Journal of the Society of Korea Industrial and Systems Engineering
Vol, 31, No. 4, pp.100－105, December 2008.

관점지향 프로그램 기반의 동적 소프트웨어 컴포넌트 조합 패턴

배성문* · 박철순**† · 박춘호***

*경상대학교 산업시스템공학부/공학연구원
**창원대학교 산업시스템공학과
***세아정보시스템

Dynamic Software Component Composition Based On Aspect-Oriented Programming

SungMoon Bae* · ChulSoon Park**† · ChunHo Park***

*Dept. of Industrial and Systems Engineering, GyeongSang National University, Engineering Research Institute
**Dept. of Industrial and Systems Engineering, ChangWon National University
***SeAH Information System

Cost reduction, time to market, and quality improvement of software product are critical issues to the software companies which try to survive in recent competitive market environments. Software Product Line Engineering (SPLE) is one of the approaches to address these issues. The goal of software product line is to maximize the software reuse and achieve the best productivity with the minimum cost. In software product line, software components are classified into the common and variable modules for composition work. In this paper, we proposed a dynamic composition process based on aspect-oriented programming methodology in which software requirements are classified into the core-concerns and cross-cutting concerns, and then assembled into the final software product. It enables developers to concentrate on the core logics of given problem, not the side-issues of software product such as transactions and logging. We also proposed useful composition patterns based on aspect oriented programming paradigm. Finally, we implemented a prototype of the proposed process using Java and AspectJ to show the proposed approach's feasibility. The scenario of the prototype is based on the embedded analysis software of telecommunication devices.

Keywords : Aspect-oriented Programming, Software Component Composition, Software Product Line

1. 서 론

다양하고 급변하는 고객의 요구를 충족시키기 위해 기업들은 경쟁력 있는 제품을 빠른 시간 내에 개발하고 생산해야 한다. 예를 들어 글로벌 기업들은 최신 휴대폰을 짧은 시간 내에 개발하여 출시하고 있다. 이러한 제품에 포함되어 제품을 작동시키는 임베디드 소프트웨어는 기존의 제품과 유사한 기능을 제공하면서 새롭게 추가된 기능을 부가적으로 빠른 시간에 구현해야 한다. 따라서 임베디드 소프트웨어 개발자들은 기존의 제품의

논문접수일 : 2008년 08월 21일 논문수정일 : 2008년 09월 03일 게재확정일 : 2008년 09월 03일
† 교신저자 cspark@changwon.ac.kr
※ 이 연구는 2006년도 경상대학교 학술진흥지원사업 연구비에 의하여 수행되었음.

코드를 활용하기 위해 다양한 방법을 사용하고 있다. 하지만, 대부분의 개발자들이 선택하고 있는 것은 단순히 일부 프로그램 코드에 대하여 복사 후 붙여넣기를 하는 것이다. 이러한 복사 후 붙여넣기는 최종 제품에서 예상치 못한 상황을 초래할 수 있고 유지보수 상의 문제를 일으키기도 한다.

소프트웨어 프로덕트 라인은 이러한 문제를 해결하기 위한 하나의 해결책을 제시해 왔다[9]. 소프트웨어 프로덕트 라인은 재사용 가능한 컴포넌트와 가변적인 기능을 구현한 컴포넌트의 조합으로 하나의 시스템을 구축하는데, 기존의 컴포넌트들을 조합하기 위해서는 접합 코드(glue code)를 별도로 컴포넌트에 삽입하거나 기존의 코드를 최소한으로 변경해야 하는 작업이 필요했다. 이러한 추가 작업은 개발자들에게 컴포넌트를 재사용하기 어렵게 만들었다.

본 연구에서는 컴포넌트 조합을 동적으로 쉽게 활용할 수 있도록 하기 위해 관점지향 프로그래밍(AOP : Aspect-Oriented Programming) 기법을 사용하였다. 관점지향 프로그래밍이란 소프트웨어의 요구사항을 "관심의 분리" 개념에 입각하여 각 모듈에서 수행해야 하는 기본적이고 대표적인 업무 기능인 핵심 관심사(core-concern)와 여러 모듈에 걸쳐 있는 시스템 전체적인 부가적인 요구사항인 횡단 관심사(crosscutting-concern)로 분류하여 두 관심사를 연결하여 전체의 시스템으로 만드는 것이다[10]. IBM에서는 선도적으로 관점지향 프로그래밍을 도입하여 자사의 Websphere를 개발함으로써 이 접근법이 실 사례에 적용 가능하다는 것을 보여주었다[12]. 그리고 실제 프로젝트에 관점지향 프로그래밍 접근법을 사용함으로써 미들웨어 플랫폼을 향상시키고, 성능을 향상시킨 사례도 있다[8, 11, 13].

이 논문은 다음과 같이 구성된다. 제 2장에서는 소프트웨어 프로덕트 라인과 컴포넌트 조합에 대한 관련 연구를 소개하고, 제 3장에서는 관점지향 프로그래밍의 특성을 반영한 동적 컴포넌트 조합 패턴을 제시한다. 제 4장에서는 통신 신호 분석 소프트웨어 제품군을 대상으로 본 연구에서 제시한 기법으로 시스템을 모델링하고 구현한 결과를 소개한다. 마지막으로 제 5장에서는 본 연구의 결론을 기술한다.

2. 관련 연구

소프트웨어 프로덕트 라인은 1970년대 코드 재사용에서부터 시작하여, 1980년대 중반 디자인 재사용, 1990년대 초 도메인 분석과 소프트웨어 아키텍처를 거쳐 최근 가장 많은 관심을 받고 있는 기술이다[1]. [7]에 의하면 스웨덴의 CelsiusTech Systems사는 군함 통합 시스템을 개발하는데 프로덕트 라인을 적용함으로써 70~80%의 컴포넌트를 재사용하였고, 이로 인해 8배의 생산성 이익을 달성했다고 한다.

소프트웨어 프로덕트 라인에서 범용 컴포넌트와 가변 컴포넌트 간의 효율적인 컴포넌트 조합을 위해 설계 패턴, 포괄적 프로그래밍, 템플릿 클래스 기반 프로그래밍 등 여러 방법론과 분석법이 개발되고 있다[2, 4-6]. 소프트웨어 프로덕트 라인에서 소프트웨어 재사용을 위한 컴포넌트 조합은 중요한 이슈이다. 공통적인 특성을 구현하는 범용 컴포넌트와 가변적인 특성을 구현하는 컴포넌트의 조합이 얼마나 효율적인가에 따라 생산성이 좌우되기 때문이다. 채은주는 컴포넌트 조립을 위해 컴포넌트 커넥터를 자동으로 생성하여 컴포넌트 포트의 정의, 명세와 아키텍처 구조에 의한 자동 생성을 제안하였다[2]. 임윤선은 비즈니스 논리를 처리하는 중간 계층 컴포넌트들의 재사용 율을 높이기 위해 컴포넌트가 정의한 인터페이스 규격에 따르는 대신 하위 컴포넌트 호출규격을 독자적으로 정의하여 사용하고 이에 대한 메타데이터를 외부로 노출 시키는 새로운 컴포넌트 구조를 제안하였다[3]. 기존의 연구들은 조합에 있어서 사용자의 개입을 많이 요구하였으며, 기존 코드에 대한 이해가 있어야 했다는 단점이 있다. 관점 지향 프로그래밍은 이러한 단점을 보완할 수 있는 접근 방법이다. [6]에서는 소프트웨어 프로덕트 라인에 관점지향 프로그래밍을 적용하여 동적인 컴포넌트 조합이 가능함을 보였다. 하지만 실제 사례에 적용하지 못했으며, 이를 새로운 패턴으로 확장하여 정의하지 못했다.

3. 동적 소프트웨어 컴포넌트 조합 패턴

관점 지향 프로그래밍은 두 관심사 중 횡단 관심사를 모듈화 하기 위해 객체 지향 기술을 기반으로 이것들의 개념과 구조를 확장한 것이다. 핵심 관심사는 기존의 선택된 기본 방법론으로 구현하고 횡단 관심사는 관점지향 기술로 구현한다. 관점지향 프로그래밍을 이해하기 위해서는 다음과 같은 개념을 이해할 필요가 있다.

- 결합점(join point) : 프로그램 실행과정에서 구별 가능한 프로그램의 지점이다. 결합점은 메서드 호출이나 객체의 멤버에 값 할당 등이 될 수 있다. 그 외에 객체 생성, 조건 검사, 비교, 예외 처리, 제어문들도 있다. AspectJ에서 결합점은 횡단 행위를 삽입하는 장소이다.
- 교차점(point cut) : 결합점들을 선택하고 결합점의 환경정보(context)를 수집하는 프로그램 구조물이다. 교차점들은 개발자들이 횡단모듈이 작동해야 하는 프

로그램의 위치를 알려주는 직조 규칙을 지정한다.

- 충고(advice) : 횡단 모듈의 행위와 의사결정을 수행하는 부분이다. 충고는 메서드와 유사한 구조물로 교차점에서 포착한 결합점에서의 횡단 행위를 서술한다. 충고의 종류는 결합점 이전에 실행하는 이전(before)충고, 결합점 이후에 실행하는 이후(after)충고, 그리고 결합점을 대체하는 대체(around)충고가 있다.
- 직조(weaving) : 교차점을 통해 결합점에 충고가 삽입되는 과정을 직조라고 한다. 개별적인 핵심 모듈을 직조 규칙에 따라 횡단 모듈과 통합하여 시스템을 만드는 과정이다. 직조 규칙은 별동의 애스팩트에 정의되는데 관심사가 구현된 모듈과는 별개의 개체이다.
- 애스팩트(aspect) : 엑스팩트는 동적 횡단과 정적 횡단을 위한 직조 규칙을 표현하는 코드를 포함한다. 교차점과 충고를 작성하는 단위이며, 클래스와 같은 프로그래밍 단위로 사용한다.

위와 같은 관점지향 프로그램의 개념은 아래와 같은 형태로 소프트웨어 프로덕트 라인에 적용된다. before() 충고와 after() 충고는 교차점 전후에 수행 할 기능을 구현한 후에 소프트웨어 컴포넌트 조합이 가능하며, before() 충고를 통해 충족 조건이 만족된 경우 수행되는 프로세스도 구현이 가능하다. 사전 조건에 따른 동적 소프트웨어 컴포넌트 조합은 관점 지향 프로그래밍의 around() 충고를 이용한다. around() 충고는 교차점에 의해 지정된 교차점에 대한 프로세스, 코드 그리고 인자에 대한 제어와 수정이 가능하다. 이러한 점을 이용하여 around() 충고를 이용한 동적 소프트웨어 컴포넌트 조합이 가능하다. 본 연구에서는 관점지향 프로그래밍의 특성을 적용하여 컴포넌트 조합에 따른 패턴을 아래와 같이 세 가지로 정의하였다.

- 파라미터 패턴 : 교차점의 입력 파라미터의 정보를 이용하여 파라미터의 값, 타입을 조작하거나, 파라미터의 조건에 따른 컴포넌트 조합이 가능하다. 이는 컴포넌트 조합에서 특정 파라미터의 값 또는 조건에 따른 동적 프로세스 수행을 위해, 또는 상이한 파라미터 타입들을 서로 매핑하기 위한 수단으로 사용될 수 있다.
- 기능 패턴 : 교차점의 기존 함수를 재사용하거나, 함수의 기능을 생략하고 새로운 코드로 대체할 수 있으며, 기존 코드에 추가적인 업무 수행 코드를 작성할 수 있다. 기능 패턴은 기존의 코드를 특정 컴포넌트 조합에서만 제거 또는 대체하거나, 새 기능을 입력하기 위한 수단으로 사용될 수 있다.
- 프로세스 패턴 : 소프트웨어 컴포넌트 조합 시 컴포넌트 간의 프로세스의 우선순위 조작이 가능하다. 프로세스 패턴은 컴포넌트 조합 시 프로세스의 흐름을 조작하기 위해 사용될 수 있다.

다음은 세 가지 패턴을 적용한 간단한 예제이다. 각 패턴들은 around()충고를 사용하였으며, 교차점으로 지정된 함수의 기능, 파라미터, 그리고 컴포넌트 조합 프로세스를 조작하여 동적 소프트웨어 컴포넌트 조합을 이루었다. <그림 1>은 Helloworld 클래스의 코드로 입력 파라미터에 "- Test!!"라는 문자열을 만들어 메시지를 출력하는 Hello 메소드를 포함하고 있다. <그림 2>는 파라미터 값을 조작하는 애스팩트를 구현한 것으로 Hello 메소드의 파라미터 값 T에 문자열 "aspect01 :"을 추가하여 변형된 파라미터 값을 생성하여 Hello 메소드를 추가한 것이다. <그림 3>은 기능 패턴의 예제로서 주어진 파라미터를 이용하기는 하지만, 출력하는 내용이 Hello 메소드를 통해 이루어지지 않고, 비슷한 기능을 수행하는 새로운 함수로 대치되는 애스팩트를 보여주고 있다. 사용자는 Hello를 호출했지만 실제로 Hello가 수행되는

```
public class Helloworld {

    public static void main(String[] args) {
        new Helloworld().Hello("Helloworld");
    }

    public void Hello(String say){
        say += "-Test!!";
        System.out.println(say);
    }
}
```

<그림 1> Helloworld 클래스

```
pointcut say(String T) : call(* hello01.Helloworld.Hello(..)) && args(T);

void around(String T) : say(T){
    T = "aspect01 : " +T;
    proceed(T);
}

-> aspect01 : Helloworld - Test!!
```

<그림 2> 파라미터 값 조작

```
pointcut say(String T) : call(* hello02.Helloworld.Hello(..)) && args(T);

void around(String T) : say(T){
    String rs = "aspect02 : " +T;
    System.out.println(rs);
}

-> aspect02 : Helloworld
```

<그림 3> 함수의 기능을 새로운 기능으로 대체

것이 아니고 새로운 기능을 가진 애스팩트로 대치된 것이다. 이는 전체 프로그램에는 다른 코드에는 영향을 주지 않으면서 새로운 기능을 바꾸어 구현할 수 있는 유용한 패턴이다. <그림 4>는 Hello 메소드를 호출하기 전과 후에 사용자가 지정한 메소드를 호출하여 프로세스의 제어를 하는 애스팩트를 보여주고 있다. 이러한 패턴은 조건에 따라 원하는 프로세스를 수행하거나 우선순위를 제어할 수 있게 한다.

```
pointcut say(String T) : call(* hello03.Helloworld.Hello(..)) && args(T) ;

void around(String T) : say(T){
    new name().Myname("pch : first") ;
    poceed(T) ;
    new name().Myname("pch : last") ;
}
-> My name is pch : frist
-> Helloworld - Test!!
-> My name is pch : last
```

<그림 4> 프로세스 우선순위 조작

4. 시스템 구현

이 장에서는 소프트웨어 프로덕트 라인과 관점지향 프로그래밍의 기법을 적용하여 통신 신호 분석 소프트웨어를 분석하고, 제 3장에서 정의한 조합 패턴을 사용하여 통신 분석 소프트웨어의 시제품을 구현한 것을 설명한다.

본 연구의 적용 대상 소프트웨어는 통신 신호를 지역별로 분석하여 신호가 약한 부분을 파악하여 기지국을 증설하거나 기지국의 통신 장비 값을 설정하는데 사용되는 제품이다. 이러한 분석 소프트웨어(이하 Analyzer)는 CDMA, WiBro, DMB 등과 같이 통신 기술의 발전에 따라 그에 상응하는 분석 모듈을 추가하여 새로운 제품을 개발해야 한다. 하지만, 기술적으로 달라지는 부분 외에는 기존의 제품과 유사하기 때문에 공통부분과 가변부분을 구별하여 재사용성을 극대화한다면 소프트웨어 개발 생산성 및 안정성을 얻을 수 있다. Analyzer는 크게 Gathering and Parsing, 후처리, Display 기능으로 구성된다. Gathering and Parsing 단계는 단말기의 통화 품질 개선을 위해 모니터링하는 단계에서 산출된 측정 파일들을 사용자가 웹상으로 Local Server에 업로드하고, 파일들을 서버에서 자동으로 Parsing하여 파라미터를 중심으로 파일화 하는 과정이다. convert된 결과는 Central Server에서 각 파라미터별로 분리되어 저장된다. 후 처리 단계는 각 파라미터별로 분리되어 저장된 결과를 분석 및 가공하여 업로드 된 파일의 간략적인 정보를 담은 리스트와 각 파일에 대한 상세 정보 페이지를 제공한다. 또한 서버에 저장된 측정 파일(convert 된 파일)에서 사용자가 원하는 조건에 따라 데이터 추출 기능을 제공한다. 마지막으로 Display 단계는 웹 상에서 이루어지는 개략적 통계 분석으로, 사용자에 의해 추출된 데이터를 이용하여 여러 그래픽 요소를 활용하여 사용자에게 원하는 통계 정보를 Report하는 기능이다.

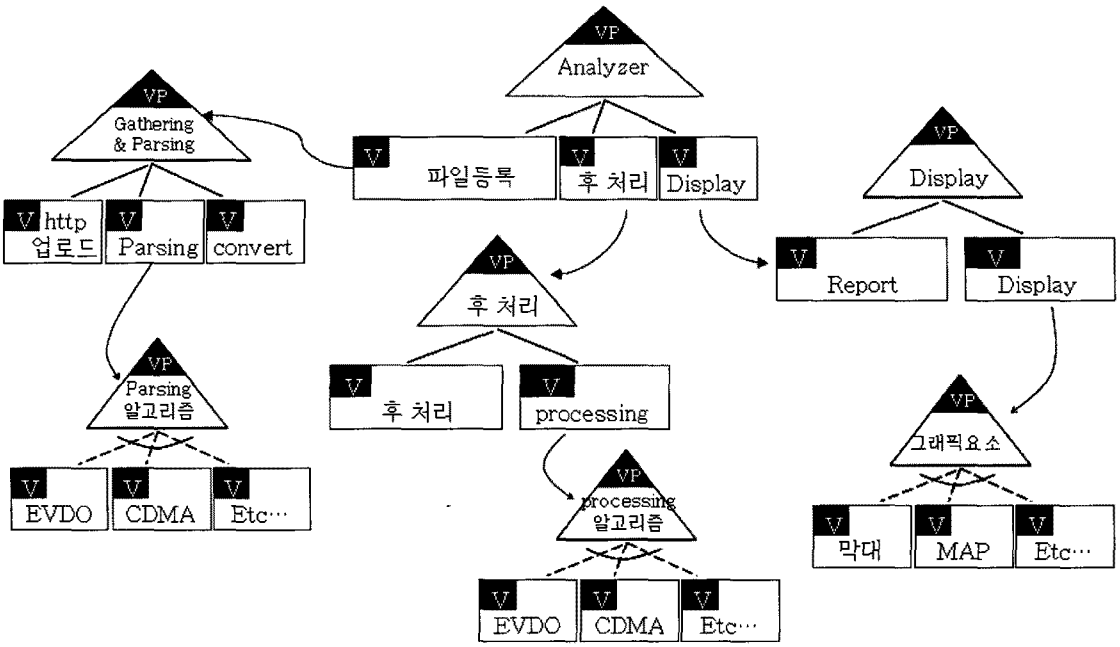
Analyzer의 각 단계별 기능 요구사항과 요구사항이 공통 기능인지 아니면 가변 기능인지를 구분한 결과는 <표 1>과 같다. 이를 소프트웨어 프로덕트 라인의 Orthogonal variability model로 표현한 것은 <그림 5>와 같다. <그림 5>에서 보듯이 파싱 부분의 가변 컴포넌트와 후처리 부분의 가변 컴포넌트, 디스플레이의 가변 컴포넌트가 정의되었다. 이 가변 컴포넌트는 관점지향 프로그램의 애스팩트로 정의되어 시스템을 구현할 때 동적으로 컴포넌트들이 조합될 수 있도록 한다.

<표 1> Analyzer 요구사항

기 능		요구 사항	구분
Gather -ing & Parsing	HTTP 업로드	◦ 측정 파일(들)을 Local Server에 업로드	공통 기능
	Parsing	◦ 파라미터를 중심으로 TXT 파일화 ◦ Central Server에서 각 파라미터별로 분리되어 저장	가변 기능
	Convert	◦ 서버에 업로드 완료된 파일(들)을 자동으로 Parsing	공통 기능
후 처리	후 처리	◦ 서버에 저장된 파일(들)의 간략한 정보 페이지 제공	공통 기능
	Process -ing	◦ 파일(들)의 상세한 정보 페이지 제공 ◦ 사용자가 원하는 조건에 따른 데이터 추출 기능 제공	가변 기능
Display	Report	◦ 사용자에게 기본 통계 분석 페이지 제공	가변 기능
	Display	◦ 여러 그래픽요소를 이용하여 기본 통계 정보 출력	가변 기능

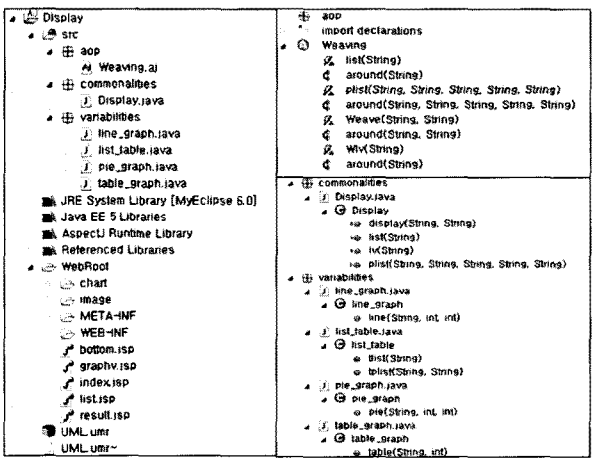
Analyzer 제품의 가변 기능을 관점지향 프로그래밍으로 구현한 시제품의 구현환경은 다음과 같다.

- 운영체제 : windows XP
- 프로그래밍언어 : Java-JDK 5.0
- 관점지향프로그래밍도구 : AJDT(AspectJ Development Tools) plug-in
- 웹 서버 : Tomcat v5.5
- 그래픽처리 : JFreeChart 1.0.8



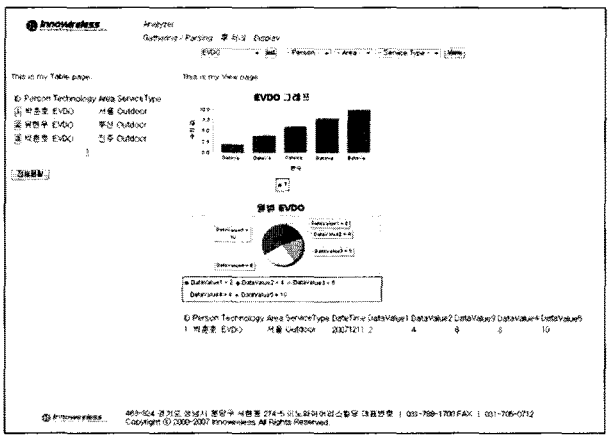
<그림 5> Analyzer 제품에 대한 Orthogonal variability model

<그림 6>는 모델링한 결과를 시스템으로 구현하면서 필요한 클래스 구조와 애스팩트를 구조화한 것이다. 웹페이지에 정보를 호출하고 그래프를 출력하는 공통 기능인 list와 display의 구현은 commonalities 패키지의 Display 클래스에 구현되어 있고, 가변 기능은 variabilities 패키지에 구현되어 있다. commonalities 패키지의 공통 기능과 variabilities 패키지의 가변 기능들을 조합하기 위해서 aop 패키지의 AspectJ 파일인 weaving 클래스에 around() 충고를 이용하여 직조 규칙을 구현하였다.



<그림 6> 클래스와 애스팩트 구조

<그림 7>은 공통 컴포넌트와 가변 컴포넌트를 조합하여 하나의 디스플레이 페이지를 구현한 것이다. 이 페이지는 적용 기술을 선택 후, 사람, 지역, 날짜, 서비스 타입에 관련된 조건을 선택 후 관련 통계 정보와 그래프 결과를 표시하고 있다.



<그림 7> AOP로 구현한 화면

5. 결 론

본 연구에서는 핵심관심과 횡단관심사를 분리한 관점지향프로그래밍기법을 사용하여 재사용가능한 컴포넌트를 조합하기 위한 기법을 제시하였다. 또한 실제 사용중인 통신 분석 소프트웨어 제품에 대해 분석을 하고

시제품을 구현함으로써 적용 가능성을 보였다. 이 기법은 각 모듈에 대한 명확한 책임 소재를 구분할 수 있고, 고도의 모듈화를 이룰 수 있으며, 핵심 모듈의 변경 없이 시스템의 개선이 용이하다. 또한 모듈 간 느슨한 결합으로 코드 재사용이 확대되며, 기능 구현의 비용이 감소하고, 시스템 개발 기간이 단축된다. 따라서 휴대폰 운영 소프트웨어와 같이 유사한 기능을 반복적으로 포함하면서 가변기능이 조금씩 추가되는 시스템을 개발하는 분야에서 이러한 접근법이 유용할 것으로 기대된다. 본 연구에서는 관점지향 프로그래밍 기법을 해당 제품의 일부 기능에 한정하여 시제품을 구현하였는데, 추후 연구에서는 적용대상을 전체 기능으로 확대하여 시스템을 구현할 계획이다.

참고문헌

[1] 강교철; "Software Product Line Engineering", SEEK 소프트웨어공학 워크샵, 2 : 15-40, 2001.

[2] 이승연, 권오천, 신규상; "아키텍처에 기반한 컴포넌트 조립시스템 설계 및 구현 방법과 지원 도구의 개발", 정보과학회논문지, 30(9) : 812-820, 2003.

[3] 임윤선, 김명, 정안모; "독립적으로 개발된 바이너리 컴포넌트들의 조립을 지원하는 컴포넌트 모델", 2007 한국컴퓨터종합학술대회 논문집, 34(1) : 138-142, 2007.

[4] 채은주, 한정수, 백순화; "컴포넌트 조립을 위한 커넥터 자동 생성", 한국콘텐츠학회 2004 추계종합학술대회논문집, 2(2) : 408-411, 2004.

[5] 최승훈, 홍의석; "실시간 임베디드 소프트웨어 프로덕트 라인을 위한 컴포넌트 코드 생성", 자연과학논문집, 13 : 197-213, 2007.

[6] 허승현, 최은만; "AOP를 적용한 프로덕트 라인 가변기능의 구현", 정보처리학회논문지(D), 13(4) : 593-602, 2006.

[7] Bass, L., Clements, P., and Kazman, R.; "Software Architecture in Practice," 2nd ed., Addison-Wesley, 2003.

[8] Colyer, A. and Clement, A.; "Large-scale AOSD for Middleware," 2004 AOSD conference, : 56-65, 2004.

[9] Griss, M.; "Software Product Lines : Practices and Patterns," Addison-Wesley, 2002.

[10] Laddad, R.; "AspectJ in Action : Practical Aspect-oriented Programming," Manning, 2003.

[11] Putrycz, E. and Bernard, G.; "Using Aspect-Oriented Programming to build a portable load balancing service, 22nd Int'l Conference. on Distributed Computing Systems Workshops, : 473-478, 2002.

[12] Sabbah, D.; "Aspects-from Promise to Reality," 2004 AOSD Conference, : 1-2, 2004.

[13] Zhang, C. and Jacobsen, H.: "Refactoring Middleware with Aspects," *IEEE Transactions and Parallel and Distributed System*, 14(11) : 1058-1073, 2003.