 하는 사람(Traveler)이 여행 예약 서비스를 제공하는 특정 웹 사이트에서 에이전트(Travel Agent)에게 여행을 신청한 뒤, 이 여행신청을 확인하면, 에이전트가 이 고객에게 과금하는 시나리오를 다룬다. 이 시나리오를 정리해보면 [그림5]와 같이 TAtoTraveler 포트타입(port Type) 형태로 표현될 수 있다. 이 시나리오에 따르면 OrderTrip, bookTickets, SendStatement 등의 오퍼레이션 (operation) 이 순차적으로 실행되어야 함을 알 수 있다.

[그림6]과 [그림7]은 이러한 시나리오를 다이어그램과 WSCI로 포현한 것이다.

TravelAgent라고 하는 인터페이스는 2개의 프로세스들 (PlanAndBookTrip과 BookSeats)로 구성되는데, 첫 번째 프로세스에는 세 개의 액션들 (ReceiveTripOrder, Receive-Confirmation, SendStatement)이 순차적으로 진행되는 안무 (Choreography)가 설정되어 있다. 각각의 액션은 특정한 포트타입의 오퍼레이션이 수행됨을 의미한다. 이 프로세스는 첫 번째 액션의 오퍼레이션과 관련된 입력 메시지가 도착하면서 실행되는데, 이것은 instantiation 속성을 message로 설정하여 정의된다. 두 번째 액션에서는 BookSeats 프로세스를 호출하여 실행한다.

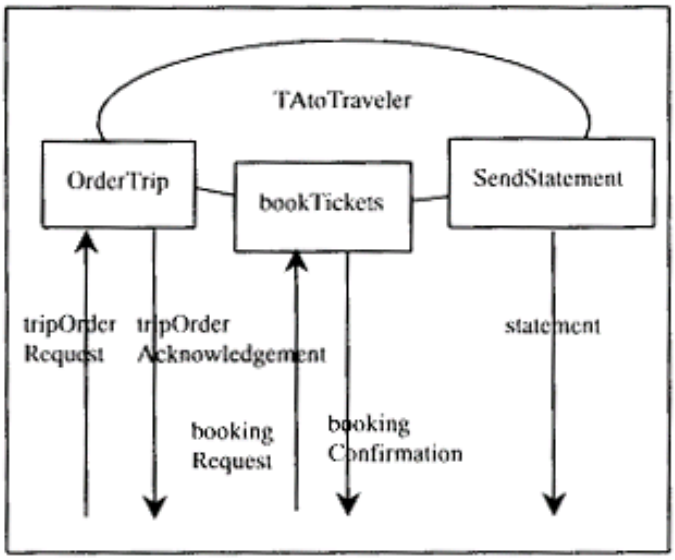

그림 5. 여행 예약 웹 서비스 시나리오

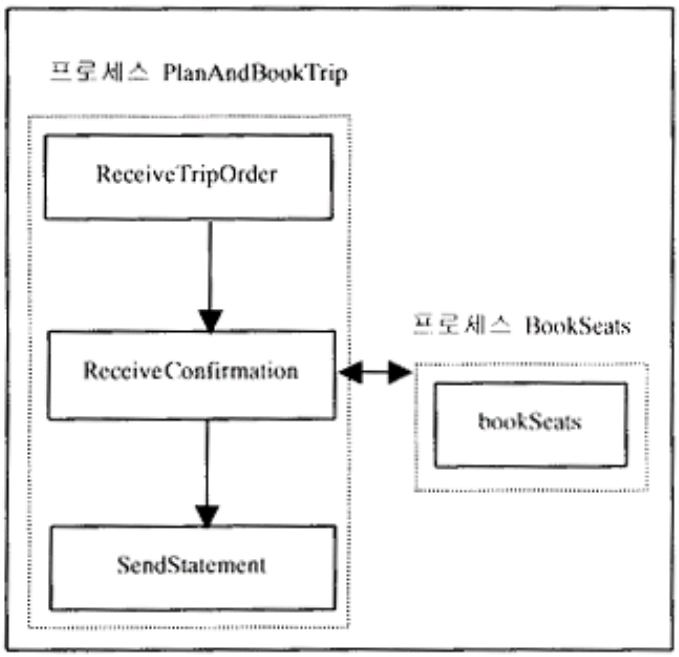

그림 6. 프로세스와 액션의 관계

[그림 7]에서 유의할 점은 correlation 을 사용하여 오랜 시간 간격을 두고 발생할 수 있는 두 액션들(ReceiveTripOrder 와 ReceiveConfirmation)을 연관시키고 있다는 것이다. bookingRequest 메시지에 포함된 itinerary ID의 값이 tripOrderRequest 메시지에 포함된 itineraryID의 값과 동일한 경우에는 두 번째 액션이 이전에 수립된 대화의 일부분에 포함됨을 의미한다.

```
<? xml version = "1.0" ?>
<definitions name = "Travel Agent Dynamic Interface"
  targetNamespace = "http://example.com/consumer/TravelAgent"
  xmlns:xsd = "http://www.w3.org/2000/10/XMLSchema"
  xmlns:tns = "http://e xample.com/consumer/TravelAgent"
  xmlns = "http://www.w3.org/TR/2002/wsci10">
  <correlation name = "itineraryCorrelation"
    property = "tns:itineraryID" />
  <interface name = "TravelAgent">
    <process name = "PlanAndBookTrip"
      instantiation = "message" >
      <sequence>
        <action name = "ReceiveTripOrder"
          role = "tns:TravelAgent"
          operation = "tns:TAtoTraveler/OrderTrip" />
        <action name = "ReceiveConfirmation"
          role = "tns:TravelAgent"
          operation = "tns:TAtoTraveler/bookTickets">
          <correlate correlation="tns:itineraryCorrelation" />
          <call process = "tns:BookSeats" />
        </action>
        <action name = "SendStatement"
          role = "tns:TravelAgent"
          operation = "tns:TAtoTraveler/SendStatement" />
      </sequence>
    </process>
    <process name = "BookSeats" instantiation = "other" >
      <action name = "bookSeats" role = "tns:TravelAgent"
        operation = "tns:TAtoAirline/bookSeats" />
    </process>
  </interface>
</definitions>
```

그림 7. WSCI로 표현된 비즈니스 프로세스

WSCI의 표현은 기본적으로 응용 시스템간의 강력한 결합을 전제로 하고 있으며, 거래에 참여하는 일방의 관점에서 정의되고 있어서, BPSS 명세와 같이 대칭적인 관점에서 정의되며 거래 참여회사 모두가 이 정의에 따라서 서비스 인터페이스를 설정하는 경우와는 거리가 멀다 [Dubray, 2002].

기업간 거래의 인터페이스를 각각 정의하고 사후에 결합하게 된다면, 여러 가지 문제점이 발견될 것은 자명한 일이다. 트랜잭션이나 예외와 같은 기능들도 정의되어 있으나, 응용 시스템간의 통합에 보다 적합하다고 할 수 있다. 기업간 거래의 의미를 규정하는 보안이나 법적인 문제 등에 대한 부문이 결여되어 있으며, 모델링에 대한 체계적인 방법론이 없다는 점도 지적 받아야 할 것이다.

3.2 BPEL4WS (Business Process Execution Language for Web Service)

BPEL4WS에서는 거래 참여자 간의 상호작용을 독립적이고 안무가 필요한 것으로 표현해야 할 필요성을 인식하고, 추상적인 프로세스로 규정되는 비즈니스 프로토콜(Business Protocol) 이라는 신조어를 사용하고 있다. 비즈니스 프로토콜이란 거래 참여자 각각의 내부적인 행위를 공개하지 않으면서 상호간의 메시지 교환 행위를 명시적으로 규정하는 것을 의미한다. 반면에 실행 가능한 비즈니스 프로세스 (Executable Business Process) 는 특정한 거래 참여자의 관점에서 실제로 발생하는 행위들을 규정한 것이다. 따라서 실행 가능한 비즈니스 프로세스는 내부적인 측면과 외부에 공개 가능한 측면으로 구문할 필요가 없다.
어떤 프로세스에 대한 정의는 하나의 액티비티(Activity)와 함께 일련의 파트너(Partner), 컨테이너(Container), 오류 핸들러(FaultHandler), 보상 핸들러 (Compensation -Handler), 그리고 연관 집합(CorrelationSet)들로 구성된다. 비즈니스 프로세스에 참여하는 참여자인 파트너는 역할 이름과 서비스 링크 유형을 정의한다. 컨테이너는 프로세스에서 사용되는 데이터를 WSDL 메시지 타입을 이용하여 규정한다. 오류 핸들러는 웹 서비스 실행시 발생되는 오류에 대응하여 수행되어야 할 행위가 정의된다. 액티비티로는 15가지 (receive, reply, invoke, assign, throw, terminate, wait, empty, sequence, switch, while, pick, flow, scope, compensate)가 있으며, 이들을 조합하여 다양한 상호작용을 표현할 수 있다. [그림 9]는 주문 메시지를 수신하고(receive 액티비티), 병렬적인 처리를 수행하며(flow 액티비티), 결과를 회신하는 (reply 액티비티) 순차적인 과정 (sequence 액티비티)를 보여주고 있다.

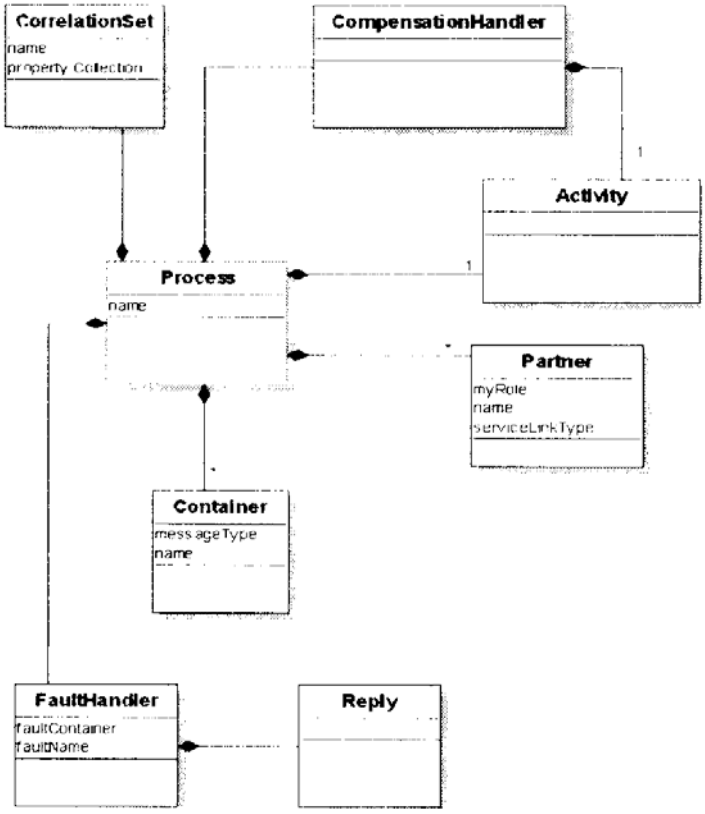

그림 8. 비즈니스 프로토콜 구성요소

비즈니스 프로세스 사례의 생성은 암묵적으로 이루어진다. 메시지를 수신할 수 있는 어떤 액티비티 (receive와 pick 액티비티)는 자신이 실행되면 새로운 비즈니스프로세스 사례를 생성하도록 사전에 설정될 수 있다. 비즈니스 프로세스 사례는 다음과 같은 3가지 방법으로 종료될 수 있다. 먼저 비즈니스 프로세스의 행위를 정의하는 액티비티가 종료되는 경우로서 이런 종류의 종료는 정상적인 것으로 간주된다. 둘째로는, 오류가 프로세스 범위에 도달한 경우인데, 비록 오류 핸들러가 있어서 오류를 조정하고 상위 수준으로 오류를 전파하지 않는다 하더라도 이런 종류의 종료는 비정상적인 것으로 간주된다. 마지막으로 비즈니스 프로세스 사례가 terminate 액티비티에 의해서 명시적으로 종료된 경우인데, 이런 종류의 종료는 모두 비정상적인 것이다. 보상(Compensation) 핸들러가 정의되어 있다면, 정상적인 종료 후에 보상이 실행된다.

```
<sequence>
   <receive partner="customer"
      portType="lns:purchaseOrderPT"
      operation="sendPurchaseOrder" container="PO">
   </receive>
   <flow>...
   </flow>
   <reply partner="customer"
      portType="lns:purchaseOrderPT"
      operation="sendPurchaseOrder" container="Invoice" >
   </reply>
</sequence>
```

그림 9. 액티비티 표현 사례

메시지의 흐름은 3가지 종류의 액티비티 (receive, reply, invoke)에 의해서 조정된다. 액티비티 간의 정보의 흐름은 암묵적인 방법으로 이루어지는데, 전역적으로 가시적인 컨테이너(Container)를 공유하여 전달된다. 응용 수준에서 오류 처리와 보상에 대한 정의가 이루어질 수 있도록 하여 트랜잭션을 지원하는데, 이것을 LRT(Long-Running Transaction) 라고 한다. 여기서 LRT라고 하는 것은 특정한 비즈니스 프로세스 안에서 지역적인 것을 의미함을 이해하는 것이 매우 중요하다. 즉, 참여자간에 사전에 합의된 결과에 대한 분산된 조정이 없다는 점이다. 응용 수준에서 보상 핸들러에 의해서 복원될 수 있는 일정한 부분은 범위로 지정될 수 있다. 오류 핸들러와 보상 핸들러와 연결된 범위들은 제약 없이 내포될 수 있다.

BPEL4WS는 2가지 방법으로 확장될 수 있는데, 하나는 임의의 BPEL4WS 엘리먼트에 이름공간을 갖는 속성을 추가하는 방법이며, 다른 하나는 임의의 BPEL4WS 엘리먼트 내부에 이름공간을 갖는 엘리먼트를 추가하는 것이다.

BPEL4WS에서는 협력이라는 개념의 중요성을 인식하고, 이것을 참여자간의 직접적인 연결관계 (즉, P2P)로서 표현한다. RPC 형태의 메커니즘을 여전히 중요하게 생각하고 있으나, 비동기 방식의 메시지 교환도 고려되고 있다. 거래 참여자간의 직접적인 관계를 정의하기 위해서 서비스 링크라고 하는 개념이 사용되는데, 실제로 사용되는 서비스는 동적으로 프로

세스 내부에서 결정된다.

서비스 링크는 [그림10]1과 같이 표현되는데, 포트유형 수준에서 협력이 표현되기 때문에 기능이 미약하며 보다 구체적인 연결은 serviceReference 엘리먼트를 사용해서 이루어진다. 협력에 관한 안무는 추상적인 프로세스로서 정의된다.

```
<serviceLinkType name="BuyerSellerLink"
   xmlns="http://schemas.xmlsoap.org /ws/2002/07/service-link/">
   <role name="Buyer">
      <portType name="buy:BuyerPortType"/>
   </role>
   <role name="Seller">
      <portType name="sell:SellerPortType"/>
   </role>
</serviceLinkType>
```

그림 10. 서비스 링크 표현 사례

4. 시맨틱 웹과 비즈니스 프로세스

시맨틱 웹은 DAML+OIL과 같은 XML기반 언어를 사용하여 매우 신속하게 구체화되고 있다. 이러한 종류의 언어는 어떤 분야의 지식을 명확히 정의하기 위한 온탈로지(Ontology)의 생성을 지원한다. XML을 이용하여 질문을 받고 대답을 제공하는 웹 서비스는 웹 정보를 자동으로 처리하기 위한 밑바탕을 제공한다. 웹 서비스에 관한 표준화는 메시지 포장 및 전송 규약(SOAP), 웹 서비스 등록기 (UDDI), 웹 서비스 기술 언어 (WSDL)와 같이 아주 세부적인 것들부터 진행되고 있다. 비즈니스 프로세스와 같이 다수의 제안이 등장하여 경쟁하고 있는 분야가 있는 반면에 문서 표준화와 같이 전혀 고려되지 않고 있는 분야도 있다. 웹 서비스 기술 언어는 웹 서비스가 사용하는 메시지와 규약에 관하여 통신 수준에서 기술하는 방법을 제공한다. WVSDL이 어떻게 전달하는가에 초점이 있다면, 이 보다 높은 응용 수준에서 무엇을 왜 전달하는가 하는 방식으로 접근하는 것이 DAML-S라고 할 수 있다. 무엇을 왜 전달하는 가를 컴퓨터가 이해할 수 있도록 기술하여 웹 서비스를 발견하고, 실행하며, 새로운 서비스를 구성하는 것이 가능하며, 웹 서비스의 특성을 검증하고 실행을 감시하는 것도 쉬워진다. DAML-S는 WSDL에서 결여된 추상적이며 응용수준에 적합한 기술을 채택하여 WSDL을 보완한다. DAML+OIL기반의 웹 서비스 온탈로지 (즉, DAML-S)를 개발하는 일은 결국 웹 서비스를 컴퓨터가 해석할 수 있도록 지원하여 특정한 제약을 만족하는 웹 서비스의 발견, 발견된 서비스의 실행, 시맨틱을 이용한 상호 운용성 증대, 자동적인 선택과 구성에 의한 새로운 서비스의 생성, 서비스 특성의 확인, 복잡한 프로세스의 업무 실행의 추적 등이 가능하게 된다.

DAML-S는 DAML+OIL을 사용하여 웹 서비스를 기술하기 위한 온탈로지이며, 서비스 프로필 (Service Profile), 프로세스 모델(Process Model), 서비스 토대 (Service Grounding) 등의 3가지 측면을 가지고 있다. 서비스 프로필은 등록기나 검색 서비스에 어떤 서비스를 등록하거나 홍보하기 위한 용도로 사용되는데, 크게 3가지의 정보 (서비스와 서비스 제공자에 대한 설명, 서비스의 기능적 행위, 다수의 기능 속성들)로 구성된다.

서비스와 서비스 제공자에 대한 설명은 일반적으로 사용자가 서비스 등록기를 검색할 경우 제공되는 정보로서 serviceName, intendedPurpose, textDescription, role, requestedBy, providedBy 등을 포함한다. 다수의 기능적인 속성들로는 geographic -Radius, degreeOfQuality, serviceParamcter, communicationThru, serviceType, service -Category, qualityGuarantees, qualityRating 등이 있다. DAML-S의 서비스 표현은 UDDI와 WSDL의 표현보다 풍부하며 서비스가 무엇을 하는 것인지의 표현에 초점을 맞추고 있다. 어디서 서비스를 발견할 것인가 하는 문제에 대해서는 DAML+OIL 온탈로지의 장점과 적절한 서비스를 발견할 가능성을 높여주는 추론 능력에 의해서 개선된다.
웹 서비스가 작동하는 방법에 관한 내용은 프로세스 모델로 기술되는데, 이것은 웹 서비스의 컨트롤 구조와 데이터 흐름의 구조를 상세히 정의한 것이다. 이러한 프로세스 모델은 ProcessModel 클래스의 하위 클래스와 속성으로 포현되며, 여러 가지의 온탈로지로 구성된다. 프로세스 온탈로지는 웹 서비스를 입력 및 출력, 사전조건, 부수효과, 하위 프로세스 등으로 표현하는 내용을 다루며, 프로세스 컨트롤 온탈로지는 프로세스의 상태를 표현하기 위한 것으로 프로세스의 시작, 수행 및 종료 등의 내용을 포함한다. 이외에도 자원과 시간을 표현하기 위한 온탈로지가 있다. 프로세스 온탈로지의 가장 중요한 개념은 프로세스인데, 이것은 크게 단위 (Atomic), 단순 (Simple), 복합(Complex) 한 종류로 분류될 수 있다. 단위프로세스는 요청자가 한 단계로 직접 실행가능하며, 하위 프로세스를 가지고 있지 않은 것이다. 단위 프로세스는 반드시 서비스 토대를 제공하여 서비스 요청자가 입력 메시지를 작성하고 응답 메시지를 해석하는 일이 가능하게 해야 한다. 반면에 단순 프로세스는 직접 실행이 불가능하며 서비스토대와 연결되어 있지 않다. 단위 프로세스처럼 한 단계에 실행될 수 있는 것으로 인식되며, 단위 프로세스의 뷰를 제공하거나 복합 프로세스를 단순하게 포현하기 위해서 사용된다. 복합 프로세스는 또 다른 프로세스로 구성된 것으로 구성내용은 Sequence, Split, Choice 등과 같은 컨트롤 구조체를 이용하여 표현된다.
서비스 토대는 WSDL과 시맨틱 웹을 연결하는 가교역할을 하기 때문에 DAML-S의 성공적인 실현에 매우 중요하다. DAML-S에서 추상적인 메시지의 내용은 단위 프로세스의 입력과 출력 속성에 의해서 암묵적으로 표현되는데, 서비스 토대 부문에서 이러한 입력과 출력을 특정한 형식으로 전달할 수 있는 구제적인 메시지로 실현된다. DAML-S에서 사용하는 서비스 토대의 개념은 WSDL의 바인딩 개념과 일치한다. WSDL의 확장가능성을 이용하여 DAML-S의 단위 프로세스의 토대를 마련하는 것은 용이한 일이다. [그림11]은 DAML-S와 WSDL의 관계를 정리한 것인데, 두 언어간의 상호 보완적인 기능을 표현하고 있다.
입력과 출력을 기술하는데 사용되는 추상적인 타입들을 정의하는 부분에서 중복되는데, WSDL에서는 XML Schema를 사용하는 반면에 DAML-S에서는 DAML+OIL 클래스들을 이용한다. WSDL과 XML Schema를 사용하는 경우에는 DAML+OIL의 시맨틱을 제대로 표현할 수 없으며, 현재의 DAML-S만으로는 WSDL의 바인딩 정보를 포현할 수 없다. 따라서 WSDL에서 정의된 메시지 부분의 추상적인 타입들로 DAML+OIL 클래스를 사용하며 갓 메시지의 형식을 규정하기 위한 WSDL 바인딩 정보를 이용하는 것은 자연스럽다고 할 수 있다. [그림11]에서 보여주는 바와 같이 DAML-S의 단위 프로세스는 WSDL-S의 오퍼레이션에, 단위 프로세스의 입력 집합과 출력 집합은 각각 WSDL의 입출력 메시지에, 그리고 각각의 입력과 출력타입은 WSDL의 확장 가능한 추상적인 타입에 해당된다.

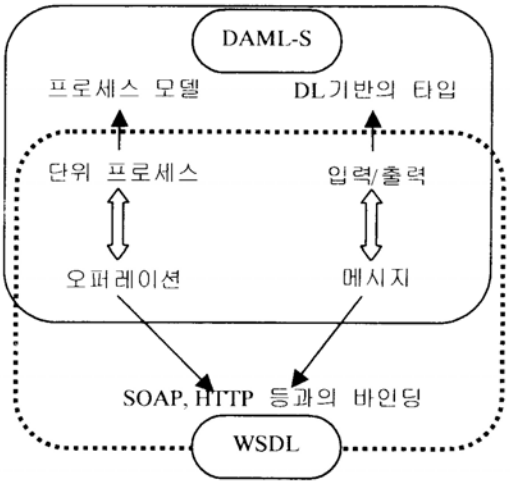

그림 11. DAML-S와 WSDL의 관계

결론적으로 DAML-S를 사용하여 비즈니스 프로세스를 표현하게 되면, DAML-S의 프로세스 모델과 DAML+OIL의 클래스표현 능력을 활용할 수 있으며, WSDL의 광범위한 결과들을 재활용 할 수 있게 된다.

5. 토의 및 결론

동적으로 변화하는 고객의 요구사항을 유연하게 반영하여, 거래 기업간의 즉흥적인 결합이 가능하도록 지원하는 인프라로서 ebXML과 웹 서비스가 경쟁하고 있다. 내용상으로 ebXML과 웹 서비스를 구분 짓는 가장 중요한 차이점은 거래에 참여하는 일방의 관점에서 거래의 특성이 정의되는 것인가 하는 여부이다. ebXML은 UN/EDIFACT의 경험을 반영하여 기업간 거래의 의미(부인방지, 법적인 문제, 시간 등)와 절차를 비즈니스 프로세스 수준에서 모델링하고, 이것을 기준으로 두 기업의 기술적 특성을 정의하며, 이러한 정의를 바탕으로 합의 안을 도출하여 거래를 실행하는 형식을 취한다.
반면에, 웹 서비스를 기술하기 위한 WSDL이나 WSDL의 오퍼레이션들을 결합하여 비즈니스 프로세스를 서술하기 위한 비표준 언어들은 서비스를 제공하는 일방의 관점에서 서술되기 때문에, 기술적인 특성에 대한 사후적인 조율이 필요한 경우가 발생하게 된다. 또한 기업간 거래의 의미에 대한 충분한 모델링이 지원되지 못하고 있음을 확인하게 된다.
웹 서비스에서의 가장 큰 단점은 비즈니스 협력을 정의하고 실행하는 과정에서 노출되는데, P2P 방식으로 작동하는 2개의 웹 서비스가 하나의 협력을 구성하고 있으며 이 협력이 실행되는 경우에 두 가지의 웹 서비스 정의가 교대로 작동된다는 점이다. 반면에 ebXML에서는 비즈니스 거래라고 하는 재사용 가능한 기본적인 메시지 교환을 정의하고 이것을 협력에서 활용하게 된다. ebXML에 일방의 관점이 아니라 중립적인 관점에서 거래와 협력을 정의하는 반면에, 웹 서비스에서는 한쪽만의 인터페이스를 정의할 수 있을 뿐이다. WSDL에 정의된 내용은 협력에 참여하는 일방이 거래에 참여할 상대방을 고려하여 작성된 것이다. ebXML과 같이 중립적인 위치에서 작성된 협력에 대한 정의가 웹 서비스에도 필요한 시점이다.

매우 복잡한 현실 세계의 기업간 프로세스에 대한 명세를 작성하기 위한 수많은 시도들은 필요한 모든 의미를 표현하는데 어려움을 겪고 있다. 최근에 제시된 WSCI나 BPEL4WS도 예외가 아니다. 무엇보다도 WSDL의 복잡함이 문제를 만들고 있다고 할 수 있는데, 이러한 복잡함을 개선하기 위해서는 두 역할간의 메시지 교환에 초점을 맞추는 작업이 필요하다. BPML을 주도하는 Intalio와 함께 Sun이 참여하는 WSCI와 가장 강력한 소프트웨어 인프라 공급자인 IBM과 MS가 지원하는 BPEL이 경쟁하는 단계로 진입하고 있기 때문에 표준하의 가능성은 높아졌다고 할 수 있다. 이러한 표준화 과정에서 ebXML BPSS와의 활발한 논의가 필요하다.

ebXML에서도 최근 비즈니스 프로세스 명세를 보완하기 위하여 동시적인 수행이나 메시지 내용에 의한 실행, 보상 및 예외 등에 대한 논의가 활발히 진행되고 있다. XML기술 또한 급속하게 발전하고 있기 때문에 XML을 기반으로 하는 ebXML과 웹 서비스도 변화를 겪게 될 것이며, 안정된 기술로서 보편화되기까지는 좀 더 시간과 노력이 필요하다. 이러한 과정에서 웹 서비스과 ebXML은 상호 보완적인 방향으로 비즈니스 프로세스의 표준화에 노력을 경주하게 될 것이다. 이러한 표준화 과정에서 시맨틱 웹을 이용한 보다 정확한 의미의 표현과 지식의 활용은 중요한 기준으로 고려되어야 할 것이다.

참 고 문 헌

[1] 김형도, "의미가 통하는 사이버 세상," e-Commerce, 2002.

[2] 김형도, "ebXML 과 웹 서비스," e-Commerce, 2002.

[3] BEA, Intalio, SAP, Sun, "Web Service Choreography Interface 1.0."
http://titan.intalio.com:8080/intalio/wsci/spec/index.html, 2002.

[4] BEA, IBM, Microsoft, "Business Process Execution Language for Web Services,"
http://dev2dev.bea.com/techtrack/BPEL4WS.jsp, 2002

[5] DAML-S Coalition, "DAML-S: Web Service Description for the Semantic Web," Proceedings of the Int'l Semantic Web Conference, 2002.

[6] Dubray, J.-J., "WSCI,"
http://www.ebpml.org/wsci.htm#Looking, 2002.

[7] Dubray, J,-J.. "BPEL4WS,"
http://www.ebpml.org/bpe14ws.htm, 2002.

[8] HP, "Web Service Conversation Language
(WSCL) 1.0," http://www.w3.org/TR/wscl10/. 2002.

[9] Intaijo. "Business Process Modeling Language (BPML) 1.0,"
http://www.bpmi.org/specifications.esp, 2002.

[10] Kim, H.D., "Conceptual Modeling and Specification Generation for B2B Business Process based on ebXML," ACM SIGMOD Record, Vol. 31, No. 1, March 2002.

[11] IBM, "Web Service Flow Language (WSFL 1.0)," http://www-ibm .com/software/solutions/webservices/pdf/WSFL.pdf. 2001.

[12] Microsoft, "XLANG: Web Services for Business Process Design,"

http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm, 2001.