결합정보 매트릭스를 활용한 애스펙트 결합식 추출 기법

최윤석*, 정기원**
*동덕여자대학교 정보학부
**숭실대학교 컴퓨터학부
e-mail: cooling@dongduk.ac.kr, chong@ssu.ac.kr

An Extracting Technique of Join Expressions of Aspects Using The Join Information Matrix

Yunseok Choi*, Kiwon Chong**
*College of Computer and Information Science, Dongduk Women's University
**School of Computing, Soongsil University

관점지향 프로그래밍은 횡단관심사를 애스펙트로 모듈화 하여 시스템의 개발 용이성, 재사용성 그리고 확장성을 향상시킨다. 이에 관점지향 프로그래밍 적용을 위한 다양한 연구가 진행되고 있으나 애스펙트를 효율적으로 개발하는 기법 관련 연구가 보다 필요한 상황이다. 본 논문에서는 애스펙트의 교차점을 구성하는 핵심요소인 결합식 추출을 위한 기법을 제안한다. 제안한 기법은 결합정보 매트릭스 작성, 결합식 작성, 그리고 결합식 정제 및 확인으로 구성한 워크플로우를 수행하여 결합정보 매트릭스와 패키지 트리를 작성하고, 결합점명 패턴의 공통성을 분석하여 애스펙트 교차점의 결합식을 추출한다. 추출한 결합식은 결합점의 패키지명, 클래스명, 그리고 메소드명 패턴의 공통성을 반영하며, 기법의 산출물은 애스펙트 결합에 대한 정확한 정보를 제공한다.

1. 서론

횡단관심사(cross-cutting concern)를 애스펙트(aspect) 단위로 모듈화 하여 개발하는 관점지향 프로그래밍(Aspect-Oriented Programming)[1]은 기존 개발방법론의 모듈화 방법을 적용하여 횡단관심사를 개발할 때 발생하는 코드의 혼재(scattering) 및 엮임(tangling) 문제 해결을 지원하며, 관심사 분리 원칙의 충실한 적용을 통해 목표 및 횡단관심사 모듈의 개발 용이성 및 재사용성, 그리고 시스템 확장성을 향상시킨다. 관점지향 프로그래밍 기반 시스템은 목표와 애스펙트의 관계가 목표 모듈의 코드에 명시적으로 표현되지 않으며, 애스펙트가 소유한 교차점(pointcut)의 결합식(join expression)에 의해 표현된다. 따라서 시스템의 수행 흐름 및 목표와 애스펙트 결합 관계를 파악하기 위해서는 교차점의 결합식이 명확하여야 한다. 그러나 이러한 정보를 포함하는 애스펙트 명세를 위한 구체적인 지침 및 산출물에 대한 연구는 미미한 상황이다[2].

본 논문에서는 애스펙트 정보 중 교차점의 핵심 구성요소인 결합식을 추출하기 위한 기법을 제안한다. 애스펙트와 목표의 관계를 표현하는 매트릭스와 애스펙트 별 패키지 트리를 작성하여 결합식을 추출하고 정제 및 확인 단계를 통해 교차점의 결합식을 구성한다. 추출한 결합식은 목표가 갖는 결합점(join point)들의 패키지명, 클래스명, 그리고 메소드명의 패턴에 나타나는 공통성을 반영한다. 제안한 기법은 교차점의 결합식 개발을 위한 체계적인 워크플로우를 포함하며, 결과로 산출한 결합정보 매트릭스와 패키지 트리는 애스펙트의 추적에 필요한 정보를 제공한다.

논문의 구성은 다음과 같다. 2장에서는 결합점의 구성과 기존의 교차점 개발 관련 연구를 살펴보며, 3장에서는 제안 기법을 기술한다. 4장에서는 적용사례를 통해 기법의 세부 적용방법을 살펴보며, 5장에서는 기법의 유용성을 토의한다. 6장에서 결론을 맺는다.

2. 관련 연구

2.1. 결합점의 구성

결합점은 시스템의 기능을 수행하는 구별이 가능한 실행지점을 지칭한다[3]. 애스펙트는 교차점의 결합식에 나타난 규칙에 따라 목표의 결합점에 적용된다. 결합점의 유형은 다음과 같다.

- **메소드 결합점 (실행, 호출)**
- **생성자 결합점 (실행, 호출)**
- **필드 참조 결합점**
- **예외 처리 결합점**
- **초기화 결합점 (클래스, 객체)**
- **애스펙트의 충고 실행 결합점**

결합점 유형은 관점지향 프로그래밍 언어 또는 지원 프레임워크의 종류에 따라 달라진다. 자바를 확장한 관점지향 프로그래밍 언어인 AspectJ의 경우 위에 나타난 모든 결합점 유형을 지원하며, 관점지향 프로그래밍 지원 프레임워크인 스프링(Spring)[4]은 인터페이스를 구현한 public 메소드만이 결합점이 된다. 비기능적 요구사항에 대한 횡단관심사를 모듈화 한 애스펙트의 경우 일반적으로 public 접근지정자를 갖는 메소드 결합점들과 결합하는 것으로 충분한 기능 지원이 가능하다. 본 논문에서는 public 접근지정자를 갖는 메소드를 결합점 대상으로 제한하고 이에 대한 교차점의 결합식 추출 기법을 제시한다.

2.2. 교차점 개발 관련 연구

관점지향 프로그래밍과 리팩토링(refactoring)을 통해 애스펙트를 추출하는 방법에 대한 연구 [5]에서는 결합점을 클러스터링하고 추론 모듈을 통해 결합식을 추출하는 방법과 자동화도구를 제시하고 있다. 목표의 결합점이 갖고 있는 메소드명 패턴의 공통성을 분석하여 교차점의 결합식으로 구성한다. 연구[5]에서 제안한 결합식 생성 방법은 결합식 구성 시 메소드명 패턴의 공통성

은 반영하고 있으나 패키지명과 클래스명 패턴의 공통성은 반영하지 않고 있다. AspectJ에서의 교차점 명세 방법을 제시한 연구[6]은 교차점을 정형적으로 명세하기 위해 사전조건/사후조건을 사용하는 방법을 제시하였다. 교차점 범주를 구분하였고, 이에 대한 결합점 유형을 제시하였다. 연구 [7]에서는 교차점의 결합식을 표현식 형태로 쓰지 않고 join point selector 클래스를 사용하여 적용하는 방법을 제시하고 있다. join point selector는 결합점의 유형을 반영하여 결합식을 메소드 형태로 구현하고 있다. 프로그램 뷰를 사용한 교차점 정의의 향상 연구[8]에서는 프로그램의 변경이 교차점에 영향을 미치지 않도록 교차점을 개발하는 방법을 제안하고 있다. 메소드명 패턴의 공통성을 반영하는 동시에 결합정보를 정확히 유지하기 위해 결합식에 추가적인 속성을 사용하고 있다.

이와 같이 교차점 개발을 위한 다양한 연구가 진행되고 있으나 교차점 개발을 위한 체계적인 절차 및 기법에 대한 연구가 보다 필요한 상황이다. 본 논문에서는 교차점 개발을 위한 핵심정보인 결합식을 결합점의 공통성을 반영하여 개발할 수 있는 기법을 제시하고 이를 수행하기 위한 워크플로우를 제시한다.

3. 결합정보 매트릭스를 활용한 결합식 추출

요구사항 분석 및 설계 단계에서 식별한 횡단관심사는 하나 이상의 애스펙트로 모듈화 할 수 있다. 각 애스펙트는 횡단관심사의 기능인 충고를 메소드 형태로 작성할 수 있다. 목표의 결합점 및 애스펙트 충고의 개발은 본 논문의 연구 범위에 포함하지 않는다. 본 논문에서는 설계 또는 개발을 완료한 목표의 결합점과 애스펙트의 충고 정보를 입력으로 확보한 시점에서 교차점의 결합식을 구성하는 단계를 연구 범위로 한다.

교차점은 목표에 대한 애스펙트의 결합규칙인 결합식을 포함한다. 본 논문에서는 교차점이 가져야 할 결합식을 추출하기 위한 워크플로우 및 기법을 제안한다. 그림 1은 교차점 결합식 추출 워크플로우를 나타낸다.

[그림 1] 결합식 추출 워크플로우

결합정보 매트릭스를 작성하여 목표의 결합점과 애스펙트의 충고 사이의 결합관계를 기술하며 패키지 트리를 작성하여 결합식을 위한 패키지 정보를 기술한다. 결합식 작성 단계에서는 작성한 결합정보 매트릭스와 패키지 트리를 활용하여 결합관계의 공통성을 분석한 후 결합식을 추출한다. 추출한 결합식은 정제 및 확인을 수행하여 추가적인 공통성을 반영하고 목표와 애스펙트 사이의 관계가 정확한지 확인한다.

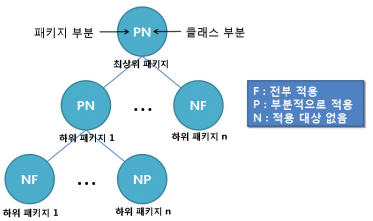
3.1. 결합정보 매트릭스 작성

본 단계에서는 결합식을 추출하기 위해 목표의 결합점과 애스펙트의 충고를 매트릭스로 구성하고 애스펙트의 패키지 적용을 표현하는 패키지 트리를 작성한다. 결합정보 매트릭스는 클래스를 소유하고 있는 각각의 패키지 별로 작성하며, 각 패키지에 적용하는 모든 애스펙트를 동일한 적용시점 별로 구분지어 구성한다. 표 1은 결합정보 매트릭스의 구성을 나타낸다.

<표 1> 결합정보 매트릭스의 구성

결합정보 매트릭스			
패키지명	*패키지 명*		
결합시점	*Before 결합시점 기술*		
목 표 / 애스펙트	*목표 클래스 명*	*목표 클래스 명*	...
애스펙트명.충고메소드명 ...	*목표 결합점 1* ... *목표 결합점 n*	*목표 결합점 1* ... *목표 결합점 n*	...
결합시점	*After 결합시점 기술*		
...	...	...	...

결합시점은 애스펙트의 충고메소드가 목표에 적용되는 시점을 의미하며, 핵심관심사에 대한 횡단관심사의 적용시점과 동일하므로, 애스펙트가 속한 횡단관심사를 검토하여 결정한다. 패키지 명은 애스펙트의 목표가 속한 패키지를 의미한다. 가로축은 패키지에 속한 모든 클래스 명을 기록하며 세로축은 애스펙트들의 목록을 '*애스펙트명.충고명*' 형태로 기록한다. 애스펙트의 우선순위를 반영하여야 하므로 우선순위가 높은 애스펙트 순으로 배치한다. 우선순위 정보는 워크플로우 적용 이전에 작성한 횡단관심사 정보와 애스펙트 정보를 토대로 결정한다. 목표와 애스펙트의 배치 후 각각의 교차지역에는 목표의 결합점들 중 애스펙트가 적용되는 결합점을 기록한다. 결합점은 메소드 시그너처를 기록하며 필요에 따라 접근지정자를 포함한다. 각각의 패키지 별로 결합정보 매트릭스 작성을 완료한 후 패키지 별 애스펙트 결합을 분석하기 위하여 패키지 트리를 작성한다. 그림 2는 패키지 트리의 구성을 나타낸다.

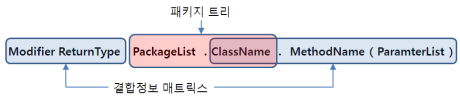


[그림 2] 패키지 트리의 구성

패키지 트리는 시스템을 구성하는 패키지를 트리형태로 표현한 것으로 각 노드에는 애스펙트의 적용에 따라 F/P/N의 값 중 하나를 선택하여 패키지 부분과 클래스 부분으로 기록한다. 패키지 부분에는 애스펙트와 결합하는 하위 노드의 존재유무에 따라 세 가지 값 중 하나를 기록하며, 클래스 부분은 현재 노드가 표현하는 패키지에 속한 클래스들이 애스펙트와 결합하는지에 따라 기록한다. F는 모든 대상이 애스펙트와 결합함(fully applied)을 의미하며, P는 일부가 결합함(partially applied)을 의미한다. N은 결합대상이 없음(not applied)을 의미한다. 값의 기록은 최하위 노드에서부터 기록하여 최종적으로 최상위 패키지를 표현하는 노드의 값을 기록한다.

3.2. 결합식 작성

결합정보 매트릭스와 패키지 트리를 토대로 교차점의 결합식을 작성한다. 결합식은 그림 3과 같이 결합정보 매트릭스와 패키지 트리의 정보를 조합하여 구성한다.



[그림 3] 결합식의 구성

결합정보 매트릭스 분석을 통해 메소드명 패턴의 공통성을 반영한 결합식을 추출한다. 분석은 다음과 같은 순서로 수행한다. 첫째로 애스펙트와 클래스의 교차지역에 기록한 결합점을 식별한다. 클래스 별로 교차지역의 결합점명 패턴의 공통성을 파악한다. 필요에 따라 접근지정자와 반환타입 패턴의 공통성도 반영한다. 결합식은 AspectJ의 문법 중 다음 두 가지 요소를 조합하여 표현한다.

- *** : 모든 값을 표현**
- **.. : 0 개 이상의 값이 포함**

예를 들어 다음과 같은 조건을 가질 경우 위의 두 가지 요소를 적용한 결합식은 다음과 같다.

- **클래스명: Tollgate**
- **접근지정자 및 반환타입: public, 모든 타입 허용**
- **메소드 명: get 으로 시작하는 모든 메소드**
- **매개변수: 0개 이상을 포함**

public * (PackagePattern).Tollgate.get*(..)

다음으로 패키지 트리의 노드 분석을 수행하여 패키지명과 클래스명의 패턴을 분석하고 분석 결과를 결합식의 PackagePattern 부분에 반영한다. 분석지침은 다음과 같다.

- **루트 패키지는 항상 포함**
- **노드의 패키지 부분이 F이면 하위패키지를 * 로 표시**
- **노드의 패키지 부분이 P이면 하위패키지명 명시**
- **노드의 패키지 부분이 N이면 해당사항 없음**
- **노드의 클래스 부분이 F이면 클래스명을 * 로 대체**
- **노드의 클래스 부분이 P이면 클래스명 명시**
- **노드의 클래스 부분이 N이면 해당사항 없음**

분석지침의 적용은 적용사례를 통해 보다 자세히 살펴본다.

3.3. 결합식 정제 및 확인

애스펙트 별로 작성한 결합식을 각 애스펙트의 교차점에 반영하기 전에 결합식의 정제 및 확인을 수행한다. 패키지명, 클래스명, 메소드 명, 그리고 매개변수 명의 추가적인 공통성을 반영하여 결합식 정제를 수행한다. 정제 후 각 애스펙트의 결합식을 적용하여 목표의 결합점을 추적한 후 각 패키지 별 결합정보 매트릭스에 나타난 결합정보와 일치하는지 확인한다. 결합식 정제 및 확인을 완료한 후에는 애스펙트 별로 교차점에 작성한 결합식을 반영하여 교차점을 구성한다.

4. 적용사례

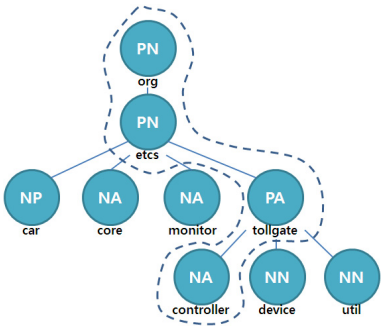
전자요금징수 시스템(이하 ETCS) 시뮬레이터를 관점지향 프로그래밍을 적용하여 개발하였고 교차점 개발 시에 제안한 기법을 적용하여 유용성을 확인하여 보았다. ETCS는 유료도로 또는 고속도로 통행 시 차량의 OBU(OnBoard Unit)와 무선통신을 통해 OBU에 장착한 통행료 충전 IC카드에서 통행료를 자동으로 징수하는 시스템이다[9]. 관점지향 프로그래밍을 적용할 경우 OBU와 OBU 통신기 사이의 정보 교환을 위한 고속암호화/복호화 처리, 각 장치들의 운용상태 이상 유무, 통행료 처리 트랜잭션을 처리, 그리고 시스템의 상태 모니터링 등을 횡단관심사로 식별하여 개발할 수 있다.

ETCS 시뮬레이터의 애스펙트 개발 시 교차점의 결합식 추출을 위해 제안한 기법을 적용하였다. 워크플로우의 첫 번째 단계를 수행하여 결합정보 매트릭스와 패키지 트리를 작성하였다. 결합점은 public 메소드만을 대상으로 제한하여 작성하였다. 표 2는 org.etcs.tollgate.controller 패키지에 대한 결합정보 매트릭스를 나타낸다.

<표 2> ETCS 결합정보 매트릭스의 예

결합정보 매트릭스		
패키지명	org.etcs.tollgate.controller	
결합시점	Before	
목 표 / 애스펙트	ExitController	EntryController
logging.log	void process() void updateObu()	void process() void updateObu()
결합시점	After	
decoding.decode	N/A	N/A

패키지에 속한 EntryController 클래스와 ExitController 클래스를 가로축에 배열하고 세로축에 logging과 decoding 애스펙트를 충고명과 함께 배열하였다. 각각의 교차지역에는 애스펙트가 결합하는 목표클래스의 결합점을 기록하였다. 매트릭스 구성 완료 후 패키지 트리를 작성하였다. controller 패키지의 경우 하위 패키지가 없고 패키지에 속한 모든 클래스가 애스펙트와 결합하므로 노드에 NA값을 기록하였다. 그림 4는 logging 애스펙트의 패키지 트리를 보여준다.



[그림 4] logging 애스펙트의 패키지 트리

다음 단계로 결합정보 매트릭스와 애스펙트 패키지 트리를 분석하여 결합식을 작성하였다. org.etcs.tollgate.controller 패키지의 경우 패키지 트리 분석 결과 하위 패키지가 없고 패키지 내의 모든 클래스가 logging 애스펙트와 결합한다. controller 패키지 노드의 상위노드인 tollgate 패키지 노드의 패키지 부분 값이 P이므로 controller 패키지의 이름은 결합식에 명시적으로 기록하여야 한다. 패키지 트리 분석을 통해 추출한 결합식의 Package Pattern은 다음과 같다.

- **org.etcs.tollgate.controller.***

목표 클래스 정보와 결합정보 매트릭스를 분석한 결과 controller 패키지에 속한 EntryController 와 ExitController의 모든 메소드가 logging 애스펙트가 결합하므로 초기 결합식을 다음과 같이 작성하였다.

- **public * (*PackagePattern*).*(..)**

패키지 트리 분석 결과인 PackagePattern과 매트릭스 분석을 통한 초기 결합식을 조합하여 작성한 결합식은 다음과 같다.

- **public * org.etcs.tollgate.controller.*.*(..)**

마지막 단계로 결합식 정제 및 확인을 수행하였다. 다른 패키지에 대한 애스펙트의 결합식과 공통적으로 나타나는 패턴을 분석하여 결합식을 정제하였다. 분석 결과 logging 애스펙트는 상위 패키지인 org.etcs 패키지에 속한 모든 대상에 대하여 공통적으로 결합하므로 결합식을 다음과 같이 정제하였다.

- **public * *..*.tollgate.controller.*.*(..)**

정제한 결합식을 logging 애스펙트의 교차점에 적용하였다. 그

림 5는 추출한 교차점의 결합식을 스프링의 AOP 설정 XML 파일에 반영한 상태를 나타낸다.

```
<bean id="loggingAspect"
    class="org.etcs.aspect.LoggingAspect" />
<bean id="entryController"
    class="org.etcs.tollgate.controller.EntryController" />
<bean id="exitController"
    class="org.etcs.tollgate.controller.ExitController" />

<aop:config>
    <aop:aspect id="logging" ref="loggingAspect" >
        <aop:pointcut id="controllerLoggingPC"
        expression="execution(public * *..*.tollgate.controller.*.*(..))" />
        <aop:before pointcut-ref="controllerLoggingPC" method= "log" />
```

[그림 5] 결합식을 적용한 AOP 설정 정보

설정 XML 파일 작성 후 이를 사용하여 시스템을 수행한 결과 EntryController와 ExitController의 메소드 수행 시 결합식에 따라 logging 애스펙트가 정상적으로 결합함을 확인할 수 있었다.

5. 토의

본 논문에서는 애스펙트 교차점의 결합식을 추출을 위한 워크플로우와 결합식 추출 기법을 제시하였다. 기법을 적용하여 추출한 결합식은 결합점명 패턴의 공통성이 반영되었으며, 결합식 추출을 위해 작성한 결합정보 매트릭스와 애스펙트의 패키지 트리는 애스펙트와 목표 사이의 정확한 관계를 추적할 수 있는 정보를 제공하므로, 애스펙트의 개선 및 확장 시에 애스펙트 추적에 활용할 수 있다. 표 3은 결합식 추출에 대한 기법을 제공하는 기존 연구[5]와 본 논문에서 제안한 기법의 비교 결과를 보여준다. 비교항목을 충실히 지원할 경우 ○, 부분적으로 지원할 경우 △, 지원이 없을 경우 × 로 표시하였다.

<표 3> 기존 기법과의 비교

기법 / 비교항목	Automated Inference 기법	제안기법
워크플로우 유무	△	○
결합점 공통성 반영	△	○
산출물 활용	△	○
자동화 도구 지원	○	×

비교 결과 제안한 기법은 결합식 추출을 위한 구체적인 워크플로우를 제시하였고 결합점의 공통성을 반영한 결합식 추출을 지원하고 있으나 자동화도구의 지원은 부족하다. 따라서 제안기법을 적용한 자동화도구의 개발이 요구되고 있다.

6. 결론

관점지향 프로그래밍은 기존 방법론의 모듈화 방법을 적용할 경우 코드의 혼재 및 엮임 문제가 발생할 수 있는 횡단관심사를 애스펙트라는 모듈 단위로 모듈화 하여 시스템의 개발 용이성, 재사용성, 그리고 확장성을 향상시킨다. 이러한 장점으로 인해 관점지향 프로그래밍을 시스템 개발에 적용하기 위한 다양한 연구가 진행되고 있으나, 애스펙트를 효율적으로 설계하고 구현하는 기법에 대한 연구는 미미한 상황이다. 특히 결합정보를 표현하는 교차점을 개발하기 위한 체계적인 기법 연구는 부족한 상황이다. 이에 본 논문에서는 교차점 개발 시 교차점의 핵심 구성요소인 결합식 추출을 위한 기법을 제안하였다. 제안한 기법은 워크플로우에 따라 애스펙트와 목표의 관계를 매트릭스로 작성하며 애스펙트와 목표를 포함하는 패키지 사이의 관계를 표현하는 패키지 트리를 작성한다. 작성한 산출물들을 분석하여 결합점명 패턴의 공통성을 파악한 후 교차점의 결합식을 작성한다. 교차점의 결합식을 생성하기 위한 워크플로우와 기법을 제공하였으며, 기법을 적용하여 생성한 결합식은 결합점 사이의 공통성을 반영하였다. 기법의 산출물은 애스펙트의 개선 및 확장 시 애스펙트와 목표 사이의 관계를 파악할 수 있는 명확한 정보를 제공할 수 있다.

향후에는 제안한 기법을 애스펙트 기반 개발방법론에 적용하기 위한 연구가 필요하며, 기법 적용의 효율성을 높이기 위해 산출물을 통해 결합식을 자동으로 생성하는 자동화 도구에 대한 연구가 필요하다.

참 고 문 헌

[1] G. Kiczales, J. Lamping, A. Menhdhekar, C. Maeda, C. Lopes, J-M. Loingtier, and J. Irwin. Aspect-oriented programming. In Mehmet Aks,it and Satoshi Matsuoka, editors, Proceedings European Conference on Object-Oriented Programming, volume 1241, pages 220 - .242. Springer-Verlag, Berlin, Heidelberg, and New York, 1997.

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

[3] http://www.eclipse.org/aspectj

[4] http://www.springframework.org

[5] Prasanth Anbalagan and Tao Xie, Automated Inference of Pointcuts in Aspect-Oriented Refectoring, Proceedings of The 29th International Conference on Software Engineering(ICSE'07), May 2007, pp.127-136

[6] Yi Wang and Jianjun Zhao, Specifying Pointcuts in AspectJ, Proceedings of The 31st Computer Software and Applications Conference(COMPSAC 2007), Vol. 2. July 2007, pp.5-10

[7] Cristiano Breuel and Francisco Reverbel, Join Point Selectors, Proceedings of The 5th Workshop on Software Engineering Properties of Languages and Aspect Technologies(SPLAT'07), March, 2007.

[8] Zifu Yang and Tian Zhao, Improve Pointcut Definitions with Program Views, Proceedings of The 5th Workshop on Software Engineering Properties of Languages and Aspect Technologies(SPLAT'07), March, 2007.

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