

コードレビュー改善に向けた 見逃し不具合の分析

和歌山大学大学院 システム工学研究科
オープンソースソフトウェア工学研究室
60190128 山崎 大輝

コードレビューとは

ソフトウェア開発において不具合を見つける方法のひとつ^[1]



コードを目視して以下の項目を検証

- コーディング規約
- コードの可読性
- ロジック

不具合の早期発見はソフトウェア開発のコスト削減につながる

[1]B. Boehm and V. R. Basili, “Software Defect Reduction Top 10 List,”IEEE Computer, vol.34, no.1 ,pp.135-137, 2001

開発プロセス



不具合修正ファイル(パッチ)作成者

開発プロセス



投稿



不具合発覚のため手戻り

不具合修正ファイル(パッチ)作成者

プロセスの後半になるほど
再修正コストが増大

コードレビューの問題点

約半数のコードレビューで不具合が見逃されている^[2]

- 進化性の不具合 - コード理解を困難にする

例) インデント, 命名規則のミス

- 機能的な不具合 - コード実行時にシステム障害を起こす

例) リソース, ロジックのミス

[2] O. Kononenko, O. Baysal, L. Guerrouj, Y. Cao, and M. W. Godfrey, "Investigating code review quality: Do people and participation matter?", IEEE International Conference on Software Maintenance and Evolution, pp. 111–120, 2015

コードレビューの既存研究

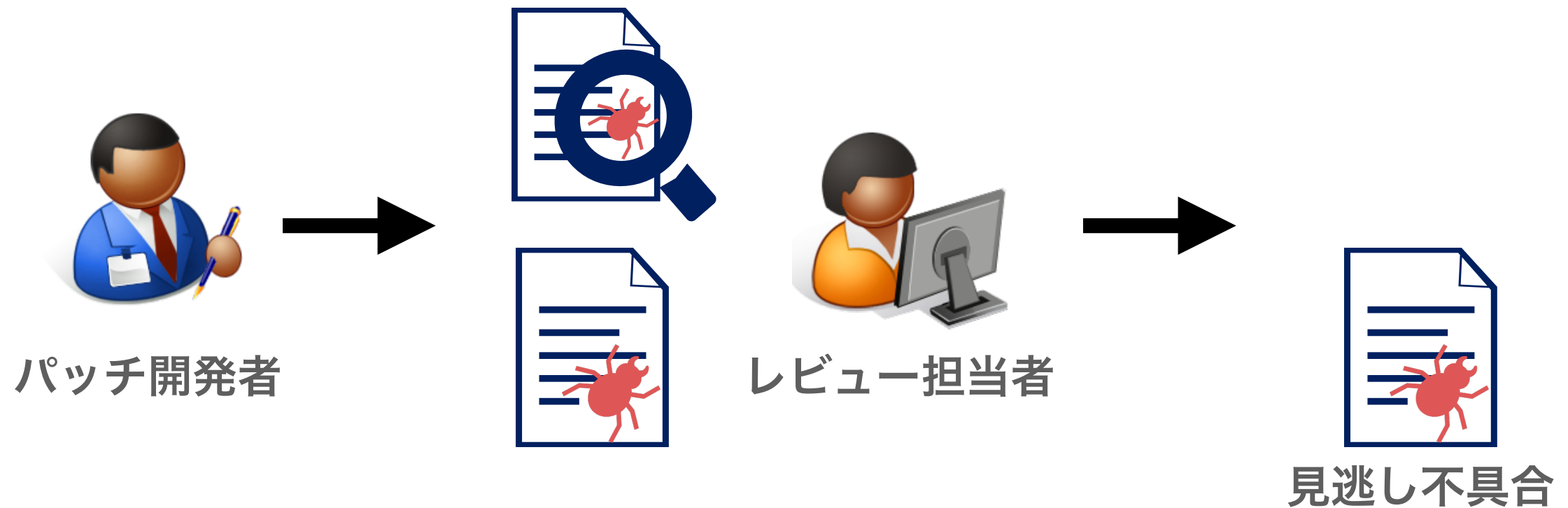
企業/学生のコードレビューで発見された不具合の分布を観察した結果, 70%以上が**進化性の不具合**^[3]

	企業のレビュー		学生のレビュー	
	件数	割合	件数	割合
進化性の不具合	276件	71.1%	287件	77.4%
機能的な不具合	83件	21.4%	49件	13.2%
偽陽性	29件	7.5%	35件	9.4%

[3] M. Mäntylä and C. Lassenius, "What Types of Defects Are Really Discovered in Code Reviews?", IEEE Transactions on Software Engineering, vol. 35, no. 3, pp. 430–448, 2009

既存研究の問題点

コードレビューで見逃される不具合の種類が不明



見逃される不具合の発見に向けた
レビュープロセスの改善ができない

本研究の目的とアプローチ

目的

レビューでの見逃し不具合を減らす手法の構築

(不具合を見逃すレビューの予兆を探す)

アプローチ

1.見逃し不具合の種類の分布を調査

2.見逃し不具合の種類とレビュープロセスの関係を調査

リサーチクエスチョン

RQ1：見逃し不具合はどのように分類されるか

発見される不具合と見逃される不具合の分布に違いがあるか

**RQ2：見逃し不具合の種類ごとにレビュープロセスに
違いがあるか**

不具合の見逃し原因を明らかにするため

RQ3：(不具合の予兆はあったか)

分析方法

RQ1：見逃し不具合はどのように分類されるか

以下の2つを目視することで見逃し不具合を分類する

①不具合管理システム(Bugzilla)での議論

②バージョン管理システム(GitHub)での修正内容

分析方法

RQ2：見逃し不具合の種類ごとにレビュープロセスに違いがあるか

以下のコードレビューメトリクスにおいて見逃し不具合の種類ごとに差があるかをKruskal-Wallis検定で調査

メトリクス	説明
コメント数	不具合に対するコメント数
レビュー時間	コードレビューに費やした時間
コード1行あたりのレビュー時間	レビュー時間をコード行数で割った時間

Kruskal-Wallis検定

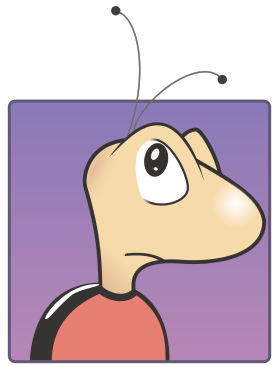
P値 > 0.05

$$H = \frac{12}{N(N + 1)} \sum_{i=1}^k \frac{R_i^2}{n_i} - 3(N + 1)$$

対応のない3群以上のグループ間に差があるか調べる検定
(どのグループ間で差があるかまでは不明)

- k : クラス数
- N : 全データ数
- n_i : クラス*i*のデータ数
- R_i : クラス*i*の順位和

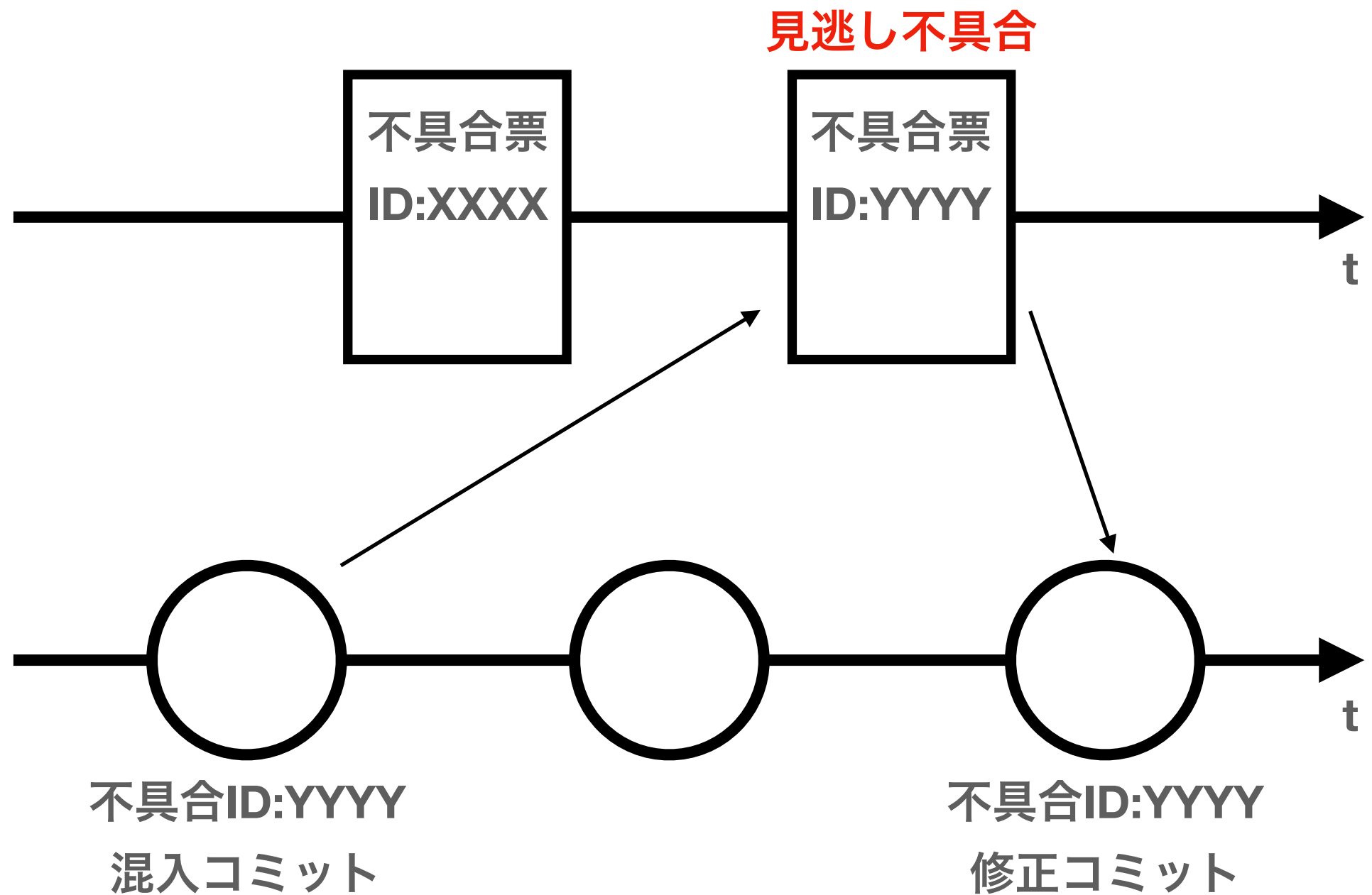
見逃し不具合の推定方法



Bugzilla

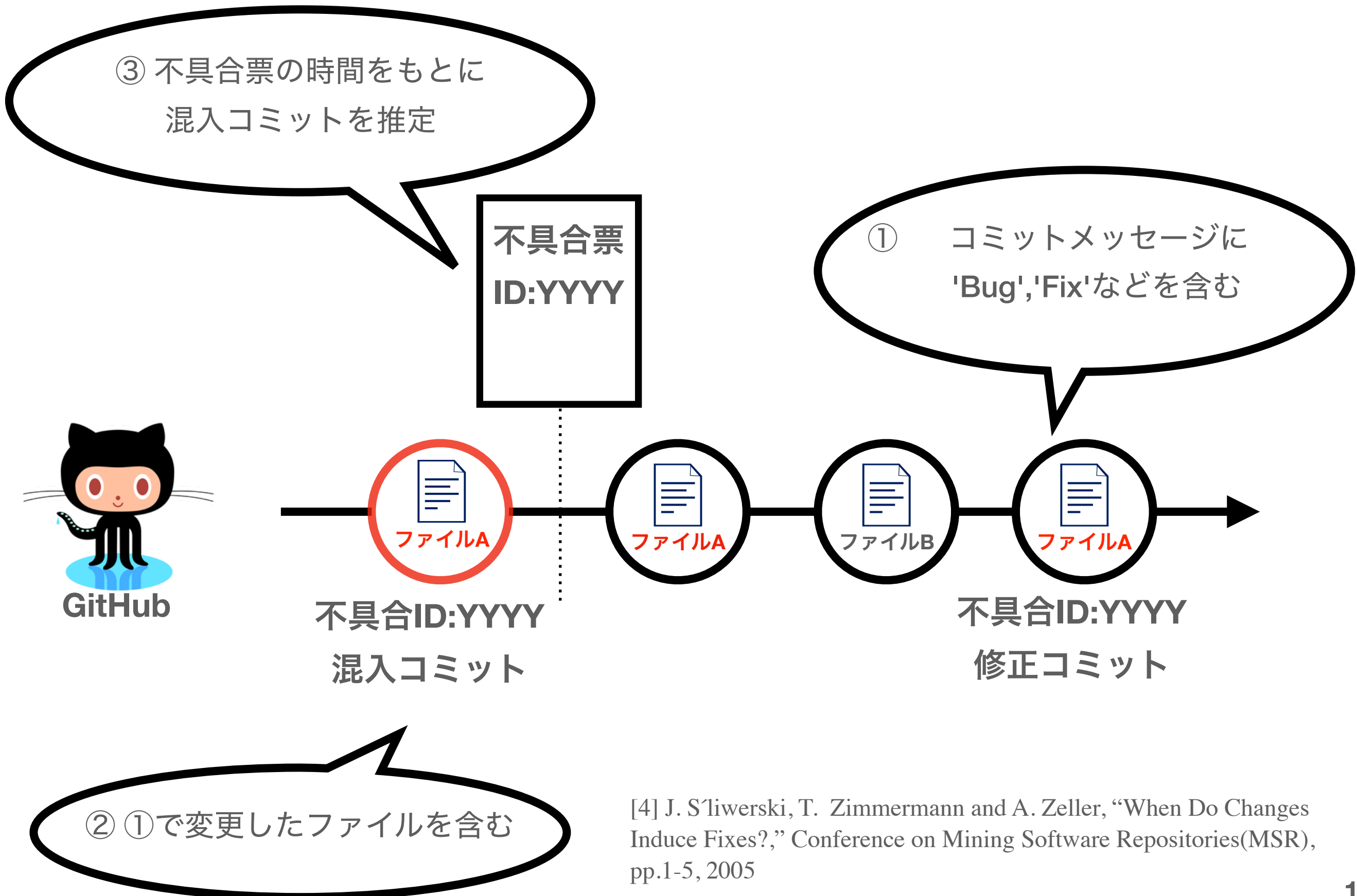


GitHub



不具合を見逃したレビュー

不具合修正/混入コミットの推定方法^[4]



データセット

対象プロジェクト：Gecko

MozillaのHTMLレンダリングエンジン

対象期間：2015/10/01 ~ 2016/09/30

1年分のコミット履歴と不具合票

不具合修正コミット	不具合混入コミット	見逃し不具合
660件	1,451件	416件

不具合の分類

コードレビューで発見された不具合を詳細に分類^[5]

不具合の分類		説明
進化性の不具合	文書化	ソースコードの意図を伝える情報が損なわれている不具合
	視覚的表現	可読性が損なわれている不具合
	構造	リファクタリングの提案
機能的な不具合	リソース	変数の初期化や操作に関する不具合
	チェック	特定の値を検出したときの検証に関する不具合
	インタフェース	ソフトウェアの外部とのやりとりに関する不具合
	ロジック	演算と実行順序に関する不具合

[5]M. V. Mantyla and C. Lassenius, “What Types of Defects Are Really Discovered in Code Reviews?,” IEEE Transactions on Software Engineering, vol. 35, no. 3, pp. 430–448, May 2009.

RQ1：見逃し不具合はどのように分類されるか

90%以上が機能的な不具合

不具合の分類		件数	割合
進化性の不具合	文書化	3	
	視覚的表現	1	
	構造	1	
機能的な不具合	リソース	96	
	チェック	114	
	インタフェース	9	
	ロジック	86	
偽陽性	機能性向上/機能欠陥	106	

RQ2：見逃し不具合の種類ごとにレビュープロセスは異なるか

各メトリクスでP値 > 0.05 のため、有意差はない

Kruskal-Wallis検定の結果

メトリクス	H値	P値
コメント数	1.69	0.79
レビュー時間	3.13	0.54
コード1行あたりの レビュー時間	1.80	0.77

RQ3 : (不具合の予兆はあったか)

境界値のミスなどは静的解析ツールで取れそう

レビューが（無意識のうちに）いい加減になっている？

考察 見逃し不具合の分類について

RQ1：見逃し不具合はどのように分類されるか

進化性の不具合は数件しか見逃されていない

既存研究では、コードレビューで発見できる不具合の70%以上が進化性の不具合

コードレビューにおいて進化性の不具合はほぼ発見されている

機能的な不具合に着目したレビューが必要となる

考察 機能的な不具合について

RQ1：見逃し不具合はどのように分類されるか

種類	原因	件数
リソース	変数の不足	28
	変数の操作ミス	28
	変数の初期化ミス	24
チェック	例外処理不足	49
	ヌルポインタ設定不足	15
	条件分岐の重複	3
ロジック	制御フローミス	28
	試行回数の調整	15
	比較演算ミス	9

考察 レビュープロセスの改善

以下の観点をもとにレビュープロセスを改善すれば
見逃し不具合の減少を期待できる

変数の操作と追加に漏れはないか

例外処理は抜けていないか

コードの実行順序は正しいか

考察 不具合の種類とプロセス

RQ2：見逃し不具合の種類ごとにレビュープロセスに違いがあるか

コメント数, レビュー時間, コード1行あたりのレビュー時間全てにおいて, 差は確認できなかった

考察 パッチ投稿者が自身でレビューしていたコミット

不具合混入コミットの約12%を占めており，無視できるコミットではないと判断

種類	件数
リソース	25
チェック	36
インターフェース	2
ロジック	26
機能欠陥	36

まとめと今後

まとめ

見逃し不具合のほとんどが機能的な不具合であり
特に「リソース」「チェック」「ロジック」が多い

見逃し不具合の種類とコメント数などに関係性はない

今後

~~~~~