THE KIPS FALL CONFERENCE 2019

제52회
2019 추계학술발표대회
프로그램

일 자 2019년 11월 1일(금) ~ 2일(토)

장 소 제주대학교(아라캠퍼스)

주 최 사단법인 한국정보처리학회 KIPS Korea Information Processing Society

주 관 제주대학교 소프트웨어중심대학사업단

협 찬 아이티센 ITCen | 삼성SDS | KCC정보통신 | 대우정보시스템 | Metanet | NAVER | 올포랜드 | Comtec Comtec Systems Co., Ltd. | KCC오토 | 비트컴퓨터 | 제주특별자치도개발공사 JEJU PROVINCE DEVELOPMENT CO. | Bigsun SYSTEMS

목 차

04. 키워드 요약의 세 가지 방법론 비교 KIPS_C2019B0139
강종렬*, 남지성, 박지나, 김웅섭(동국대학교)

05. 정규화 기법 적용에 따른 GAN 모델의 성능 비교 연구 KIPS_C2019B0149
곽정기*, 고한석(고려대학교)

우수논문 06. 로컬 커버링 규칙 획득기법을 활용한 섬망 환자의 분류 KIPS_C2019B0152
손창식*, 강원석(대구경북과학기술원), 이종하, 문경자(계명대학교)

T1-6 소프트웨어공학

좌장 심준용 수석연구원(LIG넥스원)
일시 : 11월 2일(토) 09:20~10:50, 장소 : 1층 0159호

01. 말뭉치 정규화와 의미 규칙 기반 요구사항 정제를 통한 원인-결과 그래프 자동 생성 KIPS_C2019B0253
장우성*, 김영철(홍익대학교)

02. Go 언어 기반 블록체인 코드의 품질 검증을 위한 효율적인 정적분석기 개발 KIPS_C2019B0258
안현식*(홍익대학교), 박지훈(한국정보통신기술협회), 박보경, 김영철(홍익대학교)

03. 무장데이터링크 시뮬레이션 환경에서 유도탄모델 확장성과 프로토콜 변경용이성을 고려한 네트워크기반 유도탄모델 시뮬레이션 구조 설계 KIPS_C2019B0290
김성태*, 심준용, 이원식, 위성혁(LIG넥스원㈜), 김기범(국방과학연구소)

04. 고객 요구사항으로부터 UCP 기반 소프트웨어 공수 산정 KIPS_C2019B0296
박보경*, 박영식, 김영철(홍익대학교)

05. 객체지향 코드 품질 분석을 위한 효율적인 정적분석기 개발 및 가시화 사례 KIPS_C2019B0317
이원영*, 문소영, 김영철(홍익대학교)

06. 시뮬레이션 및 실 환경에서 상호 운용이 용이한 센서 시뮬레이터 설계 KIPS_C2019B0318
심준용*, 위성혁(LIG넥스원㈜)

T1-7 사물인터넷

좌장 민복기 책임(에임시스템)
일시 : 11월 2일(토) 09:20~10:50, 장소 : 1층 0160호

01. 엣지 컴퓨팅 기반 무인 마켓 사례 연구: 자원 분배 효율성 극대화 KIPS_C2019B0028
▶ 박지훈*, 류형오, 김경률, 김세화(한국외국어대학교)

02. 웹 GUI 기반 스마트 팩토리 공정 관리 및 공유 시스템 KIPS_C2019B0108
▶ 이상정*, 홍석준, 정택성, 한건욱, 이인혜, 정민교, 민홍(호서대학교)

03. 원격 의료 서비스를 위한 EHR 데이터 비식별화 기법 제안 KIPS_C2019B0140
▶ 윤준호*(경일대학교), 김현성(경일대학교, 말라위대학교)

04. 데이터 중심의 스마트 시티를 위한 보안 공격 분류 KIPS_C2019B0148
▶ 황현재*(경일대학교), 김현성(경일대학교, 말라위대학교)

05. 초음파와 소음 감지 센서를 이용한 학교 급식실 대기 시간과 연관 요소 분석 KIPS_C2019B0172
▶ 정지민*, 신예빈, 이은지, 김지은(대전노은고등학교)

06. PID 제어를 이용한 자율주행자동차의 차선 추적 KIPS_C2019B0228
▶ 김현식*, 장재영, 김찬수, 전중남(충북대학교)

02. 태양 에너지 기반 무선 센서 네트워크에서 에너지와 링크 품질을 고려한 향상된 FEC 기법 KIPS_C2019B0340
길건욱*, 강민재, 고정현, 노동건(숭실대학교)

03. 모바일 싱크 기반의 태양 에너지 수집형 무선 센서 네트워크에서 무선 전력 전송을 이용한 효율적인 클러스터 관리 기법 KIPS_C2019B0342
손영재*, 강민재, 고정현, 노동건(숭실대학교)

04. 얼굴인식을 위한 다중입력 CNN의 기본 구현 KIPS_C2019B0328
Usman Cheema*, Seungbin Moon(Sejong University)

05. 딥CNN에서의Different Scale Information Fusion (DSIF) 의 영향에 대한 이해 KIPS_C2019B0330
Kai Liu*, Usman Cheema, Seungbin Moon(Sejong University)

06. RNN을 이용한 태양광 에너지 생산 예측 KIPS_C2019B0363
Mudassar Liaq*, 변영철, 이상준(제주대학교)

T2-4 인공지능

좌장 오세창 연구위원(솔트룩스)
일시 : 11월 2일(토) 11:00~12:30, 장소 : 1층 0154호

01. 기술적 지표 기반의 주가 움직임 예측을 위한 모델 분석 KIPS_C2019B0193
최진영*, 김민구(아주대학교)

02. X-ray 영상에서 그리드 아티팩트 제거를 위한 복합형 기법 KIPS_C2019B0212
김혜원*(한동대학교), 김경우, 김형규, 정중은(㈜제이피아이헬스케어), 박준혁, 김동현, 김호준(한동대학교)

03. 소셜 데이터의 감성 분석을 위한 신조어 및 이모티콘 감성 사전 구축 KIPS_C2019B0223
양진솔*, 윤경일, 조영훈, 정광식(한국방송통신대학교)

04. 모바일 환경에서의 감성 기반 지능형 챗봇 연구 KIPS_C2019B0225
윤경일*, 양진솔, 조영훈, 정광식(한국방송통신대학교)

05. 비디오에서 YOLOv3기반 차량 인식 및 계수 방안 KIPS_C2019B0245
이혜진*, 이은지, 박소현, 임선영, 박영호(숙명여자대학교)

06. 비용 예측 모형 기반 열처리로 작업 계획 최적화 KIPS_C2019B0248
허형록*, 김세영, 류광렬(부산대학교)

T2-5 인공지능

좌장 양근탁 박사(제주대학교)
일시 : 11월 2일(토) 11:00~12:30, 장소 : 1층 0158호

01. GAN 모델에서 손실함수 분석 KIPS_C2019B0251
이초연*(한국방송통신대학교), 박지수(동국대학교), 손진곤(한국방송통신대학교)

02. 어텐션 중심을 이용한 글자 단위 영역 검출 KIPS_C2019B0262
김지인*, 정창성(고려대학교)

03. Text-CNN 알고리즘 적용한 교육장터 플랫폼 기반 맞춤형 교육 컨텐츠 추천 메커니즘 개발 KIPS_C2019B0294
홍제성*, 박보경(홍익대학교), 곽제일(제일에듀스), 손현승(모아소프트㈜), 김영철(홍익대학교)

04. 가변 길이의 봇넷 트래픽 분류를 위한 마코브 체인 모델 설계 KIPS_C2019B0299
이현종*, 어성율, 김정미(㈜케이사인), 김준호, 김영호(단국대학교)

05. 기록물 검색 챗봇 설계 및 구축 KIPS_C2019B0311
박은비*, 박성희(한남대학교)

객체지향 코드 품질 분석을 위한 효율적인 정적분석기 개발 및 가시화 사례

이원영*, 문소영*, 김영철**
*, **홍익대학교 소프트웨어 공학 연구실
e-mail: {leewy, msy, bob}@selab.hongik.ac.kr

The Constructing & Visualizing Practices in Effective Static Analyzer for analyzing the Quality of Object Oriented Source Code

Won Young Lee*, So Young Moon*, R. Young Chul Kim**
*, **Dept of Software Engineering, Hong-Ik University

요 약

오늘날 객체지향 코드 내부 복잡도가 지속적으로 증가하는 데에 반해 IT 벤처/중소기업에서는 요구사항 및 설계문서 미비의 코드 개발과 테스트 중심의 경우가 빈번하다. 이는 시스템의 코드를 이해하고 수정, 유지보수를 하는데 많은 시간과 비용이 투자되고 있다. 본 연구는 객체지향 코드의 내부 구조 시각화를 위해 Tool-Chain방법을 이용한 정적 분석기 구축 및 가시화를 제안 한다. 이를 통해, 역공학 도구, 테스트 프로세스 등을 도입이 어려운 중소기업의 소프트웨어 품질 향상에 도움을 줄 수 있을 것으로 기대된다.

1. 서론

다기능화 및 대규모화 되는 소프트웨어를 성공적으로 개발하기 위해서는 요구사항 분석에서부터 유지보수까지 전 과정에 걸쳐 체계적인 관리와 효율적인 업무수행을 지원해 주는 소프트웨어 공학기술이 필요하다[1]. 그러나 국내 중소기업 소프트웨어 개발 환경은 소프트웨어의 비가시성, 복잡도 증가 등의 이유로 소프트웨어 품질 관리가 어렵다[2].

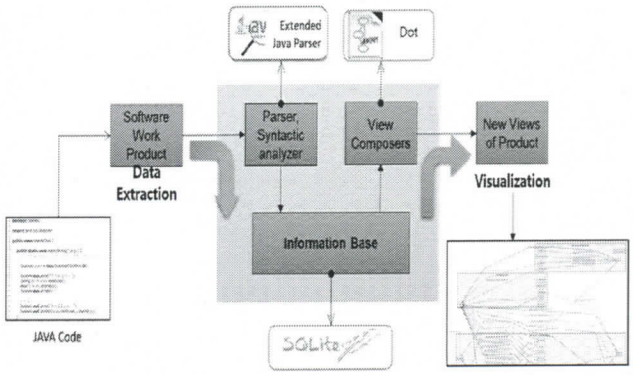
이러한 관점에서 소프트웨어의 품질 관리와 유지보수 향상을 위해서는 소프트웨어의 핵심 영역의 시각화, 개발 프로세스의 가시성, 복잡성 감소, 개발 및 설계에 관한 문서 부재 등에 초점을 맞춰 고품질 소프트웨어 개발에 기여하고자 한다.

NIPA(National IT Industrial Provision Agency)에서는 소프트웨어 시각화 기법을 적용하여 레거시 코드의 문제점을 감지하고 코드 시각화를 통한 역공학 기법을 적용하여 소프트웨어의 취약점을 파악한다. 기존 시스템에는 소스 네비게이터라는 오픈소스를 통해 DOT 스크립트 및 SQLite를 연결하여 툴 체인을 구성하여 이를 통해 JAVA 기반의 소프트웨어를 시각화에 적용한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구에 비추어 소프트웨어 시각화 및 역공학을 설명한다. 3장에서는 툴 체인을 구성하는 방법에 대해 논한다. 4장에서는 적용사례를 설명한다. 5장에서는 결론과 앞으로의 연구에 대해 언급하고 있다.

2. 관련연구

- 소프트웨어 가시화: 성공적인 소프트웨어 개발 관리를 위해서는 소스 코드와 소프트웨어 개발 프로세스에 대한 관리가 필요하다[1]. 소프트웨어 공학 프로세스가 그의 대표적인 예지만, 국내 중소기업이 수행하기에는 인력 및 비용이 부족하며 전문적이다. 그에 대한 현실적인 방안으로 소프트웨어 가시화가 대두되었다. 가시화는 시각화를 통해 소프트웨어 비가시성을 극복하고 개발 과정의 투명성을 높임으로써 소프트웨어 품질 향상에 큰 기여를 한다[3].



(그림 1) 기존의 소프트웨어 가시화 구성도

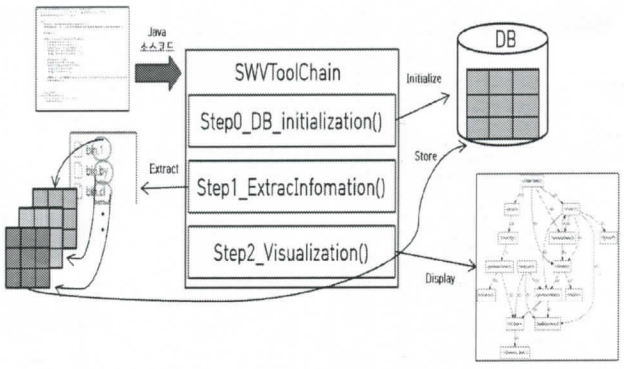
또한, 가시화의 문서화를 통해 업무 이해도가 높아지고 관리가 용이해지며 개발자들 간의 원활한 의사소통을 이루

게 한다. 그림 1은 기존의 소프트웨어 가시화 구성요소들을 도식화했다.

- 역공학: 순공학(Forward Engineering)은 요구사항 명세와 같은 높은 수준의 추상화 산출물에서 시작하여 점진적으로 분석, 설계 등 상세화를 통하여 소프트웨어를 구현하는 방법이다[1]. 반면에, 소프트웨어의 컴포넌트와 이들 간 상호관계 식별 및 또 다른 형태 또는 더 높은 추상 수준의 표현물을 복원하기 위한 시스템 분석 기법은 역공학(Reverse Engineering)이다[1]. 순공학은 전체적인 시스템에 대하여 점진적으로 접근을 하지만, 역공학은 순공학과 반대방향으로 결과물에서 코드까지 역으로 추적한다. 주로 코드 상의 정보를 추출하고 해석하는 과정에서 본래의 의도와는 다른 결과가 도출될 가능성이 있다. 그로 인해, 새로운 관점으로 소프트웨어를 인지할 수 있고, 예상하지 못했던 문제점에 대한 다양한 통찰력을 얻을 수 있다. 역공학은 순공학에서 얻기 어려운 부작용(Side effects)를 발견할 수 있지만, 순공학에서만 발견할 수 있는 설계 이슈, 채택되지 못한 설계 대안 등이 있다.

- Source Navigator: 레드햇 소스 네비게이터(Source Navigator)는 크고 복잡한 소프트웨어를 이해하고 리엔지니어링 할 수 있도록 그래픽 프레임 워크를 제공하는 강력한 코드 분석 및 컴파일 도구이다[4]. 소스 네비게이터 파서는 소스 코드를 스캔하고 기존 C, C++, Java, Tel 및 어셈블리 프로그램에서 정보를 추출하고, 이 정보를 사용하여 프로젝트 데이터베이스를 작성한다. 데이터베이스는 내부 프로그램 구조, 함수 선언의 위치, 클래스 선언의 내용 및 프로그램 구성 요소 간의 관계를 나타낸다[6].

3. 객체지향 코드 분석을 위한 Tool-chain 방법



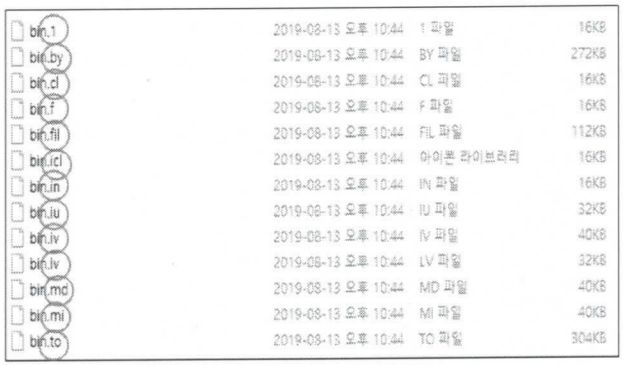
(그림 2) 코드 정적 분석을 위한 Tool-chain 구성도

정적 분석기 구축을 위한 Tool-chain은 서로 독립된 공개 도구로 구성되며 파서(Source Navigator), 데이터베이스(SQLite), 가시화 도구(Graphviz)의 세 가지 유형으로 분류된다. 파서에 소스 코드를 입력하여 분석을 하고 분석된 데이터를 바탕으로 필요한 데이터들을 추출하여 데이터베이스에 저장한다. 그림 2는 코드 정적 분석을 위한 Tool-chain 방법이다. 그리고 분류된 정보로 소스 코드에 대한 그래프를 그림으로써 가시화한다.

그림 2 내의 SWVToolChain 클래스에서 Step0부터 Step2까지의 메소드 단계로 구성한다. Step0에서는 데이터베이스를 초기화시켜주고, Step1에서는 정보를 추출하여 Step2에서 그래프를 그려 가시화하는 순서로 정의한다.

- Step0에서 데이터를 저장하기 위해 SQLite Database를 생성하고 생성된 데이터베이스에 테이블을 생성한다. 이는 데이터베이스에 저장함으로써 구조 분석 단계에서 정보 조회 및 입력시간을 단축시킬 수 있다.

- Step1에서는 Java 소스코드를 Source Navigator에 실행시켜 생성된 SNDB 파일들을 SQLite Database에 저장한다. 더불어 파일 사이의 link관계와 type관계도 함께 저장한다. 생성된 SNDB 파일은 객체지향코드의 구성요소에 대한 정보를 갖고 있다.

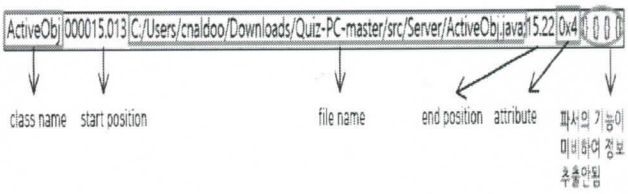


(그림 3) 소스 네비게이터 구문분석 결과파일

그림 3과 같이 확장자가 1, by, cl, f, fil, icl, in, iu, iv, lv, md, mi, to로 구성된다. 그러나 Source Navigator를 통해 추출되는 SNDB파일은 파일마다 다른 코드 구성을 가지기 때문에 각각 다른 분석방법이 필요하다.

구체적으로 하나의 예로 확장자가 cl인 파일은 클래스에 대한 정보를 가진다. 클래스의 이름, 클래스 시작 및 종료 줄 번호, 클래스가 정의된 파일 경로, 클래스 접근자 등의 정보가 있다.

그림 4는 예제 코드로부터 추출한 cl 파일의 일부분이다. 그림 5는 각 필드가 기본적으로 공백으로 구분되며, 필드 내부에서는 '.' 혹은 ';'으로 구분된다는 분석을 토대로 데이터를 추출하는 Java 소스 코드이다.



(그림 4) cl 파일의 일부분

```
public void getElementSN(String cmd, File f, String type) {
    String cmdline = "\"" + cmd + "\"   \"" + f.getAbsolutePath() + "\"";
    System.out.println("====>>" + type);

    try {
        Process p = Runtime.getRuntime().exec(cmdline);
        BufferedReader input = new BufferedReader(new InputStreamReader(p.getInputStream()));

        String line;

        while((line = input.readLine()) != null) {
            System.out.println(line);
            String[] Arr = line.split(" ");
            String[] Arr2 = Arr[1].split("\\.");
            String[] Arr3 = Arr[2].split(";");
            String[] Arr4 = Arr3[1].split("\\.");

            String name = Arr[0];
            String startLineNo = Arr2[0];
            String fileNameWithPath = Arr3[0];
            String endLineNo = Arr4[0];
            String attribute = Arr[3];

            SNDB4CL cl = new SNDB4CL(name, startLineNo, fileNameWithPath, endLineNo, attribute);
            insertSndb4cl(cl);
```

(그림 5) cl 파일에서 데이터를 추출하는 소스코드

첫 번째 필드는 클래스 이름을 나타내며 예제에서의 클래스 이름은 'ActiveObj'이다. 두 번째 필드는 코드에서 클래스에 대한 정의가 시작되는 줄과 열 번호이다. '.'을 기준으로 앞부분이 줄 번호를 의미하며, 뒷부분이 열 번호를 의미한다. 세 번째 필드는 클래스가 정의된 소스 코드의 파일 경로와 코드에서 클래스 정의가 끝나는 줄과 열 번호를 나타낸다. 이는 ';'으로 구분되는데, 앞부분이 파일 경로이다. 또, ';'으로 구분되는 뒷부분은 다시 '.'으로 구분되는데, 두 번째 필드와 마찬가지로 앞부분이 줄 번호이며 뒷부분이 열 번호이다. 네 번째 필드는 클래스의 접근자를 의미하며, 예제의 '0x4'는 Java 문법의 public을 의미한다. 다섯 번째 이후의 필드는 Java 코드와는 관계없는 내용이다.

```
public class DB {
    public boolean init(){
        Connection conn = getConnection();
        Statement stat;
        try {
            stat = conn.createStatement();

            stat.executeUpdate("drop table if exists SNDB4CL;");
            stat.executeUpdate("create table SNDB4CL "
                                                + "(NAME TEXT, "
                                                + "START_LINE_NO TEXT, "
                                                + "FILENAME_WITH_PATH TEXT, "
                                                + "END_LINE_NO TEXT, "
                                                + "ATTRIBUTES TEXT, "
                                                + "PROJECT_NAME TEXT)");

        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
        return true;
    }
```

(그림 6) 쿼리(Query)문으로 SNDB4CL 테이블 생성

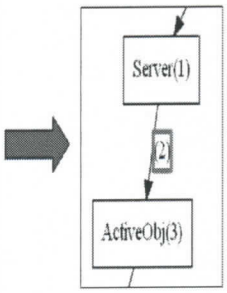
Step1에서 추론된 SNDB 파일에 대한 각각의 테이블을 만들어서 필요한 데이터를 데이터베이스 테이블에 추출하여 저장한다. 그림 6는 쿼리(Query)문을 통해 cl파일에 대한 테이블을 생성하는 Java 소스코드이다. 그림 7은 생성한 SNDB파일들의 테이블 중에서 DB Browser for SQLite로 들여다본 cl 파일의 데이터베이스이다.

	NAME	START_LINE_NO	FILENAME_WITH_PATH	END_LINE_NO	ATTRIBUTES	PROJECT_NAME
	필터	필터	필터	필터	필터	필터
1	ActiveObj	000015	C:/Users/cnaldoo/···	15	0x4	QUIZ-PC
2	AnswerListener	000086	C:/Users/cnaldoo/···	86	0x1	QUIZ-PC
3	ArrayList	000057	C:/Users/cnaldoo/···	57	0x4	QUIZ-PC
4	ConnectionToServer	000012	C:/Users/cnaldoo/···	12	0x4	QUIZ-PC
5	GController	000009	C:/Users/cnaldoo/···	9	0x4	QUIZ-PC
6	GMainClient	000006	C:/Users/cnaldoo/···	6	0x4	QUIZ-PC
7	GModel	000016	C:/Users/cnaldoo/···	16	0x4	QUIZ-PC
8	GView	000040	C:/Users/cnaldoo/···	40	0x4	QUIZ-PC
9	HelpListener	000040	C:/Users/cnaldoo/···	40	0x1	QUIZ-PC
10	IOClient	000019	C:/Users/cnaldoo/···	19	0x4	QUIZ-PC
11	IOServer	000019	C:/Users/cnaldoo/···	19	0x4	QUIZ-PC
12	MenuListener	000054	C:/Users/cnaldoo/···	54	0x1	QUIZ-PC
13	OSDetectorClient	000002	C:/Users/cnaldoo/···	2	0x4	QUIZ-PC
14	OSDetectorServer	000002	C:/Users/cnaldoo/···	2	0x4	QUIZ-PC
15	Options	000005	C:/Users/cnaldoo/···	5	0x24	QUIZ-PC
16	QuestionsClient	000017	C:/Users/cnaldoo/···	17	0x4	QUIZ-PC
17	QuestionsServer	000007	C:/Users/cnaldoo/···	7	0x4	QUIZ-PC
18	ResultListener	000147	C:/Users/cnaldoo/···	147	0x1	QUIZ-PC
19	Server	000016	C:/Users/cnaldoo/···	16	0x4	QUIZ-PC
20	SettingsListener	000123	C:/Users/cnaldoo/···	123	0x1	QUIZ-PC
21	SingleQuestion	000008	C:/Users/cnaldoo/···	8	0x4	QUIZ-PC
22	Sounds	000017	C:/Users/cnaldoo/···	17	0x4	QUIZ-PC
23	String	000031	C:/Users/cnaldoo/···	31	0x4	QUIZ-PC
24	SubmitListener	000102	C:/Users/cnaldoo/···	102	0x1	QUIZ-PC
25	TimerListener	000166	C:/Users/cnaldoo/···	166	0x1	QUIZ-PC

(그림 7) cl 파일에 대한 데이터베이스

```
public class Server {
    private ServerSocket s;
    private ActiveObj a;
    private static JFrame frame;
    public Server() throws IOException {
        s = new ServerSocket(13337);
        System.out.println("Waiting for clients...");
        Container c = frame.getContentPane();
        c.add(new JLabel("Server Is Running...", JLabel.CENTER));

        while(true) {
            Socket klient5 = s.accept();
            a = new ActiveObj(klient5);
            a.run();
        }
    }
    public static void main(String[] args) {
        try {
            frame = new JFrame("The server");
            frame.setVisible(true);
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setSize(300, 200);
            new Server();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```



(그림 8) 예제 Java 소스코드와 그의 호출관계

마지막으로 Step2에서는 저장한 SQLite Database내용을 읽어서 Graphviz에 넣기 위한 스크립트 파일로 변경시켜 module 테이블에 저장한다. 저장된 파일을 Graphviz의 dot.exe를 통해 그래프 이미지를 그린다. 그림 8은 예제 Java 코드의 일부분에서 클래스 호출 관계를 그래프로 나타낸 그림이다.

4. 적용사례

그림 10처럼 샘플 코드을 적용한 사례를 보인다. 클래스들을 모듈로 그룹화 하여 내부적인 호출과 외부적인 호출의 빈도수를 나타낸다. 각 네모들은 클래스를 나타내며, 네모 안에 있는 숫자는 그룹화된 모듈내에서의 호출 빈도

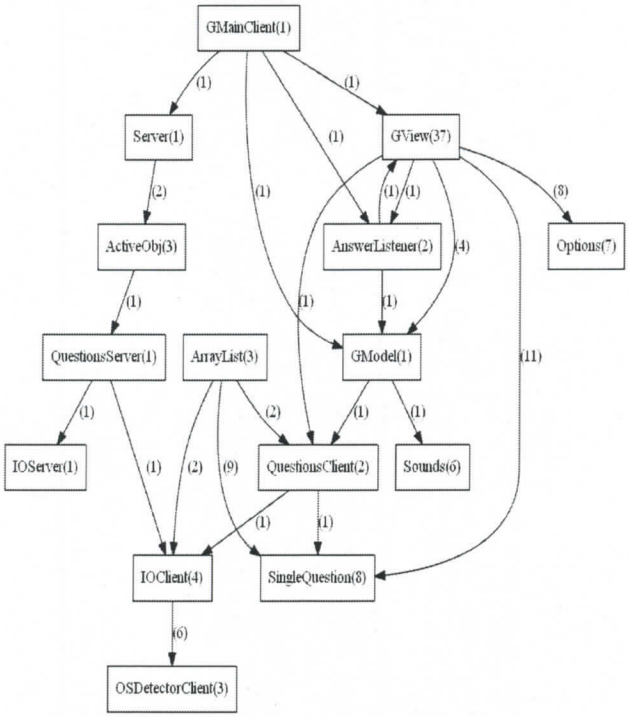
수를 나타내고, 클래스와 클래스를 연결시켜주는 화살표와 숫자는 외부적인 호출에 대한 정보이다. 그림 9는 Java 소스 코드에서 외부적인 메소드 호출 빈도수를 카운트하여 그래프로 출력하는 부분이다.

```
String dependency_query = "select AM.MODULE_NAME as FROM_MODULE, BM.MODULE_NAME as TO_MODULE , count(*) as CNT "
    + "from MODULE_MAPPING as AM, LINK, MODULE_MAPPING as BM where "
    + "AM.PROJECT_NAME ='" + project_name +"' and AM.ARCHITECTURE_NAME ='"+architecture_name+"' and "
    + "BM.PROJECT_NAME = AM.PROJECT_NAME and BM.ARCHITECTURE_NAME = AM.ARCHITECTURE_NAME and "
    + "LINK.PROJECT_NAME ='" + project_name +"' and "
    + "AM.TYPE_NAME = LINK.FROM_NAME and "
    + "LINK.TO_NAME = BM.TYPE_NAME "
    + "and AM.MODULE_NAME <> BM.MODULE_NAME "
    + "group by AM.MODULE_NAME, BM.MODULE_NAME";

rs = stat.executeQuery(dependency_query);
while (rs.next()) {
    String from_module = rs.getString("FROM_MODULE");
    String to_module = rs.getString("TO_MODULE");
    String cnt = rs.getString("CNT");

    content += "\t \"" + from_module + "\"->\""+ to_module + "\" [shape=BOX, label=\"("+cnt+")\"];\n";
}
```

(그림 9) 메소드 호출 빈도수를 출력하는 소스코드



(그림 10) 예제파일에서의 메소드 호출 가시화

5. 결론 및 향후 연구

국내 소프트웨어 프로젝트의 대부분은 중소기업에서 개발된다. 그러나 국내 중소기업의 소프트웨어 품질관리는 잘 이루어지고 있지 않다. 그 배경으로는 관련 문서가 갱신되지 않거나 구체적이지 않은 문서를 기반으로 개발이 진행된 경우가 많다. 이는 소프트웨어 유지보수의 비용을 증가시키기 마련이다.

본 논문은 이러한 문제를 해결하기 위해 역공학 기반의 소프트웨어 가시화 기법을 제안하였다. Source Navigator를 통해 소스 코드를 분석한 SNDB파일들을 생성하고, Dbdump로 각각의 파일들에 맞는 구문 분석을 통해 파일 내용을 데이터베이스에 저장한다.

현재까지의 구현으로는 클래스 호출에 대한 정보만을 가시화하고 있지만, 이 결과를 토대로 클래스와 메소드 간의 관계를 파악할 수 있다. 클래스와 메소드 간의 관계를 파악한다면 모듈의 응집도(Cohesion)와 결합도(Coupling)에 대한 정보까지 추출할 수 있다. 결과적으로 Tool-chain을 통해 코드의 품질을 높일 수 있고 예기치 못한 문제점을 파악할 수 있다.

그러므로 개발 중인 소프트웨어의 진척 상황을 실시간으로 확인하고, 코드의 정량적인 분석이 가능해질 것이라고 기대한다.

Acknowledgement

본 논문은 2019년도 산업통상자원부의'창의산업융합 특성화 인재양성사업'(과제번호 N0000717)과 2017년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(NRF-2017R1D1A3B03035421)

참고문헌

[1] NIPA SW공학센터, "SW개발 품질관리 매뉴얼(SW Visualization)", 2013.12.

[2] So Young Moon, R. Youngchul Kim, "Code Structure Visualization with A Tool-Chain Method" International Journal of Applied Engineering Research, ISSN 0973-4562 Vol. 10 No.90 2015.

[3] 박보경, 권하은, 양효석, 문소영, 김영수, 김영철, "객체지향 코드의 정적 분석을 위한 Tool-chain화 사례 연구" 한국정보과학회 학술발표논문집, 2014.6, 463-465.

[4] 권하은, "역공학 기반 소프트웨어 설계 추출 자동화 도구 개발" 2016.02.

[5] SQLite, https://www.sqlite.org/

[6] 문소영, 손현승, 김영수, 김영철 "개발자를 위한 소프트웨어 가시화", 2018.