Vol.4, No.5 pp.209~218 pISSN: 2287-5905 http://dx.doi.org/10.3745/KTSDE.2015.4.5.209

BPEL Engine Generator for adding New Functions to BPEL based on Attribute Grammar and Aspect-Oriented Programming

Dongkyu Kwak† · Jongho Kim†† · Jaeyoung Choi†††

ABSTRACT

BPEL is used in various domains since it can describe the flow of works according to conditions and rules, and it can call web services in service-oriented computing environments. However, new functions that are not provided by BPEL grammar are required in specific domains. Generally, when new functions are required, the domain-specific language should be newly defined and developed, which requires high development cost. In this regard, a new function needs to be defined and added instead of developing domain-specific language with the new functions added. However, such methods only allow an addition of a single function, and it is difficult to design and add new functions according to the needs. This paper defines XAS4B document, which extends the BPEL grammar function through XML schema in order to add new functions, and proposes BPEL engine generator that generates BPEL engine with the new functions added by processing the document. The XAS4B document enables the creation of a new grammar added to BPEL using XML schema. It also shows the process of adding new functions to BPEL engine using AspectJ, JAVA implementation of aspect-oriented programming. The proposed system can add new functions using AspectJ without modifying BPEL engine. This allows the provision of new functions at low cost in various domains.

Keywords : Aspect-Oriented Programming, Attribute Grammar, B2J, AspectJ, BPEL, Workflow

속성문법과 관점지향 프로그래밍 기법을 이용한 BPEL에 새로운 기능을 추가하는 BPEL 엔진 생성기

곽 동 규† · 김 종 호†† · 최 재 영†††

요 약

BPEL은 서비스 지향 컴퓨팅 환경에서 조건에 따른 작업의 흐름과 웹 서비스의 호출을 기술할 수 있어 다양한 도메인에서 사용되고 있다. 하지만 특정 도메인에서는 BPEL 문법에 없는 새로운 기능이 요구된다. 일반적으로 기존 언어에 없는 새로운 기능을 추가한 경우에 도메인 특화 언어를 새롭게 정의하고 개발해야 하는데, 이를 위해서는 많은 개발 비용이 소요된다. 따라서 새로운 기능이 추가된 도메인 특화 언어를 개발하는 대신에 새로운 기능을 추가하여 사용해야 한다. 그러나 이 방법들은 단일 기능을 추가할 수 있을 뿐이고, 필요에 따라 새로운 기능을 설계하고 추가하기 어렵다. 본 논문에서는 필요에 따라 새로운 기능을 추가하기 위해 XML 스키마를 통해 BPEL의 문법적 기능을 확장할 수 있는 XAS4B 문서를 정의하고, 이 문서를 처리하여 기능이 추가된 BPEL 엔진을 생성하는 BPEL 엔진 생성기를 제안한다. XAS4B 문서는 BPEL에 추가되는 문법을 XML 스키마로 작성하고 추가된 문법의 기능을 JAVA 프로그램으로 작성할 수 있도록 한다. 그리고 추가된 기능을 관점지향 프로그래밍의 JAVA 구현체인 AspectJ를 이용하여 새로운 기능의 처리 모듈을 BPEL 엔진에 추가하는 방법을 보인다. 제안하는 시스템은 AspectJ를 이용하여 BPEL 엔진을 수정하지 않고 새로운 기능을 추가할 수 있으며, 요구되는 새로운 기능에 대해 동일한 방법을 사용하여 손쉽게 추가할 수 있으므로, 다양한 분야에서 적은 비용으로 새로운 기능을 제공할 수 있다.

키워드 : 관점지향 프로그래밍, 속성문법, B2J, AspectJ, BPEL, 워크플로우

1. 서 론

BPEL[1]은 작업의 흐름을 기술하기 위해서 웹 서비스를 호출하고 또한 조건에 따른 작업의 흐름을 기술할 수 있는데, 이를 특정 도메인에 적용하여 사용할 경우에는 추가적인 기능이 요구될 수 있다. 예를 들어, 복잡한 조건을 추상화시킨 비즈니스 규칙 엔진(Business Rule Engine)[2]이 필요한 경우나 엔진이 실행되고 있는 환경에서의 응용 프로그램을 호출하는 경우에는 표준 그대로의 BPEL을 사용하기가 용이하지 않다. BPEL에 새로운 기능을 추가하기 위한 연구로는 JWX(Java Weaving XML) 문서를 이용하여 기능을 추가하는 방법이 연구되었다[3]. JWX는 BPEL에 추가적으

※ 이 논문은 2013년도 정부(교육부)의 재원으로 한국연구재단의 기초연구사업지원을 받아 수행된 것임(2013R1A1A2012118).
† 준 회 원 : 숭실대학교 컴퓨터학부 연구원
†† 정 회 원 : 숭실대학교 IT유통물류학과 석사과정
††† 종신회원 : 숭실대학교 컴퓨터학부 교수
Manuscript Received : December 4, 2014
First Revision : February 5, 2015; Second Revision : March 2, 2015
Accepted : March 2, 2015
* Corresponding Author : Jaeyoung Choi(choi@ssu.ac.kr)

로 요구되는 기능을 JWX 문서에 JAVA 프로그래밍 언어로 기술하도록 제안하였으며, BPEL과 함께 JWX에 기술된 추가적인 프로그램을 함께 실행한다. JWX 시스템은 BPEL 엔진으로 B2J(BPEL to JAVA)[4]를 사용하는데, 이 엔진은 BPEL 문서를 JAVA 프로그램으로 변환하고 이 변환된 프로그램을 실행하는 방법을 사용한다. JWX 시스템은 B2J가 생성한 JAVA 프로그램과 JWX 문서에 작성된 추가된 프로그램을 함께 실행하기 위해 관점 지향 프로그래밍 기법[5]의 구현체인 AspectJ[6]를 이용하여 직조(Weaving)하여 실행시킨다. JWX는 BPEL이 제공하지 않는 기능을 추가할 수 있는 방법을 제공하고 있으나, BPEL과 함께 작성해야 하는 새로운 기능을 JAVA 프로그램으로 기술해야 하는 단점을 가지고 있다. 이로 인하여 컴퓨터 프로그래밍에 능숙하지 않은 도메인 전문가가 사용하기는 어렵다.

본 논문은 XML 기반의 BPEL 언어에 새로운 기능에 해당하는 언어를 확장하고 확장된 언어의 기능을 처리하는 소프트웨어를 BPEL 엔진에 추가하는 방법을 연구한다. 새로운 언어를 개발하기 위해서는 언어의 문법과 의미로 나누어 설계하고 개발해야 한다. 일반적으로 언어의 문법적 요소는 BNF(Backus-Naur Form)[7]로 다룰 수 있고, 언어의 의미적 요소는 속성문법(Attribute Grammar)[8]으로 표현할 수 있다. 이렇게 표현된 언어의 문법과 의미는 자동화된 도구인 YACC(Yet Another Compiler Compiler)[9]를 이용하여 컴파일러나 언어 처리기를 생성할 수 있다. 그래서 YACC는 새로운 언어를 디자인하여 설계하고 컴파일러나 언어 처리기를 개발하는 데 많이 이용되고 있다. 하지만 이미 설계되어 개발이 완료된 언어를 확장하기는 어렵다. 즉 YACC를 이용하여 BPEL을 확장하기 위해서는 BPEL 언어에 새로운 기능에 해당하는 문법을 추가하여 다시 개발해야 한다. 이를 위하여 본 시스템에서는 BPEL에 새로운 기능을 추가하기 위해서 XML 스키마를 확장하여 BPEL에 요구되는 새로운 문법과 기능을 XML 스키마와 속성문법으로 작성할 수 있는 XAS4B(XML Attribute Schema for BPEL)를 사용한다. 프로그래머는 새로운 기능의 문법적 요소를 XML 스키마로 작성하고, 기능적 요소를 속성문법의 형태로 작성한다.

XAS4B는 XML의 문법 정보를 작성하는 XML 스키마와 함께 사용된다. XAS4B는 두 부분으로 구성되어있는데, 한 부분은 추가 기능에 필요한 클래스를 임포트(import) 하는 부분이고, 다른 한 부분은 의미 정보를 기술하는 부분이다. 의미 정보를 기술하는 부분은 새로운 기능을 기술하는 XML의 변수를 정의하고 속성문법을 이용하여 JAVA 프로그램을 기술한다. 속성문법을 다루는 방법은 YACC 시스템과 유사하다. 개발자는 YACC를 이용하여 언어 처리기나 컴파일러를 개발할 때, 언어의 문법을 BNF로 설계하고 언어의 의미는 YACC 문법을 이용하여 프로그램을 작성한다. 하지만 본 시스템은 YACC와 다르게 독립적으로 동작하는 새로운 언어를 디자인하는 것이 아니라, BPEL과 함께 동작하는 언어를 설계하는 데 사용한다.

본 논문에서 제안하는 시스템은 프로그래머, 도메인 전문가, 그리고 최종 사용자의 3개 계층으로 구분된 사용자 계층을 갖는다. 프로그래머는 도메인에서 요구되는 새로운 기능의 문법과 기능을 XAS4B 문서를 이용하여 정의한다. XAS4B 문서는 본 시스템의 컴파일러 생성기를 통해 새로운 기능을 추가하기 위한 컴파일러를 생성한다. 그리고 도메인 전문가는 해당 도메인에서 요구되는 비즈니스 흐름을 BPEL 문서로 기술한다. 이때, 프로그래머가 새로운 기능을 정의한 엘리먼트를 이용한다. 도메인 전문가가 작성한 BPEL 문서와 새로운 기능 엘리먼트는 B2J 코디네이터(B2J Coordinator : BPEL 문서를 JAVA 프로그램으로 변환하는 루틴)와 새로운 기능 컴파일러에 의해 각각 JAVA 프로그램과 AspectJ 프로그램으로 변환된다. 이 두 프로그램은 관점지향 프로그래밍 기법의 직조를 통해 단일 모듈로 동작하여 최종 사용자에게 도메인 전문가가 기술한 비즈니스 흐름에 따른 서비스를 제공한다. XAS4B 문서는 프로그래밍 언어에 능숙한 프로그래밍 개발자가 작성하고, XAS4B 문서 내에서 새로운 기능에 해당하는 엘리먼트는 XML로 정의되어 있어서, 프로그래밍 언어에 능숙하지 않은 도메인 전문가도 새로운 기능을 손쉽게 사용할 수 있다.

본 논문은 2절에서 관련 연구를 소개하고 3절에서 추가 기능을 기술하기 위한 XAS4B를 소개한다. 그리고 4절에서는 본 시스템의 구조를 제안하고 5절에서는 제안하는 시스템을 이용한 규칙 시스템에 관해 논한다. 또한 6절에서는 규칙 기능이 추가된 BPEL을 이용한 급여 시스템을 보인 후, 7절에서 결론을 맺는다.

2. 관련 연구

본 논문에서는 관점지향 프로그래밍 기술과 속성문법을 이용하여 BPEL에 새로운 기능을 추가하는 방법을 보인다. BPEL에 기능을 추가하는 대표적인 유사 연구 사례는 두 가지가 있다. 한 가지는 BPEL의 기능을 그대로 이용하면서 비즈니스 규칙을 추가하여 사용한 연구이고, 다른 한 가지는 관점지향 프로그래밍 기법을 적용하여 BPEL을 확장한 연구이다. 첫 번째 연구 사례로 Rosenberg는 BPEL에 규칙을 추가한 BRIB(Business Rule Integration in BPEL)를 제안하였다[10]. Rosenberg는 BPEL 문법을 그대로 이용하면서 규칙을 적용하기 위해 BPEL 엔진이 웹 서비스를 호출하는 'invoke'를 인터셉트하여 규칙 엔진을 호출하는 방법을 사용하였다.

BRIB는 BPEL 문법을 수정하지 않고 규칙 맵을 통해 BPEL 문서와 규칙의 재사용성을 높이고 다양한 BPEL 엔진에 적용할 수 있다는 장점을 가진다. 하지만 규칙 엔진의 호출이 'invoke'에만 국한되어있고 규칙에 비교 대상을 'invoke'의 인자 외에는 사용할 수 없기 때문에 규칙 엔진을 한정적으로 제한한다. 또한 규칙을 적용할 때 BPEL과 무관하게 invoke에만 적용되어, BPEL 문서와 규칙 간의 관계가 직관적으로 드러나지 않는다. 그리고 이 방법은 규칙이라는 단일 기능을 BPEL에 적용하였을 뿐 다른 기능을 추가하기 위해서는 다른 방법을 사용해야 한다.

```
<aspect name="Counting">
    <partnerLinks>
        <partnerLink name="javaExecWSLink" ... />
    </partnerLinks>
    <variables><variable name="invokeMethodRequest" ... /></variables>
    <pointcutandadvice type="after">
        <pointcut name="Luftansa Invocations">
            //process//invoke[@portType="LufthansaPT and @operation="searchFight"]
        </pointcut>
        <advice>
            <sequence>
                <assign>
                    <copy>
                        <from>increaseCounter</from>
                        <to variable="invokeMethodRequest" part="methodName" />
                    </copy>
                </assign>
                <invoke partnerLink="javaExecWSLink" portType="JavaExePT"
                        operation="invokeMethod" inputVariable="invokeMethodRequest" />
            </sequence>
        </advice>
    </pointcutandadvice>
</aspect>
```

Fig. 1. AO4BPEL의 에스팩트

다른 대표적인 연구 사례로서, 관점지향 프로그래밍 기법을 BPEL에 적용한 AO4BPEL(Aspect Oriented for BPEL) [11]은 BPEL에서 제기된 모듈화의 어려움을 해결하고 서비스를 동적으로 지원한다. BPEL에 관점지향 프로그래밍 기법을 적용함으로써 여러 위치에 필요로 하는 요구사항을 횡단관심사로 모듈화하는 방법을 제안하였다. Fig. 1은 AO4BPEL에서 횡단관심사를 작성하는 에스팩트(Aspect)를 보인다.

Fig. 1과 같이 AO4BPEL의 에스펙트는 관점지향 프로그래밍 기법의 포인트 컷(pointcut)과 어드바이스(advice)를 제공하고 있어 BPEL 문서를 관점지향 프로그래밍 기법으로 작성할 수 있다. 포인트 컷은 어드바이스가 실행되는 위치이고 어드바이스는 포인트 컷에서 실행될 프로그램이다. 또한 포인트 컷에 BPEL의 문서 위치와 함께 실행조건과 실행문을 기술할 수 있다. 하지만 AO4BPEL은 BPEL에 관점지향 기능을 추가하는 방법을 제공할 뿐이고 다른 기능을 추가할 수 없다.

서비스를 추상적으로 작성하여 사용한다면, 도메인에 따라 추가된 새로운 기능을 도메인 전문가가 손쉽게 사용할 수 있다. 이를 위하여 다수의 연구자들이 도메인에 따른 언어를 설계하고 이를 개발하는 방법에 관한 연구가 진행되었다[12, 13]. Simon[12]은 스마트 홈에서 편재형 컴퓨팅 서비스(Pervasive Computing Service)를 제공하기 위한 언어를 제안하였다. 일반적으로 서비스 지향 구조에서는 웹 서비스를 기반으로 BPEL을 사용하는데, BPEL을 이용하여 웹 서비스를 호출하기 위해서는 BPEL 언어의 웹 서비스 인터페이스 기능을 학습해야 한다. Simon은 SOAL(Service Oriented Architecture Language)을 제안하였으며, 이를 BPEL과 WSDL로 변환하여 시스템을 제안하였다. 이 방법은 JAVA와 유사한 SOAL 언어를 제안하여 손쉽게 사용할 수 있다. 하지만 SOAL은 프로그래밍 언어에 능숙하지 않은 도메인 전문가에게는 BPEL보다 사용하기가 어려울 수 있고, 또 다른 새로운 기능을 추가해야 할 경우에 적용하기가 용이하지 않다.

3. 추가 기능 기술을 위한 XAS4B

본 논문은 BPEL에 추가적인 기능이 요구될 때 추가 기능을 XML 스키마로 정의하여 BPEL 문법을 확장하고, 확장된 문법의 프로그램을 작성하는 문서를 제안하고 이를 위한 처리 엔진을 보인다. 본 시스템은 세 개의 사용자 계층으로 구분되어 있는데, Table 1은 제안하는 시스템의 사용자 계층에 따른 요구 사항과 개발 범위, 그리고 사용 도구를 보여준다.

Table 1. Requirements, Development Scope, and Used Tools for 3 User Layers

	요구 사항	개발 범위	사용 도구
프로그래머	추가 기능	프로그램 모듈	XAS4B JAVA 프로그램
도메인 전문가	비즈니스 흐름	워크플로우	BPEL 언어 (추가 엘리먼트)
최종 사용자	사용자 요구사항	-	BPEL 엔진

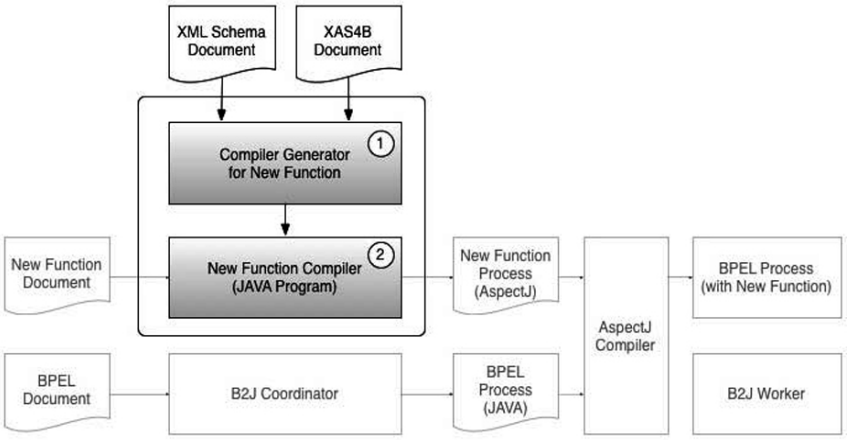


Fig. 3. A Process of Generating Engine

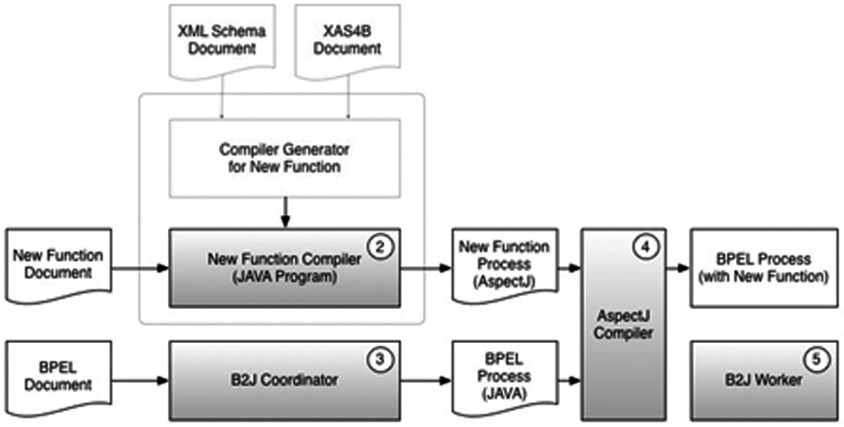


Fig. 4. Execution Environment of the Proposed System

Table 1과 같이 본 시스템의 사용자는 세 계층으로 구분된다. 프로그래머는 새롭게 추가될 기능을 개발하는 계층으로 요구되는 기능을 기술하기 위한 엘리먼트를 XAS4B 문서로 정의하고, 이를 처리하는 JAVA 프로그램을 개발한다. 그리고 도메인 전문가는 해당 도메인에서 요구되는 비즈니스 흐름을 BPEL 문서로 기술한다. 이를 위하여 프로그래머가 새로운 기능을 추가하기 위해 개발한 엘리먼트를 이용하여 기술한다. 최종 사용자는 도메인 전문가가 기술한 비즈니스 흐름에 따라 서비스를 제공받는다. 본 논문에서는 추가 기능을 정의하는 방법으로 XML 스키마와 함께 추가된 기능의 프로그램을 기술할 수 있는 XAS4B를 제안한다. Fig. 2는 XAS4B의 스키마를 보인다.

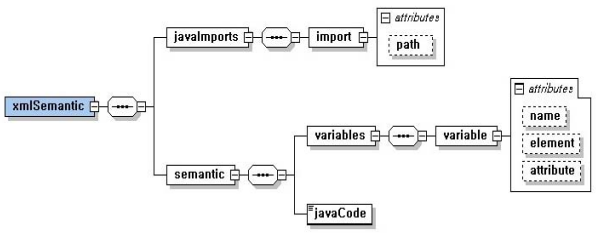


Fig. 2. Schema of XAS4B

Fig. 2에서 보듯이 XAS4B는 두 부분으로 구성되어 있다. 한 부분은 “javaImports”로 추가 기능에서 요구되는 JAVA 클래스를 임포트(import) 할 수 있도록 제공한다. 그리고 다른 한 부분은 “semantic”으로 의미 정보를 기술하는 부분인데, 이 부분은 XML 스키마에서 속성문법을 통해 의미 정보를 다루고자 하는 XML 속성을 지정하는 “variables” 엘리먼트와 새로운 기능을 프로그램으로 작성할 수 있는 “javaCode” 엘리먼트로 구성된다.

4. 시스템 구조

본 논문에서는 도메인에 따라 요구되는 BPEL의 새로운 기능을 적은 비용으로 추가하기 위한 시스템을 제안한다. 프로그래머는 새로운 기능을 직관적으로 사용하기 위한 XML 스키마를 기술하고 새로운 기능을 XAS4B 문서를 이용하여 엘리먼트의 의미 정보를 작성한다. 프로그래머를 통해 만들어진 새로운 기능의 문법은 프로그램을 잘 이해하지 못하는 도메인 전문가가 서비스를 작성하기 위해 사용되고, 이 서비스는 사용자에게 제공된다.

프로그래머가 작성한 새로운 기능의 문법 정보가 있는

XML 스키마와 의미 정보가 있는 XAS4B 문서는 컴파일러 생성기를 통해 컴파일러가 생성되고, 이 컴파일러 생성기는 B2J 코디네이터와 결합되어 BPEL 기능과 새로운 기능을 실행하는 엔진의 일부가 된다. 그리고 도메인 전문가가 작성한 새로운 기능을 포함한 BPEL 문서를 실행한다. 본 시스템에서는 B2J(BPEL to JAVA)를 이용한다. B2J는 BPEL 문서를 JAVA 프로그램으로 변환하여 실행하여 BPEL 문서의 서비스 흐름을 실행시킨다. 제안하는 시스템은 B2J에 의해 생성된 프로그램에 관점지향 프로그래밍 기법을 이용하여 새로운 기능을 추가하는 방법을 사용한다. 제안하는 시스템은 새로운 기능을 위한 엔진을 생성하는 과정과 새로운 기능이 추가된 워크플로우를 실행하는 과정으로 이루어진다.

Fig. 3은 제안하는 시스템의 엔진 생성을 보여주고 있다.

프로그래머는 도메인의 요구사항에 따라 BPEL에 추가될 새로운 XML 문법을 XML 스키마로 정의한다. 그리고 문법에 해당하는 기능을 XAS4B 문서로 작성한다. 컴파일러 생성기(① Compiler Generator for New Function)는 새로운 기능을 정의한 XML 스키마와 XAS4B 문서를 입력으로 받아 새로운 기능을 처리하기 위한 엔진(② New Function Compiler)을 생성한다. 이와 같이 생성된 엔진은 기존의 BPEL 엔진과 함께 실행된다. Fig. 3은 제안하는 시스템의 실행 환경을 보여준다.

Fig. 4와 같이 컴파일러 생성기를 통해 생성된 엔진(②)은 새로운 기능에 해당하는 문서의 부분을 AspectJ 프로그램으로 변환한다. 그리고 BPEL 문서는 B2J 엔진의 JAVA 프로그램 생성기인 B2J 코디네이터(③ B2J Coordinator)에 의해 JAVA 프로그램으로 변환된다. 이렇게 생성된 AspectJ 프로그램과 JAVA 프로그램은 AspectJ 컴파일러(④ AspectJ Compiler)로 직조되어 B2J Worker(⑤ B2J Worker : B2J 코디네이터가 생성한 JAVA 프로그램을 실행하는 루틴)에서 실행된다.

컴파일러 생성기(①)는 XML 스키마 파서와 XAS4B 파서로 구성되어 있는데, XML 스키마 파서는 새로운 기능의 문법 정보를 분석하고 XAS4B 파서는 추가된 기능의 위치를 분석하여 삽입 규칙을 생성한다. 생성된 컴파일러(②)는 추가된 기능의 문법 정보와 삽입 규칙을 통해 새로운 문법으로 작성된 새로운 기능 문서(New Function Document)를 처리하는 AspectJ 프로그램을 생성한다. 서비스를 위해 작성된 새로운 기능을 포함한 BPEL 문서는 프로그램으로 변환되고 이 프로그램은 사용자에게 서비스를 제공한다. 이와 같은 시스템 구성은 BPEL 문서를 분석하여 생성한 프로그램을 실행시키는 구조로 서비스 제공을 위한 엔진을 경량화한다.

5. 규칙 기능이 추가된 BPEL

본 연구는 BPEL의 응용에 따라 요구되는 도메인 특화 언어를 BPEL에 확장하기 위해서 XML 스키마와 함께 사용하여 확장된 언어의 기능을 기술할 수 있는 XAS4B를 제안하고 이를 처리할 수 있는 시스템을 개발한다. 본 절에서는 제안하는 시스템의 응용으로 규칙 기능이 추가된 BPEL 시스템을 보인다.

BPEL은 XML 기반으로 그래픽 편집기를 제공하고 있으므로 컴퓨터 프로그래밍 언어에 대한 이해도가 낮은 도메인 전문가도 쉽게 워크플로우를 작성할 수 있다. 하지만 복잡하고 복합적인 조건을 처리하기 위해서는 복잡한 조건문을 BPEL을 이용하여 기술해야 하는데, 조건이 복잡할수록 기술하기 어려워지고 사용자의 의도와 다르게 동작할 가능성이 높아지게 된다. 일반적으로 복잡한 조건을 처리하기 위한 방법으로는 규칙 엔진[2]을 사용할 수 있지만 BPEL은 규칙 기능을 제공하지 않는다. 규칙은 복잡하고 복합적인 조건을 단순화하고 조건을 충족할 때에 실행할 수 있는 동작을 기술하여 서비스를 모델링할 수 있다. Fig. 5는 규칙이 추가된 BPEL의 흐름을 보인다.

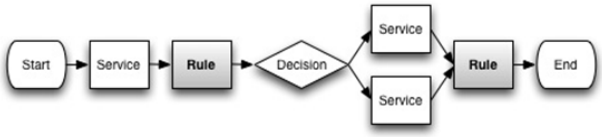


Fig. 5. BPEL Flow Added with Rules

Fig. 5와 같이 BPEL에 추가된 규칙 기능은 BPEL 흐름 사이에 작성하여 복잡한 조건을 단순화하는 효과를 가진다. 본 논문에서는 BPEL의 기능을 그대로 유지하면서 규칙을 적용할 수 있도록 BPEL 언어를 확장한다. 규칙을 사용하기 위해서는 규칙 엔진이 필요한데, 본 연구에서는 규칙 엔진으로 Drools[2]를 사용한다. 규칙을 BPEL 언어에 확장하기 위해서는, 규칙을 위한 엘리먼트를 XML 스키마로 문법적으로 확장해야 하고 XAS4B 문서로 기능을 정의해야 한다. XML 스키마와 XAS4B 문서는 컴파일러 생성기를 통해 컴파일러를 생성한다. 생성한 컴파일러는 B2J 엔진과 함께 결합하여 규칙 기반 BPEL 엔진으로 사용한다. Fig. 6은 규칙을 사용하기 위한 엘리먼트 중 Drools의 규칙을 사용하기 위한 “drools-rule-set” 엘리먼트와 규칙을 실행시키기 위한 “rule-executetion” 엘리먼트의 XML 스키마이고, Fig. 7은 XAS4B 문서이다.

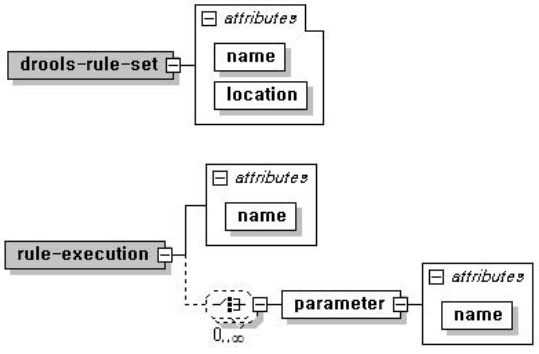


Fig. 6. XML Schema for Rules

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <xas4b:xmlSemantic targetNamespace="http://coolman.org/rule4b" xmlns:xas4b="http://coolman.org/xas4b"
03     xmlns:rule4b="http://coolman.org/rule4b" >
04     <xas4b:javaImports>
05         <xas4b:javaImport path="org.drools.FactException" />
06     </xas4b:javaImports>
07  ...
08     <xas4b:semantic>
09         <xas4b:variables>
10             <xas4b:variable name="nameV" element="drools-rule-set" attribute="name" />
11             <xas4b:variable name="locationV" element="drools-rule-set" attribute="location" />
12         </xas4b:variables>
13         <xas4b:javaCode><![CDATA[
14             RuleBase ruleBase = RuleBaseLoaderFromUrl
15             (RuleExample.class.getResource("$locationV"));
16            WorkingMemory workingMemory = ruleBase.newWorkingMemory();
17         ]]></xas4b:javaCode>
18     </xas4b:semantic>
19     <xas4b:semantic>
20         <xas4b:variables>
21             <xas4b:variable name="ruleNameV" element="rule-execution" attribute="name" />
22             <xas4b:variable name="ruleParamV" element="parameter" attribute="name" />
23         </xas4b:variables>
24         <xas4b:javaCode><![CDATA[
25             RunnerInterface engine = ((JEngineProgram_...)thisJoinPoint.getTarget()).engine;
26             RuleBase ruleBase = RuleBaseLoader.getRuleBase();
27             WorkingMemory workingMemory = ruleBase.newWorkingMemory();
28             workingMemory.assertObject($ruleParamV);
29             workingMemory.fireAllRules();
30          ]]></xas4b:javaCode>
31     </xas4b:semantic>
32 </xas4b:xmlSemantic>
```

Fig. 7. XAS4B Document for Rules

Fig. 7에서 10~11줄, 그리고 21~22줄은 Fig. 6의 XML 스키마에서 속성을 다루기 위한 변수들이고 이 변수들은 "$locationV"와 "$ruleParamV"와 같이 JAVA 프로그램과 함께 쓸 수 있다. 이런 방법은 YACC에서 속성문법을 다루는 방법과 유사하다. Fig. 3에서 보듯이 컴파일러 생성기는 XML 스키마와 XAS4B를 입력으로 하여 규칙 기능을 위한 새로운 컴파일러를 생성하였다.

생성된 컴파일러는 Fig. 4와 같은 실행환경에서 동작하는데, Fig. 8은 Fig. 6과 Fig. 7에서 보인 규칙을 위해 추가된 엘리먼트를 사용하여 작성한 간단한 규칙 기반의 BPEL 예제로서, 사용자의 이름을 조건으로 하여 텍스트를 출력한다.

Fig. 8에서 12줄은 규칙 문서를 등록하고 있다. 규칙은 조건에 따른 실행을 의미하고 규칙 문서는 규칙을 정의한 문서이다. 등록하고 있는 규칙 문서는 사용자의 이름에 따라 텍스트를 출력하는 간단한 규칙이다. 본 연구에서의 규칙 문서는 Drools 규칙 엔진[2]의 규칙 문서의 스키마를 사용하고, 이 문서에는 "Hello World"와 "Goodbye World"의 두 규칙을 정의하고 있는데, 정의된 규칙은 "rule-execution"에서 "rule" 속성에 기술하는 이름을 통해 실행할 수 있다. 그리고 14~16줄은 "Hello World" 규칙을 실행시키고 있고, 18~20줄은 "Goodbye World"를 실행시키고 있다. 이와 같은 규칙을 사용하면 조건과 조건에 따른 실행을 규칙 이름으로 추상화하여 단순화시킬 수 있으므로 복잡한 비즈니스 모델도 손쉽게 기술할 수 있다. Fig. 8의 규칙 기반 BPEL은 기존의 BPEL 부분과 "bpel" 접두어(prefix)를 갖는 BPEL 기능과 "rule4b" 접두어를 갖는 추가된 규칙 기능으로 구성되어 있다. 그중 규칙 기능은 AspectJ 프로그램으로 변환된다.

한편, Fig. 4에서 보듯이 B2J 코디네이터는 BPEL 문서를 입력으로 받아 생성한 JAVA 프로그램을 생성한다. 그리고 AspectJ 컴파일러는 이 JAVA 프로그램과 변환된 AspectJ 프로그램을 직조하여 새로운 기능이 포함된 BPEL 타깃 프로그램을 생성한다. Fig. 9는 Fig. 8의 14~16줄에 작성되어 있는 "Hello World" 규칙을 변환한 AspectJ 프로그램이다. Fig. 9에서 5줄은 관점지향 프로그래밍의 "pointcut"으로서 추가 프로그램의 삽입 위치를 나타낸다. 그리고 7~16줄은 "advice"로서 삽입되어 실행될 추가 프로그램이다.

```
01 <bpel:process name="ruleTest" ... >
02 ...
03     <bpel:variables>
04         <bpel:variable name="helloText" messageType="xsd:string">
05             <bpel:from>Kwak</bpel:from>
06         </bpel:variable>
07         <bpel:variable name="byeText" messageType="xsd:string">
08             <bpel:from>Choi</bpel:from>
09         </bpel:variable>
10     </bpel:variables>
11      ...
12     <rule4b:drools-rule-set name="hello-rule" location="http://coolman.org/rules/hello.drl" />
13     ...
14     <rule4b:rule-execution rule="Hello World">
15         <rule4b:parameter name="$helloText" />
16     </rule4b:rule-execution>
17     ...
18     <rule4b:rule-execution rule="Goodbye World">
19         <rule4b:parameter name="$byeText" />
20     </rule4b:rule-execution>
21     ...
22 </bpel>
```

Fig. 8. Example of Rule-based BPEL Document

```
01 import org.eclipse.stp.b2j.core.jengine.internal.message.Message;
02 import org.eclipse.stp.b2j.core.jengine.internal.core....;
03 import coolman.rule.exam;
04 public aspect ActivityAspect{
05     pointcut activityPoint() : execution( void EngineProgram_20131020174101425.activity20*();
06         after() : activityPoint(){
07             try{
08                 RunnerInterface engine = ((JEngineProgram_...)thisJoinPoint.getTarget()).engine;
09                 RuleBase ruleBase = RuleBaseLoader.getRuleBase();
10                 WorkingMemory workingMemory = ruleBase.newWorkingMemory();
11                 workingMemory.assertObject("Hello");
12                 workingMemory.fireAllRules();
13                 workingMemory = ruleBase.newWorkingMemory();
14                 workingMemory.assertObject("Goodbye");
15                 workingMemory.fireAllRules();
16             }catch(Exception e){
17         }
18     }
19 }
```

Fig. 9. AspectJ Program that Converted the Rule of Hello World

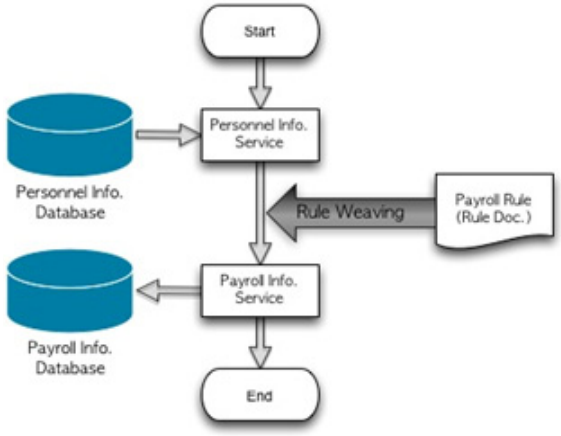


Fig. 10. 급여 시스템의 시나리오

본 절에서는 제안하는 시스템을 이용하여 BPEL에 규칙 기능을 추가하는 시스템을 보였다. 제안하는 시스템은 BPEL 엔진으로 B2J를 사용하는데, B2J는 BPEL 문서를 JAVA 프로그램으로 변환하여 이 JAVA 프로그램을 실행한다. 그리고 AspectJ는 JAVA 프로그램을 수정하지 않고 프로그램을 추가할 수 있는 방법을 제공한다. 규칙을 실행하는 Fig. 9의 AspectJ 프로그램은 B2J가 생성한 JAVA 프로그램과 직조되어 BPEL의 흐름에 추가되어 실행된다. 이와 같은 규칙 기반 BPEL은 복잡한 규칙을 추상화할 수 있고 규칙 식별자를 가지고 있으므로, 규칙의 생성과 적용, 삭제 등을 손쉽게 변경할 수 있다.

6. 규칙 기반 BPEL 시스템의 응용 사례

본 논문은 BPEL의 문법을 필요한 기능에 따라 확장하고 적용할 수 있는 엔진을 소개하고 BPEL에 규칙을 확장하는 방법에 대해 보였다. 규칙 기반 BPEL은 복잡한 규칙을 단순하게 작성할 수 있으므로, 이에 대한 연구가 진행되었다[14, 15]. 규칙 기반 BPEL[14]에서는 규칙 기반 BPEL을 제안하였고, 이를 이용한 인사정보 BPEL 시스템을 예로 보였다. 하지만 규칙 기반 BPEL 시스템은 BPEL에 규칙 기능만을 추가할 수 있어, 규칙이 아닌 다른 기능의 확장으로는 사용하기 어렵다.

본 절에서는 규칙 기반 BPEL을 활용한 예로 규칙 기반 BPEL에서 보인 인사정보를 이용한 급여 시스템을 보인다. 급여 시스템은 인사정보 데이터베이스로부터 인사정보를 받아 급여에 관한 규칙을 적용하여 급여 정보 데이터베이스에 저장한다. 이때 5절에서 보인 규칙 기능을 이용하여, 급여 규칙을 적용한다. Fig. 10은 제안하는 급여 시스템의 시나리오를 보여주고 있다.

Fig. 10에서 인사정보 데이터베이스는 기업의 인사정보를 저장하고 있으며, 인사정보 서비스는 데이터베이스로부터

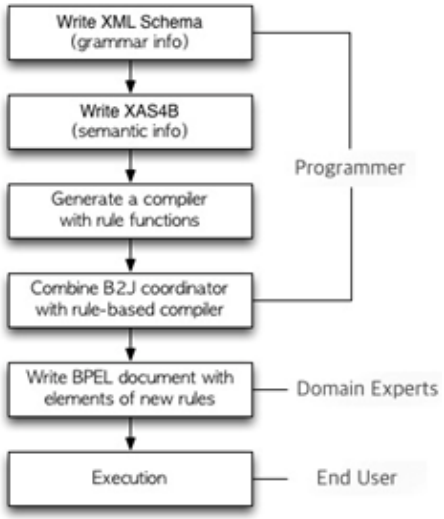


Fig. 11. 제안하는 시스템을 이용한 규칙 시스템과 급여 시스템 개발 과정

인사정보를 출력한다. 그리고 인사정보에 따라 급여지급 규칙이 적용되는데, 이 규칙은 새롭게 확장된 rule4b에 작성되어있고, 제안하는 시스템에 의해 직조되고 적용된다.

제안하는 시스템은 관점지향 프로그래밍을 이용한 직조 방법과 급여 시스템을 위한 규칙 정보를 규칙 기반 BPEL과 동일하게 사용한다. 규칙 문서의 변환 과정과 직조 과정은 규칙 기반 BPEL에서 확인할 수 있다[14]. 규칙 기반 BPEL 연구는 BPEL에 필요한 규칙 기능을 추가하기 위해 규칙 문서를 제안하였고, 규칙을 처리하는 기능을 기존의 BPEL 엔진에 추가하였다. 하지만 규칙이 아닌 다른 기능을 같은 방법으로 BPEL에 추가하기 위해서는 관점지향 프로그래밍 기법과 BPEL 엔진에 대해 잘 이해하고 있는 프로그래머가 규칙 기반 BPEL의 방법대로 BPEL 엔진을 직접 수정해야 한다. 본 논문은 속성문법을 이용한 XAS4B를 제안하여 직접 BPEL 엔진을 수정해야 하는 비용을 절감한다.

Fig. 11은 본 절에서 보이는 급여 시스템을 개발하는 과정을 보여준다. 프로그래머는 XML 스키마와 XAS4B 문서를 작성하고 컴파일러 생성기를 통해 규칙 기능 컴파일러를 생성한다. 이렇게 생성된 규칙 기능 컴파일러는 기존의 BPEL 시스템과 결합하여 규칙 기능을 제공하는 BPEL 시스템으로 동작한다. 도메인 전문가는 BPEL에 추가된 규칙 기능을 이용하여 규칙 문서와 BPEL 문서를 작성한다. 이 때, 규칙 문서와 BPEL 문서는 급여 시스템의 요구사항으로 작성한다. 규칙 문서와 BPEL 문서는 각각 규칙 기능 컴파일러와 B2J 코디네이터에 규칙을 처리하는 AspectJ 프로그램과 JAVA 프로그램으로 변환된다. AspectJ 컴파일러는 이 변환된 프로그램들을 직조하여 규칙 기능이 포함된 BPEL 프로세스를 생성하며, 이 프로세스는 B2J Worker에서 실행된다.

본 논문에서 제안하는 시스템은 프로그래머와 도메인 전

문가로 구분하여 시스템을 개발할 수 있도록 하고 있다. 예를 들어, 급여 시스템 예제에서 보듯이 프로그래머가 급여 규칙을 잘 이해하기는 어렵다. 그러므로 규칙을 도입하여 급여 규칙을 잘 이해하고 있는 도메인 전문가가 급여 시스템을 쉽게 개발할 수 있도록 BPEL에 기능을 추가할 수 있는 환경을 제공한다. 이후 급여 규칙이 변경되더라도 도메인 전문가는 프로그래머의 도움 없이 손쉽게 급여 규칙을 변경할 수 있다. 또한, 이렇게 개발된 규칙 기능은 다른 도메인에서도 재사용할 수 있다.

7. 결 론

워크플로우의 표준인 BPEL은 프로그래밍 언어에 대한 이해도가 낮은 도메인 전문가도 손쉽게 비즈니스 흐름을 작성할 수 있다. 하지만 특정 도메인에서는 BPEL 문법에 없는 새로운 기능이 요구된다. BPEL 엔진이 새로운 기능을 처리하기 위해서는 새로운 BPEL 엔진을 개발하거나 기존의 BPEL 엔진에 기능을 추가해야 하는데, 이를 위해서는 많은 비용이 요구된다. 본 논문에서는 XML 스키마를 확장하여 XAS4B 문서를 제안하고 관점지향 프로그래밍 기법을 이용하여 BPEL 엔진에 새로운 기능을 추가하는 방법을 보인다. 관점지향 프로그래밍 기법은 직조를 이용하여 핵심관심사를 수정하지 않고 횡단관심사를 추가할 수 있다. 본 연구에서는 기존의 BPEL 엔진으로 B2J를 사용하는데, 이 엔진은 BPEL 문서를 JAVA 프로그램으로 변환하고 이를 실행한다. 본 시스템은 B2J 엔진이 생성한 JAVA 프로그램을 핵심관심사로 두고, 새로운 기능을 처리하는 프로그램을 횡단관심사로 하여, 두 프로그램을 직조함으로써 BPEL 흐름에 새로운 기능의 흐름을 추가한다.

제안하는 시스템은 프로그래머, 도메인 전문가, 그리고 최종 사용자의 3개 계층으로 구분된 사용자 계층을 갖는다. 프로그래머는 도메인에서 요구되는 새로운 기능의 문법과 기능을 XAS4B 문서를 이용하여 정의한다. XAS4B 문서는 본 시스템의 컴파일러 생성기를 통해 새로운 기능을 추가하기 위한 컴파일러를 생성한다. 그리고 도메인 전문가는 해당 도메인에서 요구되는 비즈니스 흐름을 새로운 기능과 BPEL을 이용하여 기술한다. 이렇게 작성된 비즈니스 흐름은 최종 사용자에게 서비스로 제공된다. 본 시스템은 새로운 문법과 기능을 잘 설계하고 개발하였을 경우에, 기존 BPEL 엔진에 손쉽게 추가할 수 있는 방법을 제공하고 있다. 즉, 추상적으로 잘 정의된 문법을 설계하고 기능을 개발하는 부분은 프로그래머의 능력에 해당하는 부분이다. 또한 프로그래머가 요구되는 기능을 사용하는 도메인을 잘 이해해야만 좋은 문법과 기능을 설계하고 개발할 수 있다. 프로그램 개발자에 의해 잘 설계되고 개발된 좋은 문법과 기능은 한 번의 개발로 도메인 전문가에게 추상적인 문법을 제공하여, BPEL을 이용한 워크플로우 문서를 작성하기 손쉽고 이해하기 쉽게 한다.

References

[1] BPEL [Internet], http://www-128.ibm.com/developerworks/library/specification/library/specification/ws-bpel/

[2] Drools [Internet], http://www.jboss.org/drools/

[3] Donggyu Kwak, Jaeyoung Choi, “Design and Implementation of a BPEL Engine for Dynamic Function using Aspect-Oriented Programming (in Korean),” *Journal of Korean Institute of Information Scientists and Engineers*, Vol.37, No.4, pp.205-214, Aug., 2010.

[4] B2J [Internet], http://www.eclipse.org/stp/b2j/

[5] GregorKiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Videira Lopes, Jean-Marc Loingtier, and John Irwin, “Aspect-Oriented Programming,” ECOOP, pp.220-242, 1997.

[6] Eclipse AspectJ [Internet], http://www.eclipse.org/aspectj/

[7] Comsky Noam, “Syntactic Structures,” Feb., 1957.

[8] Donald Ervin Knuth, “The Genesis of Attribute Grammars,” *Proceedings of international conference on Attribute grammars and applications*, Vol.461, pp.1-12, 1990.

[9] John Levine, Tony Mason, and Doug Brown, “lex&yacc,” 1992.

[10] F. Rosenberg, S. Dustdar, “Business Rules Integration in BPEL-A Service-Oriented Approach,” In Proceedings of the 7th International IEEE Conference on E-Commerce Technology (CEC 2005), 2005.

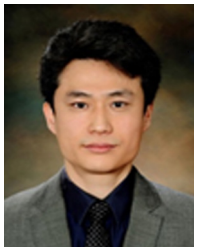
[11] AnisCharfi, Mira Mezini, “Aspect-Oriented Web Service Composition with AO4BPEL,” *Lecture Notes In Computer Science 2004*, Vol.3250, pp.168-182, 2004.

[12] Balazs Simon, Balazs Goldschmidt, and KarolyKondorosi, “A Human Readable Platform Independent Domain Specific Language for BPEL,” *Communications in Computer and Information Science*, Vol.87, pp.537-544, 2010.

[13] AbdaladhemAlbreshne, AyoubAitLahcen, and Jacques Pasquier, “A Framework and its Associated Process-Oriented Domain Specific Language for Managing Smart Residential Environments,” *International Journal of Smart Home*, Vol.7, No.6, pp.377-392, 2013.

[14] Donggyu Kwak, Jaeyoung Choi, and Chae-Woo Yoo, “Rule-based BPEL System using Aspect Oriented Programming,” *Journal of Korea Institute of Scientists and Engineers*, Vol. 39, No.2, pp.153-161, Feb., 2012.

[15] F. Rosenberg, S. Dustdar, “Business Rules Integration in BPEL-A Service-Oriented Approach,” In Proceedings of the 7th International IEEE Conference on E-Commerce Technology (CEC 2005), 2005.



곽 동 규

e-mail : kwak.coolman@gmail.com
2002년 서경대학교 응용수학과(학사)
2004년 숭실대학교 컴퓨터학과(석사)
2012년 숭실대학교 컴퓨터학부(박사)
2012년~2014년 (주)스카이컴 부설 연구소
2014년~현 재 숭실대학교 컴퓨터학부 연구원
관심분야 : 프로그래밍언어, 컴파일러, XML

김 종 호

e-mail : jongho.kim@ssu.ac.kr
2015년 숭실대학교 컴퓨터학부(학사)
2015년~현 재 숭실대학교 IT유통물류학과 석사과정
관심분야 : 분산/컴퓨팅, 시스템소프트웨어

최 재 영

e-mail : choi@ssu.ac.kr
1984년 서울대학교 제어계측공학과(학사)
1986년 미국 남가주대학교 전기·전자공학과 (석사)
1991년 미국 코넬대학교 전기·전자공학부 (박사)
1992년~1994년 미국 국립오크리지연구소 연구원
1994년~1995년 미국 테네시주립대학교 연구교수
1995년~현 재 숭실대학교 컴퓨터학부 교수
관심분야 : 시스템소프트웨어, 병렬/분산처리, 고성능컴퓨팅(HPC)