

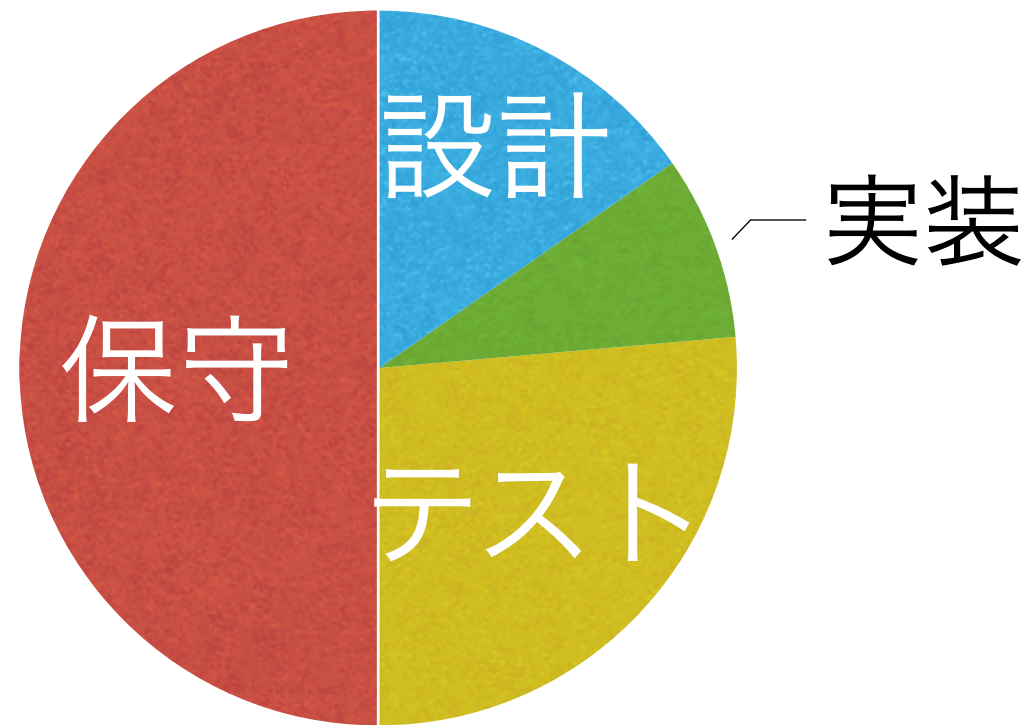
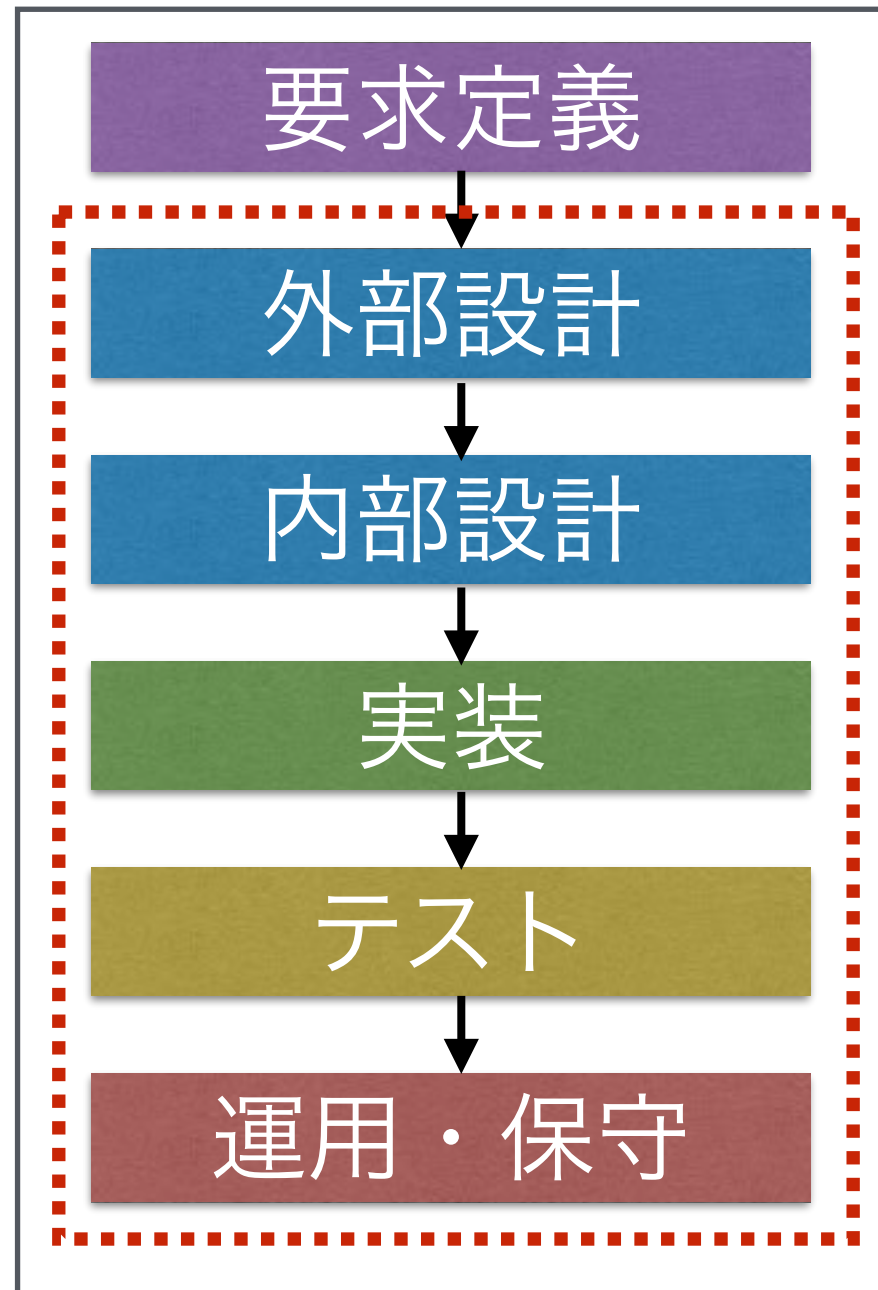
ソフトウェアレビューにおける 見逃し欠陥の削減手法に関する研究

60180081 築山文香

2017/10/31

ソフトウェア開発のライフサイクル

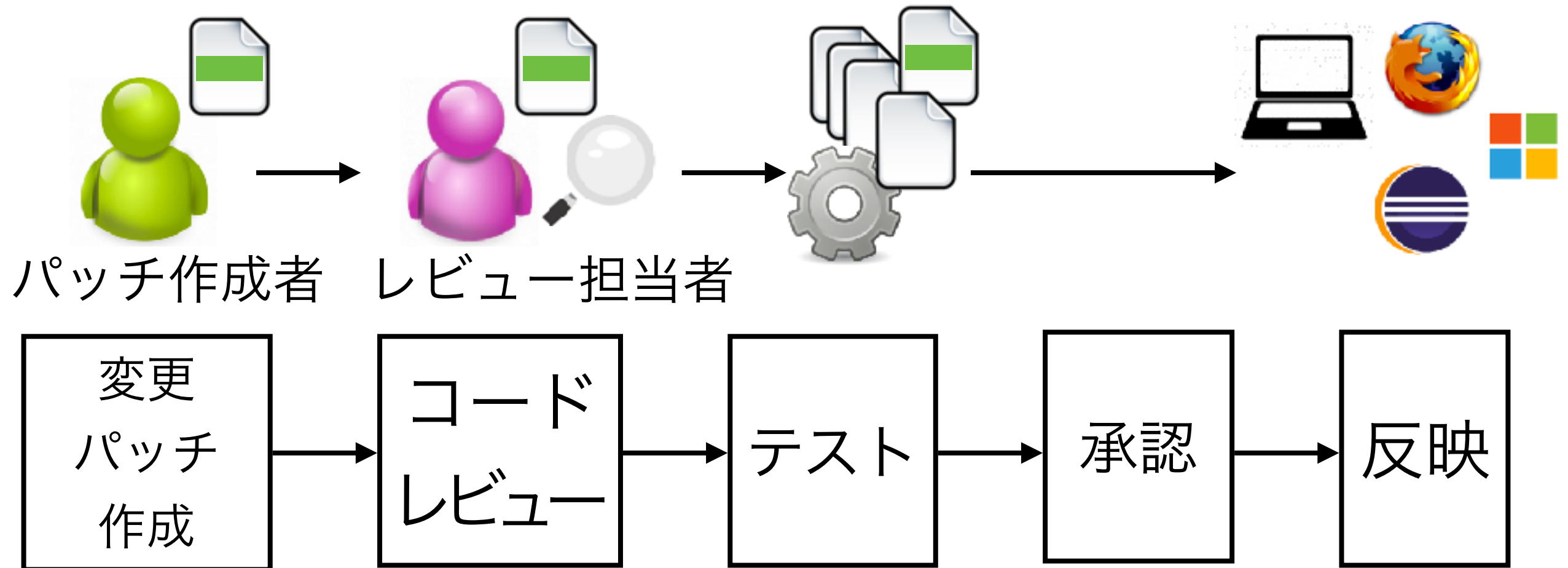
開発者のソフトウェア開発時間の内訳



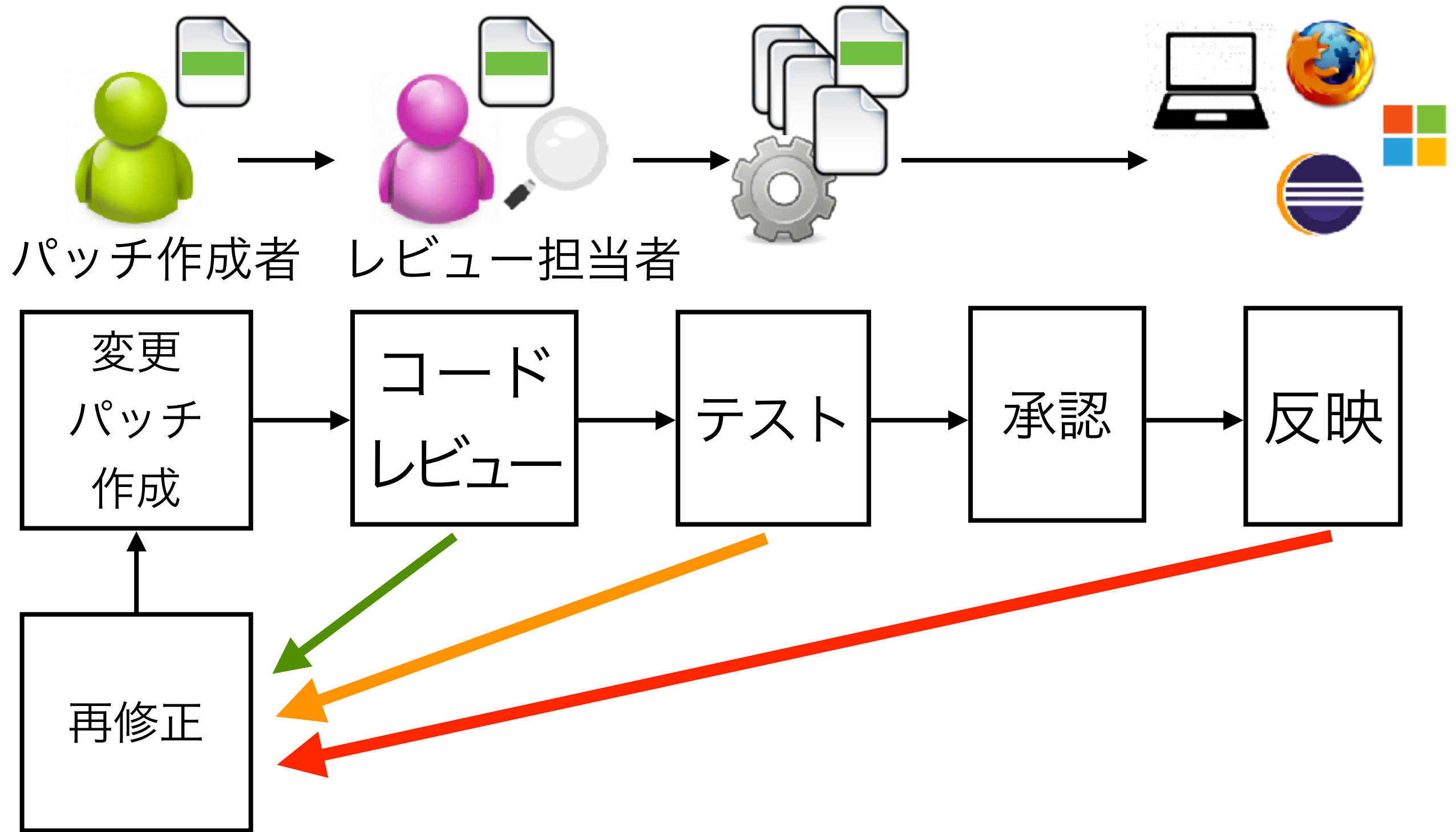
出典：Mint（経営情報処理学会）著
『ソフトウェア開発の全て』
日本実業出版社

開発者の開発時間の約半分が
保守に使われている[1]

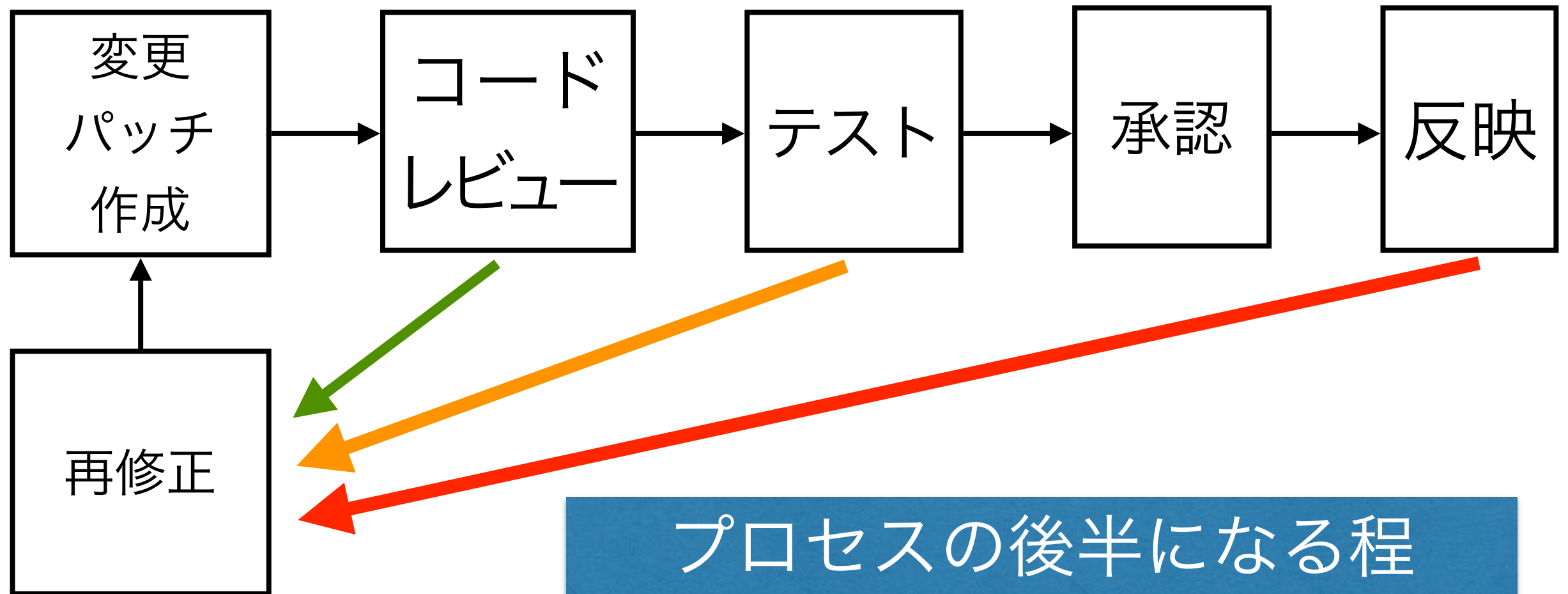
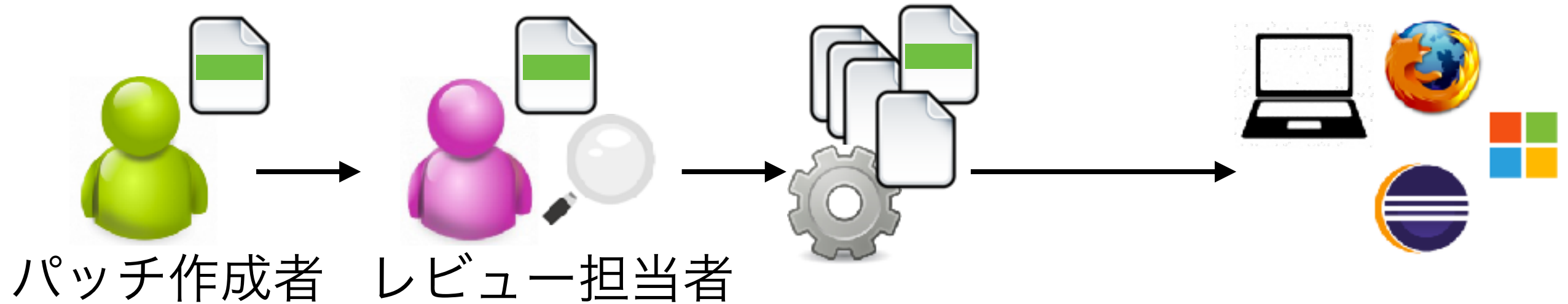
保守のプロセス



保守のプロセス

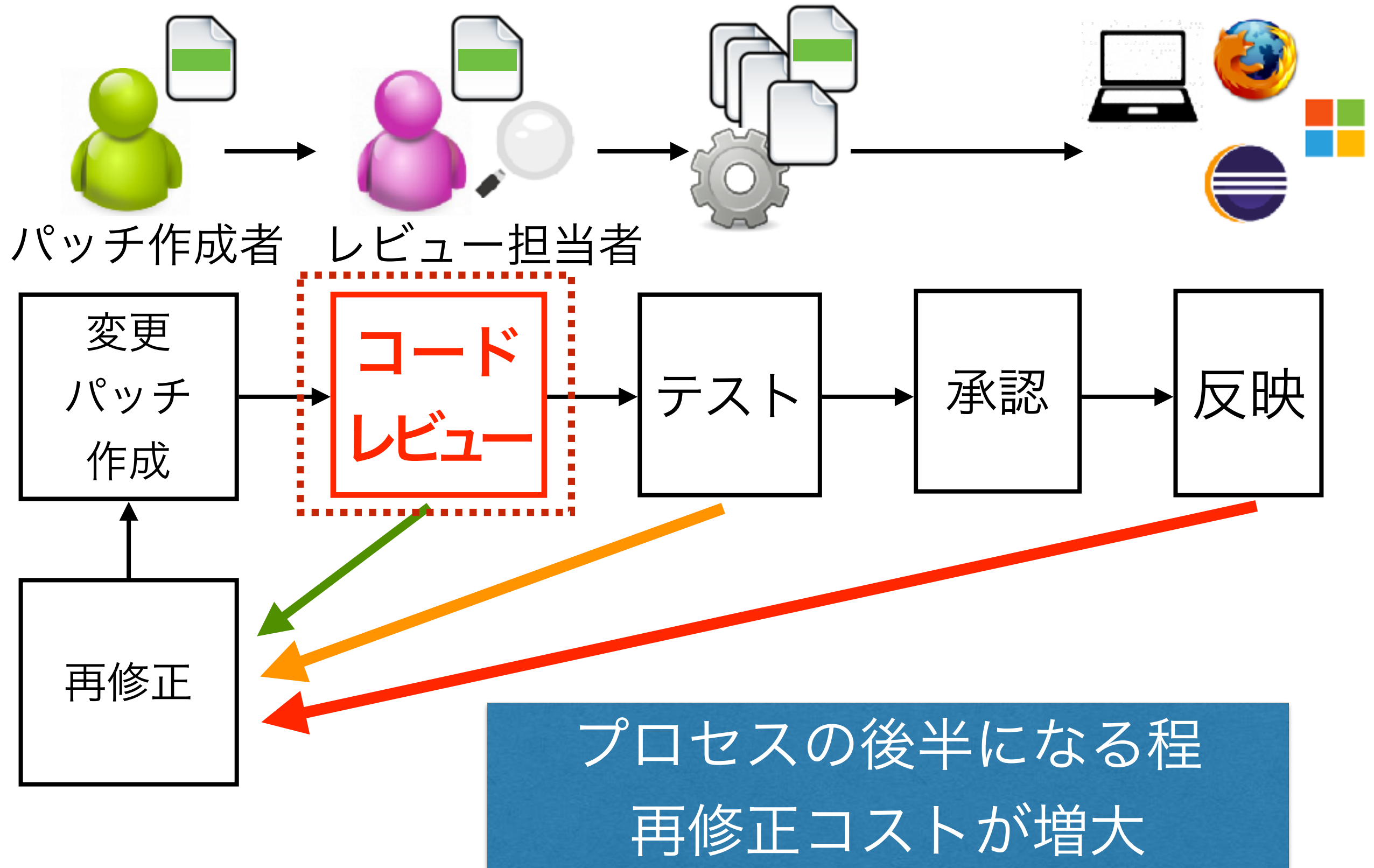


保守のプロセス



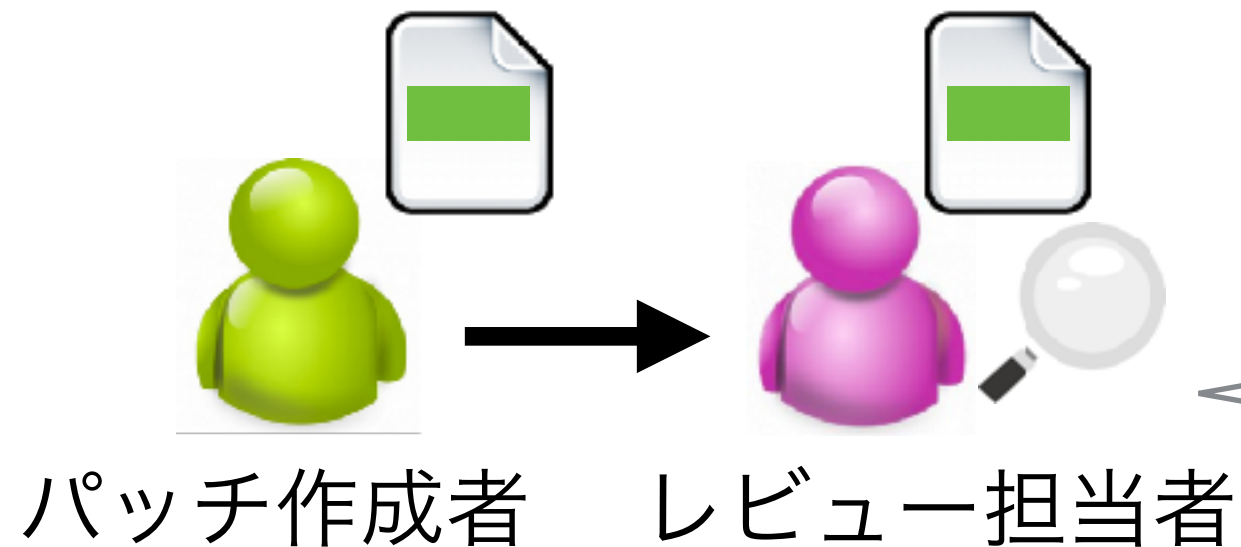
プロセスの後半になる程
再修正コストが増大

保守のプロセス



コードレビュー

- ◎ 欠陥を未然に防ぐための方法の一つ[2]



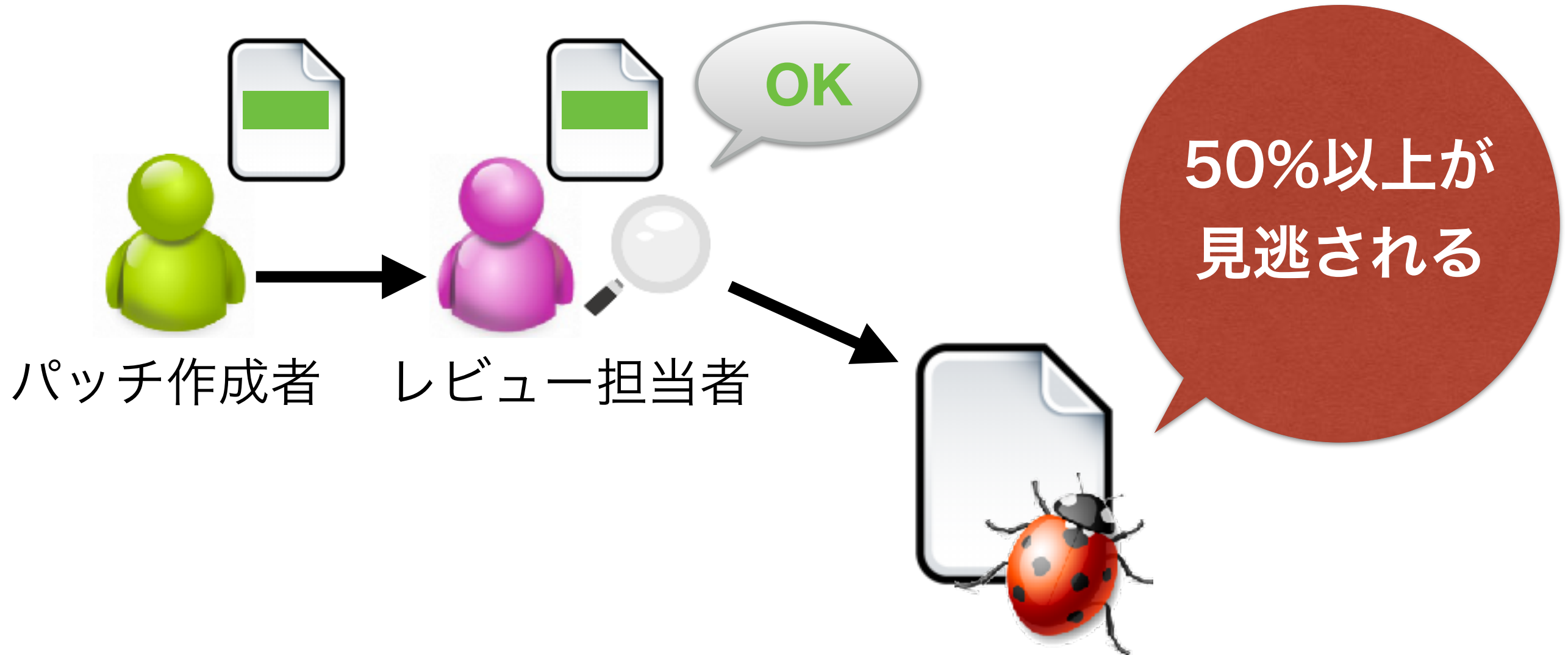
人が目視でソースコードを検証

- ▶ コードの論理的な構造
- ▶ 他のファイルへの影響
- ▶ コーディング規約の確認

欠陥を早期に発見することで
ソフトウェア開発のコストを削減できる

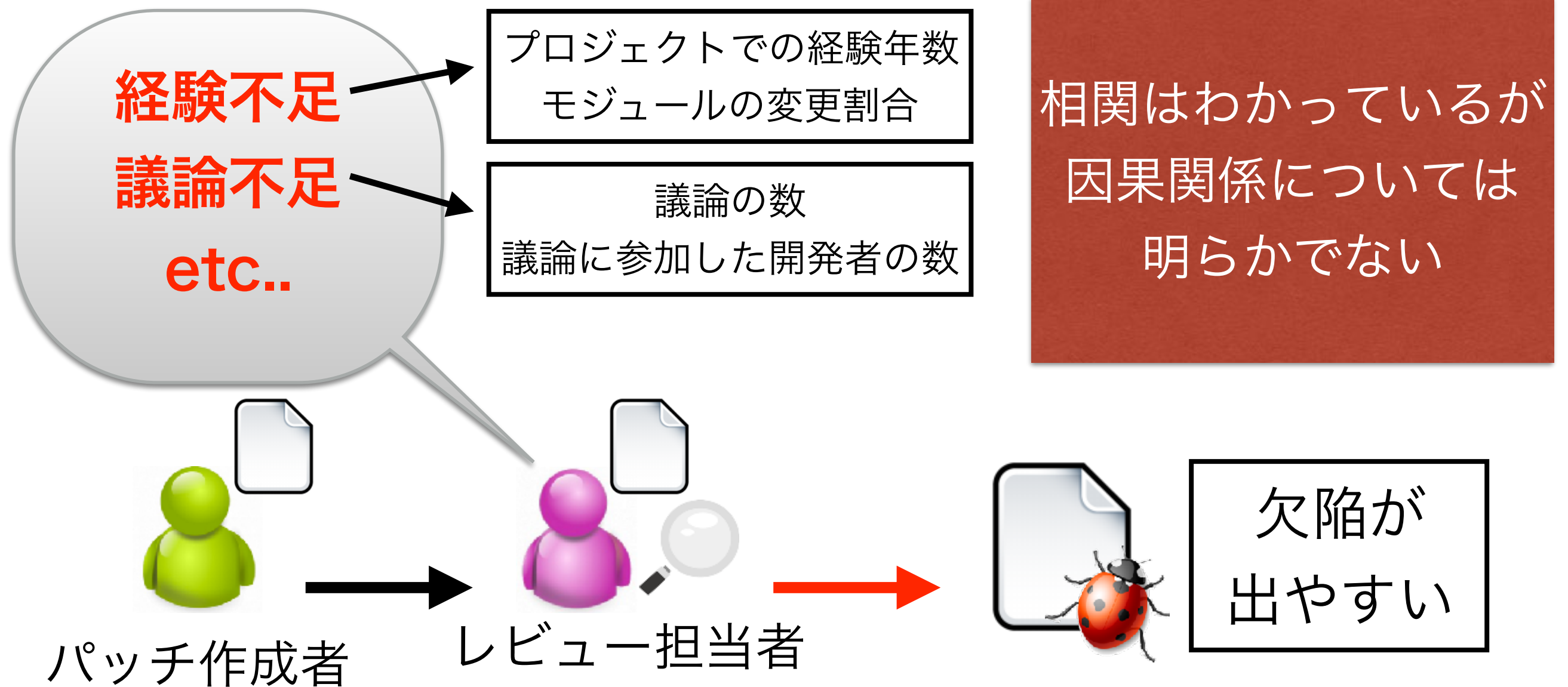
コードレビューにおける問題点

コードレビューでは半分の欠陥が見逃されている[3]



コードレビューに関する先行研究

◎ レビューが見逃される原因[4, 5]



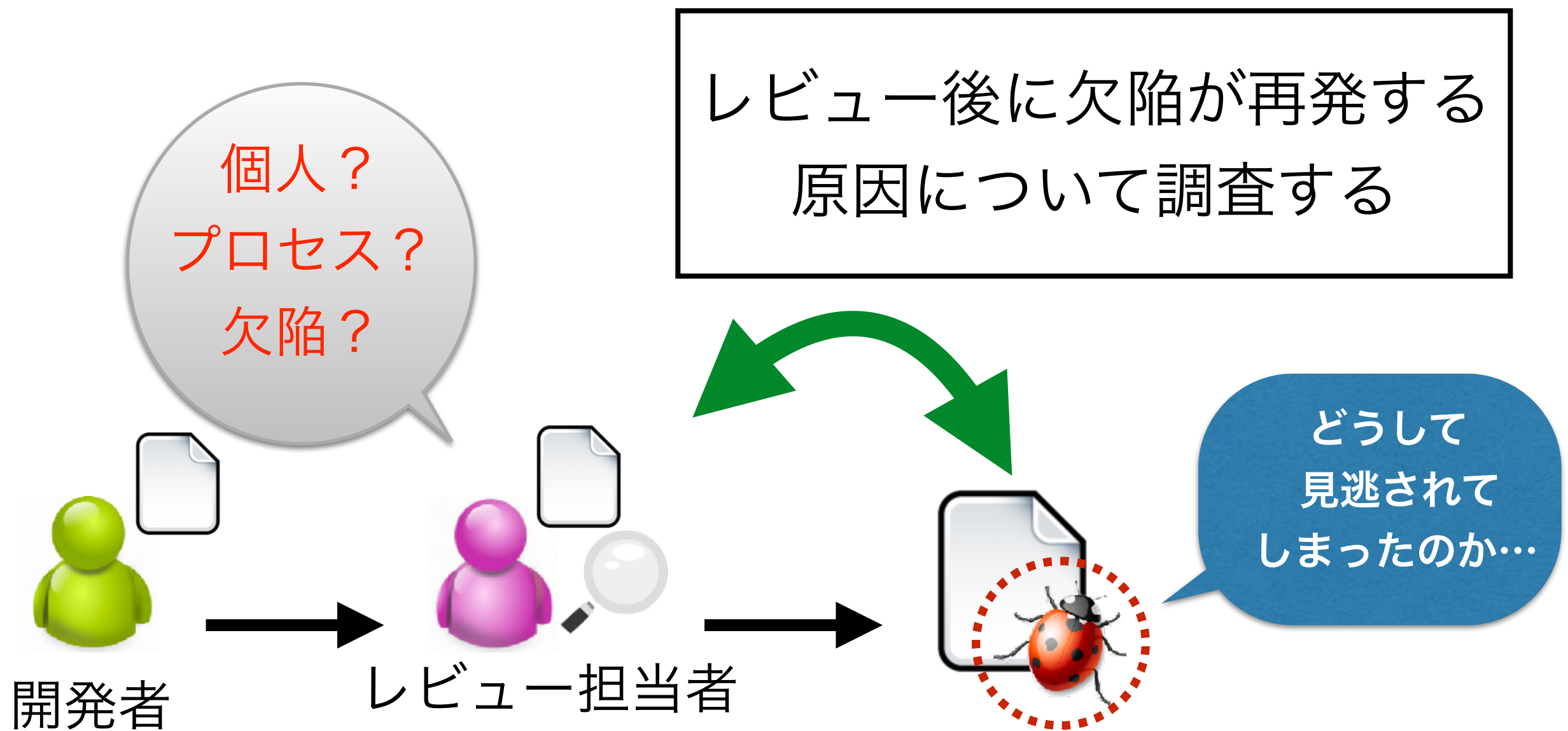
[4] Shane McIntosh, Yasutaka Kamei, Bram Adams and Ahmed E. Hassan, "The Impact of Code Review Coverage and Code Review Participation on Software Quality A Case Study of the Qt, VTK, and ITK Projects," in Proceedings of the 11th Working Conference on Mining Software Repositories(MSR), pp.192-201, 2014

[5] Patanamon Thongtanunam, Shane McIntosh, Ahmed E. Hassan and Hajimu Iida, "Investigating code review practices in defective files: an empirical study of the Qt system," in Proceedings of the 12th Working Conference on Mining Software Repositories(MSR), pp.168-179., 2015

本研究の目的

最終目標

見逃し欠陥の削減手法の構築



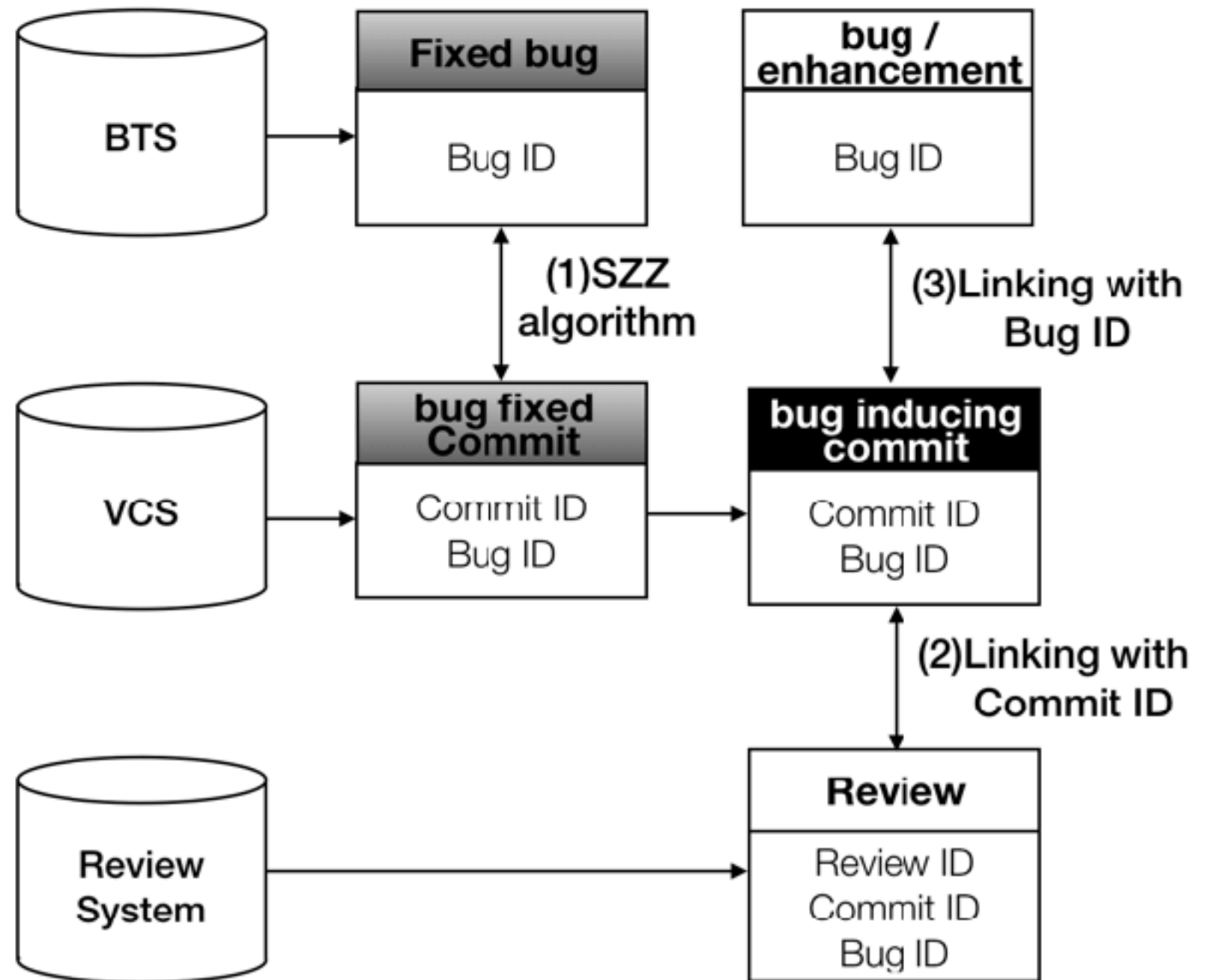
アプローチ：見逃しバグの取得

- 対象プロジェクト

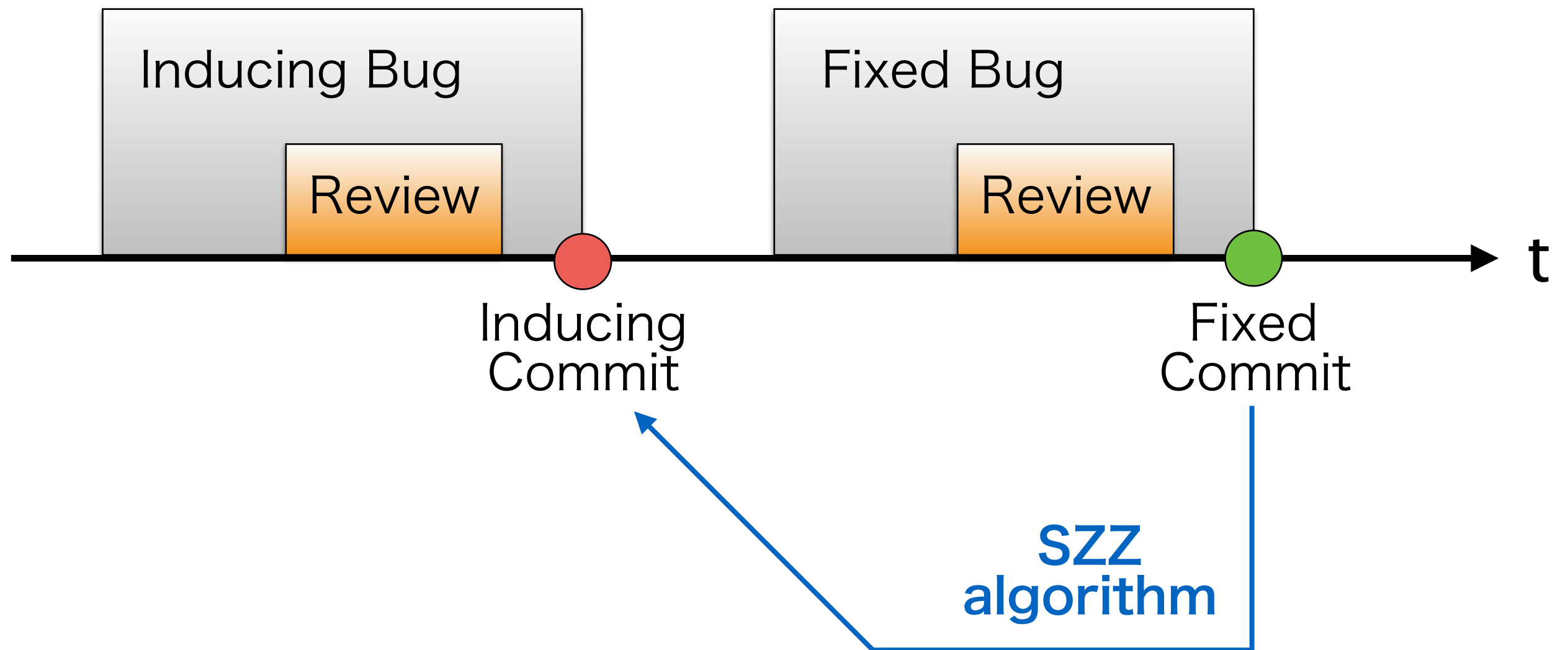
- ▶ Mozilla :
Gecko (FireFox)

- 対象リポジトリ

- ▶ Bugzilla
- ▶ Git
- ▶ ReviewBoard



時系列はこんな感じ



アプローチ：分類方法

- ◎ 本当にレビュー見逃し？

レビューミス

- 修正箇所と同じ line で不具合（レビューワーカーのケアレスミス）
- 修正箇所とは違う line, file の見落とし

パッチミス

- 修正すべきバグの内容が違う

その他

- 修正すべきバグを直したが、別の箇所で不具合が発生している
- 環境が違う, 変わった（特定の環境では動く・動いていた）

アプローチ：分類方法

- ◎ どんな欠陥が見逃されやすい？

レビューワーカーの問題

- 時間不足
- 経験不足
- 理解不足

プロセスの問題

- 参加者が少ない
- 議論不足

コードの問題

- 修正行数が多い
- 修正ファイル数が多い
- コードが複雑

見逃しバグ例 1

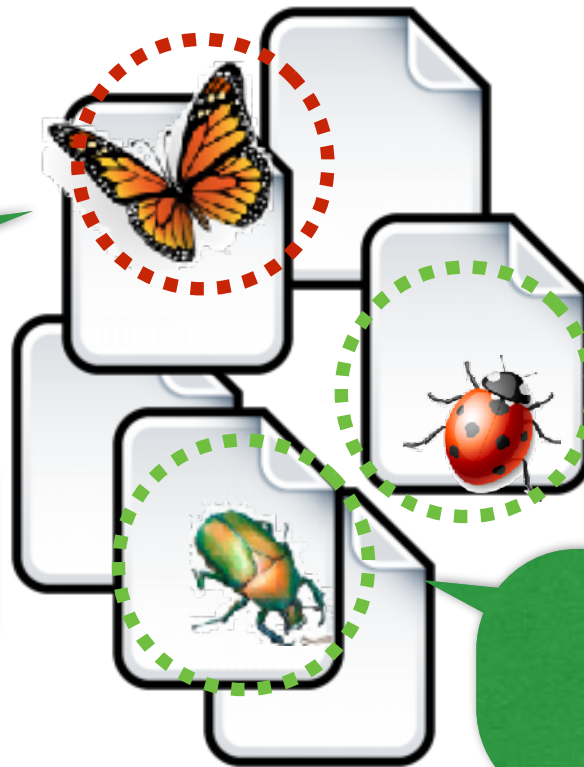
◎ 変更間違い

- ▶ 発見まで：5日
- ▶ 見逃し箇所：1行
- ▶ 追加ファイル多し
- ▶ 議論めっちゃされてる（3人：6コメント）
- ▶ 最後はバグ作成者の解説付き（@Bagzilla）

MinGW
(Windows32bit用のGNU)
ビルドしたいけど無理

これもバグ
OK..

..じゃなかったごめん
間違えたID貼り付けてた



これバグ
OK

これもバグ
OK

複数人がテスト
すればよかったと思う