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　６ 年　 ５ 月 ２４ 日現在

機関番号：１４６０３
研究種目：研究活動スタート支援
研究期間：2022～2023
課題番号：２２Ｋ２１２７９

研究課題名（和文）利用機能の実行時情報解析による厳密かつ効率的なライブラリの互換性検証手法

研究課題名（英文）Rigorous and Efficient Library Compatibility Verification Method based on Runtime Information Analysis of Used Functions

研究代表者

嶋利　一真（Shimari, Kazumasa）

奈良先端科学技術大学院大学・先端科学技術研究科・助教

研究者番号：５０９６４３７６

交付決定額（研究期間全体）：（直接経費）　　2,200,000円

研究成果の概要（和文）：本研究では，ライブラリバージョン更新の迅速な適用を目的として，テストの実行時情報を利用した厳密かつ効率的な互換性検証手法を確立した．
具体的には，（１）ライブラリメソッド内での実行の経路と値に基づく実行トレースを基にした厳密な実行トレースの比較による非互換性の検出と，（２）実行トレースを用いた自動テスト生成技術による，ユーザが利用しているライブラリの機能に対するテスト生成を実現した．これら２つの技術により，ライブラリバージョンが更新された際に，ユーザの利用機能に対してテストを生成し，実行トレースに基づく互換性検証を実現した．

研究成果の学術的意義や社会的意義
ライブラリは近年のソフトウェア開発において必要不可欠であり，バグや脆弱性の修正機能追加のために更新作業を迅速に行う必要がある．本研究で達成した厳密かつ効率的なライブラリの互換性検証手法により，ユーザの利用機能に特化して非互換性を迅速に発見してライブラリの更新を支援することができ，高品質なソフトウェア開発につながる．また利用機能に特化することで，ライブラリの非互換性の影響も受けにくく，互換性検証やそれに伴うライブラリ更新作業コストの減少も期待できる．

研究成果の概要（英文）：In this research, we developed a rigorous and efficient compatibility verification method using run-time information from tests for rapid application of library version updates.
Specifically, (1) the detection of incompatibility by rigorous execution trace comparison based on execution traces, including the path and value of execution within library methods, and (2) test case generation for the functions of the library used by the library clients using automatic test case generation techniques based on execution traces.
These two techniques enabled the generation of test cases for the functions used by library clients when the library version was updated and compatibility verification based on execution traces.

研究分野：ソフトウェア工学

キーワード：ソフトウェアテスト　動的解析　依存関係　ソフトウェア保守

１．研究開始当初の背景

近年のソフトウェア開発においてライブラリは必要不可欠であり，その利用にあたってはバージョン更新の適用作業が重要となっている．更新においてはバグ修正や脆弱性への対応が行われており，ユーザは自らが利用しているライブラリにおいて更新が実施された際に，迅速に更新を適用することが望ましい．

しかし，ライブラリの迅速な更新の適用においてはいくつかの課題が存在する．一つ目はユーザが非互換性を恐れてライブラリの更新を適用しない点である．Derr らの調査によると更新を適用しない理由として非互換性への恐れを上げたユーザが 50%にのぼっている[1]．また、Mostafa らの調査によると，ソフトウェアの単体テストのみでは十分にライブラリの非互換性を検知できない事が指摘されている[2]．そのため，ライブラリ更新の適用においては単体テストよりも詳細な実行時情報を用いた厳密な互換性検証を行う必要がある．二つ目はライブラリ互換性検証のためのテストの実施コストである．一般的に互換性検証においては，全てのテストスイートの実行による回帰テストが実施される．ただし，更新のたびに全てのテストを再実行するコストは大きい上に，ソフトウェアのライブラリ呼出しを網羅したテストが記述されているとは限らない．そのため，互換性検証においては指定したライブラリの更新に特化した網羅的かつ軽量なテスト実行が必要である．

しかし，網羅的かつ軽量なテスト実行やそれに伴って得られる実行時情報の扱いは難しく，これまでは実行を伴わない静的解析の手法が互換性検証に主に用いられ，実行時に決まるソフトウェアの挙動を捕捉するような厳密な互換性検証は実現されていない．この問題を解決するためには，変更箇所を通過するために十分なテストケースを用意した上で，実行時の分岐や値の違いといった細かな違いを捉えられる解析を用いた互換性検証を行う必要がある．

【参考文献】

[1] Erik Derr, Sven Bugiel, Sascha Fahl, Yasemin Acar, and Michael Backes, “Keep me updated: An empirical study of third-party library updatability on android,” In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17), pp. 2187-2200, 2017.

[2] Shaikh Mostafa, Rodney Rodriguez, and Xiaoyin Wang, “Experience paper: a study on behavioral backward incompatibilities of java software,” In Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2017), pp. 215-225, 2017.

２．研究の目的

本研究では，ユーザが利用しているライブラリの機能に特化した効率的かつ厳密な互換性検証手法の確立による，迅速なバージョン更新の適用支援を目的とする．ユーザが実際に利用している機能に特化することで，ユーザの利用機能以外において仕様変更などにより非互換性が生じた場合においても，ユーザは非互換性の影響を受けず効率的にライブラリの更新作業を行うことができる．また，互換性検証においては動的な情報を用いることでライブラリの振る舞いに基づいた厳密な互換性検証を行うことが出来る．

本研究の独自性は，ライブラリの互換性検証において動的情報を用いてライブラリの振る舞いを比較する点にある．既存研究におけるライブラリの互換性検証においては，主に静的解析による変更影響分析の技術が研究されていた．ライブラリのソースコード内の制御依存関係を取得し，更新があった際のソースコードの差分に着目することで互換性検証を行っていたが，動的に実行経路が決定されるコードが原因で偽陽性が生じるため，厳密な検証の実現においては動的な情報を用いる必要がある．本研究では，実開発で用いられているようなテストをもとに動的な情報を収集して比較検証を行う仕組みを作成することで，効率的かつ厳密な互換性検証の達成を目指した．

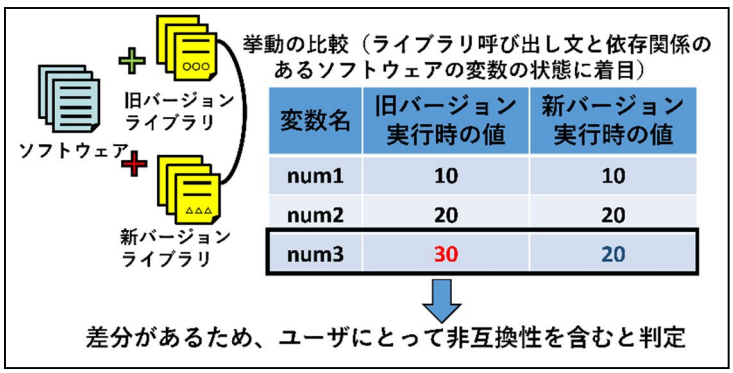

図１：研究計画当初の互換性検証の概要

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） ライブラリメソッド内での実行トレースを基にした，厳密な実行トレースの比較による非互換性の検出手法を実現する．実行トレースの比較においては，プログラム実行において，どのような実行経路に基づいてどのような値がメソッド間で受け渡されているか，に基づいて挙動の変化に着目する必要がある．

（２） ユーザが利用しているライブラリの機能に特化したテストの生成手法を実現する．実行トレースの収集においては，ソースコードの変更箇所をテストケースが通過する必要があり，ライブラリにおいて利用される機能のカバレッジを増加させる必要がある．そのため，可能な限りカバレッジが向上するようなテストケースを作成する必要がある．

４．研究成果

本研究で得られた成果を，（１）厳密な実行トレースの比較による非互換性の検出と，（２）ユーザが利用しているライブラリの機能に対する自動テスト生成に分けて述べる．

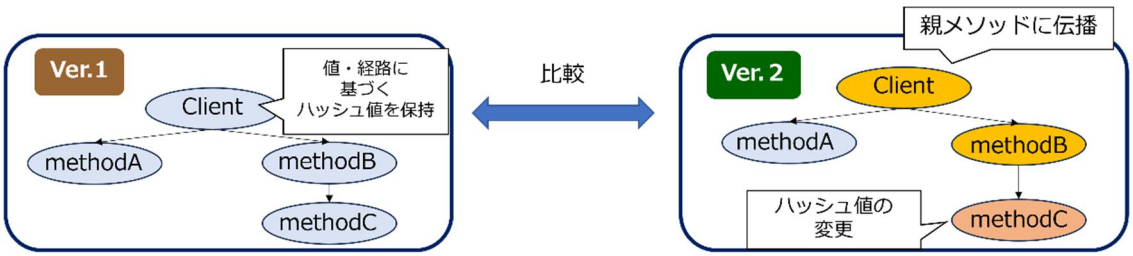

図２：厳密な互換性検証の手法の概要

（１） 厳密な実行トレースの比較による非互換性の検出

厳密な実行トレースの比較による非互換性の検出を実現するために，ライブラリの更新前後で挙動が変化する箇所を，メソッド単位で効率的に比較する．具体的には，更新されたライブラリのメソッドにおける実行命令列を基にした実行トレースと返り値を基にした実行トレースをマークル木を用いて効率的に比較し，差分が観測されたメソッドに着目した．図 2 に比較の概要を示す．メソッドごとに実行トレースからハッシュ値を計算してそれを保持する．バージョン間でハッシュ値に差分が発生した場合に，非互換性が発生したと言える．また，マークル木の構造を用いて，差分が発生した場合に子メソッドで変更があったことを親メソッドに伝播させることで，変更の影響範囲を明らかにすることができる．また，個々のメソッドにおけるハッシュ値の差分についても確認が可能である[3]．

提案手法を OSS におけるライブラリアップデートの事例に適用し，後方非互換の原因を検出できるかどうかを検証する．実験では 3 つのライブラリ更新事例における非互換性の事例を対象とした．結果として，非互換性の原因となったメソッドにおいてハッシュ値の変更が確認されたうえで，動的コールグラフのメソッドを単純にトレースする場合と比較して，約 1〜2 割の工数で非互換の原因となるメソッド候補を絞り込むことに成功した．また，記録される実行トレースの量も最大数 MB 程度であり，収集にかかる実行時間のオーバーヘッドも最大 1 分程度であった．したがって，現実的なオーバーヘッドで，ライブラリの非互換性のメソッドを特定することが可能と言える．

以上の結果から，提案手法による厳密な実行トレースの比較による非互換性の検出を現実的なコストで行えることを確認できた．

（２） ライブラリ機能に対する自動テスト生成

利用されているライブラリ機能に対する自動テスト生成テストを実現するために，対象ライブラリが OSS で利用されている事例を収集し，その OSS の単体テストを用いた自動テスト生成を行う．これにより，ライブラリの中でも実際に利用されている機能に特化したテストの生成が可能となる．図 3 に手法の概要を示す．

まず、テスト対象ライブラリを実行している OSS から実行トレースを収集する．具体的には，DynaPyt[4]というツールを用いて単体テストにおける特定のライブラリ呼び出しに関わる実行トレースを収集し，そこからテスト対象ライブラリに関連する部分を抽出する．次に，収集した実行トレースにおけるライブラリへの入出力をもとに，テストケースを作成する．最後に，テス

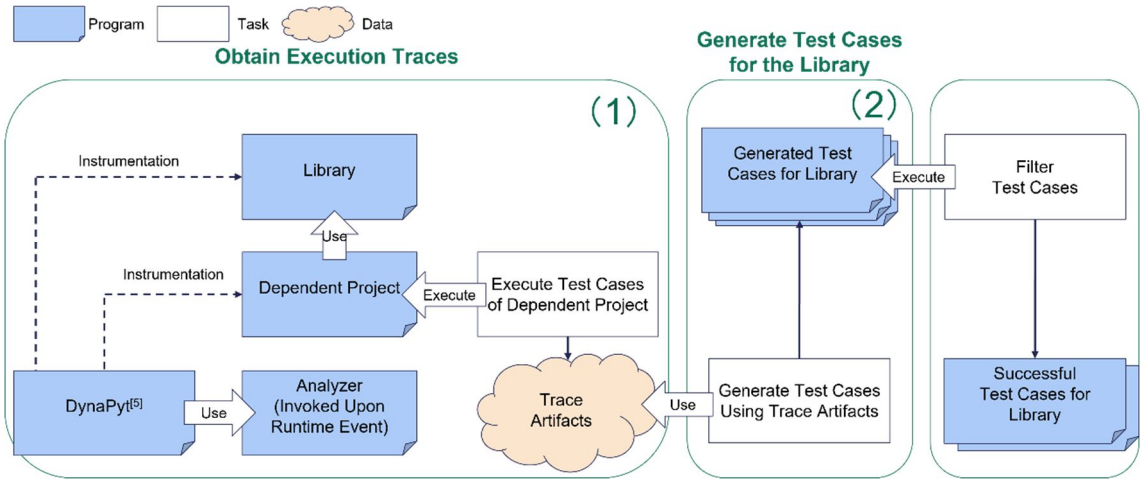

図 3：ライブラリ機能に対する自動テスト生成手法の概要

トを実行し，実行に失敗したテストケースについてフィルタリングした[5]．

提案手法を実 OSS へ適用した結果を表 1 に示す．テスト作成対象のライブラリ 1 件に対して，クライアントプロジェクトを 3 件用意し，元のライブラリのテストのカバレッジを増加させることが出来るかについて確認を行った．結果として，全てのライブラリにおいて，テスト対象ライブラリのカバレッジの増加を確認した．また，これらはすべて実際のライブラリ利用状況に基づいたテストケースであるため，実用的なテストケースの生成ができたと言える．

また，生成コストについても評価を行ったところ，いずれの事例も実行トレースの収集からテストケースの生成まで 1 時間以内で行えることが確認できた．以上の結果から，有用なテストケースの生成手法であるといえる．

表 1：カバレッジ増加率

Library	Client Project	Coverage	Improvement
flair		45.46%	
	textattack	45.50%	**+0.05%**
	pydata-wrangler	45.46%	+0%
	textwiser	45.46%	+0%
	All of above	45.50%	**+0.05%**
pdfrw		64.51%	
	pdf-annotate	64.94%	**+0.43%**
	dungeon-sheets	64.67%	**+0.16%**
	PyPDFForm	65.21%	**+0.70%**
	All of above	65.27%	**+0.75%**
sacred		78.62%	
	imitation	78.86%	**+0.25%**
	seml	78.62%	+0%
	scikit-datasets	78.82%	**+0.20%**
	All of above	78.93%	**+0.31%**

[3] Atsuhito Yamaoka, Teyon son, Kazumasa Shimari, Takashi Ishio, Kenichi Matsumoto, “Comparing Execution Trace Using Merkle-Tree to Detect Backward Incompatibilities,” Proceedings of the International Conference on Software Analysis, Evolution and Reengineering (SANER 2024), Rovaniemi, Finland, pp.649-653, 2024.

[4] Aryaz Eghbali and Michael Pradel. “DynaPyt: a dynamic analysis framework for Python,” Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2022, pp. 760-771

[5] Keita Morisaki, Kazumasa Shimari, Takashi Ishio, Kenichi Matsumoto, “Test Case Generation for Python Libraries using Dependent Projects’ Test-Suites,” Proceedings of the International Conference on Software Analysis, Evolution and Reengineering - Companion (VST 2024), Rovaniemi, Finland, pp.167-174, 2024.

５．主な発表論文等

〔雑誌論文〕　計3件（うち査読付論文　3件／うち国際共著　0件／うちオープンアクセス　1件）

１．著者名 Shimari Kazumasa、Ishio Takashi、Kanda Tetsuya、Inoue Katsuro	４．巻 236
２．論文標題 Evaluating the effectiveness of size-limited execution trace with near-omniscient debugging	５．発行年 2024年
３．雑誌名 Science of Computer Programming	６．最初と最後の頁 103117～103117
掲載論文のDOI（デジタルオブジェクト識別子） 10.1016/j.scico.2024.103117	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 KITAOKA Tetsuya、KANZAKI Yuichiro、ISHIO Takashi、SHIMARI Kazumasa、MATSUMOTO Kenichi	４．巻 40
２．論文標題 Reliability Evaluation Framework for Obfuscating Transformations in Program Code.	５．発行年 2023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_37～4_46
掲載論文のDOI（デジタルオブジェクト識別子） 10.11309/jssst.40.4_37	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 村田 優斗、石尾 隆、嶋利 一真、松本 健一	４．巻 J107-D
２．論文標題 ROS アプリケーションにおけるトピック通信の記述パターンを用いたデータフロー可視化手法	５．発行年 2024年
３．雑誌名 電子情報通信学会論文誌 D	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） 10.14923/transinfj.2023JDL8011	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 －

〔学会発表〕　計11件（うち招待講演　1件／うち国際学会　4件）

１．発表者名 Keita Morisaki, Kazumasa Shimari, Takashi Ishio, Kenichi Matsumoto
２．発表標題 Test Case Generation for Python Libraries using Dependent Projects’ Test-Suites
３．学会等名 7th Workshop on Validation, Analysis and Evolution of Software Tests（国際学会）
４．発表年 2024年

１．発表者名
Atsuhito Yamaoka, Teyon son, Kazumasa Shimari, Takashi Ishio, Kenichi Matsumoto

２．発表標題
Comparing Execution Trace Using Merkle-Tree to Detect Backward Incompatibilities

３．学会等名
International Conference on Software Analysis, Evolution and Reengineering（国際学会）

４．発表年
2024年

１．発表者名
Kazuki Fukushima, Takashi Ishio, Kazumasa Shimari, Kenichi Matsumoto

２．発表標題
Towards Assessment of Practicality of Introductory Programming Course Using Vocabulary of Textbooks, Assignments, and Actual Projects

３．学会等名
35th International Conference on Software Engineering Education and Training（国際学会）

４．発表年
2023年

１．発表者名
篠原 遼太郎, 嶋利 一真, 福島 和希, 田中 慎之佑, 石尾 隆, 松本 健一

２．発表標題
Pythonプログラミング演習におけるプログラミング経験度とエラー修正時間の関係分析

３．学会等名
第216回ソフトウェア工学研究発表会

４．発表年
2024年

１．発表者名
田中 慎之佑, 嶋利 一真, 福島 和希, 石尾 隆, 松本 健一

２．発表標題
確率モデルを用いた初学者向け構文エラー修正支援手法の検討

３．学会等名
第214回ソフトウェア工学研究発表会

４．発表年
2023年

１．発表者名
北岡 哲哉，神崎 雄一郎，石尾 隆，嶋利 一真，松本 健一

２．発表標題
コード難読化ツールの信頼性を評価するフレームワークの検討

３．学会等名
第29回ソフトウェア工学の基礎ワークショップ

４．発表年
2022年

１．発表者名
内田 啓太，石尾 隆，嶋利 一真，松本 健一

２．発表標題
2つのWebアプリケーション間の類似する操作対象の対応関係抽出

３．学会等名
第29回ソフトウェア工学の基礎ワークショップ

４．発表年
2022年

１．発表者名
嶋利 一真

２．発表標題
限られた資源を用いた効率的なデバッグ手法に関する研究

３．学会等名
第212回ソフトウェア工学研究発表会（招待講演）

４．発表年
2022年

１．発表者名
村田 優斗，石尾 隆，嶋利 一真，松本 健一

２．発表標題
Topic通信処理記述の解析によるROSアプリケーションのデータフローの可視化

３．学会等名
第212回ソフトウェア工学研究発表会

４．発表年
2022年

１．発表者名 Tetsuya Kitaoka, Yuichiro Kanzaki, Takashi Ishio, Kazumasa Shimari, Kenichi Matsumoto
２．発表標題 ObfusEval: Evaluating Reliability of Obfuscating Transformations
３．学会等名 Annual Computer Security Applications Conference（国際学会）
４．発表年 2022年

１．発表者名 大和 祐介, 石尾 隆, 嶋利 一真, 松本 健一
２．発表標題 プログラミング演習におけるエラー自動解説の有用性の評価
３．学会等名 第213回ソフトウェア工学研究発表会
４．発表年 2023年

〔図書〕　計0件

〔産業財産権〕

〔その他〕

―

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

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