한국정보과학회
KOREAN INSTITUTE OF INFORMATION SCIENTISTS AND ENGINEERS

제 19 권 제 1 호
Vol. 19 No. 1

SOFTWARE ENGINEERING SOCIETY

소프트웨어공학회 KIPS

2017

제19회 한국 소프트웨어공학 학술대회 논문집

Proceedings of the 19th Korea Conference on Software Engineering (KCSE 2017)

- 일시: 2017년 2월 8일(수) ~ 2월 10일(금)
- 장소: 강원도 평창 한화리조트(휘닉스파크점)

주최: 한국정보과학회, 한국정보처리학회
주관: 한국정보과학회 소프트웨어공학 소사이어티
한국정보처리학회 소프트웨어공학 연구회
한국전자통신연구원

후원: ㈜솔루션링크, ㈜코스콤,
㈜비트컴퓨터, ㈜모아소프트,
소프트웨어공학엑스퍼트그룹㈜, T3Q㈜,
STA 테스팅컨설팅㈜, 슈어소프트테크㈜,
TTA 소프트웨어시험인증연구소
SW 상시모니터링기술연구단
무인자율 및 적응형 소프트웨어센터

	논문 발표 C			
	C1: 리펙토링 & 유지보수 좌장: 홍장의 교수(충북대) 장소: 그랜드홀 2	**C2: 요구공학** 좌장: 김순태 교수(전북대) 장소: 세미나실 1	**C3: 설계 & 모델링** 좌장: 이정원 교수(아주대) 장소: 세미나실 2	**C4: 패턴추출 & 빅데이터** 좌장: 이찬근 교수(중앙대) 장소: 세미나실 3
13:30-15:10 (100 분)	파일들의 패키지 관계를 반영한 Change Coupling 추출 및 이를 이용한 리팩토링 대상 추천 [학부논문] 조영준, 허민재, 이찬근 (중앙대학교) 효율적인 리팩토링을 위한 조직 특성의 품질 지표 기반 우선순위 선정 자동화 [단편논문] 박지훈, 손현승, 박보경, 이진협, 김영철 (홍익대학교) Hadoop Platform 에서 UML 모델 기반 영상 처리 소스 코드 자동 생성 방안 [단편논문] 고미은, 박용범 (단국대학교) 테스트로부터 소스 코드로의 추적성 향상을 위한 소프트웨어 행위 모델 활용 [일반논문] 백승석, 이병정 (서울시립대학교), 이정원 (아주대학교)	인공지능 소프트웨어 개발을 위한 요구공학 프레임워크: 인지컴퓨팅 시스템을 중심으로 [일반논문] 안우람 김재홍, 김경모(KAIST 소프트웨어대학원), 민상윤(KAIST) 안전 요구사항 추출을 위한 안전성 분석 지침 개발 및 적용 사례 [단편논문] 정대희, 권기현 (경기대학교) 소프트웨어 요구사항 진화에서의 추적성 관리의 어려움 조사 [일반논문] 박종열, 류승희, 정세린, 이선아 (경상대학교) 오디오 컨텍스트의 라이프 시맨틱 분석 프레임워크 [최우수논문] 정한터, 김수동 (숭실대학교), 라현정 ((주)스마트랩)	JAVA 오픈 소스의 재사용성 및 확장성 향상을 위한 소프트웨어 설계 패턴 추출 기법 [일반논문] 최용석, 김두환, 홍장의 (충북대학교) 디지털 건강 분석 솔루션을 위한 소프트웨어 플랫폼 설계 [일반논문] 라현정(스마티랩), 정한터, 임성민, 김수동(숭실대학교) 기능 분석과 아키텍처 패턴을 이용한 자율주행 자동차 소프트웨어 아키텍처 설계 방법 [일반논문] 김순겸, 김두환, Nazakat Ali, 홍장의 (충북대학교) 행위 모델링 및 조합을 위한 객체지향성 [학부논문] 김희언 (세종대학교)	Cascade K-Medoids 기반 자바 메소드 식별과 패턴 자동 식별 기법 [단편논문] 김태영, 김순태, 이정휴 (전북대학교) 딥러닝을 활용한 주행 패턴 특징 추출 연구 [단편논문] 이주영, 장기태 (한국과학기술원), 이홍석, 정희진 (한국과학기술정보연구원) 추상구문트리의 연어 관계 분석을 통한 Java API 패턴 추출 및 추천 시스템 개발 [최우수논문] 권찬우, 황상원, 남영광 (연세대학교) 빅데이터 분석 방법론 [후원업체] 유태빈 ((주)코스콤)
15:10-15:20	**휴식**			

효율적인 리팩토링을 위한 조직 특성의 품질 지표 기반 우선순위 선정 자동화

박지훈 ○ 손현승 박보경 이진협 김영철
홍익대학교 컴퓨터정보통신공학과 소프트웨어공학연구실
{pjh, son, park, ljh, bob}@selab.hongik.ac.kr

Tailoring an Automatic Priority based on quality metrics of an organization for Effective Refactoring

Jihoon Park○ Hyun Seong Son Bo Kyung Park Jin Hyub Lee R. Youngchul Kim
SE Lab, Hongik University

요 약

개발 완료된 소프트웨어도 필연적으로 추가적인 요구사항이나 변경할 것들이 생긴다. 이런 경우 유지보수에 투자되는 시간과 비용이 많이 차지한다. 하지만 국내 벤처/중소기업은 구현중심으로 유지보수에 대한 투자가 부족하다. 문제는 각 기업마다 조직 특성에 맞는 품질 지표나 그에 관련된 우선순위체계가 없는 경우가 많다. 이는 기업 조직에 알맞은 리팩토링 우선순위를 감별과 산출하는 기법을 제안한다. 본 논문에서는 리팩토링을 위한 Bad Smell기반 우선순위선정 프로그램 자동화를 구현하였다. 식별된 부분들을 리팩토링으로 효율적인 유지보수가 가능하다.

1. 서론

소프트웨어가 개발되어 사용자들이 사용하게 되면 필연적으로 추가적인 요구사항이나 수정해야 할 기능들이 생기기 마련이다. 이 중 유지보수에 들어가는 비용만 전체 개발비용의 절반 이상을 차지하고 있다[11]. 소프트웨어에 대한 연구 중 대부분이 소프트웨어 개발에 관련된 연구이다. 상대적으로 소프트웨어의 유지보수에 대한 연구는 많지 않은 실정이다[7,8]. 국내 중소기업의 경우 대부분 소프트웨어에 정해진 유지보수체계가 없는 상황이다. 대부분 유지보수 노력의 60%까지를 오류가 어디에 있는가를 식별하는데 사용한다[6].

본 논문에서는 이러한 문제들로 인하여 유지보수가 쉽지 않은 소프트웨어를 가시화한다. 가시화란 소스 코드로 존재하는 소프트웨어를 그래프 형식으로 변환하는 과정을 말한다[1,2,3,4]. 가시화된 소프트웨어에는 Bad Smell을 이용하여 리팩토링 우선순위를 산정할 수 있다. 각 기업에 따라서 더 우선시하는 부분에 중요도를 주어 Bad Smell기반 리팩토링 우선순위선정 자동화 도구를 구현하였다. 기업에서는 이 자동화된 도구를 이용하여 각 Bad Smell마다 레벨을 조정함으로써 리팩토링 우선순위를 산정할 수 있다.

2. 관련 연구

2.1 Refactoring

리팩토링은 기능은 그대로 유지한 채로 내부 구조를 개선하는 방법이다. 프로그램에서 버그가 발생할 확률을 최소화함과 동시에 코드의 가독성을 높일 수 있는 정형화된 방법이다. 본질적으로는 코드가 작성된 후 코드의 디자인을 개선하는 것이다[9].

사용자의 추가적인 요구사항들을 해결하다 보면 본래의 디자인과 달라질 것이다. 그렇기 때문에 잘 만들어진 디자인일지라도 유지보수를 하다 보면 망가질 가능성이 높다.

표 1 프로세스 수준 점수 산정 결과

영역	평가항목	대기업	중소기업
개발	고객 요구사항 관리	67.2	52.1
	분석	78.5	71.2
	설계	94	33.2
	구현	82.7	67.9
	테스트	100	89.9

표 1을 보면 중소기업의 경우는 소프트웨어프로세스(SP) 수준점수에서 설계 부분의 점수가 최하점수이다[10]. 그렇기 때문에 리팩토링이 필요하다. 리팩토링은 시스템을 구축하면서 디자인을 개선한다.

2.2 Bad Smell

Bad Smell은 마틴 파울러가 제시한 리팩토링이 필요한 순간들이다[9]. 표 2는 Bad Smell 중에서도 Method와 Class별로 잠재적 오류발생률이 높다고 판단된 Bad

† 이 논문은 2015년 교육부와 한국연구재단의 지역혁신창의인력양성사업의 지원을 받아 수행된 연구임(NRF-2015H1C1A1035548)

Smell항목이다.

표 2 잠재적 오류발생률이 높은 Bad Smell 항목

범위	항목	설명
Method	Long Parameter List	파라미터의 개수가 너무 많다.
	Message Chains	메서드의 호출연결고리가 너무 복잡하다.
	Feature Envy	다른 클래스의 속성을 너무 사용한다.
	Middle Man	클래스가 간단한 위임을 너무 많이 한다.
Class	Lazy Class	사용하지 않는 기능이 존재한다.
	Data Class	필드에 get, set 메서드만 존재한다.
	Large Class	클래스의 인스턴스 변수가 너무 많다.
	Refused Bequest	상속받은 메서드를 모두 사용하지 않는다.

3. 리팩토링 우선순위산정 가시화 프로그램

3.1 오픈 소스를 활용한 역공학

본 논문에서는 기존 연구의 결합도를 표시한 Class Diagram형식에서 Bad Smell 추출을 추가하였다[1,2,3,4].

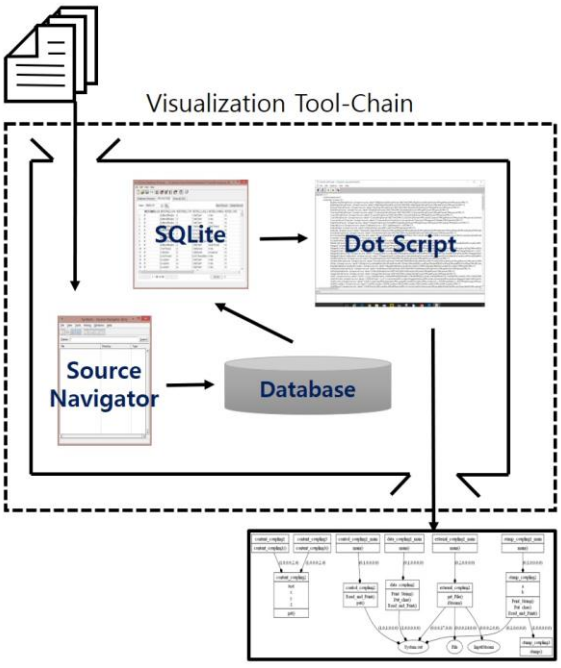

그림 1 Visualization Tool-Chain의 구성도

그림 1은 Tool-Chain의 구성도이다. 오픈 소스인 Source Navigator를 이용하여 JAVA 코드를 분석하면 생기는 여러 개의 SN파일 중 본 논문의 가시화 프로그램에 사용되는 파일은 총 6개이다. 추출된 파일의 데이터들은 재사용을 위해 새로 구성한 데이터베이스에 다시 저장한다. 데이터베이스에는 총 7개의 테이블이 있다. 6개의 SN파일들에 있던 정보들을 각각의 6개의 테이블마다 저장한다. 하나의 테이블을 더 생성하여 query문을 이용하여 추출한 결합도의 정보를 추가로 저장한다. 데이터베이스에 저장된 데이터들을 query문을 이용하여 리팩토링 우선순위가 높은 Bad Smell 항목을 찾아낸다. 각 Bad Smell은 사용자가 정한 기준에 따라 자유롭게 찾아낼 수 있다. 찾아낸 Bad Smell 항목들과 다른 정보들을 그래프로 그리기 위해 오픈 소스인 Graphviz를 이용한다. Graphviz는 Dot Script언어로 실행되기 때문에 정제된 데이터들을 이용하여 DotScript언어로 변환해주어야 한다. 본 논문에서는 이러한 과정들을 하나의 Tool-Chain으로 구축하여 자동으로 모든 과정이 진행되도록 하였다.

3.2 데이터베이스 구성

본 논문에서는 SNDB파일 들에서 추출해낸 데이터들을 재 정제하였다. 데이터베이스를 새로 만들어 저장하였고 다음은 그에 대한 설명이다.

그림 2는 클래스와 메서드의 정보가 저장된 데이터베이스이다. 4개의 테이블에 저장되어 있으며 각각 클래스, 인스턴스변수, 로컬변수, 메서드에 대한 데이터들이 저장되어 있다. SNDB_CL 테이블에는 클래스의 이름, 파일 경로, 클래스의 속성 등이 있다. SNDB_IV 테이블에는 인스턴스 변수가 소속된 클래스의 이름, 인스턴스변수의 이름, 파일 경로, 인스턴스변수의 속성 등이 있다. SNDB_LV 테이블에는 로컬변수의 이름, 파일 경로, 로컬변수의 속성 등이 있다. SNDB_MD 테이블에는 메서드가 소속된 클래스의 이름, 메서드의 이름, 파일 경로, 메서드의 속성, 메서드의 리턴 값, 메서드의 파라미터에 대한 정보 등이 있다.

Name	Object	Type
C:/Java/visualization/test/Test		
SNDB_CL	Table	
NAME	Field	TEXT
START_LINE_NO	Field	TEXT
PATH	Field	TEXT
END_LINE_NO	Field	TEXT
ATTRIBUTES	Field	TEXT
SNDB_IV	Table	
CLASS_NAME	Field	TEXT
VARIABLE_NAME	Field	TEXT
START_LINE_NO	Field	TEXT
PATH	Field	TEXT
END_LINE_NO	Field	TEXT
ATTRIBUTES	Field	TEXT
SNDB_LV	Table	
VARIABLE_NAME	Field	TEXT
START_LINE_NO	Field	TEXT
PATH	Field	TEXT
END_LINE_NO	Field	TEXT
ATTRIBUTES	Field	TEXT
SNDB_MD	Table	
CLASS_NAME	Field	TEXT
METHOD_NAME	Field	TEXT
START_LINE_NO	Field	TEXT
PATH	Field	TEXT
END_LINE_NO	Field	TEXT
ATTRIBUTES	Field	TEXT
RETURN_TYPE	Field	TEXT
ARGUMENT_TYPES	Field	TEXT
ARGUMENT_NAMES	Field	TEXT

그림 2 클래스와 메서드 정보 DB

그림 3은 각 모듈간의 호출, 상속, 결합도 정보다 저장된 데이터베이스이다. 3개의 테이블로 구성되어 있으며 각각 호출 정보, 상속 정보, 결합도 정보가 저장되어 있다. SNDB_BY 테이블에는 호출되는 클래스의 이름, 호출되는 메서드나 변수의 이름, 호출하는 클래스의 이름, 호출하는 메서드의 이름, 호출 형식, 파일 경로, 호출 간의 메서드 파라미터의 정보가 있다. SNDB_IN 테이블에는 서브클래스의 이름, 슈퍼클래스의 이름, 파일 경로, 클래스의 속성 등이 있다. SNDB_COUPLING은 앞의 6개의 테이블에 저장된 데이터들을 한번 더 정제한 테이블이다. 결합도 추출 프로그램에 의해서 SNDB파일과 다른 새 데이터들을 생산한다. 결합도에 관련된 모듈들에 대해서 호출하는 모듈, 호출당하는 모듈, 각 결합도에 대한 수치가 저장된다.

Name	Object	Type
SNDB_BY	Table	
REFERRED_CLASS_NAME	Field	TEXT
REFERRED_SYMBOL_NAME	Field	TEXT
REFERRED_TYPE	Field	TEXT
REFER_CLASS_NAME	Field	TEXT
REFER_SYMBOL_NAME	Field	TEXT
REFER_TYPE	Field	TEXT
ACCESS	Field	TEXT
LINE_NO	Field	TEXT
PATH	Field	TEXT
CALLER_ARGUMENT_TYPES	Field	TEXT
REFER_ARGUMENT_TYPES	Field	TEXT
SNDB_IN	Table	
SUB_CLASS_NAME	Field	TEXT
SUPER_CLASS_NAME	Field	TEXT
START_LINE_NO	Field	TEXT
PATH	Field	TEXT
END_LINE_NO	Field	TEXT
ATTRIBUTES	Field	TEXT
SNDB_COUPLING	Table	
CALLER	Field	TEXT
CALLEE	Field	TEXT
DATA	Field	TEXT
STAMP	Field	TEXT
CONTROL	Field	TEXT
EXTERNAL	Field	TEXT
COMMON	Field	TEXT
CONTENT	Field	TEXT

그림 3 각 모듈간의 호출, 상속, 결합도 정보 DB

3.3 우선순위 선정 프로그램

3.3.1 Long Parameter List

표 3 Long Parameter List 추출 설정 프로그램

```
Resultset lpl = statement.excutequery("
                    select ARGUMENT_TYPE from SNDB_MD");
while(lpl.next()) {
     String[] argument=lpl.getString("ARGUMENT_TYPE");
     if(argument.length > count) {
          return true;
     }
}
```

표 3에서 lpl은 SNDB_MD 테이블에서 ARGUMENT_TYPES값들을 가져온다. 가져온 값들은 파라미터의 정보이기 때문에 if문의 count값을 설정한만큼 Long Parameter List를 추출한다.

3.2.2 Middle Man

표 4 Middle Man 예시코드와 추출 프로그램

```
ResultSet mm = statement.excutequery("
                        select RETURN_TYPE from SNDB_MD where
                  RETURN_TYPE in (select REFERRED_CLASS_NAME
                                  from SNDB_BY where CLASS_NAME in
               (select [name] from SNDB_CL) and LINE_NO < 5)");
While(mm.next()){
     count ++;
}
return count;
```

표 4에서 mm은 SNDB_MD 테이블에서 5줄 미만의 클래스가 객체 형식으로 변수에 호출되는 RETURN_TYPE을 가져온다. 가져온 값들의 개수만큼 count를 추가한다. count를 리턴함으로써 count를 받는 곳에서 조건문을 설정한 만큼 Middle Man을 추출한다.

3.2.3 Large Class

표 5 Large Class 추출 설정 프로그램

```
Resultset lgc = statement.excutequery("
                    select count(*) as CNT from SNDB_IV where
          CLASS_NAME in (select [name] from SNDB_CL)");
while(lgc.next()){
     int cnt=Integer.parseInt("CNT");
     if(cnt > count) {
          return true;
     }
}
```

표 5에서 lgc는 SNDB_IV 테이블에서 인스턴스 변수의 개수를 가져온다. 인스턴스변수의 개수만큼 CNT가 저장된다. count 값을 설정한 만큼 Large Class를 추출한다.

3.2.4 Message Chains

표 6에서 mc는 SNDB_BY 테이블에서 호출되는 메서드의 이름을 가져온다. 이 과정을 한번의 값을 가져오는 과정이라고 볼 때 재귀함수로 설정하여 여러 번 수행한다. Message chain은 어떤 값을 가져오기 위해 여러 개여 객체를 거쳐야만 하는 경우이기 때문에 재귀호출을 통하여 거쳐갔던 객체의 개수를 count에 저장한다. count를 리턴함으로써 count를 받는 곳에서 조건문을 설정한 만큼 Message Chains를 추출한다.

표 6 Message Chains 추출 설정 프로그램

```
Public void re_getMC(String sN, String cN, int count){
    Resultset mc = statement.excutequery("
                    select REFERRED_SYMBOL_NAME,
                REFERRED_CLASS_NAME from SNDB_BY
          where REFERRED_SYMBOL_NAME LIKE 'get%'
     CLASS_NAME in (select [name] from SNDB_CL)");
    while(mc.next()){
        String rdsName
                =mc.getString("referred_symbol_name");
        String rdcName
                  =mc.getString("referred_class_name");
        count=re_getMC(rdsName, rdcName, count);
    }
    return count;}
```

3.2.5 Feature Envy

표 7에서 fe는 클래스를 호출하는 get-인 메서드를 체크한다. Cnt에 개수를 저장하고 count값을 설정함으로써 Feature Envy를 추출할 수 있다.

표 7 Feature Envy 추출 설정 프로그램

```
Resultset fe = statement.excutequery("
          select count(*) as CNT from SNDB_BY where
           REFERRED_SYMBOL_NAME LIKE 'get%' and
                                REFERRED_CLASS_NAME in
                          (select [name] from SNDB_CL)")
While(fe.next()) {
     int cnt=Integer.parseInt(fe.getString("CNT"));
     if(cnt > count) {
          return true;
     }
```

3.3 가시화된 Bad Smell

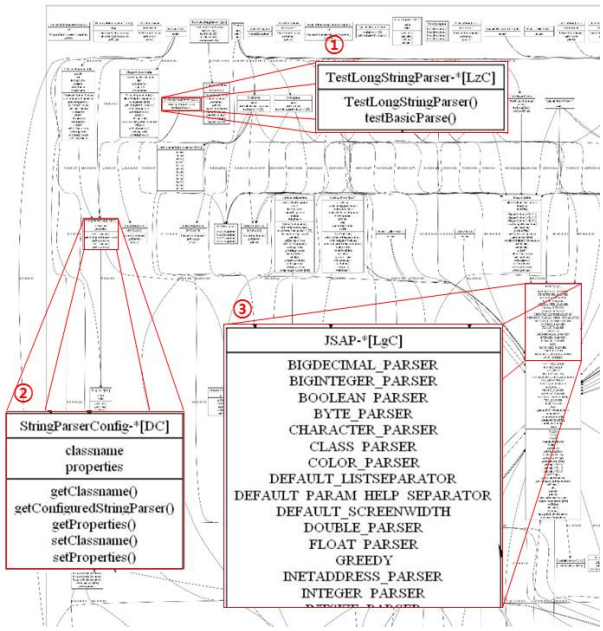

그림 4 추출한 BadSmell을 가시화한 그래프

그림 4는 오픈 소스인 Java Parser를 가시화한 그래프이다. ①번은 test가 아니면 일반적으로 사용되지 않는 클래스라는 것을 알 수 있다. ②번은 get-, set-메서드로만 이루어진 Data Class인 것을 알 수 있다. ③번은 인스턴스 변수가 일정 개수 이상이므로 중복된 코드가 존재할 가능성을 찾아볼 필요가 있다.

4. 결론

대부분의 중소기업들은 개발에만 투자를 집중한 나머지 조직의 품질 지표 기반의 유지보수 중요성에 둔감하다. 본 논문에서 제안한 가시화 시스템을 통해, 각 조직의 품질 지표 기반 맞춤형 리팩토링의 우선순위체계를 제안한다. 각 기업 조직의 맞는 지표 설정으로, 수작업으로 어려웠던 부분을 식별 가시화로 우선순위를 설정한다. 추가적으로 소스 코드를 해독 없이도 소스 코드를 쉽게 파악할 수 있다. 이를 통해 유지보수비용을 절감할 것으로 기대한다.

참 고 문 헌

[1] 박지훈, 권하은, 강건희, 이근상, 김영철, "코드 가시화를 통한 나쁜 코딩 습관 개선 방안 연구", 한국인터넷방송통신학회, 제 13권, 제 1호, pp.47-48, 2015

[2] 박지훈, 김영철, "나쁜 코딩 습관 개선을 위한 코드 가시화 연구", 한국정보처리학회, 제 23권, 제 2호, pp.497-500, 2016

[3] 박지훈, 장우성, 이진협, 손현승, 김영철, "확장된 나쁜 코딩 습관 식별 구현", 한국정보과학회, pp.401-403

[4] 강건희, 손현승, 김영수, 박용범, 김영철, "SW가시화 기반 리펙토링 기법 적용을 통한 정적 코드 복잡도 개선", 한국정보처리학회, 제 21권, 제 2호, pp.646-649, 2014

[5] 권하은, 박보경, 이근상, 박용범, 김영수, 김영철, "코드 가시화부터 모델링 추출을 통한 역공학 적용", 한국정보처리학회, 제 21권, 제 2호, pp.650-653, 2014

[6] 김지혁, 김창재, 류성열, "사례기반의 소프트웨어 유지보수 성숙도 모델 수립 방안", 정보과학회논문지, 제 36권, 제 9호, pp.718-731, 2009

[7] 조현훈, 황만수, 류성열, "효율적인 유지보수를 위한 C원시 코드 재구성에 관한 연구", 한국정보과학회 가을 학술발표논문집, 제 23권, 제 2호, pp.1573-1576, 1996

[8] 김창재, 박제원, "서비스기반의 소프트웨어 유지보수 성숙도 모델", Journal of KIIT, 제 12권, 제 5호, pp.173-184, 2014

[9] Martin Fowler, "Refactoring: Improving The Design of Existing Code", Addison-Wesley, 2002

[10] B.Boehm, Basil, "Software Defect Reduction Top 10 List IEEE Computer"

[11] NIPA, "SW Engineering 공학백서", SW 공학센터, pp.200, 2015