관점 지향 프로그래밍(AOP) 기법을 적용한 워크플로우 서비스 시스템

손인선 ○, 최종선, 조용윤, 손은미, 최재영
숭실대학교 시스템소프트웨어 연구실
{ isson, jschoi, emson, yycho@ss.ssu.ac.kr }, choi@ssu.ac.kr

A Workflow Service System Based on Aspect-Oriented Programming(AOP)

Inseon Son○, Jongsun Choi, Yongyun Cho, Eunmi Son, Jaeyoung Choi
Soongsil University System Software Laboratory

요 약

관점 지향 프로그래밍은 구조적 프로그래밍, 객체 지향 프로그래밍 등 기존 프로그래밍 방법론에서 모듈화하지 못한 여러 요구사항에 걸쳐 있는 부가적인 요구사항을 모듈화 할 수 있는 방법을 제공한다. 모듈화 하기 어려운 요구 사항을 횡단 관심으로 분류하고, 이를 애스펙트 단위로 모듈화하여 재사용성을 극대화시키는 관점 지향 프로그래밍은 새로운 프로그래밍 패러다임으로 활발하게 연구가 진행 중이다.

본 논문에서는 현재 비즈니스 및 분산 컴퓨팅 환경에서 많이 이용되는 워크플로우 표준 언어인 BPEL에 워크플로우 서비스의 전이조건의 재사용성을 극대화하기 위해 관점 지향 프로그래밍 기술을 적용한 워크플로우 서비스 시스템을 제안한다. 본 시스템은 워크플로우에 존재하는 독립적인 웹 서비스들이 실행조건으로 요구하는 동일한 정보에 대한 재사용성의 극대화 방법을 제공하는 장점을 가진다.

1. 서 론

관점 지향 프로그래밍(Aspect-Oriented Programming, AOP)[1] 기법은 1997년 Gregor Kiczales 등이 참여하여 제안한 프로그래밍 개발 방법론이다. 어떠한 시스템에서 다루고자 하는 도메인들은 주요 핵심 관심(core concern)으로 분류될 수 있는데, 이러한 핵심 관심을 다루는데 가장 많이 사용되는 방법론이 객체 지향 프로그래밍 방법론(OOP)이었다. 그러나 AOP에서는 구조적 프로그래밍, OOP와 같은 기존의 프로그래밍 방법론으로 모듈화하기 어려운 요구 사항을 횡단 관심(cross-cutting concern)으로 분류하고 이를 애스펙트(aspect) 단위로 모듈화하여 모듈의 변경이나 재사용성을 높일 수 있으며 [2], 현재 AOP 방법론은 유럽 등지에서 새로운 프로그래밍 패러다임으로 활발하게 연구가 진행 중이다 [3, 4, 5].

본 논문에서는 현재 비즈니스 및 분산 컴퓨팅 환경에서 많이 이용되는 워크플로우 표준 언어인 BPEL[6]에 워크플로우 서비스의 전이조건의 재사용성을 극대화하기 위한 관점 지향 프로그래밍 기술을 적용한 워크플로우 서비스 시스템을 제안한다. 본 시스템은 워크플로우에 존재하는 독립적인 웹 서비스들이 실행 조건으로 요구하는 동일한 정보에 대한 재사용성의 극대화 방법을 제공하는 장점을 가진다.

BPEL을 처리하기 위한 다양한 엔진 중에서 BPEL을 자바 코드로 변환하여 컴파일하고 실행하는 B2J(BPEL to Java)[7] 엔진과 객체 지향 언어인 자바(Java)에 AOP 개념을 도입한 최초의 AOP 적용 언어인 AspectJ[8]를 이용해 제안한 시스템을 구현했다.

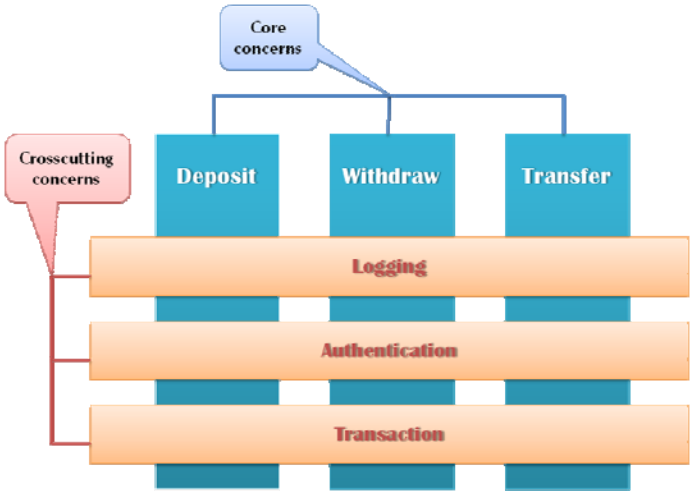
논문의 구성은 다음과 같다. 2장에서는 워크플로우 서비스 시스템에 적용한 AOP 방법론과 AspectJ에 대해 설명한다. 3장에서는 본 논문에서 제안한 BPEL의 실행을 위한 B2J 엔진, 시나리오 기반의 워크플로우 서비스 및 AOP를 이용한 워크플로우 시스템에 대하여 설명한다. 마지막으로 관련 연구와 비교 설명하고, 결론으로 본 논문을 끝맺음 한다.

2. 관점 지향 프로그래밍과 적용 언어

AOP 기법은 OOP 방법론으로 모듈화하지 못하는 부분을 횡단 관심사라는 새로운 모듈로 구분하고, 이를 해결하기 위한 방법으로 연구되었다. <그림 1>은 은행 시스템을 예로 관심사의 분석 방법을 설명한다.

<그림 1>과 같이 은행 시스템은 입금과, 출금, 계좌이체와 같은 핵심 관심사가 존재한다. 각 핵심 관심사는 객체(Object)의 메소드로 모듈화한다. 각 모듈은 로깅과 인증, 트랜잭션 등의 공통적인 처리 사항을 가지는데, 이를 횡단 관심사로 따로 분류하여 처리하면, 모듈을 변경하거나 또는 반복적으로 사용될 경우 모듈의

독립성을 최대한 유지하면서 재사용성을 극대화 시킬 수 있는 이점이 생긴다.



<그림 1> 은행 시스템의 AOP 적용 예

2.1. 관점 지향 프로그래밍의 장점

AOP은 각 관심사가 독립적으로 구현되며 다른 관심사와는 최대한 느슨하게 결합될 수 있는 방법을 제공한다 [9]. 따라서 각 모듈에 대한 명확한 책임 소재, 시스템 개선의 용이, 코드 재사용의 확대 등의 장점을 갖는다. 또한 AOP는 미래 요구사항을 향후 독립적인 애스펙트로 구현할 수 있기 때문에 시스템 설계자는 미래 요구사항에 대한 설계 결정을 보류할 수 있으며, 이러한 특성은 시스템의 신속한 설계를 가능하게 만든다.

2.2. AspectJ

본 논문에서는 AOP 방법론을 적용하기 위한 도구로써 AspectJ 언어를 선택했다. AspectJ는 객체 지향 언어인 자바에 AOP 개념을 추가하여 재사용성을 극대화 시켜 확장한 범용의 언어이다. 현재 이클립스의 서브 프로젝트로 발전하고 있으며, Spring 프레임워크[10]에서도 AOP 적용을 위한 도구로써 AspectJ 언어가 대두되고 있다. AspectJ는 자바의 확장이므로 모든 유효한 자바 프로그램은 유효한 AspectJ 프로그램이 된다. AspectJ 컴파일러는 자바 바이트 코드 명세 규격을 따르기 때문에 자바 가상 머신(Java Virtual Machine)은 AspectJ가 생성하는 클래스 파일을 실행한다. 자바를 기본 언어로 사용하기 때문에 AspectJ는 자바의 모든 기능 및 장점을 가지고 있으므로, 자바 프로그래머가 AspectJ 언어를 이해하는데 어려움이 없다.

AspectJ는 언어 명세와 언어 구현의 두 부분으로 구성된다 [11]. 언어 명세 부분은 코드를 작성하는 언어를 정의한다. 핵심 관심사는 자바 언어로 작성되고 횡단 관심사의 직조에 AspectJ가 사용된다. 언어 구현 부분은 컴파일과 디버깅을 위한 도구 및 인기있는 통합 개발 환경(Integrated Development Environment, IDE)과 통합을 위한 도구를 제공한다.

앞으로 설명할 시스템에는 AOP, 즉 AspectJ에서 사용되는 새로운 용어가 나타나므로 간략하게 설명한다.

2.2.1. 결합점(Join point)

프로그램 실행 과정에서 구별 가능한 프로그램의 지점이다. 메서드 호출이나 객체의 멤버에 값 할당 등이 결합점이 될 수 있다. 결합점이 바로 횡단 관심사 코드가 직조(Weaving)되어 들어오는 지점이다.

2.2.2. 교차점(pointcut)

결합점들을 선택하고 결합점의 환경 정보(context)를 수집하는 프로그램 구조물이다.

2.2.3. 충고(Advice)

교차점에서 지정한 결합점에서 실행되어야 할 코드이다. 결합점의 이전(before)에, 결합점의 이후(after)에 또는 결합점을 대체하여(around) 실행될 수 있다. 대체 충고(around advice)는 결합점에 원래 있던 기존 코드의 실행을 변경할 수 있는데, 기존 코드를 다른 코드로 대체하거나, 또는 아예 기존 코드가 실행되지 않게 할 수도 있다.

2.2.4. 애스펙트(Aspect)

클래스가 자바의 중심 단위인 것처럼 애스펙트는 AspectJ의 중심 단위이다. 애스펙트는 직조 규칙을 표현하는 코드를 포함한다. 앞에서 설명한 교차점, 충고 등이 애스펙트에 들어가게 된다.

3. AOP를 적용한 워크플로우 서비스 시스템

현재 비즈니스 및 분산 컴퓨팅 환경에서 많이 이용되는 워크플로우 표준 언어인 BPEL에 워크플로우 서비스 전이조건의 재사용성을 극대화하기 위해 관점 지향 프로그래밍 기법을 적용한 워크플로우 서비스 시스템을 제안한다. 본 시스템은 워크플로우에 존재하는 독립적인 웹 서비스들이 실행 조건으로 요구하는 동일한 정보에 대한 재사용성의 극대화 방법을 제공하는 장점을 가진다.

본 시스템을 설명하기에 앞서 BPEL을 처리하기 위한 다양한 엔진 중에서 BPEL을 자바 코드로 변환하여 컴파일하고 실행하는 B2J(BPEL to Java) 엔진에 대해 설명한다.

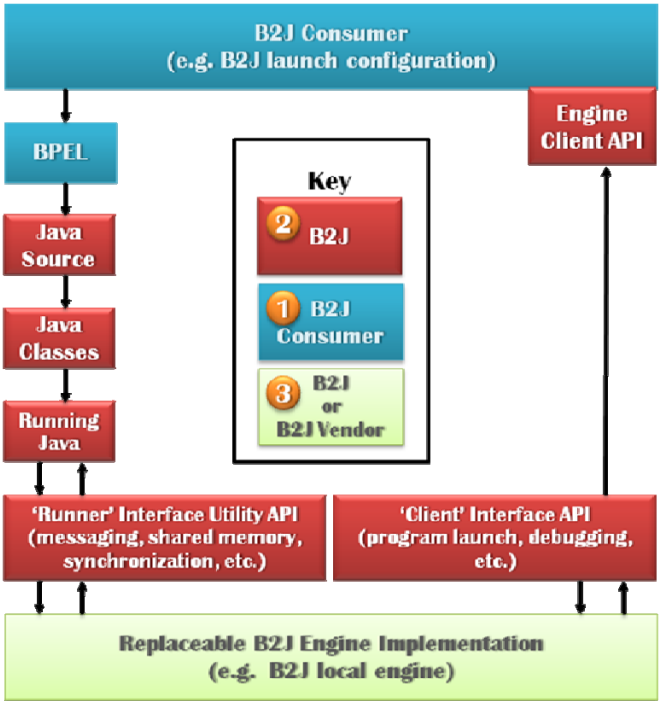
3.1. B2J(BPEL to Java) 엔진

B2J는 크게 두 부분으로 구성된다. 먼저, 첫번째 부분은 BPEL 문서를 입력받아 B2J 엔진 API(Application

Programming Interface, 응용 프로그래밍 인터페이스) 들을 이용하여 자바 소스 파일을 생성한다. 이 파일들은 자바 컴파일러에 의해 자바 클래스 파일로 컴파일된다.

두번째 부분은 컴파일된 클래스 파일을 실행하는 B2J 엔진 시스템으로 구성된다. B2J 엔진 시스템은 <그림 2>와 같이 세 부분으로 구성된다.

<그림 2>에 1번은 BPEL 문서를 입력하는 B2J 실행 환경이다. 2번은 B2J 엔진이다. B2J 엔진은 BPEL 문서를 자바 소스 파일로 변환하고, BPEL 프로그래밍 구조를 구현하기 위한 API들을 이용하여 변환된 자바 소스 파일들을 실행한다. 마지막으로 3번은 대신할 수 있는 B2J 엔진 도구이다. 이는 로컬 엔진과 분산 엔진으로 나뉘는데, 모든 작업은 B2J 엔진 도구에서 실행된다. 본 논문에서는 로컬 엔진 부분을 응용했다.



<그림 2> B2J 엔진 시스템

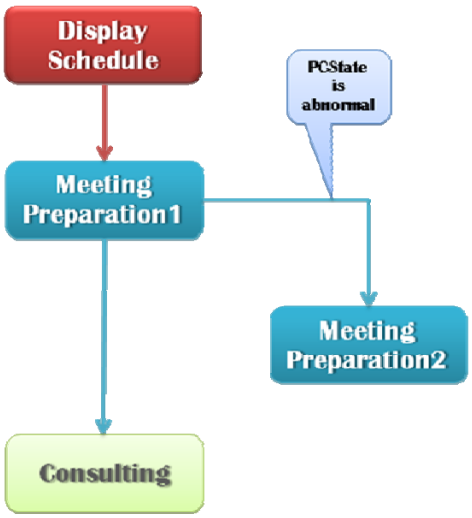
3.2. 시나리오 기반 워크플로우

본 시스템은 입력으로 BPEL 문서를 받는다. 그래서 시나리오 기반의 워크플로우를 BPEL 문서로 작성하였다. <그림 3>은 회의 준비 시나리오를 작성한 BPEL 문서의 예이다.

<그림 3>은 John의 일정을 나타낸다. John이 자신의 사무일정을 PDA에 기록하여 저장한 후, 오전 9시 이전에 출근하여 책상 앞에 앉으면 워크플로우 서비스가 활성화된다. Display Schedule에서는 John이 오전 9시 이전에 출근하여 책상 앞에 앉으면, 일정을 확인할 수 있도록 일정정보를 보여준다.

Meeting Preparation1에서는 오전 10시 313호 회의실에서 발표자료를 John의 컴퓨터로부터 313호에 있는 컴퓨터로 다운로드한다. Meeting Preparation2에서는 313호의 컴퓨터에 이상이 있는 경우, 310호 회의실에서 10분 후에 미팅을 진행할 수 있도록 발표자료를 John의 컴퓨터로부터 310호에 있는 컴퓨터로 다운로드 한다.

마지막으로 Consulting에서는 오후 1시에 고객 상담을 위해 예약된 311호 상담실로 작업환경을 John의 컴퓨터로부터 311호에 있는 컴퓨터로 마이그레이션 해준다.



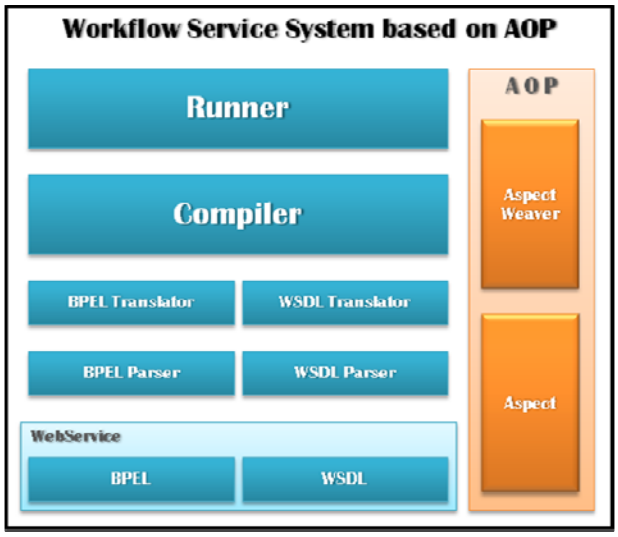
<그림 3> 회의 준비 시나리오

3.3. AOP를 적용한 워크플로우 서비스 시스템

이 절에서는 3.1에서 설명한 B2J 엔진을 기반으로 관점 지향 프로그래밍을 적용한 워크플로우 서비스 시스템을 설명하고, 본 시스템에 3.2에서 설명한 회의 준비 시나리오를 입력하여 동작하는 과정을 설명한다.

B2J 엔진은 BPEL 문서를 입력받아 자바 소스 파일로 변환한다. 그리고 변환된 자바 소스 파일을 컴파일하여 실행하는데, 횡단 관심사 모델링을 위한 모듈화 기법을 적용하기 위해 자바 컴파일러를 AspectJ 컴파일러로 변경하였다. 또, 실행 중 워크플로우 서비스 전이조건을 변경하기 위해 B2J 엔진을 상당 부분 수정하여 본 시스템에 적용하였다.

AOP를 적용한 워크플로우 서비스 시스템의 구조는 <그림 4>와 같다. 본 시스템은 크게 웹 서비스(BPEL, WSDL)가 기본적인 워크플로우를 형성하고, AOP를 이용하여 애스펙트를 실행 중에 직조한다. 먼저, WSDL 문서를 내포하고 있는 BPEL 문서를 입력으로 받는다. 각각의 파서에서 두 문서를 파싱하고, 각각의 번역기에서 자바 소스 파일로 변환한다. 변환된 자바 소스 파일은 자바 컴파일러를 포함하는 AspectJ 컴파일러를 통해 컴파일되고 실행된다. 이는 AOP를 이용하여 애스펙트를 직조하기 위함으로 애스펙트에는 교차점, 충고 등이 작성된 AspectJ 코드가 있다. 워크플로우가 시스템에서 실행 중에 교차점을 만나면 애스펙트가 직조된다.

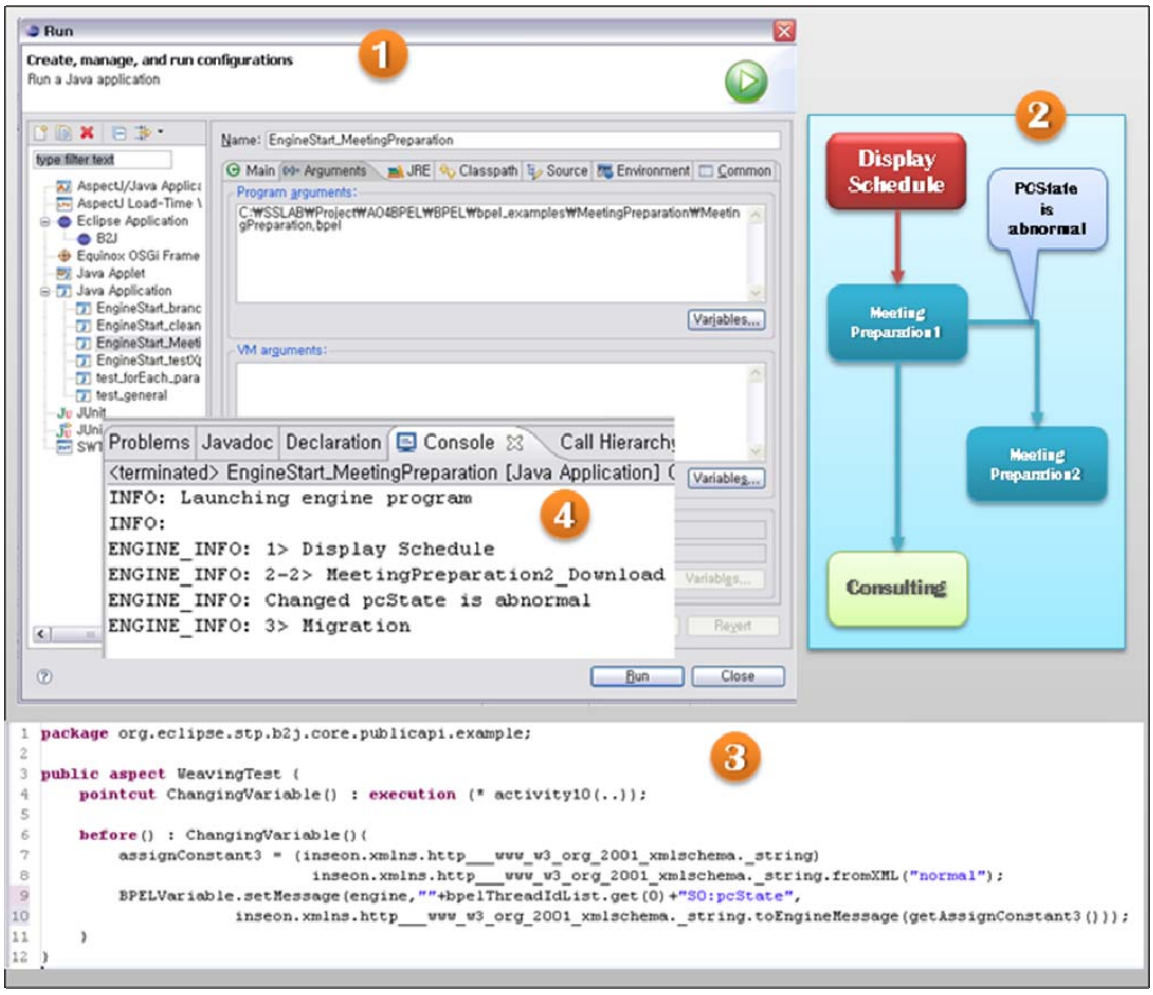


<그림 4> AOP를 적용한 워크플로우 서비스 시스템

이제 회의 준비 시나리오 기반으로 작성된 BPEL 문서가 AOP를 적용한 워크플로우 시스템에서 실행되는 과정을 설명한다. 그림 5는 실행 과정을 나타낸다.

<그림 5>의 1번은 시스템에서 BPEL 문서를 입력할 수 있는 환경이다. 3.2에서 설명한 회의 준비 시나리오를 기반으로 한 BPEL 문서를 입력하였다. 2번은 회의 준비 시나리오의 실행 흐름이다. 시스템이 실행되면 워크플로우가 처리되는데, 실행 중에 교차점을 만나면 3번의 AspectJ 코드가 직조된다. 회의 준비 시나리오 기반의 워크플로우는 먼저 일정 정보를 보여준다. 그리고 미팅 준비를 위해 발표 자료를 John의 컴퓨터로부터 313호 회의실에 있는 컴퓨터로 다운로드한다. 그러나 3번의 ApsectJ 코드가 미팅 준비전에 직조되면서 pcState를 초기 상태인 “normal”에서 “abnormal”로 변경하기 때문에 컴퓨터에 이상이 있다고 판단하고 310호 회의실에서 10분 후에 미팅 준비를 할 수 있도록 발표 자료를 John의 컴퓨터로부터 310호 회의실에 있는 컴퓨터로 다운로드 한다. 마지막으로 고객 상담을 위해 예약된 311호 상담실로 John의 작업환경을 자신의 컴퓨터로부터 311호에 있는 컴퓨터로 마이그레이션 해준다.

<그림 5>의 4번은 위의 실행 과정을 엔진에서 출력한 결과이다. 실행 중에 변경된 pcState의 값도 출력하였다. 본 시스템은 지금까지 설명한바와 같이 워크플로우에 존재하는 독립적인 웹 서비스들이 실행 조건으로 요구하는 동일한 정보에 대한 재사용성의 극대화 방법을 제공하는 장점을 가진다.



<그림 5> 시스템 실행 과정

4. 관련 연구

BPEL에 AOP를 적용한 AO4BPEL[12 , 13]은 BPEL에 있는 액티비티를 모두 수용하고, AOP 기법을 위한 태그들을 BPEL에 추가하였다. BPEL 문서 내에 AspectJ와 유사한 코드를 작성하는데, 애스펙트, 교차점, 충고 등을 일컫는 태그들이 포함된다. 또한 이를 처리하기 위해서 엔진을 설계하고 구현하였다.

그러나 BPEL 문서가 아닌 AOP가 적용된 AO4BPEL 문서를 입력으로 처리하기 때문에 워크플로우 표준 언어 본래의 의미가 손실된다.

본 논문에서 제안한 AOP를 적용한 워크플로우 시스템은 BPEL은 수정하지 않고 자바로 설계된 엔진에서 AspectJ 코드가 처리될 수 있다. 이는 횡단 관심사 모델링을 통해 직조된 BPEL 워크플로우는 기존의 워크플로우와 완전히 호환되므로 어떤 BPEL 엔진에서도 처리될 수 있다. 또한 횡단 관심사 모델링을 통해 재사용성을 극대화하고 차후에 애스펙트로 컨텍스트 처리기를 추가하여 유비쿼터스 컴퓨팅 환경에서 발생하는 컨텍스트 정보를 효율적으로 처리할 수 있다.

5. 결 론

본 논문에서는 현재 비즈니스 및 분산 컴퓨팅 환경에서 많이 이용되는 워크플로우 표준 언어인 BPEL [6]에 워크플로우 서비스의 전이조건의 재사용성을 극대화하기 위한 관점 지향 프로그래밍 기술을 적용한 워크플로우 서비스 시스템을 제안하였다. 또한 워크플로우에 존재하는 독립적인 웹 서비스들이 실행 조건으로 요구하는 동일한 정보에 대한 재사용성의 극대화 방법을 제공하는 장점을 가진다.

또한 향후 연구에서 본 시스템을 확장하여 애스펙트로 컨텍스트 처리기를 추가하면 유비쿼터스 컴퓨팅 환경처럼 컴퓨팅 환경뿐만 아니라 센서, 사용자 정보 및 환경으로부터 얻을 수 있는 모든 정보들을 이용하는 서비스들이 실행될 수 있다.

참고문헌

[1] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Videira Lopes, Jean-Marc Loingtier, and John Irwin, "Aspect-Oriented Programming," ECOOP, pp220-242, 1997

[2] "Aspect-Oriented Programming으로 모듈화 향상시키기: 자바 언어에 AOP를 가져다주는 AspectJ", http://www.ibm.com/developerworks/kr/library/j-aspectj/#resources

[3] Carine Courbis and Anthony Finkelstein, "Towards Aspect Weaving BPEL Engine", ACP4IS, 2004

[4] Carine Courbis and Anthony Finkelstein, "Towards Aspect Weaving Applications", ICSE'05, 2005

[5] Anis Charfi and Mira Mezini, "Aspect-Oriented Workflow Languages", OTM Conferences, 2006

[6] BPEL, "http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html"

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

[8] Gregor Kiczales, Erik Hilsdale, Jim Hugunin, Mik Kersten, Jeffrey Palm and William G. Griswold, "An Overview of AspectJ"

[9] R. Laddad, "AspectJ in Action: Practical Aspect-Oriented Programming", pages 52-53, Manning, 2003.

[10] Thomas Van de Velde, Bruce Snyder, Christian Dupuis, Sing Li, Anne Horton and Naveen Balani, "Beginning Spring Framework 2", pages 355-372, WROX PRESS, 2007

[11] R. Laddad, "AspectJ in Action: Practical Aspect-Oriented Programming", pages 57-62, Manning, 2003.

[12] Anis Charfi and Mira Mezini, "Aspect-Oriented Web Service Composition with AO4BPEL", ECOWS 2004, LNCS 3250, pp. 168–182, 2004

[13] Anis Charfi and Mira Mezini, "AO4BPEL: An Aspect-Oriented Extension to BPEL", World Wide Web Journal: Recent Advances in Web Services (special issue), 2007