様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

令和　元 年　 ６ 月 ２６ 日現在

機関番号：１２６０８
研究種目：若手研究(B)
研究期間：2015～2018
課題番号：１５Ｋ１５９７０

研究課題名（和文）ソフトウェア変更計画に基づいたプレファクタリング支援手法の確立

研究課題名（英文）Supporting Prefactoring Activities Based on Software Change Plans

研究代表者
林　晋平（Hayashi, Shinpei）

東京工業大学・情報理工学院・准教授

研究者番号：４０５４１９７５

交付決定額（研究期間全体）：（直接経費）　　3,000,000円

研究成果の概要（和文）：本研究では，リリースを伴いながら開発が進行していく現実のソフトウェアに対して，将来行う変更を容易にするようなリファクタリングの適用箇所を特定すること，リファクタリングが意図通りに行われたかの確認方法を開発者に提供することにより，開発のコンテキストを考慮したリファクタリングの実行を支援することを目的としている．導出された不吉な臭いを，イシュー管理システム上のバックログ情報をコンテキストとして用いて優先順位付けする手法を開発した．また，開発者が不吉な臭いの選択や優先順位付けに用いる因子を特定した．さらに，リファクタリング適用の影響を確認しやすい形で変更を表示するための分析環境を構築した．

研究成果の学術的意義や社会的意義
本研究で提案する手法は，変更に関する自然言語の説明文や議論内容をリファクタリング支援のためのコンテキストとして用いるもので，このようなコンテキストの自動推定による方法はこれまでに存在しなかった．また，この考え方は低品質化が進んでいる初期段階のモジュールの特定にも利用可能であり，リファクタリングの適用箇所として推薦可能なモジュールを拡張可能である．この手法は，GitHub等で行われるオープンソース開発のみならず，バグ票に基づく一般的なソフトウェア開発に広く適用可能であるため，適用可能性が高く，多様なソフトウェア開発の品質を向上させることが期待できる．

研究成果の概要（英文）：In this research, we aim to support developers' refactoring activities with taking their development context into consideration by (1) identifying the location where to apply refactoring operations to make it easy to apply future changes and (2) providing a way to confirm applied refactorings composed in a specific set of changes. We have developed a technique for prioritizing code smells derived from an existing smell detector using the information in an issue tracking system as developers’ task context. We have also identified the factors that developers use for selecting and prioritizing code smells. Furthermore, we have implemented change analysis environments that enable us to confirm the effects of applied refactorings.

研究分野：ソフトウェア工学

キーワード：リファクタリング　プレファクタリング　不吉な臭い　機能捜索　ソフトウェア進化　イシュー管理システム

様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９、ＣＫ－１９（共通）

１．研究開始当初の背景

リファクタリングとは，既存プログラムの外的振る舞いを変更せずにその内部構造を変換することで，該当プログラムコードの理解容易性や変更容易性を安全に向上させるプロセスを指す．継続的なソフトウェア開発においては，ソフトウェア品質は次第に劣化していくため，リファクタリングの適用等により品質を維持し続けることは重要である．リファクタリング技術は様々な実用的な統合開発環境により提供されており，すでに産業界に浸透している．また，リファクタリングが求められるソースコードの低品質な箇所（臭い）を特定する研究も盛んである．こういった手法によりリファクタリングの機会を開発者に気付かせることは，ソフトウェアの品質維持のために有効と考えられている．

一方，現実的にはリファクタリングが十分に活用されているとは言い難い．特に，コード修正の際には必ずテストの実行が必要と定めている開発組織は多く，こういった組織では，たとえリファクタリングツールが外的振る舞いの保存を保証していたとしてもテストの工数がかかってしまい，リファクタリングの実行が躊躇される．この場合，通常の開発と切り離して投資的にリファクタリングを実行するといった従来のプロセスは現実的ではなく，テストの工数を増やさずに品質維持が行えるよう，バグ修正等でプログラムを変更する際に合わせてリファクタリングを行うことが効果的である．その手段のひとつに，具体的な変更に先立ってリファクタリングを行うプレファクタリングがある．

２．研究の目的

これまでのリファクタリング支援手法はこのようなプレファクタリングへの利用には不十分であった．本研究では，以下の2点の解決を目指した．

1．従来の臭い検出器のコンテキストの未考慮．従来の臭い検出器は，プログラム全体を解析し，得られた臭いや適用すべきリファクタリングの候補を順位付きで開発者に提示する．これは，具体的な開発を意識せず投資的にソースコード品質を改善する場合には適しているが，直近で行うべき変更がすでに計画されている場合には，それに直接貢献しないようなリファクタリング候補を数多く含んでおり，非効率で適さない．

2．プレファクタリングとして適用したリファクタリングと後段の変更との混在．プレファクタリング実施の際には，リファクタリングのためのソフトウェア変更とバグ修正等の後段の変更が時間的に近接したり，互いに前後したりする．このことは変更結果をコミットとして版管理リポジトリに登録する際の労力につながる．また，この混在は，版管理リポジトリ中のコミットから過去に行われたプレファクタリングの形跡を抽出し分析する際の困難にもつながる．

３．研究の方法

本研究の支援対象として，現在開発中のソースコードに関連付いた形で，該当ソフトウェアに対して今後どのような変更を行う予定かが Bugzilla や Jira などのイシュー管理システム上で議論されているような開発プロジェクトを想定する．こういった開発形態は現在では一般的となっており，GitHub などのソフトウェア開発ホスティングサイト上で多数のプロジェクトが従っている．また，GitHub 等では，機能の実装途中の作業履歴が Work in Progress の形でブランチやプルリクエストとして開発者で共有されており，その内容に対して議論が行える．こういった情報をコンテキストに用いることにより，より適切で有用なリファクタリングの計画を提示することができると考える．これを用いて，実装前段階のイシューの情報を用いて，実際に変更を行う前に予め行うリファクタリングであるプレファクタリングの候補を導出し，ガイドとして開発者に提示する．

４．研究成果

主に，以下の４つの研究成果を達成した．

(1) 不吉な臭いの選択・優先順位付けの因子に関する調査

開発者が実際に検出された不吉な臭いのうち有用なものを選択したり，優先順位付けを行ったりする際の因子としてどういったものが考えられるかを，オープンソースソフトウェア JabRef から検出した不吉な臭いを対象に調査した．調査では，検出された臭いに対して，真に対処すべき臭いか，その優先順位はどれだけか，またそれらの理由は何かをソフトウェア開発者に問い合わせた．その結果，フィルタリングと優先順位付けの因子として，タスク関連性（Task Relevance）が最上位として選ばれたことがわかった．このことは，本研究が推奨するような，将来解決すべきタスク情報を用いて臭いを優先順位付けする手法の優位性を示唆している．

(2) イシュー情報を用いた不吉な臭いのプレファクタリング向けの優先順位付け

イシュー管理システム上の未解決イシューの情報を用いて，リファクタリングを行うべき箇所の候補として利用される不吉な臭いの検出結果の優先順位付けを行う手法を開発した．提案手法では，未解決イシューを開発者が将来すべき変更のコンテキストを表現したものであるとみなし，イシュー中の自然言語記述とソースコードとをテキスト検索に基づく既存のインパクト解析技術を用いて対応付けることによって，将来変更されうるソースコード中のモジュールを確度つきで特定する．最後に，既存の不吉な臭いの検出器を利用して得た臭いのリストを，特定したモジュールの確度に基づき並べ替えて出力する．

提案手法により得られたコンテキスト指向の臭いのリストの特徴を分析した．また，従来の危険度に基づくリストよりもコンテキスト指向の臭いのリストのほうが，開発者の判断に近いランキングが生成できていることを確認した．さらに，提案した不吉な臭いの優先度指標が，不吉な臭いとして検出されるよりも前段階のモジュールの悪化予測にも活用可能であることを確認した．

(3) リファクタリングを含む変更の可視化

リファクタリングの適用が混在した変更を整理して表示したり，部分的な変更を分離するためのインタフェースを提供したりすることにより，プログラム変更の読解や解きほぐしを支援するための変更分析環境を試作した．試作した環境では，多種多数のリファクタリングの適用を伴った変更に対して，行ったリファクタリング操作を階層的に管理することにより，最終的に得られた変更のより細かい粒度への分割を支援する．また，変更をその種類や特徴に基づき複数の観点に分類した半束構造を導入し，構造の要素間の関連性を開発者にフィードバックすることにより，変更の解きほぐしを支援するための環境を構築した．

(4) 多様なリファクタリングコンテキストの分析

その他，名前変更リファクタリングや責務割り当てのリファクタリング，アーキテクチャ適合リファクタリング，ゴール指向要求モデルリファクタリング，変更履歴リファクタリングなど，様々な種類のソフトウェア成果物に対するリファクタリングに対して，その支援手法の開発を通じて，リファクタリング適用の際に考慮すべきコンテキストを検討した．

５．主な発表論文等

〔雑誌論文〕（計 24 件）

1. 高橋 碧，セーリム ナッタウット，林 晋平，佐伯 元司："情報検索に基づく Bug Localization への不吉な臭いの利用".情報処理学会論文誌，査読有，vol. 60，no. 4，pp. 1040-1050，2019.
2. Natthawute Sae-Lim, Shinpei Hayashi, Motoshi Saeki: "Context-Based Approach to Prioritize Code Smells for Prefactoring".Journal of Software: Evolution and Process, 査読有, vol. 30, no. 6:e1886, pp. 1-24, 2018. DOI: 10.1002/smr.1886
3. Natthawute Sae-Lim, Shinpei Hayashi, Motoshi Saeki: "An Investigative Study on How Developers Filter and Prioritize Code Smells".IEICE Transactions on Information and Systems, 査読有, vol. E101-D, no. 7, pp. 1733-1742, 2018. DOI: 10.1587/transinf.2017KBP0006
4. Shinpei Hayashi, Fumiki Minami, Motoshi Saeki:"Detecting Architectural Violations Using Responsibility and Dependency Constraints of Components".IEICE Transactions on Information and Systems, 査読有, vol. E101-D, no. 7, pp. 1780-1789, 2018. DOI: 10.1587/transinf.2017KBP0023
5. Mohamed Wiem Mkaouer, Marouane Kessentini, Mel Ó Cinnéide, Shinpei Hayashi, Kalyanmoy Deb: "A Robust Multi-Objective Approach to Balance Severity and Importance of Refactoring Opportunities". Empirical Software Engineering, 査読有, vol. 22, no. 2, pp. 894-927, 2017. DOI: 10.1007/s10664-016-9426-8
6. 林 晋平，柳田 拓人，佐伯 元司，三村 秀典："クラス責務割当てのファジィ制約充足問題としての定式化".情報処理学会論文誌，vol. 58，no. 4，pp. 795-806，2017.
7. 風戸 広史，林 晋平，大島 剛志，小林 隆志，夏川 勝行，星野 隆，佐伯 元司："多層システムに対する横断的な機能捜索". 情報処理学会論文誌，vol. 58，no. 4，pp. 885-897，2017.
8. Katsuhisa Maruyama, Takayuki Omori, Shinpei Hayashi: "Slicing Fine-Grained Code Change History". IEICE Transactions on Information and Systems, 査読有, vol. E99-D, no. 3, pp. 671-687, 2016. DOI: 10.1587/transinf.2015EDP7282
9. 加藤 哲平，林 晋平，佐伯 元司："呼び出し関係グラフ分割手法の動的機能捜索手法との組合せの検討".電子情報通信学会論文誌，査読有，vol. J98-D，no. 11，pp. 1374-1376，2015.

他 15 件

〔学会発表〕（計 36 件）

1. Natthawute Sae-Lim, Shinpei Hayashi, Motoshi Saeki: "Toward Proactive Refactoring: An Exploratory Study on Decaying Modules". 3rd International Workshop on Refactoring (IWoR 2019), 2019.
2. Ryosuke Funaki, Shinpei Hayashi, Motoshi Saeki: "The Impact of Systematic Edits in History Slicing". 16th International Conference on Mining Software Repositories (MSR 2019), 2019.
3. Aoi Takahashi, Natthawute Sae-Lim, Shinpei Hayashi, Motoshi Saeki: "A Preliminary Study on Using Code Smells to Improve Bug Localization". 26th IEEE/ACM International Conference on Program Comprehension (ICPC 2018), 2018.
4. Sarocha Sothornprapakorn, Shinpei Hayashi, Motoshi Saeki: "Visualizing a Tangled Change for Supporting Its Decomposition and Commit Construction". 42nd IEEE Computer Software and Applications Conference (COMPSAC 2018), 2018.
5. Katsuhisa Maruyama, Shinpei Hayashi, Takayuki Omori: "ChangeMacroRecorder: Recording Fine-Grained Textual Changes of Source Code". 25th IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER 2018), 2018.
6. Natthawute Sae-Lim, Shinpei Hayashi, Motoshi Saeki: "How Do Developers Select and Prioritize Code Smells? A Preliminary Study". 33rd IEEE International Conference

on Software Maintenance and Evolution (ICSME 2017), 2017.

7. Maaki Nakano, Kunihiro Noda, Shinpei Hayashi, Takashi Kobayashi: "Mediating Turf Battles! Prioritizing Shared Modules in Locating Multiple Features". 41st IEEE Computer Society Signature Conference on Computers, Software and Applications (COMPSAC 2017), 2017.
8. Shinpei Hayashi, Fumiki Minami, Motoshi Saeki: "Inference-Based Detection of Architectural Violations in MVC2". 12th International Conference on Software Technologies (ICSOFT 2017), 2017.
9. Natthawute Sae-Lim, Shinpei Hayashi, Motoshi Saeki: "Revisiting Context-Based Code Smells Prioritization: On Supporting Referred Context". 9th International Workshop on Managing Technical Debt (MTD 2017), 2017.
10. Natthawute Sae-Lim, Shinpei Hayashi, Motoshi Saeki: "Context-Based Code Smells Prioritization for Prefactoring". 24th International Conference on Program Comprehension (ICPC 2016), 2016.
11. Shinpei Hayashi, Hiroshi Kazato, Takashi Kobayashi, Tsuyoshi Oshima, Katsuyuki Natsukawa, Takashi Hoshino, Motoshi Saeki: "Guiding Identification of Missing Scenarios for Dynamic Feature Location". 23rd Asia-Pacific Software Engineering Conference (APSEC 2016), 2016.
12. Jumpei Matsuda, Shinpei Hayashi, Motoshi Saeki: "Hierarchical Categorization of Edit Operations for Separately Committing Large Refactoring Results". 14th International Workshop on Principles of Software Evolution (IWPSE 2015), 2015.

他 24 件

〔図書〕（計 0 件）

〔産業財産権〕
○出願状況（計 0 件）

○取得状況（計 0 件）

〔その他〕
ホームページ等
http://www.se.cs.titech.ac.jp/~hayashi/

６．研究組織

(1)研究分担者
なし

(2)研究協力者
研究協力者氏名：佐伯 元司
ローマ字氏名：Motoshi Saeki

研究協力者氏名：丸山 勝久
ローマ字氏名：Katsuhisa Maruyama

研究協力者氏名：セーリム ナッタウット
ローマ字氏名：Natthawute Sae-Lim