DOI: 10.3745/KIPSTD.2008.15-D.5.669

애스펙트와 목표의 결합정보 중심 애스펙트 명세 기법

최 윤 석[†] · 정 기 원[††]

요 약

로깅이나 보안 등과 같은 횡단관심사를 효과적으로 모듈화하고 적용하는 관점지향 프로그래밍에 대한 다양한 연구가 진행되고 있다. 그러나 횡단관심사를 모듈화 한 애스펙트의 정보 및 애스펙트와 목표의 결합 정보를 명세하는 기법에 대한 연구가 필요한 상황이다. 본 논문에서는 애스펙트와 목표 모듈의 결합 관계를 명확히 하는 애스펙트 명세 기법을 제안한다. 제안한 기법은 애스펙트 명세, 우선순위 결정, 결합정보 명세, 그리고 교차점 명세 단계를 수행하여 애스펙트 정보를 명세한다. 애스펙트와 목표 모듈 사이의 결합 관계를 표현하는 결합정보 매트릭스를 기술하며, 결합점의 공통성을 분석하여 교차점 명세서를 기술한다. 제안한 명세기법은 애스펙트 및 애스펙트와 목표 모듈의 결합에 대한 구체적인 정보를 제공하며, 이를 통해 애스펙트 구현에 필요한 상세한 정보를 사용할 수 있다.

키워드 : 관점지향 프로그래밍, 횡단관심사, 애스펙트 명세, 교차점

A Specification Technique for Aspects Focusing on Join Information Between Aspects and Targets

Yunseok Choi[†] · Kiwon Chong[††]

ABSTRACT

There are various studies about AOP(Aspect-Oriented Programming) which modularizes cross-cutting concerns like logging and security effectively. But, we need to utilize techniques which specify the information of aspects modularizing cross-cutting concerns and detailed join information between aspects and targets. We propose a specification technique for aspects which focuses on clarifying the join information between aspects and targets. The technique includes the activities of specifying aspects, defining priority, specifying join information, and specifying pointcuts. We describe the join matrix which represents relationships of aspects and targets and the pointcut specification which is made by analyzing the commonality of join points. The proposed specification technique supports detailed information of the aspects and the join information between aspects and targets so that we can use detailed information to implement aspects.

Keywords : Aspect-Oriented Programming, Cross-Cutting Concerns, Aspect Specification, Pointcuts

1. 서 론

소프트웨어 시스템 개발 시 관심사의 분리(separation of concerns) 원칙에 따라 시스템의 관심사는 함수 또는 클래스 등과 같은 모듈로 구조화된다. 로깅이나 보안 등과 같이 여러 다수의 모듈이나 시스템 전반에 걸쳐 기능이 사용되는 횡단관심사(cross-cutting concern)를 함수 또는 클래스로 모듈화 할 경우 횡단관심사의 기능을 필요로 하는 다른 모듈에 구현 정보가 혼재되고(scattering) 얽히게 되는(tangling) 문제가 발생한다[1, 2]. 그 결과 시스템 구조의 이해가 어렵게 되고 결과적으로 유지보수 및 확장, 그리고 각 모듈의 재사용이 어려워진다. 이러한 문제를 해결하기 위하여 관점지향 프로그래밍(Aspect-Oriented Programming, AOP)이 등장하였다. 관점지향 프로그래밍은 횡단관심사를 애스펙트(aspect) 모듈 단위로 개발한 후 기존 개발방법의 모듈 단위로 개발한 목표와 컴파일 시, 클래스 로딩 시, 또는 실행 시 필요에 따라 결합(weaving)하는 방법으로 기능을 수행한다. 관점지향 프로그래밍을 적용할 경우 횡단관심사와 그 외의 관심사를 독립적인 모듈 단위로 개발할 수 있어 유지보수성 및 확장성, 그리고 재사용성 향상을 기대할 수 있다.

이에 관점지향 프로그래밍을 시스템 개발에 적용하기 위한 다양한 연구가 진행되고 있다. 개발초기에 횡단관심사를 식별하고 분석하기 위한 관점지향 요구공학(Aspect-Oriented Requirement Engineering, AORE)관련 연구[3]와 설계 및 구현 수행 중 애스펙트를 식별하는데 초점을 둔 애스펙트 마

※ 본 연구는 숭실대학교 교내연구비 지원으로 이루어졌음.
† 정 회 원 : 동덕여자대학교 정보학부 전임강사
†† 종신회원 : 숭실대학교 컴퓨터학부 교수
논문접수 : 2008년 8월 8일
수 정 일 : 1차 2008년 9월 8일
심사완료 : 2008년 9월 17일

이닝(mining) 연구[4] 등, 다양한 관점지향 소프트웨어 개발(Aspect-Oriented Software Developments, AOSD) 연구가 진행되고 있다. 그러나 애스펙트 구현에 필요한 정보를 명세하는 표준화 한 명세서 및 명세지침에 대한 연구는 부족한 상황이다[5]. 특히 관점지향 프로그래밍 기반 시스템의 구조를 이해하기 위해 필수적인 애스펙트와 목표의 결합정보에 대한 명세 기법 연구는 부족한 상황이다. 애스펙트와 목표 사이의 결합정보를 표현하는 결합점(join point) 및 교차점(pointcut)에 대한 정확한 명세 없이 시스템을 개발할 경우 실행결과 확인을 통해서만이 결합이상 유무를 확인할 수 있으므로 시스템 구조의 이해가 어려워질 수 있다. 또한 새로 개발하는 애스펙트의 적용 범위 및 우선순위 결정을 위해 기존 애스펙트의 결합정보를 분석하는 추가적인 노력이 요구될 수 있다. 이처럼 애스펙트와 목표 사이의 결합정보에 대한 구체적인 명세가 없을 경우 관점지향 프로그래밍 적용 의도와 달리 시스템의 유지보수 및 확장, 그리고 재사용이 어려워질 수 있다.

본 논문에서는 애스펙트 정보와 더불어 애스펙트의 결합정보에 초점을 둔 명세 기법을 제안한다. 제안한 기법은 애스펙트 명세, 우선순위 결정, 결합정보 명세, 그리고 교차점 명세 단계를 수행하며 애스펙트를 명세한다. 애스펙트의 구현 클래스 및 메소드, 적용 시점, 애스펙트의 적용 우선순위 등을 명세한다. 목표 모듈과 애스펙트의 관계를 매트릭스 형태로 정리하여 목표 모듈에 대한 애스펙트의 적용 범위를 분석하고 결합정보를 파악한다. 최종적으로 애스펙트의 결합대상을 나타내는 교차점 명세서를 작성한다. 교차점 명세서는 애스펙트 결합 대상을 나타내는 표현식을 포함한다.

논문의 구성은 다음과 같다. 2장에서는 관점지향 소프트웨어 개발에 대한 연구와 기존의 횡단관심사 및 애스펙트 명세 방법에 대하여 알아본다. 3장에서는 본 논문에서 제안한 애스펙트 명세 기법의 절차와 지침, 그리고 절차 수행을 통해 얻어지는 명세서를 제시한다. 4장에서는 제안한 기법을 전자요금징수 시스템 시뮬레이터에 적용한 사례를 살펴보며, 5장에서는 제안한 기법의 유용성을 토의한다. 6장에서 결론을 맺는다.

2. 관련 연구

2.1 관점지향 소프트웨어 개발

관점지향 소프트웨어 개발방법은 객체지향 소프트웨어 개발 방법을 보완하여 보다 높은 수준의 모듈화, 확장성, 이해성 그리고 재사용성을 지원한다[6]. 관점지향 소프트웨어 개발방법 연구는 개발 단계 초기에 횡단관심사를 식별하고 이를 애스펙트로 개발하는 방법인 관점지향 요구공학 관련 연구[7-9]와 기 개발한 코드로부터 애스펙트를 추출하는 애스펙트 마이닝 연구[4, 10] 등으로 구분할 수 있다.

관점지향 요구공학에서의 횡단관심사 식별 연구는 분석 단계에서 요구사항 산출물을 분석하여 횡단관심사를 식별하고 이를 애스펙트로 모듈화 하는 것에 초점을 맞추고 있다. 횡단관심사 식별은 주로 요구사항 분석과 아키텍처 설계 단계에서 이루어진다. 횡단관심사 식별 관련 연구는 횡단관심사를 식별 방법과 식별한 횡단관심사를 표현하는 모델을 제시하고 있으나 모델의 표준화에 대한 연구가 필요한 상황[11]이며 분석 단계에 기반하고 있으므로 구체적인 설계 및 구현정보가 요구되는 애스펙트 명세 정보는 제공하지 않고 있다. 애스펙트 마이닝 관련 연구의 경우 기존의 레거시 코드로부터 애스펙트를 추출하는 방법이므로 구현 단계를 중심으로 수행하며, 분석 및 설계 단계에서 필요한 애스펙트와 횡단관심사와의 관계 정보 등에 대한 명세는 제시하지 않고 있다.

2.2 횡단관심사 및 애스펙트 명세 기법

연구 [12]에서는 UML을 활용한 횡단관심사 표현 방법을 제시하고 있으며 횡단관심사를 명세하기 위한 템플릿을 제공하고 있다. 사용사례도 상에서 스테레오 타입을 사용하여 횡단관심사에 해당하는 사용사례와 목표에 해당하는 사용사례의 관계를 표현하며, 순차도를 사용하여 횡단관심사의 수행 절차를 표현한다. 이 연구는 요구사항 분석 수준에서의 횡단관심사를 표현하며 결합점 및 교차점에 대한 구체적인 명세는 다루지 않고 있다.

연구 [13]에서는 객체지향 시스템의 분석과 설계 절차에 애스펙트를 반영하는 방법을 제시하고 있다. 사용사례도 상에서 횡단관심사를 식별하여 애스펙트 테이블에 명세하고 애스펙트 명세 테이블에 애스펙트의 상세정보를 기술한다. 애스펙트 자체에 대한 명세 기법은 제공하고 있으나 결합점과 교차점 정보는 구체적 기술 지침 없이 서술식으로 기술하므로 결합정보를 명확하게 표현하지 못하는 문제점이 있다.

연구 [5]는 애스펙트와 목표 모듈인 핵심 클래스의 관계를 표현하기 위해 교차점, 결합점 및 충고 정의에 대한 명세와 지침을 제시하였다. 제시한 방법은 횡단관심사 명시, 횡단관심사 상세화, 그리고 코드 구현의 단계를 수행한다. 시스템의 모든 횡단관심사를 기록한 전체 횡단관심사 테이블을 기반으로 핵심 클래스 참조 테이블과 애스펙트 클래스 참조 테이블을 명세하며 작성한 산출물을 토대로 한 애스펙트 구현 방법을 제시하고 있다. 애스펙트와 목표의 결합정보는 애스펙트 참조 테이블과 핵심클래스 참조 테이블의 상호참조를 통해 파악하며, 실행기반 교차점 선언 방법을 사용하여 결합정보에 대한 교차점 명세를 작성한다. 이 연구는 횡단관심사의 분석에서 애스펙트 구현에 이르는 체계적인 절차 및 지침을 제공하나 애스펙트 명세가 표현할 수 있는 충고점의 개수가 제한적이며, 실행기반 교차점 선언 방법을 통해 생성한 교차점 표현식은 결합대상의 공통성을 반영하지 못하고 있다.

본 논문에서는 애스펙트의 구현 시 고려해야 할 애스펙트의 정보 및 애스펙트와 목표 결합의 구체적인 정보를 제공할 수 있는 애스펙트 명세 기법을 제안한다.

3. 애스펙트 명세 기법

애스펙트는 횡단관심사를 모듈화 한 것이므로 애스펙트

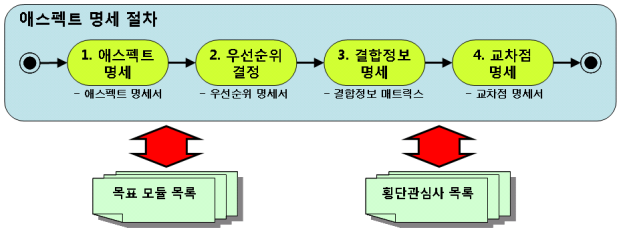
명세 전에 횡단관심사가 우선적으로 식별되어야 한다. 횡단관심사는 요구사항 명세서 분석 및 사용사례 분석, 아키텍처 분석 또는 설계 및 구현 시 개발자의 판단으로 식별할 수 있다. 본 논문은 이미 식별한 횡단관심사를 애스펙트로 개발하기 위해 필요한 명세서 및 명세기법이 주요 관심사이므로 목표 모듈 및 횡단관심사의 식별은 기존 연구를 적용하여 완료한 상태로 가정하며, 식별한 정보를 목표 모듈 목록 및 횡단관심사 목록으로 재구성한 후 애스펙트 명세 기법을 적용한다.

3.1 애스펙트 명세 절차

(그림 1)은 애스펙트 명세 절차를 나타낸다.

제시한 애스펙트 명세 기법은 애스펙트 명세, 우선순위 결정, 결합정보 명세, 그리고 교차점 명세 단계를 수행한다. 명세 절차는 목표 모듈 목록과 애스펙트로 모듈화 할 횡단관심사 목록을 입력으로 필요로 한다. 앞서 언급한 바와 같이 목표 모듈과 횡단관심사는 식별되었다는 전제 하에 절차를 진행하며, 각각 <표 1, 2>의 형식으로 재구성한 형태를 사용한다. <표 1>은 모듈의 단위가 클래스일 경우의 목표 모듈 목록의 구성을 나타내며, <표 2>는 횡단관심사 목록의 구성을 나타낸다.

목표 모듈 목록은 목표 모듈의 명, 모듈 내의 속성 명, 그리고 모듈의 메소드 명을 기술하며, 기존의 모듈 설계 명세에서 해당 정보만을 추출하여 구성한다. 횡단관심사 목록은 횡단관심사의 식별 및 애스펙트 마이닝을 통해 식별한 정보를 토대로 횡단관심사의 명칭과 횡단관심사 요약을 추출하여 구성한다. 횡단관심사는 구현 단위가 아닌 분석 단위이므로 애스펙트 명세 절차를 거쳐 모듈 단위인 애스펙트로 변환하여야 한다. 제시한 애스펙트 명세 절차는 위의 두 목록 정보를 기반으로 진행한다.



(그림 1) 애스펙트 명세 절차

목표 모듈 목록과 횡단관심사 목록을 토대로 애스펙트 명세서, 우선순위 명세서, 결합정보 매트릭스, 그리고 교차점 명세서를 명세 절차에 따라 작성한다. 첫 번째 단계의 산출물인 애스펙트 명세서는 횡단관심사를 구현 단위인 애스펙트로 변환하고 관련 정보를 표현한다. 하나의 횡단관심사는 하나 이상의 애스펙트로 변환할 수 있다. 횡단관심사와 애스펙트의 관계를 추적하기 위해 애스펙트 명세에 어떤 횡단관심사를 모듈화 한 것인지 기록한다. 애스펙트 명세 후 우선순위 결정 단계를 수행하여 애스펙트가 목표에 적용되는 순서를 결정한다. 우선순위는 횡단관심사 우선순위, 애스펙트 우선순위, 그리고 애스펙트의 충고 우선순위로 구분할 수 있다. 애스펙트는 복수의 충고를 가질 수 있으므로 충고별로 우선순위를 결정하여 우선순위 명세서에 기록한다. 우선순위 결정은 분석 및 설계 단계에서 결정한 횡단관심사 및 애스펙트의 제약조건과 개발자의 판단에 따라 수행한다. 우선순위 결정 완료 후 충고를 소유한 애스펙트의 명세서에 우선순위를 반영한다. 다음 단계로 결합정보 명세를 수행하여 우선순위 정보를 반영한 애스펙트 명세의 충고와 목표 모듈의 결합점을 매트릭스 형태로 구성하고 결합 관계를 정의한다. 애스펙트 충고의 적용 시점 별로 하나 이상의 매트릭스를 작성한다. 최종적으로 애스펙트 명세와 결합정보 매트릭스를 기반으로 교차점 명세서를 작성한다. 애스펙트는 하나 이상의 교차점을 가질 수 있으므로 애스펙트 별로 모든 교차점을 명세한다.

애스펙트 명세 절차를 통해 산출한 애스펙트 명세서는 애스펙트 구현에 활용하며, 교차점 명세서는 애스펙트의 결합정보 구현에 활용한다. (그림 2)는 애스펙트 명세 단계에서 산출한 산출물 사이의 관계를 나타낸다. 3.2절부터 3.5절은 각 단계의 작업과 산출물의 구성에 대하여 기술한다.

3.2 애스펙트 명세

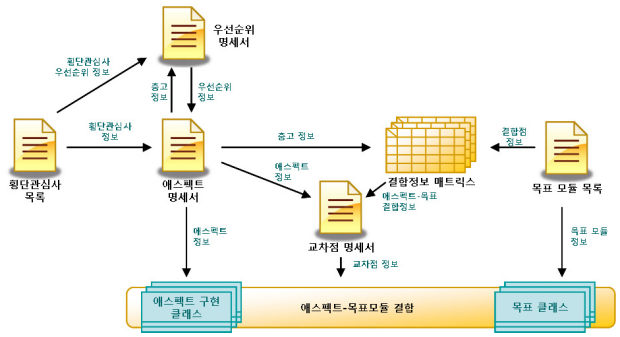
애스펙트 명세 기법의 첫 번째 단계인 애스펙트 명세는

〈표 1〉 목표 모듈 목록의 구성

목표 모듈 목록				
번호	패키지 명	클래스 명	속성 명	메소드 명
순번 기록	목표모듈의 패키지 명을 기술	목표 클래스 명을 기술	클래스의 멤버속성 명 기술	클래스의 멤버메소드 명 기술
			...	...

〈표 2〉 횡단관심사 목록의 구성

횡단관심사 목록		
번호	횡단관심사 명	요약
순번 기술	횡단관심사 명 기술	횡단관심사의 특징 및 제약사항 기술
...	...	...



(그림 2) 산출물 관계 개념도

횡단관심사를 구현하는 애스펙트를 명세하는 단계이다. 산출물은 애스펙트 명세서이다. 애스펙트 명세서는 애스펙트의 명, 애스펙트가 구체화 하는 횡단관심사의 명, 애스펙트의 구현 모듈 명, 그리고 애스펙트의 충고메소드 관련 정보로 구성한다. <표 3>은 모듈의 단위를 클래스 단위로 명세할 때 사용할 수 있는 애스펙트 명세서의 구성을 나타낸다.

애스펙트 명은 애스펙트 식별자를 의미하며 횡단관심사 명은 애스펙트가 구현하는 횡단관심사를 의미한다. 횡단관심사 명은 횡단관심사 목록에 나타난 횡단관심사 중 하나이며 이를 사용하여 횡단관심사와 애스펙트의 관계를 추적할 수 있다. 횡단관심사와 애스펙트의 대응관계는 일대일 또는 일대다가 될 수 있다. 구현모듈 명은 애스펙트를 구현할 모듈 명을 기록하며 클래스일 경우 풀네임(패키지.클래스명)을 기록한다. 특성 항목에는 애스펙트의 특성과 구현 시의 제약사항 등을 기록한다. 충고메소드는 횡단관심사의 기능을 제공하는 애스펙트의 멤버이며 메소드 명에 충고메소드 명을 기록한다. 결합시점에는 충고메소드가 목표 모듈에 결합하는 시점을 기록한다. 충고메소드의 결합 시점은 Before / After / Around 등으로 구분할 수 있다. Before는 목표 모듈의 기능 실행 이전에 충고메소드가 수행되는 것을 의미하며, After는 목표 모듈의 기능 수행 완료 후 충고메소드가 실행됨을 의미한다. Around는 목표 모듈의 기능 수행 전/후에 충고메소드를 실행됨을 의미한다. 제시한 결합시점 외에 목표 모듈의 기능 수행을 생략하고 충고메소드의 기능만을 수행하는 결합 시점과 목표 모듈이 예외가 발생할 때만 결합하는 시점 등을 추가할 수도 있다. 결합시점은 애스펙트 적용 우선순위를 결정 시에 애스펙트를 분류하기 위해 필요하다. 우선순위는 같은 결합시점을 갖는 충고메소드들의 목표 모듈에 대한 적용순서를 표현한다. 같은 결합시점을 갖는 모든 애스펙트들의 충고메소드들을 대상으로 우선순위를 할당하며, 애스펙트 명세 이후 우선순위 결정에서 결정한 우선순위를 참고하여 기록한다.

3.3 우선순위 결정

우선순위 결정은 목표에 대한 애스펙트 적용 순서를 결정하는 단계로서 산출물은 우선순위 명세서이다. 하나의 애스펙트는 여러 충고메소드를 가질 수 있으며 충고메소드 별로 우선순위를 할당한다. 우선순위는 분석 및 설계 단계에서 결정한 제약조건 및 개발자의 판단에 따라 부여한다. <표 4>는 우선순위 명세서의 구성을 나타낸다.

3.4 결합정보 명세

본 단계에서는 목표 모듈과 애스펙트 사이의 결합정보를 매트릭스로 작성한다. 가로축은 목표 모듈 명을 기록하며, 세로축은 애스펙트를 *애스펙트.충고메소드명* 형태로 기록한다. 단일 충고메소드를 갖는 애스펙트일 경우 애스펙트 명칭만을 기록한다. 목표 모듈과 애스펙트가 교차하는 지점에 애스펙트를 적용하는 결합점을 기록한다. 결합점은 구현 시 사용할 관점지향 프로그래밍 지원 도구에 따라 목표 모듈의 속성 참조문 또는 메소드 등이 될 수 있다[15,16]. 결합정보

〈표 3〉 애스펙트 명세서의 구성

애스펙트 명세서					
순번 기술	애스펙트명	애스펙트 명 기술			
	횡단관심사명	애스펙트가 구체화 하는 횡단관심사 명 기술			
	구현모듈 명	애스펙트를 구현한 클래스 명 기술			
	특성	애스펙트의 특성 및 기타 고려사항 기술			
	충고메소드	메소드 명	결합시점	적용 우선순위	특성
		충고메소드명 기술	충고메소드의 결합시점 기술 Before/After/Around 등	충고메소드의 적용 우선순위 기술 (우선순위 결정 후 기록)	충고메소드의 특성 및 고려사항 기술

〈표 4〉 우선순위 명세서의 구성

우선순위 명세서			
번호	결합시점	애스펙트 충고명	적용 우선순위
순번 기술	충고메소드의 결합시점 기술 애스펙트 명세에 기록한 내용을 참고	애스펙트가 갖고 있는 충고메소드 명 기술 애스펙트명.충고메소드명	동일한 결합시점을 갖는 애스펙트 적용 우선순위 기술
...	...	...	...

〈표 5〉 결합정보 매트릭스의 구성

결합정보 매트릭스			
결합시점	충고메소드의 결합시점 기술		
목 표 / 애스펙트	목표 모듈 명	목표 모듈 명	...
애스펙트명.충고메소드명	애스펙트가 적용될 목표 모듈의 결합점 명 ...	애스펙트가 적용될 목표 모듈의 결합점 명	...
...	...	...	...

〈표 6〉 교차점 유형과 범위 조합의 적용 대상

유형	범위	적용대상
메소드/속성	단일	특정 클래스의 특정 이름(전체/부분)에 해당하는 메소드/속성
클래스	단일	특정 패키지의 특정 이름(전체/부분)에 해당하는 클래스의 모든 메소드/속성
패키지	단일	특정 이름(전체/부분)에 해당하는 패키지의 모든 클래스의 모든 메소드/속성
메소드/속성	다중	임의의 클래스들이 소유한 동일한 이름(전체/부분)에 해당하는 메소드/속성
클래스	다중	임의의 패키지에 속한 동일한 이름(전체/부분)에 해당하는 클래스들의 모든 메소드/속성
패키지	다중	동일한 이름(전체/부분)에 해당하는 패키지의 모든 클래스의 모든 메소드/속성

〈표 7〉 교차점 명세서의 구성

번호	교차점 명세서		
순번 기술	교차점 명	교차점 명 기술 (pc_교차점명)	
	애스펙트 명		충고메소드 명
	교차점을 사용하는 애스펙트 명 기술		교차점에 의해 실행하는 메소드 명
	유형	범위	교차점 표현식
	교차점이 적용되는 유형 기술	교차점 표현식의 범위가 단일/다중 대상인지 기술	교차점을 표현하는 표현 식 기술
	상세정보	교차점의 특징 및 제약사항 목표 표현에 대한 구체적인 기술을 기록	

매트릭스는 충고메소드의 결합시점인 Before/After/Around 별로 작성하며, 필요에 따라 모듈의 집합 단위(클래스일 경우 패키지 단위)로 구분하여 작성한다. <표 5>는 결합정보 매트릭스의 구성을 나타낸다.

3.5 교차점 명세

이전 단계들의 산출물을 토대로 교차점 명세서를 작성한다. 교차점은 목표 모듈의 결합점 중 애스펙트가 적용되는 곳을 의미하며 결합규칙을 나타내는 표현식을 갖는다. 애스펙트의 충고메소드 별로 하나 이상의 교차점을 명세할 수 있다. 교차점 명 작성 시 *'pc_'* 접두어를 붙여 교차점임을 식별한다. 접두어 다음에는 *'애스펙트명목표모듈명'* 형식으로 애스펙트와 목표 모듈을 표현하여 작성한다. 특정 메소드 명 또는 속성 명에 적용하는 교차점일 경우 교차점 명 마지막에 '_메소드명/_속성명'을 추가한다. 구체적인 예는 적용사례에서 살펴보도록 한다. 교차점의 범위 항목에는 패키지/클래스/메소드/속성 중 하나의 값을 기록한다. 교차점의 결합범위가 패키지 단위일 경우 해당 패키지에 속한 클래스가 애스펙트의 결합후보가 되며, 클래스일 경우는 클래스의 결합점이 결합후보가 된다. 속성 또는 메소드일 경우 해당 결합점만이 애스펙트와 결합후보가 된다. 유형 항목은 애스펙트가 교차점 표현식에 나타난 항목에만 유일하게 적용되는지, 항목과 같은 명칭부분을 갖는 모든 대상에 적용되는지 구분하여 기록한다. <표 6>은 교차점의 범위와 유형의 조합에 따른 애스펙트의 적용 대상을 나타낸다.

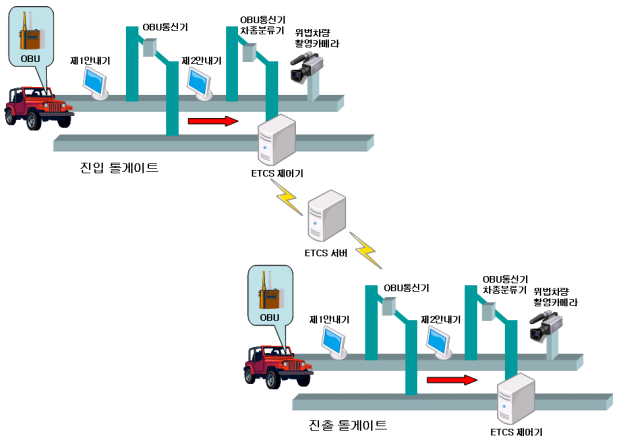
교차점 표현식은 애스펙트를 적용할 결합점 형태를 표현한다. 결합정보 매트릭스를 토대로 애스펙트가 적용되는 결합점들의 공통성을 분석하여 애스펙트 적용 범위 및 형태를 결정하고 이를 기반으로 표현식을 작성한다. 교차점 표현식의 대상은 패키지 명, 클래스 명, 메소드 명, 그리고 속성 명의 전체 또는 부분이 될 수 있다. 교차점 표현식은 자바의 정규 표현식[14] 또는 AspectJ[15]의 표현식 형태로 작성한다. 구체적인 표현식의 예는 적용사례를 통해 살펴보도록 한다. 교차점 명세서의 구성은 <표 7>을 통해 확인할 수 있다.

명세 절차 수행 후 산출한 명세서들을 토대로 애스펙트의 상세 설계 및 구현을 수행한다. 자바 언어의 애스펙트 개발용 확장인 AspectJ를 사용할 경우 명세서의 정보를 기반으로 애스펙트 및 교차점을 구현한다. 최근 의존관계 주입

(dependency injection)과 관점지향 프로그래밍 프레임워크로 각광받고 있는 스프링(Spring Framework)[16]을 사용할 경우 명세서의 정보를 바탕으로 애스펙트 관련 정보를 외부 설정파일로 작성하고 애스펙트를 개발한다.

4. 적용사례

본 논문에서는 지능형 교통 체계(Intelligent Transport System, ITS)의 기본 구성요소 중 하나인 전자요금징수 시스템(이하 ETCS) 시뮬레이터를 제안한 명세 기법을 적용하여 관점지향 프로그래밍 기반 시스템으로 개발하였다. (그림 3)은 ETCS의 개념도를 나타낸다.



(그림 3) ETCS 개념도

ETCS는 차량이 유료도로 또는 고속도로에 진입 및 진출 시 차량 내에 설치한 단말기인 OBU(Onboard Unit)와의 무선통신을 통해 OBU에 삽입되어 있는 통행료 충전 IC카드에서 통행료를 자동으로 징수하는 시스템이다[17]. ETCS 이용고객은 차량에 OBU를 장착한 후 통행료를 충전한 IC카드를 OBU에 삽입한 상태로 ETCS가 운영 중인 진입 톨게이트에 진입한다. ETCS제어기는 OBU통신기를 통해 차량 내에 설치한 OBU와 통신하여 OBU에 기록되어 있는 차종 정보 등을 파악하고 과금에 필요한 정보를 저장한다. 또 차종분류기를 통해 진입한 차량의 차종을 식별한다. 차량이 목적지의 ETCS 진출 톨게이트에 진입하면 ETCS 제어기는 OBU통신기와 차종분류기를 통해 획득한 차량의 정보와 ETCS 서버의 정보를 활용하여 OBU 내의 IC카드에서 운행거리 및 차종에 따른 통행료를 차감한다. 차량 운전자는 OBU의 디스플레이 및 ETCS의 안내기를 통해 통행료 정보를 살필 수 있다.

ETCS는 차량에 장착한 OBU와 OBU통신기 사이의 정보교환 시 고속암호화/복호화 처리를 수행하여야 하며, 각 장치들의 운용상태 이상 유무를 기록하여야 한다. 또한 통행료 처리 트랜잭션을 처리할 수 있어야 하며, 시스템의 상태를 모니터링 할 수 있어야 한다. 이러한 요소들은 횡단관심사로 식별할 수 있다. 이 외에 고속 주행 중인 차량과의 통신을 전제로 하고 있으므로 성능 향상과 관련한 다양한 횡단관심사를 식별할 수 있다. 적용사례에서는 이러한 횡단관심사와 목표 모듈을 제안한 명세 기법 절차의 입력인 목표 모듈 목록과 횡단관심사 목록으로 재구성하여 표현하였으며 <표 8, 9>는 그 일부를 보여준다.

〈표 8〉 ETCS 목표 모듈 목록

ETCS 목표 모듈 목록				
번호	패키지명	클래스명	속성명	메소드명
1	org.etcs.core	ETCSServer	Map trafficData	void add(SavedData)
				Map getTrafficData()
2	org.etcs.tollgate.controller	EntryController	N/A	void process()
				void updateObu()
3	org.etcs.tollgate.controller	ExitController	N/A	void process()
				void updateObu()
4	org.etcs.tollgate.device	CarSensor	N/A	void sensing(Car)
5	org.etcs.tollgate.device	ObuCommunicator	N/A	chargeTollgateFee(long)
				sensing(Car)
				connecting(Car)

〈표 9〉 ETCS 횡단관심사 목록

ETCS 횡단관심사 목록		
번호	횡단관심사명	요 약
1	LoggingConcern	ETCS의 모든 운영 상태를 기록
2	EncodingConcern	차량의 OBU에 송신할 정보를 암호화함
3	DecodingConcern	차량의 OBU로부터 수신한 정보를 해독함
4	MonitoringConcern	ETCS의 운영 정보를 출력함

4.1 애스펙트 명세

<표 8, 9>의 내용을 바탕으로 애스펙트 명세 단계를 수행하여 애스펙트 명세서를 작성하였다. 적용사례에서는 횡단관심사 목록의 횡단관심사 별로 각각 하나의 애스펙트를 작성하였으며 단일 충고메소드를 갖는 형태로 작성하였다. <표 10>은 ETCS의 애스펙트 명세서 일부를 나타낸다.

4.2 우선순위 결정

애스펙트 명세 후 다음 단계로 우선순위 결정을 수행하였다. <표 10>의 애스펙트만을 대상으로 할 경우 logging과 encoding에 각각의 결합시점 분류에서 가장 높은 우선순위를 할당할 수 있다. 본 적용사례에서는 전체가 아닌 일부 애스펙트에 대하여 명세를 수행하였으므로 우선순위 명세서를 별도로 작성하지 않았으며 애스펙트 명세서를 기반으로 우선순위를 결정하여 기록하였다.

4.3 결합정보 명세

우선순위 결정 후 결합정보 명세를 수행하여 결합정보 매트릭스를 작성하였다. 목표 모듈인 ETCSServer, EntryController, ExitController 그리고 ObuCommunicator를 상단 가로축으로 배치하였으며, logging, decoding, monitoring, 그리고 encoding 애스펙트를 결합시점 별로 분류하여 세로축에 배치하였다. 본 적용사례에선 하나의 애스펙트가 하나의 충고메소드를 갖고 있으므로 충고메소드 명은 생략하였다. 애스펙트가 목표 모듈에 적용될 경우 매트릭스의 교차지점에 애스펙트가 결합하는 결합점들을 기록하였다. ETCS의 결합정보 매트릭스는 <표 11>과 같다.

4.4 교차점 명세

목표 모듈 목록과 명세 절차를 통해 산출한 애스펙트 명세서 그리고 결합정보 매트릭스를 사용하여 교차점 명세서를

〈표 10〉 ETCS 애스펙트 명세서

ETCS 애스펙트 명세서					
1	애스펙트 명	logging			
	횡단관심사 명	LoggingConcern			
	구현모듈 명	org.etcs.aspect.LoggingAspect			
	특성	목표 모듈의 함수 명을 수행시간과 함께 기록			
	충고메소드	메소드명	결합시점	적용우선순위	특성
		log	Before	(우선순위 결정 수행 후) 1	목표 모듈의 수행상태를 기록
2	애스펙트 명	encoding			
	횡단관심사 명	EncodingConcern			
	구현모듈 명	org.etcs.aspect.EncodingAspect			
	특성	목표 모듈이 생성한 정보를 고속암호화 알고리즘을 사용하여 암호화 함			
	충고메소드	메소드명	결합시점	적용우선순위	특성
		encode	After	(우선순위 결정 수행 후) 1	목표 모듈의 결과를 암호화 암호화 오류 발생 시 예외사항 발생
3	애스펙트 명	decoding			
	횡단관심사 명	DecodingConcern			
	구현모듈 명	org.etcs.aspect.DecodingAspect			
	특성	목표 모듈이 전달 받을 정보를 고속복호화 알고리즘을 사용하여 복호화 함			
	충고메소드	메소드명	결합시점	적용우선순위	특성
		decode	Before	(우선순위 결정 수행 후) 2	목표 모듈의 입력을 복호화 복호화 오류 발생 시 예외사항 발생
4	애스펙트 명	monitoring			
	횡단관심사 명	MonitoringConcern			
	구현모듈 명	org.etcs.aspect.MonitoringAspect			
	특성	목표 모듈의 수행 상태와 처리 정보를 출력함			
	충고메소드	메소드명	결합시점	적용우선순위	특성
		monitor	After	(우선순위 결정 수행 후) 2	목표 모듈의 수행결과를 출력

〈표 11〉 ETCS 결합정보 매트릭스

ETCS 결합정보 매트릭스				
결합시점	Before			
목표 / 애스펙트	ETCSServer	EntryController	ExitController	ObuCommunicator
logging	void add(SavedData) Map getTrafficData()	void process() void updateObu()	void process() void updateObu()	void chargeTollgateFee(long) void sensing(Car) void connecting(Car)
decoding	void add(SavedData)	N/A	N/A	void sensing(Car) void connecting(Car)
결합시점	After			
목표 / 애스펙트	ETCSServer	EntryController	ExitController	ObuCommunicator
monitoring	void add(SavedData)	void process() void updateObu()	void process() void updateObu()	void chargeTollgateFee(long) void sensing(Car)
encoding	Map getTrafficData()	void updateObu()	void updateObu()	N/A

〈표 12〉 ETCS 교차점 명세서

번호	교차점 명세서		
1	교차점 명		pc_LoggingETCSServer
	애스펙트 명		충고메소드 명
	logging		log
	유형	범위	교차점 표현식
	클래스	단일	org.etcs.core.ETCSServer
	상세정보	ETCSServer의 모든 메소드 호출 정보를 log 기록으로 남기기 위한 logging 애스펙트의 교차점	
2	교차점 명		pc_LoggingController
	애스펙트 명		충고메소드 명
	logging		log
	유형	범위	교차점 표현식
	패키지	단일	org.etcs.tollgate.controller
	상세정보	org.etcs.tollgate.controller패키지에 속한 모든 클래스의 모든 메소드 호출 정보를 log 기록으로 남기기 위한 logging 애스펙트의교차점	
3	교차점 명		pc_DecodingETCSServer_add
	애스펙트 명		충고메소드 명
	decoding		decode
	유형	범위	교차점 표현식
	메소드	단일	org.etcs.core.ETCSServer.add
	상세정보	ETCSServer의 add() 사용 시 입력으로 전달되는 정보를 decoding 하기 위해 적용하는 decoding 애스펙트의 교차점	

작성하였다. <표 12>는 교차점 명세서의 일부를 나타낸다.

결합정보 매트릭스 상의 ETCSServer 목표 모듈은 모든 결합점이 logging 애스펙트와 결합하고 다른 목표 모듈의 명칭과 공통점이 없으므로 클래스 유형, 단일 범위 형태로 결정하였다. 클래스 모든 결합점에 결합하므로 교차점 명은 pc_LoggingETCSServer로 명명하였다. decoding 애스펙트의 경우 ETCSServer의 add() 메소드에만 적용되므로 교차점 명은 pc_DecodingETCSServer_add로 명명하였으며 유형은 메소드, 범위는 단일형태로 작성하였다. EntryController와 ExitController의 경우 두 목표 모듈의 모든 메소드가 logging 애스펙트의 적용을 받으며 해당 패키지에는 두 목표 모듈만 존재하므로 애스펙트를 패키지 단위로 적용할 수 있다. 패키지 단위의 적용은 명세 상의 정보를 통해서 결정하거나 '특정 패키지의 모든 모듈이 애스펙트 적용을 받아야

한다'와 같은 제약조건을 참고하여 결정한다. 두 Controller 모듈에 대한 교차점은 logging 애스펙트의 패키지 유형, 단일 범위 형태로 작성하였다.

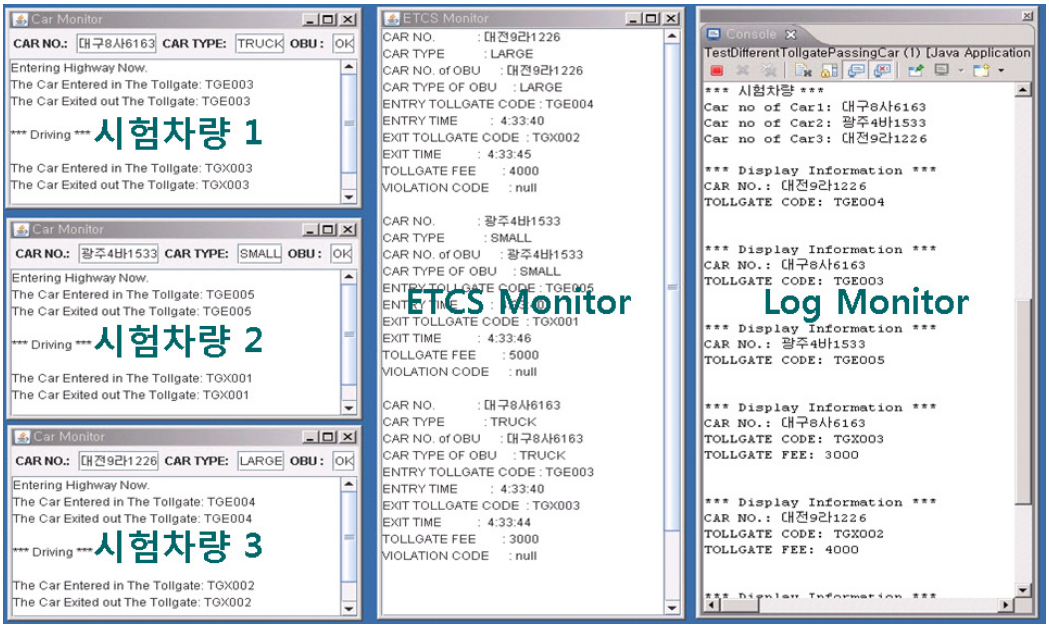
작성한 명세서들을 토대로 애스펙트를 설계하고 구현을 수행하였다. 관점지향 프로그래밍 을 지원을 위해 스프링을 선택하여 적용하였다. 스프링의 경우 다양한 관점지향 프로그래밍 지원 클래스들을 내장하고 있으나 본 적용사례에서는 스프링과의 결합도를 낮추기 위해 POJO(Plain Old Java Object) 기반으로 애스펙트를 구현하였다.

스프링의 애스펙트 설정 정보는 별도의 XML 파일로 작성한다. 애스펙트 설정 파일을 작성하기 위해 애스펙트 명세서를 기반으로 애스펙트 명, 충고메소드 명과 적용시점, 교차점의 적용 유형 및 범위를 작성하였고, 결합정보 매트릭스의 결합정보를 토대로 교차점 표현식을 작성하였다. 작성한 명세서로부터 애스펙트 구현 및 적용에 필요한 정보들을 제공받을 수 있었으며, 명세 정보에서 설계 및 구현 산출물의 정보로, 또 그 역으로의 정보 추적이 가능하였다. (그림 4)는 명세 기법의 산출물을 토대로 관점지향 프로그래밍을 적용하여 개발한 ETCS 시뮬레이터의 실행화면을 나타낸다.

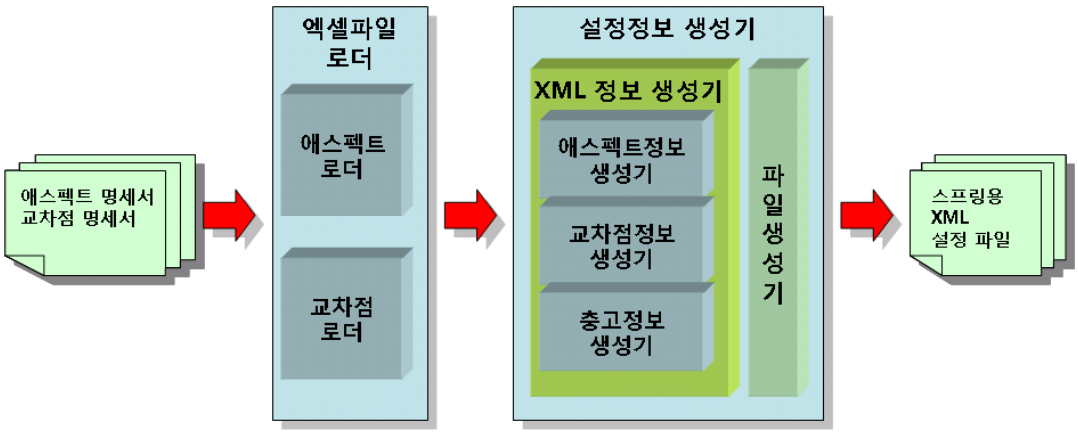
4.5 애스펙트 설정정보 자동생성도구의 적용

제안한 명세 기법의 산출물들이 애스펙트 정보 및 결합정보를 충분히 지원하고 있음을 확인하기 위해 산출물로부터 스프링의 애스펙트 설정정보를 자동으로 생성하는 도구를 개발하고 ETCS 시뮬레이터를 명세한 산출물에 적용하였다. 애스펙트 설정정보 자동생성도구는 마이크로소프트사의 엑셀(Excel) 파일포맷으로 작성한 산출물들을 입력 받아 스프링의 애스펙트 설정 정보 XML 파일을 생성한다. (그림 5)는 설정정보 자동생성도구의 개념도를 나타낸다.

자동생성도구의 엑셀파일로더는 엑셀 파일포맷인 애스펙



(그림 4) ETCS 시뮬레이터 실행화면



(그림 5) 애스펙트 설정정보 자동생성도구의 개념도

```
<bean id="loggingAspect" class="org.etcs.aspect.LoggingAspect" />
<bean id="encodingAspect" class="org.etcs.aspect.EncodingAspect" />
<bean id="decodingAspect" class="org.etcs.aspect.DecodingAspect" />
<bean id="monitoringAspect" class="org.etcs.aspect.MonitoringAspect" />

<aop:config>
   <aop:aspect id="logging" ref ="loggingAspect">
      <aop:pointcut id="pc_LoggingETCSServer" expression="bean(etcsServer)" />
      <aop:before pointcut-ref="pc_loggingETCSServer" method="log" />
      <aop:pointcut id="pc_LoggingController" expression="within(org.etcs.tollgate.controller.*)" />
      <aop:before pointcut-ref="pc_LoggingController" method="log" />
      <aop:pointcut id="pc_LoggingObuCommunicator" expression="bean(obuCommunicator)" />
      <aop:before pointcut-ref="pc_LoggingObuCommunicator" method="log" />
   </aop:aspect>

   <aop:aspect id="decoding" ref ="decodingAspect">
      <aop:pointcut id="pc_DecodingETCSServer_add" expression="execution(public void add(*))" />
      <aop:before pointcut-ref="pc_DecodingETCSServer_add" method="decode" />
```

(그림 6) 자동 생성한 ETCS 용 스프링 설정정보 XML 파일

트 명세서와 교차점 명세서를 입력받아 애스펙트와 애스펙트가 소유한 교차점을 분석한 후 각각의 정보를 객체화한다. 설정정보 생성기는 액셀파일 로더를 통해 객체화 한 애스펙트 목록을 전달 받은 후 스프링의 애스펙트 설정정보 항목으로 재구성한다. 애스펙트의 교차점 별로 충고정보를 생성하며 충고 별로 교차점 적용 규칙을 설정한다. 교차점의 우선순위는 스프링의 교차점 우선순위 표현방식에 따라 우선순위가 높은 교차점부터 순차적으로 출력될 수 있게 조정한다. 재구성된 애스펙트 관련 정보는 파일 생성기에 의해 스프링의 애스펙트 설정정보 XML 파일로 출력된다. (그림 6)은 적용사례의 ETCS 애스펙트 명세서와 교차점 명세서를 입력으로 자동 생성한 스프링용 애스펙트 설정정보 XML 파일의 일부를 나타낸다.

생성한 애스펙트 설정정보 XML 파일을 적용하여 적용사례에서 개발한 ETCS 시뮬레이터를 실행하여 보았으며, 직접 작성한 애스펙트 설정정보 XML 파일을 사용할 때와 다름없이 정상적으로 실행됨을 확인할 수 있었다.

5. 토 의

이 장에서는 본 논문에서 제안한 애스펙트 명세 기법의 유용성을 기존의 명세 기법과 비교하여 평가한다. 명세 기법의 유용성은 기법의 산출물을 대상으로 애스펙트 구현에 필요한 정보의 표현 유무와 구현 시의 산출물 활용성으로 구분하여 평가하였다. 정보 표현성은 애스펙트-목표의 관계 정보, 애스펙트 정보, 그리고 교차점 결합규칙에 대하여 명세 기법이 구현에 필요한 정보를 충분히 제공할 경우 '높음', 명세 정보 이외에 추가적인 정보가 필요할 경우 '보통', 정보를 제공하나 개발자의 판단에 의존적일 경우 '낮음', 관련 정보가 없을 경우 '없음'으로 평가하였다. 산출물 활용성은 정보 표현성에 기초하여 개발의 용이성, 수정의 용이성, 그리고 확장의 용이성에 대하여 '높음', '보통', 그리고 '낮음'으로 평가하였다. <표 13>은 제안한 명세 기법과 기존의 명세 기법에 대한 유용성 비교 및 평가를 나타낸다.

연구 [12]에서 제시한 UML 활용 기법의 경우 사용사례를 통해 횡단관심사와 목표 사이의 관계를 잘 표현하고 있으나 횡단관심사의 애스펙트 구현 시에 필요한 정보들을 충분히 표현하지 못하고 있다. 이에 따라 개발 시 구현에 필요한 별도의 정보 명세가 요구된다. [13]에서 제시한 애스펙트 분석 설계 기법은 교차점 결합규칙에 대한 구체적인 명세방법을 제공하지 않고 서술식으로 표현하며 구현에 필요한 구체적인 정보를 제공하지 못하므로 개발 시 교차점 개발을 위한 추가적인 노력이 요구된다. 연구 [5]의 개발 및 유지보수 지원 기법은 기존 연구에서 부족하였던 애스펙트 명세와 작성지침을 제공하고 있으나 다음과 같은 제약사항이 존재한다. 첫째로 애스펙트 당 하나의 충고만을 갖는 형태로 애스펙트의 구성을 제한하고 있다. 애스펙트가 목표 모듈의 유형에 따라 세부동작이 구별되는 충고를 하나 이상 가질 경우 명세서 상에서 표현할 수 있는 방법을 제공하고 있지 않다. 둘째로 교차점 선언 시 결합점들 사이에 존재할 수 있는 공통적인 특징을 반영하지 않고 교차점과 결합점의 일대일 대응만을 표현하고 있다. 본 논문에서 제안한 기법은 애스펙트가 복수의 충고를 가질 수 있는 명세방법을 제공하였으며, 애스펙트와 목표의 결합관계를 결합관계 매트릭스를 적용하여 구체적으로 명세할 수 있었다. 더불어 애스펙트 구현 시 필요한 교차점의 적용규칙을 특정 명칭에 해당하는 하나 이상의 패키지, 클래스, 또는 메소드 단위로 작성 가능하여 구현 시 요구되는 수준의 교차점 정보를 제공할 수 있었다.

6. 결 론

로깅이나 보안 등과 같은 횡단관심사를 효과적으로 모듈화하기 위하여 등장한 관점지향 프로그래밍은 목표 모듈과 횡단관심사를 구현한 애스펙트를 분리하여 개발하고 각 모듈의 책임에만 집중할 수 있다. 그 결과 모듈의 유지보수성과 확장성, 그리고 재사용성 향상을 기대할 수 있다. 이러한 장점으

〈표 13〉 명세 기법의 유용성 비교 및 평가

평가 항목 / 명세 기법	정보 표현성			활용성		
	애스펙트- 목표 관계	애스펙트	교차점 결합규칙	개발용이성	수정용이성	확장용이성
UML 활용 기법	낮음	낮음	없음	낮음	보통	보통
애스펙트 분석 설계 기법	보통	보통	낮음	보통	보통	보통
개발 및 유지보수 지원 기법	높음	보통	보통	높음	보통	높음
제안 기법	높음	높음	높음	높음	보통	높음

로 관점지향 프로그래밍 적용에 대한 다양한 연구가 진행되고 있으나 애스펙트 정보를 명세할 수 있는 명세서와 명세지침은 부족한 상황이다. 특히 애스펙트와 목표의 결합에 대한 정보를 제공하는 명세서와 명세지침은 부족한 상황이다.

본 논문에서는 애스펙트 정보 및 애스펙트와 목표 모듈의 결합정보에 초점을 둔 애스펙트 명세 기법을 제안하였다. 제안한 기법은 애스펙트 명세, 우선순위 결정, 결합정보 명세, 그리고 교차점 명세 단계를 수행하며 애스펙트 정보를 명세하였다. 애스펙트의 정보를 표현하는 애스펙트 명세서와 애스펙트의 적용 우선순위 명세서, 목표 모듈과 애스펙트 사이의 결합관계를 살펴볼 수 있는 결합정보 매트릭스, 그리고 목표 모듈의 결합점 중 애스펙트가 결합하는 결합점들을 표현한 교차점을 명세하기 위해 교차점 명세서를 제시하였다. 각 산출물은 애스펙트 개발에 필요한 정보를 제공하였고 애스펙트와 목표 모듈 사이의 세부적인 결합정보를 명세할 수 있었다. 산출물들을 사용하여 관점지향 프로그래밍 프레임워크의 애스펙트 정보 설정 파일을 자동으로 생성할 수 있었다. 제안한 명세 기법은 상세한 애스펙트 정보 명세 및 애스펙트와 목표의 결합정보 명세를 통해 애스펙트 구현에 필요한 구체적인 정보를 제공하므로 애스펙트 개발 용이성, 이해도, 그리고 재사용성의 향상에 도움을 줄 것으로 기대한다.

향후에는 명세 기법을 확장하여 분석 단계에서부터 시험 단계에 이르기까지 전체 개발 프로세스 상에 적용하는 연구가 필요하며, RUP(Rational Unified Process)나 애자일(Agile) 방법론과 같은 반복적 개발 프로세스의 특징을 반영할 수 있는 애스펙트 명세 기법 및 관점지향 개발방법론에 대한 연구가 필요하다.

참 고 문 헌

[1] Tarr, P., Ossher, H., Harrison, W., Stanley M. and Sutton, J., “N degrees of separation multi-dimensional separation of concerns,” International Conference on Software Engineering (ICSE), IEEE Computer Society Press, pp.107-119, 1999.

[2] Ivar Jacobson, “Use Cases and Aspects-Working Seamlessly Together,” Journal of Object Technology, Vol.2, No.4, 2003.

[3] Isabel Sofia Brito and Ana Moreira, “Towards an Integrated Approach for Aspectual Requirements,” Requirements Engineering, 14th IEEE International Conference, pp.341-342, 2006.

[4] Andy Kellens and Kim Mens, “A Survey of Aspect Mining Tools and Techniques,” Aspect Lab. June, 2005.

[5] 박옥자, 유철중, 장옥배, “프로그램 개발 및 유지보수를 지원하는 횡단관심사 명세 기법”, 한국정보과학회 논문지, Vol. 34 No.09, pp.773-784, 2007.

[6] Khan, S.S and Jaffar-ur-Rehman, M, “A Survey on Early Separation of Concerns,” Proceedings of the 12th APSEC ’05, December, 2005

[7] B. Tekinerdogan and M. Aksit, “Deriving Design Aspects from Canonical Models,” in Object-Oriented Technology, LNCS 1543, ECOOP’98, Springer Verlag, pp.410-413, July, 1998.

[8] Baniassad, E and Clarke. S. “Theme: An Approach for Aspect-Oriented Analysis and Design,” Proceedings on ICSE 2004, pp.158-167, May, 2004.

[9] L. Bass, M. Klein, and L. Northrop, “Identifying Aspects Using Architectural Reasoning,” Workshop of the 3rd International Conference AOSD, 2004.

[10] Andy Kellens, Kim Mens and Paolo Tonella, “A Survey of Automated Code-Level Aspect Mining Techniques,” Transactions on Aspect-Oriented Software Development IV, LNCS 4640, Springer Verlag, pp.143-162, 2007.

[11] J. Bakker, B. Tekinerdogan, and M. Aksit, “Characterization of Early Aspects Approaches,” Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design, held in conjunction with AOSD Conference, 2005.

[12] João Araújo, Isabel Brito, and Awais Rashid, “Aspect-Oriented Requirements with UML,” Proceedings of Workshop on Aspect-oriented Modeling with UML, UML 2002, October, 2002.

[13] Yanhong Guo, Guifa Teng, Yueli Li, Fang Wang and Jianbin Ma, “Improvement of Object-Oriented System Analysis and Design with Aspects,” Proceedings of the 31st Annual International Computer Software and Applications Conference (COMSAC. 2007), Vol.2, pp.379-384, July, 2007.

[14] Java Regular Expression, http://java.sun.com/javase/6/docs/api/java.util.regex.Pattern

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

[16] Spring framework, http://www.springframework.org/

[17] 김현도, “전자요금징수시스템(ETCS) 및 특허 동향”, 특허동향 보고서, 한국특허정보원, http://kor.forx.org/

최 윤 석

e-mail : cooling@dongduk.ac.kr

1997년 숭실대학교 소프트웨어공학과 (공학사)

1999년 숭실대학교 대학원 컴퓨터학과 (공학석사)

1999년～현 재 숭실대학교 대학원 컴퓨터학과 (박사수료)

2005년～현재 동덕여자대학교 정보학부 전임강사

관심분야 : 소프트웨어 개발방법론, 소프트웨어 아키텍처, 관점지향 프로그래밍(AOP)

정 기 원

e-mail : chong@ssu.ac.kr

1967년 서울대학교 전기공학과(공학사)

1981년 미국 알라바마주립대(헌츠빌) 전산학과(전산학석사)

1983년 미국 텍사스주립대(알링턴) 전산학과(전산학박사)

1971년~1975년 한국과학기술연구소 연구원

1975년~1990년 국방과학연구소 책임연구원

1990년~현 재 숭실대학교 컴퓨터학부 교수

2002년~2003년 한국전자거래학회 회장

2007년~현 재 소프트웨어사업선진화포럼 회장

관심분야 : 소프트웨어공학, 소프트웨어프로세스, 정보시스템감리, 전자거래(CALS/EC), 유비쿼터스 컴퓨팅