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　 ６ 月 １７ 日現在

機関番号：１４４０１
研究種目：基盤研究(B)（一般）
研究期間：2020～2023
課題番号：２０Ｈ０４１６６

研究課題名（和文）自動プログラム修正における欠陥限局に関する研究

研究課題名（英文）A Study on Fault Localization for Automated Program Repair

研究代表者
肥後　芳樹（Higo, Yoshiki）

大阪大学・大学院情報科学研究科・教授

研究者番号：７０４５２４１４

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）：本研究課題では，自動プログラム修正技術との親和性が高い欠陥限局技術を開発すべく，研究に取り組んだ．具体的には，「テストケースの自動生成による欠陥限局精度の向上」，「テストケースの重み付けによる欠陥限局精度の向上」，「対象プログラムと欠陥限局技術との親和性を定量的に評価する仕組みの考案」，および「例外処理を検査するテストと欠陥限局精度の関係調査」について取り組んだ．

研究成果の学術的意義や社会的意義
発生したバグを完全に自動で修正する自動プログラム修正技術が注目を集めているが，現在の技術レベルでは修正可能なバグは少なく，この技術を実システムで利用するのは現実的ではない．本研究では，自動プログラム修正技術の中で利用する欠陥限局技術に注目し，その高精度化を行った．本研究成果を用いることにより，自動プログラム修正技術の精度向上や高速化もある程度達成されるため，自動プログラム修正技術を実用的な技術として高める一助になる．

研究成果の概要（英文）：In this research project, we have worked to develop fault localization techniques that have a high affinity with automated program repair techniques. Specifically, we worked on "improving fault localization accuracy through automated test case generation", "improving fault localization accuracy through test case weighting", "developing a mechanism for quantitatively evaluating the affinity between the target program and fault localization techniques", and "investigating the relationship between test cases to check exception handling and fault localization accuracy".

研究分野：ソフトウェア工学

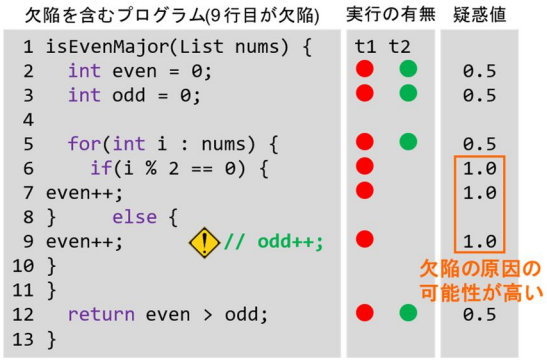
キーワード：欠陥限局　自動プログラム修正　ソースコード解析　テスト自動生成

様　式　C－１９、F－１９－１（共通）

１．研究開始当初の背景

ソフトウェア開発においてデバッグは多大な労力を必要とする作業であり，デバッグを効率的に行うための支援技術が必要不可欠である．デバッグ支援技術のひとつとして欠陥限局がある．**欠陥限局(Fault Localization)**とは，欠陥が存在するプログラムからその欠陥の原因箇所を推定する技術である．最も広く利用されているのはスペクトルベースの欠陥限局(Spectrum-based Fault Localization)である．この技術は，欠陥の存在するプログラムとテストケース(以降，テストと略す)群を入力として受け取り，「各テストが失敗したかどうか」，および「各テストが対象プログラムのどの要素を実行したか」の情報に基づき，プログラムの各要素（文や条件式）の疑惑値を計算する．疑惑値は[0,1]の値を取り，1 に近いほど欠陥の原因箇所である可能性が高いと判断されたことを表す．

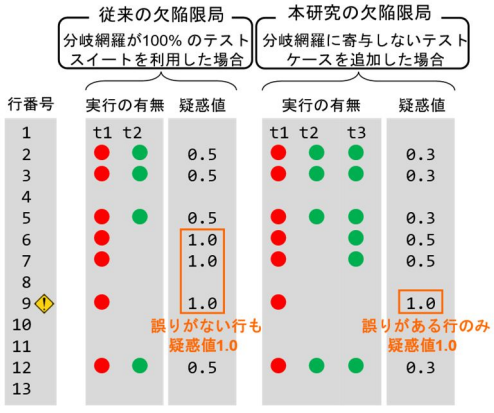
図 1 は，この技術を利用して欠陥限局を行った例を表している．この例では，対象のプログラムは整数のリストを入力として受け取り，そのリストに含まれている数字の過半数が偶数かどうかを判定する．この例では 2 つのテストが与えられている．テスト t1 は 2 と 5 から成るリストである．偶数が過半数ではないため，t1 に対するこのプログラムの期待値は false であるが，実際には true を返してしまう．一方，テスト t2 は空のリストである．t2 においても偶数が過半数ではないため，対象プログラムの期待値は false であり，このプログラムの実行結果も false となる．つまり，t1 が失敗テスト，t2 が成功テストである．



テストケース	入力値	期待値	出力値	成否
t1	[2, 5]	false	true	失敗
t2	[]	false	false	成功

図 1 欠陥限局の例

ソフトウェア開発におけるテスト工程では，対象プログラムをどの程度テストしたかを計る指標として命令網羅・分岐網羅・条件網羅がよく利用される．t1 および t2 を利用した場合，いずれの網羅も 100%となり，「欠陥の有無を確認するには十分なテスト群である」との評価になる．この 2 つのテストを利用して欠陥限局を行った結果が，図 1 の右端の``疑惑値"の列に示されている．6 行目，7 行目，9 行目が最も高い疑惑値となっている．この欠陥の原因箇所は 9 行目であり，``even++"を``odd++"に変更することでこの欠陥を除去できる．t1 と t2 を利用して欠陥限局を行った結果，ある程度は欠陥の原因箇所を正しく推定できたが，完全に特定するには至っていないことがわかる．



テストケース	入力値	期待値	出力値	成否
t1	[2, 5]	false	true	失敗
t2	[]	false	false	成功
t3(追加)	[4]	true	true	成功

図 2 新しいテストを追加した場合

ここで，新しいテストケース t3 を加えることを考える．t3 を加える前から命令網羅・分岐網羅・条件網羅はすでに 100%であるため，t3 はこれらの網羅率の向上には寄与しないテストである．t3 は 4 のみから成るリストである．t3 の期待値は true であり，実行結果も true となるため，成功テストである．t3 を加えた場合の欠陥限局の結果を図 2 に示す．t3 を加えることにより欠陥の原因箇所である 9 行目のみ 1.0 となり，より高精度に欠陥限局できたことがわかる．例を用いて示したように，欠陥の有無を確認するためのテストの指標である命令網羅・分岐網羅・条件網羅は，与えられたテスト群が欠陥限局において十分であるかどうかの指標にはならない．欠陥限局の結果がどの程度信頼できるかを表現するためには新たな指標が必要であり，本研究ではその指標の開発に取り組む．この新たな指標と，既存のテスト自動生成ツールを組み合わせることで，欠陥限局に必要十分なテストの自動生成ができるようになる．

２．研究の目的

本研究における欠陥限局手法は，**自動プログラム修正**(Automated Program Repair)において利用することを想定している．自動プログラム修正とは，欠陥のあるプログラムから完全に自動で欠陥を取り除く技術である．自動プログラム修正技術は，欠陥のあるプログラムのソースコードとテスト群（そのうちのいくつかは失敗テスト）を入力として受け取り，全てのテストが成功するようにソースコードを改変し，出力する．自動プログラム修正は，2 つの手順，欠陥限局とプログラム改変からなる．欠陥限局で欠陥の原因箇所として可能性の高いプログラム要素を推定し，プログラム改変でそのプログラム要素を変更することで欠陥の除去を試みる．研究代表者がいくつかの自動プログラム修正技術を利用して多数の欠陥に適用したところ，「欠陥とは無関係であるプログラム要素が改変されているが欠陥の原因箇所であるプログラム要素は改変されてない」ことが頻繁に発生していることを発見した．この発見により，自動プログラム修正技術の修正能力を向上させるためには，欠陥限局の精度を上げることが必要不可欠であると確信した．

３．研究の方法

欠陥限局の精度向上に向けて，本研究では下記の項目について取り組んだ．

A.　テストケースの自動生成による欠陥限局精度の向上
B.　テストケースの重み付けによる欠陥限局精度の向上
C.　対象プログラムと欠陥限局技術との親和性を定量的に評価する仕組みの考案
D.　例外処理を検査するテストと欠陥限局精度の関係調査

紙面の都合上，全ての項目について詳細に述べることはできない．よって，ABD についてはその概要のみを示し，特に大きな成果となった C について，「4.研究成果」で詳細に説明する．

(1) 「A.テストケースの自動生成による欠陥限局精度の向上」について

本研究では，欠陥限局の精度を向上させるため，既存のテストスイートの経路網羅を高めるテストケースを生成する手法について取り組んだ．提案手法は，欠陥を含むプログラムに対するテストケースを自動生成し，その中から既存のテストスイートの経路網羅を高めるテストケースのみを選択する．経路網羅とは，プログラムの全ての実行経路のうち，テストで実行された実行経路の割合である．経路網羅では，実行経路を文の実行順序や実行回数を考慮する文の列として考えるため，ループを含むプログラムでは実行経路の総数が無数になる．そのため，本研究では実行経路を文の列ではなく，文の実行順序や実行回数を考慮しない文の集合として考えることで実行経路の総数を有限に捉える．評価実験として，提案手法で生成したテストケースを既存のテストスイートに追加して欠陥限局を行った．その結果，70.6%の欠陥箇所において，既存テストケースのみを用いる場合よりも提案手法で生成したテストケースを追加した方が限局の精度が向上した．

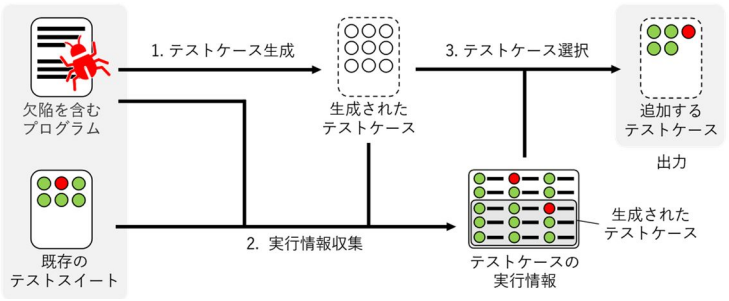


図 3 テストケース自動選択の流れ

(2) 「B.テストケースの重み付けによる欠陥限局精度の向上」について

本研究では，テストケースの実行経路が失敗テストケースにより近い成功テストケースに大きな重みを付与する手法を提案した．テストケースの成否を分けた小さな違いにこそ大きな価値があると考えたためである．しかし，先行研究[1]により実行経路の類似度を単純に比較し重み付けを行う手法ではあまり精度が向上しないことが報告されている．研究代表者はこの原因を，連続して実行される行の存在が重み付けに悪影響を与えている

ブロック化前

		ta	tb	tc
1:	if(n>5)	●	●	●
2:	n -= 8;	●	●	
3:	if(n<0)	●	●	●
4:	n += 1;	●		●
5:	n /= 3;	●		●
6:	n -= 5;	●		●
8:	if(n<-5)	●	●	●
9:	n += 1;	●		●
10:	return n;	●	●	●
	Result	F	P	P

ブロック化後

Block	ta	tb	tc
B1	●	●	●
B2	●	●	
B3	●	●	●
B4	●		●
B5	●	●	●
B6	●		●
B7	●	●	●
Result	F	P	P

図 4 ブロック化の例

ためだと捉えた．提案手法では．ブロック化という処理を行いこの問題を解決した上で実行経路を比較し，重み付けを行う．実験として，既存の SBFL と提案手法を比較し，ブロック化を行なった状態での重み付けにより，SBFL と差異が生じた欠陥を含む行の順位が最大 17.05%向上することを確認した．

(3)「D.例外処理を検査するテストと欠陥限局精度の関係調査」

これまでの研究により，実行経路に基づく欠陥限局手法は与えるテストによって欠陥限局の正確さに差異が生じることが示唆されている[2]．本研究では，例外処理を検査するテストに着目し，これらが実行経路に基づく欠陥限局手法の正確さに与える影響を調査した．例外処理は通常の制御フローと切り離されるため，例外処理を検査するテストと通常の制御フローを検査するテストでは実行経路が異なり，実行経路に基づく欠陥限局手法の正確さに大きな影響を与えると考えたためである．実際の開発過程で生じた欠陥とミューテーションツールで人工的に生成した欠陥を対象に調査し，失敗テストが全て例外期待テストの場合，実行経路に基づく欠陥限局手法の結果は信頼できる傾向にあることを確認した．

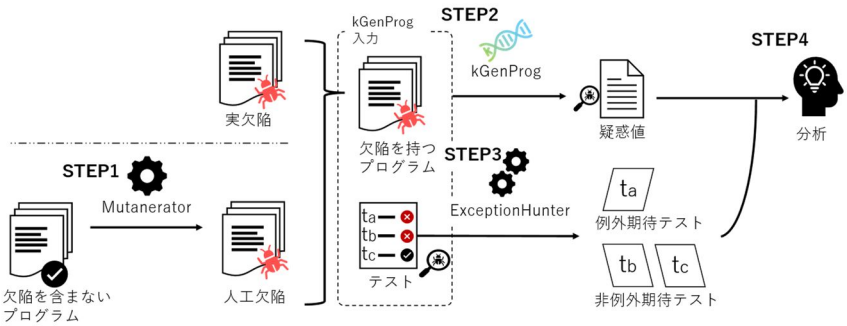


図 5 例外期待テストの影響を調査する実験の流れ

4．研究成果

ここでは，本研究で一番大きな成果となった「(3) 対象プログラムが欠陥限局技術との親和性を定量的に評価する仕組みの考案」について説明する．

(1) 研究の動機と提案手法の概要

現代のプログラミング言語にはさまざまな記述方法が用意されており，開発者は自身の好みや組織の方針に従って，必要な機能の実装方法を決定する．実装方法が変わるとテストケース毎の実行経路情報が変わる可能性があり，更に文の疑惑値やその順位も変わる可能性がある．従って，プログラム自体が SBFL を用いた欠陥箇所の特定にどの程度適しているかという特性を持っていると考えることができる．そこで本研究では，あるプログラムに対する SBFL を用いた欠陥限局の結果がどの程度正確かを，プログラムの SBFL 適合性と定義する．これは，あるプログラムに対する SBFL 適用技術との親和性の一種であり，品質特性である保守性，及びその副特性である解析性に含まれる 1 つの観点とみなすことができる．ソフトウェアの品質特性の 1 つの観点として SBFL 適合性を扱うことにより，下記の活動が可能になる．

- 現在のプログラムに対する SBFL 結果がどの程度信頼できるかを事前に把握する．信頼できると判断された場合は，SBFL 技術を利用することにより欠陥限局を行い，その結果を利用して開発者がデバッグ作業を行えば良い．
- SBFL 適合性が低いプログラムに対し，SBFL 適合性が向上するようなプログラム変換を行う．

プログラム (入力: a, b)	*susp*	t_1	t_2	t_3	t_4
s_1: `boolean result = false;`	0.50	✓	✓	✓	✓
s_2: `if (0 < a)`	0.50	✓	✓	✓	✓
s_3: `  result = true;`	0.00	✓	✓		
🐞s_4: `if (0 <= b) //correct: 0 < b`	0.50	✓	✓	✓	✓
s_5: `  result = true;`	0.50	✓	✓	✓	✓
s_6: `return result;`	0.50	✓	✓	✓	✓
テスト結果：		P	P	P	F

(a)リファクタリング前

プログラム (入力: a, b)	*susp*	t_1	t_2	t_3	t_4
s'_2: `if (0 < a)`	0.50	✓	✓	✓	✓
s'_3: `  return true;`	0.00	✓	✓		
🐞s'_4: `if (0 <= b) //correct: 0 < b`	0.71			✓	✓
s'_5: `  return true;`	0.71			✓	✓
s'_6: `return false;`	-				
テスト結果：		P	P	P	F

(b)リファクタリング後

テストケース	入力 (a, b)	期待値	実際の値
t_1:	(1, 1)	true	true
t_2:	(1, 0)	true	true
t_3:	(0, 1)	true	true
t_4:	(0, 0)	false	true

(c) テストスイート

図 6 SBFL 適合性が異なるプログラムの例

プログラム構造の違いによる SBFL 適合性の変化について，例を用いて説明する．図 6 に示すプログラム(a)とプログラム(b)は，機能が同じだが，構造が異なる．図 6 に示すテスト(c)を用いて両方のプログラムに SBFL を実行すると，各文の疑惑値が算出される．プログラム(a)では，欠陥がある箇所と同じ疑惑値を持つ文が 4 個存在する．一方，プログラム(b)では，欠陥がある箇所と同じ疑惑値を持つ文は 1 個である．欠陥がある箇所と同じ疑惑値を持つ文が少ないほど，確認しなくてはならない文が少なくなるため，SBFL による欠陥限局の精度が高いといえる．よって，この場合，(b)の方が SBFL 適合性が高い．

SBFL 適合性はソフトウェア品質を表す概念であるが，その具体的な計測方法として SBFL スコアを定義した．紙面の都合により詳細な計測方法は割愛するが，基本的なアイディアは与えられたプログラムに意図的に変更を加え，人工的な欠陥を混入することである．これらの人工的な欠陥を SBFL 技術でどの程度正確に特定できるかを計測することにより，そのプログラムがどの程度 SBFL に適しているかを定量的に評価する．

(2) 評価

研究代表者らが予め作成した機能等価メソッドのデータセット[3]に対して，実験を行った．実験の目的はどのようなプログラム構造だと SBFL 技術との親和性が高いのかを明らかにすることである．実験は下記の手順で行った．
手順 1：実験対象プログラムから SBFL スコアを計測
手順 2：SBFL スコアに差がないグループを考察対象から除外
手順 3：SBFL スコアが高くなる要因を考察

実験対象である 133 のグループのうち，120 のグループで SBFL スコアに差が生じていた．よって，これらが考察の対象である．考察の結果，図 7 に示す要因を明らかにすることができた．

要因	グループ数
分岐がない処理の代わりに `if` 文や `for` 文を用いている	21
early return のための `if` 文を追加している	15
`if` 文の条件式に論理演算子を用いず，別の文に分けている	12
同一条件分岐先で実行される文が少ない	62
不明	10

図 7 SBFL スコアが高くなる要因とグループ数

＜引用文献＞

[1] A. Bandyopadhyay and S. Ghosh, “Proximity based weighting of test cases to improve spectrum based fault localization,” Proc. International Conference on Automated Software Engineering, pp.420-423, 2011.
[2] Ali, S., Andrews, J. H., Dhandapani, T. and Wang, W.: Evaluating the Accuracy of Fault Localization Techniques, Proc. Int’l Conf. on Automated Software Engineering, pp. 76-87, 2009.
[3] Yoshiki Higo, Shinsuke Matsumoto, Shinji Kusumoto, and Kazuya Yasuda, "Constructing Dataset of Functionally Equivalent Java Methods", In Proc. of the 19th International Conference on Mining Software Repositories (MSR2022), Pennsylvania, May 23--24, 2022.

５．主な発表論文等

〔雑誌論文〕　計8件（うち査読付論文　8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 渡辺大登，松本真佑，肥後芳樹，楠本真二，倉林利行，切貫弘之，丹野治門	４．巻 63
２．論文標題 自動プログラム生成に対する多目的遺伝的アルゴリズムの導入 ー相補的な個体選択を目的としてー	５．発行年 2022年
３．雑誌名 情報処理学会論文誌	６．最初と最後の頁 1564-1573
掲載論文のDOI（デジタルオブジェクト識別子） なし	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 荻野翔,肥後芳樹,楠本真二	４．巻 63
２．論文標題 現実的な設定におけるメソッド粒度バグ予測モデルの構築および精度評価	５．発行年 2022年
３．雑誌名 情報処理学会論文誌	６．最初と最後の頁 973-985
掲載論文のDOI（デジタルオブジェクト識別子） なし	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 荻野翔，肥後芳樹，楠本真二	４．巻 63
２．論文標題 現実的な設定におけるメソッド粒度バグ予測モデルの構築および精度評価	５．発行年 2022年
３．雑誌名 情報処理学会論文誌	６．最初と最後の頁 973--985
掲載論文のDOI（デジタルオブジェクト識別子） 10.20729/00217601	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 中川将，肥後芳樹，楠本真二	４．巻 J104-D
２．論文標題 プルリクエスト型開発への統合を目的としたコードクローン修正支援システムCLIONE	５．発行年 2021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 690-701
掲載論文のDOI（デジタルオブジェクト識別子） なし	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 土居真之，肥後芳樹，楠本真二	4．巻 62
2．論文標題 ギャップを含むクローンセットの検出と評価	5．発行年 2021年
3．雑誌名 情報処理学会論文誌	6．最初と最後の頁 1350-1357
掲載論文のDOI（デジタルオブジェクト識別子） 10.20729/00211545	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 佐々木唯，肥後芳樹，松本真佑，楠本真二	4．巻 62
2．論文標題 プログラムに対する欠陥限局の適合性計測	5．発行年 2021年
3．雑誌名 情報処理学会論文誌	6．最初と最後の頁 1029-1038
掲載論文のDOI（デジタルオブジェクト識別子） 10.20729/00210553	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 丸山勝久，シン小茜	4．巻 62
2．論文標題 自動プログラム修正を用いたマージ競合の解決	5．発行年 2021年
3．雑誌名 情報処理学会論文誌	6．最初と最後の頁 2041-2055
掲載論文のDOI（デジタルオブジェクト識別子） 10.20729/00214246	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 松田直也，丸山勝久	4．巻 37
2．論文標題 テストケースが自動バグ修正に与える影響の調査	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 31-37
掲載論文のDOI（デジタルオブジェクト識別子） なし	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24件（うち招待講演　0件／うち国際学会　5件）

１．発表者名
Masashi Iriyama

２．発表標題
Classification of Changes Based on API

３．学会等名
the 23rd International Conference on Product-Focused Software Process Improvement (PROFES2022)

４．発表年
2022年

１．発表者名
Riku Takaichi

２．発表標題
Are NLP Metrics Suitable for Evaluating Generated Code

３．学会等名
the 23rd International Conference on Product-Focused Software Process Improvement (PROFES2022)

４．発表年
2022年

１．発表者名
Haruka Yoshikoka

２．発表標題
Improving Weighted-SBFL by Blocking Spectrum

３．学会等名
the 22nd IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM2022)

４．発表年
2022年

１．発表者名
Yoshiki Higo

２．発表標題
Constructing Dataset of Functionally Equivalent Java Methods

３．学会等名
the 19th International Conference on Mining Software Repositories (MSR2022)

４．発表年
2022年

１．発表者名
Kanon Harada

２．発表標題
A Preliminary Finding on Programs Fixed by an APR Tool based on a Genetic Algorithm

３．学会等名
the 29th Asia-Pacific Software Engineering Conference (APSEC 2022)

４．発表年
2022年

１．発表者名
原田和音

２．発表標題
GAベースの自動プログラム修正における出力プログラムの調査

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

４．発表年
2022年

１．発表者名
トウ ハクブン

２．発表標題
プルリクエストのレビューを支援するツールプラットフォーム

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
2023年

１．発表者名
中村 碧海

２．発表標題
コマンドの編集距離に基づくDockerfileにおける類似記述の検索

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
2023年

１．発表者名
古藤 寛大

２．発表標題
事前構文定義を必要としないリファクタリング検出手法の提案

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
2023年

１．発表者名
渡邉 凌雅

２．発表標題
プログラム構造が自動生成テストの網羅率に与える影響の調査

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

４．発表年
2023年

１．発表者名
久保 光生

２．発表標題
スペクトラムに基づく欠陥限局に適したプログラム構造の再調査

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

４．発表年
2023年

１．発表者名
橋本 周

２．発表標題
イミュータブルクラスを利用する必要性に関する調査 ~ハッシュ値を利用するデータ型を対象として~

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
2022年

１．発表者名
高市 陸

２．発表標題
文誤りを含むプログラムを評価可能なソースコード用自動評価尺度の調査

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

４．発表年
2022年

１．発表者名
Yoshiki Higo

２．発表標題
Tree-based Mining of Fine-grained Code Changes to Detect Unknown Change Patterns

３．学会等名
the 28th Asia-Pacific Software Engineering Conference (APSEC2021)（国際学会）

４．発表年
2021年

１．発表者名
Akira Fujimoto

２．発表標題
Tree-based Mining of Fine-grained Code Changes to Detect Unknown Change Patterns

３．学会等名
the 28th Asia-Pacific Software Engineering Conference (APSEC2021)（国際学会）

４．発表年
2021年

１．発表者名
吉岡遼

２．発表標題
実行経路の近似度を用いたテストケースの重み付けによるSBFLの精度向上

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
2022年

１．発表者名
松田直也

２．発表標題
テストケース選択による自動プログラム修正の効率化

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2021

４．発表年
2021年

１．発表者名
トウハクブン

２．発表標題
テストコードの変更に基づくプルリクエストのレビュー支援

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

４．発表年
2022年

１．発表者名
Tetsushi Kuma

２．発表標題
Improving the Accuracy of Spectrum-Based Fault Localization for Automated Program Repair

３．学会等名
28th International Conference on Program Comprehension（国際学会）

４．発表年
2020年

１．発表者名
Yui Sasaki

２．発表標題
SBFL-Suitability: A Software Characteristic for Fault Localization

３．学会等名
the 36th IEEE International Conference on Software Maintenance and Evolution（国際学会）

４．発表年
2020年

１．発表者名
Sho Ogino

２．発表標題
Evaluating Bug Prediction under Realistic Settings

３．学会等名
the 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
佐々木唯

２．発表標題
ソースコード記述に着目したFault Localizationに対する適合性の提案

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2020

４．発表年
2020年

１．発表者名
荻野翔

２．発表標題
現実的な設定に基づいたバグ予測モデルの構築及び精度評価

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
2020年

１．発表者名
丸山勝久

２．発表標題
自動プログラム修正によるマージ競合の自動解決を目指して

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2020

４．発表年
2020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
研究分担者	丸山　勝久 (Maruyama Katsuhisa) (30330012)	立命館大学・情報理工学部・教授 (34315)	
研究分担者	松本　真佑 (Matsumoto Shinsuke) (90583948)	大阪大学・大学院情報科学研究科・助教 (14401)	

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			
カナダ	ヴィクトリア大学			
タイ	マヒドン大学			