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 １９ 日現在

機関番号：１２６０８
研究種目：基盤研究(B)（一般）
研究期間：2015～2018
課題番号：１５Ｈ０２６８３

研究課題名（和文）ハイブリッドプログラム解析を利用した機能識別に基づくプログラム理解支援技術

研究課題名（英文）Program comprehension based on feature identification using hybrid program analysis

研究代表者

小林　隆志（Kobayashi, Takashi）

東京工業大学・情報理工学院・准教授

研究者番号：５０３４５３８６

交付決定額（研究期間全体）：（直接経費）　　13,900,000円

研究成果の概要（和文）：本課題では，ソフトウェア保守活動中のプログラム理解支援を目的とし，外的機能と内的機能の関係を識別し可視化する手法の研究を行った．主要な成果としては，形式概念分析に基づく動的な機能識別手法に関係の類似性尺度を導入する新たな手法を提案した．共有モジュールに対し既存手法よりも優れた識別が可能であることを示した．また，重要な構成要素のみで表現される要約シーケンス図を自動復元する手法を提案した．さらに要約シーケンス図に対し検索や要約された構成要素の展開の機能を有する可視化ツールを開発したことで，プログラム理解の初期段階において段階的に外的機能の実現に関する情報を提供し支援することを可能とした．

研究成果の学術的意義や社会的意義
外的機能と内的機能に着目して機能識別を支援するという体系化を行っており，動的な機能識別と欠陥箇所特定手法を応用した新しい手法を開発しており，外的機能と内的機能の関係識別の精度を改善できることを示した点と，重要オブジェクトの特定と静的構造特徴に基づくグループ化を行うことで動的情報に対して段階的詳細化を行う手段を確立した点，実用規模のソフトウェアに対してその有効性を明らかにした点は学術的に意義があるものと考える．

研究成果の概要（英文）：In this research, we aim to support program comprehension during software maintenance and develop methods to identify and visualize the relationship between external and internal features.
We proposed a dynamic feature location technique (DFLT) using the formal concept analysis to identify relationships between modules and features. We use the similarity coefficient, which is used in fault localization techniques, as a relationship. Our DFLT make better orders shared modules compared with an existing DFLT. We also proposed a fully automated technique for recovering a summarized sequence diagram of a reasonable size. The recovered diagram depicts a behavioral overview of important concepts in a subject system, which can support developers to comprehend external features in an early stage of program comprehension. Our developed visualization tool is a flexible and lightweight tool and enable to explore a massive-scale sequence diagram by using searching, grouping and folding features.

研究分野：ソフトウェア工学

キーワード：ソフトウェア工学　ソフトウェア保守　プログラム理解　リバースエンジニアリング

１．研究開始当初の背景

ソフトウェアの更新は，新機能の追加や既存機能の修正のほかに，実行環境となる OS や利用しているライブラリ等のセキュリティ対応，新しいデバイスに対する対応などの多様な要因によって必要となる．

このような更新作業を適切に実施するためには，ソフトウェアが利用者に提供する**外的機能**を正確に理解し，それらの外的機能を実現するためにプログラム上でどのような**内的機能**として実装しているかを把握する**機能識別**が必要である．内的機能はクラスや関数といった実装上の単位の集合であり外的機能を構成する各処理に相当する．ソースコードを修正するためには，単に関数単位の識別をするだけではなく，関数内部の制御構造といった詳細な挙動までを理解する必要がある．しかしながら，大規模システムに対する長期にわたる保守活動では，時間的な制約から設計文書や正確な仕様は適切に追加・更新されず，上述の機能およびその実現方法に関する文書情報は不正確かつ断片的であることがほとんどである．開発者はプログラムからそれらの情報を読み取る必要があるが，現代におけるソフトウェアは大規模かつ複雑であるため，膨大な作業工数が必要となる．

プログラムから設計情報を復元する研究は以前よりなされているが，ソースコードを静的解析し，その構造と依存関係から設計情報を復元する方法が主流である．しかし，現代のソフトウェアはシステムの集合体であり，さらに個々のシステムは第三者が作成したフレームワーク，ライブラリを用いて実現されている．そのため，開発対象となっているプログラムコードに対する静的解析だけでは正確かつ十分な情報を取得できない．

２．研究の目的

上述のように，適切にソフトウェア更新をするためには，システムが提供する機能の実現方法をプログラムから正確に読み取る必要がある．本研究の目的は，ソフトウェアが提供する外的機能とその実現方法である内的機能とを区別して識別することでソフトウェア更新を効果的に行うためのプログラム理解の支援である．この支援のために，本研究では，外的機能の組み合わせを実行し，その実行トレースおよびソースコードの静的構造特徴，改版履歴を対象とした**ハイブリッドプログラム解析**を行うことで，外的機能を実現する内的機能およびその関係を識別し開発者のプログラム理解を支援する．段階的な抽象化を可能にすることで，従来困難であった中間の抽象度の設計情報復元を可能にする手法を確立する．

３．研究の方法

ソースコードのみではなく，外的機能の組み合わせを実行した実行トレースおよびソースコードの改版履歴を対象とした複数種の対象を解析するハイブリッドプログラム解析を行うことで，内的機能およびその関係を識別し開発者のプログラム理解を支援する手法の開発を目指す．具体的には以下の 4 つの技術の開発を行う．

・内的機能構造の識別の基盤技術（テーマ A）

本課題の目的である，ソフトウェア更新を効果的に行うためのプログラム理解支援の基盤技術として，実行トレースと様々な情報に形式概念分析（FCA）を適用することで， 開発者の視点の機能である内的機能を表現する関数・メソッドなどのプログラム構成要素の集合とそれらの間の関係を特定する機能識別手法の技術を確立する．内的機能の理解を支援するための，目的に応じた可視化を行うための段階的な抽象化を実現する手法を開発する．また，内的機能構造を利用することで，利用者視点の機能である外的機能に対する理解を支援し，機能の復元・詳細補完を行う手法を確立する．

・機能識別に特化した過去の開発における変更箇所の関係性分析手法（テーマ B）

リポジトリに含まれる改版履歴を解析することによって，上述の内的機能構造の識別のために有効な，プログラム構成要素間の論理的な関係を抽出する手法を確立する．また，プログラムを開発する際の操作履歴にも着目し，開発履歴中の改版履歴と操作履歴から進化結合を抽出し，静的な依存関係のみでなく進化結合を利用した外的機能の抽出手法について検討を行う．

・機能識別に特化した設計意図・静的構造特徴の活用手法（テーマ C）

特定の静的構造特徴や，デザインパターンやメタパターンの適用箇所を検出することで，その箇所に対する設計意図を抽出し，可読性や変更容易性の観点で分散記述されたソースコードの意味的なまとまりを検出し，内的機能間の関係性の分析およびプログラム理解のための抽象化に活用する手法を開発する．

・機能識別に特化した詳細な動的解析の応用手法（テーマ D）

実行トレースの動的特徴に基づいて，実行トレースを自動分割する実行フェーズ解析手法，登場するオブジェクトやクラスの重要度を算出する重要オブジェクト解析手法およびテーマ C の静的解析に基づく設計意図抽出手法を応用し，段階的抽象化のためのグループ化手法を開発する．

. . . . . .

４．研究成果

形式概念分析に基づいた機能識別手法の例題を整備し初期解析を行った．外的機能の組み合わせを実行した複数の実行トレースに FCA を適用することで,実行されたプログラム要素の共通性とその包含関係からプログラム要素の集合に対する束構造(概念束)が得られる．我々はこれまでに，外的機能がこの概念束の一部に対するラベルとして表現できることを示してきた．

本研究課題では，まずテーマ A として，形式概念分析によって，共通性及び可変性がどのように分析され，その結果重要な箇所が形式概念上のどの領域に出現する可能性があるかの分析・検討を行った． 分析結果に基づき，ハイブリッド解析で用いる静的解析および動的解析の解析基盤を整備し，動的解析に基づく機能捜索手法に対して欠陥箇所特定手法を応用することにより外的機能と関連の強い内的機能を構成するソフトウェアモジュールの特定を支援する手法を提案した．実際の OSS を題材に評価を行い，国際会議にて発表した．

また，この結果である形式概念束上の構造特徴を解析することにより外的機能間の関係を導出する手法を開発した．さらに，形式概念分析を利用する動的機能捜索を支援するために，既存のテストケース群が実行される際に呼び出すソフトウェアモジュールの同一性を分析することで，実行トレース内の内的機能の境界点を発見し，それらのモジュール名から外的機能名を推定する手法も開発し，論文発表を行った．これらの手法により，内的機能を構成するモジュールと外的機能との関係の効果的な解析が可能となった．

上記に並行してテーマ B の基礎となる，進化結合 (evolutionary coupling) を利用する方法に関して研究を行った．進化結合は改版履歴を分析することで得られる同時に変更される傾向の高いプログラム構成要素間の論理的な関係である．進化結合は単純な静的関係よりも強い依存関係であるため内的機能の構造を構成する関係として利用できる．

本課題では，より有益な進化結合を発見するためには時間的近接性が重要であることを３つの大規模オープンソースの履歴を解析することで明らかにし論文にまとめた．当該論文は，情報処理学会論文誌 2017 年度特選論文および，2017 年度論文賞を受賞した．また，進化結合を発展させ，改版履歴だけでなくプログラムの機能に対する設定情報とプログラムとの関係の解析を行うことで，特定の機能に関連した進化結合を抽出する手法を提案し口頭発表を行った．さらに，開発行動履歴を用いることで再現性の高い進化予測が可能となることも明らかにし，国際会議にて発表を行った．

テーマ C として,ソースコードから設計意図および静的構造特徴を抽出する手法の開発を行った．ソースコードから設計者の意図を完全に読み取ることは困難であるが，一部の設計特徴は詳細なプログラム解析によって抽出することができる．たとえば，デザインパターンやさらに低レベルのパターン Pree のメタパターンは，ソースコードを静的解析することによって適用箇所を特定することができる．本研究では Pree のメタパターンに着目することで，ソースコード中に分散記述された意味的な単位を発見し，テーマ D の段階的抽象化に用いる情報として利用する手法を提案した．実用規模の OSS を対象とした適用実験を行い，オブジェクトの重要度と合わることにより，効果的に対象ソフトウェアの概要を表現できることを示した．

また，変化し続けるソースコードにおいて，コード中の静的構造特徴を識別し，より理解しやすい表示を可能とする MAFP の開発を行った.MAFP では静的検査ツールにおける警告箇所を,ソースコードの変化に追随して追跡し，バージョン間における同じプログラム要素を識別可能にした．また，それぞれの警告箇所において false-positive な警告箇所など，修正が不要な警告箇所をマークすることで，その警告箇所を表示しないようにすることによって，膨大な警告を非表示にすることを可能にした.本手法で実現したプログラム要素のバージョン間追跡は,プログラム理解におけるバージョン間の理解支援にも応用可能である．

テーマ D として，動的解析に戻づくプログラム理解支援に関する複数の手法を開発した．ソフトウェア更新におけるプログラム理解支援では，作業工程によって異なる抽象度の情報が必要である．例えば，作業開始時は外的機能の一覧表示や，着目している外的機能を構成する内的機能の概要表示といった抽象度の高い表示が有効であり，理解が進み修正方法を検討する際には，着目する内的機能の詳細といった実装に近い抽象度を表示する必要がある．

本研究では，詳細な動的解析基盤を構築し，それらから得られたプログラムの実行時情報を用いて，プログラム中の要素の重要度および要素間の関係を分析し，実行トレースの要約を行う手法を開発した.提案手法は従来手法よりも高精度に重要要素を特定することが可能であり,実行時情報から抽出した関係性を考慮することによって，より高精度に重要箇所を特定することが可能であることを実験により示した.この重要箇所特定手法に関する発表は,2017 年度 情報処理学会コンピュータサイエンス領域奨励賞を受賞した．

また，プログラム理解の効率化の為には，各抽象度の情報表示をシームレスに行う必要がある．そのため，本研究では重要度や設計情報・静的構造特徴を利用した詳細の畳み込み・展開を対話的に実施できるインタフェースを実装し，段階的抽象化を可能化による可視化手法の開発を行った． 膨大な量となる実行トレースを可視化する手段として,既存の商用ツールであっても対応できない巨大なシーケンス図であっても描画・操作可能である可視化ツールの開発に

成功した．このツールは，プログラム理解の国際会議 IEEE International Conference on Program Comprehension にてツールデモ発表を行い，Best Tool Demo paper Award を受賞した．さらにその可視化ツールを用いて開発者にプログラムの振る舞いを図示することで，プログラムの理解を促進する手法についても研究を行った．また，研究の過程で，膨大な量の実行トレース情報を解析する基盤の重要性に着目し，オンライン型での制約検査器の構成方法について研究を行い，分散ストリームエンジンを用いた実行トレース検査基盤の開発を行った．

５．主な発表論文等

〔雑誌論文〕（計　24　件）

1. 加藤，林，佐伯：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，Vol. J98-D，pp.1374-1376，2015. （査読有）
   DOI：http://doi.org/10.14923/transinfj.2015SSL0001
2. 秦野，石尾，井上，SOBA：シンプルな Java バイトコード解析ツールキット，コンピュータソフトウェア，Vol.33，No.4，pp.4-15，2016(査読有)
   DOI：http://doi.org/10.11309/jssst.33.4_4
3. 渥美，桑原：MAFP：ソースコードに対する静的検査における警告の管理ツール，コンピュータソフトウェア，Vol.33，No.4，pp.1262-1273，2016(査読有)
   DOI：http://doi.org/10.11309/jssst.33.4_50
4. 風戸，林，大島，小林，夏川，星野，佐伯：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(査読有)
   https://ci.nii.ac.jp/naid/170000148548
5. N. Sae-Lim, S. Hayashi, M. Saeki：Context-Based Approach to Prioritize Code Smells for Prefactoring, Journal of Software：Evolution and Process, pp.1-24, 2018（査読有）DOI：https://doi.org/10.1002/smr.1886
6. 竹之内，石尾，井上：変数のデータフローによる API 利用コード例の検索，コンピュータソフトウェア Vol.34，No.4，pp. 68-74，2017（査読有）
   DOI：https://doi.org/10.11309/jssst.34.4_68
7. M. W. Mkaouer, M. Kessentini, M. O Cinneide, S. Hayashi, K. Deb：A Robust Multi-Objective Approach to Balance Severity and Importance of Refactoring Opportunities, Empirical Software Engineering, Vol.22, pp.894-927, 2017(査読有)
   DOI：https://doi.org/10.1007/s10664-016-9426-8
8. 林，柳田，佐伯，三村：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（査読有）
   https://ci.nii.ac.jp/naid/170000148540
9. 森，Hagward,小林：改版履歴の分析に基づく変更支援手法における時間的近接性の考慮と同一作業コミットの統合による影響，情報処理学会論文誌 Vol.58，No.4，pp. 807-817，2017（査読有）**[情報処理学会論文誌 特選論文 および 論文賞受賞]**
   https://ci.nii.ac.jp/naid/170000148541
10. K. Noda, T. Kobayashi, N. Atsumi：Identifying Core Objects for Trace Summarization by Analyzing Reference Relations and Dynamic Properties, IEICE Transactions on Information and Systems, Vol.E101-D, pp. 1751-1765, 2018（査読有）
    DOI：https://doi.org/10.1587/transinf.2017KBP0018
11. S. Hayashi, F. Minami, M. Saeki：Detecting Architectural Violations Using Responsibility and Dependency Constraints of Components, IEICE Transactions on Information and Systems, Vol.181-D, pp. 1780-1789, 2018（査読有）
    DOI：https://doi.org/10.1587/transinf.2017KBP0023
12. N. Sae-Lim, S. Hayashi, M. Saeki：An Investigative Study on How Developers Filter and Prioritize Code Smells, IEICE Transactions on Information and Systems Vol.101-D, pp.1733-1742, 2018（査読有）DOI：https://doi.org/10.1587/transinf.2017KBP0006
13. K. Lyu, K. Noda, T. Kobayashi：Toward Interaction based Evaluation of Visualization Approaches to Comprehending the Program Behavior, Proc. 2nd International workshop on Mining and Analyzing INTeraction Histories(MAINT 2019),2019(査読有)
    DOI：https://doi.org/10.1109/MAINT.2019.8666933
14. 高橋，セーリム，林，佐伯：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，Vol.60，pp. 1040-1050，2019（査読有）
    https://ci.nii.ac.jp/naid/170000150280/
15. K. Lyu, K. Noda, T. Kobayashi：SDExplorer：a generic toolkit for smoothly exploring massive-scale sequence diagram, Proc. 26th IEEE/ACM International Conference on Program Comprehension (ICPC 2018), pp. 380-384, 2018（査読有）**[Best Tool Demo Paper Award 受賞]** DOI：https://doi.org/10.1145/3196321.3196366
16. （他 9 件）

〔学会発表〕(計　54　件)

1. T. Mori, A. Hagward, T. Kobayashi: Effects of Recency and Commits Aggregation on Change Guide Method Based on Change History Analysis, Tenth International Conference on Software Engineering Advances (ICSEA2015) 2015
2. A. Yamamori and T. Kobayashi, Can Developers' Interaction Data Improve Change Recommendation? 23rd IEEE Intl Conf. Software Analysis, Evolution, and Reengineering (SANER2016) 2016.
3. 今西, 渥美, 森崎, 山本, 阿草, 前処理命令の制御構造とその構造内のコード改変に関する調査, 情報処理学会 第191回ソフトウェア工 学研究発表会, 2016
4. 相澤, 小林: メソッドを横断するコードテンプレート発見のためのインライン展開戦略の検討, 電子情報通信学会 信学技報 No.SS2015-96, (電子情報通信学会 ソフトウェアサイエンス研究会 3月研究会), 2016.
5. 竹之内, 石尾, 井上, プログラミング言語の構造を考慮したAPI利用例検索ツール, 第23回 ソフトウェア工学の基礎ワークショップ(FOSE2016), 2016
6. 森, 小林, 林, 渥美: 前処理命令による可変点を考慮した共変更ルール抽出, 電子情報通信学会 信学技報 SS2016-70 (ソフトウェアサイエンス研究会 3月研究会), 2017
7. S. Lappalainen and T. Kobayashi: A Pattern Language for MVC Derivatives, 6th Asian Conference on Pattern Languages of Program, 2017
8. 野田, 小林, 渥美: 実行トレース抽象化を目的とした参照関係・アクセス解析によるコアオブジェクト特定, ソフトウェア工学研究会 3 月研究会(情報処理学会 情処研報 Vol.2017-SE-195, No.2, pp.1-8) 2017. **[2017 年度 情報処理学会コンピュータサイエンス領域奨励賞受賞]**
9. K. Maruyama, S. Hayashi, T. Omori: ChangeMacroRecorder: Recording Fine-Grained Textual Changes of Source Code, The 25th IEEE International Conference on Software Analysis, Evolution and Reengineering, 2018
10. R. Kula, C. Roover, D.German, T. Ishio, K.Inoue: A Generalized Model for Visualizing Library Popularity, Adoption, and Diffusion within a Software Ecosystem, The 25th IEEE International Conference on Software Analysis, Evolution and Reengineering, 2018
11. S. Hayashi, F. Minami, M. Saeki: Inference-Based Detection of Architectural Violations in MVC2, 12th International Conference on Software Technologies (ICSOFT 2017), 2017
12. N. Sae-Lim, S. Hayashi, M. Saeki: How Do Developers Select and Prioritize Code Smells? A Preliminary Study, 33rd IEEE International Conference on Software Maintenance and Evolution (ICSME 2017), 2017
13. A. Yamamori, A. Hagward, T. Kobayashi: Can Developers' Interaction Data Improve Change Recommendation? 41st IEEE Computer Society International Conference on Computers, Software & Applications (COMPSAC2017) 2017
14. K. Noda, T. Toda, T. Kobayashi, N. Atsumi: Identifying Core Objects for Trace Summarization Using Reference Relations and Access Analysis, 41st IEEE Computer Society International Conference on Computers, Software & Applications (COMPSAC2017), 2017
15. M. Nakano, K. Noda, S. Hayashi, T. Kobayashi: Mediating Turf Battles! Prioritizing Shared Modules in Locating Multiple Features, 41st IEEE Computer Society International Conference on Computers, Software & Applications (COMPSAC2017) 2017
16. T. Ishio, Y. Sakaguchi, K. Ito, K. Inoue: Source File Set Search for Clone-and-Own Reuse Analysis, IEEE/ACM 14th International Conference on Mining Software Repositories (MSR2017), 2017
17. K. Maruyama, S. Hayashi, A Tool Supporting Postponable Refactoring, 39th International Conference on Software Engineering (ICSE 2017), 2017
18. N. Sae-Lim, S. Hayashi, M.Saeki: Revisiting Context-Based Code Smells Prioritization: On Supporting Referred Context, 9th International Workshop on Managing Technical Debt (MTD 2017), 2017
19. 藤原, 小林: 行動履歴分析に基づく変更支援における部分履歴抽出の影響調査, 電子情報通信学会 ソフトウェアサイエンス研究会 7月研究会, 2017
20. 中野, 野田, 小林, 林: 実行トレースの共通性分析に基づく機能開始点の特定, IEICE ソフトウェアサイエンス研究会3月研究会, 2018
21. Y. Ishida, Y. Arimatsu, K. Lyu, G. Takagi, K. Noda, T. Kobayashi: Generating Interactive View of Dynamic Aspect of API Usage Example, 3rd International Workshop on Dynamic Software Documentation (DySDoc3), 2018
22. Y. Arimatsu, Y. Ishida, K. Noda, T. Kobayashi: Enriching API Documentation by Relevant API Methods Recommendation based on Version History, 3rd International Workshop on Dynamic Software Documentation (DySDoc3), 2018

23. A. Takahashi, N. Sae-Lim, S. Hayashi, M.Saeki: A Preliminary Study on Using Code Smells to Improve Bug Localization, 26th IEEE/ACM International Conference on Program Comprehension (ICPC 2018), 2018
24. S. Sothornprapakorn, S. Hayashi, M. Saeki: Visualizing a Tangled Change for Supporting Its Decomposition and Commit Construction,42nd IEEE Computer Software and Applications Conference (COMPSAC 2018) 2018
25. 高橋, セーリム, 林, 佐伯: 情報検索に基づくBug Localizationに不吉な臭いが与える影響の調査, ソフトウェアエンジニアリングシンポジウム2018, 2018
26. R. E. Zapata, T. Ishio(4番目) et al: Towards Smoother Library Migrations: A Look at Vulnerable Dependency Migrations at Function Level for npm JavaScript Package, 34th IEEE International Conference on Software Maintenance and Evolution, 2018
27. Y. Ueda, T. Ishio, A. Ihara, K. Matsumoto, Impact of Coding Style Checker on Code Review -A case study on the OpenStack projects, 9th International Workshop on Empirical Software Engineering in Practice, 2018
28. 石田, 小林: 改版履歴分析に基づく変更漏れ防止支援における変更ルール集約と順位付けの効果, IEICE ソフトウェアサイエンス研究会 3月研究会, 2019
29. R. Funaki, S. Hayashi, M. Saeki: The Impact of Systematic Edits in History Slicing, 16th International Conference on Mining Software Repositories, 2019
30. (他 25件)

６．研究組織

(1)研究分担者

研究分担者氏名：石尾　隆

ローマ字氏名：Takashi Isihio

所属研究機関名：奈良先端科学技術大学院大学

部局名：先端科学技術研究科

職名：准教授

研究者番号（8桁）：60452413

研究分担者氏名：林　晋平

ローマ字氏名：Shinpei Hayashi

所属研究機関名：東京工業大学

部局名：情報理工学院

職名：准教授

研究者番号（8桁）：40541975

研究分担者氏名：渥美 紀寿

ローマ字氏名：Noritoshi Atsumi

所属研究機関名：京都大学

部局名：学術情報メディアセンター

職名：助教

研究者番号（8桁）：70397446

(2)研究協力者

研究協力者氏名：野田　訓広

ローマ字氏名：Kunihiro Noda

研究協力者氏名：相澤　遥也

ローマ字氏名：Yuya Aizawa