| Oral Session IX : Big Data, Smart Energy ICT, Smart Information

좌장 : 신춘성 (전남대)

역공학 기반의 Refactoring을 위한 설계 추출 방안

김현태[1], 진예진[2], 박찬솔[3], 문소영[4], 김영철[5]
홍익대학교 소프트웨어융합학과
e-mail : {00gusxo[1], dpwls6207[2], c2193102[3]}@g.hongik.ac.kr,
symoon[4]@selab.hongik.ac.kr, bob[5]@hongik.ac.kr

Design Extraction Approach for Refactoring Based on Reverse Engineering

Hyun Tae Kim, Jin Ye Jin, Chan Sol Park, So Young Moon,
and R. Young Chul Kim
SE Lab., Dept. of Software and Communications Engineering, Hongik University

요 약

오늘날 소프트웨어 규모는 시간이 지남에 따라 점차 커지고 있다. 따라서 설계에 중요성이 대두되고 Refactoring에 대한 중요성 또한 커지고 있다. 기존의 가시화 품질 관리 방법 중 내부 코드 가시화를 통해 품질 지표 수준 미달 부분의 식별이 가능하다. 따라서 본 논문에서는 역공학 기반의 코드 가시화로 품질 지표 미달의 모듈을 식별하기 위한 원천 설계를 추출하고자 한다. 이를 통해 설계 개선을 위한 리팩토링이 가능하다. 또한 소프트웨어의 가독성을 높이고 유지보수 비용이 절감하며 소프트웨어의 생산성과 유지보수성이 향상되므로 소프트웨어 품질의 향상을 기대한다.

키워드: 역공학, 리팩토링, 설계 추출, 소프트웨어 가시화, 객체지향 소프트웨어

1. 서 론

소프트웨어는 비가시성 특성 때문에 코드 가독성이 떨어지고, 거대한 소프트웨어에서는 코드 내의 관계가 매우 복잡하여 관리나 수정이 어려우며 품질이 떨어질 수 있다[1]. 이러한 소프트웨어의 품질을 향상시키기 위해서는 인증을 통한 향상 방법, 최신 소프트웨어 개발 방법론이나 프로세스를 통한 개발, 그리고 완벽한 테스트 프로세스와 테스팅을 통한 오류 제거 등의 방법이 있다. 하지만 기존 방법은 내부 코드의 가시화에 초점을 두지 않는다[2].

본 논문은 역공학 기반의 리팩토링을 위한 소프트웨어 아키텍처 가시화 기법 기반 설계 추출 방법을 제안한다. 소스 코드에서 클래스 간 정보를 추출하고, 추출된 정보를 DB에 저장하고 이를 리팩토링하여 소프트웨어의 가독성을 높이고 유지보수 비용이 절감, 소프트웨어 생산성 향상과 유지보수성 향상을 기대한다.

2. 관련 연구

2.1 소프트웨어 가시화

소프트웨어 가시화란 비가시성 특성을 가진 소프트웨어를 시각적으로 표현하는 기법이다[2]. 이를 통해 소프트웨어를 쉽게 이해할 수 있어 소프트웨어 개발 및 유지보수가 용이해 비용 절감이 가능하다. 소프트웨어 가시화는 소프트웨어 아키텍처 가시화, 런타임 행동 가시화, 소프트웨어 프로세스 가시화, 문서 자동화 등으로 분류된다[3]. 본 논문에서는 아키텍처 가시화를 진행할 예정이다.

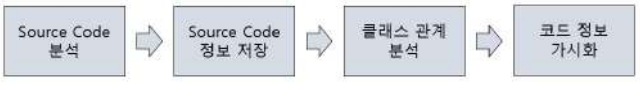
2.2 소프트웨어 아키텍처 가시화

소프트웨어 아키텍처 가시화는 역공학 기법을 기반으로 소스코드로부터 소프트웨어의 구조를 역설계한다. 소스 코드로부터 설계 모델의 정보를 추출하고, 그래프를 통해 가시화하는 기술을 포함한다[4]. 본 논문은 Java Parser를 이용하여 코드의 정보를 추출하고, Class Diagram을 자동 생성하여 소프트웨어의 구조를 파악함으로써 품질 관리가 가능한 방법을 제안한다.

3. 역공학 기반의 Refactoring을 위한 설계 추출 방법

3.1. 역공학 기반의 설계 추출 프로세스

역공학 기반의 설계 추출 프로세스는 그림1과 같다.



(그림1) 역공학 기반의 설계 추출 프로세스

3.1.1. 소스 코드 분석

코드 분석 단계에서 JavaParser를 이용하여 Java 코드를 분석한다. Java 코드를 AST 구조를 바탕으로 분석하여 필요한 요소들로 분리한다.

3.1.2. Database에 소스코드 정보 저장

분리된 정보들을 데이터베이스에 저장하는 단계이다. Package, Class, Method, Field 등으로 테이블을 나누어 data를 알맞은 테이블에 저장시킨다.

3.1.3. 클래스 간 관계 분석

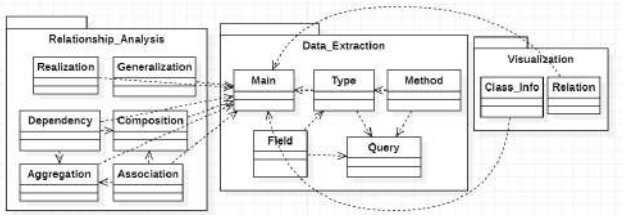
database에 저장된 data를 이용하여 class의 관계를 분석하는 단계이다. 관계는 상속, 실체화, 구현, 의존, 연관 등의 관계로 구분한다. 이러한 관계 구분에 필요한 정보들을 데이터베이스에서 추출하여 해당 관계가 있는지 없는지 여부를 판별하여 관계에 대한 정보를 relation 테이블에 저장시킨다. 메소드 호출 관계와 호출 횟수 또한 분석을 하여 function_call 테이블에 저장한다.

3.1.4. 코드 정보 가시화

데이터베이스에서 Class, Method, Field, Relation, 함수 호출 횟수 등의 필요한 정보들을 추출하여 그러한 정보들을 바탕으로 가시화에 필요한 dot 언어[5]를 추출한다. 이렇게 추출한 dot 언어를 이용하여 Class Diagram을 추출한다.

3.2. 역공학 기반의 설계 추출기 설계

역공학 기반 설계 추출기의 구조는 그림 2와 같다. 추출기의 구조는 3개의 패키지로 구성된다. Data_Extraction 패키지는 클래스 간 관계 분석을 위한 정보를 추출하고 데이터베이스에 저장하는 패키지다. Relationship_Analysis 패키지는 추출된 정보로 클래스 간 관계를 분석하고 그 분석 결과를 데이터베이스에 저장시키는 패키지다. Visualization 패키지는 추출된 클래스 간 관계, 클래스의 정보를 바탕으로 코드 정보를 가시화하는 패키지이다.



(그림2) 역공학 기반의 설계도 추출기 구조

4. 실험 결과

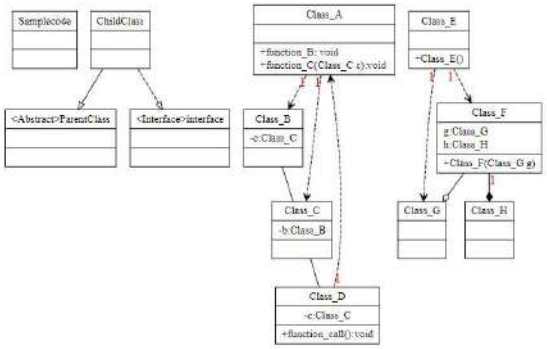
표1은 테스트를 위한 코드의 일부이다. 그림 3는 표1의 코드를 입력하여 역공학 기반의 설계도 추출기를 통해 자동 생성된 클래스 다이어그램이다. 기존 클래스 다이어그램의 기능을 포함하고, 클래스 간의 호출 횟수를 출력한다.

(표 1) 테스트 코드의 일부

```
class Class_E {
        public Class_E() {
                Class_G g = new Class_G();
                Class_F f = new Class_F(g);
        }
}

class Class_F {
        Class_G g;
        Class_H h;

        public Class_F(Class_G g) {
                this.g = g;
                this.h = new Class_H();
        }
}
```



(그림3) 테스트 코드의 가시화 결과

5. 향후 연구

본 논문은 역공학을 기반으로 Refactoring을 위한 코드 가시화를 통해 설계 추출 방법을 제안하였다. 추출기를 통해 소프트웨어의 비가시성 특성을 극복하고 대규모 소프트웨어에서도 손쉽게 코드를 기반으로 설계를 추출할 수 있다. 또한 함수의 호출 횟수를 클래스 다이어그램에 표시를 함으로써 결합도를 측정하는 것에 기반이 될 수 있다. 하지만 본 연구에서는 품질 지표를 설정하지 않아 소프트웨어의 품질을 판단할 수 없다.

향후 연구로써, 품질 지표를 통해 품질 지표 미달의 리팩토링이 필요한 모듈의 설계를 추출하고 가시화를 하고자 한다. 이를 통하여 유지보수 비용 감소 및 품질 향상의 효과를 기대하고자 한다.

ACKNOWLEDGMENT

이 논문은 교육부 및 한국연구재단의 4단계 두뇌한국21 사업의 지원(F21YY8102068)과 2022년도 정부(교육부)의 재원으로 한국연구재단의 지원(No. 2021R1I1A305040711, No. 2021R1I1A1A01044060)을 받아 수행된 연구임.

참 고 문 헌

[1] So Young Moon, R. YoungChul Kim, “Code Structure Visualization with A Tool-Chain Method”, International Journal of Applied Engineering Research, Vol.10 No.99, 2015

[2] 박보경, 권하은, 손현승, 김영수, 이상은, 김영철. “소프트웨어 가시화를 통한 품질 개선 사례 연구”, 『한국정보과학회 정보과학회논문지』 제41권 제11호, 2014. pp.935-942

[3] 권하은, 박보경, 이근상, 박용범, 김영수, 김영철. “"코드 가시화부터 모델링 추출을 통한 역공학 적용”, 『한국정보처리학회 학술대회 논문집』 제21권 제2호, 2014. pp.646-649

[4] 강건희, 손현승, 김영수, 박용범, 김영철. “SW 가시화 기반 리팩토링 기법 적용을 통한 정적 코드 복잡도 개선”, 『한국정보처리학회 추계학술발표대회 논문집』 제21권 제2호, 2014. pp.650-653

[5] Graphviz, http://graphviz.org/doc/info/lang.html, 2022