소스코드 재사용성 향상을 위한 점검 및 분석 사례 연구

장진욱

농협대학교

jjw@nonghyup.ac.kr

Case study on Inspection and Analysis to Improve Source code Reusability

Jin-Wook Jang

Agricultural Cooperative Univ.

요 약

본 연구는 커스터마이징 프로젝트를 수행하는 과정에서 소프트웨어의 크기와 복잡도가 증가하고 있다. 그에 따른 개발 시간과 비용 절감을 위한 사례 연구이다. 절감을 위한 방법으로 기 개발된 시스템을 체계적으로 분석 평가할 수 있는 소프트웨어 구조 분석 도구인 Stan4J와 Sonarqube 이 두 가지 도구를 사용하였다. 위 툴을 사용하여 개발된 N사 교육지원 소프트웨어에 대한 진단을 실행하였다. 설계 및 구현에 있어서 개선해야 할 점과 관련 지표를 수립하였다. 그리고 해당 도구들을 사용하여 향후 지속적으로 관리해야 할 지표를 선정하고 가이드라인을 제시하였다. 이후 개발조직은 재사용성 향상을 통한 품질향상에 대한 필요성을 공감하고 지속해 개선 활동을 할 수 있다.

Ⅰ. 서 론

소프트웨어의 재사용에 관한 관심이 높다. 특히 하나의 솔루션을 기반으로 다양한 프로젝트에 커스터마이즈(customize)하는 프로젝트의 경우 더욱 중요하다. 한번 개발된 소프트웨어는 사용자의 요구와 필요에 따라 동일한 소스 코드를 커스터마이즈하여 재사용되며 이런 과정에서 다양한 형태의 품질 이슈가 발견되고 있다.

재사용은 레거시시스템(legacy system)에서 개발되어 그 품질을 인정받은 모듈을 재차 개발하지 않고 상당한 부분을 다시 사용함으로써 생산성을 증대시킴과 동시에 소프트웨어의 품질을 향상시키고 신뢰성을 높이는 방법이다. 또한 개발에 필요한 시간과 비용을 절감시킬 수 있는 중요한 방법이다. 본 연구는 N사 교육지원 소프트웨어의 코드 재사용 품질향상을 위해 객체지향 설계 및 프로그래밍에 대한 설계원칙과 설계패턴, 리팩토링 방법을 적용하였다. [1, 3, 4] 그리고 시스템에 대한 다양한 척도(metric)와 체계적으로 분석 평가할 수 있는 Stan4J, Sonarqube인 정적분석 도구를 적용하였다.

본 연구의 구성은으로 2장에서는 본 연구에서 사용한 소프트웨어 구조 분석 도구인 Stan4J에서 제공하는 메트릭을 소개한다. 또한 Sonarqube에서 제공하는 인지 복잡성에 관해 설명한다. 3장에서는 Stan4J와 Sonarqube로 분석한 교육 지원소프트웨어 분석 결과를 제시한다. 4장에서는 제안하는 척도 관리 방안에 관해 설명하고 5장에서 본 연구내용을 요약하고 지표를 제시한다.

Ⅱ. 소프트웨어 품질 메트릭

2.1 CK(chidamber & kemerer) 메트릭

CK 메트릭(metric)은 클래스 및 인터페이스에 대해 정의되며 각각의 응용 프로그램, 라이브러리 또는 패키지에 대한 해당 평균값 메트릭이 있다. [5] 대표적인 재사용성 측정 metric이며, 객체지향 소프트웨어에서 복잡도, 재사용성, 모듈화, 캡슐화를 측정하는 여러 요소로 구성된다. 클래스 WMC(weighted methods per class)는 클래스의 모든 메소드에 대한 순환 복잡도 메트릭의 합계이며 해당 메소드(method)에는 클래스 초기화뿐 아니라 명시적 및 암시적 생성자가 포함되지만 중첩 클래스의 메소드는 포함되지 않는다. 클래스별 가중 계산식은 수시 1.와 같다.

$$WMC\ (c) = \sum_{m \in M_{\text{local}}(c)} VG\,(m)$$

수식 1. 클래스별 가중 계산

2.2 순환 복잡도 매트릭(cyclomatic complexity metrics)

순환 복잡도(cyclomatic complexity)는 메소드의 CC는 제어 흐름 그래프에서 디지인 포인트의 수 + 1로 계산된다. 디자인 포인트는 제어 흐름 중 선형이 아닌 if, for, while 문, case, catch 절 및 유사한 소스 코드 요소에서 발생한다. [2, 5]

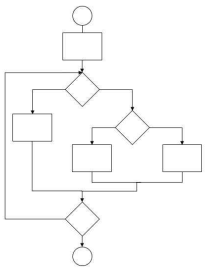

그림 1. 제어흐름 그래프

위 예제에서 나타난 제어 흐름 그래프에 디자인 포인트가 3개 있으므로 CC는 4이다. CC의 값은 해당 메소드나 함수에서 버그가 포함되어 있을 확률과 매우 관련성이 높다. 표 1.는 알려진 CC 값의 범위와 위험도이다.

표 1. McCabe's CC 값의 범위와 위험도

V(G)	Risk
1 - 10	easy program, low risk
11 - 20	complex program, tolerable risk
21 - 50	complex program, high risk
>50	impossible to test, extremely high risk

Ⅲ. 분 석

본 장에서는 Stan4J와 Sonarqube 도구 등을 활용하여 교육지원 소프트웨어의 산출물을 통해 분석한 결과를 제시한다.

다음 그림 2.은 Stan4J를 통해 분석된 교육지원 소프트웨어의 Pollution 요약 결과이다. 가장 많은 오염도를 차지하는 두 문제는 Tangled와 ELOC(estimated lines of code)로 보인다. 그리고 거리도 다소 문제점으로 지적 되고 있다. 다음은 앞서 소개한 Stan4J에서 제공하는 매트릭 들에 대한 요약이다. 이중 Tangled가 14.58%로 다소 문제가 있는 것으로 표시되고 있다.

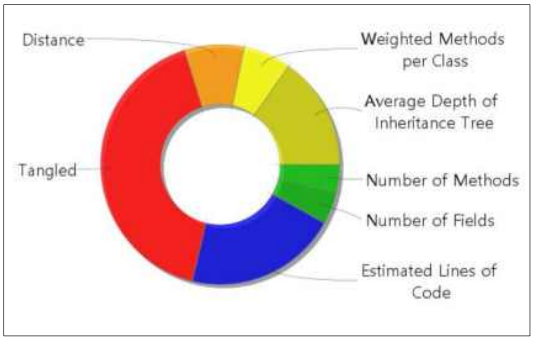

그림 2. Stan4J 분석결과 Pollution 요약

Ⅳ. 코드개선 방안

앞서 제시한 바와 같이 Stan4J와 Sonarqube를 이용하여 교육 지원소프트웨어의 코드 및 설계 품질을 점검한 결과 다양한 메트릭(metric)을 통해 개선될 수 있는 포인트를 찾을 수 있다. [6]

표 2. 측정지표

도구	지표
Stan4J	순환복잡도(CC : Cyclomatic Complexity)
	예상 코드 라인(ELOC : Estimated Lines of Code)
Sonarqube	인지 복잡성(Cognitive Complexity)
	중복코드(Duplicate Code)

표 2.는 4개의 지표는 많은 소프트웨어 개발 업체에서 공통으로 또는 우선적으로 관리하는 것으로 측정하기도 매우 쉽고 또한 해석하기도 쉬운 특성이 있다. 그리고 불필요한 코드의 중복이나 과도한 복잡도를 갖는 코드들이 눈에 띄는 현 상황에서 가장 효과적인 지표로 보이며 이들을 개선한 후 다른 지표들에 대한 확대를 고려하는 것이 적절하다고 판단된다.

특히 Sonarqube의 경우 지속적통합(continuous integration)을 통해 리파지토리에 커밋(commit)되는 동시에 분석 작업이 동시에 이루어질 수 있다. 이를 통해 코드 품질을 모니터링하는 관리자는 중점 지표들에 대한 리포트를 통해 효과적으로 관리할 수 있다. [7]

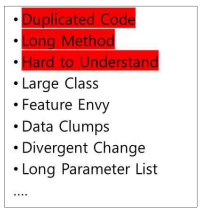

그림 3. 코드악취

심각한 위반에 해당하는 수정사항에 대해서는 해당 개발자에게 자동적으로 관련 메트릭과 코드를 알려주어 지속적인 코드 품질 향상에 큰 도움을 줄 것으로 기대한다. 코드 악취 진단과 리펙토링 프로세스는 그림 4.와 같다.

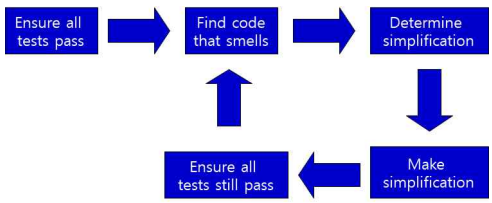

그림 4. 코드 악취 진단과 리팩토링 프로세스

Ⅴ. 결 론

향후 N사의 지속적인 코드재사용 품질 향상을 위해, 재사용성 측정의 대표적 지표인 응집도와 결합도를 포함하는 CK메트릭을 비롯하여, 산업체에서 가장 광범위하게 쓰이는 CC 그리고 다양한 복잡도 관련 메트릭을 소개했다. 또한 패키지 단위의 재사용과 배포를 위해 관리해야할 Tangled를 다뤘으며 RC Martin의 패키지 수준의 메트릭을 설명하였다. 그리고 Sonarqube에서 제공하는 Cognitive Complexity과 CC와의 관계를 다뤘다. Stan4J와 Sonarqube를 사용하여 N사 교육지원 소프트웨어에 대한 분석을 수행하였으며 설계 및 구현에 있어서 개선해야 할 점과 관련 지표를 제시하였다.

ACKNOWLEDGMENT

본 연구는 2024년 대한민국 교육부와 한국연구재단의 인문사회분야 신진연구자지원사업의 지원을 받아 수행된 연구임(NRF-2022S1A5A8049255)

참 고 문 헌

[1] Chang-Ju Moon, Design and Implementation of an Object-Oriented Reverse Engineering Tool for Legacy Fortran Code Reuse. 2012. 6.

[2] Yang-sun Moon, The Applicability Analysis of the Object-Oriented Design and Programming Guidelines, 1999. 11.

[3] Jaejin Park, Analysis of Energy Efficiency for Code Refactoring Techniques, 2014.

[4] Xiao Liu, Using SonarQube to Evaluate the Code Quality of C/C++ Programming Assignments, 2019. 6.

[5] Eun Young Byun, Constructing an Open Source Based Software System for Reusable Module Extraction, 2017. 9.

[6] Hyukcheol Jeong, An Early Reliability Prediction Model using Design Complexity Metrics. 1996.

[7] 박양환, 최진영, 기계학습을 이용한 소스코드 정적 분석 개선에 관한 연구, 정보보호학회논문지, 2020.