BPMN2XPDL : 비즈니스 프로세스에 대한 BPMN 표기법을 XPDL 언어로의 변환

박 정 업[†] · 정 문 영[††] · 조 명 현[†††] · 김 학 수[††††] · 손 진 현[†††††]

요 약

비즈니스 프로세스 관리 측면에서 XPDL, BPML, BPEL4WS와 같은 많은 비즈니스 프로세스 실행 언어들이 각기 다른 기반과 목적에 따라 정의되었다. 이 중에 WfMC에서 제안한 XPDL은 서로 상호 작용할 수 있는 개념의 워크플로우 관련 비즈니스 프로세스 애플리케이션에서 범용적으로 이용되고 있다. 한편, 최근 BPMI에서 주도하는 BPMN(Business Process Modeling Notation)은 비즈니스 프로세스를 위한 표준화된 그래픽 표기법으로써 정의되었다. 그래서 BPMN을 지원하는 디자인 툴을 이용하면 다양한 비즈니스 프로세스를 일반화된 형태로 디자인하고 분석할 수 있다. BPMN 형식의 비즈니스 프로세스가 비즈니스 프로세스 실행 엔진에서 실행되기 위해서는 XPDL과 같은 비즈니스 프로세스 언어로 의미적으로 동일하게 변환되어야 한다. 이러한 관점에서 본 논문에서는 BPMN 형식의 비즈니스 프로세스로부터 이와 대응되는 XPDL 프로세스로의 변환 기법을 제안한다. 본 논문을 통하여 프로세스 모델링 표기법(BPMN)과 프로세스 실행언어(XPDL) 사이의 의미적 간격을 줄임으로써 현업의 프로세스 설계자와 프로세스 실행 모듈의 차이를 최소화하였다.

키워드 : BPMN, XPDL, 비즈니스 프로세스

BPMN2XPDL : Transformation from BPMN to XPDL for a business process

Jung Up Park[†] · Moon young Jung[††] · Myung Hyun Jo[†††] · Hak Soo Kim[††††] · Jin Hyun Son[†††††]

ABSTRACT

To formally describe business process, many business process languages have been so far specified with different origins and goals such as XPDL, BPML and BPEL4WS. Especially, XPDL proposed by WfMC has been widely used in various business process environments for a long time. On the other hand, the necessity of a standard graphical notation for a business process may create BPMN driven by BPMI. Because BPMN is composed of graphical constructs which can be used to graphically depict business process, BPMN-formed business processes should ultimately be converted to their corresponding semantically equivalent business process language(XPDL). Then, the business process languages can be consequently executed by business process engines. In this paper, we proposed a transformation mechanism from BPMN to XPDL for a business process. By this paper, We minimized the difference between process designers and process execution modules as reducing the gap of semantics between BPMN and XPDL.

Key Words : BPMN, XPDL, Business Process

1. 서 론

최근 HP, IBM, Microsoft와 같은 수많은 기업들은 자사의 비즈니스 프로세스를 자동화하기 위해 비즈니스 프로세스 모델링에 대한 다양한 기술들을 개발하고 있다. 비즈니스 프로세스를 자동화하는 것은 오프라인으로 진행되었던 작업들에 대한 비용 절감을 통해 생산성 향상에 대한 고품질 서비스를 제공한다는 것을 의미한다. 예를 들어, 카탈로그 요청, 주문, 주문 처리, 배송 의뢰, 발송 등으로 이루어지던 비즈니스 프로세스를 자동화함으로써, 인력 비용을 감소시킬 뿐만 아니라 고품질의 서비스를 신속하고 안전하게 처리함으로써 고객관리에 대한 효율성을 증대시킬 수 있다. 즉, 비즈니스 프로세스를 자동화한다면, 비즈니스를 제공하는 업체뿐만 아니라 비즈니스에 참여하는 고객, 파트너, 공급자에게 모두 안전하고 신뢰성을 갖춘 고품질의 서비스를 제공할 수 있게 된다.

비즈니스 프로세스를 자동화하기 위해서는 무엇보다도 다양한 비즈니스 상황에 적용될 수 있는 비즈니스 프로세스 언어를 정의해야 한다. C나 Java와 같은 프로그래밍 언어가

※ 본 연구는 대학 IT연구 센터 육성 · 지원 사업의 연구결과로 수행되었음.
† 준 회 원 : 한양대학교 컴퓨터공학과 석사과정
†† 준 회 원 : 미라콤아이앤씨
††† 준 회 원 : 코난테크놀로지
†††† 준 회 원 : 한양대학교 컴퓨터공학과 박사과정
††††† 종신회원 : 한양대학교 컴퓨터공학과 조교수
논문접수 : 2006년 3월 31일, 심사완료 : 2006년 5월 18일

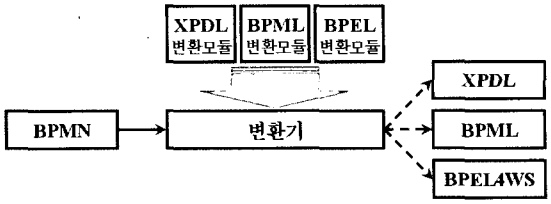
다양한 컴퓨팅 리소스를 구성할 수 있도록 정의된 것과 같은 맥락이다. 하지만 비즈니스 프로세스 언어는 운영체제나 디바이스와 같은 물리적 개체에 의존하지 않고, 비즈니스 환경이라는 논리적 의미에 적응하도록 구성되어야 한다. 특히 외부의 비즈니스 프로세스와 동적으로 연동될 수 있도록 구성되어야 한다.

위와 같은 요구를 충족시키기 위해 비즈니스 프로세스 언어에 대한 연구가 오래 전부터 진행되었다. 지금까지 비즈니스 프로세스 언어에 대한 연구는 크게 두 가지 방향으로 이루어 졌다.

첫째, 시스템이 이해할 수 있는 비즈니스 프로세스 실행 언어에 대한 연구이다. 일반적으로 비즈니스 프로세스 언어는 실행 언어를 말한다. 비즈니스 프로세스 실행 언어는 비즈니스 프로세스 시스템에서 작동 및 상호 운영을 위해 구성된다. 대표적으로 IBM, BEA, Microsoft에서 공동 개발한 BPEL4WS(Business Process Execution Language for Web Service), BPMI에서 개발한 BPML(Business Process Modeling Language), WfMC에서 개발한 XPDL(XML Process Definition Language)[2]을 예로들 수 있다. 본 논문은 기존 워크플로우 표준을 분산된 이기종의 시스템에 적용될 수 있도록 XML형태로 변환한 XPDL을 선택했다. 위 세 가지 비즈니스 프로세스는 각각 장단점을 가지고 있지만, 기존 워크플로우의 기반 개념을 정확히 표준화한 것은 XPDL이기 때문이다. 특히 워크플로우에 대한 연구는 1990년대부터 활성화되어 최근 많은 상용제품이 출시되어 사용되었기 때문에, 본 논문은 기존 기술의 재사용성을 고려하여 XPDL을 연구 방향으로 선택하였다.

둘째, 비즈니스 분석가가 이해할 수 있는 비즈니스 프로세스 모델링 표기법에 대한 연구이다. 비즈니스 프로세스 모델링 표기법은 최근 BPMN Working Group에 의해 그 의미가 더욱 부각되고 있다. 비즈니스 프로세스 모델링 표기법도 비즈니스 프로세스 실행 언어처럼 오랫동안 연구되었고, BPMN(Business Process Modeling Notation)[1], UML 액티비티 다이어그램, UML EDOC 비즈니스 프로세스, IDEF, ebXML BPSS, ADF(Activity-Decision Flow) 다이어그램, RosettaNet, LOVeM, EPCs(Event Process Chains) 등을 예로 들 수 있다. 비즈니스 분석가는 카탈로그 요청, 주문, 주문 처리, 배송 의뢰, 발송 등의 비즈니스 프로세스를 위와 같은 그래픽 표기법을 이용해 정의할 수 있다. 비즈니스 프로세스 모델링 표기법은 비즈니스 분석가가 비즈니스 프로세스의 내부적 수행을 알지 못해도 정의할 수 있도록 한다. BPMN은 다른 어떤 비즈니스 프로세스 모델링 표기법보다 많은 기능과 확장성을 갖고 있기 때문에, 본 논문의 비즈니스 프로세스 모델링 표기법으로 선택되었다.

최근 비즈니스 프로세스 자동화에 따른 다양한 성공 사례가 나타남으로써, 비즈니스 프로세스 언어의 두 흐름의 중요성은 극대화되었다. 하지만 두 가지 흐름은 각기 다른 기관에서 다른 방법론을 통해 구현되었기 때문에, 동일한 프로세스를 정의하여도 비즈니스 의미에 대한 차이가 발생한다. 비즈니스 의미의 차이는 리스크를 일으키는 위협 요소로 부각될 수 있다. 본 논문은 사용자에 적합할 수 있는 비즈니스 프로세스 모델링 표기법(BPMN)과 시스템에 적합할 수 있는 비즈니스 프로세스 실행 언어(XPDL)의 의미적 차이를 줄이기 위한 매핑 기술을 제시한다. 매핑 기술은 (그림 1)의 변환기에 적용되며, 향후 BPEL4WS, BPML의 매핑 기술도 연구가 진행될 것이다.



(그림 1) BPMN에서 프로세스 실행 언어로의 변환 구조

논문의 구성은 다음과 같다. 제 2장에서는 관련 연구로써 기존 매핑에 대한 연구를 분석하고, 기존 매핑에 대한 문제점을 제시한다. 제 3장에서는 BPMN과 XPDL에 대한 구성 요소들을 요약 기술한다. 제 4장에서는 BPMN과 XPDL 사이의 차이점을 분석하고, BPMN에서 XPDL로의 매핑에 적용될 수 있는 기법을 제시한다. 5장에서는 4장에서 기술한 매핑 기법의 구현 및 예제를 설명하고, 6장에서 결론을 맺는다.

2. 관련 연구

BPMI와 WfMC는 2002년 함께 작업을 하기로 합의하면서, WfMC는 BPMN을 XPDL의 표기법으로 사용하기로 수락하였다. 그리고 그 결과물로써 2003년 BPMN 스펙의 저자인 "Stephen A. White"에 의해 XPDL과 BPMN 간의 간단한 매핑이 이루어졌다[3]. 이 논문에서는 BPMN과 XPDL은 모두 그래프 형식의 구조여서 BPMN에서 BPML이나 BPEL4WS로의 매핑보다 간단히 매핑된다고 설명하고, BPMN의 그래픽 표기법 중 10가지에 대해서 XPDL로의 직접적인 매핑을 제시하였다. 그리고 이를 예제를 통해 분석하였다. 하지만 이는 BPMN의 모든 요소에 대해 XPDL로 매핑한 것이 아니라 매핑이 잘되는 요소만을 보여준 것이고, 아직 남은 BPMN 요소에 대한 매핑이 더 진행되어야 할 것이다.

[1]에서는 BPMN은 비즈니스의 인터페이스의 역할을 하고, 비즈니스 프로세스 실행 시스템에서의 실행을 위해서는 이에 최적화된 실행 언어로의 변환이 이루어져야 한다고 하면서, BPMN과 BPEL4WS 간의 매핑 기법을 제공하였다. 이는 각각의 BPMN 요소에 대한 매핑과 경우에 따른 분석을 진행하였고, 거의 마무리된 것으로 보인다. 이는 BPMN에서 하나의 실행 언어로 매핑한 하나의 예이며, 비즈니스 프로세스 실행 시스템에서 지원하는 언어에 따라 XPDL이나 BPML에 대해서도 이 정도 수준의 매핑이 이루어져야 할 것이다.

[4]와 [5]에서는 UML로 비즈니스 프로세스를 표현하는

방법을 제시하면서, 이의 실행을 위한 XPDL로의 변환을 보였다. [4]에서는 하나의 전체 비즈니스 프로세스는 유스케이스 다이어그램으로 나타내고, 이는 다시 시퀀스 다이어그램, 스테이트차트 다이어그램, 액티비티 다이어그램을 통해 실행 수준의 프로세스로 구체화 시킬 수 있음을 보이고, 이들을 다시 XPDL로 매핑하는 과정을 기술했다. 그리고 XPDL에 있는 상세한 속성에 대한 매핑을 위해서 스테레오타입을 정의하였다. [5]에서는 스테이트차트 다이어그램 없이 액티비티 다이어그램과 XPDL이 일대 일 매핑되도록 하였다. 다시 말해서 액티비티 다이어그램 하나에 여러 비즈니스 프로세스가 스윔레인을 기준으로 나누어지고 이들 사이의 메시지 통신은 <<Business Document>>와 같이 스테레오타입을 정의하여 사용하였다.

[6]에서는 클래스 다이어그램과 <<Process>>, <<Activity>>와 같은 스테레오타입을 이용해 프로세스와 액티비티의 애트리뷰트와 메시지를 표현하고 액티비티 다이어그램을 이용해 이들의 흐름을 표현함으로써 기업 간의 비즈니스 프로세스를 나타내었다.

[20]에서는 표준적인 모델링 도구로써 활동(Activity) 다이어그램을 XPDL로 변환하는 매핑을 기술하였다. 표기법과 XPDL의 부족한 점을 보완하기 위해 확장적인 방법을 이용하였다. 활동 다이어그램은 XPDL 요소인 '패키지'와 '액티비티 집합'을 표현하기 위해 각각 해당하는 심벌을 추가하였고, XPDL에서의 동기/비동기 서브프로세스를 표현하기 위해 프로세스 심벌을 색으로 구분하였다. 한편 XPDL은 활동 다이어그램의 '객체'를 표현하기 위해 확장 속성을 가지고 구성 요소를 확장하였다.

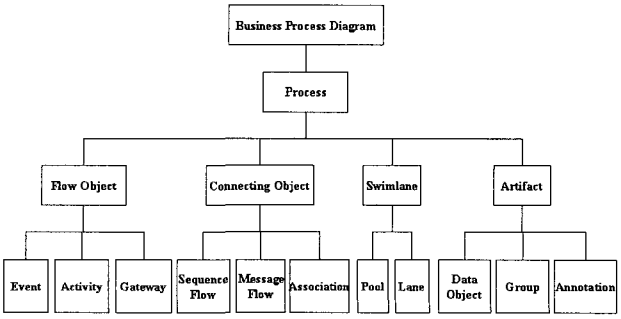
비즈니스 프로세스 표현력에 대한 연구는 프로세스 언어의 워크플로우 패턴에 따른 분석[9-11], 그래픽 표기법의 워크플로우 패턴에 대한 표현[8], 그리고 각 언어들에 대한 표현 요소별 비교[12] 등으로 진행되었다. [8-11]는 프로세스 언어가 각 워크플로우 패턴에 대해 표현할 수 있는지 여부와 어떻게 표현될 수 있는지를 보였고, [12]에서는 각 프로세스 언어가 갖고 있는 요소와 갖고 있지 않은 요소를 비교 분석하였다.

현재까지는 BPMN에서 비즈니스 프로세스 실행 언어로의 매핑보다는 UML 다이어그램에서 비즈니스 프로세스 실행 언어로의 매핑이 더 활발하게 진행되었다. 하지만 관련 연구에서 보듯이 UML을 통한 비즈니스 프로세스의 표현은 매우 다양한 방법으로 이루어질 수 있을 뿐 아니라, 어느 언어로 매핑하는지에 따라 스테레오타입을 정의해 주어야 했기 때문에 여러 언어 간에 표기법을 표준화시키기에 적합하지 않은 것으로 보인다. 본 논문에서는 여러 비즈니스 프로세스 실행 언어 간에 표준화된 표기법으로써의 BPMN으로 부터 비즈니스 프로세스 실행 언어인 XPDL로의 매핑을 진행한다.

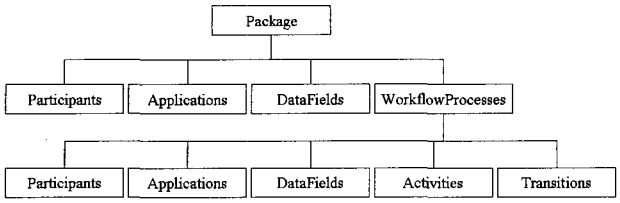
3. BPMN과 XPDL

BPMN의 구조를 살펴보면, (그림 2)에서와 같이 최상위 단위로 비즈니스 프로세스 다이어그램(Business Process Diagram)이 있고, 하나의 다이어그램은 여러 프로세스로 구성될 수 있다. 프로세스(Process)는 하나의 업무를 수행하기 위해 규칙들을 정의하고, 참여자들의 정보 전달을 정의한다. 프로세스는 크게 플로우 오브젝트(Flow Object), 연결형 오브젝트(Connecting Object), 스윔레인(Swimlane), 아티펙트(Artifact)로 나누어지며, 이들은 다시 각각 다음의 세부적인 요소로 나누어진다. 첫째, 플로우 오브젝트는 비즈니스 행동의 종류에 따라 구별된다. 프로세스 중 발생하는 사건의 행동을 나타내는 이벤트(Event), 실제적인 작업을 수행하는 액티비티(Activity), 그리고 프로세스의 흐름을 결정하는 게이트웨이 (Gateway)로 나누어진다. 또한 액티비티는 다시 태스크(Task)와 서브프로세스(SubProcess)로 나누어지는데, 서브프로세스는 다른 프로세스를 호출함으로써 주어진 작업을 다른 프로세스에 할당하며, 루프 형식에 따라 같은 행동을 여러 번 반복하거나(Standard Loop), 동시에 여러 번 행할 수 있다(MultipleInstance Loop). 둘째, 연결형 오브젝트는 순차 플로우(Sequence Flow), 메시지 플로우 (Message Flow), 어소시에이션(Association)로 나누어지는데, 순차 플로우는 프로세스 내부 개체들의 흐름을 나타내고, 메시지 플로우는 프로세스 간의 개체들의 흐름을 나타내며, 어소시에이션은 흐름과 관계없이 단지 관계성을 가진 개체들을 연결한다. 여기서 순차 플로우는 조건에 따라 게이트웨이와 함께 작동하여 액티비티의 흐름을 제어할 수 있다. 셋째, 스윔레인은 프로세스를 대표하는 참가자의 역할을 나타내는 풀(Pool)과 프로세스를 사용하는 참가자의 역할을 나타내는 레인(Lane)으로 나누어진다. 마지막으로 아티펙트는 데이터 오브젝트(DataObject), 그룹(Group), 주석(Annotation)로 나눠지는데, 이것은 실제 프로세스의 흐름에는 영향을 주지 않는 요소로서 프로세스와 연관된 메타 정보를 기술하기 위해 이용된다.

XPDL의 구조를 살펴보면, (그림 3)에서와 같이 패키지(Package)는 여러 워크플로우 프로세스를 정의하는데, 각 프로세스는 참가자(Participants), 애플리케이션(Applications), 데이터필드(DataFields), 액티비티(Activities), 트랜지션(Transitions)의 5개의 구성요소로 정의된다. 여기서 참가자, 애플리케이션, 데이터필드는 패키지 수준에서 정의되어, 여러 워크플로우 프로세스에서 공유할 수 있다. 첫째, 참가자는 프



(그림 2) BPMN의 구성 및 특징



(그림 3) XPDL의 구성 및 특징

로세스 안에서 일어나는 비즈니스 행동의 수행자를 나타낸다. 둘째, 애플리케이션은 외부 파티와 연동하거나 참가자에 의해 행동이 수행되는데 필요한 도구이다. 셋째, 데이터필드는 프로세스 안의 각 액티비티가 수행되는데 필요한 데이터로서 참여자나 애플리케이션에 필요한 정보이거나 프로세스의 흐름을 결정하는 요소이다. 넷째, 액티비티는 프로세스 안에서 작업을 수행하는 최소 단위로서, 애플리케이션과 참가자를 이용하는 애토믹(Atomic) 액티비티, 다른 프로세스를 호출하는 서브플로우(Subflow) 액티비티, 액티비티 집합(ActivitySet)을 호출하는 블록(Block) 액티비티, 그리고 프로세스 흐름의 방향을 결정하는 라우트(Route) 액티비티로 나누어지며, 다섯째 구성요소인 트랜지션을 이용해 액티비티들의 흐름을 제어한다.

4. BPMN에서 XPDL로의 매핑

비즈니스 프로세스란 하나의 비즈니스에 참여하는 파트너와 액티비티들 사이의 비즈니스 트랜잭션의 모음이라고 정의된다. 일반적으로 비즈니스 프로세스는 크게 두 부분으로 나누어진다. 기업 내의 비즈니스를 정의하는 프로세스와 기업 간의 비즈니스를 정의하는 프로세스이다. XPDL과 같은 워크플로를 정의하는 비즈니스 프로세스 언어들은 기업 내의 참여자들의 작업 흐름을 결정하기 위해 사용되었다. 하지만 기업 간의 비즈니스가 요구됨에 따라 복잡한 형태의 비즈니스가 필요하게 되었다. 그래서 WS-CDL, WSCI 등과 같은 협력(Collaboration) 언어가 출현하게 되었다. 뿐만 아니라, 기업 내와 기업 간의 모든 비즈니스 프로세스를 정의할 수 있는 BPEL4WS와 BPML과 같은 실행언어와 BPMN과 같은 그래픽 표기법도 개발되었다. 즉, BPMN과 XPDL은 비즈니스를 정의하는 범위가 다르기 때문에, 본 논문은 다음과 같은 가정을 기술한다.

가정 1. BPMN에서 XPDL로의 매핑은 기업 내에서 구성될 수 있는 비즈니스 엔터티만을 고려한다.

3장에서 기술했던바와 같이 BPMN은 비즈니스 프로세스 모델링을 위해서 설계되었고, XPDL은 비즈니스 프로세스 실행을 위해 설계되었다. 이와 같은 목적의 차이는 동일한 비즈니스 프로세스를 설계하는 데 서로 표현하는 영역을 달리할 뿐 아니라, 심지어 같은 의미의 비즈니스 엔터티를 서로 다르게 표현하는 문제를 발생시킨다. 예를 들어, 주문 프로세스를 정의할 때, BPMN은 설계의 정확성을 위해 전송되는 송장을 데이터 오브젝트를 이용해 모델링할 수 있지만, XPDL에는 이런 기능이 없다. 사실 이런 기능은 실행에 관계없는 기능이기 때문에, XPDL에는 존재할 필요 없는 기능들이다. 즉, BPMN과 XPDL은 표현하는 비즈니스의 영역이 다르다. 또, BPMN과 XPDL은 구조적으로 모두 그래프 형식을 갖고 있기 때문에, 동일한 의미의 비즈니스 객체를 많이 포함하고 있지만 다르게 표현하고 있다. 예를 들어 비즈니스 행동을 나타내는 BPMN의 태스크 액티비티와 서브프로세스 액티비티는 각각 XPDL의 애토믹 액티비티와 서브플로우 액티비티에 매핑되고, 작업 간의 흐름을 나타내는 BPMN의 순차 플로우는 XPDL의 트랜지션에 매핑된다[3]. 그래서 본 논문은 다음과 같은 가정을 기술한다.

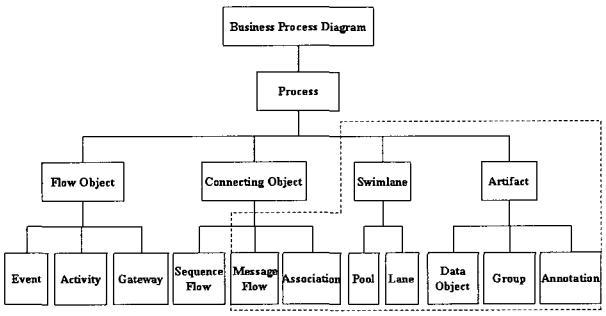
가정 2. BPMN에서 XPDL로의 매핑은 XPDL을 기반으로 한 실행에 관련된 요소들만 고려한다.

위와 같은 가정을 바탕으로 다음과 같은 방식으로 매핑을 진행한다. 4.1절에서는 BPMN의 요소 중 XPDL로 매핑이 필요 없는 요소를 구분하기위해, BPMN의 요소를 분석한다. 4.2절에서는 BPMN과 XPDL의 구조적인 비교를 통해 비즈니스 환경에 대한 프로세스 모델링 방식을 분석한다. 마지막으로, 4.3절의 심플 매핑과 4.4절의 복잡 매핑에서 BPMN에서 XPDL로의 상세 매핑 및 이행적 매핑 기법을 제안한다.

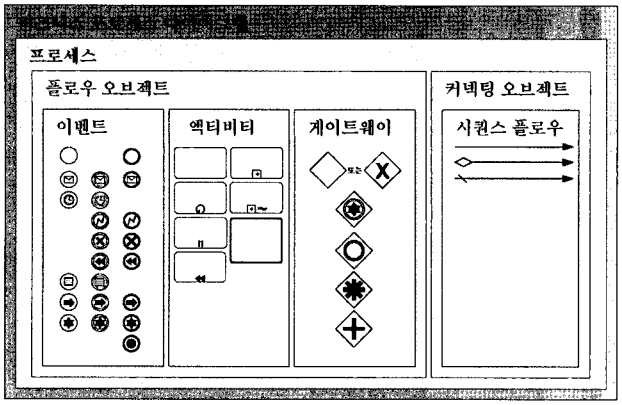
4.1 XPDL로 매핑해야 할 BPMN 요소 분석

BPMN은 비즈니스 프로세스를 그래픽 표기법으로 정의할 뿐만 아니라, 실행과 모니터링에 관련된 기능들도 부분적으로 포함하고 있다. 다시 말해서, BPMN에는 시스템을 통한 프로세스의 실행과 관련된 표현뿐만 아니라 사람에 의한 모델링, 관리, 모니터링을 위한 표현들도 포함하고 있다. 본 절에서는 가정 1과 가정 2에 의해 BPMN에서 XPDL로 매핑 할 필요가 없는 엔터티를 선택한다. (그림 4)에서 점선으로 포함된 비즈니스 엔터티들은 BPMN에서 XPDL로 매핑이 고려되지 않는 요소이다.

먼저 가정 1에 의해, BPMN의 요소 중 기업 간의 비즈니스만을 담당하는 요소를 판별한다. 연결형 오브젝트 중 메시지 플로우는 조직 간 또는 독립된 프로세스 간의 메시지를 전달하는 기능을 한다. 메시지 플로우와 같은 데이터의 흐름은 XPDL이 처리하는 비즈니스 영역을 벗어나는 요소이다. 이와 같은 제한점은 WfXML[18]이라는 비동기 실행언어에 의해 구성될 수 있기 때문에, XPDL로의 매핑에는 포



(그림 4) BPMN에서 매핑할 필요가 없는 요소



(그림 5) XPDL로 매핑이 이루어져야 하는 BPMN 요소의 매핑 세트

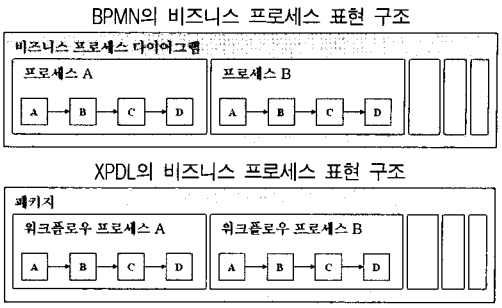
함할 필요가 없다.

가정 2에 의해 실행과 관계없는 요소를 선택한다. 실행과 관련이 없는 요소로는 어소시에이션, 스윔레인(풀, 레인), 아티펙트(데이터 오브젝트, 그룹, 주석)가 있다. 첫째, 어소시에이션은 프로세스의 실제 흐름과 관계없는 아티펙트를 연결하거나 이미 실행된 태스크에 대한 보상 관계에 있는 보상 태스크 (Compensation Task)와 연결함으로써 플로우 오브젝트들 간의 연관성만을 보여준다. 어소시에이션은 순차 플로우처럼 프로세스의 흐름을 제어하는 기능을 담고 있지 않기 때문에 실행과 관계없는 요소이다. 둘째, 스윔레인은 비즈니스 프로세스의 외부 프로세스와의 역할 관계나 프로세스 내부적인 플로우 오브젝트의 역할 관계를 나타낸다. 역할이라는 의미는 플로우 오브젝트와 관련이 있는 참가자를 의미하다. 즉, 스윔레인은 액티비티를 수행하는 특정 참가자를 지칭하기 보다는 조직들 사이 또는 조직 내부의 참가자들 사이의 역할 관계를 지정하기 위해 사용되는 요소이다. 외부 고객이나 회사원과 같은 사람이 작업을 수행하기 위해서는 사용자 태스크(User Task)라는 엔터티를 이용하면 되기 때문에, 스윔레인은 프로세스의 실행과는 관계가 없는 요소이다. 마지막으로, 아티펙트는 프로세스의 좀 더 정확한 흐름을 위해, 또는 사용자의 가독성을 위해 프로세스 관련 메타 정보를 표현하기 때문에, 실행과는 관련이 없는 요소이다.

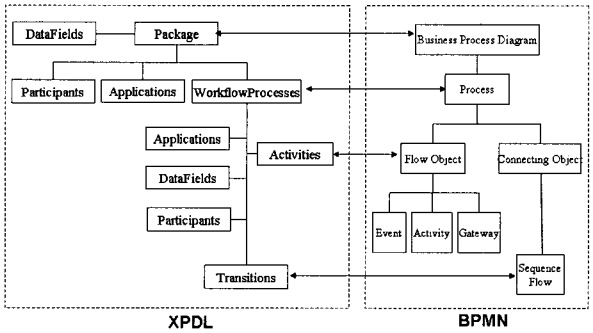
(그림 5)는 위에서 제시한 매핑할 필요가 없는 요소들을 제외한 BPMN 요소이다. 즉, 이것들은 BPMN의 요소들 중 프로세스의 실행과 관계된 요소들로서 XPDL로 반드시 매핑 되어야 한다. 다음 절부터 이것들에 상세한 매핑을 기술한다.

4.2 BPMN에서 XPDL로의 구조적 분석을 통한 매핑

프로세스란 비즈니스에서 일어나는 행동들의 흐름을 말한다. 다시 말해서 누가 어떤 일을 언제 해야 하는지, 그리고 그 일을 하기 위해서는 어떤 정보와 도구가 필요한지가 하나의 프로세스 안에 절차적으로 표현된다. 이러한 절차적 순서는 PetriNet[17]과 같은 그래프 구조를 통해 명확하게 표현할 수 있는데, BPMN과 XPDL도 그래프 구조로써 비즈니스 프로세스를 표현한다. 그래서 공통적인 구조적 특징에 따라서 BPMN과 XPDL 간에는 구조적 정의는 다르더라도 같은 의미를 나타내는 요소를 갖고 있다. 본 절에서는 이러한 요소를 중심으로 매핑을 분석한다.



(그림 6) BPMN과 XPDL의 비즈니스 프로세스 표현 구조



(그림 7) BPMN과 XPDL의 구조적 매핑

BPMN과 XPDL은 (그림 6)과 같이 비즈니스 프로세스를 거의 유사한 구조로 표현한다. BPMN은 비즈니스 프로세스 다이어그램 안에 여러 프로세스를 정의하고, 각 프로세스 안에 비즈니스 행동을 나타내는 여러 플로우 오브젝트들이 연결형 오브젝트에 따라 흐름을 형성하는 구조이다. 유사하게, XPDL은 패키지 안에 여러 워크플로우 프로세스를 정의하고 각 워크플로우 프로세스 안에 비즈니스 행동을 나타내는 여러 액티비티들이 트랜지션에 따라 흐름을 형성하는 구조이다. (그림 7)은 BPMN과 XPDL의 구조를 형성하는 각 엔터티들을 연결한 그림이다. (그림 7)의 화살표는 BPMN과 XPDL에서 구조적으로 동일한 의미의 비즈니스 엔터티를 연결하고 있다.

BPMN의 비즈니스 프로세스 다이어그램은 여러 프로세스를 관리하는 컨테이너이다. 이것은 프로세스 설계자에 의해 조작되기 때문에, 설계자에 필요한 정보나 프로세스가 설계된 날짜와 같은 메타 정보를 기록한다. (그림 7)과 같이 BPMN의 비즈니스 프로세스 다이어그램은 XPDL의 패키지와 매핑되는데, 일부는 <PackageHeader>나 <RedefinableHeader>의 요소와 매핑된다. XPDL에서 패키지의 프로세스 설계자의 메타정보를 담당하는 것은 <PackageHeader>나 <RedefinableHeader>이기 때문이다. 세부적인 속성에 관한 매핑은 <표 1>과 같이 정의된다. <표 1>에 정의된 속성들은 실행과는 관계없는

〈표 1〉 BPMN의 비즈니스 프로세스 다이어그램 (BusinessProcess-Diagram)과 XPDL의 패키지(Package)와의 속성 매핑

BPMN Attributes	XPDL Elements
Id	Id
Name	Name
Version	<RedefinableHeader>/<Author>
Author	<RedefinableHeader>/<Version>
Language	X
ExpressionLanguage	<Script>
QueryLanguage	X
CreationDate	<PackageHeader>/<Created>
ModificationDate	X
Pools	X
Documentation	<PackageHeader>/<Description>

것이기 때문에, 매핑되지 않는 요소는 복잡 매핑으로 구별하지 않는다.

BPMN의 프로세스는 작업을 수행하는 액티비티의 흐름을 조장하는 데, 프로세스의 흐름에서 발생하는 관계 데이터나 프로세스의 상태 정보를 관리한다. (그림 7)과 같이 BPMN의 프로세스는 XPDL의 워크플로우 프로세스와 매핑되는데, 세부 속성은 표2와 같이 매핑된다. 특히, BPMN 프로세스의 속성 중 실행과 관련된 속성으로 프로퍼티(Property), 할당(Assignment), 애드혹(Adhoc)이 있다. 프로퍼티는 프로세스 실행 중에 사용되는 데이터를 표현하며 할당은 변수에 값을 대입하는 기능을 한다. 이런 기능들은 프로세스의 흐름에서 발생할 수 있는 데이터들을 생성하거나 변경할 수 있다. XPDL의 메타모델[2]에서, 프로세스에서 발생할 수 있는 데이터는 시스템 및 환경 데이터, 액티비티의 흐름을 조절하는 데이터, 애플리케이션을 호출하기 위한 데이터로 구분한다. XPDL에서 이런 데이터들은 관계데이터(Relevant Data)로 규정하며, 데이터필드(DateField)에 의해 정의될 수 있다. 하지만 BPMN의 할당이라는 속성과는 달리, 스크립트 언어를 이용해서 데이터를 변수에 할당할 수 있다. 그리고 애드혹 속성은 복잡한 매핑에 해당하므로 4.4절에서 살펴본다.

〈표 2〉 BPMN의 프로세스(Process)와 XPDL의 워크플로우 프로세스(WorkflowProcess)와의 속성 매핑

BPMN Attributes	XPDL Elements
Id	Id
Name	Name
ProcessType	AccessLevel
Status	X
GraphicalElements	X
Assignments	Script언어에 의해 기술
Properties	<DataFields>
CreationDate	<PackageHeader>/<Created>
Adhoc	복잡 매핑
SuppressJoinFailure	X
EnableInstanceCompensation	X
Catalogies	X
Documentation	<ProcessHeader>/<Description>

하지만 BPMN의 변수와 할당은 프로세스뿐만 아니라 여러 다른 요소에서 정의될 수 있기 때문에, XPDL에서 구별되어 매핑되어야 한다. 변수는 액티비티에서, 할당은 모든 플로우 오브젝트에서 정의될 수 있다. 할당은 프로세스의 할당에서 제시한 방법을 통해서 XPDL에 매핑할 수 있지만, 변수와 매핑되는 XPDL의 데이터필드는 액티비티 단위로는 정의할 수 없기 때문에 여러 액티비티가 공유할 수 있는 프로세스 영역에 정의할 수밖에 없다. 그래서 이것은 [*액티비티 아이디*].[*데이터 필드*]와 같은 형태로 식별되어야 한다.

다음으로 BPMN의 플로우 오브젝트는 크게 이벤트, 액티비티, 게이트웨이로 나누어지는데, 이것들은 XPDL의 애토믹 액티비티, 서브프로세스 액티비티, 블록 액티비티, 라우트 액티비티와 매핑된다. 그래서 이것은 4.3절과 4.4절에서 기술한다.

BPMN의 연결형 오브젝트인 순차 플로우는 일반형, 조건형, 기본형으로 나누어지는데, 이것들은 XPDL의 트랜지션과 조건 타입에 따라 직접적인 매핑된다. 이에 대한 자세한 매핑은 4.3절에서 기술한다.

4.3 심플 매핑(Simple Mapping)

4장에서 기술했듯이, BPMN과 XPDL은 비즈니스를 표현하려는 범위가 다르기 때문에, 이것들을 두 단계로 나누어 매핑할 필요가 있다. 동일한 의미를 갖는 요소들 사이의 매핑을 심플 매핑이라 정의하며, 다른 요소로의 변환이나 흐름·구조 변경을 필요로 하는 경우를 복잡 매핑이라 정의한다. 본 절에서는 심플 매핑에 대해서 기술한다.

BPMN의 플로우 오브젝트는 크게 이벤트, 액티비티, 게이트웨이로 나누어지는데, XPDL의 액티비티와 매핑이 이루어진다. 그리고 BPMN의 순차 플로우는 XPDL의 트랜지션으로 매핑될 수 있는데, 본 절은 앞으로 BPMN의 비즈니스 엔터티를 중심으로 XPDL로의 매핑을 설명한다.

4.3.1 이벤트(Event)

BPMN의 이벤트는 비즈니스 프로세스 중 발생하는 사건을 표현한다. 다시 말해서 이벤트는 어떤 일을 발생시키거나 발생한 일에 대해 반응을 하는 기능을 한다. 예를 들어 정상적으로 주문 처리되던 비즈니스 프로세스가 수행 도중에 주문 취소나 주문 변경이 발생한다면, 이에 따른 프로세스 변경의 작업을 이벤트가 담당하게 된다. 이벤트는 속성에 따라 종류가 나누어지는데 그 종류에는 메시지(Message), 타이머(Timer), 에러(Error), 취소(Cancel), 보상(Compensation), 규칙(Rule), 링크(Link), 다중(Multiple), 종료(Terminate), 그

〈표 3〉 BPMN의 일반(None) 이벤트와 XPDL의 라우트 액티비티와의 심플 매핑

BPMN	BPMN Attributes	XPDL Elements	XPDL Schema
	X	X	<Activity Id="[Event_Id]" Name="[Event_Name]"> <Route/> </Activity>

리고 일반(None) 이벤트가 있다. 이것들은 프로세스의 모델링 위치에 따라 프로세스의 시작, 중간, 끝임을 나타내며, 속성에 따라 각자 다른 기능을 하게 된다.

첫째로, 일반 이벤트는 어떤 특정 사건과도 연관이 없는 것을 의미하며, 프로세스의 시작과 끝에 위치하여 단순히 프로세스의 시작과 끝임을 나타낸다. 시작 이벤트는 XPDL의 라우트 액티비티와 동일한 기능을 하기 때문에, 표 3처럼 매핑된다. 하지만 종료 이벤트는 복잡한 매핑에 해당하기 때문에 4.4절에서 기술한다.

둘째로, 메시지 이벤트는 외부 프로세스로부터 '메시지 받기', '메시지 보내기', 또는 '메시지 받고 예외처리하기'와 같은 기능을 한다. 이것은 XPDL에서는 '메시지 받고 예외처리하기'를 제외하고는 애토믹 액티비티가 동일한 기능을 한다. 다시 말해서 애토믹 액티비티는 매개변수를 통해 메시지를 받는 애플리케이션이나 메시지를 보내는 애플리케이션을 사용하여 '메시지 받기', '메시지 보내기'의 기능 하게 되며, 이러한 애토믹 액티비티가 BPMN의 메시지 이벤트와 매핑된다. 하지만 메시지 시작 이벤트는 조금 다르다. 왜냐하면 메시지 시작 이벤트는 해당하는 메시지를 받았을 때 프로세스를 시작한다는 의미인데 XPDL에서는 이러한 메시지를 워크플로우 프로세스의 입력 형식파라미터(FormalParameter)로 표현하기 때문이다. 그러므로 BPMN의 메시지 시작 이벤트에서 요구하는 메시지를 XPDL 워크플로우 프로세스의 입력 형식파라미터로 매핑하고 라우트 액티비티를 통해 프로세스의 시작임을 나타내어야 한다. 또한 메시지 끝 이벤트의 매핑에서도 애토믹 액티비티로만 매핑한다면 프로세스의 끝임이 표현되지 않는다. 그러므로 애토믹 액티비티 이후 프로세스의 끝을 나타내는 라우트 액티비티를 추가해 주어야 한다. 표 4는 BPMN의 메시지 이벤트의 심플 매핑을 나타낸다. 여기서 XPDL 스키마는 메시지 중간 이벤트를 기술하고 있다. 메시지 시작/끝 이벤트는 약간의 위에서 기술한데로 XPDL 스키마를 변형하면 된다.

셋째로, 타이머 이벤트는 프로세스의 시작 또는 중간에 위치하여 특정 시기가 되었을 때 프로세스를 시작한다거나 '프로세스의 흐름을 지연시키기', '프로세스를 주기적으로 실행하기' 등의 기능을 한다. 이것은 XPDL에서는 시간에 따른 행동에 대해서 '데드라인(Deadline)' 이라는 액티비티 요소와 이를 핸들링 하는 예외(Exception) 트랜지션을 제공하며, 비주기적인 기능을 하는 BPMN의 타이머 이벤트와 같은 표현을 할 수 있다. 하지만 XPDL의 데드라인은 주기적인 실행에 대해서는 직접적으로 표현할 수 없다. 이러한 시간반복(TimeCycle) 타이머 이벤트에 대한 XPDL 매핑은 복잡 매핑에 해당하므로 4.4절에서 기술한다. 표 5는 BPMN의 타이머 이벤트와 XPDL의 구현 액티비티와의 심플 매핑을 나타낸다.

〈표 4〉 BPMN의 메시지(Message) 이벤트와 XPDL의 구현 액티비티와의 심플 매핑

BPMN	BPMN Attributes	XPDL Elements	XPDL Schema
	Message	<ActualParameters>	<Activity Id="[Event_Id]" Name="[Event_Name]"> <Implementation> <Tool Id="[Implementation]"> <ActualParameters/> </Tool> </Implementation> </Activity>
	Implementation	<Implementation>/<Tool>	

〈표 5〉 BPMN의 타이머(Timer) 이벤트와 XPDL의 구현 액티비티와의 심플 매핑

BPMN	BPMN Attributes	XPDL Elements	XPDL Schema
	TimerDate	<Deadline>/<DeadlineCondition>	<Activities> <Activity Id="[Target_Activity_Id]" Name="[Target_Activity_Name]"> <Implementation/> <Deadline Execution="SYNCHR"> <DeadlineCondition>[TimeDate]</DeadlineCondition> <ExceptionName>[Event_Id]_Exception</ExceptionName> </Deadline> </Activity> </Activities> <Transitions> <Transition Id="[Next_Transition_Id]" Name="[Next_Transition_Name]" From="[Event_Id]" To="[Next_Activity_Id]"> <Condition Type="EXCEPTION">[Event_Id]_Exception</Condition> </Transition> </Transitions>
	TimerCycle	복잡 매핑	

이벤트에 대한 간단한 매핑을 하였지만 아직 많은 이벤트가 매핑되지 않았다. 메시지 이벤트에 대한 예외 핸들링, 주기를 나타내는 타이머 이벤트, 에러 이벤트, 취소 이벤트, 보상 이벤트, 규칙 이벤트, 링크 이벤트, 다중 이벤트, 종료 이벤트가 아직 매핑되지 않은 이벤트들이다. 이것들의 XPDL 매핑은 복잡 매핑에 해당하므로 모두 4.4절에서 기술한다.

4.3.2 액티비티(Activity)

BPMN의 액티비티는 비즈니스 프로세스 중에 작업을 처리하는 요소이다. 예를 들어 신용정보 조회하기, 결제하기와 같은 일이 이에 해당된다. 이러한 액티비티는 작업의 단위에 따라 다시 서브프로세스와 태스크로 나누어진다.

서브 프로세스는 특정 작업을 외부 또는 내부 프로세스에 일을 전담하는 액티비티를 지칭한다. 서브프로세스는 다시 타입에 따라 내장형(Embedded), 독립형(Independent), 참조형(Reference)으로 나누어진다.

내장형 서브프로세스는 부모 프로세스의 데이터를 공유하며 부모 프로세스에 의해 초기화된다. 한편 XPDL의 블록

〈표 6〉 BPMN의 내장형(Embedded) 서브프로세스와 XPDL의 블록 액티비티의 심플 매핑

BPMN	BPMN Attributes	XPDL Elements	XPDL Schema
(내장형)	Graphical Elements	X	<ActivitySet Id="[Sub-Process_Id]_Ref"> <Activities/> <Transitions/> </ActivitySet> <Activities> <Activity Id="[Sub-Process_Id]" Name="[Sub-Process_Name]"> <BlockActivity BlockId=""/> </Activity> </Activities>
	AdHoc	복잡매핑	
	AdHocOrdering		
	AdHocCompletionCondition		

〈표 7〉 BPMN의 독립형(Independent) 서브프로세스와 XPDL의 블록 액티비티의 심플 매핑

BPMN	BPMN Attributes	XPDL Elements	XPDL Schema
(독립형)	DiagramRef	<External-Package>	<Activity Id="[Sub-Process_Id]" Name="[Sub-Process_Name]"> <Implementation> <Subflow Id="[ProcessRef_Id]" Execution="SYNCHR"> <ActualParameters/> </Subflow> </Implementation> </Activity>
	ProcessRef	<Implementation> /<Subflow>/Id	
	Input-PropertyMap	<Actual-Parameters> :INMODE	
	Output-PropertyMap	<Actual-Parameters> :OUTMODE	

액티비티도 같은 워크플로우 프로세스 상에 정의된 액티비티 집합을 호출하는데, 액티비티 집합은 워크플로우 프로세스의 데이터를 공유하며 여러 액티비티와 트랜지션들의 집합으로 작업을 수행한다. 그러므로 BPMN의 내장형 서브프로세스는 XPDL의 블록 액티비티와 매핑된다. 표 6은 BPMN의 내장형 서브프로세스와 XPDL의 블록 액티비티의 매핑을 나타낸다.

독립형 서브프로세스는 부모 프로세스와의 의존성 없이 존재하기 때문에, 부모 프로세스로부터 초기화되지 않고, 전역 데이터를 공유하지도 않는다. 이것은 XPDL의 서브플로우 액티비티와 매핑될 수 있는데, 서브플로우 액티비티는 독립된 워크플로우 프로세스로써 독자적인 관계 데이터와 라이프 사이클이 존재하기 때문이다. <표 7>은 BPMN의 독립형 서브프로세스와 XPDL의 블록 액티비티의 매핑을 나타낸다.

마지막으로 참조형 서브프로세스는 미리 정의되어 있는 서브프로세스를 참조하는 프로세스이다. 이것은 두 서브프로세스가 정확히 동일한 행동과 성질을 요구할 때 사용되는 것으로, XPDL에서는 미리 정의된 서브프로세스의 아이디를 참조하여 정의할 수 있다.

태스크는 타입에 따라 총 8가지로 구성되어 있다. 시스템에 의해서 자동으로 실행되는 일반(None) 태스크, 입/출력 메시지 모두를 필요로 하는 서비스(Service) 태스크, 입력 메시지를 필요로 하는 수신(Receive) 태스크, 출력 메시지를 필요로 하는 송신(Send) 태스크, 미리 정의된 애플리케이션의 도움으로 사람이 태스크를 수행하는 유저(User) 태스크, 시스템에서 지원하는 스크립트가 실행되는 스크립트(Script) 태스크, 시스템의 도움 없이 사람에 의해 수행되는 수동(Manual) 태스크, 그리고 미리 정의된 태스크를 참조하는 참조(Reference) 태스크로 나누어진다. 이것들은 스크립트 태스크를 제외하고 모두 XPDL의 애토믹 태스크로 매핑될 수 있다.

BPMN의 일반 태스크는 어떤 작업도 수행하지 않는 태스크이다. 이것은 표 3의 일반 이벤트와 동일하게 XPDL의 라우트 액티비티로 매핑될 수 있다. 일반 태스크도 세부 속성이 존재하지 않기 때문에 표 3과 동일한 XPDL 스키마를 갖는다.

BPMN의 서비스 태스크, 수신 태스크, 송신 태스크와 유저 태스크는 XPDL의 워크플로우 프로세스 안에 정의된 애플리케이션을 이용하여 수행하는 애토믹 태스크와 같으며, 이용하는 애플리케이션에서 필요로 하는 입력/출력 매개변수(ActualParameter), 수행자(Performer)에 따라 표 8처럼 직접적으로 매핑된다.

BPMN의 수동 태스크는 애플리케이션을 이용하지 않고 사용자에 의해 작업이 수행되는 수동 모드 애토믹 액티비티로 매핑되기 때문에, 표 9처럼 매핑된다.

BPMN의 스크립트 태스크는 특정 비즈니스 프로세스 시스템에 의존적인 요소이므로, 추가적인 확장 속성(Extended Attribute)을 정의하여 매핑하여야 한다. 마지막으로 참조 태스크는 XPDL에서 미리 정의된 태스크의 아이디를 참조하여 정의할 수 있다.

〈표 8〉 BPMN의 수신(Receive), 송신(Send), 서비스(Service), 사용자(User) 태스크와 XPDL의 구현 액티비티와의 심플 매핑

BPMN	BPMN Attributes	XPDL Elements	XPDL Schema
수신	Message	<ActualParameters>:INMODE	<Activity Id="[Task_Id]" Name="[Task_Name]"> <Implementation> <Tool Id="[Implementation]"> <ActualParameters/> </Tool> </Implementation> [<Performer>[Performers]</Performer>] </Activity>
	Instantiate	X	
	Implementation	<Implementation>/<Tool>	
송신	Message	<ActualParameters>:OUTMODE	
	Implementation	<Implementation>/<Tool>	
서비스	InMessage	<ActualParameters>:INMODE	
	OutMessage	<ActualParameters>:OUTMODE	
	Implementation	<Implementation>/<Tool>	
사용자	Performer	<Performer>	
	InMessage	<ActualParameters>:INMODE	
	OutMessage	<ActualParameters>:OUTMODE	
	Implementation	<Implementation>/<Tool>	

〈표 9〉 BPMN의 수동(None) 태스크와 XPDL의 구현 액티비티와의 심플 매핑

BPMN	BPMN Attributes	XPDL Elements	XPDL Schema
수동	Performer	<Performer>	<Activity Id="[Task_Id]" Name="[Task_Name]"> <Implementation> <No/> </Implementation> <Performer>[Performers]</Performer> <StartMode> <Manual/> </StartMode> <FinishMode> <Manual/> </FinishMode> </Activity>

〈표 10〉 BPMN의 게이트웨이와 XPDL의 라우트 액티비티와의 심플 매핑

BPMN	BPMN Attributes	XPDL Elements	XPDL Schema
	X	X	<Activity Id="[Gateway_Id]" Name="[Gateway_Name]"> <Route/> <TransitionRestrictions> <TransitionRestriction> <[Split | Join] Type="AND|XOR"> <TransitionRefs/> </Split> </TransitionRestriction> </TransitionRestrictions> </Activity>
	DefaultGate	X	
	MarkerVisible	X	
	DefaultGate	X	
	IncomingCondition	X	여러 라우트 액티비티의 조합
	OutgoingCondition	X	

BPMN의 액티비티는 추가적으로 시작 토큰 횟수(Start-Quantity), 루프 타입과 서브프로세스의 트랜잭션(Transaction), 독립형 서브프로세스의 입력변수 맵(InputPropertyMap)과 출력변수 맵 (OutputPropertyMap), 그리고 태스크의 구현(Implementation)과 같은 프로세스 실행과 관련된 속성을 포함하고 있다. 입력/출력변수 맵이란 속성은 호출되는 서브프로세스와 호출하는 액티비티 간의 데이터를 연결시키는 요소로써 XPDL에서 서브플로우 액티비티의 입력/출력 매개변수(ActualParameter)와 호출되는 프로세스의 입력/출력 형식파라미터(FormalParameter)간의 관계로 매핑된다. 태스크의 구현 속성은 태스크 실행을 위해 사용하는 애플리케이션이나 웹서비스와 같은 서비스를 나타내는 요소로써 XPDL의 액티비티 실행을 위해 사용하는 애플리케이션을 나타내는 툴(Tool)과 매핑된다. 그리고 시작 토큰 횟수, 루프타입, 트랜잭션에 대한 매핑은 복잡 매핑에 해당하므로 4.4절에서 기술한다.

4.3.3 게이트웨이(Gateway)

BPMN의 게이트웨이는 여러 순차 플로우가 어떻게 분기 또는 합병할지에 대한 제어를 담당한다. 그 종류에는 병렬 분기를 하는 병렬 포크(Parallel Fork), 병렬 동기화 합병하는 병렬 조인(Parallel Join), 배타적 선택을 하는 데이터기반(Data-based)과 이벤트기반(Event-based) XOR 분기(Decision), 단일 합병을 하는 XOR 합병(Merge), 다중 선택을 하는 OR 분기, 다중 선택 동기화 합병을 하는 OR 합병, 그리고 여러 개의 게이트웨이로 표현할 분기 또는 합병을 하나의 게이트웨이로 나타내는 복합(Complex) 분기와 합병이 있다. XPDL에는 BPMN과 유사하게 병렬 분기와 다중 선택을 하는 AND 분할(Split), 병렬 동기화 합병과 다중 선택 동기화 합병을 하는 AND 조인(Join), 배타적 선택을 하는 XOR 분할, 단일 합병을 하는 XOR 조인이 있다. BPMN의 게이트웨이는 BPMN의 이벤트기반 분기와 복합 게이트웨이를 제외하고 XPDL의 라우트 액티비티로 직접적인 매핑을 할 수 있다. 표 10은 BPMN 게이트웨이와 XPDL의 라우트 액티비티의 매핑을 나타낸다. BPMN의 게이트웨이는 게이트웨이에 연결된 순차 플로우마다 게이트(Gate)를 갖는데, 이러한 요소는 XPDL에 없다. 하지만 XPDL의 트랜지션(Transition)은 흐름에 대한 조건이나 예외 처리를 할 수 있기 때문에, BPMN의 게이트와 순차 플로우의 기능을 모두 표현할 수 있다. 복합 게이트웨이는 그 내부 속성인 입력/출력 조건의 표현에 따라 여러 개의 게이트웨이로 매핑될 수 있다. 이벤트기반 분기에 대한 XPDL 매핑은 복잡 매핑에 해당하므로 4.4절에서 기술한다.

4.3.4 순차 플로우(Sequence Flow)

BPMN의 순차 플로우는 한 프로세스 안에서의 플로우 오브젝트 간 흐름을 나타낸다. 순차 플로우는 조건 속성의 값에 따라 일반형(Common), 조건형(Condition), 기본형(Default)의 세 가지 타입으로 나누어진다. 첫째로 일반형 순차 플로우는 실행 토큰이 도착했을 때 아무 조건이 없이 다음 플로우 오브젝트로 토큰을 전송하여 프로세스를 진행하는 것을 의미한다. 둘째로 조건형 순차 플로우는 실행 토큰이 도착했을 때 조건이 만족하면 다음 플로우 오브젝트로 진행하는 것을 의미한다. 셋째, 기본형 순차 플로우는 다른 순차 플로우의 모든 조건이 만족하지 않을 때 다음 플로우 오브젝트로 진행하는 것을 의미한다. XPDL에서 이와 같은 기능을 하는 요소로는 하나의 워크플로우 프로세스 안에서 액티비티 간의 흐름을 나타내는 트랜지션이 있다. 트랜지션은 조건 요소의 유무와 조건 타입에 따라 표 12처럼 나누어지는데, 이것은 BPMN과 상당히 유사한 구조를 갖고 있기 때문에 직접적인 매핑이 이루어질 수 있다. BPMN의 순차 플로우와 XPDL의 트랜지션은 흐름의 시작과 끝은 정의할 수 있고, 조건에 따라 흐름을 조절할 수 있다. 이 때, 관계 데이터의 값을 이용해서 프로세스의 흐름을 정의한다. 이러

〈표 11〉 BPMN의 순차 플로우(Sequence Flow)와 XPDL의 트랜지션(Transition)과의 속성 매핑

BPMN Attributes	XPDL Elements
Name	Name
Source	From
Target	To
ConditionType	<ConditionType>
ConditionExpression	<Condition>
Quantity	X

〈표 12〉 BPMN의 순차 플로우 매핑

BPMN	XPDL Schema
	<Transition Id="[SequenceFlow_Id]" Name="[SequenceFlow_Name]" From="[Source]" To="[Target]"/>
	<Transition Id="[SequenceFlow_Id]" Name="[SequenceFlow_Name]" From="[Source]" To="[Target]"> <Condition Type="CONDITION"> [ConditionalExpression] </Condition> </Transition>
	<Transition Id="[SequenceFlow_Id]" Name="[SequenceFlow_Name]" From="[Source]" To="[Target]"> <Condition Type="OTHERWISE"/> </Transition>

한 기능을 일반적으로 데이터 플로우(Data Flow)라고 지칭하기도 한다. 표 11은 BPMN의 순차플로우와 XPDL의 트랜지션의 세부 속성에 대한 매핑이다.

4.4 복잡 매핑(Complex Mapping)

지금까지의 매핑을 통해서 BPMN과 XPDL이 비즈니스 프로세스에 대해 서로 유사한 구조와 요소들로 이루어져 있음을 보았다. 그래서 이것들은 직접적으로 매핑할 수 있었다. 하지만 BPMN의 목적이 기업 간 상황에 적합하도록 비즈니스 엔터티를 구성하였기 때문에, XPDL보다는 좀 더 풍부한 표현력을 갖추고 있어 심플 매핑으로는 매핑에 한계가 있다. 예를 들어, 에이전트에게 주문에 대한 결제 방법을 선택하기 위해 이벤트기반 게이트웨이를 이용하였다고 가정하자. 이벤트기반 게이트웨이는 여러 개의 결제 방법을 이벤트로 연결하여 프로세스의 흐름을 결정하는데, XPDL과 같은 기존 기업 내에 관련 비즈니스 언어는 결정된 비즈니스에서만 프로세스를 진행했기 때문에 이벤트기반 게이트웨이처럼 복합적인 의미의 엔터티를 필요로 하지 않았다. 즉, BPMN의 이벤트나 이벤트기반 분기 이벤트를 기반으로 하는 요소나 애드혹 프로세스 같은 XPDL에서 정의되지 않은 요소, 복합 게이트웨이나 다중 이벤트와 같이 함축적으로 표현하는 요소들은 XPDL로 직접적으로 매핑되지 않는다. 그래서 본 절에서는 정의 1에 기반한 이행적인 변환 기법을 이용해 복잡한 의미의 엔터티들을 매핑한다.

아직 BPMN에서 XPDL로 매핑을 이루지 못한 요소는 애드혹 프로세스, 메시지 이벤트에 대한 예외 처리, 주기를 나타내는 타이머 이벤트, 에러 이벤트, 취소 이벤트, 보상 이벤트, 규칙 이벤트, 링크 이벤트, 다중 이벤트, 종료 이벤트, 이벤트기반 XOR 분기, 액티비티의 속성인 시작 토큰 회수와 루프타입, 그리고 서브프로세스의 속성인 트랜잭션이다. 4장의 가정에 의해 취소 이벤트, 보상 이벤트, 서브프로세스의 트랜잭션은 기업 내보다는 기업 간의 상황에 특성화된 요소들이기 때문에 본 논문의 매핑 요소에 포함시키지 않는다. 또, 규칙 이벤트는 비즈니스 규칙(Business Rule)을 정의하는 요소이기 때문에, 각 벤더에 따라 다양하게 정의될 수 있기 때문에 본 논문의 매핑 요소에 포함시키지 않는다. 이것들을 제외한 나머지 매핑하지 않은 요소들은 정의 1에 의해 구성한 세 가지 기법을 이용하여 매핑을 적용한다.

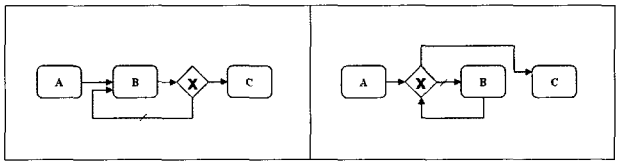
[정의 1] 비즈니스 공통성(Business Commonality)

비즈니스(BE)를 구성하는 하나의 비즈니스 엔터티(e_i)가 다른 비즈니스 엔터티들(e_j , ... , e_k)의 집합으로 정의될 수 있다는 것을 의미한다.

$$ei = \{ej , \ldots , ek\} \text{ where } BE = \{e1 , \ldots , ei . \ldots , en\}$$

[기법 1] 루프(Loop)

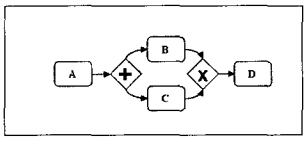
루프란 같은 액티비티를 여러 번 수행하는 것을 의미한다. BPMN에서는 루프타입 액티비티를 통해 이러한 기능을 제공한다. 그러나 루프 타입 액티비티를 이용하지 않더라도 정의 1에 의해 BPMN의 액티비티와 게이트웨이를 통해서 같은 표현을 할 수 있다. 즉, (그림 8)과 같이 액티비티와 게이트웨이의 흐름을 구성하면 B 액티비티는 게이트웨이의 제어에 따라 여러 번 수행될 수 있다. 아직 매핑이 정의되지 않은 요소들 중 애드혹 프로세스, 주기를 나타내는 타이머 이벤트, 액티비티의 속성인 시작 토큰 횟수와 루프타입이 루프 기법을 이용해 매핑될 수 있다.



(그림 8) BPMN의 액티비티와 게이트웨이를 통한 루프 흐름의 표현

[기법 2] 식별화(Discriminator)

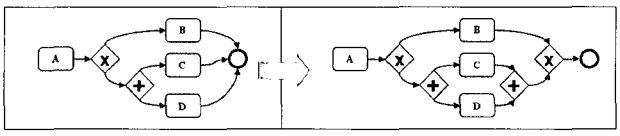
식별화는 동시에 실행 중인 두 개 이상의 액티비티에 대해 하나의 액티비티만을 완수하는 제어 흐름의 특징을 이용하여 프로세스를 재구성하는 기법이다. 이것은 기존에 연구되었던 21가지 워크플로우 패턴[7]중의 9번째 패턴을 매핑에 응용한 것이다. BPMN은 액티비티에 이벤트를 부착하여 예외 처리를 가능하게 하거나 다중 이벤트처럼 복합적인 의미의 엔터티를 포함하고 있다. 이와 같은 BPMN의 엔터티들은 복합적인 의미를 갖고 있지만 실행에서는 오직 하나의 엔터티만을 수행하도록 하였다. 그래서 정의 1에 의해 (그림 9)의 식별화 기법을 정의한다. 아직 매핑이 정의되지 않은 요소들 중 메시지 이벤트에 대한 예외 처리, 에러 이벤트, 다중 이벤트, 이벤트기반 XOR 분기가 식별화 기법을 이용해 매핑될 수 있다.



(그림 9) 식별화(Discriminator) 기법

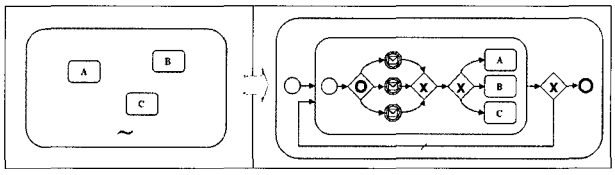
[기법 3] 순서화(Serialize)

순서화는 BPMN으로 구성된 프로세스에 대해 의미적인 순서를 파악하여 프로세스의 흐름을 순서에 따라 재구성하는 기법이다. 이것은 각 요소마다 서로 다른 패턴으로 나타나며 매 경우마다 다르게 처리되어야 한다. (그림 10)은 끝(End) 이벤트에 대한 순서화 기법 적용 예이다. BPMN의 끝 이벤트는 임의적으로 표현된다. 즉, 여러 다른 합병을 필요로 하는 순차 플로우를 모두 끝 이벤트로 직접 연결하기 때문에 이는 (그림 10)에서처럼 나중에 분기된 게이트웨이부터 먼저 분기한 게이트웨이로 순서화하여 합병하여야 한다. 아직 매핑이 정의되지 않은 요소들 중 링크 이벤트, 종료 이벤트가 순서화기법을 이용해 매핑될 수 있다.



(그림 10) BPMN의 임의적인 종료에 대한 순서화 기법 적용

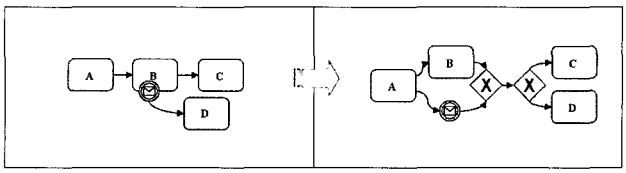
첫째로, 애드혹 프로세스는 프로세스 안의 액티비티들이 서로 순차 플로우의 연결 없이 순서가 정해지지 않은 채로 있는 경우이다. 프로세스 안의 각 액티비티가 언제 수행될 것이며, 언제 끝나는지, 그리고 어떤 순서로 진행될 것인지는 실행시점에 수행자에 의해 결정된다. 그래서 애드혹 프로세스는 (그림 11)에서와 같이 식별화 기법과 루프 기법을 통해 같은 의미로 재구성할 수 있다. 즉, 식별화 기법을 통해 아직 실행되지 않은 액티비티에 대해 선택적으로 실행되게 하고, 루프 기법을 통해 완료 조건이 만족하지 않았을 경우 다음 실행할 액티비티를 다시 선택할 수 있도록 애드혹 프로세스를 표현할 수 있다.



(그림 11) 애드혹 프로세스의 이행적 변환

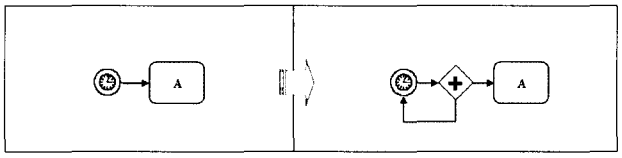
둘째, 메시지 이벤트에 대한 예외 처리는 액티비티에 메시지 이벤트가 붙어서 액티비티 실행 중 만약 해당하는 메시지가 도착할 경우 예외 처리하는 기능을 한다. 이것은 액티비티와 액티비티에 부착된 이벤트 둘 중에 하나만을 수행하도록 처리하면 되기 때문에 식별화 기법을 적용한다. (그림 12)는 식별화 기법을 이용해 액티비티에 부착된 메시지 이벤트를 같은 의미로 재구성한 그림이다. 변경된 프로세스에서는 원래의 프로세스에서와 마찬가지로 액티비티 B에 대해서만 흐름을 진행하거나 또는 메시지 이벤트에 대해서만 흐름을 진행 할 수 있다.

셋째, 주기적으로 실행하는 타이머 이벤트는 일정한 주기에 따라 프로세스를 새로 시작하는 기능을 한다. 예를 들어



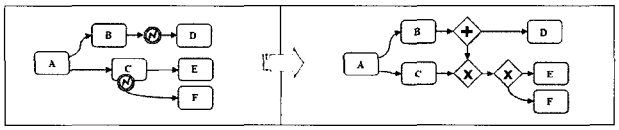
(그림 12) 메시지 이벤트에 대한 예외 처리의 이행적 변환

일주일 간격으로 자료 백업을 처리하는 프로세스가 바로 이에 해당된다. 이것은 특정 시간마다 토큰을 발생해야 하기 때문에, (그림 13)에서와 같이 XPDL에서 표현할 수 있는 비주기적인 타이머 이벤트와 루프 기법을 통해 같은 의미로 재구성할 수 있다.



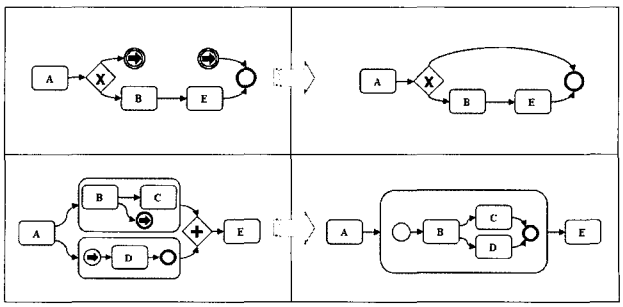
(그림 13) 주기적으로 실행하는 타이머 이벤트의 이행적 변환

넷째, 에러 이벤트는 에러 이벤트를 발생시키거나, 액티비티에 붙어서 발생하는 에러 이벤트에 반응하여 액티비티에 대한 에러 핸들링을 한다. 이것은 메시지 이벤트가 액티비티에 부착된 상황과 상당히 유사하기 때문에 (그림 14)처럼 식별화 기법을 이용해 같은 의미로 재구성할 수 있다.



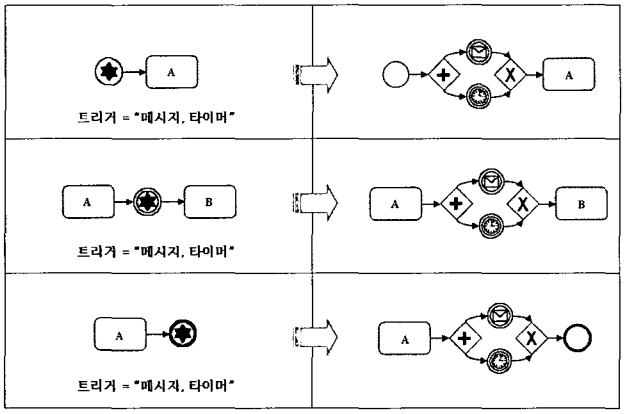
(그림 14) 에러 이벤트의 이행적 변환

다섯째, 링크 이벤트는 두 가지 기능을 하는데, 첫 번째 기능은 프로세스 디자이너를 위한 것이고 두 번째는 프로세스의 흐름에서 동기화를 맞추기 위한 것이다. 먼저 프로세스 디자이너는 한 도면을 벗어나는 프로세스를 디자인할 수 있기 때문에 링크 이벤트가 필요하다. 이와 같은 링크 이벤트는 단순히 링크 아이디에 따라 연결만 하면 되기 때문에, (그림 15)의 상단 그림처럼 나타난다. 두 번째는 링크 이벤트가 BPEL4WS의 <link>나 BPML의 <signal>과 비슷한 기능으로 사용된 예이다. 이것은 프로세스 흐름의 동기화를 구성하기 위한 것이기 때문에 (그림 15)의 하단 그림처럼 순서화 기법을 이용해 같은 의미로 재구성할 수 있다.



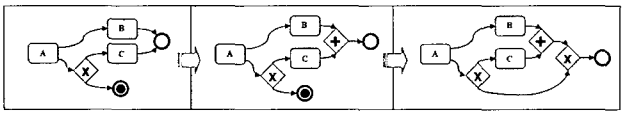
(그림 15) 링크 이벤트의 이행적 변환

여섯째, 다중 이벤트는 다중 이벤트가 포함하고 있는 여러 이벤트 중 하나라도 만족하는 이벤트가 발생하면 다음 흐름을 진행하는 이벤트이다. 이것은 복합적 의미의 엔터티를 여러 단순 엔터티로 분할 하여 특정 엔터티만 수행하면 되기 때문에 (그림 16)과 같이 식별화 기법에 따라 재구성할 수 있다.



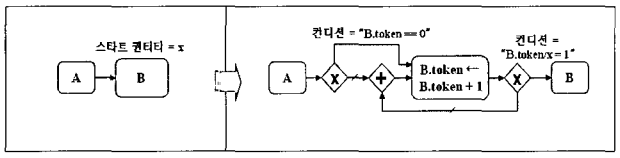
(그림 16) 다중 이벤트의 이행적 변환

일곱째, 종료 이벤트는 종료 이벤트에 도달하면 이벤트가 속한 프로세스에 다른 액티비티가 진행 중이라도 아무런 보상 없이 모두 중단하고 프로세스를 종료하는 기능을 한다. 예를 들어 (그림 17)의 프로세스는 액티비티 B의 실행 중에 종료 이벤트에도 도달할 수 있다. 이 경우 만약 액티비티 B가 실행 중이라도 종료 이벤트에 도달하면 바로 프로세스는 중단되어야 하기 때문에, 식별화 기법을 이용해 종료 이벤트 도달 시 다른 액티비티를 무시하고 바로 종료하도록 재구성할 수 있다.



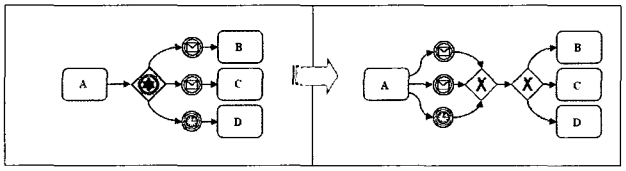
(그림 17) 종료 이벤트의 이행적 변환

여덟째, 액티비티의 속성인 시작 토큰 횟수는 액티비티가 수행되기 위해 연결된 하나의 순차 플로우 당 필요로 하는 토큰의 개수를 의미한다. 예를 들어 (그림 18)에서 만약 액티비티 B의 시작 토큰 횟수가 '3'이라면 이전 액티비티 A로부터 연결된 순차 플로우로부터 3개의 토큰이 도착해야 다음을 진행하게 된다. 그리고 이것은 루프 기법을 이용하여 3개의 토큰이 도착할 때까지 대기하도록 재구성할 수 있다.



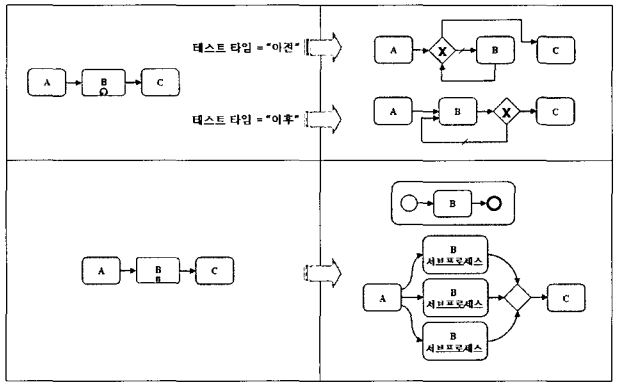
(그림 18) 액티비티의 속성인 시작 토큰 횟수의 이행적 변환

아홉째로, 이벤트기반 XOR 분기는 분기된 이벤트 액티비티 중 먼저 발생하는 이벤트에 대해서만 처리하는 배타적 분기를 표현한다. 이것은 여럿으로 분기된 흐름 중 가장 먼저 도달하는 흐름에 대해서만 처리하면 되기 때문에 식별화 기법을 적용할 수 있다. 따라서 이벤트기반 XOR 분기는 (그림 19)에서와 같이 식별화 기법을 통해 같은 의미로 재구성할 수 있다.



(그림 19) 이벤트기반 XOR 분기의 이행적 변환

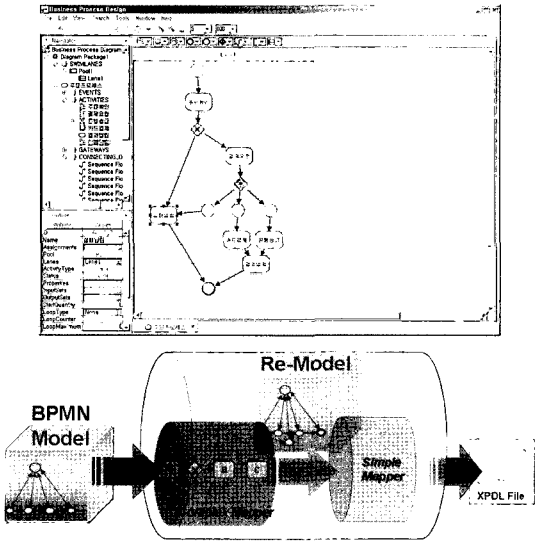
마지막으로, 액티비티에는 루프 타입 속성에 따라 표준(Standard) 루프와 다중 인스턴스 (Multiple Instance)를 표현할 수 있다. 또 다중 인스턴스는 타입에 따라 순차 다중 인스턴스와 병렬 다중 인스턴스로 나누어진다. 이 중 표준 루프와 순차 다중 인스턴스는 조건에 따라 같은 액티비티를 순서대로 여러 번 수행하는 기능을 하기 때문에, 루프 기법에 따라 같은 의미로 재구성될 수 있다. 그리고 병렬 다중 인스턴스는 같은 액티비티를 조건에 따라 동시에 여러 번 병렬 수행하는 기능을 한다. 이것은 병렬 수행할 액티비티를 여러 개의 서브프로세스로 병렬 분기하여 호출함으로써 병렬 다중 인스턴스 액티비티를 표현할 수 있기 때문에 순서화 기법을 적용한다. (그림 20)은 액티비티의 속성인 루프 타입에 관련된 매핑을 나타낸다.



(그림 20) 액티비티의 속성인 루프타입의 이행적 변환

5. 구현 및 예제

본 장에서는 4장에서 기술한 매핑 기법들이 어떻게 사용되는 지를 실제 예제를 기반으로 설명하겠다. 본 논문은 BPMN에 비해 XPDL의 부족한 비즈니스 표현력을 보완하기 위해 이행적인 매핑 기법을 제시하였다. 이행적인 매핑 기법은 복합적 의미의 비즈니스 엔터티를 간단한 비즈니스

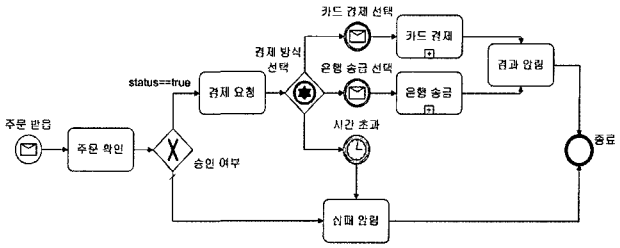


(그림 21) BPMNtoXPDL 구현 및 모델

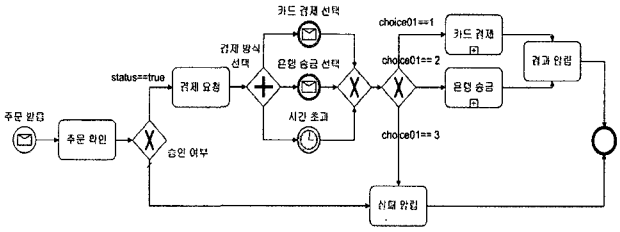
엔터티로 바꾸는 역할만 하기 때문에, 최종적으로 XPDL로 바꾸기 위해서는 심플 매핑을 이용해야만 한다. 즉, 복잡 매핑을 통한 비즈니스 엔터티들도 심플 매핑을 통해야지만 XPDL로 변환될 수 있다. (그림 21)의 왼쪽 그림은 본 논문의 적합성을 판단하기 위한 BPMN 프로세스 디자인 툴이다. 다자인된 비즈니스 프로세스는 오른쪽 그림의 BPMN Model로 구성된다. 그리고 구성된 모델로부터 (그림 21)의 아키텍처를 이용해 XPDL 문서를 결과를 얻을 수 있다.

(그림 22)는 일반적인 서비스 업체에서 많이 응용될 수 있는 주문 프로세스이다. 프로세스는 사용자 또는 에이전트로부터 주문 메시지를 받음으로써 수행이 시작된다. 먼저 주문된 메시지가 유효한지 확인하기 위해 "주문 확인" 태스크가 수행된다. 예를 들어, 제품 재고가 남아있지 않거나 제품의 유효 기간이 벗어날 경우가 있는지 확인한다. 그 후, 주문의 승인 여부에 따라 "결제 요청" 태스크가 수행되거나 "실패 알림" 태스크가 수행된다. "결제 요청" 태스크는 송신 태스크로 사용자에게 결제에 대한 메시지를 송신한다. 이때 메시지는 여러 가지 결제 방법을 포함하고 있다. 사용자는 "카드 결제"나 "은행 송금"을 통해 주문에 대한 결제를 할 수 있다. 하지만 사용자가 특정 시간 내에 결제를 처리하지 않으면, 프로세스는 "실패 알림" 태스크를 수행하여 사용자에게 주문 실패에 대한 메시지를 송신한다. "카드 결제"는 외부 파티에서 수행하는 프로세스로 독립형(Independent) 서브프로세스로 구성하였으며, "은행 송금"은 내부에서도 처리할 수 있다고 가정하여 내장형(Embedded) 서브프로세스로 구성하였다. 마지막으로 "결과 알림" 태스크가 수행되어 사용자에게 주문 결제가 정확히 처리되었다는 메시지를 송신한다.

(그림 22)의 예제를 (그림 21)의 모델을 기반으로 매핑을 하기 위해서, 먼저 복잡 매핑에 해당하는 비즈니스 엔터티들을 검색한다. 본 예제에서 복잡 매핑에 해당하는 엔터티는 이벤트기반 게이트웨이의 XOR 분기뿐이다. 이것은 (그림 19)에서와 같이 식별화 기법을 이용하여 프로세스를 재구성할 수 있다. (그림 23)은 복잡 매핑에 해당하는 엔터티를 제거하여 재구성한 프로세스이다. (그림 23)의 모든 비즈니스 엔터티들은 심플 매핑을 적용할 수 있는 엔터티들이다.



(그림 22) BPMN의 주문 프로세스



(그림 23) 이벤트기반 XOR 분기를 제거하여 재구성한 주문 프로세스

6. 결론 및 향후 과제

본 논문은 프로세스 모델링 표기법(BPMN)과 프로세스 실행언어(XPDL) 사이의 간격을 줄임으로써 현업의 프로세스 설계자와 프로세스 실행 모듈의 차이를 최소하려는 방법을 연구하였다. 뿐만 아니라 모델링 하는 비즈니스 영역에 대한 차이를 최소하기 위한 방법들도 제안하였다. XPDL은 기업 내의 비즈니스를 모델링하는 언어이고, BPMN은 기업 내외의 비즈니스를 모델링하는 표기법이기 때문에, 비즈니스 영역에 대한 차이가 발생한다. 이런 문제를 해결하기 위해 본 연구에서는 이행적인 매핑기법을 제시하였다.

기존의 BPMN과 같은 비즈니스 프로세스 표기법에서 XPDL과 같은 비즈니스 프로세스 언어로 매핑하는 논문에서는 직접적으로 매핑되는 요소만을 제시하였다. 하지만 본 논문에서 제안한 루프 기법, 식별화 기법, 순서화 기법은 서로 매핑되지 않는 요소에 대해서도 매핑을 가능하게 한다. 즉, BPMN의 복합적 비즈니스 엔터티를 단순한 비즈니스 엔터티들의 집합으로 재구성함으로써, BPMN의 매핑되지 않는 요소들을 매핑가능하게 하는 것이다.

본 연구는 아직까지 이행적 매핑으로 재구성된 비즈니스 프로세스가 기존 프로세스와 동일한 의미를 갖고 있다는 것을 증명하지 못했다. 그래서 향후, π-calculus[19]을 바탕으로 본 연구의 의미 기반 비즈니스 프로세스 매핑 기법을 연구할 것이다. 그리고 XPDL뿐만 아니라 BPML이나 BPEL4WS과 같은 다른 비즈니스 프로세스 언어에 대해서도 BPMN으로부터의 매핑을 제시할 것이다.

참 고 문 헌

[1] BPMI.org, "Business Process Modeling Notation (BPMN) Version 1.0," May 3, 2004.

[2] WfMC, "Workflow Process Definition Language - XML Process Definition Language," Document Number WFMC-TC-1025, Documentation Status - 1.0 Final Draft, October 25, 2002, Version 1.0.

[3] Stephen A. White., "XPDL AND BPMN," Workflow Handbook 2003.

[4] Jiang, P., Q. Mair, et al., "Using UML to Design Distributed Collaboration Workflows: from UML to XPDL," Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), Linz, Austria, 2003.

[5] R. Eshuis, P. Brimont, E. Dubois, B. Grégoire, S. Ramel, "Animating ebXML Transactions with a Workflow Engine," CoopIS 2003, Catania, Italy, Springer, 2003.

[6] Gardner, T., "UML Modelling of Automated Business Processes with a Mapping BPEL4WS," European Workshop on Object Orientation and Web Services, Darmstadt, Germany, 2003.

[7] Workflow Patterns Home Page. http://www.workflowpatterns.com.

[8] Stephen A. White., "Process Modeling Notations and Workflow Patterns," Workflow Handbook 2004.

[9] W.M.P. van der Alast., "Patterns and XPDL: A Critical Evaluation of the XML Process Definition Language"

[10] P. Wohed., W.M.P. van der Aalst., M. Dumas. and A.H.M. ter Hofstede., "Pattern Based Anaysis of BPEL4WS," QUT Technical report, FIT-TR-2002-04, Queensland University of Technology, Brisbane, 2002.

[11] W.M.P. van der Aalst., M.Dumas., A.H.M. ter Hofstede. and P. Wohed., "Pattern Based Anaysis of BPML (and WSCI)," QUT Technical report, FIT-TR-2002-04, Queensland University of Techniology, Brisbane, 2002.

[12] Robert Shapiro, "A Technical Comparison of XPDL, BPML and BPEL4WS," 2002.

[13] Han, K. and Kim, K., "Development of an XPDL-Based Workflow Management System Using the Light-Weight Component Structure," IE Interfaces, Vol.17, No.2, pp.190- 199, 2004.

[14] Xiao Ying, Chen Deren and Chen Min, "Research of Web Services Workflow and its Key Technology Based on XPDL," IEEE International Conference on Systems, pp. 2137-2142, 2004.

[15] Ping Jiang, Quentin Mair and Julian Newman, "Using UML to Design Distributed Collaborative Workflows from UML to XPDL," WETICE 2003, pp.71-76, 2003.

[16] Enhydra Shark: Open Source XPDL Workflow Engine in Java (http://shark.objectweb.org/)

[17] Adam, N., Atluri, V. and Huang, W., "Modeling and Analysis of Workflows using Petri Nets," Journal of Intelligent Information Systems, Vol.10, No.2, pp.131 - 158, 1998.

[18] WfMC, "Wf-XML 2.0 XML Based Protocol for Run-Time Integration of Process Engines," Documentation Status - 2.0 Draft, October 8, 2004, Version 2.0.

[19] Robin Milner., "A Calculus of Mobile Processes, Part I," September, 1990.

[20] Han, K. and Hwang, T., "An UML/XML-Based Business Process Definition Tool," IE Interfaces, Vol.16, No.2, pp. 156-166, 2003.

박 정 업

e-mail : jupark@cse.hanyang.ac.kr
2005년 한양대학교 전자컴퓨터공학부 (학사)
2005년~현재 한양대학교 컴퓨터공학과 석사과정
관심분야 : e-비즈니스, 그리드 컴퓨팅, 센서 네트워크

정 문 영

e-mail : myjeong@cse.hanyang.ac.kr
2003년 강남대학교 무역학과(학사)
2005년 한양대학교 컴퓨터공학과(석사)
2005년~현재 미라콤아이앤씨 근무
관심분야 : e-비즈니스, 시멘틱 웹, XML, 데이터베이스

조 명 현

e-mail : mhjo@cse.hanyang.ac.kr
2004년 한양대학교 전자컴퓨터공학부 (학사)
2006년 한양대학교 컴퓨터공학과(석사)
2006년~현재 코난테크놀로지 근무
관심분야 : 그리드 컴퓨팅, 센서 네트워크, 시멘틱 웹, e-비즈니스

김 학 수

e-mail : hagsoo@cse.hanyang.ac.kr
2004년 한양대학교 전자컴퓨터공학부(학사)
2006년 한양대학교 컴퓨터공학과(석사)
2006년~현재 한양대학교 컴퓨터공학과 박사과정
관심분야 : 그리드 컴퓨팅, XQuery, 시멘틱 웹, XML, 데이터베이스

손 진 현

e-mail : jhson@cse.hanyang.ac.kr
1996년 서강대학교 전산학과(학사)
1998년 한국과학기술원 전산학과(석사)
2001년 한국과학기술원 전자전산학과 (박사)
2001년 9월~2002년 8월 한국과학기술원 전자전산학과 박사후 연구원
2002년 9월~현재 한양대학교 컴퓨터공학과 조교수
관심분야 : 데이터베이스, e-비즈니스, 유비쿼터스 컴퓨팅, 임베디드 시스템