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　４ 年　 ６ 月　 ２ 日現在

機関番号：１４６０３
研究種目：基盤研究(B)（一般）
研究期間：2018～2021
課題番号：１８Ｈ０３２２１

研究課題名（和文）実行トレース共有リポジトリを用いたソフトウェア変更の影響レビュー技術の研究

研究課題名（英文）Research on Code Reveiw Technologies for Softtwware Change Impact Analysis Using Execution Trace Repository

研究代表者
石尾　隆（Ishio, Takashi）

奈良先端科学技術大学院大学・先端科学技術研究科・准教授

研究者番号：６０４５２４１３

交付決定額（研究期間全体）：（直接経費）　　12,900,000円

研究成果の概要（和文）：本研究では、プログラムの実行トレースを収集し、それを開発者が共有することで、開発者がソフトウェアに対する変更の影響を分析する作業を支援する技術の開発を行った。
実行トレースの収集を実施するための方法として、収集手順そのものをプログラムとして記述可能とする環境を実現した。実行トレースの共有を実現するために、あらゆる動作を記録する全知デバッグ技術を発展させ、記録するデータ量の上限をあらかじめ決めた上で有用な情報のみを記録する準全知デバッグ技術を開発した。そして、２つの実行トレースの効率的な比較により動作の変化の影響範囲の分析を支援する技術を実現した。

研究成果の学術的意義や社会的意義
ソフトウェアの動作は、機能追加やバグ修正、実行環境の変更など、様々な要因で変化するものであり、その影響を分析するための新しい技術を開発したことで、ソフトウェア保守の効率を高めることに貢献すると考えている。
また、プログラムの動作を記録したトレースのデータは、従来、どれだけのデータ量になるのか予測できないという問題があり、実用的に使うことは難しいと考えられていた。本研究では、実行トレースのデータ量をあらかじめ制限した形でデータ収集を行っても有効性が損なわれないことを示したことから、今後、ソフトウェア工学研究における実行トレースデータの活用が広がることを期待している。

研究成果の概要（英文）：In this research, we have developed technologies to collect execution traces, to share the traces among software developers, and to support change impact analysis activities by utilizing the traces.
As a technology to collect execution traces, we have developed an environment to write the procedure of trace collection as a program.　To enable developers to share execution traces, we have developed near-omniscient debugging that record execution traces using a predefined size of data storage.　To support change impact analysis, we have developed a method to efficiently compare a pair of execution traces and detect the behavioral changes.

研究分野：ソフトウェア工学

キーワード：ソフトウェア品質管理　動的解析　コードレビュー　デバッグ

１．研究開始当初の背景

ソフトウェアは現代社会の様々な活動を支える重要な基盤であり、ソフトウェアのバグの発生は社会的に重大な損失となっている。英国ケンブリッジ大学の報告 [1] では、ソフトウェアのバグによって生じる損失は１年あたり 3120 億ドルであり、2008 年から 5 年間の損失の合計はギリシャ経済危機による損失の倍以上であると述べられている。

ソフトウェアにバグが混入することを防ぐため、ソフトウェアの変更に対しては回帰テストが行われるほか、変更をレビューしてからソフトウェアに取り込む Modern Code Review も開発者に広く受け入れられている [2]。このような努力にもかかわらず、ソフトウェアに対する変更は依然としてバグの原因となることが多い。たとえば、多くのオープンソースソフトウェアプロジェクトにおいて、ライブラリの最新バージョンの利用によって問題が発生し、過去のバージョンに入れ替えた事例が観測されている [3]。これは、ライブラリ開発プロジェクトにおいて、開発者による機能拡張などの変更が既存の機能の振舞いを破壊したにも関わらず、それがテストやレビューで見逃されたことを示している。つまり、現在行われているレビューによる局所的なコードの精読や，テストによる限定的な実行結果の確認だけでは，ソフトウェアに対する変更によって引き起こされるシステム全体の振舞いの変化が意図したものとなっているかを確認することは困難である。

この問題に対処するには、ソフトウェアの変更によって生じた振舞いの変化を開発者が効果的に確認し、予期せぬ振舞いの変化に早い段階で気付くことを可能とする手段が必要である。そうすれば、バグを未然に防ぐことが可能であると考えられる。

＜引用文献＞

[1] Cambridge University Study States Software Bugs Cost Economy $312 Billion Per Year. http://www.prweb.com/releases/2013/1/prweb10298185.htm
[2] A. Bacchelli and C. Bird: “Expectations, Outcomes, and Challenges of Modern Code Review.” Proceedings of the 35th IEEE/ACM International Conference on Software Engineering (ICSE), 2013.
[3] A. Ihara, D. Fujibayashi, H. Suwa, R. G. Kula and K. Matsumoto: “Understanding When to Adopt a Library: A Case Study on ASF Projects.” Proceedings of the 13th International Conference on Open Source Systems (OSS), 2017.

２．研究の目的

本研究では、ソフトウェアの実行トレースをリポジトリとして蓄積、共有することで、ソフトウェアの変更をレビューする開発者が、その変更によって生じるソフトウェアの予期せぬ振舞いの変化を効果的に発見することを可能とする手法を開発する。開発者が実行トレース収集のための設定やトレース自体を他の開発者と共有し、ソフトウェアを実行すると、その実行トレースがこれまで蓄積された実行トレースと照合され、開発者が注目すべき振舞いの変化の情報を自動的に得られる環境とする。

本研究の独自性は、実行トレースを用いて、ソフトウェアに対して「これまでと同じ振舞いか」という判断を可能とすることにある。従来のテストやコードレビュー技術は、ソフトウェアが「正しく動作するか」を判断の基準に用いるが、たとえばセキュリティ脆弱性の修正や設定ファイルの更新のように、開発者にとっても正しい動作が明確でない場合には適用が難しい。本研究における動作の変化の分析は、テストのような形での動作の表現を必要としないため、品質保証の適用範囲が拡大する。

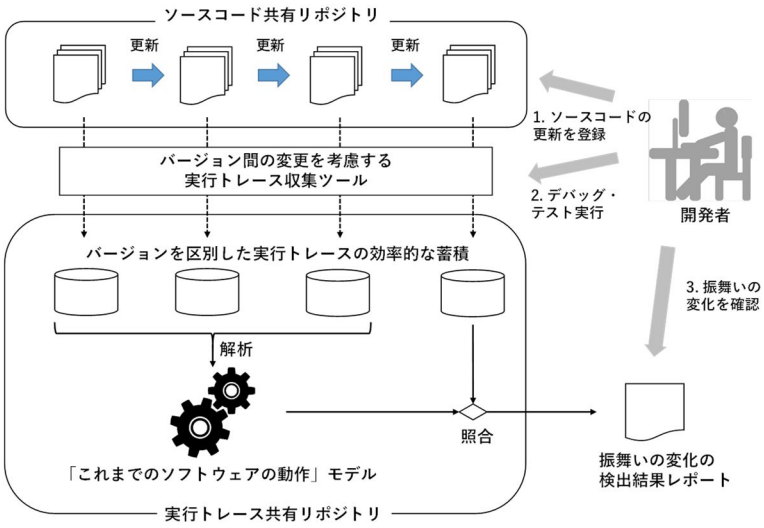

図 1　研究計画当初の実行トレース共有モデル

３．研究の方法

本研究では、目的を達成するための要素技術を以下のように整理し、研究を実施した。

(1) ソフトウェア開発者が、異なるソフトウェアのバージョン、異なる実行環境での動作を比較するための実行トレースを継続的に収集する方法。実行トレースを比較するには、ソフトウェアごとに、どのような条件で、またどのような環境で収集したのかが明確に記録される必要があると考えた。
(2) リポジトリに蓄えることが現実的と考えられる大きさに収まるソフトウェアの実行トレース収集法。従来、実行トレースの収集そのものは、あらゆる実行の振る舞いを記録する全知デバッグ技術という名前で提案されているが、単純な記録では実行トレースのデータ量が大きくなりすぎる。そこで、研究当初は、複数の実行トレースを効果的にまとめて保存する仕組みを研究することが必要であると考えた。研究の遂行途上で、小さな実行トレースで有用な情報を十分に保存できることが判明し、必要な情報のみを保存する実行トレースの収集法の研究へと切り替えた。
(3) ソースコードの変更内容や、ソフトウェア動作の変化を、実行トレースを活用して効率的に分析するための技術。ソースコードの更新による差分が、実行トレースとして観測された動作の違いと一貫しているかどうかを開発者が検査するための技術が必要であると考えた。

４．研究成果

本研究で得た成果を、研究方法の(1)実行トレースの収集法、(2)実行トレースの保存法、(3)実行トレースの活用法に対応付けると、以下のようなものとなる。

(1) 実行トレースの収集法としては、手順の記述方法と、実行環境そのものの分析・管理のための技術の研究を行った。

① 実行トレースの収集手順そのものをプログラムとして記述する環境 JISDLab を実現した[4]。この環境は、データ分析の手順、文書、プログラムをまとめて記述する Jupyter Notebook 環境に、Scriptable Debugging と呼ばれるプログラム制御のための技術を組み込み、プログラムを実行する環境、収集すべきデータ、プログラムに対して行う操作など一連の手順を実行可能な文書として記述することのできる環境である。図 2 はその実行環境で作成した文書の例であるが、何をしようとしているかを先頭に記述し、１つ目のコードブロック（灰色で示されている領域）でデバッガの起動を、２つめのコードブロックでデータ収集のための設定などを記述している。実行の結果もそのまま文書内に保存されるため、この文書自体を実行トレースの収集結果としても引き渡せるほか、開発者が再実行することで、実行結果に変化がないかも簡単に確認することができるようになっている。ファイル形式は Jupyter Notebook 形式と互換性があるため、作成した手順や記録したデータの閲覧だけであれば、GitHub 等の標準的な環境に簡単にアップロードできる。Java を対象とした実装を構築し、オープンソースソフトウェアとして公開した。

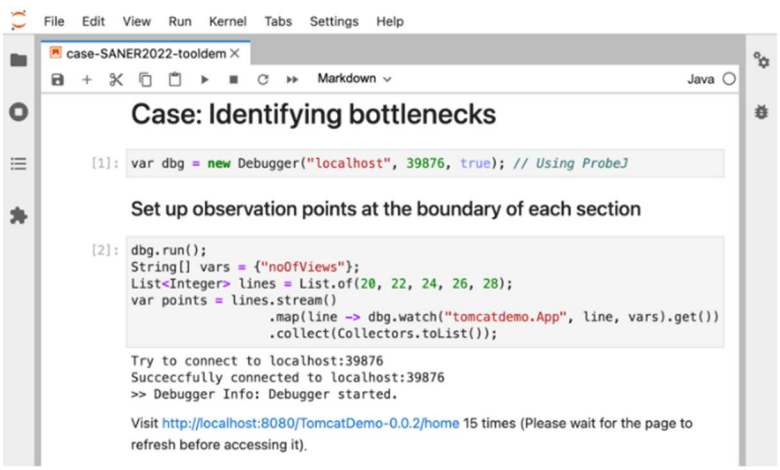

図 2　JISDLab の動作例

② ソフトウェアを作成する際に外部から取り込んだライブラリなどのファイルを、既知のファイルのデータベースと高速に照合し、バージョンを自動的に特定する技術を実現した。ファイルを効率よく比較する技術として、ファイル内容を要約したハッシュ値をあらかじめ計算し、比較する手法が知られているが、通常はファイルがわずかでも異なるとまったく異なる値を取るようなハッシュ関数を設計し、改ざん検知などに用いる。本研究では、ファイルが似ているほど類似したハッシュ値になる確率が高く、同一なときは必ず同じハッシュ値となるようなハッシュ関数を b-bit MinHash と呼ばれる技術に基づいて構築し、高速な（定数時間での）類似度推定を可能とした。

③ ソフトウェアを実行するために用いられたコンパイラ、ソースファイル等の環境情報を観測し、ハッシュ値を付与して記録、保存する仕組みを開発した。これらの環境情報は、開発者ごとに異なるため、実行トレースの収集環境の違いなどを検出するために役立つ。また、実行環境を準備するために必要な外部データのダウンロードなど、実行ごとに変化してしまう可能性のある要素の管理に有効である。

(2) 実行トレースの保存法としては、記録するデータ量の上限をあらかじめ決めた上で有用な情報のみを記録する準全知デバッグ技術を開発した。また、異常な動作（実行フェイズ）が発生したときのみ詳細な動作を記録するフェイズ検出技術も開発した。

① 実行トレースは、従来、プログラムの実行に関するあらゆる情報を詳細に保存する全知デバッグ技術が知られていた。これに対して、繰り返し実行される処理の経過のような、開発者にとっては相対的に価値が低い命令についてはデータを部分的に保存するという発想でデータの記録法を変更したのが準全知デバッグである。図 3 が準全知デバッグのモデルを示したもので、6 個の命令が実行されており、1～9 の数字が各命令の実行順序を示している。ここで、記録できるデータの量が、命令 6 回分に限られている場合を考える。単純に記録データ量に制限を設けるのであれば、「記録されたデータのうち、最も古いデータから破棄する」というのが一般的な発想である。図の下側、青色で示す枠が、そのような形で保存した場合に記録されるデータである。これに対して、準全知デバッグでは、各命令に対して最新の 1 回分ずつのデータを記録するようにした（図の右側、赤色の枠）。これにより、プログラムの初期化や終了処理のように、1 回しか実行されないが重要な命令については動作を完全に保存しておき、反復計算のような途中経過にあまり意味のない情報については、最終的な計算結果だけを保持することができる。プログラムの命令数が実行中に変動することは通常は起こらないため、プログラムの大きさから事前に記録する命令の回数、必要なデータ量を計画することができるようになった。結果として、保存が必要なデータ量は 1% 以下に削減しながら、プログラムの実行の調査にはほとんど支障が起きないことを確認した[5]。

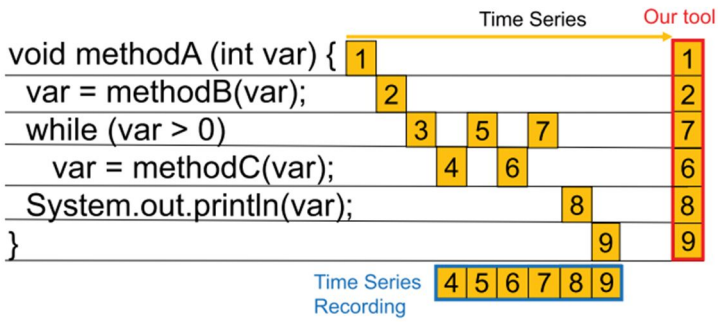

図 3 準全知デバッグのモデル

全知デバッグおよび準全知デバッグ技術を Java 向けに実現した実行トレース収集ツール SELogger を実装した。また、その実行トレースに記録されたプログラムの変数の値などを閲覧できる環境 NOD4J を構築した。いずれもオープンソースソフトウェアとして GitHub 上で公開している。図 4 は NOD4J の画面の一部を抜き出したものであり、プログラムのソースコード上に、変数の値が記録されている場合はハイライト表示を行い、値を確認できるようになっている。

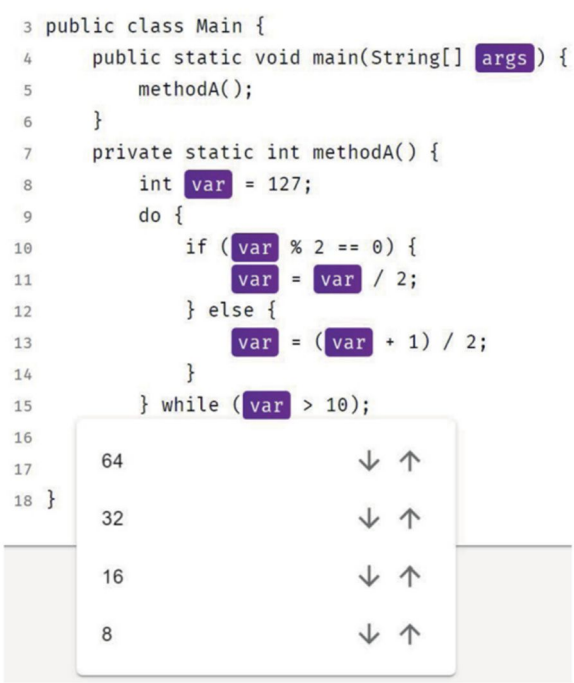

図 4 実行トレースの閲覧環境 NOD4J

② ソフトウェアの実行では、プログラムの書き方から、機能ごとに実行される主な命令はある程度固定されている。この情報を用いて、プログラムが現在実行している機能を識別する技術がフェイズ検出と呼ばれている。本研究では、通常の機能の実行と、異常な動作とを異なるフェイズとして識別する手法を開発した。これにより、ソフトウェアの外部から通常と異なる入力を受け取った場合など、異常な動作を検知した場合にのみ、詳細な動作を記録するようにソフトウェアを制御することが可能であることを示した。①の技術との併用により、実行トレースの効果的な収集が可能になる。

(3) 実行トレースの活用法としては、実行トレースの差分検出による振る舞い変化の検出技術を実現した。また、振る舞いの変化がコードレビューで実際に考慮されていることを明らかにした。

① 実行トレースの変化を検出する技術として、ソースコードのバージョン管理に用いられるツール git が持つ内部構造にヒントを得て、実行トレースの中で観測された値の系列の変化をハッシュ値として要約しておき、実行の差分を迅速に検出する手法を実現した。図 5 に模式図を示すが、一連の処理を表現する関数の 1 回の計算ごとに、ハッシュ値を割り当てておき、呼び出し先関数のハッシュ値の変化を呼び出し元関数のハッシュ値の変化へと伝播させることで、呼び出し先に変化があったことを検出できる。さらに、個々の関数ごとで実行された処理内容を表現するハッシュ値を別途保持しておくことで、呼び出し先の関数の動作には変化があったが、呼び出し元には影響がなかったことまで、ハッシュ値のみから検出することができる[6]。

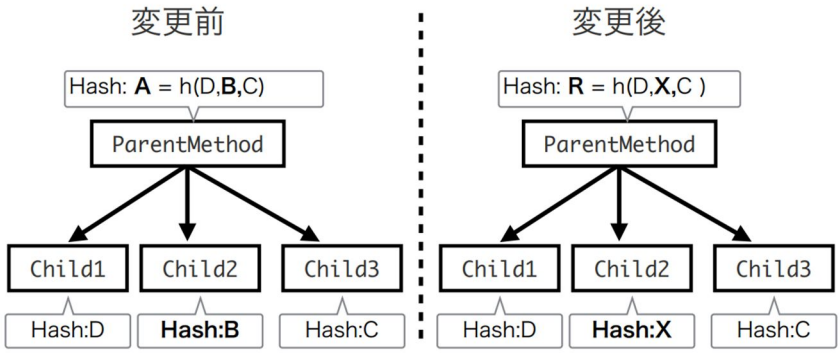

図 5　実行トレースのためのマークル木

② オープンソースソフトウェア開発で行われているソースコードレビューの活動データから、開発者が大きな変更を避けているなど、作業の方法に関する傾向が明らかとなっている。本研究では、ソフトウェアテストの自動生成技術を用いて、変更ごとにソフトウェアの動作が変化するかどうかを機械的に調査する方法を実現した。そして、実行トレースの比較により、実行時情報が大きく変化するような変更提案が、バグ修正などを除くと受け入れられにくい傾向があることを明らかにした。開発者が動作の違いを実際に考慮していることから、①で構築した実行トレースの差分の可視化が、コードレビュー等の開発者の活動の支援にも有効に働くと期待できる。

＜引用文献＞

[4] S. Sugiyama, T. Kobayashi, K. Shimari and T. Ishio: “JISDLab: A web-based interactive literate debugging environment.” Proceedings of the 29th IEEE International Conference on Software Analysis, Evolution and Reengineering, 2022.
[5] 嶋利 一真，石尾 隆，井上 克郎: “限られた保存領域を使用する Java プログラムの実行トレース記録手法.”コンピュータ ソフトウェア vol.36，no.4，pp.107-113，2019.
[6] 成 泰鏞，石尾 隆，松本 健一:“実行トレースのマークル木を用いたプログラム変更前後の差分検出法の提案.” 情報処理学会 研究報告，vol.2022-SE-210，no.25，2022 年.

５．主な発表論文等

〔雑誌論文〕　計6件（うち査読付論文　6件／うち国際共著　1件／うちオープンアクセス　5件）

１．著者名 上田 裕己、石尾 隆、伊原 彰紀、松本 健一	４．巻 37(2)
２．論文標題 コードレビュー作業において頻繁に修正されるソースコード改善内容の分析	５．発行年 2020年
３．雑誌名 コンピュータ ソフトウェア	６．最初と最後の頁 76～85
掲載論文のDOI（デジタルオブジェクト識別子） 10.11309/jssst.37.2_76	査読の有無 有
オープンアクセス オープンアクセスとしている（また、その予定である）	国際共著 －

１．著者名 伊藤 薫、石尾 隆、神田 哲也、井上 克郎	４．巻 J103-D
２．論文標題 軽量な類似度計算によるプロジェクト間のソースファイル集合の再利用検出	５．発行年 2020年
３．雑誌名 電子情報通信学会論文誌D 情報・システム	６．最初と最後の頁 542～554
掲載論文のDOI（デジタルオブジェクト識別子） 10.14923/transinfj.2019JDP7077	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

１．著者名 Bodin Chinthanet、Raula Gaikovina Kula、Shane McIntosh、Takashi Ishio、Akinori Ihara、Kenichi Matsumoto	４．巻 26
２．論文標題 Lags in the release, adoption, and propagation of npm vulnerability fixes	５．発行年 2021年
３．雑誌名 Empirical Software Engineering	６．最初と最後の頁 －
掲載論文のDOI（デジタルオブジェクト識別子） 10.1007/s10664-021-09951-x	査読の有無 有
オープンアクセス オープンアクセスとしている（また、その予定である）	国際共著 該当する

１．著者名 Kazumasa Shimari、Takashi Ishio、Tetsuya Kanda、Naoto Ishida、Katsuro Inoue	４．巻 206
２．論文標題 NOD4J: Near-omniscient debugging tool for Java using size-limited execution trace	５．発行年 2021年
３．雑誌名 Science of Computer Programming	６．最初と最後の頁 －
掲載論文のDOI（デジタルオブジェクト識別子） 10.1016/j.scico.2021.102630	査読の有無 有
オープンアクセス オープンアクセスとしている（また、その予定である）	国際共著 －

1．著者名 嶋利 一真、石尾 隆、井上 克郎	4．巻 36
2．論文標題 限られた保存領域を使用する Java プログラムの実行トレース記録手法	5．発行年 2019年
3．雑誌名 コンピュータ ソフトウェア	6．最初と最後の頁 4_107～4_113
掲載論文のDOI（デジタルオブジェクト識別子） 10.11309/jssst.36.4_107	査読の有無 有
オープンアクセス オープンアクセスとしている（また、その予定である）	国際共著 －

1．著者名 Kunihiro Noda、Takashi Kobayashi、Noritoshi Atsumi	4．巻 E101.D
2．論文標題 Identifying Core Objects for Trace Summarization by Analyzing Reference Relations and Dynamic Properties	5．発行年 2018年
3．雑誌名 IEICE Transactions on Information and Systems	6．最初と最後の頁 1751～1765
掲載論文のDOI（デジタルオブジェクト識別子） 10.1587/transinf.2017KBP0018	査読の有無 有
オープンアクセス オープンアクセスとしている（また、その予定である）	国際共著 －

〔学会発表〕　計29件（うち招待講演　1件／うち国際学会　9件）

1．発表者名 Bodin Chinthanet, Serena Elisa Ponta, Henrik Plate, Antonino Sabetta, Raula Gaikovina Kula, Takashi Ishio, Kenichi Matsumoto
2．発表標題 Code-based Vulnerability Detection in Node.js Applications: How far are we?
3．学会等名 The 35th IEEE/ACM International Conference on Automated Software Engineering（国際学会）
4．発表年 2020年

1．発表者名 Yuki Ueda, Takashi Ishio, Kenichi Matsumoto
2．発表標題 Automatically Customizing Static Analysis Tools to Coding Rules Really Followed by Developers
3．学会等名 28th IEEE International Conference on Software Analysis, Evolution and Reengineering（国際学会）
4．発表年 2021年

１．発表者名
上田 裕己，石尾 隆，松本 健一

２．発表標題
静的解析ツールの誤検出および検出漏れの最小化支援

３．学会等名
第206回ソフトウェア工学研究発表会

４．発表年
2020年

１．発表者名
西 陽太，石尾 隆，松本 健一

２．発表標題
プログラミング入門科目における提出プログラムのセマンティクスを考慮した自動分類手法

３．学会等名
第207回ソフトウェア工学研究発表会

４．発表年
2020年

１．発表者名
平ノ内 奎太，小林 隆志

２．発表標題
ImperSD：Java言語向け命令型スクリプタブルデバッグ環境

３．学会等名
電子情報通信学会ソフトウェアサイエンス研究会

４．発表年
2020年

１．発表者名
眞田 行隆、小林 隆志

２．発表標題
変更個所の構造的特徴の学習に基づく複合コミットの分割

３．学会等名
電子情報通信学会ソフトウェアサイエンス研究会

４．発表年
2021年

１．発表者名
石田 義八、小林 隆志

２．発表標題
共変更分析に基づく変更支援のための不完全な変更の収集と評価

３．学会等名
電子情報通信学会ソフトウェアサイエンス研究会

４．発表年
2021年

１．発表者名
杉山 朔太郎、小林 隆志

２．発表標題
Webベースの対話的な文芸的デバッグ環境の試作

３．学会等名
第207回ソフトウェア工学研究発表会

４．発表年
2021年

１．発表者名
内藤 祐樹、小林 隆志

２．発表標題
変更ルールマイニングのための解析範囲動的決定手法

３．学会等名
第207回ソフトウェア工学研究発表会

４．発表年
2021年

１．発表者名
李 聡、小林 隆志

２．発表標題
Untangling Composite Changes Using Tree-based Convolution Neural Network

３．学会等名
電子情報通信学会ソフトウェアサイエンス研究会

４．発表年
2021年

１．発表者名
Tsuyoshi Mizouchi, Kazumasa Shimari, Takashi Ishio and Katsuro Inoue

２．発表標題
PADLA: A Dynamic Log Level Adapter Using Online Phase Detection

３．学会等名
27th IEEE/ACM International Conference on Program Comprehension（国際学会）

４．発表年
2019年

１．発表者名
Kazumasa Shimari, Takashi Ishio, Tetsuya Kanda and Katsuro Inoue

２．発表標題
Near-Omniscient Debugging for Java Using Size-Limited Execution Trace

３．学会等名
35th IEEE International Conference on Software Maintenance and Evolution（国際学会）

４．発表年
2019年

１．発表者名
福元 春輝、伊原 彰紀、石尾 隆、上田 祐己

２．発表標題
プルリクエストにおける開発者の変更提案の分類

３．学会等名
情報処理学会関西支部支部大会

４．発表年
2019年

１．発表者名
相澤 遥也、森園 宏紀、小林隆志

２．発表標題
前処理命令解析と関数呼出し解析に基づく機能スイッチ特定

３．学会等名
ソフトウェアエンジニアリングシンポジウム2019

４．発表年
2019年

１．発表者名
酒井 宏樹，石尾 隆，井上 克郎

２．発表標題
類似した要素を検出できるブルームフィルタを用いた高速コード片検索手法

３．学会等名
電子情報通信学会ソフトウェアサイエンス研究会2019年3月研究会

４．発表年
2019年

１．発表者名
Takashi Ishio

２．発表標題
Software Ingredients: Detection of Third-party Components in Software Release

３．学会等名
Asia Pacific Society for Computing and Information Technology Annual Meeting（招待講演）（国際学会）

４．発表年
2018年

１．発表者名
嶋利 一真，石尾 隆，井上 克郎

２．発表標題
部分的な実行再現を目的とした実行トレース収集手法の調査

３．学会等名
IPSJ/SIGSE ソフトウェアエンジニアリングシンポジウム2018

４．発表年
2018年

１．発表者名
溝内 剛，嶋利 一真，石尾 隆，神田 哲也，井上 克郎

２．発表標題
フェイズ検出を用いたプログラムの性能バグ発生の自動検知

３．学会等名
IPSJ/SIGSE ソフトウェアエンジニアリングシンポジウム2018

４．発表年
2018年

１．発表者名
幾谷 吉晴，石尾 隆，吉上 康平，畑 秀明，松本 健一

２．発表標題
ブロックチェーンを用いたソフトウェア情報の組織間共有

３．学会等名
第25回ソフトウェア工学の基礎ワークショップ

４．発表年
2018年

１．発表者名
上田 裕己，伊原 彰紀，石尾 隆，松本 健一

２．発表標題
コードレビューを通じて行われるコーディングスタイル修正の分析

３．学会等名
第25回ソフトウェア工学の基礎ワークショップ

４．発表年
2018年

１．発表者名
Daiki Takata, Abdulaziz Alhefdhi, Maipradit Rungroj, Hideaki Hata, Hoa Khanh Dam, Takashi Ishio, Kenichi Matsumoto

２．発表標題
Catalogen: Generating Catalogs of Code Examples Collected from OSS

３．学会等名
3rd International Workshop on Dynamic Software Documentation（国際学会）

４．発表年
2018年

１．発表者名
石田 義八，小林 隆志

２．発表標題
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

３．学会等名
電子情報通信学会ソフトウェアサイエンス研究会2019年3月研究会

４．発表年
2019年

１．発表者名
有松 優，野田 訓広，小林 隆志

２．発表標題
分散ストリーム処理エンジンを用いたMTLによる大規模トレース検査

３．学会等名
電子情報通信学会ソフトウェアサイエンス研究会2019年3月研究会

４．発表年
2019年

１．発表者名
Kaixie Lyu, Kunihiro Noda, Takashi Kobayashi

２．発表標題
Toward Interaction based Evaluation of Visualization Approaches to Comprehending the Program Behavior

３．学会等名
2nd International Workshop on Mining and Analyzing Interaction Histories（国際学会）

４．発表年
2019年

１．発表者名
夏目 雅槻，相澤 遥也，渥美 紀寿，小林 隆志

２．発表標題
ソースコードのXML表現のための選択例を用いた対話的XPath生成支援

３．学会等名
電子情報通信学会ソフトウェアサイエンス研究会2018年10月研究会

４．発表年
2018年

１．発表者名
Yuu Arimatsu, Yoshiya Ishida, Kunihiro Noda, Takashi Kobayashi

２．発表標題
Enriching API Documentation by Relevant API Methods Recommendation based on Version History

３．学会等名
3rd International Workshop on Dynamic Software Documentation（国際学会）

４．発表年
2018年

１．発表者名 Yoshiya Ishida, Yuu Arimatsu, Kaixie Lyu, Go Takagi, Kunihiro Noda, Takashi Kobayashi
２．発表標題 Generating Interactive View of Dynamic Aspect of API Usage Example
３．学会等名 3rd International Workshop on Dynamic Software Documentation（国際学会）
４．発表年 2018年

１．発表者名 原口 大和，野田 訓広，小林 隆志
２．発表標題 メソッド入退出情報を利用した階層的欠陥箇所特定支援手法
３．学会等名 電子情報通信学会ソフトウェアサイエンス研究会2018年7月研究会
４．発表年 2018年

１．発表者名 平ノ内 奎太，野田 訓広，小林 隆志
２．発表標題 仮想ファイルシステムを用いたプログラム内部状態観測ツールの試作
３．学会等名 電子情報通信学会ソフトウェアサイエンス研究会2018年7月研究会
４．発表年 2018年

〔図書〕　計0件

〔出願〕　計0件

〔取得〕　計1件

産業財産権の名称	発明者	権利者
ソフトウェアビルドプロセス記録システムおよび方法ならびにソフトウェアビルドプロセス監視システム	幾谷 吉晴，石尾 隆，吉上 康平，畑 秀明，松本 健一	奈良先端科学技術大学院大学
産業財産権の種類、番号	取得年	国内・外国の別
特許、6692013	2020年	国内

〔その他〕

JISDLab: Java scriptable debugger on JupyterLab
https://github.com/tklab-group/JISDLab

SELogger: (Near-)omniscient logging tool for Java
https://github.com/takashi-ishio/SELogger

NOD4J: Near-Omniscient Debugger for Java
https://github.com/k-shimari/nod4j

6．研究組織

	氏名 （ローマ字氏名） （研究者番号）	所属研究機関・部局・職 （機関番号）	備考
研究分担者	伊原　彰紀 (Ihara Akinori) (40638392)	和歌山大学・システム工学部・講師 (14701)	
研究分担者	小林　隆志 (Kobayashi Takashi) (50345386)	東京工業大学・情報理工学院・准教授 (12608)	

7．科研費を使用して開催した国際研究集会

〔国際研究集会〕　計0件

8．本研究に関連して実施した国際共同研究の実施状況

共同研究相手国	相手方研究機関