ISSN 1226-9182

정보처리학회지

Korea Information Processing Society Review

www.kips.or.kr

KIPS

2016년 11월 | 제23권 제6호 |

오픈소스 기반 프레임워크 기술

- 오픈소스 기반의 자동차 SW 개발 프로세스 관리 방안
- 오픈소스를 활용한 개방형 클라우드 플랫폼의 적용 사례
- 벤처중소기업 유지보수 솔루션을 위한 통합 서비스 도구의 오픈소스화
- 컨테이너 기반의 코딩 실습 및 운영 플랫폼
- 오픈소스 기반 MOOC 플랫폼 설계 원칙 및 구축
- 아세안 대학이러닝 지원 OER 플랫폼 개발 및 운영

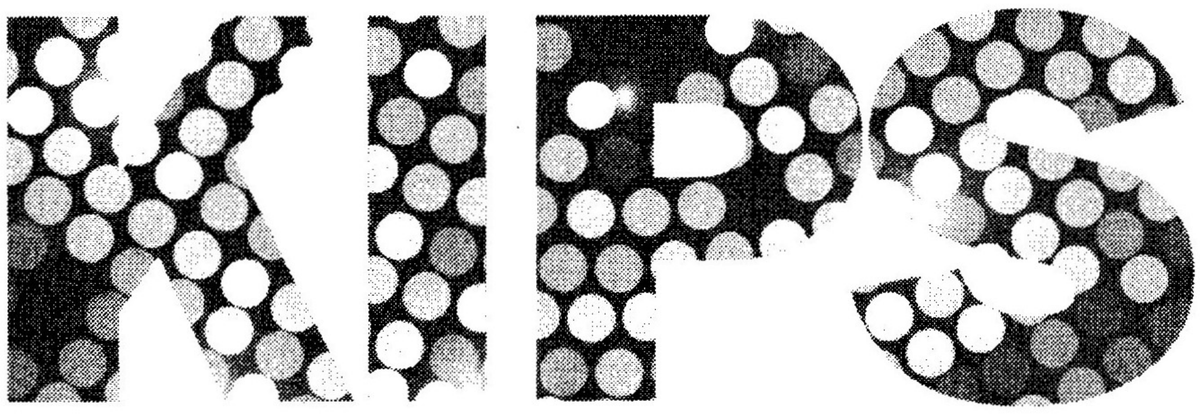
사단법인 한국정보처리학회
KIPS Korea Information Processing Society

ISSN 1226-9182

정보처리학회지

Korea Information Processing Society Review

www.kips.or.kr



2016년 11월 | 제23권 제6호 |

특집 03

벤처중소기업 유지보수 솔루션을 위한 통합 서비스 도구의 오픈소스화

서채연·문소영·손현승·이근상 (홍익대학교), 김영수 (NIPA), 김영철 (홍익대학교)

목 차

과거의 NIPA 부설 SW공학센터에서는 벤처/중소기업의 SW 품질을 위해, 1) SW 프로세스 강화 교육 서비스 제공과 2) SW 가시화를 위한 오픈소스를 이용한 툴-체인 구축 서비스를 지원하였다. 기존 고수준의 방법론, 프로세스 등과는 다르게, 이 방법은 작은 기업 업체들의 대표, 개발자, 테스터, 품질 담당자 스스로 코드 내부의 복잡도, 결합도, 응집도를 개선하는 동기/자극으로 고품질 노력을 부여한다. 우리는 SW 가시화 I, II, III 경험으로 업체들에게 공개소스 기반 툴-체인의 구축과 확장의 많은 서비스 제공이 가능하다. 기존 체인화된 한 도구의 출력이 다른 도구의 입력이 가능하도록 필요한 정보 추출과 프로그래밍을 통해, 많은 서비스 구축 및 공개 서비스화 할 수 있다.

1. 서 론

좋은 SW를 위한 많은 방안이 존재하지만 여전히 벤처/중소기업의 SW 품질관리는 어렵다. 이것은 SW 개발 과정 전반에 대한 관리와 SW 개발 프로세스의 도입이 필요하기 때문이다.

성공적인 SW 개발에는 SW 자체, 즉 소스 코드와 SW 개발 프로세스에 대한 관리가 필요하다. SW 프로세스는 이러한 관리를 위한 전통적인 방법이다. 그러나 SW 공학 프로세스를 통해 품질관리를 수행하기에는 국내 중소기업의 인력 및 비용이 부족하다. 이러한 현실에서 국내 중소기업에 적합한 SW 품질관리의 현실적인 방안이 필요한 시점이다.

SW 가시화(Visualization)[2-3]는 이러한 방안으로 프로세스, 아키텍처, 문서를 시각화한다. 첫째, 시각화는 SW 개발의 가장 어려운 점인 SW 비가시성을 극복함으로써 SW 개발의 전체 과정을 파악하며, 이를 통하여 SW 개발 품질 관리를 실현하고자 한다. 둘째, 문서화는 기업의 개발 노하우 관리 및 내부 인력간의 업무 이해도 향상과 특정 상황에서 외부와의 의사소통이다.

즉, SW 가시화는 소스코드와 개발 프로세스를 관리 목적이고, 시각화와 문서화하여 SW개발 품질관리를 수행한다. 이에 따라 ①개발 프로세스의 시각화, ②소스코드의 시각화, ③소스코드의 문서화, ④개발 프로세스의 문서화로 SW 개발 품질관리를 한다[2].

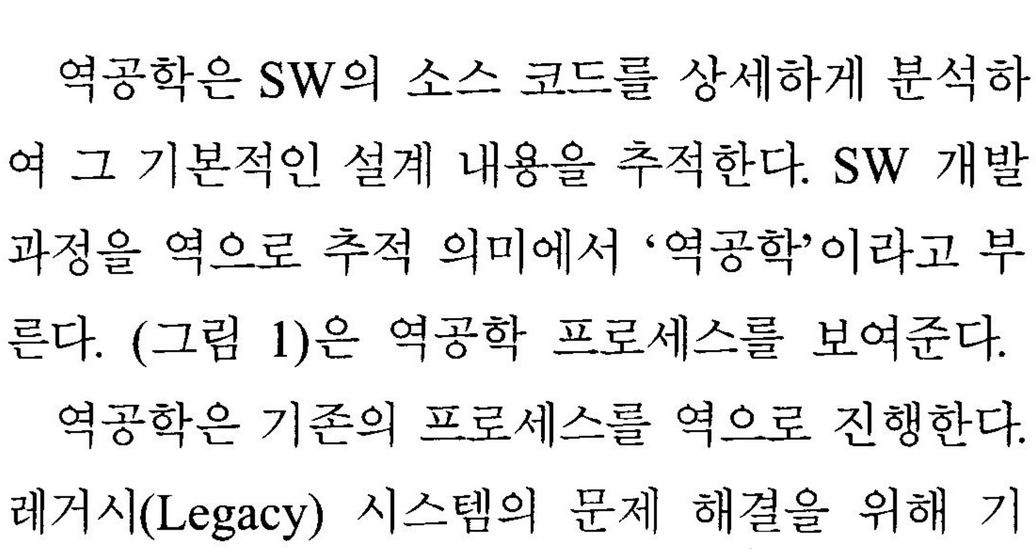
Ball & Eick는 "소프트웨어는 방대한 크기의 프로젝트 내의 비가시성으로 개발자 생산성이 복잡도 의해 낮아진다."라고 말했다. 이로 인해 유지보수도 어렵다[3]. 그러므로 본 논문에서는 벤처/중소기업들이 SW 유지보수를 원활히 수행할 수 있도록 돕는 SW 가시화 서비스들에 대해 소개한다.

2. 관련연구

2.1 역공학

역공학은 SW의 소스 코드를 상세하게 분석하여 그 기본적인 설계 내용을 추적한다. SW 개발 과정을 역으로 추적 의미에서 '역공학'이라고 부른다. (그림 1)은 역공학 프로세스를 보여준다.

역공학은 기존의 프로세스를 역으로 진행한다. 레거시(Legacy) 시스템의 문제 해결을 위해 기본적으로 역공학을 적용한다. 역공학에서 SW 가시화는 Source Navigator, xCodeParser와 같은 분석기를 통해서 소스 코드의 구조를 가시화 한다. 이 가시화된 구조를 통해 쉽게 SW의 아키텍처를 파악가능하고 설계 부재 문제를 해결한다. 또한 품질 지표 적용으로 SW 품질 측정과 추가적인 문제점을 파악이 가능하다. 따라서 고품질 SW를 개발할 수 있다[4].



(그림 1) 역공학 프로세스

2.2 레거시(Legacy) 시스템

과거 SW의 개발은 전문 인력과 시간의 부족으로 코드 중심으로 수행되었다. 그러나 설계 문서의 부재와 개발자 중심의 코드작성은 개발자가 이직하면 코드 개선이 어려운 상태가 된다. 또한 SW 개발 과정 중에서 분석과 설계단계에 취약할 수밖에 없고 코드 내부의 복잡도에 큰 영향을 준다. 이런 이유로 테스팅과 유지보수 단계에 소요되는 비용이 전체 개발 비용의 절반을 차지한다[1].

2.3 리팩토링(Refactoring)

리팩토링은 소스 코드에서 복잡한 부분을 간결하게 재구성한다. 기본적으로 SW를 개선하는 설계를 구성하고, 이해하고 수정하기 쉽게 만든다. 리팩토링은 버그를 없애나거나 새로운 기능을 추가하지는 않는다. 그러므로 SW의 기능 자체는 변경되지 않아야 한다. 즉, 내부 논리나 구조를 바꾸고 개선해 SW 유지보수가 가능하다[5]. 이 방법은 나쁜 냄세(Bad Smell), 개선이 필요한 모듈, 응집도, 결합도를 추출하여 리팩토링해 소스 코드의 품질을 향상 시킨다.

2.4 오픈소스 도구 비교

벤처 중소기업이 고품질의 SW를 개발하기 위해서는 개발 전체 생명주기의 도구들인 프로젝트 계획 및 관리, 데이터베이스, 요구사항 관리, UML 설계, 빌드 도구 등의 확보가 필요하다. 본

장에서는 기 존재하는 여러 오픈소스 도구들을 비교한다.

<표 1>은 데이터베이스와 프로젝트 계획 및 관리에 사용되는 오픈소스 도구를 비교한 테이블이다[6]. MySQL은 무료 오픈소스 플랫폼으로 중소기업이 사용하기 용이한 데이터베이스이다. ProjectLibre는 자바만 설치되어 있다면 어느 PC 환경에서든 실행이 가능하고 쉽게 다양한 기능을 사용하게 구성되어 있어 프로젝트 계획 및 관리로 사용이 용이하다.

<표 2>는 요구사항 관리와 UML 오픈소스 도구를 비교한 테이블이다. 요구사항 관리에 적용 가능한 도구는 JRequisite이다. 이 도구는 순서도를 제공하는 도구로써 자바 환경에서 실행이 가능하기 때문에 다양한 기능을 사용하기 쉽다. AgilelStructureViews는 클래스 다이어그램으로부터 자바를 자동으로 생성해주기 때문에 UML 도구로 활용가능하다.

〈표 1〉 오픈소스 데이터베이스 & 프로젝트 도구 비교

데이터베이스			프로젝트 계획 및 관리		
도구	필요	이유	도구	필요	이유
Oracle		유료	Project Libre	O	자바설치PC
MySQL	O	무료	Libre Plan		웹기반오픈소스
Access		–	Open Project		최적화 불안정
DB2		–			

〈표 2〉 오픈소스 요구 사항 관리 및 UML 도구 비교

요구사항 관리			UML 도구		
도구	필요	이유	도구	필요	이유
JRequisite	O	자바설치 PC	AgileStructureViews	O	자동화
OSRMT		별도의 DB 연동	ArgoUML		비자동화
JFeature		불필요	BOUML		–

<표 3>은 빌드 도구의 비교 테이블이다. Jenkins는 SW 프로젝트 빌드, 반복 작업을 모니터 하기 위한 웹 어플리케이션으로 프로젝트 통합 빌드 시스템이다. 이 도구는 지속통합관리 툴이고, 여러 프로젝트의 소스 및 빌드, 배포, 이슈를 총체적으로 관리가능하다. 그러므로 빌드 도구에는 Jenkins가 적합하다.

〈표 3〉 오픈소스 빌드 도구 비교

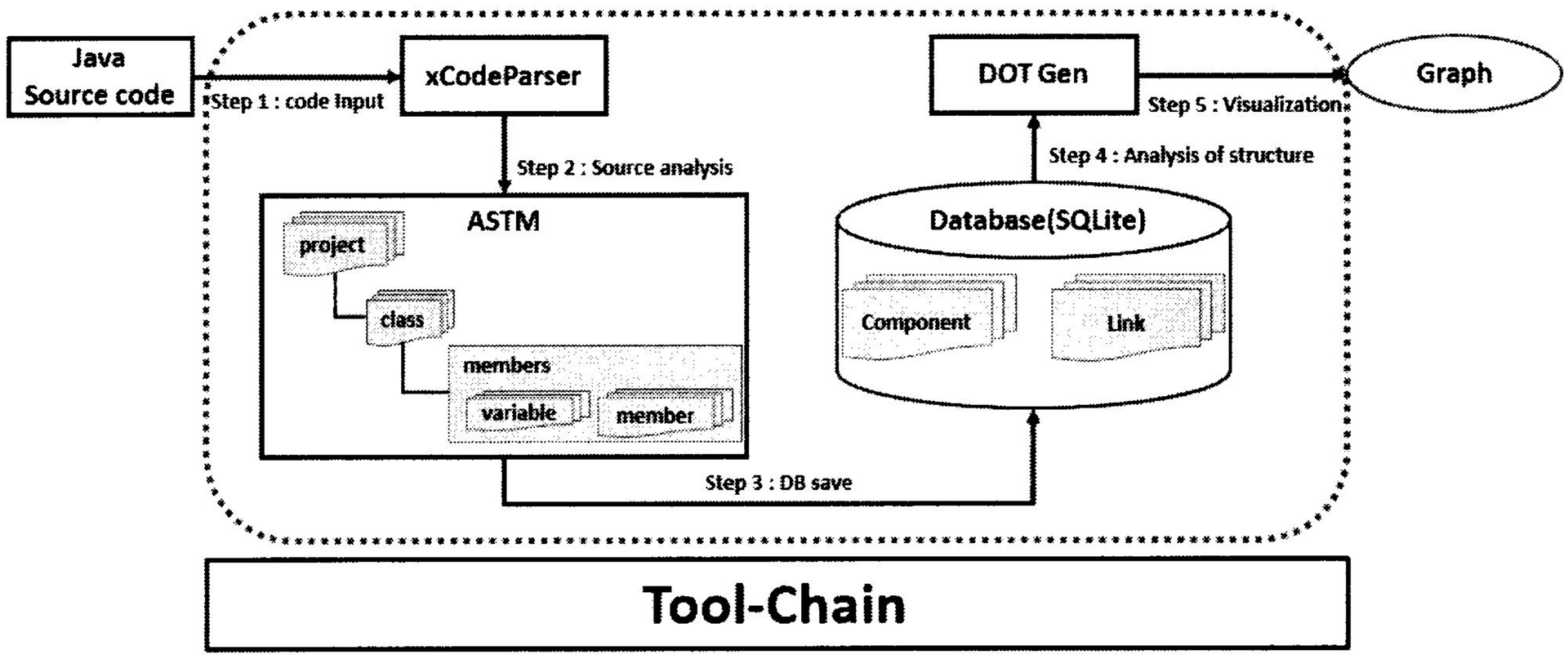
빌드 도구		
도구	필요	이 유
Maven		유료
Gradle		검토대상
CruiseControl.Net		보안설정의 수작업화
Jenkins	O	무료

3. 오픈소스 기반의 서비스 패러다임

3.1 툴-체인 구축 제공

각 단계별 필요한 오픈소스를 사용하면 (그림 2)와 같이 SW 가시화를 위한 툴-체인화 할 수 있다. 이것은 SW 개발 전 단계를 필요한 오픈소스로 통합 가능하다. 또한 SW 가시화를 위한 툴-체인화는 여러 개의 오픈소스들을 하나의 도구처럼 사용하게 한다. 이 SW 가시화는 내부 소스 코드의 품질 개선을 위해 사용한다[8].

(그림 2)는 오픈소스를 이용한 툴-체인 구축 서비스이다. 코드 입력, 소스 분석, 데이터베이스 저장, 구조 분석 및 가시화 총 5단계이다. 이 과정은 소스 코드를 분석하여 그 결과를 ASTM(Abstract Syntax Tree Metamodel) 형태로 바꾼다. 그 후에 ASTM 데이터를 가지고 원하는 구조로 구조화하여 데이터베이스에 저장한다. 그리고 Dot를 통해 데이터베이스에 저장된



(그림 2) 오픈소스를 이용한 도구 체인 구축 서비스

정보로 가시화 그래프를 추출한다. 각 단계를 자세한 내용은 다음과 같다.

- Step 1(Code Input) : 코드 입력 단계는 대상이 되는 소스 코드를 xCodeParser에 입력한다. 이 논문에서는 객체 지향 언어인 Java 코드를 입력한다.
- Step 2(Source Analysis) : 소스 분석 단계는 입력된 코드를 xCodeParser를 사용해 분석한다. 이때, java 파일의 데이터를 가지고 ASTM 파일들이 생성된다. 각 파일에는 프로젝트, 클래스, 멤버, 변수, 메서드 등에 대한 정보를 XML 형태와 유사하게 저장한다.
- Step 3(DB save) : 데이터베이스 저장 단계는 Step 2에서 분석된 정보인 ASTM 데이터를 원하는 구조로 구조화해서 데이터베이스에 저장한다. 이 논문에서는 모든 변수와 메서드의 정보를 컴포넌트로 구조화하고 메서드 간의 호출 관계 정보를 링크로 구조화하여 파일 데이터베이스인 SQLite에 저장한다.
- Step 4(Analysis of Structure) : 구조 분석 단계는 미리 정의된 모듈에 따라서 Step 3에서 데이터베이스에 저장된 정보를 재해석한다.
- Step 5(Visualization) : 가시화 단계는 Step 4에서 재해석한 정보를 DOT Gen(Graphviz)를 이용해 가시화한다.

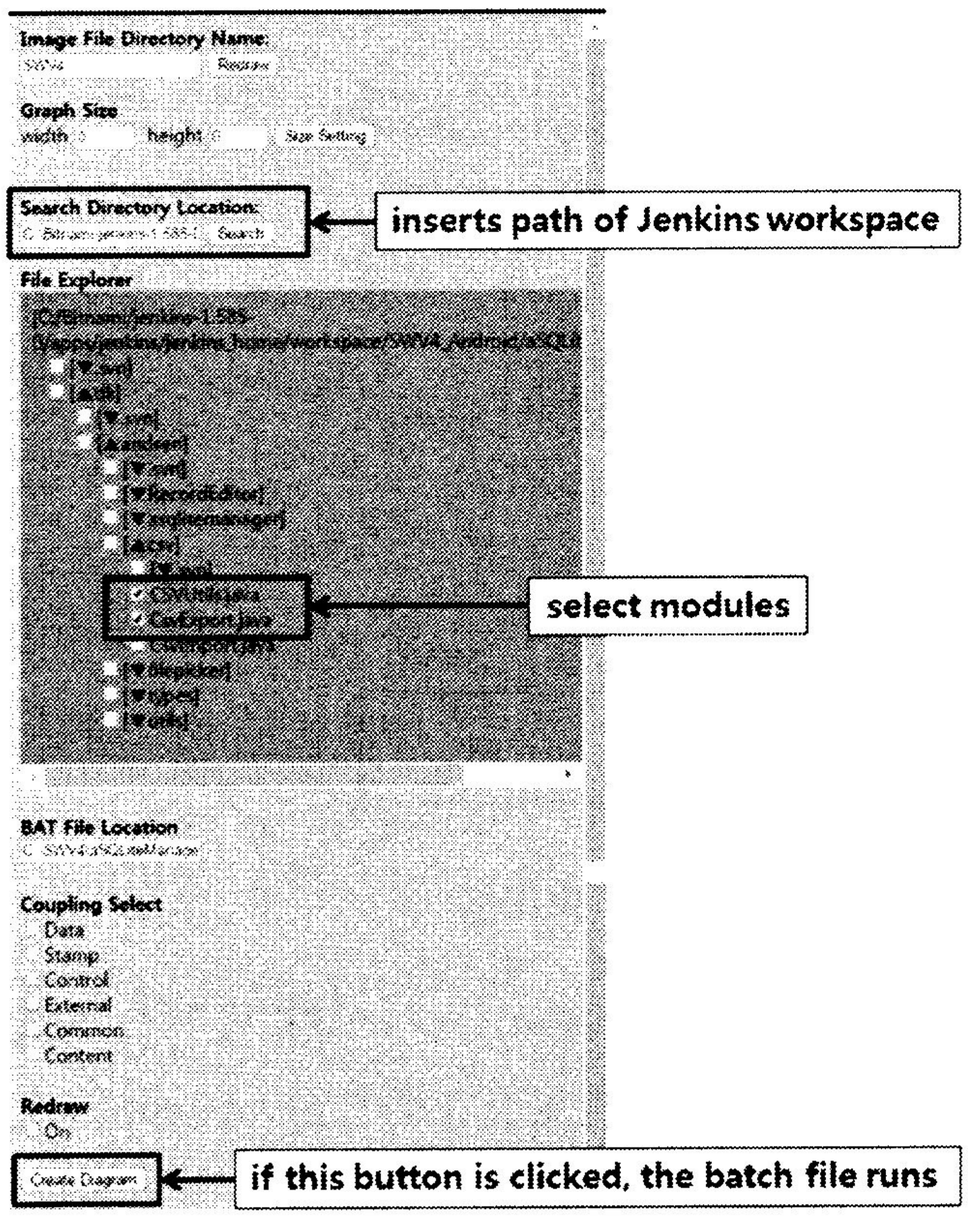
3.2 서비스화의 패러다임

벤처/중소기업의 SW 품질을 위해 오픈소스 도구들을 이용해, 많은 서비스를 제공하고자 한다. 또한 자동 SW 프로세스 구축, SW 가시화 구축, 필요한 품질 서비스를 제공함으로써 고품질을 쉽게 이루고자 한다.

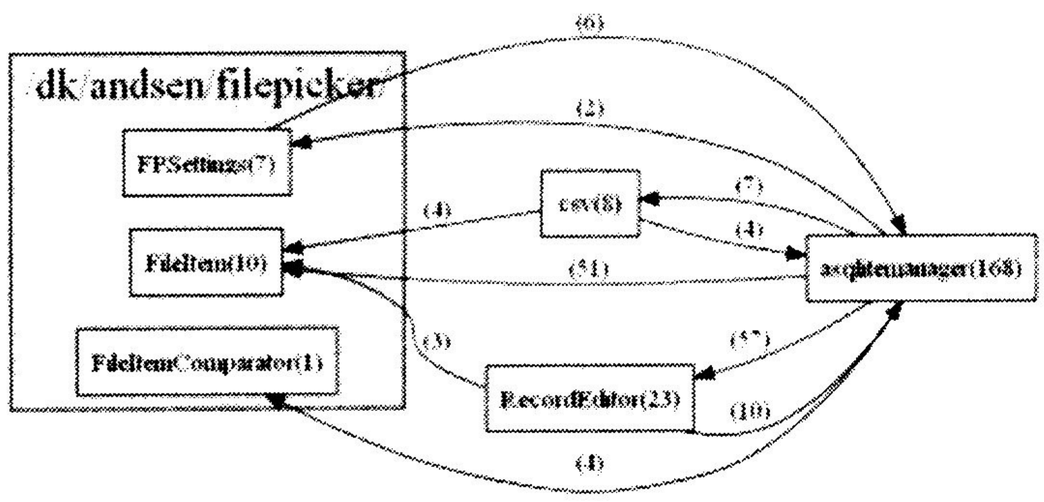
3.2.1 아키텍처 추출 서비스

역공학 기반으로 소스 코드로부터 아키텍처를 추출하는 서비스를 제공하면, 전체 구조를 효율적으로 변경 가능하다. 또한 단위(모듈, 객체, 컴포넌트, 패케이지) 별 인터페이스 선택을 통해, 단위별 내부 코드 구조를 파악 할 수 있는 서비스를 제공할 수 있다.

(그림 3)과 같이 단위별 추출하고자 하는 모듈을 선택으로 보고 싶은 영역을 가시화한다. (그림 3)은 웹 페이지 UI로 메뉴이다. 이 UI 메뉴에서 특정 모듈을 선택하면 (그림 4)와 같은 모듈의 호출 관계를 가시화한다. (그림 4)는 모듈 간



(그림 3) 웹 페이지 UI



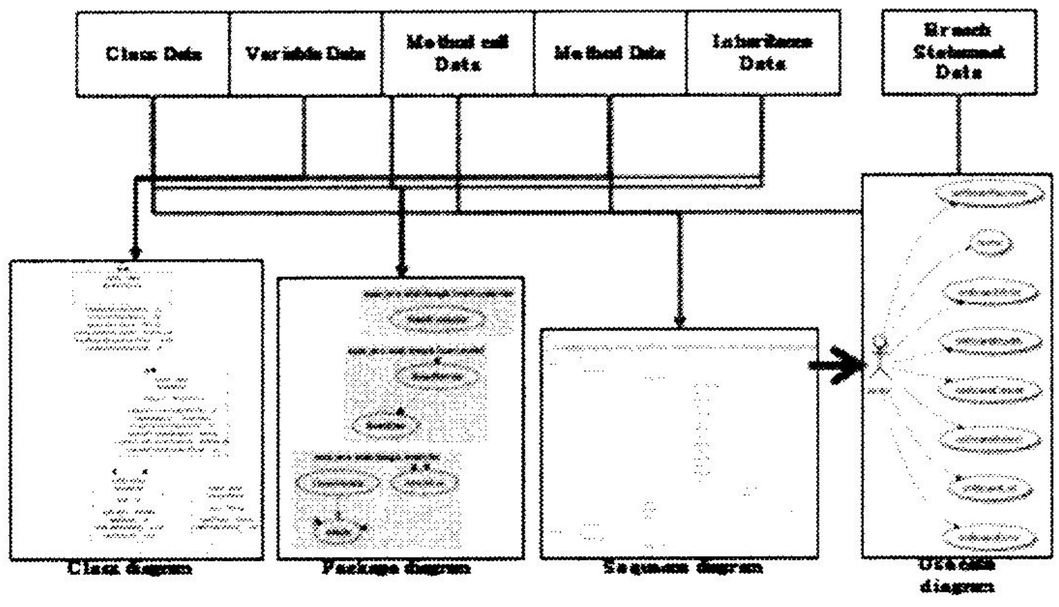
(그림 4) 모듈 호출 구조

의 호출 구조를 나타낸다[13].

3.2.2 UML Diagram 추출

소스 코드로부터 설계문서, 즉 구조적 설계문서인 클래스 다이어그램과 패키지 다이어그램을 추출한다. (그림 5)는 설계도면 추출 프로세스이다. 코드 분석단계에서 파서(Parser)를 통해 코드 분석 후, 구성 요소를 분리한다.

데이터베이스 저장 단계에서 코드 구성 요소를 테이블에 분류한다. 구조 분석 단계는 분류된 정보를 바탕으로 클래스 다이어그램, 패키지 다이어그램, 시퀀스 다이어그램에 필요한 정보를 넣는다. 가시화 도구를 통해 다이어그램 모델을 이미지 파일로 가시화한다[1,7].



(그림 5) UML 다이어그램 추출 서비스

3.2.3 결합도 추출 서비스

결합도는 모듈 간의 상호 의존 정도를 나타내는 품질 지표이다. 즉, 모듈 간의 각 구성 요소들이 얼마나 관련이 있는지 연관 정도를 뜻한다. 따라서 모듈 내부 요소들 사이의 결합도를 낮게 설계하는 것이 바람직하다.

〈표 4〉 객체 지향 관점에서 결합도

결합도 (Grade, Weight)	정의
Data Coupling(1,1)	· 메서드 간의 호출 파라미터가 기본 자료형인 경우로 정의
Stamp Coupling(2,1)	· 모듈 간의 호출 파라미터가 배열이나 오브젝트, 구조체인 경우로 정의
Control Coupling(3,1)	· 모듈 간의 호출 파라미터가 if문이나 switch문 같은 분기 조건으로 사용된 경우로 정의
External Coupling(4, 1.25)	· 모듈 간의 호출 파라미터에 외부 데이터를 전달하는 경우나 동일한 외부 데이터를 사용하는 경우로 정의
Common Coupling(5, 1.4)	· 모듈 간의 호출 시 static으로 선언된 메서드를 호출하는 경우로 정의
Content Coupling (6, 1.7)	· 모듈 간의 호출 시 get/set 메서드를 호출하고 해당 메서드가 있는 클래스에 private형의 변수가 있는 경우로 정의

기존의 결합도의 정의는 절차식 언어 관점에 적합하다. 따라서 <표 4>는 객체지향 패러다임의 재사용 성숙도를 높이도록 재 정의하고 정확한 메트릭을 위해서 Grade와 Weight를 정의한다. Grade는 1부터 커플링이 높아질수록 1씩 증가하고, 각 차이의 구분을 위해서 Weight를 점차 증가하도록 설정된다[8]. 이러한 서비스도 가능하다.

3.2.4 응집도 추출 서비스

응집도는 모듈 내부가 얼마나 단단히 뭉쳐져 있는가를 나타내는 품질 지표이다. 즉, 모듈 내의 각 구성 요소들이 공통의 목적을 달성하기 위하여 서로 얼마나 관련이 있는지 연관 정도를 뜻한다. 따라서 모듈 내부 요소들 사이의 응집도를 높게 설계하는 것이 바람직하다.

기존의 응집도의 정의는 절차식 언어 관점에 적합하다. 따라서 <표 5>는 객체지향 패러다임의 재사용 성숙도를 높이도록 재 정의하고 정확한 메트릭을 위해서 Grade와 Weight를 정의한다. Grade는 1부터 응집도가 높아질수록 1씩 증가하고, 각 차이의 구분을 위해서 Weight를 점차 증가하도록 설정한다[8-9].

〈표 5〉 객체 지향 관점에서 응집도

응집도 (Grade, Weight)	정의
Functional Cohesion (7, 1.45)	· 대입 되는 변수가 공통적으로 사용되는 경우 예외 경우가 발생하는 변수가 있다면 해당하지 않음
Sequential Cohesion (6, 1.34)	· 메서드의 반환 값이 다음 메서드의 파라미터로 쓰이는 경우
Communicational Cohesion (5, 1.2)	· 메서드 호출에 공통된 파라미터가 입력되는 경우
Procedural Cohesion (4, 1.1)	· 하나의 클래스에 있는 메서드들을 여러 개 호출하는 경우 이 때, 응집도는 하나의 모듈에 대한 것이기 때문에 자신의 클래스에 있는 메서드들을 호출하는 경우로 제한
Temporal Cohesion (3, 1)	· 메서드 호출이 일어나지 않고 변수의 초기화만 실행된 경우
Logical Cohesion (2, 1)	· switch문이 쓰여 case에 따라 비슷하지만 다른 작업을 수행하는 경우

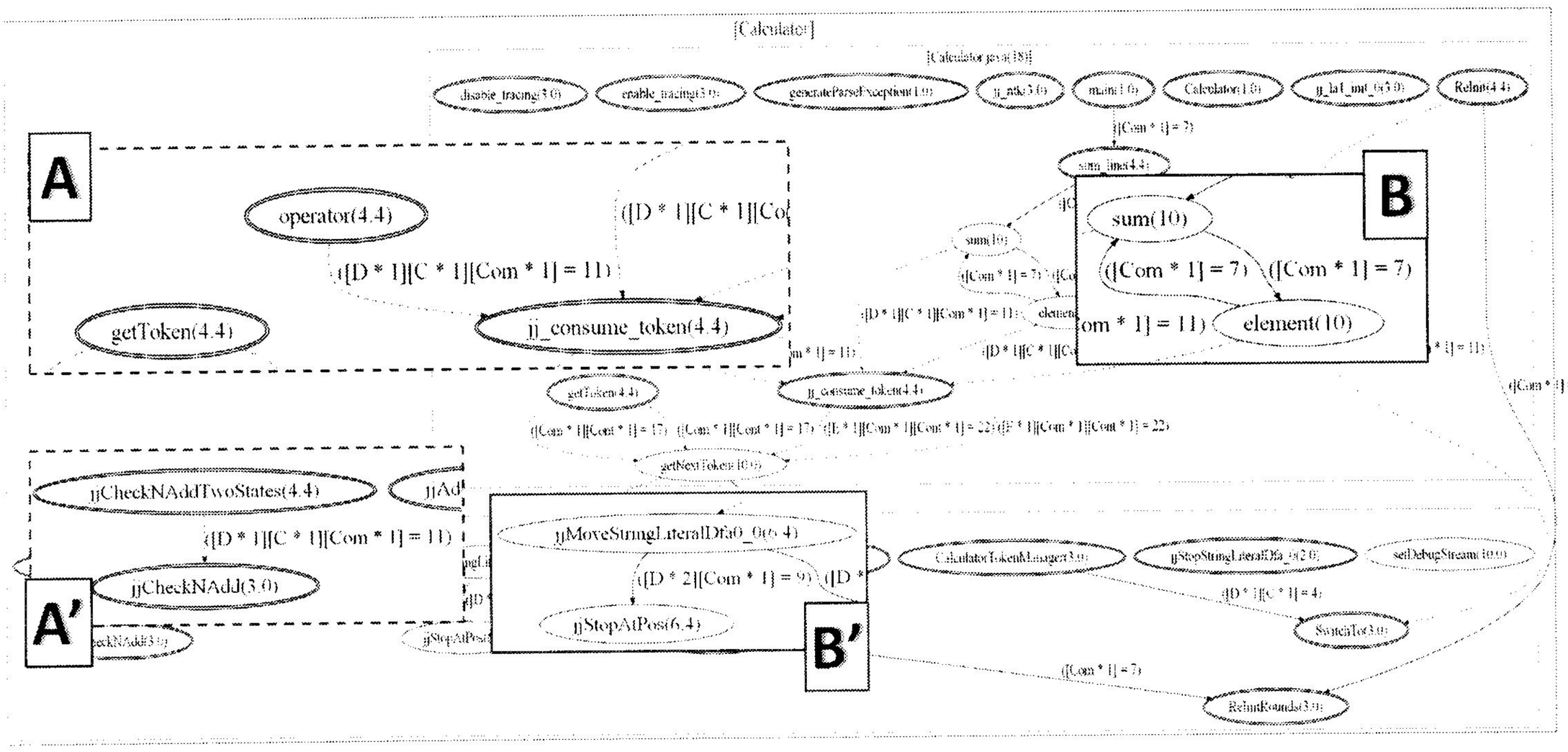
3.2.5 가중치 추출 서비스

결합도와 응집도 추출의 대상이 될 코드는 객체 지향 언어인 결합도와 응집도 추출의 대상이 될 코드는 객체 지향 언어인 java 기반의 Calculator코드이다. 이 코드는 ASTM 모델링을 통해 생성한 코드이다. 가시화 시 모듈의 단위는 메서드로 정의한다.

(그림 6)은 프로그램 구조, 응집도, 결합도를 가시화한 그래프이다. 제일 바깥에 위치한 서브 그래프는 패키지, 그 안에 위치한 서브 그래프는 클래스를 의미한다. 클래스 이름 뒤에 괄호 안에 해당 클래스의 메서드의 수를 표시한다. 클래스 안에 타원형 노드는 메서드를 의미하고 노드 안에 이름을 표현하고 메서드의 응집도 수치를 괄호 안에 표시한다. 화살표는 메서드 간의 호출을 표현하고 화살표의 라벨은 메서드 간의 결합도 수치를 표시한다. 응집도가 높은 메서드는 일반 타원으로 표현하고 응집도가 낮은 메서드는 품질이 나쁘기 때문에 빨간색인 이중 타원으로 표현한다. 결합도가 낮은 메서드는 실선 화살표로 표현되고 결합도가 높은 메서드 간의 호출은 품질이 나쁘기 때문에 빨간색 점선 화살표로 표현한다[9]. 이런 내부 코드에 대해 복잡도 가시화 서비스도 제공 가능하다.

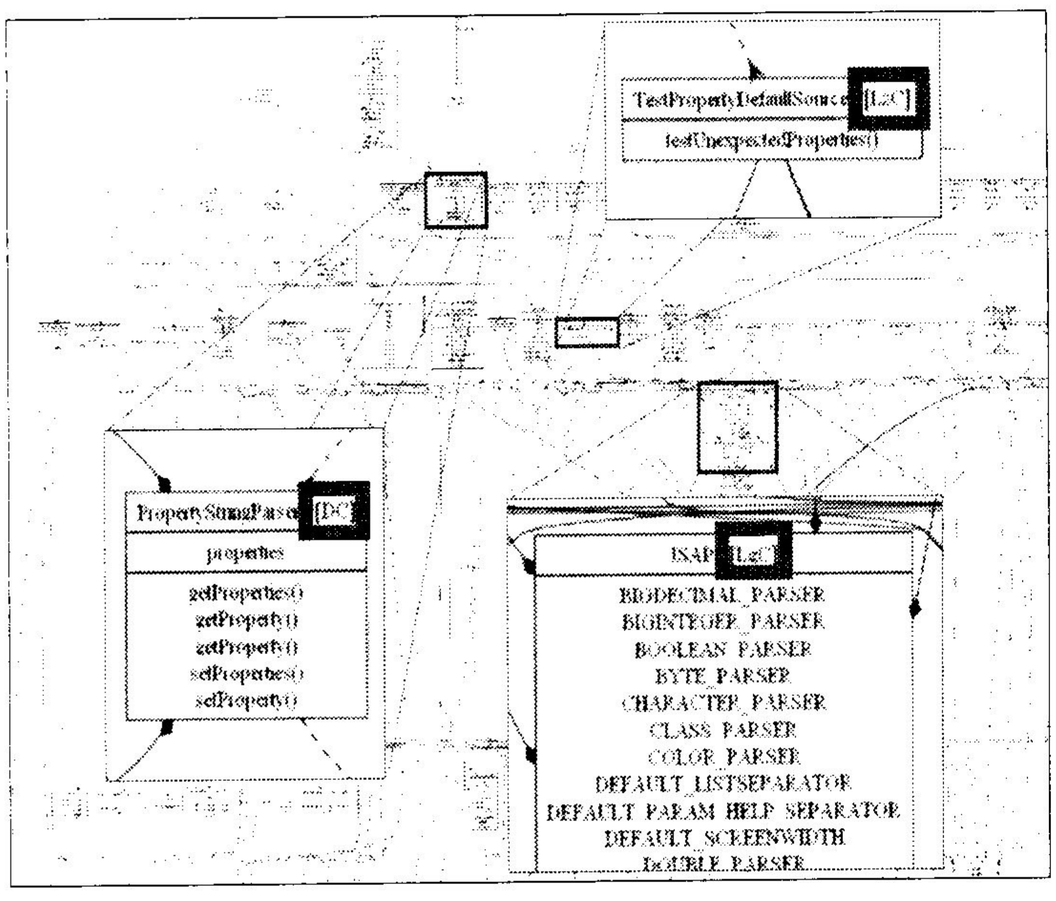
3.2.6 나쁜 냄새 추출 서비스

나쁜 냄새(Bad Smell)는 Martin Fowler가 리



(그림 6) 결합도와 응집도 추출 서비스

팩토링 책에서 처음으로 사용된 비유이다. 이 책에서 소스 코드를 작성하면서 쓰이는 나쁜 코딩 습관들을 언급하였다. 겉으로 보이는 SW의 기능에는 문제가 없다. 하지만 내부적으로는 불필요한 구조로 인하여 복잡한 디자인으로 구축 할 수 있다. 이런 경우 SW의 의도된 디자인은 시간이 갈수록 문제가 발생 가능성이 있다. 이러한 불필요한 구조의 나쁜 코딩 습관은 리팩토링하여 더 좋은 코드로 수정이 가능하다. Martin Fowler는 총 22가지의 나쁜 냄새를 정의하였다[5].



(그림 7) 오픈소스 Java Parser의 배드 스멜을 가시화

(그림 7)은 오픈 소스인 JAVA Parser의 결합도와 나쁜 냄새를 가시화한 그래프이다. 네모 친 부분들은 소스 코드에서 나쁜 냄새가 발견된 부분들이다[10].

PropertyStringParser클래스는 get-, set-메서드만 가지고 있기 때문에 Data Class에 해당하여 [DC]가 표시됐다. TestPropertyDefaultSource클래스는 두 개의 호출 선으로 보아 어떠한 기능을 수행한다. 하지만 자신을 호출하는 선이 없으므로 사용되지 않는 클래스에 판단한다. Lazy Class에 해당하므로 [LzC]가 표시됐다. JSAP클래스는 인스턴스 변수를 10개 이상 가지고 있다. Large Class에 해당하므로 [LgC]가 표시됐다. 이와 같은 코드 내 나쁜 습관을 가시화 하는 서비스도 제공 가능하다.

3.2.7 재사용성 추출 서비스

<표 6>은 응집도와 결합도를 수치를 이용한 재사용화 성숙도 메트릭이다. R 값이 5보다 크고 8보다 작을 경우 재사용에 적합한 모듈로 판단한다.

〈표 6〉 재사용화 성숙도 메트릭

$$\sum_{i=0}^{n} ce = ce_f + ce_s + ce_m + ce_p + ce_t + ce_l + ce_c$$

...① 응집도의 총합

$$\sum_{j=0}^{m} co = co_d + co_s + co_c + co_e + co_m + co_n$$

...② 결합도의 총합

$$R = \frac{1}{2}\left(\frac{\sum_{i=0}^{n} ce_i}{n} + \frac{\sum_{j=0}^{m} co_j}{m}\right)$$

(5 < R < 8)...③ 재사용화 성숙도 메트릭

<표 7>은 (그림 6)의 결합도 응집도 가중치 추출 서비스 결과에 재사용화 성숙도 메트릭을 적용한 결과이다.

〈표 7〉 가시화된 재사용화 성숙도

A (operator - jj_consume_token)

$$R = \frac{1}{2}\left(\frac{4.4+4.4}{2} + \frac{11}{3}\right) = \frac{1}{2}(4.4+3.7) = 4$$

부적합

A' (jjCheckNAdd - jjCheckNAdd)

$$R = \frac{1}{2}\left(\frac{4.4+3.0}{2} + \frac{11}{3}\right) = \frac{1}{2}(3.7+3.7) = 3.7$$

부적합

B (sum - element)

$$R = \frac{1}{2}\left(\frac{10+10}{2} + \frac{7+7}{2}\right) = \frac{1}{2}(5.0+7.0) = 6.0$$

적합

B' (jjMoveStringLiteral - JJStopAtPos)

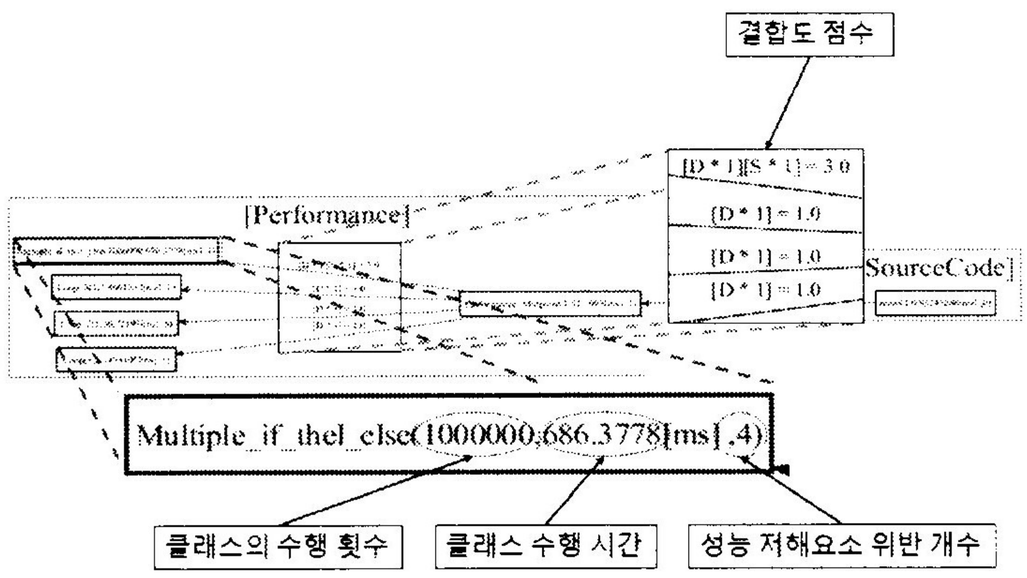
$$R = \frac{1}{2}\left(\frac{6.4+6.4}{2} + \frac{9}{2}\right) = \frac{1}{2}(6.4+4.5) = 5.45$$

적합

따라서 (그림 6)의 A와 A'로 표시한 빨간색 점선 사각형은 점수가 5~8에 속하지 않으므로 재사용에 부적합한 모듈임을 확인할 수 있다. (그림 6)의 B와 B'로 표현한 실선 사각형 부분은 5~8에 속하므로 재사용에 적합한 모듈임을 확인할 수 있다. 이와 같이 재사용 모듈 등을 식별하는 서비스도 제공 가능하다.

3.2.8 성능 추출 서비스

(그림 8)은 성능 가시화이다. (그림 8)과 같이 클래스의 실제 호출 수, 클래스 전체의 수행시간, 성능저하요소 순으로 표기된다. 그리고 모듈 간의 화살표에는 모듈 간의 결합도의 개수와 결합도 점수를 표기하는 서비스를 제공한다[11].

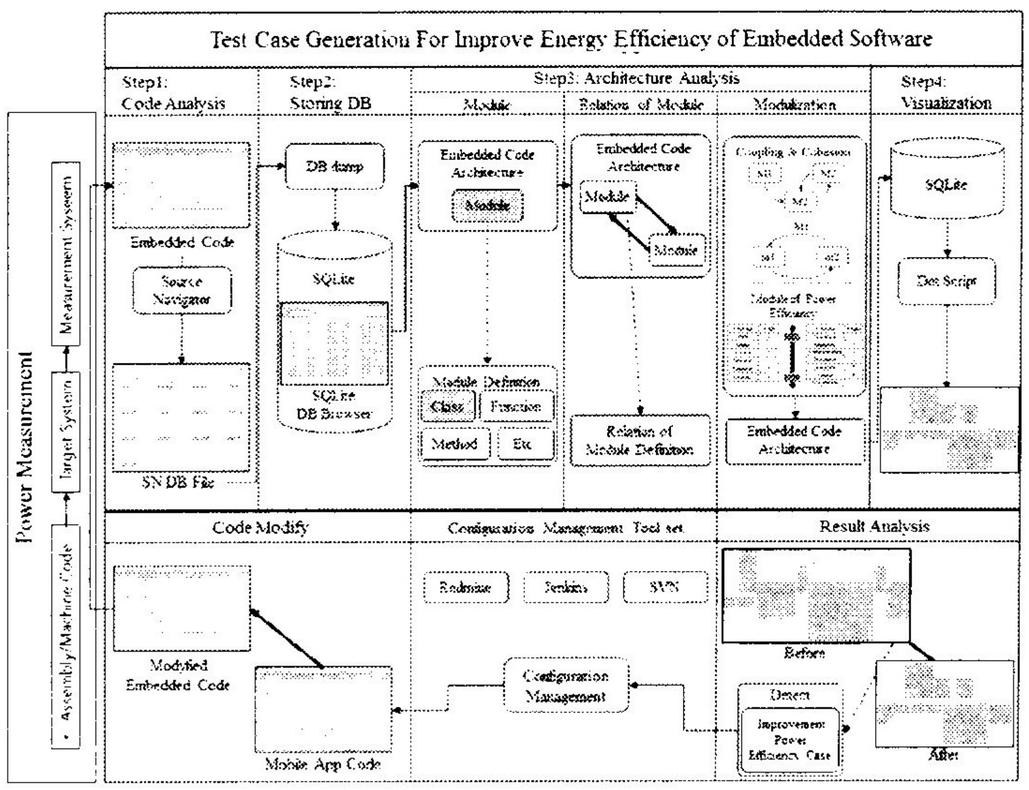


(그림 8) 성능 가시화

3.2.9 저전력 서비스

고사양의 하드웨어로 인해 임베디드 시스템의 소비전력이 증가하고 있다. 가용시간이 단축되고 발열량이 증가하여 기기의 오작동, 수명단축 등의 문제가 발생한다.

(그림 9)는 이런 문제를 해결하기 위한 저전력 추출 서비스이다. 파서를 통해 임베디드 코드를



(그림 9) 저전력 추출 서비스

분석한 후, 전력 소비와 관련된 그래픽, 네트워킹 관련 클래스, 메서드, 변수간의 연관관계를 일대일 매핑하여 데이터베이스에 저장한다. 가시화 도구를 통해 소스 코드의 구조 분석을 바탕으로 가시화한다. 저전력 추출 서비스를 통해 임베디드 시스템의 소비전력의 절감과 에너지 사용 효율성을 높일 수 있다[12]. 이런 서비스도 가능하리라 본다.

4. 결 론

과거의 NIPA 부설 SW공학센터에서는 벤처/중소기업의 SW 품질을 위해 SW 프로세스 강화를 위한 서비스를 제공과 SW 가시화를 위해 오픈소스를 이용한 툴-체인 구축 서비스 지원이 중요했다. 기존의 고수준의 방법론, 프로세스 등의 방법 외에, 이 방법은 업체들의 대표, 개발자, 테스터, 품질 담당자들에게 가시화를 통해 스스로 코드의 문제를 자극해, 고품질을 위한 동기를 부여 한 것 같다. 우리는 SW 가시화 I, II, III 경험으로 업체들에게 공개소스 기반 툴-체인 구축과 확장으로 많은 서비스를 제공이 가능하다. 이러한 오픈소스 기반의 툴 체인 서비스는 중소기업을 위한 유지보수 서비스를 제공 할 수 있다.

참 고 문 헌

[1] Roger S. Pressman "Software Engineering A Practitioners' Approach" 3rd Ed. McGraw Hill

[2] NIPA 소프트웨어공학센터, Software Visualization

[3] T. Ball & S. Eick, "Software Visualization in the Large", IEEE 1996

[4] Robert W. messler, JR. "Reverse engineering", McGraw-Hill Education

[5] Martin Fowler, "Refactoring: Improving the Design of Existing Code", Addison-Wesley Professional

[6] 서채연, 박보경, 변은영, 박용범, 김영철, "기존 오픈 소스 도구들 비교 분석을 통한 정적 분석 및 가시화 도구 구축", 한국스마트미디어학회, Vol.5 No.1, pp. 292-295, 2016.

[7] 권하은, 박보경, 김영수, 김영철, "역공학을 통한 소스 코드로부터 유스케이스 설계 추출", 한국스마트미디어학회 Vol.5 No.1, pp.289-291, 2016.

[8] Johann Eder, Gerti Kappel, Michael Schre, "Coupling and Cohesion in Object-Oriented Systems", 1992

[9] 변은영, 박보경, 장우성, 김영철, "객체 지향 패러다임에서의 코드 재사용을 위한 응집도 레벨 식별 모범 사례", 한국정보처리학회 Vol.23 No.2, pp. 475-478, 2016.

[10] 박지훈, 김영철, "나쁜 코딩 습관 개선을 위한 코드 가시화 연구", 한국정보처리학회 Vol.23 No.2, pp. 497-500, 2016.

[11] 강건희, 이근상, 이진협, 김영철, "프로파일러를 이용한 소프트웨어 메모리 성능 가시화 방법", 한국스마트미디어학회 Vol.5 No.1, pp. 296-297, 2016.

[12] 이근상, 김영철, "저전력 관련 코드 메카니즘과 소프트웨어 가시화 접목", 한국스마트미디어학회 Vol.5 No.1, pp. 286-288, 2016.

[13] Junsun Hwang, Woo Sung Jang, R. Y. Kim, "An Automatic Visualization Mechanism of extracting Diverse Unit Level for Software Complexity", AACL 2016, p155-159

저 자 약 력

서 채 연

이메일 : chyun@selab.hongik.ac.kr

- 2003년 홍익대학교 소프트웨어전공 석사
- 2005년 홍익대학교 소프트웨어전공 박사
- 2007년~2013년 ㈜맥스컴 연구원
- 2007년~2013년 유한대학교 경영정보과 겸임교수
- 2016년 선문대학교 IT교육학부 강의전담계약교수
- 관심분야 : 비즈니스 프로세스 모델링, 의료 비즈니스 데이터베이스 스키마 자동화, 메타 모델,

손 현 승

이메일 : hson@live.co.kr

- 2009년 홍익대학교 일반대학원 소프트웨어공학 석사
- 2015년 홍익대학교 일반대학원 소프트웨어공학 박사
- 2015년~현재 소프트웨어 가시화 멘토링 및 SP 교육 강사
- 관심분야 : 임베디드 소프트웨어 자동화 도구 개발, 소프트웨어 프로세스 및 가시화, 메타모델 설계 및 모델 변환

문 소 영

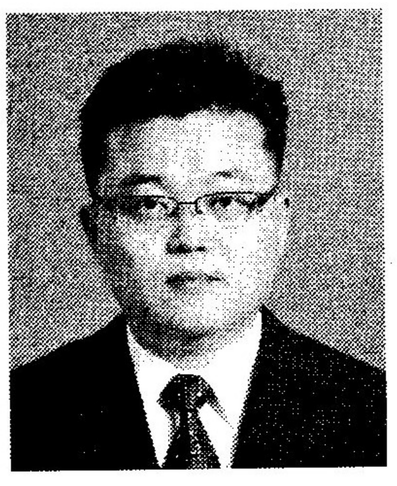
이메일 : msy@selab.hongik.ac.kr

- 2007년 홍익대학교 소프트웨어공학 (석사)
- 2007년~2012년 ㈜액트 원자력발전소 노심 관련 소프트웨어 개발
- 2010년~현재 홍익대학교 전자전산공학과 소프트웨어 공학 (박사과정)
- 관심분야 : Software Visualization, 임베디드 소프트웨어 개발, 모델변환, 테스팅, 프로그래밍언어

이 근 상

이메일 : yi@jbtp.or.kr

- 1999년 광운대학교 전자계산학과 (이학석사)
- 2008년~현재 (재)전북테크노파크 산업정보지원팀장
- 2014년~현재 홍익대학교 일반대학원 소프트웨어공학 박사과정
- 관심분야 : 저전력 소프트웨어, SW Visualization 소프트웨어 모델링, 모델설계 및 검증기법 연구,



김 영 수

이메일 : ysgold@nipa.kr

- 1992년 광운대학교 수학과 (학사)
- 1994년 광운대학교 전자계산학과 (석사)
- 2010년~현재 정보통신산업진흥원 SW공학센터, 정보처리기술사(시스템 응용), 현장 멘토링 총괄, SW Visualization 품질 멘토링
- 관심분야 : 임베디드 소프트웨어공학, 역공학 기법, 소프트웨어 프로세스 담당

김 영 철

이메일 : bob@selab.hongik.ac.kr

- 2000년 LG산전 중앙연구소 Embedded system 부장
- 2001년~현재 홍익대학교 컴퓨터정보통신 교수
- 관심분야 : 테스트 성숙도 모델(TMM) 경량화, 모델 기반 테스팅, 메타모델, 테스트 프로세스, 소프트웨어 프로세스 & 가시화