한국산학기술학회논문지
Vol. 11, No. 3, pp. 943-950, 2010

관점지향 소프트웨어 개발 방법론과 디자인 패턴을 적용한 출입 보안 시스템 개발

김태호[1], 천현재[1], 이홍철[1*]
[1]고려대학교 정보경영공학과

Development of Secure Entrance System using AOP and Design Pattern

Taeho Kim[1], Hyeonjae Cheon[1] and Hongchul Lee[1*]
[1]**Department of Information Management Engineering, Korea University**

요 약 출입 보안 시스템은 감시, 로깅, 추적, 인증, 권한부여, 직원 위치 파악, 직원 출입관리, 출입문 관리 등 수많은 기능을 해야 하는 복잡한 시스템이다. 본 논문에서는 관점지향 소프트웨어 개발 방법론(Aspect Oriented Programming: AOP)과 디자인 패턴을 적용해 국내 원자력 발전소의 출입 보안 시스템을 구축하였다. AOP를 이용하면 시스템의 비즈니스 로직과 보안 로직을 완전히 독립적으로 분리해서 시스템 구축이 가능하므로, 출입 보안 시스템의 각 기능별 모듈에 대하여 명확하게 그 역할을 구분해 줄 수 있는 장점이 있다. 이는 잦은 외부환경의 변화에 의한 시스템 변경을 유연하게 대처할 수 있게 하며 AOP의 본래의 장점인 코드 재사용성의 확대, 효율적인 기능 구현 등 이 가능해 진다. 이와 함께 디자인 패턴을 활용하면 일반적인 소프트웨어 개발에서 나타나는 복잡한 문제를 구조화 하여 설계 할 수 있어, 시스템의 안전성 또한 보장 받을 수 있다. 두 방법론의 장점을 활용하여, 그 기능이 복잡한 출입보안 시스템을 안정적으로 설계·구현 할 수 있다.

Abstract A secure entrance system is complicated because it should have various functions like monitoring, logging, tracing, authentication, authorization, staff locating, managing staff enter-and-leave, and gate control. In this paper, we built and applied a secure entrance system for a domestic nuclear plant using Aspect Oriented Programming(AOP) and design pattern. Using AOP has an advantage of clearly distinguishing the role for each functional module because building a system separated independently from the system's business logic and security logic is possible. It can manage system alternation flexibility by frequent change of external environment, building a more flexible system based on increased code reuse, efficient functioning is possible which is an original advantage of AOP. Using design pattern enables to design by structuring the complicated problems that arise in general software development. Therefore, the safety of the system can also be guaranteed.

Key Words : Aspect Oriented Programming, Design Pattern, Secure Entrance System

1. 서론

기술 산업이 발전하면서 수많은 첨단 제품들이 쏟아지고, 이에 따라 기술 집약적인 거대한 첨단 기업들은 막대한 이익을 내고 있다. 이러한 환경에서 기업들은 유출될 경우 그 피해가 기업존폐까지 위협당할 수 있는 자기회사만의 핵심 기술을 지키기 위해 기업의 보안 관리에 총력을 다하고 있다. 보안 시스템은 감시(monitoring), 로깅(logging), 추적(tracing), 인증(authentication), 권한부여(authorization), 직원 위치 파악, 직원 출입관리, 출입문

이 연구에 참여한 연구자(의 일부)는 '2단계BK21사업'의 지원비를 받았음.
*교신저자 : 이홍철(hclee@korea.ac..kr)
접수일 09년 10월 07일 수정일 (1차 10년 02월 04일, 2차 10년 03월 12일) 게재확정일 10년 03월 18일

관리, 등 수많은 기능을 해야 하는 복잡한 시스템이다. 보안시스템은 일반기업뿐만 아니라 연구소, 공항, 공공시설 등 보안이 요구되는 모든 곳에서 필요로 하는 중요한 시스템이다.

현재까지는 이러한 보안 시스템을 구축하기 위해서는 보통 Object Oriented Programming(OOP)을 이용해 왔다. 하지만 OOP로 잘 구축된 시스템 일지라도, 중복되는 코드, 생산성의 저하, 재활용성의 저하, 구현 후 기능 변경의 어려움 등의 단점이 있었다[1]. 따라서 OOP만으로는 다양한 기능을 가지는 복잡한 구조의 시스템을 각각의 기능별로 명확히 구분해서 설계·구현 하기는 어렵다.

이에 본 논문에서는 OOP 방법론의 단점을 보안해 줄 수 있는 대안으로 등장한 AOP와 구조화된 시스템 설계를 도와주는 방법론인 디자인 패턴을 적용해서 원자력 발전소 출입 보안 시스템을 구현하였다. 원자력 발전소의 특성상 시설 출입에 대한 보안이 매우 중요하다. 이에 여러 구성 요소 중, 필수요소인 실시간 모니터링과 로깅(Logging)시스템을 설계·구현하고 , OOP로만 구현한 시스템에 비해 향상된 부분을 제시하였다.

이를 위해 Ⅱ에서 기존연구를 분석하고, 보안시스템의 특징을 살펴본 후, 본 논문에서 쓰이는 AOP와 디자인 패턴 대해 설명한다. Ⅲ에서 출입 보안 시스템의 핵심 요소인 실시간 모니터링 시스템과 로깅시스템을 AOP와 디자인 패턴을 적용해 설계한 후, 원자력 발전소 출입 보안 시스템을 구현한다. Ⅳ에서 제안된 시스템의 장점과 효용성을 확인하였다.

2. 이론적 배경

2.1 기존연구

보안 시스템 전반에 관한 연구인 Charles B. Haley et. al.[2]는 보안시스템에서 필요한 요구사항들을 제시하고, 도출된 요구사항에 맞는 프레임워크를 제시하였다.

많은 연구에서 AOP를 보안시스템에 적용해서, 비즈니스 로직과 보안 로직을 분리함으로써 시스템의 구조와 유연성을 증가시키고, 재사용성을 높이는 효과를 보여주고 있다[3,4].

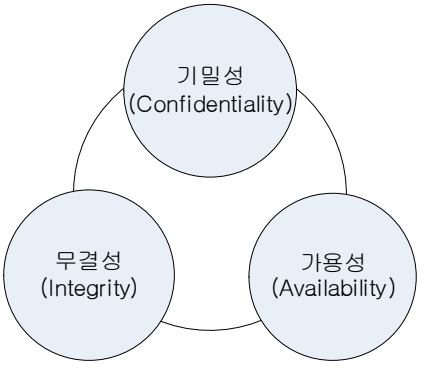
John Viega et. al.[5]은 AOP가 보안시스템 구축시 다양한 분야에 이용될 수 있다고 하고, AOP 방법론이 이미 구성되어 있는 비즈니스 로직을 변경하지 않고, 보안 로직을 추가하는 작업을 data overflow 문제로 설명하였다. Geri Georg et. al.[6]는 보안시스템을 규정한 후 Aspects 방법론을 이용해 보안시스템을 설계(design)해 봤지만, 단지 설계 단계에서만 Aspects 방법론을 결합해보고, 실제 구현 단계까지는 나아가지 못했다. Bart De Win et. al.[7]는 AOP를 이용해 보안 시스템을 구축해 보았지만, 그 구축 영역을 개인보안시스템(A personal information management system)과, 파일전송시스템(FTP Server)시스템으로 한정하였다.

AOP 언어인 AspectJ를 디자인 패턴에 적용한 연구인 Jan Hannemann et. al.[8]은 GoF 디자인 패턴에 AOP 언어인 AspectJ를 접목시켜, 이러한 방법으로 기존의 객체지향 디자인 패턴으로 구축한 시스템 보다 더 명확하고 명시적으로 표현 할 수 있음을 보였다. 하지만 이론적인 제시에만 그쳤다.

기존연구를 통해 AOP 방법론이 보안시스템의 설계에 다양한 장점이 있음을 확인하였다. AOP 방법론이 디자인패턴에 적용되면, 디자인 패턴의 구조를 유연하게 발전시킬 수 있음을 확인하였다. 이에 이론적 설계에서 벗어나, AOP 방법론과 디자인 패턴을 혼합하여 실제로 보안 시스템을 설계·구현하였다.

2.2 보안 시스템의 특징

보안 시스템 구축 시 고려해야 하는 요소[9]는 다음 그림1과 같다.



[그림 1] 보안 시스템의 3요소

- 기밀성(機密性, confidentiality)

허락 되지 않은 사용자 또는 객체가 정보의 내용을 알 수 없도록 하는 것이다. 비밀 보장이라고 할 수도 있다. 원치 않는 정보의 공개를 막는다는 의미에서 프라이버시 보호와 밀접한 관계가 있다.

- 무결성(無缺性, integrity)

허락 되지 않은 사용자 또는 객체가 정보를 함부로 수정할 수 없도록 하는 것이다. 다시 말하면, 수신자가 정보를 수신했을 때, 또는 보관돼 있던 정보를 꺼내 보았을

때 그 정보가 중간에 수정 또는 첨삭되지 않았음을 확인할 수 있도록 하는 것이다.

- 가용성(可用性, availability)

허락된 사용자 또는 객체가 정보에 접근하려 하고자 할 때 이것이 방해받지 않도록 하는 것이다. 최근에 네트워크의 고도화로 대중에 많이 알려진 서비스 거부 공격(DoS 공격, Denial of Service Attack)이 이러한 가용성을 해치는 공격이다.

보안 시스템의 경우, 많은 보안 전문가들과 소프트웨어 엔지니어들이 보안과 관련된 요소들을 추출하여 독립적인 컴포넌트로 분리 하려고 노력해 왔다. 그러나 보안 시스템이 적용되는 곳의 환경이 서로 상이하고, 시스템에 대한 사용자의 요구사항이 달라 보안 관련 기능들을 독립적인 요소로 추출하기가 어렵다. 이를 충족시키기 위해서는 커스트마이징이 많이 필요하기 때문에 보안시스템은 보통 그 구성이 매우 복잡하다[2,7].

보안관련 문제의 경우, 보안 전문가와 소프트웨어 엔지니어의 지식이 다르기 때문에, 보안 시스템 구축 시 서로 충분한 의사소통을 하지 않는 이상, 소프트웨어 엔지니어들이 보안관련 사항들을 깊숙이 고려해서 설계하기가 싶지 않다. 이러한 결과로 종종 보안 관련 이슈들이 프로그램 개발 후반부에 반영되어서, 프로그램의 전체적인 기회비용을 올리고, 설계 및 소프트웨어의 질을 떨어뜨리는 경우가 많다. 따라서 안정적인 보안시스템 구축을 위해서는 소프트웨어 엔지니어가 개발 초기부터 구축되는 환경에 필요한 보안 관련 이슈를 충분히 이해하고 설계에 반영하는 것이 필요하다[3].

따라서 다양한 기능을 가지는 복잡한 구조의 보안시스템을 유연하고 안전성 있게 설계하기 위해서는, 보안관련 이슈에 해당하는 보안로직과 프로그램의 주요 임무인 비즈니스 로직을 분리 시켜 설계 하는 것이 필수적이다. 이렇게 보안 로직과 비즈니스 로직을 분리해서 설계·개발함으로써 복잡한 보안시스템의 기능들을 명확한 역할을 가진 모듈로 분리하여 구현할 수 있고, 끊임없이 변하는 사용자의 요구사항에 쉽게 반응하는 유연한 시스템 설계가 가능하다. 이를 위해 본 논문에서는 AOP 방법론과 디자인 패턴을 활용하였다.

2.3 Aspect Oriented Programming

AOP는 1997년 Gregor Kiczales et. al.[10]가 "Aspect-Oriented Programming"에서 제안한 프로그램 개발 방법론이다. AOP에서 프로그램의 요구사항을 concern이라고 한다. Concern은 core concern과 cross-cutting concern으로 나뉜다. core concern과 cross-cutting concern을 설명하기 위해 가장 널리 쓰이는 예는 은행 시스템이다. 은행 시스템에서 core concern은 은행의 주요업무인 계좌이체, 입출금 등이다. 이에 반해 Logging, Security, Transaction 등과 같이 은행의 모든 업무에서 필요한 기능들을 cross-cutting concern 이라 한다.

OOP는 core concern을 다루는데 현재 가장 많이 사용되는 방법론이지만 cross-cutting concern을 처리하기에는 부족하다. 특히 복잡한 어플리케이션인 경우에는 더욱 그렇다[11].

AOP는 aspect라는 새로운 모듈을 통해 cross-cutting concern을 분리하여 구현 할 수 있도록 해준다.

2.3.1 AspectJ

AspecJ[1]는 대표적 OOP언어인 Java기반의 AOP개발 언어이다. Gregor Kiczales를 포함한 제록스 연구원들이 1990년대 말에 AspectJ를 만들었다. 후에 제록스는 AspectJ 프로젝트를 오픈 소스 커뮤니티 eclips.org에 이전했고, 이 커뮤니티에서 AspectJ를 지원하며 발전시키고 있다. AspectJ를 사용하면 책임소재가 명확한 모듈, 높은 모듈화, 시스템 변경 시 유연함, 코드 재사용성 증가, 적시성 증가(개발기간 단축) 등의 장점이 있다[7].

2.3.1.1 AspectJ 요소

AspectJ는 Java언어로 만들어진 프로그램에 AOP를 적용하는 컴퓨터 언어이다. AspectJ는 다음의 요소로 구성되어 있다.

- 결합점(Join point)

결합점은 프로그램 실행 과정에서 구별 가능한 프로그램의 지점이다. 메서드 호출이나 객체의 멤버에 값 할당 등이 결합점이 될 수 있다. 결합점이 바로 횡단 코드가 직조되어 들어오는 지점이이 때문에 AspectJ에서는 모든 것이 결합점을 중심으로 이루어진다.

- 교차점(Pointcut)

교차점은 결합점들을 선택하고 결합점의 환경정보(context)를 수집하는 프로그램 구조이다. 예들 들어, 교차점은 메서드 호출을 지정하고 메서드가 호출되었을 때의 대상 객체와 메서드의 매개변수 등과 같은 메서드의 환경정보를 얻을 수 있다.

- 충고(Advice)

충고는 교차점에서 지정한 결합점에서 실행되어야 할 코드이다. 충고는 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.

- 도입(Introduction)

도입은 시스템의 클래스, 인터페이스, 그리고 애스팩트에 변화를 도입하는 정적 횡단 명령이다. 도입은 모듈의 행위에 직접 영향을 미치지 않는 정적 변화를 만든다. 예를 들어 클래스에 메서드나 필드를 추가할 수 있다.

- 애스펙트(Aspect)

클래스가 Java의 중심 단위인 것처럼 애스펙트는 AspectJ의 중심 단위이다. 앞에서 설명한 교차점, 충고, 도입과 선언이 애스펙트에 들어가게 된다. AspectJ의 요소 외에, 애스펙트는 클래스처럼 자료, 메서드와 중첩 클래스 멤버를 포함할 수 있다.

2.4 디자인 패턴

디자인 패턴은 컴퓨터 프로그램 개발에서 자주 나타나는 과제를 해결하는 정형화된 방법이다. 특정한 상황에서 구조적인 문제를 해결하는 방식을 설명해 준다. 일반적으로 알려진 GoF(Gang of Four) 디자인 패턴은 소프트웨어 개발 문제에 대한 구조적인 해결 방법을 제시해준다[8]. 최근 10년간 소프트웨어의 규모가 커지고 그 구조가 복잡해지면서, 대규모 소프트웨어 시스템 설계에 디자인 패턴을 적용하기 시작했다. 특정한 상황에 맞는 검증된 패턴을 활용해서 시스템을 설계한다면, 그 시스템의 안정성 또한 보장 받을 수 있다[12,13].

3. 출입 보안 시스템 구축

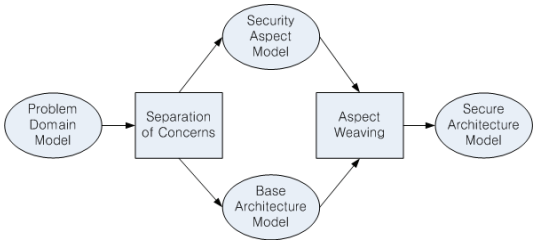
보안 시스템은 수많은 기능을 해야 하는 복잡한 시스템이다. 이와 같은 보안 시스템은 많은 cross-cutting concern이 존재해서 AOP를 활용해서 구축하기에 최적 조건의 시스템이다.

시스템에 AOP를 효과적으로 적용하기 위해서는 다음을 고려해야 한다. 첫째 해당 시스템의 구조를 Object Oriented(OO) 관점에서 잘 정의해야 한다. OO관점으로 잘 정의된 구조일수록 AOP를 적용했을 때 그 효과가 커진다. 둘째, 개발자는 설계뿐만 아니라 요구분석, 구현 등 시스템 개발의 모든 단계에서 AOP 개념을 숙지하면서 개발을 진행하여야 한다. 셋째, AOP가 어떻게 동작하고, AOP를 도입하면 우리 설계에 어떤 변화가 생기는지 정확하게 이해하고 있어야 한다. 충분한 사전 테스트와 AOP에 대한 정확한 이해가 설계에 AOP를 효과적으로 적용시킬 수 있는 발판이 된다[14].

따라서 AOP를 이용한 시스템을 효과적으로 구축하기 위해서는 해당시스템을 철저히 분석하여 cross-cutting concern을 찾아내야 한다.

보안 시스템의 요구사항 중에서 직원 위치 파악, 직원 출입 관리, 출입문 관리등 그 하나하나를 분리하여 구현해야 하는 기능은 보안 시스템의 core concern에 해당되고, 감시, 로깅 추적, 인증, 권한부여 등은 시스템 전체에 걸쳐서 구현해야 하는 기능들은 cross-cutting concern 이다.

보안시스템에 AOP방법론을 적용해 설계하는 절차는 그림 2 와 같다.



[그림 2] AOP 방법론을 이용한 보안 시스템 구축 절차[15]

먼저 보안 시스템이 작동하게 될 환경과, 사용자의 요구사항 등을 고려해 모델을 설계한다. 그 후 비즈니스 로직과 보안 로직을 분리하여 개발하기 위해, Aspect 모듈로서 분리할 보안 로직 모델을 설계하고, 시스템의 비즈니스 로직을 담당하는 기본 모델을 설계한다. 이후 aspect 모듈과 기본 모듈이 혼합되는 aspect weaving 과정을 거쳐, 보안 시스템 전체 아키텍처 모델이 완성된다.

3.1 Case Study

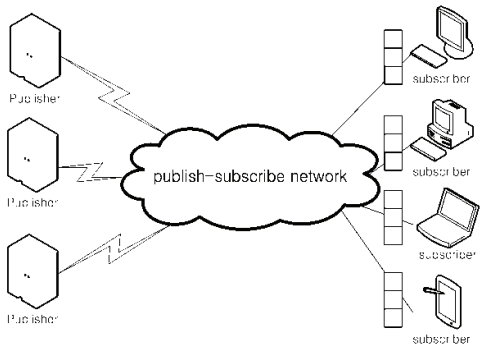
본 논문에서는 원자력 발전소 출입 보안 시스템을 설계·구현하였다. 원자력 발전소는 그 시설의 특성상 엄격한 출입 관리가 이루어진다. 발전소내의 모든 시설의 출입 관련 사항은 누락되지 않고 실시간으로 모니터링 되고 기록되어져야 한다. 직원들의 업무와 직위에 따라, 출입 가능한 시설이 달라지고, 관리자가 원할 시에는 시스템 상에서 쉽게 출입 권한을 변경할 수 있어야 한다. 4곳의 원자력 발전소 마다 물리적으로 존재하고 있는 출입 관련 시설물이 상이하게 때문에, 출입 보안 시스템 상에서 추가와 삭제가 자유로워야 한다.

원자력 발전소의 이러한 특성을 고려하여, 단순히 AOP 방법론만을 적용하기 보다는, 출입 보안 시스템의 필수 요소인 core concern은 시스템의 안정적인 성능을 위해 디자인패턴을 이용해 설계하고, 각각의 발전소 마다 변경 될 수 있는 구성요소들은 AOP 방법론을 적용하여, 시스템 구축시 기설계된 디자인패턴 구조를 바꾸지 않고서도, 각 발전소의 특성에 맞게 쉽게 변경 가능하도록 구

현하였다.

이에 본 논문에서는 원자력 발전소 출입 보안 시스템의 여러 기능들 중 필수 구성요소인 실시간 모니터링과 로깅(Logging) 시스템을 설계·구현하였다.

실시간 모니터링 시스템의 전체적인 구조는 그림 3과 같다.



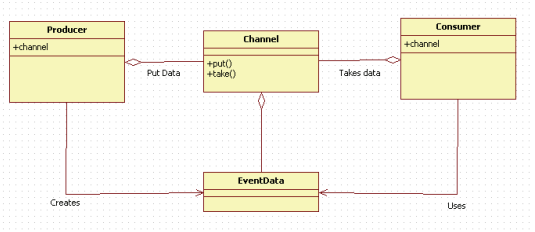
[그림 3] 실시간 모니터링 시스템 아키텍처

물리적으로 여러 대인 Publisher(출입시설, 경보시설)에서 직원출입, 경보 정보와 같은 이벤트가 발생하면 Publisher에 등록된 다양한 종류의 Subscriber들이 그 이벤트를 받아 내부 처리과정을 거쳐 사용자에게 실시간으로 전달한다. Subscriber의 종류가 다양해서 그 내부 처리 속도가 상이하기 때문에 Publisher로부터 받은 이벤트 데이타를 임시 보관하기 위한 버퍼(Buffer)를 추가하였다.

실시간 모니터링 시스템은 다음의 사항을 고려해서 설계해야 한다.

- Publisher에서 이벤트 발생 시 빠른 시간 안에 사용자에게 모니터링이 되어야 하며, 데이터의 유실이 발생하면 안 된다.
- 많은 이벤트가 발생하더라도 모니터링 시스템 때문에 시스템이 과부하 되어 전체 보안시스템이 다운되지 않도록 메모리 소모를 고려해서 설계되어야 한다.
- Publisher에서 발생하는 이벤트의 속도와 Subscriber에서 이를 처리하는 속도가 달라, 비동기적으로 작동해야 하기 때문에 이를 고려하여 설계해야 한다.

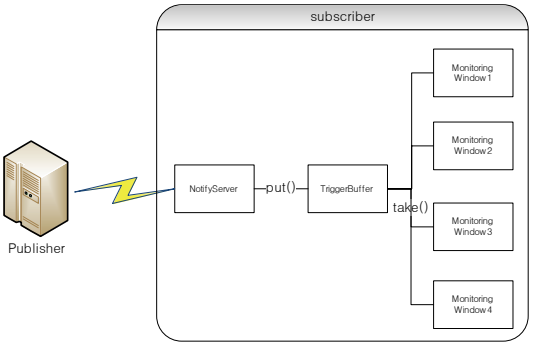
이러한 요구사항을 충족시키기 위해 디자인 패턴 중 Producer-Consumer 패턴을 이용하였다[16]. 이 패턴은 시스템 구조가 생산자(Producer)와 소비자(Consumer)와의 관계일 때 활용할 수 있는 패턴이다. 생산자와 소비자의 처리속도가 달라, 비동기 처리가 발생할 때 효과적 이다. 생산자와 소비자 사이에 버퍼 역할을 하는 채널을 두어 비동기 처리를 지원한다. 일반적인 Producer-Consumer 패턴의 구조는 그림 4와 같다.



[그림 4] Producer-Consumer 패턴[16]

Producer 클래스에서 발생한 이벤트 데이터가 Channel 클래스에 임시저장(put())되고, 준비된 Consumer 클래스가 Channel 클래스에 있는 이벤트 데이터를 비동기적으로 가지고 오는(take()) 구조이다. Channel 클래스가 버퍼 역할을 하여, 비동기적인 처리를 가능하게 한다.

실시간 모니터링 시스템에 앞서 설명한 Producer-Consumer 패턴을 적용해서 구현한 각각의 Subscriber가 작동하는 모습은 다음과 같다.



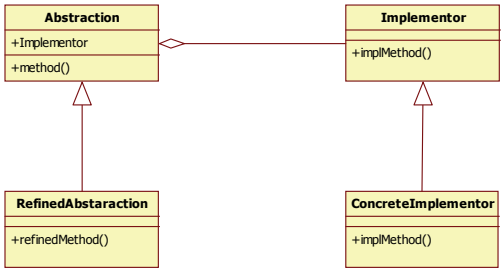
[그림 5] Subscriber 클래스 다이어그램

Publisher에서 받은 이벤트를 Producer-Consumer 패턴의 Channel 클래스역할을 하는 TriggerBuffer 클래스에서 보관 하고 각 이벤트 데이터를 이벤트 종류에 따라 구분하여 사용자가 실시간으로 확인할 수 있도록 각각의 모니터링 창(그림 5는 4개의 창)에 뿌려준다.

전달받은 이벤트를 화면으로 보여주는 창의 모양과 기능은 Bridge 패턴[16]을 적용해 설계했다. 각각의 창은 전달된 Event를 보여주는 서로 공통적인 기능을 하면서도, 세부적으로는 각자의 역할에 따라 전송된 데이터를 검

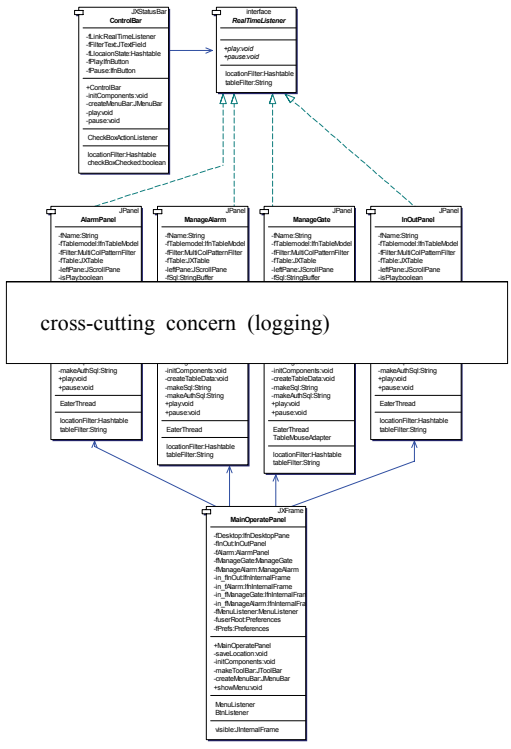
색하고, 필터링 하는 등의 다른 기능을 수행하여야 한다. 따라서 디자인 패턴 중에서 기능 클래스와 구현 클래스를 분리하여 설계 할 수 있는 Bridge 패턴 그림 6을 사용하였다.

Abstraction 클래스는 기능 클래스 최상위 클래스로서 기본적인 기능만을 기술한다. RefiendAbstraction 클래스는 Abstraction 클래스 외의 기능들을 추가한다. Implementor 클래스는 구현 클래스 계층의 최상위 클래스로 Abstraction에서 기술된 기능들을 구현하기 위해 정의하고 ConcreteImplementor 클래스에서는 위에서 정의된 기능들을 실제 구현한다.

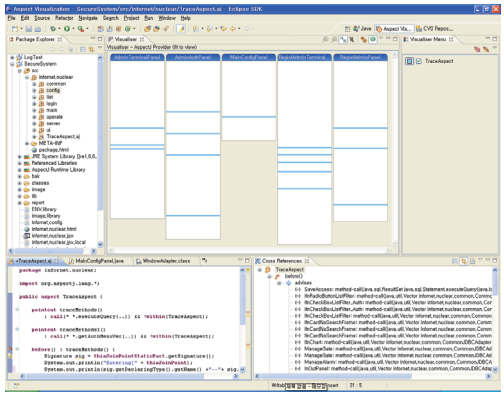


[그림 6] Bridge 패턴[16]

최종적으로 Producer-Consumer 패턴과 Bridge 패턴을 사용하여 실시간 모니터링 시스템을 그림 7과 같이 설계·구현하였다. RealTimeListener 클래스가 Bridge 패턴의 Implementor 역할을 하고, 4개의 창을 나타내는 AlarmPanel, ManageAlarm, ManageGate, InOutPanel 클래스가 ConcreteImplementor 역할을 한다. 그림 7의 클래스 다이어그램을 보면 Object Oriented 관점으로 잘 정의된 것처럼 보인다. 하지만 이렇게 구현된 클래스에 보안 관련 cross-cutting concern 기능인 로깅 기능을 추가하기 위해서는, 로깅 기능이 필요한 각각의 클래스마다 로깅 관련 코드를 삽입해야만 한다. 실제로 로깅 관련 코드를 aspect 모듈로 분리해 내지 않고 구현 했을 때의, 클래스 다이어그램을 Eclipse AspectJ 툴[11]로 분석한 결과 그림 8과 같다. 그림 8의 각각의 큰 직사각형이 하나의 클래스이고 그 안의 줄들이 로깅 관련 소스들이다. 그림에서 볼 수 있듯이 로깅관련 코드들이 여러 클래스에 걸쳐 산재되어 존재한다. 이렇듯 보안 로직이 비즈니스 로직과 함께 클래스에 구현되면 해당 클래스의 기능이 불명확해진다. 또한 개발 이후에 사용자들의 추가 요구사항에 대해 시스템 코드를 쉽게 변경할 수 없는 유연하지 못한 구조가 된다.



[그림 7] 실시간 모니터링 시스템 핵심 클래스 다이어그램



[그림 8] 로깅 코드(AOP 적용전)

따라서 로깅같은 cross-cutting concern을 AOP를 이용해 그림 9와 같이 aspect 모듈로 분리해서 구현하였다[17]. AspectJ로 짜여진 이 소스는 기존 코드에서 생성자와 객체가 생성될 때 해당객체가 생성됨을 알 수 있도록 객체의 이름을 출력하는 로깅 기능을 하는 소스이다. AspectJ 소스를 적용한 후, 그림 10에서 보듯이 클래스에 존재하는 로깅관련 코드의 수가 줄었음을 확인 할 수 있다.

AOP 방법론을 적용한 후, 그림 7의 핵심 클래스에 존재하는 로깅관련 코드가 그림 10과 같이 16개에서 5개로 약 70% 줄었음을 확인 할 수 있다. 이를 통해 핵심 모듈이 경량화 되었다.

이렇게 기존 OOP 환경에서 cross-cutting concern을 aspect 모듈로 분리함으로써, 비즈니스 로직과 보안 로직이 완벽하게 분리하여 구현 할 수 있다. 또한 디자인 패턴을 적용해 시스템을 OOP 환경으로 안정적으로 설계하고, 디자인 패턴에서 구현할 수 없었던 cross-cutting concern 같은 부족했던 부분을 AOP 방법론을 이용해 보완함으로써 이 시스템의 각 클래스들은 책임 소재가 명확해 질뿐만 아니라, 사용자의 추가 요구사항에 대해서도 유연하게 대처 할 수 있는 구조의 시스템 설계·구현이 가능하다.

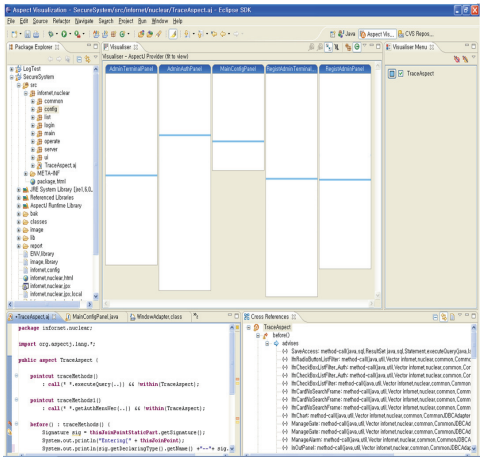
```
import org.aspectj.lang.*;

public aspect TraceAspect {
   pointcut traceMethods() : (execution(* *.*(..)) || execution(*.new(..))) && !within(TraceAspect);

   before() : traceMethods() {
      Signature sig = thisJoinPointStaticPart.getSignature();
      System.out.println("Entering [" + sig.getDeclaringType().getName() + "." + sig.getName() + "]");
   }

}
```

[그림 9] Logging 관련 Aspect module[17]



[그림 10] 로깅 코드(AOP 적용후)

4. 결 론

본 논문에서는 Aspect Oriented 방법론과 디자인 패턴을 활용해 원자력 발전소 출입 보안 시스템을 개발하였다. 출입 보안 시스템의 필수 요소인 직원 출입 관리와 같이 원자력 발전소 공통적으로 구현해야 하는 기능과 발전소 각각의 특성에 맞게 커스터마이징 되어야 할 기능이 존재하였다. AOP를 이용하여 보안시스템을 개발한 결과, 가장 큰 장점은 바로 비즈니스 로직과 보안 로직의 완전한 분리이다. Cross-cutting concern을 aspect 모듈로 하나하나 분리해서 개발할 수 있었기 때문이다. 실제로 한곳의 원자력 발전소에서 성공적으로 구축된 시스템은, 새로 적용될 발전소의 구성에 맞는 간단한 커스트마이징 작업을 하면, 핵심 모듈과 기본 보안 모듈의 변경 없이 빠른 구축이 가능하다. 이렇게 비즈니스 로직과 보안 로직이 완전하게 분리되게 구축함으로써, 이 시스템은 다음과 같은 장점을 가진다. 시스템 개발자는 시스템 설계 시 보안관련 사항을 독립된 모듈로써 따로 구분하여 설계할 수 있으므로, 시스템 기본 모듈을 중심으로 구조적인 설계가 가능하다. 개발자는 OOP를 이용해 개발하는 것 보다 시스템의 기능들을 모듈화 할 수 있어, 명확한 역할로 구분된 소스를 구현할 수 있다. 개발자는 사용자의 지속적인 변경 요구사항에 대해 좀 더 유연하게 대처할 수 있다. 또한 AOP의 본래의 장점인 코드 재사용성의 확대, 효율적인 기능 구현 등 기민한 시스템 구축이 가능하다.

또한 여려 기존연구를 통해 그 효과가 검증된 디자인 패턴과 AOP를 시스템 설계에 혼합하여 적용함으로써, 시스템의 성능과 효율이 중요한 고려대상이 되는 보안 시스템의 안정성을 향상시키고, 명확한 설계가 가능하다.

참고문헌

[1] Gregor Kiczales, Erik Hilsdale, Jim Hugunin, Mik Kersten, Jeffrey Palm and William G. Griswold, “An Overview of AspectJ”, ECOOP 2001, LNCS 2072, pp. 327-353, 2001.

[2] Charles B. Haley, Jonathan D. Moffett, Robin Laney, and Bashar Nuseibeh. “A framework for security requirements engineering.”, ACM In SESS’06, Shanghai, China, May 2006.

[3] Paolo Falcarin, Maurizio Morisio. "Developing Secure Software and Systems", IEC Network Security: Technology Advances, Strategies, and ChangeDrivers, 2004.

[4] Bart De Win, Frank Piessens, and Wouter Joosen. “How secure is AOP and what can we do about it.”, ACM, In SESS’06, Shanghai, China, May 2006.

[5] John Viega, J.T. Bloch, and Pravir Chandra, “Applying Aspect-Oriented Programming to Security”, Cutter IT Journal, Vol. 14, No.2, pp31-39, February 2001.

[6] Geri Georg, Indrakshi Ray, Robert France, “Using Aspects to Design a Secure System”, Proceedings of the Eighth IEEE international Conference on Engineering of Complex Computer Systems (ICECCS’02), 2002.
[7] Bart De Win, Wouter Joosen and Frank Piessens, “Developing secure applications through aspect-oriented programming. In Aspect-Oriented Software Development”, pages 633–650. Addison- Wesley, Boston, 2005.
[8] Jan Hannemann, Gregor Kiczales, “Design Pattern Implementation in Java and AspectJ”, OOPSLA 2002.
[9] C.PPfleegler, “Security in Computing”, 2nd Edition. Prentice-Hall, 1997.
[10] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Videira Lopes, Jean-Marc Loingtier, and John Irwin, "Aspect-Oriented Programming", ECOOP, pp220-242, 1997
[11] Adrian Colyer, Andy Clement, George Harley, and Matthew Webster, “Aspect-Oriented Programming with AspectJ and the Eclipse AspectJ Development Tools”, Pearson Endcation, Inc. 2005.
[12] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal., "Pattern-oriented software architecture: A system of patterns.", Wiley, 1996.
[13] E. Gamma, R. Helm, R. Johnson, and J. Vlissides., "Design Patterns: Elements of Reusable OO Software.", Addison-Wesley, 1995.
[14] Roger Alexander, "The Real Costs of Aspect-Oriented Programming", IEEE software, Vol.20, No.6, pp. 92-93, 2003.
[15] Yu, H. et. Al., “Secure Software Architectures Design by Aspect Orientation”, In Proc. 10th Int’l Conf. on Eng. Of Complex Computer Sys (ICECCS’05), pp. 45-57, 2005.
[16] Hiroshi Yuki, “Java gengo de manabu design pattern nyumon multi thread hen”, Softback publishing, 2002.
[17] Ramnivas Laddad, "AspectJ in Action: Practical Aspect-Oriented Programming", Manning Publications, 2003.

김 태 호(Taeho Kim) **[정회원]**

- 2006년 2월 : 고려대학교 산업시스템정보공학과 학사
- 2008년 2월 : 고려대학교 산업시스템정보공학과 석사
- 2008년 3월 ~ 현재 : 고려대학교 정보경영공학과 박사과정

<관심분야>
시스템 통합, 정보시스템설계

천 현 재(Hyeonjae Cheon) **[정회원]**

- 1997년 2월 : 인천대학교 산업공학 학사
- 1999년 2월 : 고려대학교 산업공학 석사
- 2006년 2월 : 고려대학교 산업공학 박사
- 2009년 3월 ~ 현재 : 고려대학교 정보보호연구원 연구교수

<관심분야>
SCM, 데이터마이닝, 시뮬레이션

이 홍 철(Hongchul Lee) **[정회원]**

- 1983년 2월 : 고려대학교 산업공학 학사
- 1988년 2월 : Univ. of Texas 산업공학 석사
- 1993년 2월 : Texas A&M Univ. 산업공학 박사
- 1996년 3월 ~ 현재 : 고려대학교 정보경영공학부 교수

<관심분야>
SCM, 생산 및 물류 정보시스템