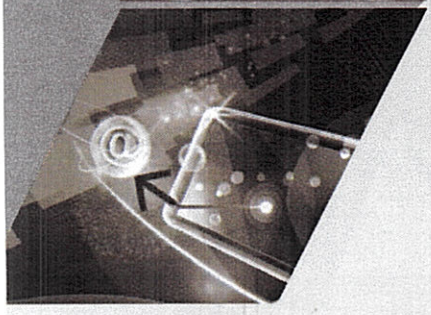
한국정보과학회
KOREAN INSTITUTE OF INFORMATION SCIENTISTS AND ENGINEERS

http://www.kiise.or.kr

2014 한국컴퓨터종합학술대회
Korea Computer Congress 2014

"사물 인터넷 시대의 SW 기술"



2014년 6월 25일(수)~27일(금)
부경대학교&해운대그랜드호텔

- 자연어처리 및 정보검색 최근 동향 워크샵(6.25)
- HPC 최근 연구 동향 워크샵(6.25)
- 데이터베이스 최근 연구 동향 워크샵(6.25)
- GPU Standard API Workshop(6.25)
- 인공지능 최근 연구 동향 워크샵(6. 26)

|후 원|

플래티넘 NAVER KOFST

골 드 SAMSUNG 삼성전자 SAMSUNG 삼성테크윈 btc 부산관광공사

실 버 SAMSUNG 삼성SDS 신한데이타시스템 SHINHAN DATA SYSTEM

브론즈 hancom 한글과컴퓨터 KCC정보통신 We pave your road Suresoft

소프트웨어공학

객체지향 코드의 정적 분석을 위한 Tool-chain화 사례 연구

박보경[O1] 권하은[1] 양효석[1] 문소영[1] 김영수[2] 김영철[1]
[1]홍익대학교 컴퓨터정보통신공학과 소프트웨어공학연구실
[2]정보통신산업진흥원
[1]{bkpark, kwon, yhs, msy, bob}@hongik.ac.kr, [2]ysgold@nipa.kr

A Study on Tool-Chain for statically analyzing Object Oriented Code

Bokyung Park[O1] Haeun Kwon[1] Hyeoseok Yang[1] Soyoung Moon[1] Youngsoo Kim[2] R. Youngchul Kim[1]
[1]SE lab, Dept. of Computer Information Comunication, Hongik University
[2]National IT Industry Promotion Agency

요 약

오늘날 소프트웨어는 규모가 크고 시장 출하 기간의 단축 상황에서도 고품질로 개발해야 한다. 산업 현장에서는 빠른 개발을 위해서 코드 중심으로 개발한다. 즉, 소프트웨어 문서화 및 코드 내부 구조화가 없는 상태로 유지보수가 이루어지고 있다. 이에 역공학의 필요성이 대두되고 있다. 본 논문에서는 객체지향 코드의 내부 구조 시각화 방법을 위해 Tool-Chain을 이용한 내부 구조 가시화 방법과 프로세스를 제안한다. 사례로써 NIPA의 SW Visualization 기법을 실제 객체 코드에 적용한다. 또한 객체지향 코드의 모듈 단위를 클래스로 정의하였고, 가시화를 보였다. 이는 개발 중인 코드의 복잡도(Code Complexity) 상황을 실시간 확인하여, 코드의 정량적 분석이 가능할 것이라 기대한다.

1. 서 론

오늘날 소프트웨어 시장의 규모는 급속하게 성장하여 기존의 하드웨어 시장 규모를 넘어섰다. 또한 IT 융복합화로 소프트웨어 기능과 역할의 중요성이 갈수록 증가한다. 하지만, 소프트웨어의 비가시성, 복잡도 증가 및 국내 중소기업의 소프트웨어 개발환경은 소프트웨어 품질 관리를 어렵게 한다[1].

현재 소프트웨어 산업계에서는 소프트웨어의 품질 향상과 시장 출하 기간 단축을 위한 개발 프로세스에 집중하고 있다. 반면에 문서화 및 구조화되지 않은 상태의 레거시 소프트웨어의 유지보수는 미흡한 상태이다[2].

소프트웨어의 품질 관리와 유지보수 향상을 위해서, 역공학 기법이 필요하다. 역공학은 기존 시스템에 대한 점검을 통해 개발자의 도움 없이 소프트웨어 구조 파악이 용이하다. 또한 소프트웨어 컴포넌트 간 상호관계 식별 및 더 높은 추상 수준의 표현물 복원이 가능하다.

NIPA SW 센터는 소프트웨어 품질 관리 및 유지보수 향상을 위해 SW Visualization 기법을 사용한다. SW Visualization은 소스코드와 개발 프로세스 관리를 목적으로 시각화와 문서화를 통해 고품질 SW관리를 언급한다[1].

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로 SW Visualization에 대해서 소개한다. 3장에서는 객체지향 기반의 정적 분석을 위한 Tool-chain 방법에 대해서 설명하고, 4장에서는 실제 소스코드를 SW Visualization 기법에 적용한다. 5장에서는 결론 및 향후연구에 대해서 언급한다.

* 본 연구는 미래창조과학부 및 한국산업기술평가관리원의 산업원천기술개발사업[10035708, 고신뢰 자율제어 SW를 위한 CPS(Cyber-Physical Systems) 핵심 기술 개발]과 2014년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(NRF-2013R1A1A2011601).

2. 관련 연구

순공학(Forward Engineering)은 요구사항 명세와 같은 추상화 산출물에서 분석, 설계 등 상세화를 통해 소프트웨어를 구현하는 전통적인 절차이다[1]. 역공학(Reverse Engineering)은 대상 시스템을 분석하여 시스템의 구성요소와 그 관계를 파악하고, 시스템을 다르게 표현하거나 고수준의 추상화 작업이다[3]. 이 기법으로 시각화를 하는 의도는 소프트웨어 개발자의 도움 없이 소프트웨어에 대한 이해를 얻기 위해서이다. 또한 대상 시스템을 깊은 이해 없이도 기존 시스템을 점검할 수 있기 때문이다. 순공학은 점진적으로 상세화하여 복잡한 기능들을 구현한다. 그러나 역공학은 주로 코드 상의 정보들을 추출하여 보다 큰 그림을 복원한다[3]. 즉, 소프트웨어의 유지보수 및 품질 향상에 기여하며, 소프트웨어 시스템과 구조를 이해하는데 도움이 된다.

3. 객체지향 언어 기반의 코드 정적 분석을 위한 툴 체인화 방법

그림 1은 Code 정적 분석을 위한 Tool-chain 프로세스이다. 프로세스는 Parser(혹은 Analyzer), Database, View Composer의 3가지 도구(Tool)가 사용되며, 코드 분석, 정보 저장 및 분류, 시각화의 서로 독립적인 기능을 수행한다. Tool-chain은 각 도구를 연계하여 자바 소스 코드에서 시각화된 뷰를 생성하는 과정을 자동으로 수행한다.

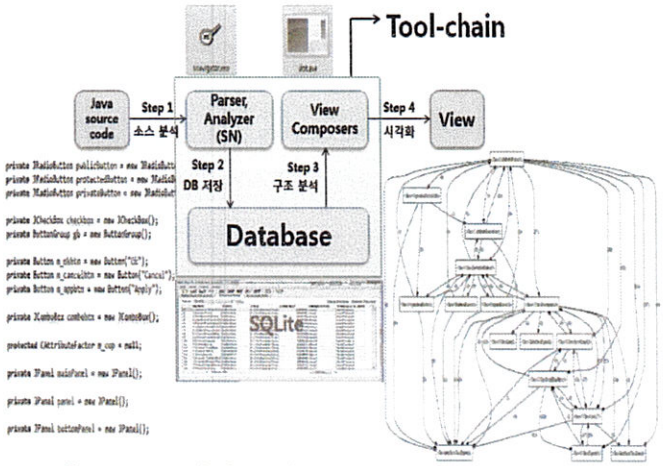


그림 1 Code 정적 분석을 위한 Tool-chain 프로세스

또한 본 프로세스는 소스 분석, DB 저장, 구조 분석, 시각화의 4가지 단계로 구성된다. 1단계의 소스 분석 단계에서는 Parser를 통해 자바 코드를 분석한다. 자바 코드는 클래스, 메소드, 변수 등의 요소들로 구성되는데, 이 단계에서는 자바 코드를 이러한 요소들로 분해하는 작업이 수행된다. 이와 함께 각 요소들 간의 연관 관계도 추출한다. 예를 들어 어떤 클래스 내에서 다른 클래스를 호출하거나, 클래스가 갖고 있는 메소드의 정보 등이 이에 포함된다.

2단계의 DB 저장 단계에서는 추출된 정보를 분류하여 데이터베이스에 저장한다. 클래스, 메소드, 변수 등의 요소들은 하나의 요소 테이블에 저장한다. 요소들 간의 연관 관계는 1:1로 매핑(Mapping)하여 저장한다. 예를 들어 클래스 A가 메소드 a, b, c를 갖고 있다면 A-a, A-b, A-c의 3가지 매핑 정보가 저장된다.

3단계의 구조 분석 단계는 데이터베이스에 분류된 정보를 미리 정한 모듈 정의에 따라 재해석한다. 먼저 2단계에서 분류된 요소들로부터 모듈을 추출한다. 다음으로 추출된 모듈과 나머지 요소들 간의 관계 정보를 새롭게 생성한다. 본 논문에서는 모듈의 단위를 클래스로 정의하기 때문에 클래스와 클래스, 클래스와 메소드, 클래스와 변수의 연관 관계 정보를 생성한다.

마지막 4단계의 시각화 단계에서는 3단계에서 재해석한 정보에서 의미를 찾고 시각화한다. 예를 들어 2단계에서 예로든 A-a, A-b, A-c의 정보에서는 클래스 A가 3개의 메소드를 포함하는 의미이다. 이러한 의미들을 View Composer가 인식 가능한 스크립트로 작성하여 시각화된 뷰를 얻는다.

4. 사례 연구

본 장에서는 기존에 개발된 *Usecase diagram drawing tool*의 코드를 제안한 프로세스에 적용한다. 소스 분석을 위한 Parser로는 Source Navigator 6.0(SN)을 사용한다.[4] SN에 코드를 입력하면 그림 2에 나타낸 *.1, *.by, *.cl, *.f, *.lv, *.mi 등과 같은 san 파일을 생성한다. 각 파일은 소스 코드의 특정 구성 요소에 대한 정보를 저장하고 있다. 예를 들어 *.cl 파일은 클래스, *.mi 파일은 메소드, *.by 파일은 클래스와 메소드 각각의 연관 관계 정보를 저장하고 있다. 본 사례에서는 예로든 *.cl, *.mi, *.by의 3가지 파일을 통해 클래스와 메소드 정보를 토대로 시각화된 뷰를 추출한다.

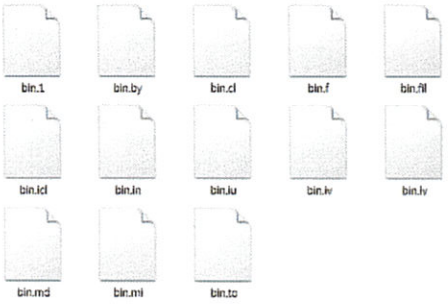


그림 2 Usecase diagram draw tool 코드 san 파일

san 파일은 바이너리 형태로 정보를 저장하고 있기 때문에 SN에 포함된 dbdump.exe(dbdump)를 이용한다. 그림 3은 dbdump를 통해 추출한 *.cl 파일의 내용 중 일부이다. 앞서 언급한 바와 같이 *.cl 파일은 클래스에 관한 정보를 담고 클래스의 식별자, 라인 수, 파일 경로 등과 같은 정보가 확인된다. 마찬가지의 방법으로 *.mi, *.by 파일의 내용을 추출 및 분류하고 데이터베이스에 저장하여 DB 저장 단계를 진행한다. 데이터베이스로는 SQLite를 사용한다[5]. 그림 4는 분류된 코드 요소의 일부분이다.

```
Vector elementAt mi CJTable_re
getValueAt mi r 000025
D:/Software/Source_Navigator_6.0/bin/Pac
kageUML_Project_v2/src/Main/CJTable_re.j
ava:() (int,int)
Vector elementAt mi CJTable_re
getValueAt mi r 000026
D:/Software/Source_Navigator_6.0/bin/Pac
kageUML_Project_v2/src/Main/CJTable_re.j
ava:() (int,int)
Vector get mi CFileFilter accept mi p
000022
D:/Software/Source_Navigator_6.0/bin/Pac
kageUML_Project_v2/src/MenuBar/CFileFil
ter.java:() (File)
```

그림 3 *.cl 파일 내용 추출

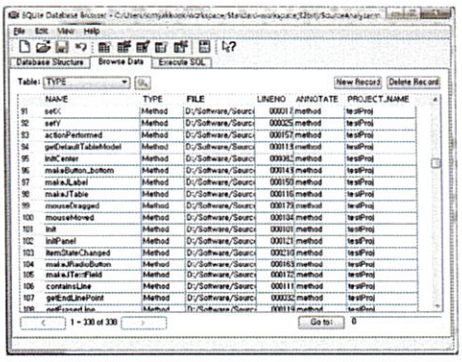


그림 4 코드 요소 분류

3단계의 구조 분석 단계와 4단계의 시각화 단계에서는 데이터베이스에 저장된 클래스, 메소드, 연관 관계 정보를 재해석하여 정량적인 수치 값을 측정한다. 수치 값으로는 응집도(Cohesion)와 결합도(Coupling)를 사용한다. 본 논문에서 제안한 모듈 단위는 클래스이므로 모듈의

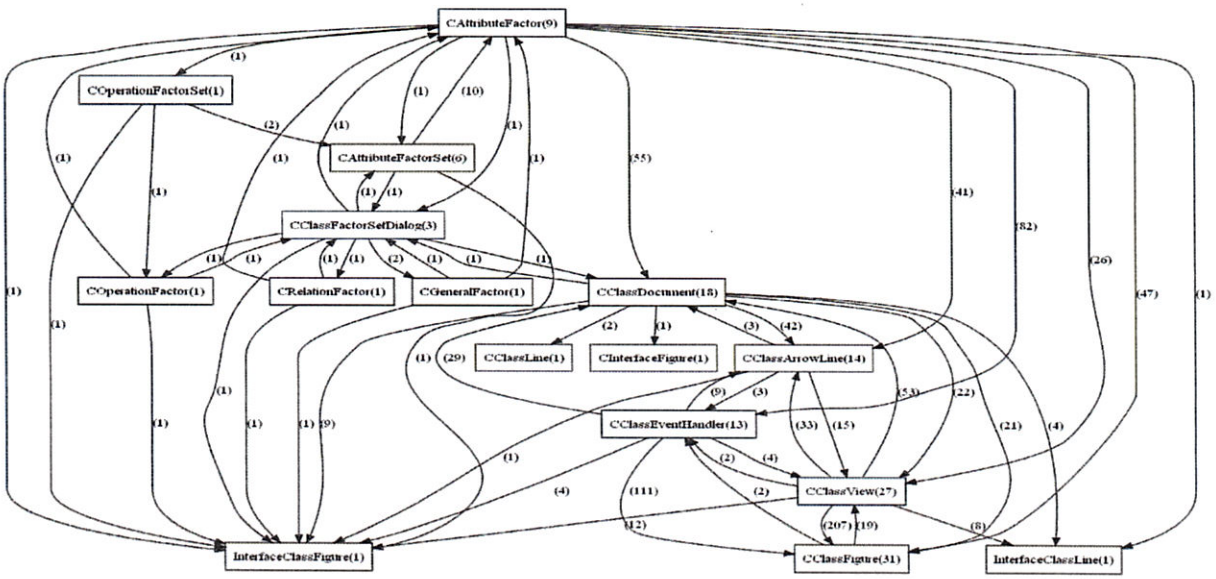


그림 6 코드 시각화 그래프

응집도는 클래스와 메소드 간의 관계에, 결합도는 클래스와 메소드간의 관계에 기인한다.

그림 5는 정의한 응집도와 결합도를 나타내며, 클래스 A와 B가 각각 메소드 a, b, c, d와 e, f, g, h, i, j를 갖고 있다. 즉, 클래스를 모듈 단위로 응집도와 결합도를 측정 가능하다. 다시 말해서 하나의 클래스가 소유한 메소드의 개수로 응집도를 측정한다. 그림의 객체 A는 4개의 메소드를 소유하므로 응집도는 4이다. 마찬가지로 결합도는 클래스와 다른 클래스의 메소드와의 관계에 기인한다. 그림 6는 객체 A의 메소드 c와 d에서 각각 객체 B의 메소드 e, f가 호출됨을 나타낸다. 객체 내에서 다른 클래스의 메소드를 호출함은 둘 사이에 의존관계가 존재함을 나타내므로, 두 클래스간에 결합도가 있다고 판단한다. 그림에서는 2가지 관계가 나타나며 결합도는 2로 측정된다.

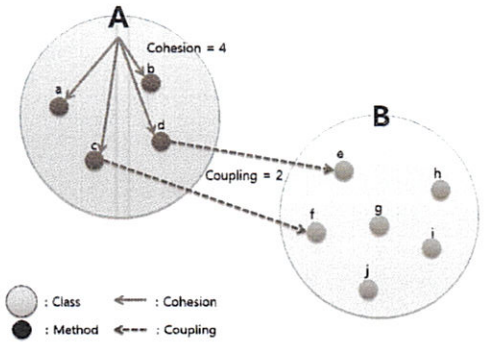


그림 5 클래스 간의 응집도와 결합도 정의

최종적으로 측정된 클래스의 응집도와 결합도를 View Composer를 통해 시각화한다. 본 사례에서는 Graphviz 2.24에 포함된 dot.exe를 통해 그림 6과 같은 결과를 얻었다[6]. 그래프의 노드 내부에는 클래스의 식별자와 해당 클래스의 응집도가 표시된다. 노드와 노드를 연결하는 화살표는 출발 클래스의 가리키는 클래스에 대한 결합도를 의미한다. 그림에서는 CClassView 클래스로부터 출발한 화살표가 CClassFigure 클래스를 가리키고 있다. 이는 두 클래스 사이에 결합도가 존재함을 의미한다. 게다가 결합도 값은 207로 다른 결합도에 비해 높은 값을 가진다. 이는 의존도가 높아 CClassFigure 클래스가 수정될 경우, 많은 코드 수정이 일어날 수 있음을 의미한다.

5. 결 론

일반적인 소프트웨어는 특정 단계까지 개발이 진행되지 않고서는 눈으로 볼 수 없는 비가시적인 특성을 지니고 있다. 하지만 본 논문에서 제안하는 프로세스를 통해 얻은 코드 시각화 그래프는 가시성뿐만 아니라, 응집도와 결합도를 적용한 정량적인 수치를 제공한다. 또한 이러한 일련의 과정이 Tool-chain에 의해 자동적으로 수행된다. 그러므로 이를 통해 개발 중인 소프트웨어의 진척 상황을 실시간으로 확인하고, 코드의 정량적인 분석이 가능할 것이라 기대한다.

참 고 문 헌

[1] NIPA SW공학센터, "SW개발 품질관리 매뉴얼(SW Visualization)", 2013. 12.

[2] 조현훈, 최용락, 류성열, "McCabe 및 BP Win도구를 이용한 소프트웨어 역공학 사례연구", 정보과학회논문지:컴퓨터의 실제 레터, 제 6권, 제 5호, 528-535, 2000. 10.

[3] Bernd Bruegge, Allen H. Dutoit, "Pbject-Oriented Software Engineering Suing UML, Patterns And Java", PEARSON, 2009.

[4] emthornber, "Source Navigator NG", http://sourcenav.sourceforge.net/

[5] SQLite, "About SQLite", http://www.sqlite.org/about.html

[6] graphviz, "DOT tutorial and specification", http://www.graphviz.org/Documentation.php