

Benefits and pitfalls of token-level SZZ: An empirical study on OSS projects

Hiroya Watanabe*, Masanari Kondo[†], Eunjong Choi* and Osamu Mizuno*

*Kyoto Institute of Technology, Kyoto, Japan

[†]Kyushu University, Fukuoka, Japan

Email: h-watanabe@se.is.kit.ac.jp, kondo@ait.kyushu-u.ac.jp, echoi@kit.ac.jp, o-mizuno@kit.ac.jp

The 31th IEEE International Conference on Software Analysis, Evolution and Reengineering
(SANER2024)

発表者：松尾 和輝

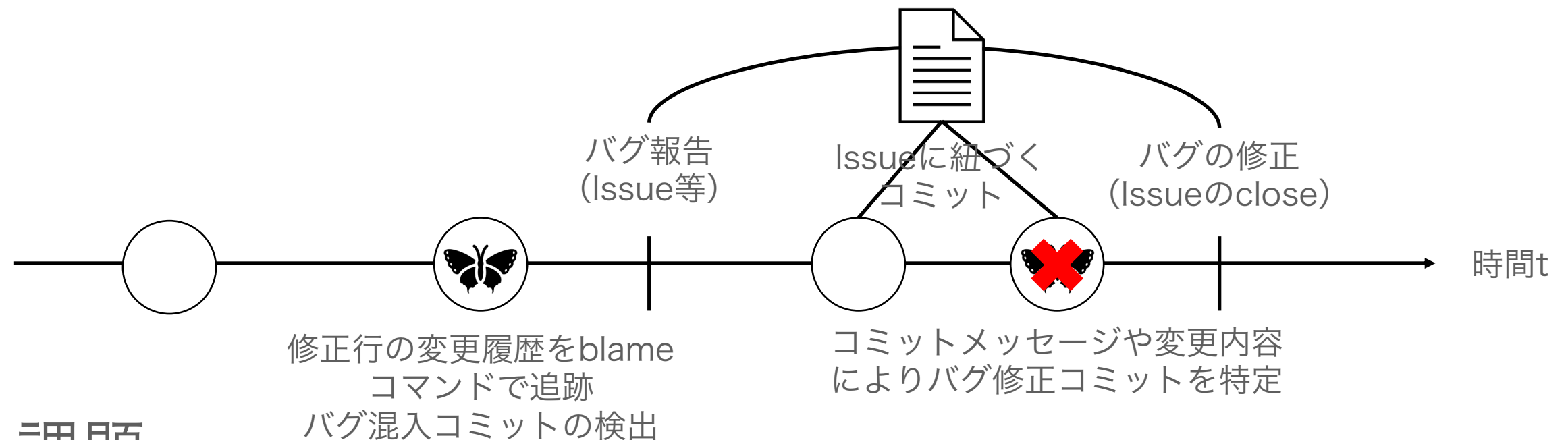
背景

- バグはソフトウェア工学における主要な研究分野
 - 意図しないソフトウェアの動作を引き起こし^[1,2,3], ソフトウェア開発企業にとって大きな財務的影響を与える^[4,5]
- 研究を推進するためにはデータセットが重要
 - 作成のコストがかかるため, 比較的小規模
- データセットの拡張にSZZアルゴリズムが活用
 - ただし, SZZアルゴリズムの精度依存である

[1] Z. Yin, D. Yuan, Y. Zhou, S. Pasupathy, and L. Bairavasundaram, "How do fixes become bugs?" in Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering, 2011, pp. 26–36.
[2] G. Bavota, B. De Carluccio, A. De Lucia, M. Di Penta, R. Oliveto, and O. Strollo, "When Does a Refactoring Induce Bugs? An Empirical Study," in Proceedings of the 12th International Working Conference on Source Code Analysis and Manipulation, 2012, pp. 104–113.
[3] D. Costa, S. McIntosh, W. Shang, U. Kulesza, R. Coelho, and A. E. Hassan, "A Framework for Evaluating the Results of the SZZ Approach for Identifying Bug-Introducing Changes," IEEE Transactions on Software Engineering, vol. 43, no. 7, pp. 641–657, 2017.
[4] "Updated NIST Software Uses Combination Testing to Catch Bugs Fast and Easy." [Online]. Available: <https://www.nist.gov/news-events/news/2010/11/updated-nist-software-uses-combination-testing-catchbugs-fast-and-easy>
[5] M. Zhivich and R. K. Cunningham, "The Real Cost of Software Errors," IEEE Security and Privacy, vol. 7, no. 2, pp. 87–90, 2009.

SZZアルゴリズム[12]

- バグが混入したコミットを検出する手法
 - 誰が, いつ, どこに混入したかを推定



- 課題
 - 行レベルの荒い検出手法である
- 着眼点
 - トークンレベルで検出することでより精度を向上させることができるのでは

[12] J. Sliwinski, T. Zimmermann, and A. Zeller, "When do changes induce fixes?" ACM SIGSOFT Software Engineering Notes, vol. 30, no. 4, pp. 1–5, 2005.

目的・リサーチクエスチョン

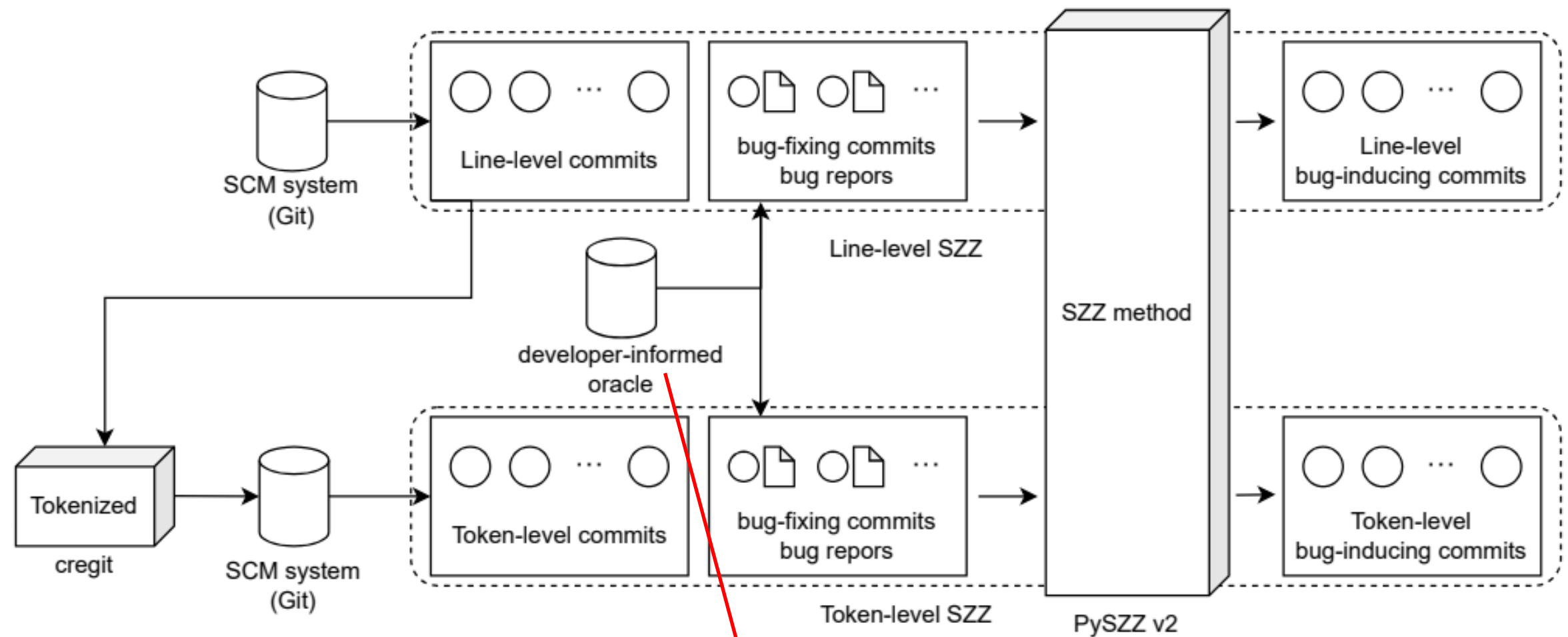
- 目的

- トークンレベルの変更履歴を用いたSZZの利点や落とし穴の調査

- リサーチクエスチョン

- RQ1：トークンレベルの変更はどの程度SZZの全体的な精度を向上させるか？
- RQ2：行レベルとトークンレベルのSZZを使用して識別されたバグ混入コミットが異なる理由は何か？
- RQ3：行レベルとトークンレベルのSZZで誤検出されたコミットが異なる理由は何か？

バグ混入コミット検出の流れ

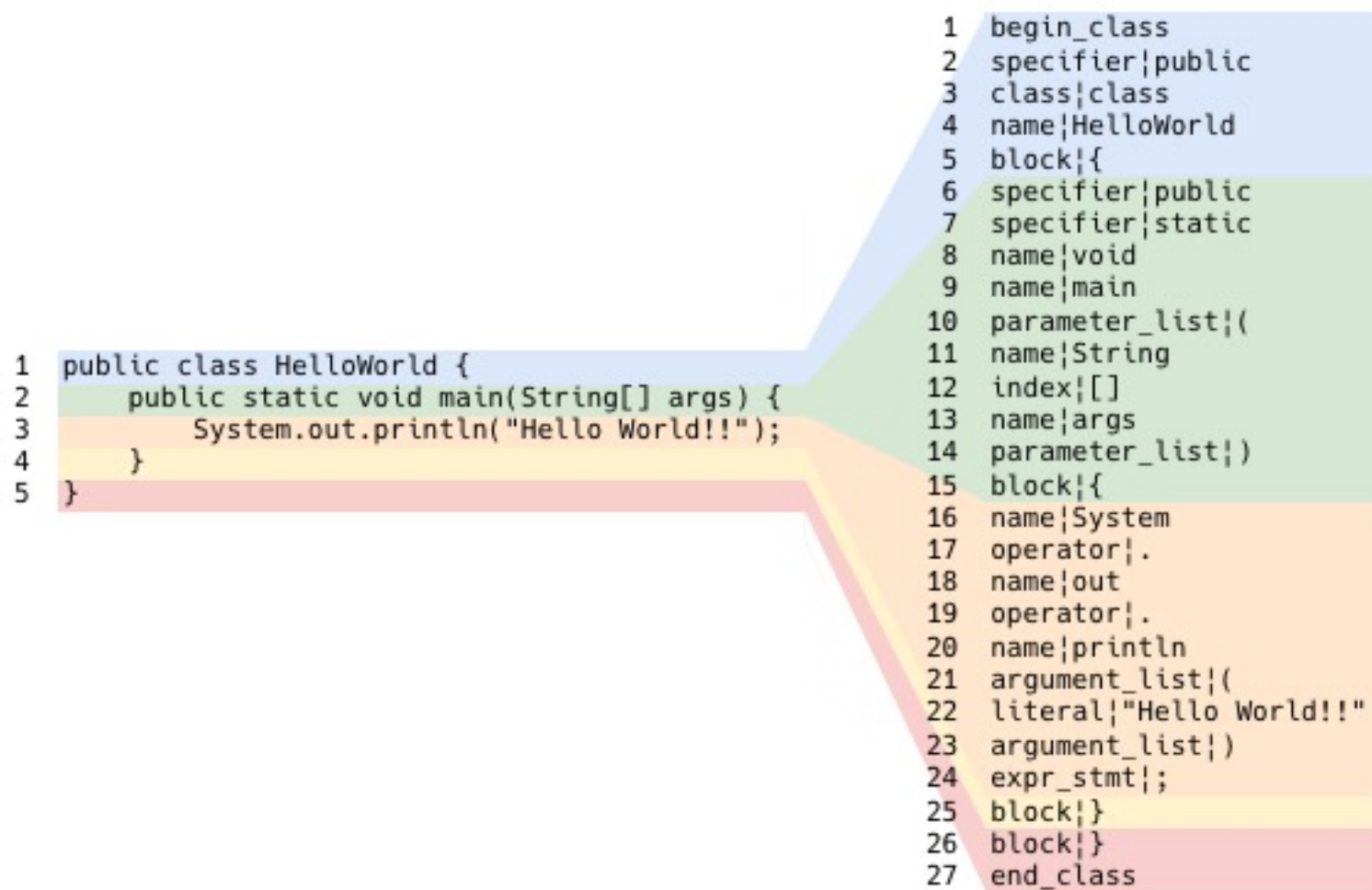


データセット [23]
68リポジトリ, 997,287コミット

[23] G. Rosa, L. Pascarella, S. Scalabrino, R. Tufano, G. Bavota, M. Lanza, and R. Oliveto, "A comprehensive evaluation of SZZ Variants through a developer-informed oracle," Journal of Systems and Software, vol. 202, no. C, 2023.

cregit^[13]

- トークン単位で変更を追跡するツール
 - トークン化後のコミットはオリジナルのコミットと紐付け



各行をトークン化

型 | トークン
の形式を取る

[13] D. M. German, B. Adams, and K. Stewart, "Cregit: Token-level blame information in git version control repositories," Empirical Software Engineering, vol. 24, no. 4, pp. 2725–2763, 2019.

RQ1 | 精度比較

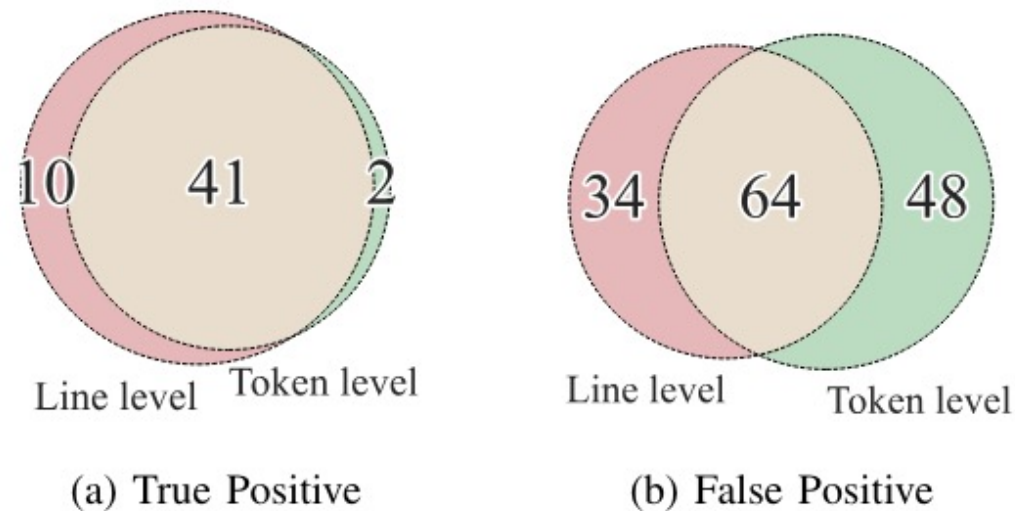
- アプローチ
 - Precision, Recall, F1 値で評価
- 結果
 - パフォーマンスについて

	Precision	Recall	F1 score	TP	FP
Line level	0.342	0.671	0.453	51	98
Token level	0.277	0.566	0.372	43	112
Difference	-0.065	-0.105	-0.081	-8	+14

行レベルのSZZと比較して、トークンレベルのSZZはF1値で
0.081低下

RQ1 | 精度比較

- 結果
 - オーバーラップについて



それぞれの手法でしか検出できないバグ混入コミットが存在

RQ1の回答

行レベルのSZZと比較してトークンレベルのSZZは全体的な精度は低下するものの、
それぞれの手法でしか検出できないバグ混入コミットが存在する
→組み合わせることで、精度を向上させることができる可能性

RQ2 | 識別されたバグ混入コミットが異なる理由

- アプローチ
 - 目視
- 結果
 - 行レベルのSZZのみ検出できるバグ混入コミット
 - 行レベルでの追加と削除を含み, トークンレベルでの追加のみを含むバグ修正コミット (8件)
 - 空白やインデントなどの外観上の変更を追跡することで特定できるバグ混入コミット (2件)

```
1928  if (Keyboard.getEventKey() ==  
    ↪  Keyboard.KEY_ESCAPE) {  
1929 -    if (!packetHandler.isLoadingLevel) {  
1929 +    if (!packetHandler.isLoadingLevel  
    ↪    || !isOnline()) {  
1930        pause();
```

(a) Changes at line-level

```
11405  name|packetHandler  
11406  operator|.   
11407  name|isLoadingLevel  
11408 + operator|||  
11409 + operator|!  
11410 + name|isOnline  
11411 + argument_list|()  
11412  condition|)  
11413  block|{
```

(b) Changes at token-level

git blameは追加のみの変更を追跡できないため

RQ2 | 識別されたバグ混入コミットが異なる理由

- 結果
 - トークンレベルのSZZのみ検出できるバグ混入コミット
 - diffが異なる(2件)

```
132 + logger.info("[eXist Home : "  
133 +         + System.getProperty("exist.home", "unknown") + "]);  
134 logger.info("[eXist Version : "  
135         + SystemProperties.getInstance().getSystemProperty("product-version", "unknown") + "]);  
136 logger.info("[eXist Build : "  
137         + SystemProperties.getInstance().getSystemProperty("product-build", "unknown") + "]);  
137 - logger.info("[eXist Home : "  
138 -         + SystemProperties.getInstance().getSystemProperty("exist.home", "unknown") + "]);  
138 logger.info("[Git commit : "  
139         + SystemProperties.getInstance().getSystemProperty("git-commit", "unknown") + "]);
```

(a) Changes at line-level tracking

行レベルのSZZは137行目, 138行目を132行目, 133行目へ配置変えを行なっていると判断

RQ2 | 識別されたバグ混入コミットが異なる理由

- 結果

- トークンレベルのSZZのみ検出できるバグ混入コミット
 - より正確なdiffを取得できる(2件)

トークンレベルのSZZは前ページ例の134行目～137行目の書き換えと判断

行の移動だと別の行だと判断してしまうため
追跡できなくなる

RQ2の回答

トークンレベルSZZはより正確なdiffを取得できるため、行レベルSZZでは検出できなかったバグ混入コミットを検出できる

```
843 operator|.
844 name|info
845 argument_list|(
846 - literal|"[eXist Version : "
846 + literal|"[eXist Home : "
847 operator|+
848 - name|SystemProperties
849 - operator|.
850 - name|getInstance
851 - argument_list|()
848 + name|System
849 operator|.
853 - name|getSystemProperty
850 + name|getProperty
851 argument_list|(
855 - literal|"product-version"
852 + literal|"exist.home"
853 argument_list|,
854 literal|"unknown"
855 argument_list|)
@@ -864,7 +861,7 @@
861 operator|.
862 name|info
863 argument_list|(
867 - literal|"[eXist Build : "
864 + literal|"[eXist Version : "
865 operator|+
866 name|SystemProperties
867 operator|.
@@ -873,7 +870,7 @@
870 operator|.
871 name|getSystemProperty
872 argument_list|(
876 - literal|"product-build"
873 + literal|"product-version"
```

(b) Changes at token-level tracking

RQ3 | 誤検出されたコミットが異なる理由

- アプローチ
 - 目視
- 結果
 - 行レベルSZZの34件の偽陽性のうち,
 - ソースコードの整形によるもの（7件）
 - 空白の修正によるもの（2件）
 - トークンレベルSZZではこれらのものを検出しない

RQ3の回答

トークンレベルのSZZを用いることで、特定のタイプの誤検出を無くすることが可能

利点・落とし穴

- トークンレベルSZZ

- 利点

- 見た目の変更（書式変更や空白の変更など）を無視できる
 - 行の一部の変更（トークンの変更など）を追跡できる

- 落とし穴

- トークンの追加だけのコミットを追跡できない
 - バグを誘発する偽陽性のコミットが大量に発生する可能性がある
 - 括弧（{}や()）やセミコロンを追跡することで、誤検出を引き起こすことが多い

- 行レベルSZZ

- 利点

- トークンを追加しただけの変更を追跡できる

- 落とし穴

- 見た目の変更（書式変更や空行の変更など）を追跡できる
 - 行を追加しただけのコミットを追跡できない

まとめ

- 目的
 - トークンレベルの変更履歴を用いたSZZの利点や落とし穴の調査
- 結果
 - トークンレベルSZZ単体では精度低, 行レベルSZZと組み合わせることで精度向上可能
- 所感
 - SZZにこだわる必要があるの？データ拡張がモチベーションなら深層学習とか使ったらいいんじゃない？（精度上がるかは知らんけど）
 - RQに答えてるような…, 答えてないような…, よくわからん論文