# BPEL to WS-ECA 변환 방법

이원석* · 이규철**

한국전자통신연구원 표준연구센터*

충남대학교 컴퓨터공학과**

## 요 약

최근 유비쿼터스 환경으로 발전하면서 이종의 하드웨어와 소프트웨어 플랫폼을 가지고 있는 디바이스들 간의 연동 기술이 점점 중요해지고 있다. 이에 대한 연구 중의 하나로 웹서비스를 이용한 이벤트 기반 다바이스 연동 프레임워크가 있으며, 이는 XML을 기반으로 이벤트 기반의 다바이스 연동을 정의하는 룰 언어인 WS-ECA를 제안하고 있다. 그러나, 아직까지 WS-ECA를 효과적으로 작성할 수 있는 방법이 제시되지 않고 있다. 따라서, 본 논문은 비즈니스 프로세스에 대한 로직을 표현하는데 가장 활용이 많이 되고 있는 BPEL을 WS-ECA로 변환하는 방법을 제안하며, 이는 BPEL 저작 툴을 이용하여 기본적인 디바이스 연동을 정의할 수 있으므로 효과적으로 WS-ECA을 생성할 수 있는 방법을 제공할 수 있다.

키워드

BPEL, WS-ECA, 디바이스 연동

## Ⅰ. 서 론

최근 컴퓨팅 환경이 수십억 개의 디바이스가 긴밀하게 네트워크로 연결되는 유비쿼터스 환경으로 빠르게 발전하면서, 이종의 하드웨어와 소프트웨어 플랫폼을 가지고 있는 다양한 디바이스들이 상호운용성을 보장하면서 자연스럽게 연동할 수 있는 기술에 대한 요구가 시급히 요구되고 있다. 이와 관련하여, 최근 SOA(Service Oriented Architecture) 개념을 기반으로 하는 웹서비스 기술을 유비쿼터스 환경에 적용하기 위한 연구들이 활발하게 진행되고 있다. 웹서비스 기술은 인터넷 환경을 기반으로 고안된 표준 기술로 인터넷에 존재하는 다양한 서비스들을 서로 공유하여 이용할 수 있는 환경을 제공할 수 있다.

최근에 진행된 연구 중의 하나로 웹서비스를 이용한 이벤트 기반 다바이스 연동 프레임워크[1]가 있다. 이는 XML을 기반으로 이벤트 기반의 다바이스 연동을 정의하는 룰 언어인 WS-ECA를 제안하고 있으며, 유비쿼터스 환경의 디바이스들 간의 연동이 필요할 경우 연동하는 일련의 순서를 정의할 수 있다. 즉, 특정한 목적을 위한 일련의 시나리오를 정의할 수 있으며, 이렇게 정의된 룰은 일련의 절차를 통해 해당 디바이스에 저장된다. 이렇게 저장된 룰은 기본적으로 이벤트를 기반으로 동작하도록 되어 있어, 특정한 이벤트를 받으면, 이를 기반으로 룰에 정의된 동작을 수행한다. 이러한 일련의 방식으로 각각의 디바이스들은 서로의 룰을 수행하며, 필요한 경우 디바이스 간에 이벤트를 주고받으며 연동을 수행하는 구조이다.

그러나, 그러나, 아직까지 XML을 형식으로 표현되는 WS-ECA를 효과적으로 작성할 수 있는 방법에 아직까지 제안된 방법이 없다. 본 논문에서는 기존의 웹서비스 환경에서 비즈니스 응용간의 프로세스를 정의하고 실행하는 언어인 WS-BPEL(Business Process Execution Language for Web Services)[4]을 이용하여 정의된 내용을 WS-ECA로 변환하는 방법을 제안한다. WS-BPEL은 현재 많은 제품에 적용되어 있으며, 프로세스를 저작하기 위한 GUI 기반의 훌륭한 저작도구가 개발되어 있어 사용자는 쉽게 이러한 툴을 이용하여 원하는 프로세스의 저작 및 검증을 수행할 수 있다. 따라서, WS-BPEL을 이용하여 디바이스 간의 연동에 대한 정의를 하고, 이를 기반으로 WS-ECA의 표현 방식으로 변환 할 수 있다면 이 방법을 통해 WS-ECA 저작 방법은 획기적으로

개선될 수 있다.

2장에서는 WS-BPEL과 WS-ECA 비교 및 변환 방법에 대해서 설명하고, 3장에서 결론 및 향후 연구 방향에 대해서 설명한다.

## II. WS-BPEL과 WS-ECA 비교 및 변환 방법

### 2.1 WS-BPEL

WS-BPEL에 대한 배경을 보면, 이 언어는 XML과 웹 서비스를 기반으로 하는 표준으로, SOAP, WSDL, UDDI, WS-Reliable Messaging, WS-Addressing, WS-Coordination, WS-Transaction 등을 포함하는 웹 서비스 테크놀로지 스택을 지원하기 위해 XML 기반 언어를 사용한다[2].

WS-BPEL 프로세스는, 프로세스에 참여하는 웹 서비스들의 정확한 호출 순서(순차적 또는 병렬적)를 정의하며, 이는 조건부 실행문을 지원한다. 예를 들어, 이전 호출 결과값에 따라 웹 서비스의 호출 여부를 결정할 수 있으며, 이를 위해 루프를 정의하고, 변수를 선언하고, 변수 값을 복제/할당하고, 에러 핸들러를 정의할 수 있다. 이러한 요소들을 조합하여 알고리즘에 기반한 복잡한 비즈니스 프로세스를 정의할 수 있으며, 실제로 대부분의 비즈니스 프로세스가 다양한 액티비티(activity)를 포함하는 그래프 형태로 표현될 수 있다[2].

WS-BPEL은 다음과 같은 구조를 갖는다.

```
<process>
<partnerLinks>
</partnerLinks>

<variables>
</variables>

<sequence>
</sequence>

<eventHandlers>
</eventHandlers>

</process>
```

### 2.2 WS-ECA

ECA 기반의 룰 기술 언어는 웹서비스를 복합해서 활용하는 과정을 이벤트(event), 조건(condition), 액션(action)으로 구분하여 사용자가 원하는 방식으로 웹서비스를 복합적으로 활용할 수 있는 방법을 XML을 이용하여 기술하도록 하는 언어이다. ECA 기반의 룰 기술 언어는 기본적으로 다음과 같은 포맷을 갖는다.

```
on event
if condition
do action
```

즉, 특정한 event가 발생하면 condition을 검사하고, 이 조건을 만족하는 경우 action을 수행 하도록 한다.

WS-ECA는 다음과 같은 구조를 갖는다.

```
<ECARule name="ncname"
targetNamespace="xsd:anyURI"
issuedDate="xsd:dateTime"
expiredDate="xsd:anyURI">

 <types>···</types>
 <variables>···</variables>

 <events>···</events>
 <actions>···</actions>

 <rules>
  <rule>
   <documentation>···</documentation>
   <event>···</event>
   <condition>···</condition>
   <action>···</action>
  </rule>
  ...
 </rules>
<ECARule>
```

ECARule은 ECA 기반의 룰 기술 언어의 루트 엘리먼트로 name, targetNamespace, issuedDate, expiredDate 속성을 갖는다. name 속성은 ECA 룰 문서를 나타내기 위한 이름을 가지며, targetNamespace는 정의된 ECA 룰 문서가 갖게 되는 네임스페이스를 지정한다. issuedDate와 expiredDate는 각각 ECA 룰 문서가 효력을 발휘하기 시작하는 시각과 끝나는 시각을 지정한다. issuedDate와 expiredDate 두 속성의 값이 모두 없으면 그 룰 문서는 영원히 유효하며, issuedDate만 있으면 issuedDate 이후로 영원히 유효하고, expiredDate만 있으면 '현재'로부터 expiredDate 까지만 유효하다.

ECA 기반의 룰 기술 언어에는 ECA 룰을 정의하는데 필요한 변수의 타입(types), 변수(variables), 이벤트(events), 액션(actions)을 정의하는 부분과 ECA 룰을 정의하는 부분(rules)으로 나뉜다.

### 2.3 WS-BPEL과 WS-ECA 간의 비교

WS-BPEL로 정의된 내용은 기본적으로 WS-ECA로 표현이 가능하지만, 몇 가지 고려해야 할 사항이 있다.

첫째는 WS-BPEL의 eventHandler에서는 전달받는 이벤트 하나 만을 표현할 수 있지만, WS-ECA에서는 여러 개의 이벤트에 대한 표현이 가능하며, 또한 이들 간의 논리적인 연산을 통해 여러 가지 조건에 대한 표현이 가능하다.

둘째는 WS-ECA는 내부적인 기능을 호출할 때도 내부 이벤트를 전달하는 형태로 표현하도록 되어 있다. 따라서, 이에 대한 정보가 WS-BPEL에서 표현 가능한 부분에 대한 정보의 매핑을 고려해야한다.

셋째는 기본적으로 표현하는 마크업 이름과 전체적인 구조가 다르다. WS-BPEL은 WS-ECA 보다 훨씬 풍부한 표현 방법을 가지고 있기 때문에 이러한 부분을 WS-ECA의 구조로 변환하는 것이 구조적인 단순화와 마크업 간의 매핑을 통해 변환이 되어야 한다.

넷째는 WS-BPEL은 일반적인 웹서비스를 기반으로 비즈니스 응용간의 연동을 정의 및 실행을 하는 언어이므로, 기본적으로 인터페이스 정보는 WSDL을 활용한다. 그러나 WS-ECA의 경우 이벤트를 주고 받는 방법으로 연동을 하기 때문에, 기본적인 인터페이스가 이벤트를 받고, 이벤트를 전달하는 형식을 사용 한다.

### 2.4 WS-BPEL과 WS-ECA 간의 변환 방법

WS-BPEL과 WS-ECA간의 효율적인 변환을 위해서는 기본적으로 WS-BPEL을 작성할 때 WS-ECA 표현을 고려하는 것이 필요하다. 즉, WS-BPEL은 비동기적인 이벤트를 지원하는 eventHander를 지원하고 있으므로 WS-ECA와 흡사한 형태로 정의할 수 있다.

WS-BPEL과 WS-ECA 간의 이벤트 표현의 차이는 WS-BPEL에서 조건 표현을 할 때 약간의 수정을 통해서 WS-ECA와 같은 의미를 표현할 수 있는 방법이 있다. 즉, e1, e2, e3, e4의 이벤트를 기반으로 하는 아래와 같은 WS-ECA 룰이 있다고 가정해 보자.

```
e1∧e2∧e3∧e4 [condition a] {do a}
```

위의 WS-ECA 룰에 대한 표현을 WS-BPEL에서 아래와 같은 형태를 통해서 표현할 수 있다. 즉, 하나의 WS-BPEL은 이벤트 만을 표현할 수 있기 나머지 이벤트에 대한 표현은 e1 이후의 조건을 무조건 TRUE로 다음 액션을 수행할 수 있도록 하여, 액션 부분에서 나머지 이벤트를 모두 받은 후에 원하는 로직을 표현한다.

```
e1 [TRUE] {receive e2 || e3 || e4;
        if condition a
        then do a;}
```

## III 결론

본 논문에서는 최근의 진행되고 있는 웹서비스를 이용한 이벤트 기반 다바이스 연동 프레임워크에서 사용하는 WS-ECA를 효과적 하기 위한 방법으로, 비즈니스 프로세스에 대한 로직을 표현하는데 가장 활용이 많이 되고 있는 WS-BPEL을 WS-ECA로 변환하는 방법을 제안하였다. 기본적으로 WS-BPEL과 WS-ECA 간의 구조적인 차이점을 비교하고 이를 기반으로 변환 방법에 대한 내용을 설명하였다. 향후에는 유비쿼터스 환경에서의 비즈니스 도메인 간의 연동을 고려하여 WS-BPEL과 WS-ECA 간의 연동에 대한 연구가 필요하다.

## 참고문헌


[1] Kangchan Lee, Wonsuk Lee, Jonghong Jeon, Seungyun Lee, Jonghun Park, "Event-driven Coordination Rule of Web Services enabled Devices in Uniquitous environments", W3C Ubiquitous Web Workshop, March 2006.

[2] Matjaz B. Juric, "A Hands-on Introduction to BPEL", Oracle Technology and network, 2006.

[3] M. Cilia and A. Buchmann. An Active Functionality Service for E-Business Applications. ACM SIGMOD Record, 31(1): 24-30, 2002.

[4] OASIS, "Web Services Business Process Execution Language Version 2.0", April, 2007.

[5] S. Vinoski. Integration with Web Services. IEEE internet computing, 7(6): 75-77, 2003.

[6] W3C, "Web Services Description Language(WSDL) Version 1.2", W3C Working Draft, March 3, 2003.