論文 / 著書情報
Article / Book Information

題目(和文)	
Title(English)	Supporting Reactive and Proactive Source Code Refactoring: A Context-Based Approach
著者(和文)	Sae-LimNatthawute
Author(English)	Natthawute Sae-Lim
出典(和文)	学位:博士(学術), 学位授与機関:東京工業大学, 報告番号:甲第11328号, 授与年月日:2019年9月20日, 学位の種別:課程博士, 審査員:佐伯 元司,権藤 克彦,渡部 卓雄,小林 隆志,林 晋平
Citation(English)	Degree:Doctor (Academic), Conferring organization: Tokyo Institute of Technology, Report number:甲第11328号, Conferred date:2019/9/20, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Category(English)	Doctoral Thesis
種別(和文)	論文要旨
Type(English)	Summary

（博士課程）
Doctoral Program

論 文 要 旨

THESIS SUMMARY

系・コース： Department of, Graduate major in	情報工学 系 情報工学 コース	申請学位（専攻分野）： Academic Degree Requested	博士 （ 学術 ） Doctor of
学生氏名： Student's Name	Sae-Lim Natthawute	指導教員（主）： Academic Supervisor(main)	佐伯元司
		指導教員（副）： Academic Supervisor(sub)	

要旨（英文 800 語程度）
Thesis Summary（approx.800 English Words ）

Software quality is a major concern in software development due to the rapidly growing demand of software system. Code smell is often used as an indicator of a design flaw or problem in source code. Code fragments containing code smell (i.e., smelly code) are handled by applying refactoring technique to improve the quality. Refactoring is a technique to restructure source code without changing its external behavior to improve source code quality. It can be divided to reactive and proactive refactoring based on timing and purpose of the operation. Reactive refactoring is the refactoring process that is applied after the source code become smelly to remove the code smells whereas proactive refactoring is the refactoring process that is applied before the source code become smelly to prevent code smell.

In issue-driven software development projects, development teams tend to adopt an issue tracking system to manage their tasks such as bug fixing or feature implementation. In this situation, we can regard modules that developers are going to modify to complete the issues as their context. Many studies have shown that developers are more likely to refactor source code that are related to their context, e.g., to prepare source code for implementation. However, most of the tools that recommend location for refactoring do not consider such a context. Therefore, developers have to perform the time-consuming process of identifying relevant outputs.

In this dissertation, we propose an approach to support reactive and proactive refactoring recommendation by considering developers' context. This dissertation can be divided into three parts: (1) a study on factors that developers use to filter and prioritize code smells; (2) an approach to support reactive refactoring; and (3) an approach to support proactive refactoring.

(1) A study on factors that developers use to filter and prioritize code smells.

In order to improve the results of existing code smell detection tools, we conduct a study on how professional developers filter and prioritize code smells. A study is conducted with ten professional developers under the condition that they complete a list of tasks. The results show that developers are more likely to refactor code smells that are related to their context. Therefore, the result of this part lays a foundation of this dissertation in a sense that developers' context should be considered when recommending modules for refactoring. The results from this study may also facilitate further research into code smell detection, prioritization, and filtration to better focus on the actual needs of developers.

(2) An approach to support reactive refactoring.

In this dissertation, we propose a technique to prioritize code smells using developers' context to improve the result of existing code smell detection tools. We use impact analysis techniques to predict modules that developers are likely to modify which can be regard as their context. Then, we prioritize code smells based on such information. In addition, we also study the factors that affect the performance and the optimized parameters of the technique. The proposed technique is evaluated using open source software projects and a controlled experiment with professional developers. The results show that our technique can provide more relevant results that are aligned with how developers actually prioritize their code smells.

(3) An approach to support proactive refactoring.

In order to identify the target for proactive refactoring, we propose a concept of decaying modules which are non-smelly modules but are getting closer to become code smells. Proactively refactoring such modules can prevent source code from becoming smelly. In this dissertation, we present an investigation on the characteristics of decaying modules. Furthermore, we propose an approach to use a machine learning technique to predict decaying modules which can be used to support developers during refactoring strategy planning. The prediction approach uses source code quality metrics as predictor variables and whether a module will get decayed in the next release as a response variable. We also study the use of developers' context to improve the performance of the prediction model. The developers' context can be estimated by applying impact analysis technique to change descriptions and source code.

備考 ： 論文要旨は、和文 2000 字と英文 300 語を 1 部ずつ提出するか、もしくは英文 800 語を 1 部提出してください。
Note : Thesis Summary should be submitted in either a copy of 2000 Japanese Characters and 300 Words (English) or 1copy of 800 Words (English).
注意：論文要旨は、東工大リサーチリポジトリ(T2R2)にてインターネット公表されますので、公表可能な範囲の内容で作成してください。
Attention: Thesis Summary will be published on Tokyo Tech Research Repository Website (T2R2).