웹서비스 기반의 실행 가능한 비즈니스 프로세스를 위한 BPEL4WS와 BPML간의 상호 변환 알고리즘에 관한 연구

문진영° 송병열 조현규
한국전자통신연구원
{jymoon°, sby, hkcho}@etri.re.kr

Electronics and Telecommunication Research Institute (ETRI)

Jinyoung Moon° Byoungyol Song Hyunkyu Cho
Intelligent Robot Research Division, ETRI

요 약

웹서비스란 XML 포맷의 벤더 중립적인 표준을 따라 웹에 산재한 웹 애플리케이션들을 통합하는 새로운 분산 컴퓨팅 기술이다. 웹서비스를 이용해서 비즈니스 로직을 바탕으로 유용한 서비스를 제공하는 새로운 비즈니스 프로세스를 구현하기 위해서 기존의 웹서비스를 조합하여 비즈니스 프로세스를 기술하는 웹서비스 컴포지션 스펙이 제안되었다. IBM과 Microsoft에서는 BPEL4WS, Sun에서는 WSCI, BPMI.org에서는 BPML을 제안하였는데, 그 중에 BPEL4WS와 BPML은 기업 내부 시스템에서 사용되는 실행 가능한 비즈니스 프로세스를 기술한다.

본 논문에서는 실행 가능한 비즈니스 프로세스를 위한 이 두 스펙간의 상호 운영성을 증대하기 위해서 BPEL4WS와 BPML간의 상호 변환 알고리즘을 제안한다. 예를 들어 BPEL4WS 구현 시스템에서 BPML에서 BPEL4WS로의 변환 알고리즘을 이용하여 BPEL4WS 뿐만 아니라 BPML로 기술된 비즈니스 프로세스를 참조할 수 있고, BPEL4WS에서 BPML로의 변환 알고리즘을 이용하여 BPEL4WS 구현 시스템에서 사용 중인 비즈니스 프로세스 인스턴스를 BPML 프로세스로 출력하여 BPML 구현 시스템에서 사용할 수 있다.

1. 서 론

웹서비스란 초경량의 벤더 독립적인 통신 프로토콜을 이용하여 네트워크 상의 다른 시스템으로부터의 XML 형식의 요청을 받아들이는 웹 애플리케이션이다 [1]. 웹서비스를 재사용 가능한 소프트웨어 컴포넌트로 간주하고 새로운 비즈니스 로직을 기술하는 비즈니스 프로세스를 이 웹서비스를 조합하여 구현하기 위해서 BPEL4WS (Business Process Execution Language for Web Services) [2], WSCI (Web Service Choreography Interface) [3], BPML (Business Process Modeling Language) [4] 과 같은 다양한 웹서비스 컴포지션 스펙이 제안되었다. 그 중에서 실행 가능한 비즈니스 프로세스를 기술하기 위한 스펙은 BPEL4WS와 BPML이다.

BPEL4WS(약자로 BPEL)는 2002년 8월에 IBM, BEA, Microsoft가 함께 만들었다. 원래 IBM의 그래프 기반 WSFL과 Microsoft의 블록 기반 XLANG의 언어적 특징을 합병해서 만든 것으로 실행 가능한 프로세스 뿐만 아니라 프로세스의 인터페이스를 기술하는 추상 프로세스도 지원한다. 2003년 4월에 OASIS에 제출되어 표준화 작업이 진행되고 있다.

BPML은 BPMI.org (Business Process Management Initiative)에 의해 제안된 XML 기반의 비즈니스 프로세스를 모델링하는 메타 언어이다. BPMI.org는 Intalio, SAP, Sun 등이 참여하고 있는데, 실행 프로세스를 기술한다는 점에서 추상 프로세스를 기술하는 WSCI와는 상보적인 관계에 있다 [5].

IBM, Microsoft, Sun과 같은 메이져 벤더들은 각기 다른 기관에서 자신의 스펙에 대한 표준화 작업과 스펙에 기반한 구현 시스템 개발을 활발히 진행 중이다. 그리고 기업 소프트웨어를 개발하는 업체에서는 어느 한 스펙 또는 두 스펙을 다 지원하는 시스템을 출시하고 있다. BPEL4WS와 WSCI/BPML간의 표준화 경쟁은 어느 한쪽이 완전한 우위를 차지하기 전까지 당분간 지속될 전망이다.

이런 상황에서 어느 한 스펙을 따르는 시스템의 상호 운영성을 증대시키기 위해서 본 논문에서는 실행 가능한 프로세스를 위한 BPEL4WS와 BPML간의 상호 변환 알고리즘을 제안한다.

2. 변환 알고리즘

본 장에서는 BPEL과 BPML간의 상호 변환 알고리즘을 제안한다.

먼저 3.1절에서 BPEL에서 BPML로의 변환 알고리즘을 기술한다. 우선 각각의 BPEL의 소스 엘리먼트를 어떤 BPML의 결과 엘리먼트로 변환하는지를 설명한다. 만약 BPEL 엘리먼트에 정확히 대응하는 BPML 엘리먼트가 존재하지 않으면 BPML 엘리먼트를 이용하여 BPEL 엘리먼트와 동일한 효과를 내는 해법을 제안한다. 그러나 BPEL 엘리먼트를 BPML 엘리먼트로 전혀 표현할 수 없는 경우에는 BPEL 소스 엘리먼트의 존재를 BPML 문서내에서 주석으로 처리한다. 따라서 본 논문의 변환 알고리즘은 개별 엘리먼트의 변환보다는 BPEL 형식의 비즈니스 프로세스의 전체 흐름을 BPML 문서에서 나타내는데 초점을 두고 있다. 따라서 BPEL 소스 엘리먼트를 BPML의 어느 엘리먼트를 사용하더라도 나타낼 수 없는 경우에, BPML 문서를 BPEL 문서로 완벽하게 변환하기 위해서는 본 논문 제안한 변환 알고리즘 외에 추가적인 작업이 필요하다. 그리고 3.2절에서는 3.1절과 유사한 방법으로 BPML에서 BPEL로의 변환 알고리즘을 기술한다.

기본적으로 변환 알고리즘을 이해하기 위해서는 BPEL과 BPML의 구성요소와 구조 및 의미를 알아야 하므로 보다 자세히 알고자한다면 스펙 [2]와 [4]를 참조하기 바란다.

2.1 BPEL에서 BPML로의 변환 알고리즘

BPEL 프로세스를 구성하는 기본 단위는 액티버티인데 액티버티는 크게 기본 액티버티와 구조화된 액티버티로 구분된다. 기본 액티버티는 웹서비스 호출, 특정 기간 휴지 상태, 실행중인 액티버티 종료 등을 기술한다. 구조화된 액티버티는 기본 액티버티들과 구조화된 액티버티들의 집합체에 대한 제어 플로우를 기술한다. 따라서 BPEL의 기본 액티버티는 BPML의 단순 액티

버티로 변환하고, BPEL의 구조화된 액티버티는 BPML의 복합 액티버티로 변환한다.

BPEL 기본 액티버티는 표1에 따라 대응하는 BPML 단순 액티버티로 변환한다. BPEL의 receive 액티버티와 reply 액티버티의 복합 구조체는 BPML에서 명명된 프로세스를 호출하는 call 액티버티를 자식 엘리먼트로 가지는 action 엘리먼트로 변환한다. 이 때 명명된 프로세스는 receive와 reply 사이에 위치한 액티버티를 포함한다. 이를 위해서 이 프로세스를 로컬 프로세스로 정의한다. 이 외에 다른 기본 액티버티들은 표 1을 따라 기계적으로 변환 가능하다.

BPEL	BPML
invoke	*action*
receive - reply	*action* *- call*
empty	*empty*
throw	*fault*
compensate	*compensate*
assign	*assign*
wait	*delay*

표 1. BPEL 기본 액티버티와 BPML 단순 액티버티간의 매핑

그리고, BPEL의 구조화된 액티버티는 표 2에 따라 BPML의 복합 액티버티로 변환한다.

BPEL	BPML
flow	*all*
sequence	*sequence*
while	*while*
scope	*call*
(none)	*until*
(none)	*foreach*
switch	*switch*
- case	*- case*
- otherwise	*- default*
pick	*choice*
- onMessage	*- action*
- onAlarm	*- delay*
- (none)	*- synch*

표 2. BPEL 구조화된 액티버티와 BPML 복합 액티버티간의 매핑

그러나, BPEL 엘리먼트가 source나 target 엘리먼트를 가지는 경우에는 BPML 구조를 재구성하는 작업이 추가로 필요하다. 블록 구조에 기반한 BPML과는 대조적으로 BPEL은 모든 액티버티에 flow 액티버티 내부에서 link 엘리먼트에 의해 연결되는 두 액티버티 간의 실행 순서를 규정하는 링크 메카니즘을 제공한다. 링크 메카니즘에 의해서 병렬적으로 수행되는 flow 내부의 액티버티들 가운데 target 엘리먼트를 자식 엘리먼트로 가진 액티버티는 반드시 source 엘리먼트를 자식 엘리먼트로 가진 액티버티 다음에 수행되어야 한다. 링크 메카니즘이 없는 BPML에서 링크 메카니즘을 구현하기 위해서 BPML의 시그널 메카니즘을 이용한다. BPML은 동기화를 위해 signal과 연결된 raise와 synch 액티버티를 제공한다. synch 액티버티는 signal이 raise 액티버티에 의해서 활성화 될 때까지 기다린다. 따라서 source 엘리먼트는 raise 액티버티를, target엘리먼트는 synch 액티버티를 이용한다. target 엘리먼트를 자식 엘리먼트로 가지는 액티버티는 대응하는 source 엘리먼트를 자식 엘리먼트로 가지는 액티버티가 수행 완료될 때까지 기다려야 하기 때문에 target 엘리먼트를 자식 엘리먼트로 가지는 BPEL 엘리먼트에 대응하는 BPML 엘리먼트 앞에 synch 액티버티를 둔다. 그리고, synch 엘리먼트와 변환된 BPML 엘리먼트를 sequence 엘리먼트로 감싸서 이 두 액티버티 간의 순차적 실행을 보장한다. 그리고 source 엘리먼트를 자식 엘리먼트로 가지는 액티버티가 완료된 후에 target 엘리먼트를 자식 엘리먼트로 가지는 액티버티가 시작되어야 하기 때문에 source 엘리먼트를 자식엘리먼트로 가지는 BPEL 엘리먼트에 대응하는 BPML 엘리먼트 뒤에 raise 엘리먼트를 둔다. 그리고 변환된 BPML 엘리먼트와 raise 엘리먼트를 sequence로 감싸서 이 두 액티버티 간의 순차적 실행을 보장한다. 그리고 flow 액티버티 내부의 link 정의는 BPEL의 flow 액티버티에 대응하는 all 액티버티의 context 엘리먼트 내부에 signal 정의로 변환한다.

그림 1은 flow 액티버티 내부에서 souce 엘리먼트와 target 엘리먼트를 자식 엘리먼트로 가진 invoke들이 BPML의 all 액티버티내부에서 어떻게 변환되는지를 보여준다.

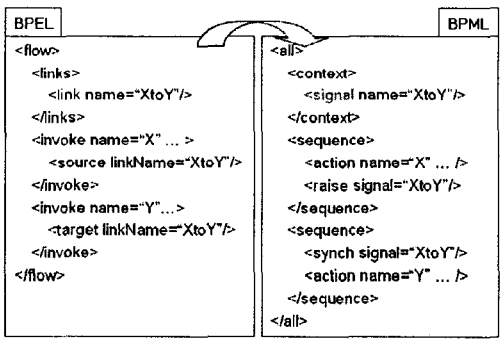

그림 1. source/target 을 가지는 invoke 액티버티의 변환

그리고, 자신만의 실행 환경을 기술하는 BPEL의 scope 액티버티는 실행환경을 기술하는 context 엘리먼트를 가지는 프로세스를 이용하여 변환한다. 이 때, context를 엘리먼트로 가지는 sequence 액티버티를 사용하지 않고 프로세스 메카니즘을 이용하는 것은 보상 메카니즘의 기본 단위가 BPEL에서는 scope이고, BPML에서는 프로세스이기 때문이다.

BPEL	BPML
variable	*property*
eventHandler	
- onMessage	*- exception*
- onAlarm	*- schedule - faluts*
faultHandler	*faults*
compensationHandler	*compensation*

표 3. BPEL의 scope 액티버티의 자식 엘리먼트와 BPML의 context 엘리먼트 사이의 매핑

먼저 scope 액티버티에 대응하는 프로세스를 정의한다. 실행환경을 기술하는 scope 내부의 각 엘리먼트는 표 3에 따라 프로세스의 context 엘리먼트 내부의 엘리먼트로 변환한다.

BPEL의 eventHandler와 faultHandler는 BPML의 것과는 많이 다르다. BPEL에서의 이벤트가 수신 메시지와 알람인 반면에, BPML의 이벤트는 메시지와 시그널이다. BPML에서 시간 관련 이벤트는 따로 schedule 엘리먼트로 정의한다. BPEL과 BPML 간의 의미와 구조의 상이성으로 인해서 변환에 세심한 주의가 요구된다.

다음으로, scope 액티버티 내부에 둔 BPEL 액티버티를 대응되는 BPML 액티버티로 변환한다. scope의 내용을 담은 process는 scope 액티버티의 부모 엘리먼트에 대응되는 부모 process의 context에 로컬 프로세스로 정의한다. 마지막으로 scope 액티버티를 이 프로세스를 호출하는 call 액티버티로 대체한다. 그림 2는 scope 액티버티가 프로세스 정의부와 이 프로세스를 호출하는 call 액티버티로 변환되는 것을 보여준다.

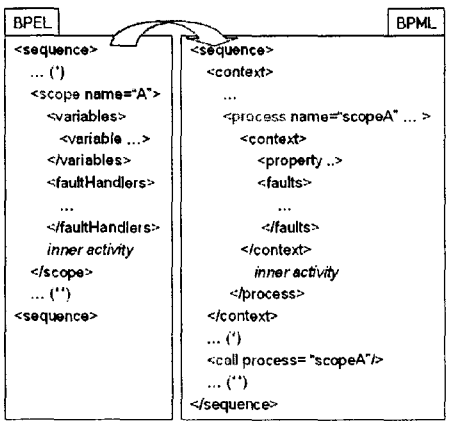

그림 2. scope 액티버티의 변환

3.2 BPML에서 BPEL로의 변환 알고리즘

BPML의 단순 액티버티를 표 1에 따라 대응되는 BPEL 기본 액티버티로 변환할 수 있다. 그리고 foreach와 until 액티버티를 제외한 다른 BPML 복합 액티버티는 표 2에 따라 대응되는 BPEL 구조화된 액티버티로 변환할 수 있다. foreach 와 until 액티버티는 정확히 대응하는 BPEL 엘리먼트가 존재하지 않지만, 이들을 while 액티버티의 변형으로 보고 while 액티버티를 이용하여 변환한다. 그러나 BPEL의 pick 엘리먼트는 시그널 이벤트를 지원하지 않으므로 BPML의 choice 액티버티의 synch 엘리먼트는 대체될 수가 없어서 주석처리로 대신한다.

BPEL에는 프로세스 메카니즘이 존재하지 않으므로, call과 spawn 액티버티는 sequence 액티버티로 대체한다. 물론 이렇게 하면 spawn 액티버티의 병렬적 수행 특성을 반영할 수 없으나 수행 결과는 동일하게 된다.

BPML의 복합 액티버티가 자신만의 로컬 컨텍스트를 가지는 경우 BPEL 구조화된 액티버티로의 변형이 좀 더 복잡해진다. 먼저 BPML 복합 액티버티가 자신의 컨텍스트를 가진 경우에는 BPML 복합 액티버티에 대응하는 BPEL 액티버티를 scope 액티버티로 감싼다. 이는 BPEL에서 scope만이 내부에 자신의 실행 환경을 정의할 수 있기 때문이다. 그리고 표 3에 따라 context 엘리먼트의 내부 엘리먼트를 scope 내부의 실행 환경을 기술하는 각각의 엘리먼트로 변환한다. 그 중에서 process 정의는 context에서 제거되는데 이는 BPEL에서 프로세스를 지원하지 않기 때문이다. 그림 3은 context를 가지는 sequence 액티버티의 변환을 보여준다.

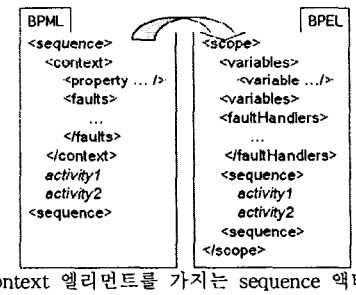

그림 3. context 엘리먼트를 가지는 sequence 액티버티의 변환

3. 결론

실행 가능한 비즈니스 프로세스를 위한 웹서비스 컴포지션 표준이 크게 BPEL과 BPML로 양분되어 있는 상태에서, 본 논문에서 우리는 BPEL과 BPML간의 상호 변환 알고리즘을 제안하였다. BPEL에서 BPML로의 변환 알고리즘은 BPEL 스펙을 따라 기술된 프로세스 인스턴스에 대응하는 BPML 프로세스를 생성하고, BPML에서 BPEL로의 변환 알고리즘은 BPML 형식을 따르는 프로세스 인스턴스에 대응하는 BPEL 프로세스 생성하는 알고리즘이다.

이 상호 변환 알고리즘은 BPEL 과 BPML 구현 시스템 간의 상호 운영성을 증대 시켜줄 것이다. 예를 들어, BPEL 기반의 비즈니스 프로세스 모델러에서는 BPML에서 BPEL로의 변환 알고리즘을 사용하여 BPML 형식의 비즈니스 프로세스를 참조할 수 있다. 그리고 이 모델러에서 BPEL에서 BPML로의 변환 알고리즘을 사용하여 모델링한 프로세스를 BPML 형식으로도 출력할 수 있다.

그러나 이 변환 알고리즘이 BPEL과 BPML간의 효율적인 변환 기술을 제시하나 완전한 자동은 아니다. 왜냐하면, BPEL의 표현력이 BPML의 표현력과 동일하지 않고, 제공하는 메카니즘이 다르기 때문이다. 예를 들어, BPML은 조건 반복문으로 while 이외에도 foreach, until 액티버티를 제공한다. until 액티버티는 기계적인 변환이 가능하나, foreach 액티버티는 BPEL로의 변환을 위해서는 의미를 고려한 재구성이 필요하다. 또 BPEL에서는 시그널 메카니즘이 존재하지 않으므로 BPML의 choice 액티버티의 synch 엘리먼트는 BPEL의 pick 액티버티의 어느 자식 엘리먼트로도 표현될 수 없다. 이런 한계에도 불구하고 본 논문에서 제안한 상호 변환 알고리즘은 소스 비즈니스 프로세스의 전체적인 비즈니스 로직을 제대로 반영한 결과 비즈니스 프로세스를 생성하는 효율적인 변환 기법을 지원한다.

4. 참조

[1] Carlos Valcarcel et al., Introduction to Web services and the WSDK V5.1, September 2003 http://www-106.ibm.com/developerworks/edu/ws-dw-ws-intwsdk51-i.html

[2] Tony Andrew et al., Business Process Execution Language for Web Services (BPEL4WS) version 1.1, 5 May 2003.

[3] Assaf Arkin et al, Web Service Choreography Interface (WSCI) 1.0, W3C Note 8 August 2002.

[4] Assaf arkin et al., Business Process Modeling Language (BPML) BPMI Proposed Recommendation, 24 January 2003.

[5] Web Service Choreography Interface (WSCI) 1.0 FAQs; http://www.sun.com/software/xml/developers/wsci/faq.html