

レビュープロセス改善のための 見逃し不具合の特徴分析

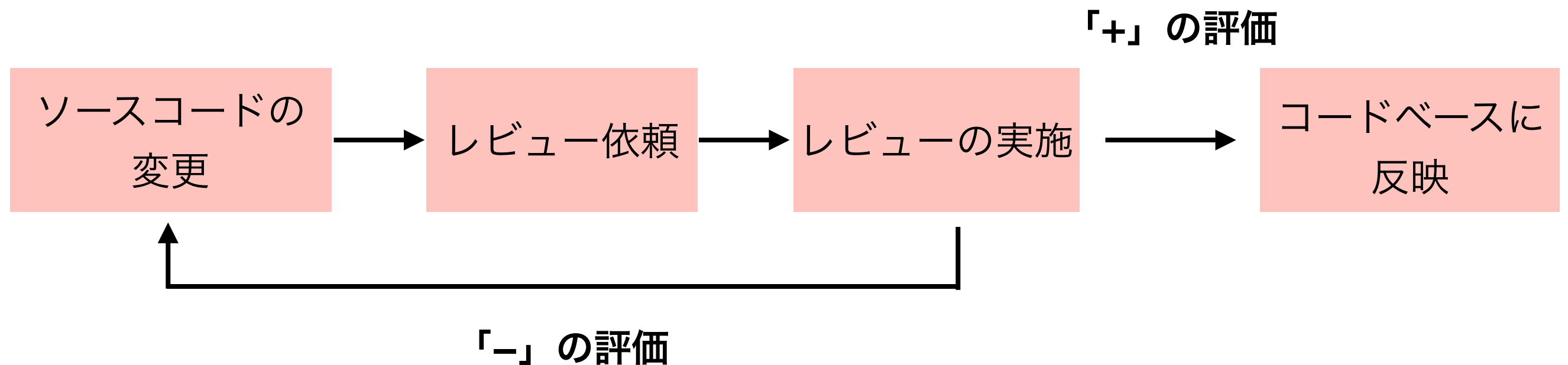
オープンソースソフトウェア工学研究室

学籍番号 60206052

岡坂 亮汰

コードレビュー

- コードレビューとは
不具合を発見するために、ソースコードの査読を行うこと
- コードレビューの流れ



背景

- コードレビューは、バージョン管理システムにソースコードが反映される前に不具合を特定する効果的な方法
- コードレビューが行われていても、バージョン管理システムにソースコードが反映された後に 不具合が見つかることがある

本研究では、コードレビューで見つけることができなかった不具合を**見逃し不具合**と呼ぶ

コードレビューに関する研究

– コードレビューで発見された不具合の分類^[1]

1, 進化性の不具合

- ソースコードの理解を困難にする不具合

2, 機能的な不具合

- ソースコードを実行したときにシステム障害を引き起こす可能性のある不具合

[1] 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

コードレビューに関する研究

コードレビューで発見された不具合の分類

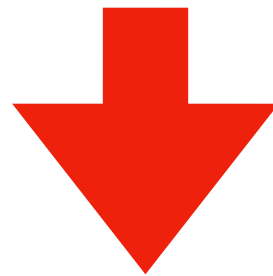
企業と学生が行ったコードレビューを観察した結果、
いずれも70%以上が進化性の不具合

	企業が行なったレビュー		学生が行なったレビュー	
進化性の不具合	276	71.1%	287	77.4%
機能的な不具合	83	21.4%	49	13.2%
偽陽性	29	7.5%	35	9.4%

発見された不具合の分布

既存研究の問題点

- コードレビューでの見逃し不具合の種類が明らかにされていない



見逃されやすい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

– 目的

レビュープロセス改善のために、見逃し不具合の特徴を明らかにすること

– アプローチ

1. 見逃し不具合の種類と分布を調査
2. 見逃し不具合の種類とコードレビュープロセスの関係を分析

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

- 見逃されやすい不具合の検出を行えるようなレビュープロセスの改善ができる

RQ2: 見逃し不具合の種類によって、コードレビュープロセスに違いはあるか

- 不具合が見逃される原因が明らかとなり、不具合混入防止のためのレビュープロセスの改善ができる

データセット

- 対象プロジェクト：Mozillaプロジェクト
- 対象期間：2015/10/01 ~ 2016/9/30

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

データセット

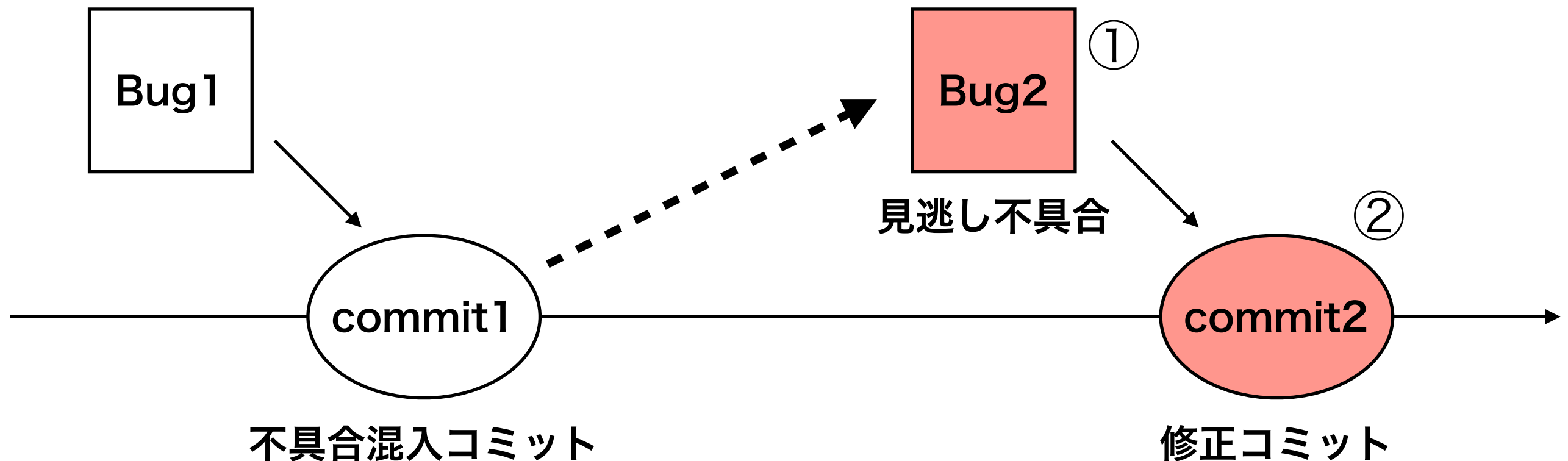
分析方法

RQ1：見逃し不具合はどのように分類されるか

- 見逃し不具合に関する以下の2つを目視して分類を行う

① Bugzillaでの議論

② Githubに投稿されている修正コミットの修正内容



不具合の種類

	不具合の種類	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
	インターフェース	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
	機能性向上	新機能の実装やパフォーマンスの改善
	機能欠陥	機能不足あるいは正しく実装されていない

不具合の種類

分析方法

RQ2：見逃し不具合の種類によって、コードレビュープロセスに違いはあるか

- 不具合混入コミットの各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間に有意差があるかどうか検定
- Kruskal-Wallis検定を用いて有意差検定
- Kruskal-Wallis検定で有意差が存在することが判明した場合、各見逃し不具合の種類において多重比較を行う

コードレビューメトリクス

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	レビュー時間を変更コード行数で割った時間

分析結果

– RQ1：見逃し不具合はどのように分類されるか

見逃し不具合を進化性の不具合と機能的な不具合に分類した結果、見逃し不具合のほとんどが**機能的な不具合**であることがわかった

	不具合の件数	割合
進化性の不具合	5	1.20%
機能的な不具合	411	98.80%

進化性の不具合と機能的な不具合の割合

分析結果

– RQ1：見逃し不具合はどのように分類されるか

「リソース」「チェック」「ロジック」の不具合が多く見られた

	不具合の種類	不具合の件数	割合
進化性の不具合	文書化	3	0.72%
	視覚的表現	1	0.24%
	構造	1	0.24%
機能的な不具合	リソース	96	23.08%
	チェック	114	27.40%
	インターフェース	9	2.16%
	ロジック	86	20.67%
	機能性向上	31	7.45%
	機能欠陥	75	18.03%

見逃し不具合の種類の件数と割合

分析結果

- RQ2：見逃し不具合の種類によって、コードレビュープロセスに違いはあるか

- ・コメント数

P値 > 0.05 であるので、**有意差はない**

メトリクス	不具合の種類	N	H値	P値
コメント数	リソース	231	1.69	0.79
	チェック	273		
	インターフェース	32		
	ロジック	270		
	機能欠陥	494		

コメント数におけるKruskal-Wallis検定の結果

分析結果

- RQ2：見逃し不具合の種類によって，コードレビュープロセスに違いはあるか

- ・レビュー時間

P値 > 0.05 であるので，**有意差はない**

メトリクス	不具合の種類	N	H値	P値
レビュー時間	リソース	172	3.13	0.54
	チェック	204		
	インターフェース	27		
	ロジック	204		
	機能欠陥	320		

レビュー時間におけるKruskal-Wallis検定の結果

分析結果

- RQ2：見逃し不具合の種類によって，コードレビュープロセスに違いはあるか
- ・コード1行あたりのレビュー時間

P値 > 0.05 であるので，**有意差はない**

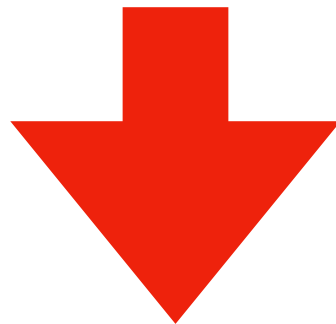
メトリクス	不具合の種類	N	H値	P値
コード1行あたりの レビュー時間	リソース	172	1.80	0.77
	チェック	204		
	インターフェース	27		
	ロジック	204		
	機能欠陥	320		

コード1行あたりのレビュー時間におけるKruskal-Wallis検定の結果

考察

– RQ1：見逃し不具合はどのように分類されるか

- ・ 進化性の不具合はわずか1.2%しか見逃されていない
- ・ 先行研究では、コードレビューで発見される不具合は70%以上が進化性の不具合であることがわかっている



コードレビューにおいて、進化性の不具合の多くは発見できると考えられる

考察

– RQ1：見逃し不具合はどのように分類されるか

見逃し不具合の件数が多かった「リソース」「チェック」「ロジック」の不具合の原因を調査

見逃し不具合の種類	見逃し不具合の原因	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

見逃し不具合の原因

考察

– RQ1：見逃し不具合はどのように分類されるか

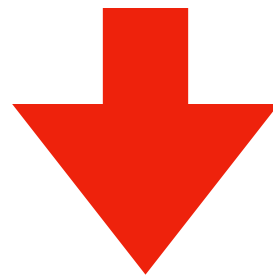
以下の観点に注意し，コードレビュープロセスの改善を行えば見逃し不具合の大幅な減少を期待できる

- 変数の操作と追加の漏れはないか
- 例外処理は抜けていないか
- 実行の順序は正しいかどうか

考察

- RQ2：見逃し不具合の種類によって、コードレビュープロセスに違いはあるか

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のいずれのメトリクスにおいても, 差は見られなかった



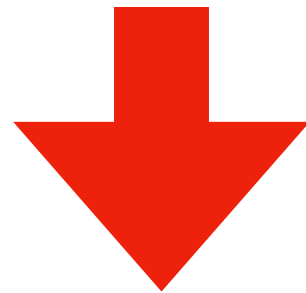
メトリクス間に関係性が存在する可能性

「コメント数が少ないコミットはレビューで厳格にチェックされている」
「レビュー時間が短いコミットはコメント数が多い」

考察

– パッチ投稿者が自身でレビューを行っていたコミット

対象の不具合混入コミットの約12%を占めることから、
無視できるコミットではないと判断



パッチ投稿者が自身でレビューを行っていたコミットが
どの種類の不具合を見逃しているか調査

考察

– パッチ投稿者が自身でレビューを行っていたコミット

「チェック」と「機能欠陥」の不具合が多く見逃されている

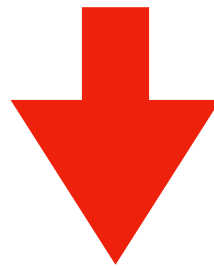
見逃し不具合の種類	件数
リソース	25
チェック	36
インターフェース	2
ロジック	26
機能欠陥	36

パッチ投稿者がレビューを行ったコミットと見逃し不具合の関係

考察

– パッチ投稿者が自身でレビューを行っていたコミット

「機能欠陥」の不具合は、機能不足など重度の症状



パッチ投稿者が行うレビューにはリスクが伴う

レビュープロセス改善案

少なくとも1人のレビュー者を指名する
レビュープロセスを確立

まとめ

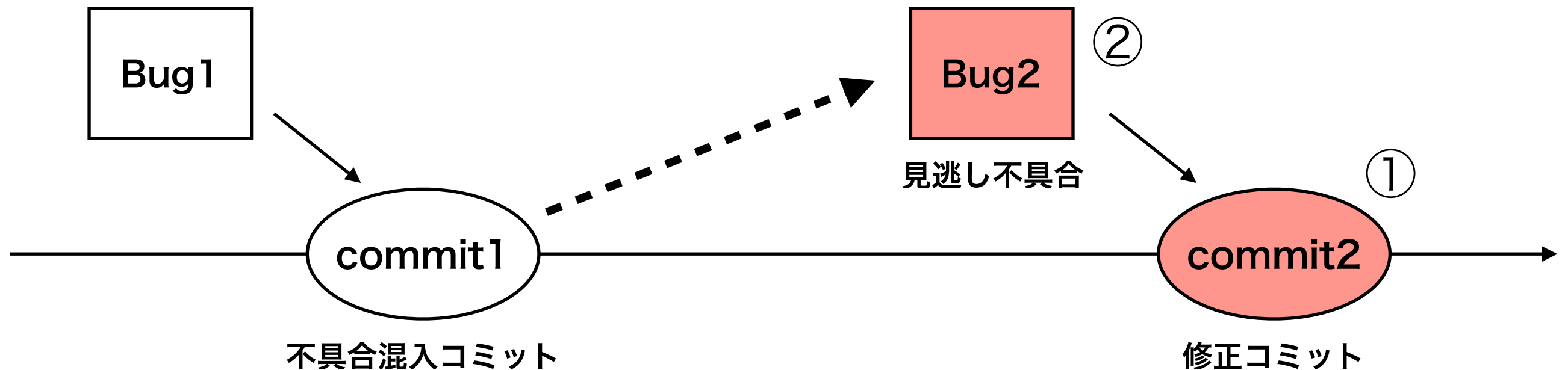
- 見逃し不具合のほとんどが機能的な不具合である
- 機能的な不具合では, 「リソース」「チェック」「ロジック」の不具合が見逃されやすい傾向にある
- 見逃し不具合の種類とコメント数, レビュー時間, コード1行あたりのレビュー時間との関係性はない
- パッチ投稿者が自身でレビューを行っているコミットは, 「チェック」「機能欠陥」の不具合を多く見逃している

データセットの取得

- ① バージョン管理システムからコミットを取得
- ② コミットのコミットメッセージから不具合IDを取得

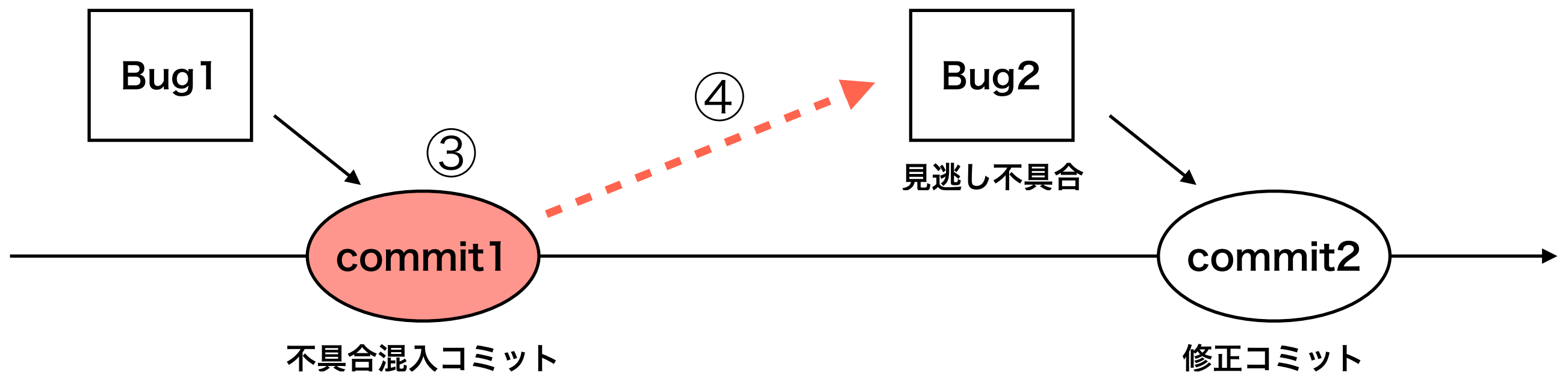
※①のコミットのメッセージに不具合IDが書いてあれば,

①のコミットを修正コミット, ②の不具合を見逃し不具合とする



データセットの取得

- ③ SZZアルゴリズム^[3]で見逃し不具合が混入したコミットを特定
- ④ 見逃し不具合と不具合混入コミットを結びつけ



Kruskal-Wallis検定

- 正規分布に従わない3つ以上のグループ間の有意差を調べる検定
- Kruskal-Wallis検定では、どのグループ間で有意差が存在するかまでは明らかにできない