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

平成　３０　年　　６　月　２１　日現在

機関番号：　１２６０８
研究種目：　基盤研究(C)（一般）
研究期間：　2014～2017
課題番号：　２６３３００７８

研究課題名（和文）開発者背景情報を活用するコード推薦システム

研究課題名（英文）Code recommendation system with developer's background information

研究代表者
増原　英彦（MASUHARA, Hidehiko）

東京工業大学・情報理工学院・教授

研究者番号：４０２８０９３７

交付決定額（研究期間全体）：（直接経費）　　3,600,000円

研究成果の概要（和文）：開発者と開発環境の相互作用に焦点を当てて研究進め、(1)開発者の編集操作履歴から開発者の意図を推定し、コード推薦システムの推薦精度を向上させる手法の提案、(2)プログラムの大域的な構造を変更するようなリファクタリングアルゴズムの提案と実証、(3)細粒度の編集操作履歴と、開発者が認識する作業単位の関連性を発見する手法の提案、(4)開発者の意図に反してプログラムに漏れ(omission)がある場合の巻き戻し型デバガの提案、(5)ライブプログラミング環境を実用的にする目的で、テスト実行機能との統合およびデータ構造可視化機能の統合を行い、また開発者の行動変化ついての利用者実験を行った。

研究成果の概要（英文）：We focused on interaction between developers and development environment, and achieved the following research results. (1) We proposed a method to improve code recommendation by estimating the developer's intention from editing history. (2) We proposed and implemented a refactoring algorithm that modifies global program structures. (3) We proposed a method that correlates fine-grained editing history and developer's tasks. (4) We proposed a novel debugger that finds relevant code when a program omits some of the programmer's intentions. (5) We proposed novel features to live programming environments that integrate unit-testing features, and also data structure visualization. We also carried out a user study to observe developer's behavior with a live programming environment.

研究分野：　ソフトウェア

キーワード：　開発者の意図　開発環境　コード推薦システム　ライブプログラミング　リファクタリング　巻き戻しデバガ　編集操作履歴

1. 研究開始当初の背景

今日のソフトウェア開発では、巨大な API (application programming interface, つまりクラス・メソッド・関数定義の集合) を持った多数のライブラリやフレームワークを利用することが一般的である。そのため Eclipse や Visual Studio のような統合ソフトウェア開発環境が提供するコード補完機能 (編集中プログラムの入力位置で利用可能なクラス名、メソッド名、あるいは典型的な一連の命令列を提示したり自動的に入力する機能) に対する期待が高まっている。

しかしながら標準的なコード補完機能は、編集プログラム中のごく限られた情報 (具体的には入力位置の型情報と入力中の識別子名等) しか利用せず、提示方法も単純 (具体的には入力可能な識別子名の一覧を表示する等) なため、一覧することが難しいほど大量の候補を提示してしまったり、間接的・複雑なメソッド呼び出し列のような使用法は提示できないといった弱点がある。

これに対して近年はコード推薦システムの研究が注目されている。コード推薦システムは、大量のオープンソースプログラムから抽出した統計情報を利用して補完候補を利用頻度順に提示したり、何個かのメソッドを呼び出す一連の操作まで補完することで、開発者が知りたい、またはこれから入力したい内容により近い推薦を行うことを目指している。このようなアプローチを集中して議論する SUITE (International Workshop on Search-driven development: Users, Infrastructure, Tools and Evaluation), RSSE (International Workshop on Recommendation Systems for Software Engineering) 等の国際ワークショップもすでに何回か開催されて多数の参加者を集めている。

研究代表者も、大量のオープンソースプログラムと高速なキーワード検索エンジンを活用した **コード推薦システム Selene** の提案・作成を行ってきた。コード推薦システムは開発者が編集中のプログラムからキーワードを抽出し、それと類似したプログラムを大規模なソースコードリポジトリから検索する。さらに、類似プログラム中のコード断片を編集中プログラムの入力位置付近のコード片に対する類似度で順位付けして、開発者に提示する。類似したコード片は編集中プログラムに書かれていないメソッド呼び出しなどを含むため、開発者が知りたい・次に入力したい内容を推薦することができる。

Selene は約 200 万ファイルのオープンソースプログラムリポジトリと高速な検索エンジン GETA を利用することにより、リアルタイムにコード推薦を行うシステムとしては**最大の規模を達成している。**

しかし従来のコード推薦システムはオープンソースプログラムからの検索や統計情報を利用することはしているものの、開発者や編集プログラムの背景情報、つまり開発者が持っている知識や編集中のコードの目的といった性質をほとんど利用していない。オンライン書類販売サービスなどが利用者の背景情報を積極的に推薦に利用していることを考えると、コード推薦システムにおいても同様のアプローチが有効だと考えられる。実際、ソフトウェアデバッグ環境の研究においては、プログラムの構造やデータフローを解析したり、コメント文やバグ報告を自然言語処理によって分析するといった利用者の背景情報を活用するアプローチがあり [Holmes and Murphy, 2005]、優れた結果を出している。

2. 研究の目的

このような背景の下、本研究はコード推薦システムに**開発者の背景情報**を積極的に取り入れ、より**高精度で利便性の高い推薦システム**を作るための基礎技術の開発を目標とする。具体的には以下の点に関して従来のコード推薦システムを拡張する研究を行う。

開発者知識の推定: 開発者が使用方法を熟知しているクラスやメソッドよりも、よく知らないものに関する情報を提示する方が有用性が高いと予見できる。そのために開発者の知識を編集中のプログラムや開発時の行動から推定する手法を提案する。

「あてはめやすさ」測度の導入: 推薦コード片は、編集中プログラムに「あてはめやすい」ものほど関連性が高く再利用コストも小さいため、有用性が高いと予見できる。従来の単語列や型情報による類似性測度にかえて、「あてはめやすさ」に基づいた測度を導入し、推薦コードの順位付けを改善する手法を提案する。

開発者注目点の利用: プログラムは一般に多数の関心事が混在している一方、開発者がある瞬間に注意を払っているのはその一部にすぎない。そこでプログラムの中でも開発者がコード推薦の時点で注目している部分とそれに関係している部分のみを抽出することで、推薦

精度の向上と提示情報の可読性向上をさせる手法を提案する。

これらの提案を Selene を拡張する形で実現し、ソフトウェア開発に利用する実験を通して有効性を検証する。

3. 研究の方法

研究開始当初の計画は、4ヶ年度にわたって (1) 開発者知識の推定の提案、(2)「あてはめやすさ」測度の提案、(3) 開発者注目点の利用法の提案の 3 点について、研究代表者および連携研究者が並行して進めてゆくものであった。このうち (3) の開発者注目点の検討を進めたところ、ソフトウェア開発環境と開発者の間の相互作用全体についての課題を整理する重要性を見い出し、開発環境の機能のうち特に、リファクタリング機能、バージョン管理機能、デバグ機能、ライブプログラミング機能についても研究を行うこととした。

研究全体を研究代表者および青谷 (連携研究者) が担当し、(1) と (2) については村上 (研究協力者) と共同で Kersten らが提案した開発環境の操作履歴から抽出する**関心度** [Kersten and Murphy, 2005] を応用することと、コード再利用アルゴリズム [Cottrell et al., 2008] から編集距離を定義することで行った。(3) およびそから発展させた点については、リファクタリング機能を櫻井 (連携研究者) および Khatchadourian(研究協力者, City University of New York) と共同で、バージョン管理機能を Hirschfeld(研究協力者, Hasso-Plattner Institute, University of Potsdam) と共同で、デバグ機能を櫻井 (連携研究者) と共同で、ライブプログラミング機能を今井および岡 (研究協力者) と共同で実施した。

4. 研究成果

開発者知識を用いたコード推薦手法　コード推薦システムに開発者知識を推定する手法を取り入れることを試みた。具体的には Keresten の関心度モデルに基づいて編集操作履歴から開発者の興味を推定することで推薦精度の向上を目指した。Eclipse 統合開発環境および Selene コード推薦システムの上に、関心度追跡アルゴリズムと推薦重み付けアルゴリズムを追加し、実験的に取得したソフトウェア開発時の操作履歴の記録から、重み付けパラメータの最適化を行った。その上で利用者実験を行い、開発者が少数のモジュールを集中して編集している場合に推薦の質の向上を得た。この結果は International Conference on Software Engineering (ICSE) に併設される査読付き国際ワークショップ RSSE に採録・発表された (雑誌論文 9)。

リファクタリング機能　ソフトウェア開発環境における有用な機能の 1 つにリファクタリング機能がある。リファクタリング機能は、プログラムの振舞いを変えずにプログラム構造を整理する機能であり、開発者の意図をソフトウェアの構造に反映させるための重要な手段となっている。本研究中では、新しい言語機能を開発者が受け入れる過程に注目し、(1)Java 言語の新機能であるデフォルトメソッドを利用したリファクタリング機能の提案と実現、(2) 開発者が言語の新機能を受け入れる際の注目的に関する実施調査を行った。

前者については、整理後のプログラムを型安全性に基づく解析によって検証するアプローチでプログラム全体に渡る安全なリファクタリングを保証する方式を提案、実現し、オープンソフトウェアへの適用によってその有効性を検証した。その成果はソフトウェア工学分野のトップ開発の 1 つである International Conference on Software Engineering (ICSE) に採択されている (雑誌論文 1,3,6)(学会発表 2,12)。

後者については、オープンソフトウェアプロジェクトに対してリファクタリングを行ったプログラムを修正案として提案することにより、開発者が新言語機能についてのどのように受け止めているかを評価する調査手法を提案した。従来、新言語機能に対する受容過程の調査は、受容後の結果を観察することで行われていたため、時間がかかることや否定的な反応を得にくかった。提案手法では、調査者から修正案を送ることによって、これまで得られなかった点についての意見が得られる画期的なものと言える。この成果は論文誌論文として採択されている (雑誌論文 1)。

バージョン管理機能　開発者知識を推定するために用いる編集操作履歴は、開発者自身が変更の取り消しや、追加した機能を概観する目的でも用いられる。開発者はしばしば複数の作業を並行して進めるため、開発環境に対する操作履歴のような、細粒度の編集履歴に対しては、履歴上の各操作がどの作業に対応するかを推定する必要がある。そのための手法として、編集履歴と編集対象間の依存関係を解析する手法を提案した (雑誌論文 5)。

アスペクト指向プログラミングでは、プログラムに対して横断的に作用するアスペクトがあるため、プログラムの変更が予期せぬ影響を生むことがある。そのために、プログラム変更前後のバージョンに対してポイントカットの作用対象集合の変化を監視し、開発者が意図しない影響が生じる可能性を発見する解析方法を提案した。さらにEclipse開発環境のプラグインとして実際に解析プラグインを作成し、検出力に関する評価を行った。この研究はソフトウェア工学分野のトップ会議の1つである Automated Software Engineering に採択された他、それを発展させた論文誌論文として発表している (雑誌論文 2,7), (学会発表 8)。

デバグ機能　実行履歴に基づくデバッガの改良を行った。実行履歴デバッガはプログラムの実行内容を記録し、それを利用者が精査することで効率的に誤りを発見させるようなシステムである。本研究では、そのようなデバッガにプログラム静的解析結果を利用する方法を提案･発表した。特に「実行されなかったことによる誤り (Omission Bug) の発見」に関して、問題提起とその解決手法を示した査読付論文が ACM/SIGAPP Symposium On Applied Computing に採択され発表を行った (雑誌論文 8)。

ライブプログラミング環境　ライブプログラミングは開発者の編集行動に対してプログラムの実行結果を即座に与えるような開発環境であり、試行錯誤や誤りの早期発見に適していることで注目されている。本研究課題で行っているような開発者の意図を推定によって、フィードバックをより適切なものにすることが可能になると考えられる。この発想の下、本研究ではライブプログラミングの基本機能についての改良を行った。

ライブプログラミング環境においてユニットテスト技法を簡便に用いる手法を提案・実現した。これまでのライブプログラミングが教育などの小規模なソフトウェア記述のみを対象としていたのに対し、本格的なソフトウェア開発にも利用できる新たな可能性を示すことができた。この成果は International Conference on Systems, Programming, Languages and Applications: Software for Humanity (SPLASH) 国際会議においてポスターおよびデモとしてそれぞれ採択・発表されている (学会発表 7,9,10,11,13)。

ライブプログラミング環境の効果のモデル化に関しては、これまで即時のフィールドバックがどのようにプログラミング体験を変化させるかが不明であったところ、我々は実験的なライブプログラミング環境を使用した経験を基に、特に誤りを発見するタイミングに影響を与えている可能性に注目してモデル化を行った。さらに予備的な実験を数名の被験者を対象に行い、モデルが不足する点や他の要因についての検討を行った。この結果は、日本ソフトウェア科学会プログラミングとプログラミング言語ワークショップにおいてポスター発表された (学会発表 5)。

リスト構造や木構造を代表とするデータ構造とそれに対する処理を記述する際に、ライブプログラミング方式でデータを自動的グラフィクス表示するような開発環境を設計し、予備的な処理系を作成した。データ構造に関する処理は、参照関係が複雑になるためにコード動きを理解するのが困難なため、ライブプログラミングによる効果が高いであろうという予想に基づいている。予備段階ではあるが、構想と処理系作成の際の問題解決を中心とした内容で、情報処理学会プログラミング研究会での発表を行った。また国際ワークショップ Programming Experience 2017 (PX/17) に査読付論文として採択された (雑誌論文 4),(学会発表 4,6)。

本研究を遂行する中で、ライブプログラミングをはじめとして、開発者のプログラミング行為を研究対象とする気運が高まってきている。プログラミング言語、ソフトウェア工学の研究コミュニティだけでなく、human-computer interaction 分野、教育、認知科学、芸術などの分野で研究・開発が行われている。研究代表者も産業総合技術研究所の加藤淳博士や、研究協力者の Hirschfeld 教授と協力して、そのような広がりのある多くの研究どうしで刺激を与えあう分野横断的なコミュニユティを作る活動を開始している (学会発表 1,3)。

引用文献

[Cottrell et al., 2008] Cottrell, R., Walker, R. J., and Denzinger, J. (2008). Semi-automating small-scale source code reuse via structural correspondence. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, SIGSOFT '08/FSE-16, pages 214–225, New York, NY, USA. ACM.

[Holmes and Murphy, 2005] Holmes, R. and Murphy, G. C. (2005). Using structural context to recommend source code examples. In

Proceedings of the 27th International Conference on Software Engineering (ICSE 2005), pages 117–125.

[Kersten and Murphy, 2005] Kersten, M. and Murphy, G. C. (2005). Mylar: a degree-of-interest model for ides. In *Proceedings of the 4th international conference on Aspect-oriented software development*, AOSD '05, pages 159–168, New York, NY, USA. ACM.

5. 主な発表論文等

（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕　(計 9 件)

1. Khatchadourian and Masuhara. Proactive empirical assessment of new language feature adoption via automated refactoring: The case of Java 8 default methods. *The Art, Science, and Engineering of Programming*, Vol. 2, No. 1, March 2018. Article no.6. DOI: 10.22152/programming-journal.org/2018/2/6 査読有

2. Khatchadourian, Rashid, Masuhara, and Watanabe. Detecting broken pointcuts using structural commonality and degree of interest. *Science of Computer Programming*, Vol. 150, pp. 56–74, December 2017. DOI: 10.1016/j.scico.2017.06.011 査読有

3. Khatchadourian and Masuhara. Automated refactoring of legacy Java software to default methods. In *Proceedings of International Conference on Software Engineering (ICSE'17)*, pp. 82–93, May 2017. DOI: 10.1109/ICSE.2017.16 査読有

4. Oka, Masuhara, Imai, and Aotani. Live data structure programming. In *Proceedings of the 2nd Edition of the Programming Experience Workshop (PX/17)*, pp. 26:1–26:7. April 2017. DOI: 10.1145/3079368.3079400 査読有

5. Taeumel, Platz, Steinert, Hirschfeld, and Masuhara. Unravel programming sessions with THRESHER: Identifying coherent and complete sets of fine-granular source code changes. *Computer Software*, Vol. 34, No. 1, pp. 103–118, February 2017. DOI: 10.11309/jssst.34.1_103 査読有

6. Khatchadourian, Moore, and Masuhara. Towards improving interface modularity in legacy Java software through automated refactoring. In *Proceedings of the Workshop on Language Modularity À La Mode (LaMOD'16)*, pp. 104–106. March 2016. DOI: 10.1145/2892664.2892677 査読有

7. Khatchadourian, Rashid, Masuhara, and Watanabe. Detecting broken pointcuts using structural commonality and degree of interest. In*Proceedings of 30th IEEE/ACM International Conference on Automated Software Engineering (ASE 2015)*, pp. 641–646, November 2015. DOI: 10.1109/ASE.2015.80 査読有

8. Sakurai and Masuhara. The omission finder for debugging what-should-have-happened bugs in object-oriented programs. In *In Proceedings of The 30th ACM/SIGAPP Symposium On Applied Computing (SAC 2015)*, pp. 1962–1969. April 2015. DOI: 10.1145/2695664.2695735 査読有

9. Murakami, Masuhara, and Aotani. Code recommendation based on a degree-of-interest model. In *Proeceedings of the Fourth International Workshop on Recommendation Systems in Software Engineering (RSSE 2014)*, pp. 28–29. June 2014. DOI: 10.1145/2593822.2593828 査読有

〔学会発表〕　(計 13 件)

1. 加藤, 増原. プログラミング・エクスペリエンスの新潮流 -言語設計から産業応用まで: 編集にあたって. 情報処理学会会誌, Vol. 58, No. 11, pp. 1006–1009, November 2017.

2. Khatchadourian and Masuhara. Defaultification refactoring: A tool for automatically converting Java methods to default. *International Conference on Automated Software Engineering (ASE)*, pp. 984–989, November 2017. University of Illinois at Urbana-Champaign, デモ発表.

3. 増原. 実用的なライブプログラミングに向けて. 電気関係学会北陸支部連合大会, September 2017. 富山大学, 招待講演.

4. 岡, 増原, 今井, 青谷. Kanon ライブプログラミング環境を用いたデータ構造のプログラミング. 第 19 回プログラミングおよびプログラミング言語ワークショップ (PPL2017), March 2017. 山梨県笛吹市, ポスター発表.

5. 今井, 増原, 青谷. ライブプログラミング環境によるプログラマの行動と生産性への影響に関する実証研究. 第 19 回プログラミングおよびプログラミング言語ワークショップ (PPL2017), March 2017. 山梨県笛吹市, ポスター発表.

6. 岡, 増原, 青谷. ライブプログラミングのためのデータ構造の可視化と対話機能. 情報処理学会第 113 回プログラミング研究会発表, pp. 2016–5–(8), March 2017. 東京大学.

7. 今井, 増原, 青谷. ライブプログラミング環境におけるユニットテスト機能の設計と実現方法. 情報処理学会第 111 回プログラミング研究会発表, pp. 2016–3–(7), October 2016. 日本アイ・ビー・エム株式会社本社事業所.

8. Khatchadourian, Rashid, Masuhara, and Watanabe. Fraglight: Shedding light on broken pointcuts in evolving aspect-oriented software. *International Conference on Systems, Programming, Languages and Applications: Software for Humanity (SPLASH)*, pp. 17–18. October 2015. Pittsburg, PA, USA, デモ発表.

9. Imai, Masuhara, and Aotani. Shiranui: A live programming with support for unit testing. *International Conference on Systems, Programming, Languages and Applications: Software for Humanity (SPLASH)*, pp. 36–37. October 2015. Pittsburg, PA, USA, ポスター発表.

10. Imai, Masuhara, and Aotani. Making live programming practical by bridging the gap between trial-and-error development and unit testing. In *International Conference on Systems, Programming, Languages and Applications: Software for Humanity (SPLASH)*, pp. 11–12. October 2015. Pittsburg, PA, USA, デモ発表.

11. 今井, 増原, 青谷. ライブプログラミングにユニットテストを統合する機能の提案. 第 17 回プログラミングおよびプログラミング言語ワークショップ (PPL2015), March 2015. 愛媛県松山市, ポスター発表.

12. Sakurai and Masuhara. Crossver: a code transformation language for crosscutting changes. In *Proceedings of the 9th International Workshop on Advanced Modularization Techniques (AOAsia/Pacific 2014)*, November 2014. Hong Kong, China.

13. 今井, 増原, 青谷. Shiranui: テストフレンドリーなライブプログラミング言語環境, 日本ソフトウェア科学会大会, September 2014. 名古屋大学, ポスター発表.

〔図書〕 (計 0 件)

〔産業財産権〕 (計 0 件)

〔その他〕 (計 1 件)
ホームページ等:
`http://prg.is.titech.ac.jp/`

6. 研究組織

(1) 研究代表者
増原英彦 (MASUHARA, Hidehiko)
東京工業大学情報理工学院・教授
研究者番号: 40280937

(2) 連携研究者
青谷知幸 (AOTANI, Tomoyuki)
東京工業大学情報理工学院・助教
研究者番号: 20582919

櫻井孝平 (SAKURAI, Kouhei)
金沢大学電子情報学系・助教
研究者番号: 80597021

(3) 研究協力者

ロベルト・フィルシュフェルド
(HIRSCHFELD, Robert)

今井朝貴 (IMAI, Tomoki)

ラフィ・カチャドリアン
(KHATCHADOURIAN, Raffi)

村上直也 (MURAKAMI, Naoya)

岡明央 (OKA, Akio)