동적 기능 추가를 위하여 관점지향 프로그래밍 기법을 이용한 BPEL 엔진의 설계와 구현

(Design and Implementation of a BPEL Engine for Dynamic Function using Aspect-Oriented Programming)

곽 동 규 † 최 재 영 ††
(Donggyu Kwak) (Jaeyoung Choi)

요 약 BPEL은 웹 서비스와 상호작용하는 워크플로우 언어의 표준으로서 다양한 응용에서 사용되고 있다. 하지만 특정 응용에서는 BPEL에 없는 추가적인 기능이 요구되어 적용하기 어렵다. 본 논문은 관점지향(aspect-oriented) 프로그래밍 기법을 이용하여 BPEL 엔진에 새로운 기능을 추가할 수 있는 시스템을 보인다. BPEL에 새로운 기능을 동적으로 추가하기 위해 새로운 기능을 기술할 수 있는 JWX 문서를 정의하고, 이를 BPEL에 적용하기 위한 관점지향 프로그래밍 기법의 시스템을 제안한다. JWX 문서는 BPEL 문서에 새로운 기능을 추가하기 위하여 자바 프로그램을 기술할 수 있는 XML 기반의 문서이다. 관점지향 프로그래밍 기법은 핵심적인 요구사항과 부가적인 요구사항 사이의 낮은 결합도를 보장하고 있으므로 기존의 프로그램을 수정하지 않고 기능을 추가할 수 있다. 또한 본 시스템은 B2J라는 BPEL 엔진을 관점지향 프로그래밍 기법을 통해 확장하여, 자바 프로그램과 JWX 문서의 새로운 기능 프로그램을 직조하고 실행시킨다. 이 방법을 이용하면, B2J가 제공하는 BPEL의 기능은 그대로 사용하면서 새로운 기능만 개발하여 적은 노력과 비용으로 기능이 추가된 BPEL 엔진을 개발할 수 있다. 본 시스템은 현재 BPEL이 제공하고 있지 않은 조건을 처리할 수 있는 규칙엔진을 BPEL에 추가하거나, 상황인지 환경에 사용할 수 있는 BPEL 워크플로우 시스템을 연구하는데 이용할 수 있다.

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

Abstract BPEL is a standard workflow language, which interacts with Web Services and is used in various applications. But it is difficult to use BPEL for specific applications which require additional functions. In this paper, we present a system which can add new functions to BPEL based on an aspect-oriented programming (AOP) technique. In order to add new functions to BPEL, we define a JWX document format that can describe new functions to apply to BPEL. JWX is XML-oriented document that can code the corresponding Java program in order to dynamically add new functions to BPEL documents. It is possible for BPEL workflow to add new functions without modifying the existing programs using the AOP technique, which guarantees low degree of coupling between key and additional requirements. Additionally this systems weaves based on new functions of Java program and JWX document by expanding BPEL engine called B2J based on AOP and execute them. Therefore it is possible to develop a new BPEL engine with additional functions easily and with low cost. The new system can execute additional conditions that the current BPEL engine doesn't provide. The new system using functions of BPEL supplied by B2J. The new system can be used to add a new rule engine, which isn't currently provided.

Key words : AOP (Aspect-Oriented Programming), workflow, BPEL, B2J, AspectJ

· 본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT 연구센터지원산업의 연구결과로 수행되었습니다(NIPA-2009-(C1090-0902-0007))

† 학생회원 : 숭실대학교 컴퓨터학과
coolman@ss.ssu.ac.kr
†† 종신회원 : 숭실대학교 컴퓨터학과 교수
choi@ssu.ac.kr

논문접수 : 2009년 12월 23일
심사완료 : 2010년 4월 27일

Copyright©2010 한국정보과학회 : 개인 목적이나 교육 목적인 경우, 이 저작물의 전체 또는 일부에 대한 복사본 혹은 디지털 사본의 제작을 허가합니다. 이 때, 사본은 상업적 수단으로 사용할 수 없으며 첫 페이지에 본 문구와 출처를 반드시 명시해야 합니다. 이 외의 목적으로 복제, 배포, 출판, 전송 등 모든 유형의 사용행위를 하는 경우에는 사전에 허가를 얻고 비용을 지불해야 합니다.

정보과학회논문지: 시스템 및 이론 제37권 제4호(2010.8)

1. 서 론

BPEL은 웹 서비스 환경에서 비즈니스 프로세스를 정의하고 실행하는 표준 워크플로우 언어이다[1]. BPEL은 XML을 기반으로 하고 있어 학습이 용이하고 다양한 그래픽 기반의 편집기[2]가 있어 컴퓨터 언어에 대한 이해도가 낮은 도메인 전문가도 쉽게 워크플로우를 작성할 수 있는 장점을 갖는다. 주로 비즈니스 모델을 기술하는 BPEL은 웹 기반의 비즈니스 모델과 공장 자동화 시스템에 도입되어 사용되고 있다.

BPEL은 작업의 흐름을 기술하기 위해서 웹 서비스를 호출과 조건에 따른 작업의 흐름을 기술할 수 있는데 특정 도메인에 적용할 경우에 새로운 기능을 요구하게 된다. 예를 들어 복잡한 조건을 추상화시킨 비즈니스 규칙 엔진(Business Rule Engine)[3]이 필요한 경우나 엔진이 실행되고 있는 환경에서의 응용 프로그램을 호출하는 경우에는 표준 그대로의 BPEL을 사용하기는 어렵다. BPEL의 기능을 그대로 유지하면서 비즈니스 규칙 엔진을 추가하기 위한 대표적인 연구자로는 *Florian* Rosenberg가 있다[4]. BPEL은 "invoke"를 이용하여 서비스를 호출하는데, Rosenberg는 invoke의 이전과 이후에 적용할 규칙을 기술하고 규칙 처리 서비스(Rule Interceptor Service)를 이용하는 방법을 제안하였다. 이 방법은 기존의 BPEL 엔진을 그대로 사용할 수 있는 장점을 갖는다. 하지만 규칙으로 생성된 값을 BPEL의 분기나 다른 서비스의 입력 값으로 사용하기 어렵다. 이는 규칙을 한정적으로 사용하도록 제한한다. 특정 기능이 요구되는 다른 예로는 BPEL 문서의 프로파일링 엔진이 있다. Mingjie Sun 연구자는 실행 시간에 BPEL의 모니터링이 필요함을 보이고, 이를 위한 엔진을 연구하였다[5]. Sun 연구자는 기존의 BPEL 엔진에 관점지향 프로그래밍 기법[6]을 이용하여 BPEL 진행(trace) 경로 로그를 생성할 수 있는 추가하는 방법을 고안하였다. 그리고 생성된 로그 정보를 이용하여 UML 기반의 모니터링 결과를 생성하는 BPEL 모니터링 프레임워크를 개발하였다. BPEL 모니터링 프레임워크는 기존의 BPEL 엔진을 수정하지 않고 진행 경로 로그를 생성하는 새로운 기능을 추가한다. 하지만 다른 새로운 기능이 요구에 대해서는 고려하고 있지 않다.

본 논문은 관점지향 프로그래밍 기법을 이용하여 BPEL에 새로운 기능을 동적으로 추가하기 위한 방법을 제안한다. 이를 위해 새로운 기능과 함께 이를 추가할 BPEL 문서의 위치를 포함하는 JWX(Java Weaving XML)를 정의한다. 그리고 관점지향 프로그래밍 기법을 이용하여 JWX의 새로운 기능을 BPEL에 추가하는 엔진을 구현한다. BPEL 엔진으로 B2J(BPEL to Java)[7]를 이용한다. B2J는 BPEL 문서를 입력으로 받아서, 분석하고 파싱하여 자바 소스 파일로 변환한다. 변환된 자바 소스 파일은 컴파일하여 자바 클래스를 생성하고, 이를 실행하여 서비스를 제공한다. 제안하는 시스템은 B2J가 생성한 BPEL 변환 자바 프로그램을 핵심관심사[1]로 두고, 추가되는 다른 기능을 횡단관심사[2]로 적용한다. 관점지향 프로그래밍 기법은 핵심관심사와 횡단관심사 사이의 낮은 결합도를 보장한다. 그러므로 기존의 프로그램을 핵심관심사로, 새로운 기능을 횡단관심사로 설계하여 개발하면 기존 프로그램에 영향을 주지 않고 새로운 기능이 추가된 엔진을 개발할 수 있다.

제안하는 시스템은 B2J 프로그램을 거의 수정하지 않고 새로운 기능을 추가할 수 있으므로, 아주 적은 비용으로 BPEL이 제공하지 않는 새로운 기능을 동적으로 추가할 수 있다. 본 연구는 이후 BPEL이 제공하고 있지 않는 규칙 엔진을 추가하거나, 온톨로지 기반의 상황 인지를 지원하는 워크플로우 시스템[8]을 개발하는데 유용하게 활용할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서 관련 연구를 소개하고, 3장에서 시스템 모델에 관해 설명하고, 4장에서 시스템 구조를 보인 후 5장에서 활용 사례에 관해 논의하고, 6장에서 결론에 관해 논한다.

2. 관련 연구

본 연구의 목적은 관점지향 프로그래밍을 이용하여 새로운 기능을 BPEL에 추가하는데 있다. 본 장에서는 BPEL에 새로운 기능을 추가한 사례와 관점지향 프로그래밍을 설명하고, 본 연구에서 사용한 B2J 엔진에 관해 설명한다.

2.1 유사 연구 사례

Mingjie Sun 연구자는 워크플로우 시스템의 복잡도가 증가함에 따라, 워크플로우 개발에서 BPEL 문서의 작성과 서비스 설정에 BPEL의 실시간 모니터링이 요구됨을 지적하였다. 그리고 BPEL을 실시간으로 모니터링하기 위해 BPEL 모니터링 프레임워크를 개발하였다[5]. BPEL 모니터링 프레임워크는 실행 시간에 BPEL의 모니터링 로그를 생성하기 위해 관점지향 프로그래밍을 이용하였다. BPEL 모니터링 프레임워크는 모니터링 규칙을 관점지향 프로그램으로 작성하고 이를 직조하여 로그를 생성하는 엔진을 생성한다. 관점지향 프로그래밍 기법을 이용한 방법은 핵심적인 BPEL 엔진은 수정하지 않고 모니터링 로그를 생성하는 새로운 요구사항을 추가할 수 있는 장점을 가진다. 하지만 다른 새로운 기능이 요구될 경우에 대해서는 고려하고 있지 않아서 적용

1) 관점지향 프로그래밍에서 단일 모듈이 가지는 주된 요구사항
2) 관점지향 프로그래밍에서 여러 개의 모듈에 공통적으로 적용하는 부가적인 요구사항

하기가 용이하지 않다.

AO4BPEL은 BPEL에 관점지향 프로그래밍 기법을 이용하여 BPEL의 재사용성과 모듈화를 높이기 위한 연구이다[9]. AO4BPEL은 BPEL에서 제기된 모듈화의 어려움을 해결하고 서비스를 동적으로 지원하기 위한 방법에 초점을 맞추고 있다. BPEL의 기존 실행 단위(Acitivity)들은 모두 수용하고 관점지향 프로그래밍 기법을 적용한 실행 단위들을 추가하였다. 즉 BPEL문서 내에 관점지향 프로그래밍 기법에 해당하는 관점(Aspect), 교차점(Pointcut), 충고(Advice) 등을 일컫는 실행 단위들이 포함된다. 또한 이를 실행하기 위해 BPEL 엔진 시스템에 관점을 등록하고 활성화하는 도구와 관점의 실행을 제어하는 관점 매니저를 추가하여 설계하였다. AO4BPEL에서는 주로 비기능적 관심사를 모듈화하고 컴포지션 변경을 모듈화할 수 있음을 보이고 있다.

기존에 작성된 프로그램에 새로운 기능을 추가하기 위한 연구로 관점지향 프로그래밍을 이용한 AML(Aspect Markup Language)이 있는데, AML은 자바를 이용하여 JAML (Java Aspect Markup Language)을 구현하였다[10]. 이 연구는 자바의 목적 코드(.class 파일이나 .jar)만 존재하는 프로그램의 기능을 추가하기 위하여, 인터셉터(interceptor)를 정의한 XML 기반의 AML 문서를 제안하고 있다. 하지만 이 방법은 동적으로 코드를 생성하고 실행시키는 프로그램에 적용하기가 어렵다. B2J 엔진은 자동으로 BPEL 문서를 자바 소스로 변환한다. 그리고 변환된 소스 파일을 자동으로 컴파일링하여 목적 프로그램을 생성하고, 이를 엔진 내부의 실행기(Runner)에서 실행시킨다. 그러므로 B2J와 같이 코드를 생성하는 프로그램에 적용하여 사용하기 위해서는 B2J에 합당한 구조가 필요하다.

BPEL의 기능을 그대로 이용하면서 비즈니스 규칙을 사용하기 위한 시스템으로 Rosenberg[4]는 서비스 기반의 BRIB(Business Rule Integration in BPEL)을 제안하였다. Rosenberg는 BPEL 문법을 그대로 이용하면서 규칙을 적용하기 위해 BPEL 엔진이 웹 서비스를 호출하는 "invoke"를 인터셉트하여 규칙 엔진을 호출하는 방법을 사용하였다. 그림 1은 BRIB에서의 웹 서비스 호출 구조를 보인다.

BRIB는 BPEL 문법을 수정하지 않고 규칙 맵을 통해 BPEL 문서와 규칙의 재사용성을 높이고 다양한 BPEL 엔진에 적용할 수 있는 장점을 가진다. 하지만 규칙 엔진의 호출이 "invoke"에만 국한되어 있고 규칙에 비교 대상을 invoke의 인자 외에는 비교할 수 없어 규칙 엔진을 한정적으로 제한한다.

2.2 관점지향 프로그래밍

관점지향 프로그래밍 기법은 1997년 Gregor Kiczales 등이 참여하여 제안한 프로그래밍 개발 방법론이다[6]. 프로그래밍 개발 방법론은 일괄처리 프로그래밍(Batch Process Programming)에서 구조적 프로그래밍(Structured Programming), 객체지향 프로그래밍(Object Oriented Programming)으로 발전해왔다. 구조적 프로그래밍과 객체지향 프로그래밍은 함수를 이용하여 요구사항

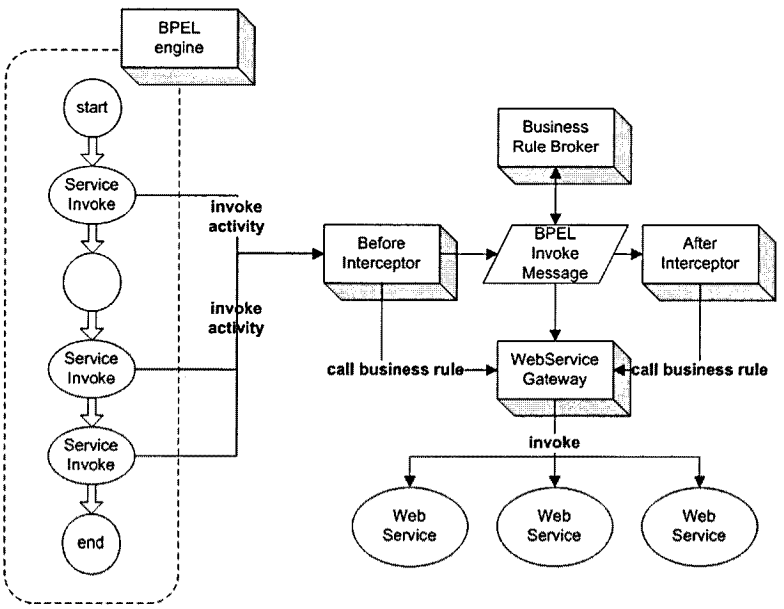

그림 1 Rosenberg가 제안한 규칙 기반 서비스

을 모듈화할 수 있는 방법을 제공한다. 개발자는 어떤 시스템에서 다루고자 하는 도메인을 관심사(concern)로 구분할 수 있는데, 관점지향 프로그래밍에서는 관심사를 핵심 관심사(Core concern)와 횡단 관심사(Cross-cutting concern)로 분류한다. 그리고 관점지향 프로그래밍 기법은 구조적 프로그래밍이나 객체지향 프로그래밍과 같은 기존의 프로그래밍 방법론으로 모듈화하기 어려운 요구사항을 횡단 관심사로 분류하고, 이를 관점(Aspect) 단위로 모듈화하여 모듈의 재사용성을 높였다.

그림 2는 관점지향 프로그래밍을 보이기 위한 대표적인 예로써 은행 시스템을 설명한다. 은행 시스템은 입금과 출금, 계좌이체와 같은 핵심 관심사가 존재한다. 각 핵심 관심사는 객체의 함수로 모듈화한다. 각 모듈은 로깅과 보안, 트랜잭션 등의 공통적인 처리 사항을 가진다. 만약 이러한 공통적인 처리 사항들을 각각 횡단 관심사로 분류하여 처리하면, 각각의 모듈을 변경하거나 또는 반복적으로 사용할 경우 모듈의 독립성을 최대한 유지하면서도 재사용성을 극대화시킬 수 있는 이점이 있다.

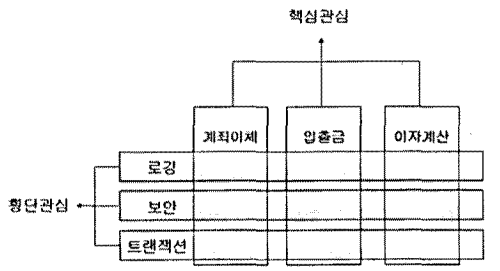

그림 2 은행 시스템의 관점지향 프로그래밍 적용 예

관점지향 프로그래밍에서는 각 관심사가 독립적으로 구현되며, 다른 관심사와는 최대한 느슨하게 결합될 수 있는 방법을 제공한다. 따라서 관점지향 프로그래밍은 각 모듈에 대한 명확한 책임 소재와 시스템 개선의 용이, 코드 재사용의 확대 등의 장점을 갖는다. 본 논문에서는 관점지향 프로그래밍 기법을 적용하기 위한 도구로서 AspectJ[11]를 선택하였다. AspectJ는 객체지향 언어인 자바에 관점지향 프로그래밍 개념을 추가하여 재사용성을 극대화시켜 확장한 범용의 언어이다. 이는 현재 이클립스(Eclipse)[12]의 서브 프로젝트로 발전하고 있으며, Spring 프레임워크[13]에서도 관점지향 프로그래밍 기법을 적용하기 위한 도구로써 사용되고 있다.

2.3 B2J (BPEL to Java)

B2J는 BPEL 문서를 입력으로 받아 BPEL 프로세스들을 처리하고 결과를 사용자에게 제공하는 BPEL 엔진 중 하나이다[7]. 웹 서비스를 기반으로 하는 B2J 엔진은 코디네이터(Coordinator)와 워커(Worker), 그리고 실행기(Runner)로 구성되어 있다. BPEL 코디네이터는 BPEL 문서를 입력으로 받아 자바 소스 코드로 변환시키고, 이를 컴파일링하여 자바 목적 프로그램(.class 파일)을 생성하는 프로그램이다. 그리고 실행기와 워커는 코디네이터가 생성한 자바 목적 프로그램을 실행하는 프로그램이다. B2J의 코디네이터는 BPEL 문서를 자바 프로그램으로 변환시켜 컴파일링하는 모듈로써 B2J의 핵심에 해당한다. 또한 B2J는 코디네이터와 워커, 실행기를 분산 환경으로 구성할 수 있고, 코디네이터를 제외한 워커와 실행기는 코디네이터가 생성하고 컴파일한 목적 프로그램을 실행하기 때문에 경량화되어 있다.

본 연구에서는 B2J 엔진에 관점지향 프로그래밍 기법을 적용하여 새로운 기능을 추가할 수 있는 시스템을 구현한다. 즉 B2J 엔진이 생성한 자바 프로그램에 새로운 기능을 횡단관심사로 추가하는 것이다. B2J가 생성한 프로그램에 새로운 기능 프로그램을 추가하기 위해서는 B2J가 프로그램을 생성하는 규칙을 분석하여 추가하고자 하는 위치를 찾아 새로운 기능 프로그램을 추가해야 한다. B2J는 하나의 자바 클래스로 BPEL 문서의 흐름을 기술하는데, BPEL의 실행(Activity) 단위로 함수를 생성하여 BPEL의 흐름에 따라 호출한다. 제안하는 시스템은 B2J가 생성한 실행 단위의 함수에 새로운 기능을 횡단관심사로 작성한다. 그리고 B2J 코디네이터의 자바 컴파일러를 AspectJ 컴파일러로 변경하여 새로운 기능을 BPEL에 추가한다.

3. 시스템 모델

제안하는 시스템은 BPEL 문서로 작성된 워크플로우가 실행되는 특정 시점에 사용자의 필요에 따라 BPEL에서 제공하지 않는 기능이 실행되는 엔진이다. BPEL로 작성된 워크플로우는 기존 BPEL 엔진을 통해 실행된다. 그리고 특정 지점에서 사용자가 요구하는 새로운 기능이 실행되고 다시 BPEL로 복귀하여 작업을 진행한다. 그림 3은 기존 워크플로우에 새로운 기능이 추가된 워크플로우를 보인다.

그림 3과 같이 제안하는 워크플로우는 기존의 워크플로우 실행 흐름과 새로운 기능으로 구분되어 있다. BPEL 프로그램 흐름은 기존의 엔진을 통해 실행된다. 그리고 새로운 기능은 관점지향 프로그래밍 기법의 직조(Weaving)를 통해 기존 워크플로우 실행 흐름에 삽입된다. 관점지향 프로그래밍 기법을 이용한 직조는 기존 워크플로우의 영향을 주지 않고 다른 프로그램을 삽입할 수 있는 방법이 된다. 일반적으로 BPEL 엔진에 새로운 기능을 추가하기 위해서는 엔진을 수정해야 하는데, 이를 위해서는 많은 시간과 노력이 필요하다. 하지만 자바 프로그램을 생성하는 B2J 엔진과 관점지향

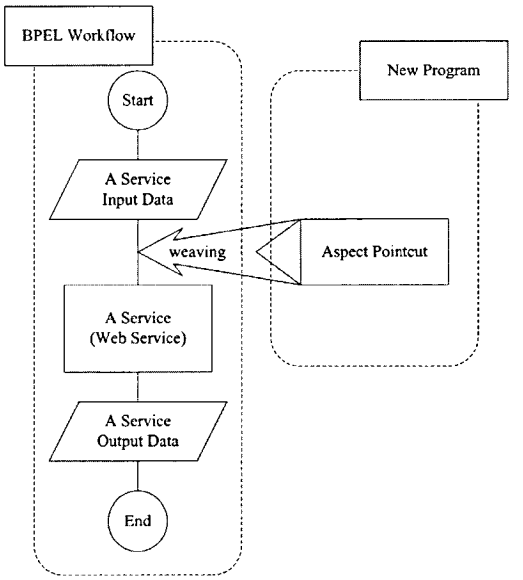

그림 3 새로운 기능이 추가된 워크플로우

프로그래밍 기법을 이용하면 적은 비용으로 새로운 기능을 추가할 수 있다. 그림 4는 기존 B2J가 생성한 자바 프로그램과 새로운 기능이 추가된 JWX 문서를 직조하는 과정을 보이고 있다.

그림 4는 기존의 프로그램에 새로운 기능을 추가하는 방법을 보이고 있다. 일반적으로 기존의 프로그램에 새로운 기능을 추가할 경우에는 프로그램의 복잡도가 증가하게 되는데, B2J와 같이 동적으로 프로그램을 생성할 경우에는 복잡도가 더욱 높아질 수 있다. 제안하는 시스템은 그림 4에서 보는 것처럼 기존 프로그램을 수정하지 않고 새로운 기능을 추가할 수 있으므로, 새로운 기능의 추가에 따른 복잡도 증가를 줄일 수 있다. 그림 4에서 AspectJ 프로그램 생성기(AspectJ Program Generator)는 JWX 문서의 새로운 프로그램이 추가될 BPEL의 위치에 해당하는 함수를 찾고, 그 위치를 교차점으로 직조된 새로운 프로그램을 생성한다.

4. 시스템 구조

제안하는 시스템은 기존의 BPEL 엔진을 수정하지 않고 특정 지점에서 새로운 기능을 동적으로 추가할 수 있는 구조를 갖는다. 그림 5는 제안하는 시스템의 구조를 보인다.

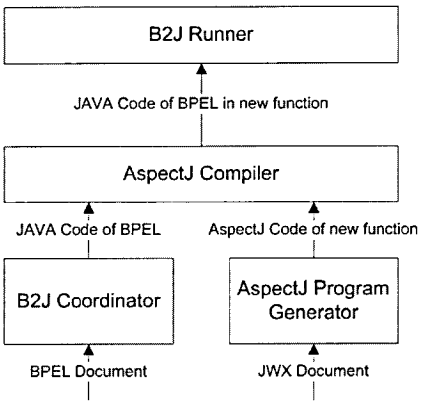

그림 5 제안하는 시스템 구조

본 논문에서 제안하는 시스템은 그림 5와 같이 B2J 코디네이터(Coordinator)와 AspectJ 프로그램 생성기(AspectJ Program Generator), AspectJ Compiler, B2J Runner로 구성되어 있다. B2J 코디네이터는 BPEL 문서를 입력으로 받아 자바 프로그램을 변환하고 AspectJ 프로그램 생성기는 새로운 기능을 작성한 JWX 문서를 입력으로 받아 AspectJ 프로그램을 생성한다. JWX 문서는 BPEL에 추가될 새로운 기능을 작성할 수 있도록 정의된 문서이다. 그리고 생성된 자바 프로그램과 AspectJ 프로그램은 AspectJ 컴파일러를 통해 컴파일되고 B2J 실행 환경인 B2J 실행기(Runner)에서 실행된다. 그림 6은 AspectJ 프로그램 생성기의 구조를 보인다.

AspectJ 프로그램 생성기는 DOM 파서와 JWX 파서, 그리고 AspectJ 프로그램 인코더로 구성되어 있다. DOM 파서는 BPEL 문서를 파싱하고 XPath[14]로 기

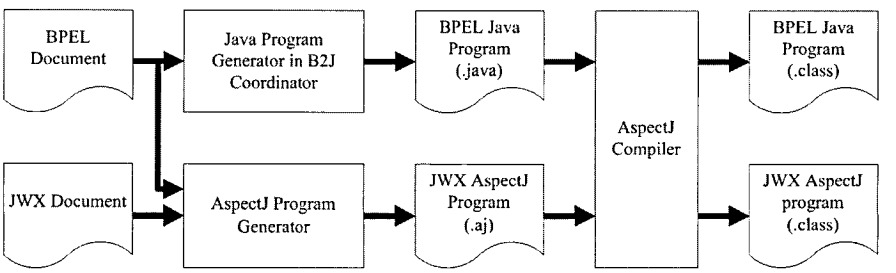

그림 4 제안하는 시스템의 목적 프로그램 생성 과정

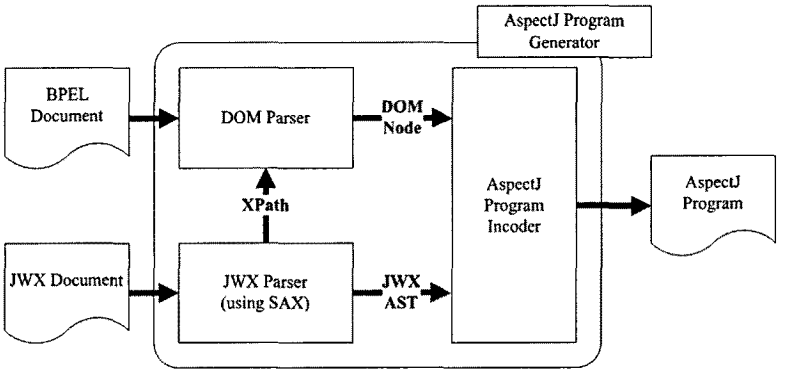

그림 6 AspectJ 프로그램 생성기의 구조

술된 프로그램의 삽입될 위치를 알려준다. 그리고 JWX 파서는 JWX의 AST(Abstract Syntax Tree)를 생성하고, AspectJ 프로그램 인코더는 JWX의 AST와 삽입될 위치 정보를 바탕으로 AspectJ 프로그램을 생성한다.

JWX는 BPEL에 새로운 기능을 추가하기 위해 자바 프로그램을 작성하는 문서이다. JWX는 임포트되는 프로그램 패키지와 교차점이 되는 BPEL 문서의 위치, 자바 프로그램을 작성할 수 있는 구조를 갖는다. 그림 7은 시스템의 실행을 보여주는 BPEL 문서 예제이다. 이 문서는 SMS를 이용하여 다른 사용자에게 메시지를 보내는 흐름을 기술하고 있다. 이 후 예제에서는 새로운 기능을 JWX 문서로 추가한다. 추가되는 새로운 기능은 BPEL이 SMS 웹 서비스로 메시지를 보낸 후 수신자를 사용자에게 보여주는 프로그램이다. 그리고 관점지향 프로그래밍을 이용하여 새로운 기능을 실행시키는 과정을 보인다.

그림 7의 예제 BPEL 문서는 메시지를 받을 사용자의 정보와 메시지 내용을 저장할 변수와 메시지 전송여부

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <process ...>
3   <import importType="http://schemas.xmlsoap.org/wsdl/" ... />
4    <scope>
5     <variables>
6      <variable name="sendWord" messageType="sms:WordMessage" />
7      <variable name="isSucess" messageType="sms:ReturnMessage" />
8     </variables>
9     ...
10    <assign>
11     <copy><from>
12     <user>
13       <name>Kim</name>
14        <phone_number>123-456-5678</phone_number>
15       </user>
16       <word>this is test message</word>
17      </from><to variable="sendWord" part="sendWord" />
18     </copy>
19    </assign>
20    <invoke partnerLink="smsSendWordService" operation="SendWord"
   inputVariable="sendWord" outputVariable="isSuccess" />
21    <assign>
22     <copy><from>
23      <user>
24       <name>Park</name>
25       <phone_number>345-678-9012</phone_number>
26      </user>
27      <word>this is test message</word>
28     </from><to variable="sendWord" part="sendWord" />
29    </copy>
30   </assign>
31   <invoke partnerLink="smsSendWordService" operation="SendWord"
   inputVariable="sendWord" outputVariable="isSuccess" />
32  </scope>
33 </process>
```

그림 7 예제 BPEL 문서

```
<jwx>
	<targetBPEL target="/process/scope/invoke" position="after" />
	<imports>
		<import pkgPath="coolman.test.SMSAlarmMessage" />
	</imports>
	<javaCode>
		SMSAlarmMessage smsAlarmMessage = new SMSAlarmMessage();
		smsAlarmMessage.print(engine);
	</javaCode>
</jwx>
```

그림 8 JWX 문서의 예

를 확인하는 변수를 6~7줄에 생성한다. 그리고 6줄의 변수(sendWord)에 값을 할당하고(10줄~19줄) 웹 서비스로 개발된 SMS를 20줄에서 호출한다. 그리고 21~30줄에서는 새로운 값을 변수에 할당하고 31줄에서 SMS를 호출한다. 그림 8은 그림 7의 BPEL 문서에 새로운 기능을 적용하기 위한 JWX 문서이다.

JWX는 "targetBPEL"과 "imports", 그리고 "javaCode"의 세 부분으로 구성되어 있다. "targetBPEL"은 BPEL 문서에서의 적용 위치를 XPath의 형태로 기술하고, "imports"는 프로그램 호출에 필요한 자바 프로그램 패키지를 기술하며, "javaCode"에 프로그램을 기술한다. 그리고 "print" 함수에서 인수로 사용된 engine은 B2J 엔진에서 사용하고 있는 인스턴스로서 B2J 엔진의 콘솔이나 BPEL 문서의 변수와 같이 B2J의 기능을 접근하기 위해 사용된다. 그림 8의 JWX 문서에서는 BPEL 문서의 "/process/scope" 태그의 하위 태그 중 "invoke" 태그 이후에 프로그램을 삽입하도록 작성되어 있다. 그림 7의 BPEL 예제에서는 20줄과 31줄, 두 위치에서 "invoke"를 사용하였다. 그러므로 두 위치 이후에서 프로그램이 삽입되어야 한다. 그림 8의 JWX 문서는 그림 9에서 보여지는 AspectJ 코드로 변환된다.

그림 9에서 1~2줄은 B2J 엔진의 기능을 사용하기 위한 클래스를 임포트시킨다. 그리고, 3줄은 사용자가 JWX 문서에서 지정한 새로운 프로그램이 사용하는 클래스이다. 본 논문에서는 사용자에게 엔진의 콘솔을 이용하여 SMS가 사용되었음을 알리기 위한 프로그램을 작성하였다. 4~17줄은 AspectJ 프로그래밍 언어의 실행단위가 되는 관점(Aspect)이다. 5~7줄은 교차점(pointcut)으로 JWX 문서에서 "targetBPEL"을 분석하여 위치를 찾는다. 본 예제에서는 22줄과 27줄에서 "invoke"가 사용되었고, 이에 따라 교차점이 해당 함수가 교차점이 된다. 그리고 8~15줄은 충고(Advice)로서 교차점에서 실행될 프로그램을 의미한다. 그림 10은 그림 9의 AspectJ 프로그램과 B2J 코디네이터가 생성한 자바 프로그램을 실행시킨 결과이다.

그림 10은 새로운 프로그램이 engine 인스턴스를 이용하여 엔진의 콘솔을 통해 사용자에게 메시지를 전달할 수 있다는 것을 보이고 있다. 그림 10에서 ①은 그림 6의 예제 BPEL 문서에서 20줄의 "invoke"태그를 교차점으로 하여 출력된 메시지이고, ②는 31줄의 "invork" 태그를 교차점으로 하여 출력된 메시지이다. 예제로 보인 추가된 새로운 기능은 사용자에게 엔진 콘솔을 이용하여 메시지의 발송을 알려주는 프로그램이다.

5. 본 연구의 활용 사례

본 장에서는 BPEL이 문법적으로 가지고 있지 않은 요소를 보이고 제안하는 시스템의 활용 방안에 대해 논한다.

```
1  import org.eclipse.stp.b2j.core.jengine.internal.message.Message;
2  import org.eclipse.stp.b2j.core.jengine.internal.core.RunnerInterface;
3  import coolman.text.SMSAlarmMessage;
4  public aspect ActivityAspect{
5    pointcut activityPoint() : execution(
6      void EngineProgram_20091020174101425.activity*20*() ||
7      void EngineProgram_20091020174101425.activity*31*());
8    after() : activityPoint(){
9      try{
10       RunnerInterface engine =
11 ((JEngineProgram_20091020174101425)thisJoinPoint.getTarget()).engine;
12       SMSAlarmMessage smsAlarmMessage = new SMSAlarmMessage();
13       smsAlarmMessage.print(engine);
14     }catch(Exception e){
15     }
16   }
17 }
```

그림 9 JWX 문서를 이용하여 생성한 AspectJ 프로그램

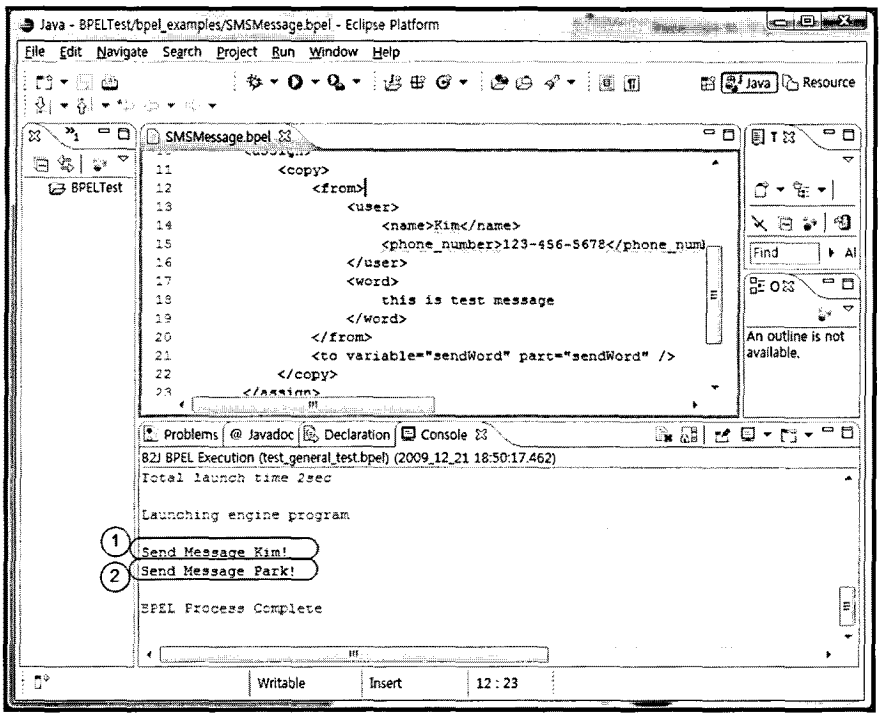

그림 10 실행 화면

5.1 규칙 기능이 추가된 BPEL

BPEL은 작업의 흐름을 기술하는 워크플로우 언어로서 조건을 이용하여 흐름의 분기를 작성할 수 있다. 하지만 복잡한 조건이 요구되는 경우 단순한 조건 비교만으로는 작업의 흐름을 기술하기 어렵다. 하지만 규칙 엔진을 이용하면 복잡한 조건을 규칙을 이용하여 기술할 수 있으며, 단순화된 규칙을 워크플로우에 적용하면 워크플로우의 복잡도를 줄일 수 있다. 그림 11은 규칙이 적용된 워크플로우를 보인다.

그림 11은 규칙의 결과에 따라 분기를 이루거나 규칙의 결과가 다른 서비스의 입력으로 적용되는 워크플로우를 보인다. 워크플로우에 규칙을 적용하기 위해서는 규칙 기반의 워크플로우 언어를 개발하거나[8], 기존의 워크플로우 언어에 규칙을 처리하는 요소를 추가하는 방법이 있다[4]. 하지만 기존의 두 방법에 관한 연구는 이미 개발되어 있는 워크플로우 엔진이 가지고 있는 기능까지 다시 개발하여야 한다.

본 논문은 관점지향 프로그래밍을 이용하여 기존의 BPEL 엔진이 가지고 있는 기능은 그대로 사용하면서 새롭게 기능을 추가하는 엔진을 소개한다. 제안하는 방법은 관점지향 프로그래밍 기법이 제공하는 모듈화로 BPEL 엔진과 새로운 기능의 엔진(예를 들어 규칙 엔진 등) 간의 결합도를 낮추는 효과를 갖고, 이 효과는 엔진의 수정을 바로 적용할 수 있는 장점을 갖는다.

5.2 BPEL 디버깅/프로파일링 엔진

BPEL은 작업의 흐름을 기술하고 실행하는 언어로서 다양한 비즈니스 환경에 적용되고 있다. BPEL의 요구사항이 복잡해짐에 따라 BPEL 문서가 복잡해져 신뢰성이 높은 워크플로우를 작성하기 어려워지고 있다. 신뢰성이 높은 워크플로우는 개발자의 의도와 동일하게 동

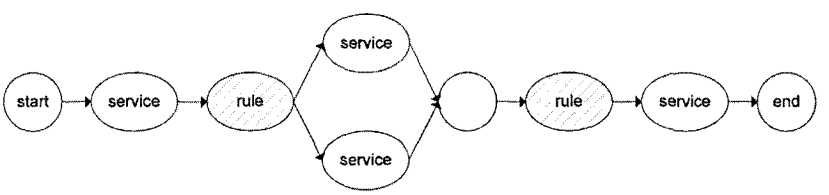

그림 11 규칙이 적용된 워크플로우

작하는 워크플로우이다. 그러므로 신뢰성이 높은 워크플로우를 작성하기 위해서는 실행 시간의 BPEL 모니터링이 요구된다. 일반적으로 기존의 엔진에 로그를 발생시키는 방법은 소스 코드에 로그를 발생시키는 코드를 자동으로 삽입하는 방법[15]과 엔진이나 코어를 수정하여 로그를 발생시키는 방법[16,17]이 있다. 하지만 코드를 자동으로 삽입하는 방법은 응용 실행 시간에 로그를 발생시키는 시간이 포함되어 정확한 응용 실행 시간을 측정하기 어렵고, 엔진이나 코어를 수정하는 방법은 수정 자체에 대한 비용이 크다.

동적 기능 추가가 가능한 BPEL 엔진은 기존의 BPEL의 기능을 핵심관심사로 프로파일링에 관련된 기능을 횡단관심사로 모듈화하여 개발할 수 있어, 두 기능간의 결합도를 낮추고 응집도는 높이는 효과를 갖는다. Sun 연구자는 BPEL을 실행 시간에 모니터링하기 위해 관점지향 프로그래밍 기법을 이용하여 추가적인 실시간 로그를 발생하는 기능을 횡단관심사로 적용하였다. 이 방법은 BPEL 엔진을 수정하지 않고 로그를 발생시키는 기능을 쉽게 추가할 수 있음을 보였다. 하지만 이 방법은 모니터링을 위한 기능을 관점지향 프로그래밍으로 모듈화하였고, 또 다른 추가기능에 대해서는 고려하고 있지 않는다.

6. 결 론

관점지향 프로그래밍 개발 방법론은 프로그램 개발 방법론으로 관심사를 핵심관심사와 횡단관심사로 분류하여 횡단관심사의 재사용성을 높이기 위해 고안되었다. 그 후 관점지향 프로그래밍 방법론은 BPEL에 적용되어 관점지향 방법의 BPEL 시스템[9]에 적용되거나 BPEL 모니터링 워크프레임에 관한 연구[5]에 사용되었다. 즉 BPEL에 새로운 기능을 추가하는 연구가 진행 중이다. 본 논문은 BPEL에 새로운 기능을 추가하기 위해, 그 기능을 기술하기 위한 JWX 문서를 제안하였다. 그리고 JWX 문서를 AspectJ 프로그램으로 변환하고, 이를 컴파일링하여 실행하기 위해 B2J 엔진을 수정하였다. B2J 엔진은 BPEL 문서를 입력 받아 자바 프로그램을 생성하고, 이를 실행하는 BPEL 엔진이다. 제안하는 시스템은 기존 엔진을 거의 수정하지 않고 새로운 기능을 추가할 수 있다.

현재 BPEL은 웹 서비스를 기반으로 하는 다양한 응용에서 사용되고 있고, 유비쿼터스 환경의 발전과 함께 상황인지 시스템에 적용하기 위해 규칙 엔진과 결합하는 방법[4]에 관한 연구가 진행 중이다. 또한 온톨로지를 기반으로 규칙 엔진을 BPEL에 적용하기 위한 연구[18]도 진행 중이다. 하지만 상황 인지를 지원하는 워크플로우 시스템에 관한 연구는 새로운 문법을 제안[8]하거나, 규칙 적용이 서비스를 호출하는데 한정적이어서, 규칙을 적용하기가 용이하지 않다.

본 연구에서 제안한 기존 BPEL 시스템에 요구되는 규칙 엔진을 적용하거나 온톨로지 기반의 상황인지 시스템을 적용하는데 아주 유용하게 사용할 수 있다. 또한 B2J 엔진은 코디네이터와 워커, 그리고 실행기를 분산 환경으로 구성할 수 있고, 코디네이터를 제외한 워크와 실행기는 코디네이터가 생성하고 컴파일링한 목적 프로그램을 실행하기 때문에 경량화되어 있다. 그러므로 컴퓨팅 파워가 작은 편재형 컴퓨팅(pervasive computing) 환경에서도 쉽게 적용할 수 있는 장점을 가진다.

참 고 문 헌

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

[2] Eclipse BPEL Project, http://www.eclipse.org/bpel/.

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

[4] F. Rosenberg and 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.

[5] Mingjie Sun, Bixin Li, Pengcheng Zhang, "Monitoring BPEL-based Web Service Composition Using AOP," *Proceedings of the 2009 Eigth IEEE/ASIC International Conference on Computing and Information Science*, pp.1172-1177, 2009.

[6] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Videira Lopes, Jean-Marc Loingtier, and John Irwin, "Aspect-Oriented Programming," *ECOOP*, pp.220-242, 1997.

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

[8] J. Choi, Y. Cho, "A context-aware workflow system for dynamic service Adaptation," *Computational Science and Its Applications ICCSA 2007*, pp.335-345, 2007.

[9] A. Charfi, M. Mezini, "Aspect Oriented Web Service Composition With AO4BPEL," *ECOWS 2004 volume 3250 of LNCS*. Springer, 2004.

[10] C. V. Lopes, T. C. Ngo, "The Aspect Markup Language and its support of Aspect Plugins," ISR Technical Report UCI-ISR-04-8, University of California, Irvine, 2004.

[11] eclipse AspectJ, http://www.eclipse.org/aspectj.

[12] eclipse, http://www.eclipse.org.

[13] Spring Framework, http://www.springsource.org.

[14] XPath, http://www.w3.org/TR/xpath.

[15] 곽동규, 유재우, "내장형 시스템 소프트웨어를 위한 XML 기반의 프로파일링 도구의 설계와 구현", *한국인터넷정보학회 논문지 제11권 1호*, pp.143-151, 2010. 2.

[16] DTrace, http://www.solarisinternals.com/wiki/index.php/DTrace_Topics.

[17] RTRT, http://www.ibm.com/developerworks/down-

load/s/r/rtrt.

[18] J. Shen, Y. Yang, "From BPEL4WS to OWL-S: Integrating E-Business Process Descriptions," In *SCC '05: Proceedings of th 2005 IEEE International Conference on Services Computing*, pp.181-190, 2005.

곽 동 규

2002년 서경대학교 응용수학과(학사). 2004년 숭실대학교 대학원 컴퓨터학과(석사) 2004년~현재 숭실대학교 대학원 컴퓨터학과 박사과정. 관심분야는 프로그래밍 언어, 컴파일러, XML, 임베디드, 유비쿼터스, etc.

최 재 영

1984년 서울대학교 제어계측공학과(학사) 1986년 미국 남가주대학교 컴퓨터공학(석사). 1991년 미국 코넬대학교 컴퓨터공학(박사). 1992년~1994년 미국 국립 오크리지연구소 연구원. 1994년~1995년 미국 테네시 주립대학교 연구교수. 2001년~2002년 미국 국립 수퍼컴퓨팅 응용센터(NCSA) 초빙연구원. 1995년~현재 숭실대학교 컴퓨터학부 교수. 관심분야는 시스템소프트웨어, 고성능컴퓨팅(HPC), 유비쿼터스 컴퓨팅