ERP 근태 관리 시스템의 코드 스멜 분석

최원재*, 김태환*, 문현준O, 김정선*
*한양대학교 컴퓨터공학과,
O한양대학교 컴퓨터공학과
e-mail: mhj5730@hanyang.ac.krO, {znzndnjswo2, kimagic, kimjs}@hanyang.ac.kr*

Code Smell Analysis of ERP Time Attendance Management System

Wonjae Choi*, Taehwan Kim*, Hyunjun MoonO, Jungsun Kim*
*Dept. of Computer Science and Engineering, Hanyang University,
ODept. of Computer Science and Engineering, Hanyang University

● 요 약 ●

배포된 소프트웨어는 고객의 새로운 요구사항을 반영하면서 기능을 추가하거나 수정한다. 수정된 소프트웨어는 문제없이 동작하지만, 내부의 구조는 점점 더 복잡해지면서 유지보수가 어려워지고, 수정 시 버그 발생 확률이 높아지게 된다.

이에 따라 잘못된 코드 구조를 개선하고, 소프트웨어의 품질을 향상시켜야한다. 본 논문에서는 개발된 ERP 근태관리 시스템이 지속적인 요구사항을 반영하면서 발생한 잘못된 코드 구조인 코드 스멜을 탐지하고 분석하는 연구를 진행하였다.

키워드: 코드 스멜(code smell), 리팩토링(refactoring), 소프트웨어구조(software structure)

I. Introduction

소프트웨어는 시간이 흐르면서 고객의 새로운 요구사항을 만족하기 위해 기능을 추가하거나 기존의 기능을 수정한다. 수정이 이루어진 뒤 프로그램은 겉으로는 문제없이 완벽하게 동작하지만, 내부의 구조는 서서히 변형되고 왜곡되어간다. 이러한 과정이 누적되어 코드가 복잡해지면서 작성자도 알아보기 힘들만큼 유지보수가 어려워질 뿐만 아니라 프로그램 수정 시 버그 발생 확률이 높아지게 된다.[1] 따라서, 소프트웨어의 구조는 지속적인 개선을 통해 요구사항을 반영하면서 생기는 품질 저하를 막아야한다. 이렇게 시간이 흐르면서 잘못된 소프트웨어의 구조를 개선하는 방법이 리팩토링이다. 리팩토링은 마틴 파울러와 켄트 벡에 의해 소개된 개념으로 겉으로 드러나는 소프트웨어의 기능을 바꾸지 않으면서 소프트웨어의 내부를 개선하는 과정이다.[1]

리팩토링을 통해 소프트웨어를 개선하기 위해서는 현재 소프트웨어의 문제점이 무엇인지 파악해야한다.[2] 마틴 파울러와 켄트 벡은 리팩토링이 필요한 잘못된 코드 구조들을 코드 스멜[1]로 분류하고, 코드 스멜을 제거하기 위한 리팩토링 방법들을 제시하였다. 따라서 리팩토링에 앞서 소프트웨어의 품질을 저하시키는 코드 스멜을 탐지하고 분석하는 것이 중요하다

본 논문은 최근 개발한 ERP 근태관리프로그램이 새로운 요구사항을 반영하면서 발생된 리팩토링이 필요한 코드 구조인 코드 스멜을 분석하였다.

II. The Proposed Scheme

개발된 ERP 근태관리프로그램에 발생된 문제점은 세가지로 1) 잘못된 데이터베이스 설계 2) 개발자의 인식 부족과 개발자들 간 개발 소스 코드 공유가 안 되어 발생된 중복 코드 3) 읽기/쓰기 클래스의 비 모듈화이다. 이 문제점들로 인해 발생된 코드 스멜은 다음과 같다.

2.1 수정의 산발

새로운 요구사항인 연장 신청과 일반 및 경조 휴가 신청에 대해 마감기한의 압박으로 대응되는 데이터베이스 테이블을 생성하지 않였다. 그 결과, 기존 근태신청 테이블에 해당 신청들을 저장하면서 테이블의 값을 객체 모델에 매핑하는 Repository클래스가 근태 신청 이외의 요구사항으로 수정되어하는 수정의 산발 코드 스멜이 발생되었다.

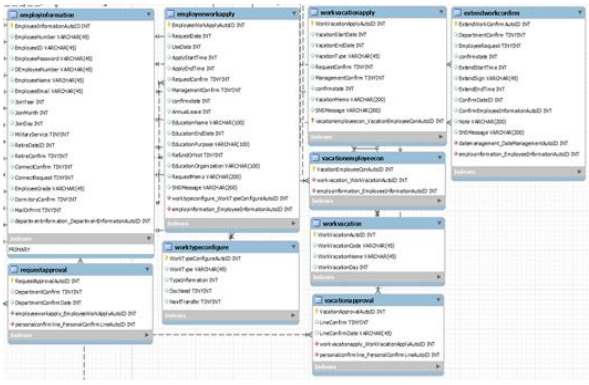

Fig. 1. 데이터베이스 스키마

2.2 중복 코드

개발자의 인식 부족과 개발자들 간 개발 소스 코드 공유가 안되어 시스템 내 불 필요한 중복 코드의 코드 스멜이 발생되었다. 또한 2.1절에서의 일반 휴가와 경조 휴가에 대한 정보가 데이터베이스 테이블이 아닌 코드 상의 조건문으로 존재하여, 휴가 정보를 조회하는 코드에서 조건문이 반복적으로 나타나게 되었다.

2.3 방대한 클래스

개발 단계에서 테이블에 매핑되는 읽기/쓰기 클래스를 생성하였지만, 새로운 요구사항을 반영하면서 기존 클래스에 테이블에 존재하지 않는 불 필요한 필드를 지속적으로 추가하였다. 또한 특정 테이블A를 외래키로 갖는 테이블에 매핑되는 클래스들이 A테이블 내의 특정 필드 정보가 필요하여 중복 필드를 갖게되거나 A테이블에 매핑되는 거대한 클래스를 필드로 갖게되어 방대한 클래스의 코드 스멜이 발생되었다.

III. Conclusions

본 논문에서는 개발된 ERP 근태 관리 시스템이 지속적으로 새로운 요구사항을 반영하면서 코드 내 발생한 코드 스멜을 분석하였다. 코드 스멜이 존재하는 소프트웨어는 겉으로는 문제 없이 동작하지만 유지보수성과 미래적응성을 저하시키게되므로 리팩토링을 통해 이를 해결하는 것이 중요하다.

추후 연구에서는 코드 스멜을 제거하기 위한 리팩토링과 실시된 리팩토링을 통한 구조 개선점 및 성능 향상을 분석할 예정이다.

REFERENCES

[1] M.Fowler. Refactoring: improving the design of existing code. Addison Wesley, 1999

[2] Joshua Kerievsky, “Refactoring to Patterns”, Addison Wesley, 2005