

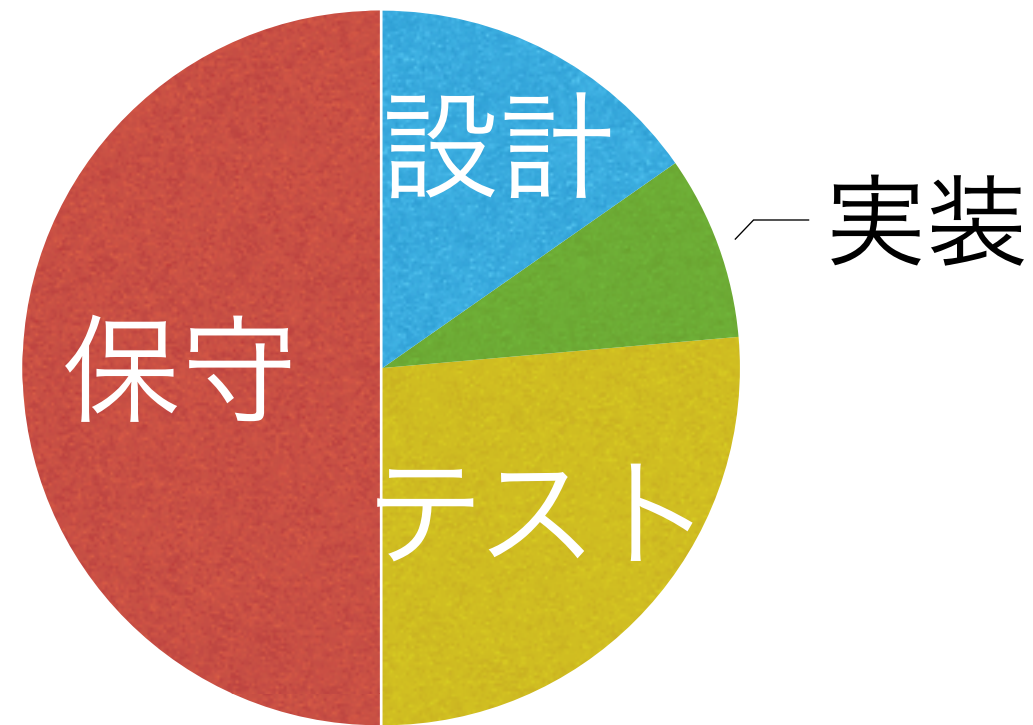
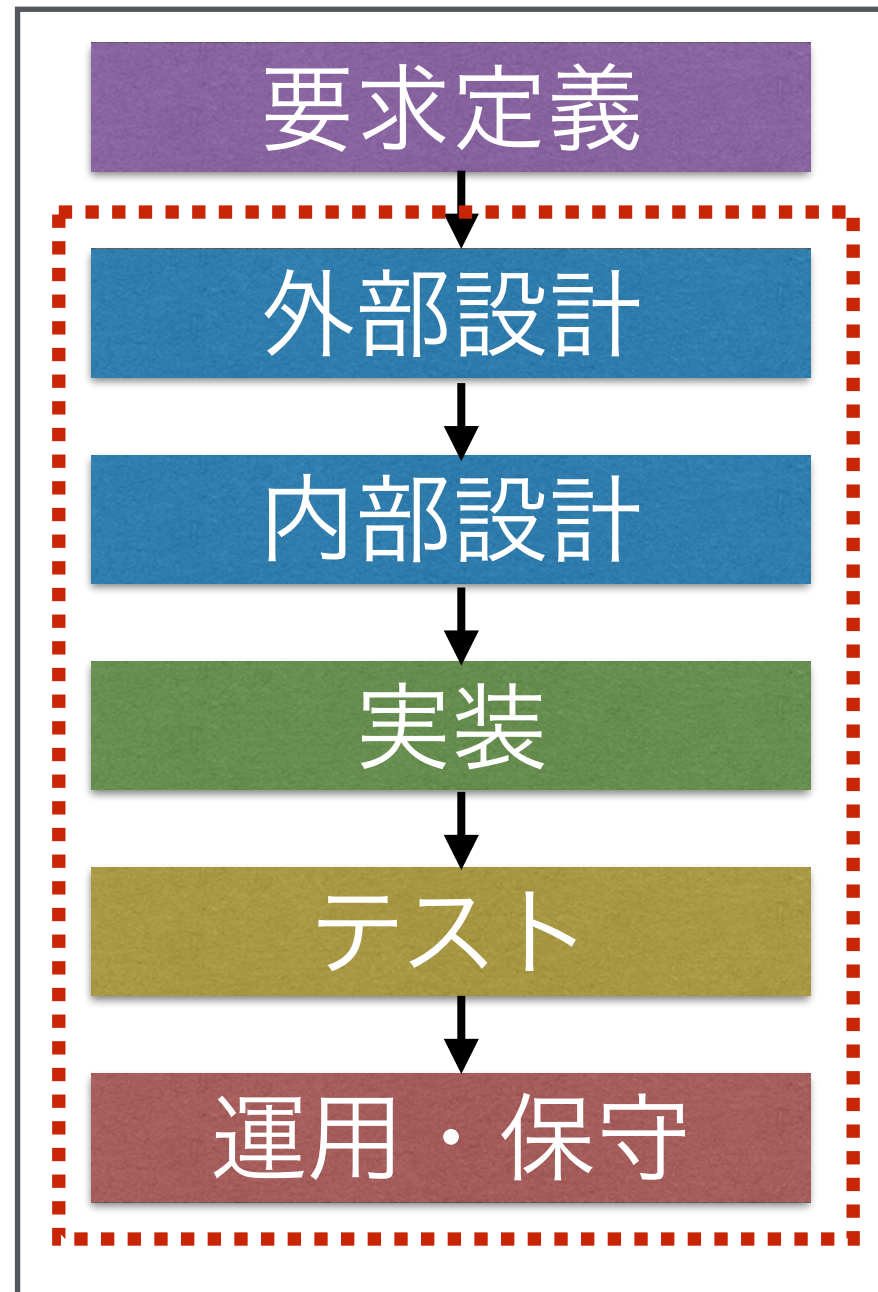
ソフトウェアレビューにおける 見逃し欠陥の削減手法に関する研究

大平研究室

60180081 築山文香

ソフトウェア開発のライフサイクル

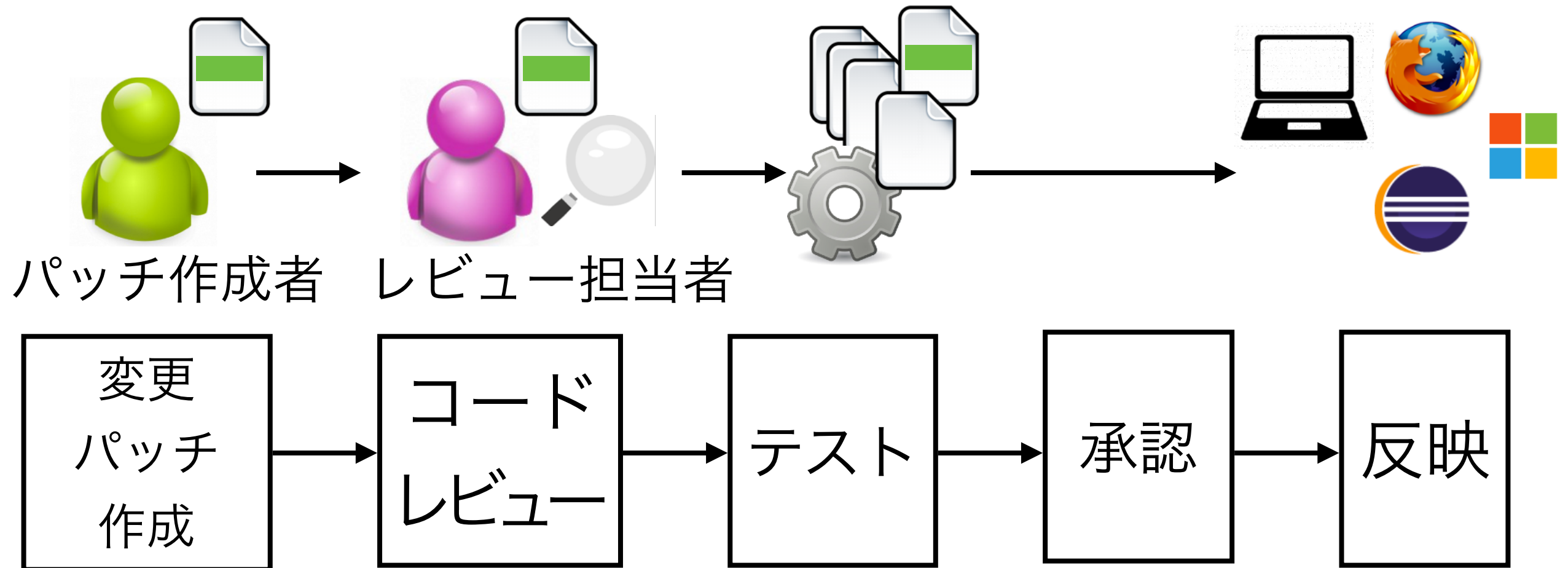
開発者のソフトウェア開発時間の内訳



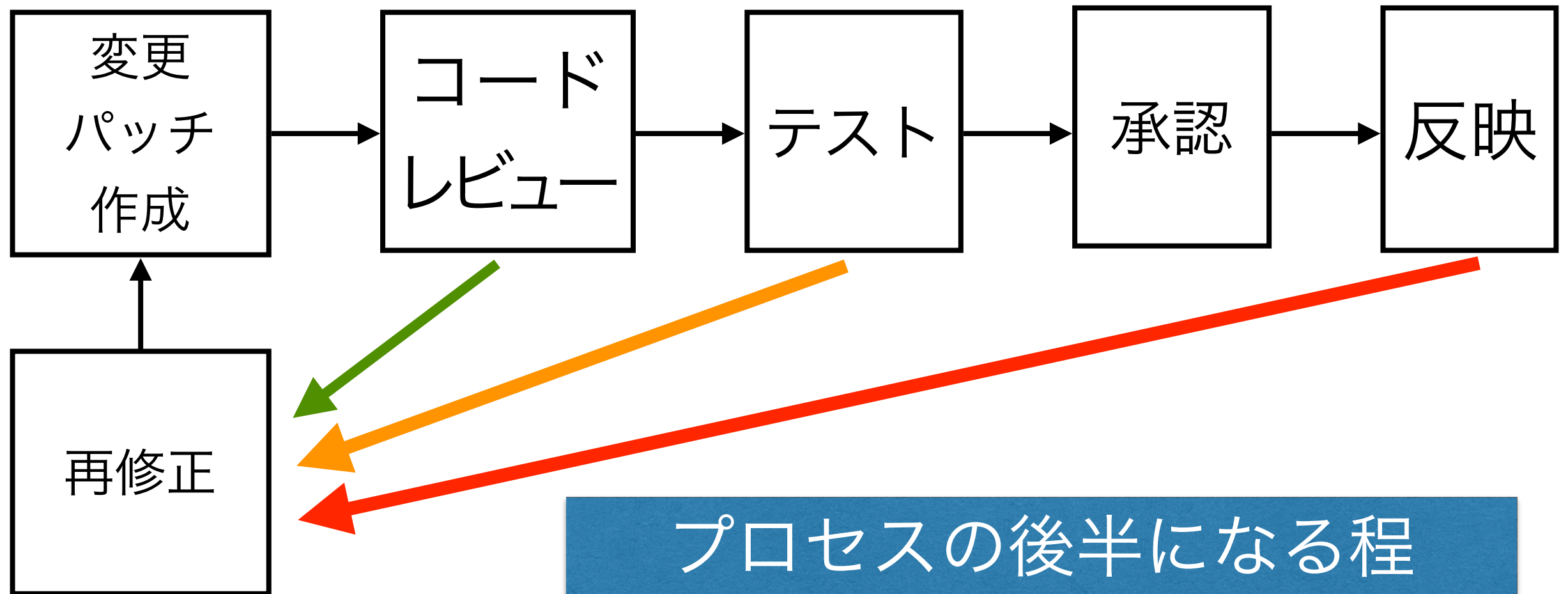
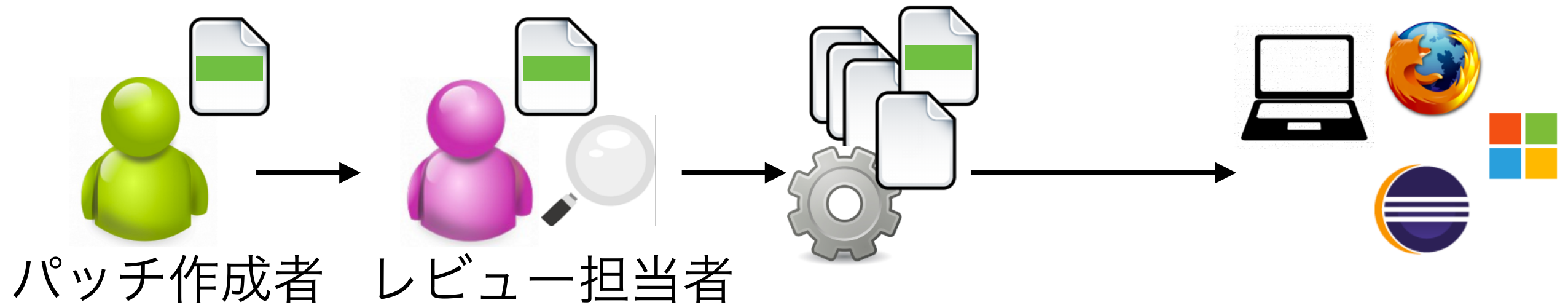
出典：Mint（経営情報処理学会）著
『ソフトウェア開発の全て』
日本実業出版社

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

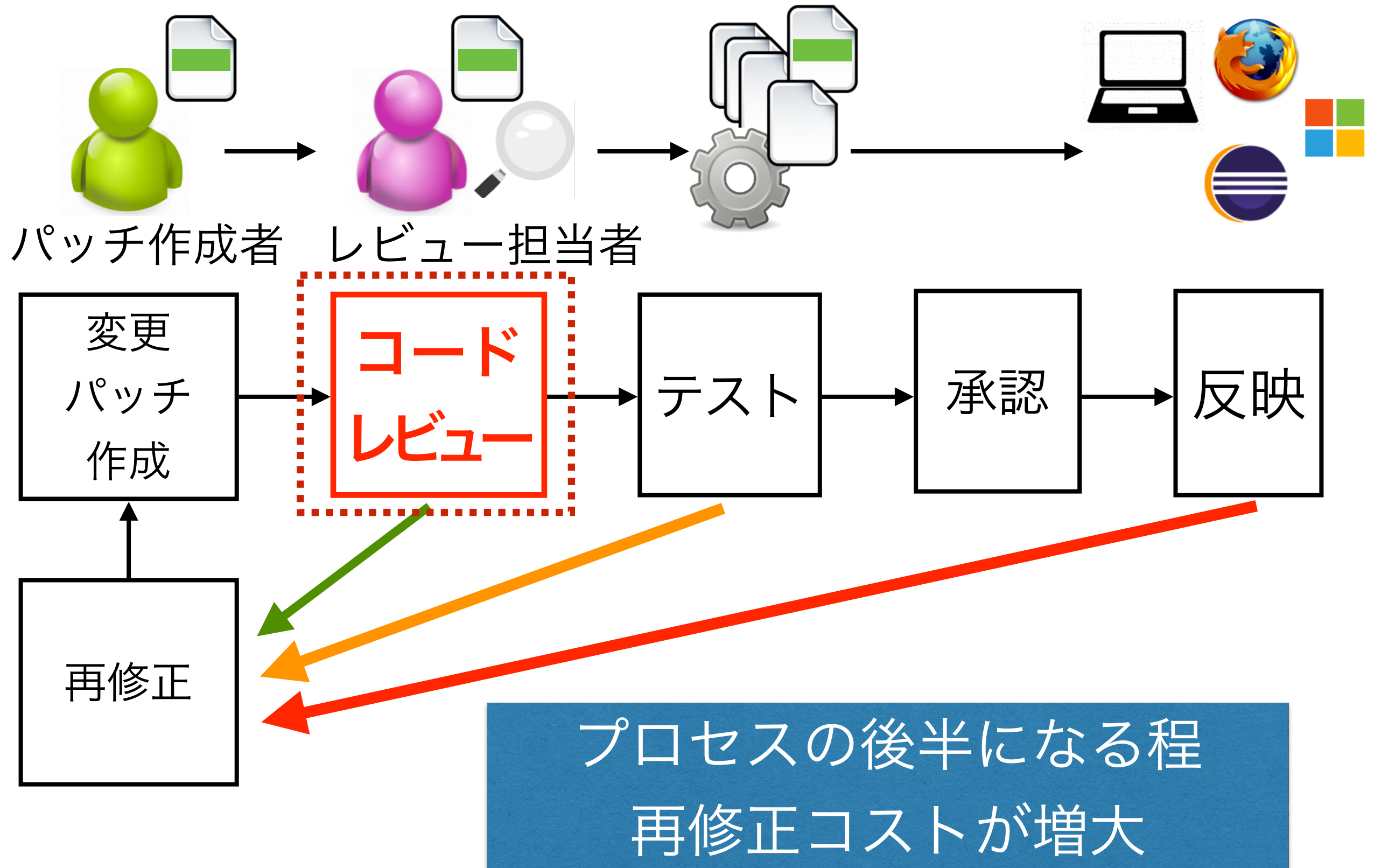


保守のプロセス



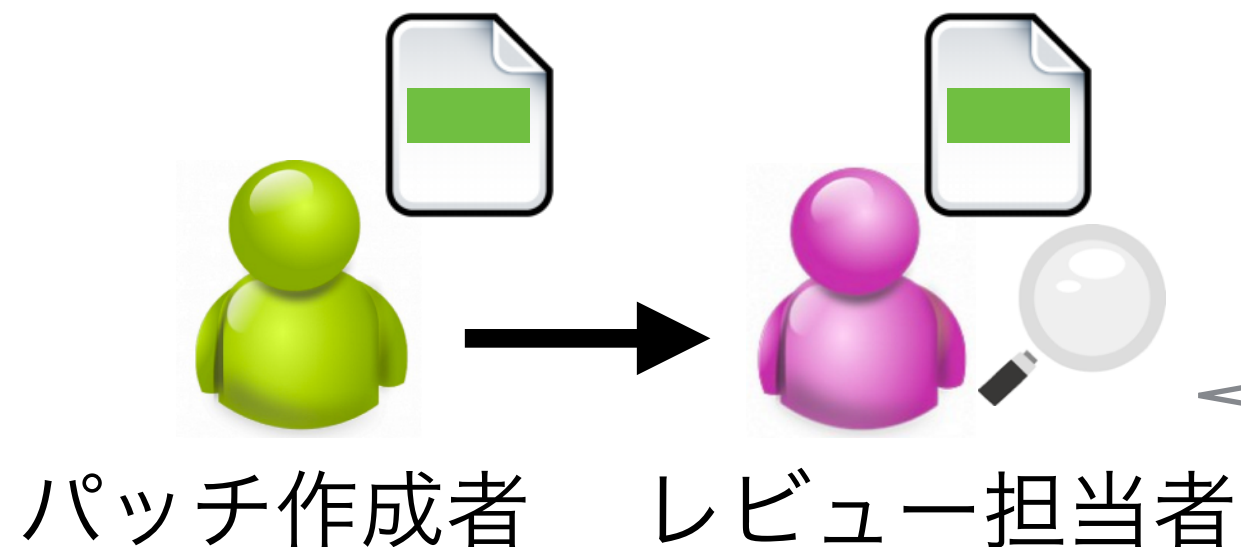
プロセスの後半になる程
再修正コストが増大

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]



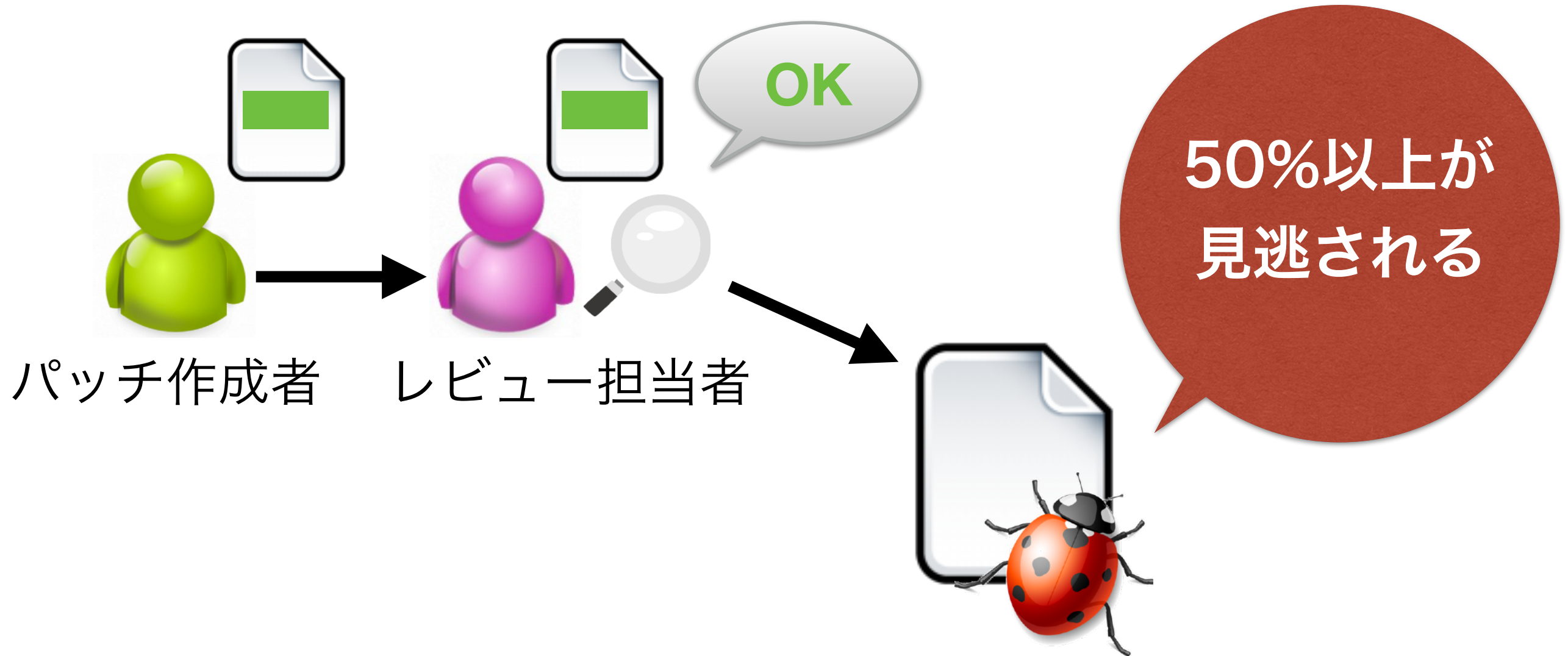
人が目視でソースコードを検証

- ▶ 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 & 保守性の向上から
ソフトウェア開発のコストを削減できる

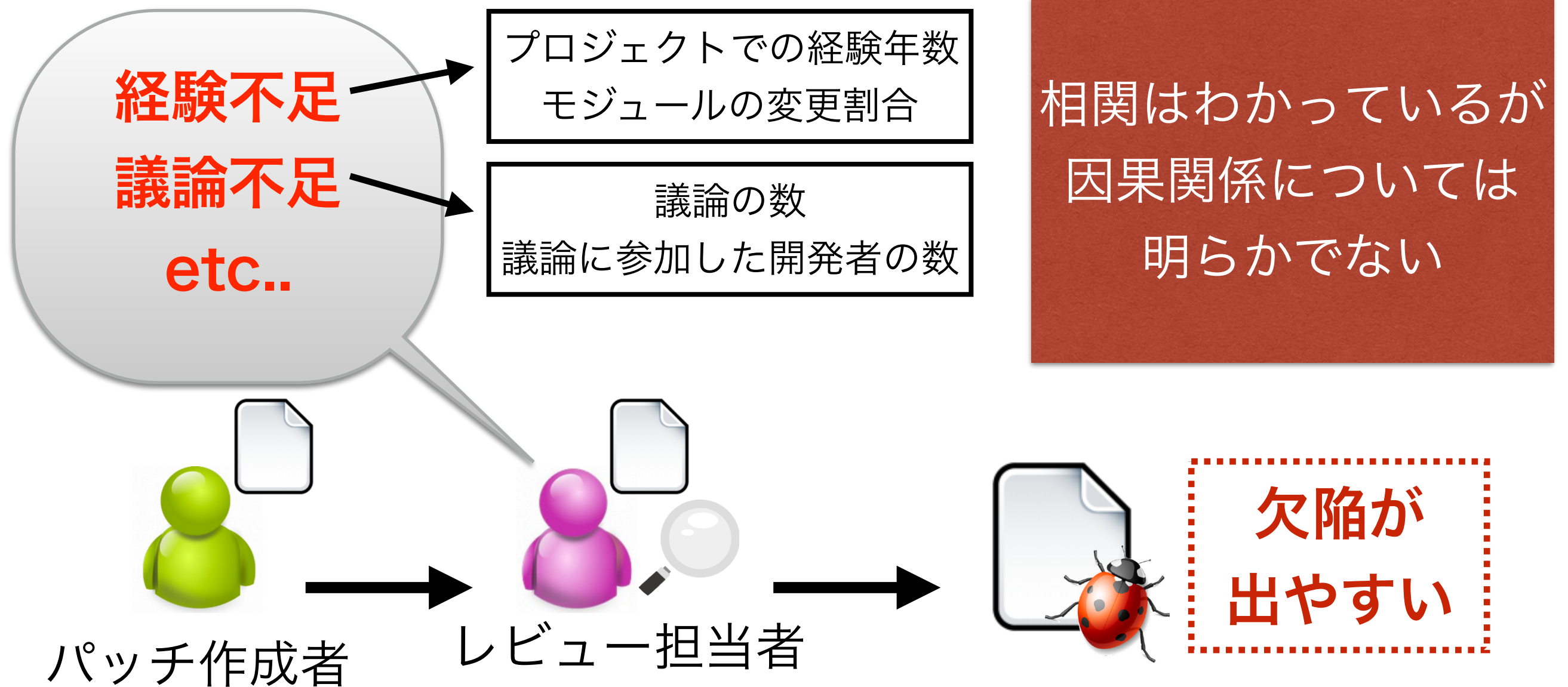
コードレビューにおける問題点

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]



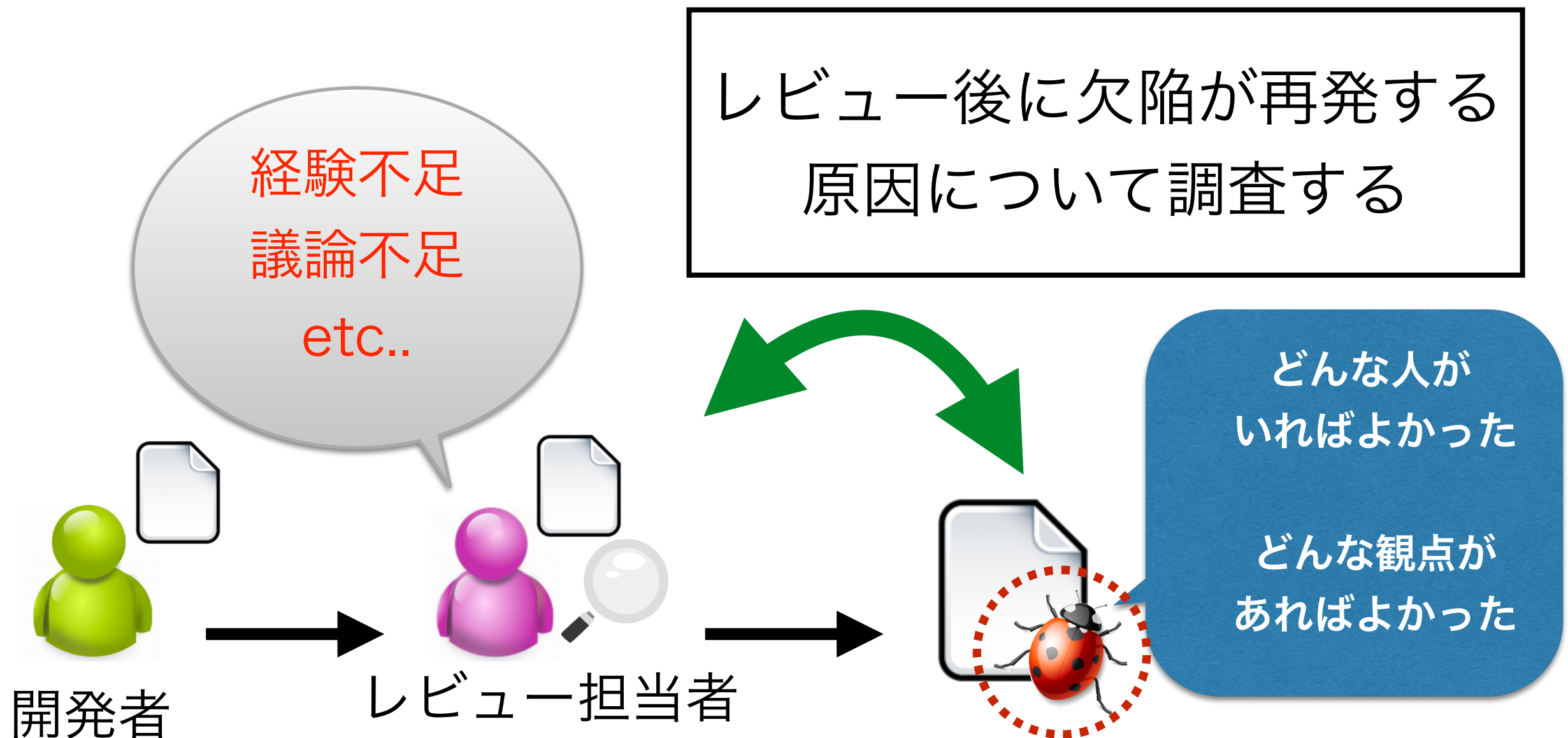
[4] Shane McIntosh, Yasutaka Kamei, Bram Adams and Ahmed E. Hassan, "The Impact of Code Review Coverage and Code Review Participation on Software Quality A Case Study of the Qt, VTK, and ITK Projects," in Proceedings of the 11th Working Conference on Mining Software Repositories(MSR), pp.192-201, 2014

[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

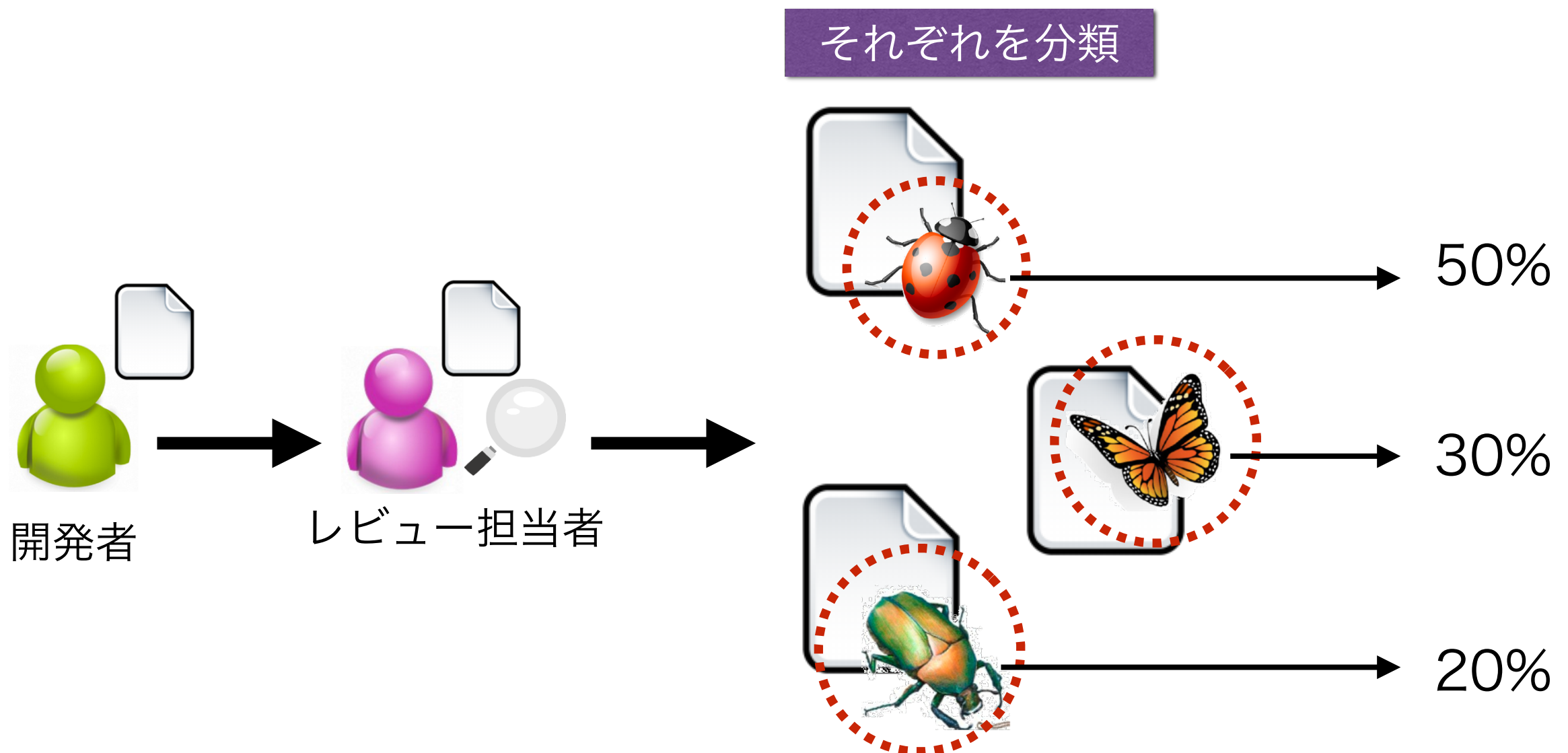
見逃し欠陥の削減手法の構築



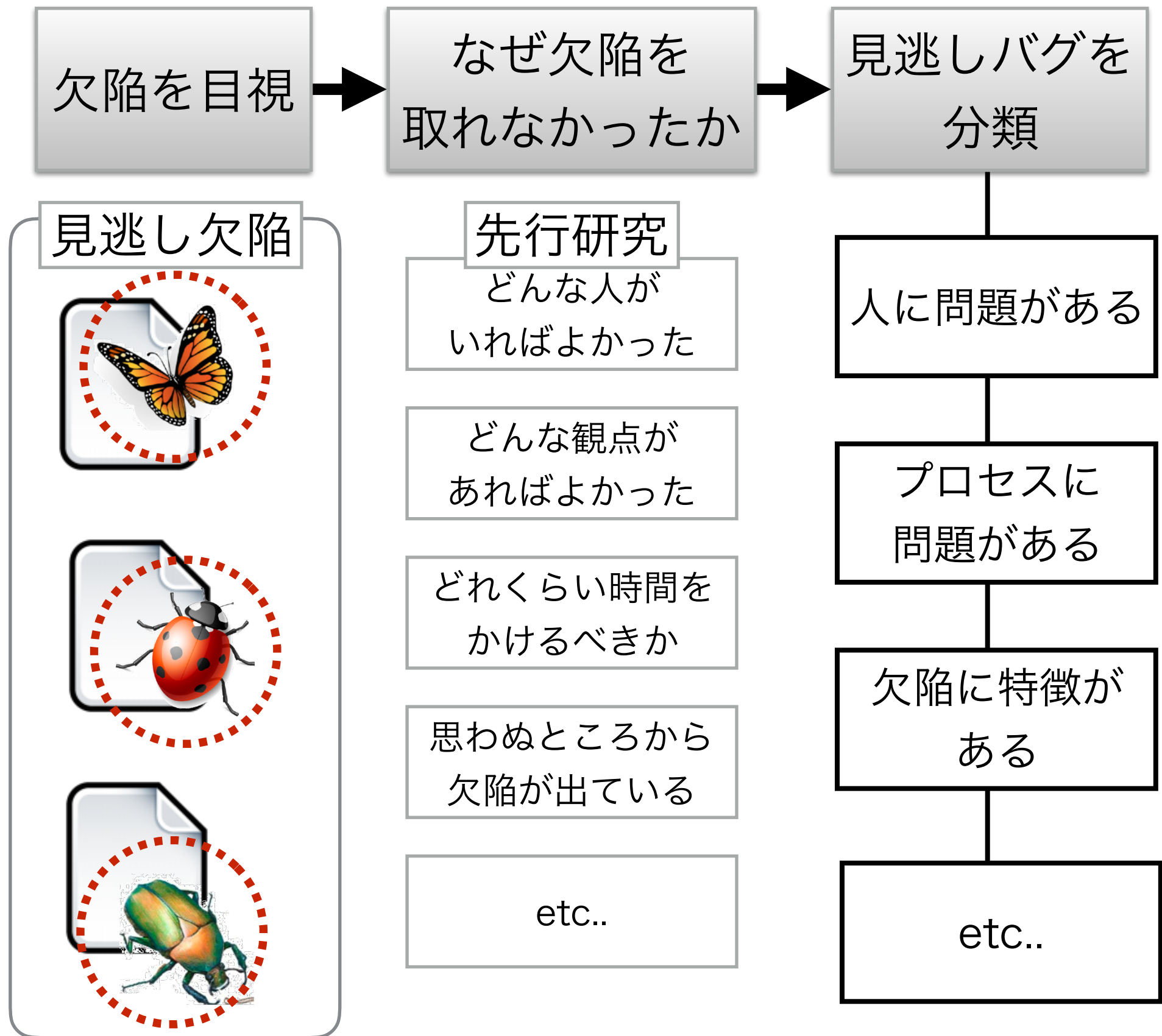
本研究の目的

コードレビューで欠陥が見逃される原因を調査する

- ▶ まず、どこに問題があるのかを明らかにする



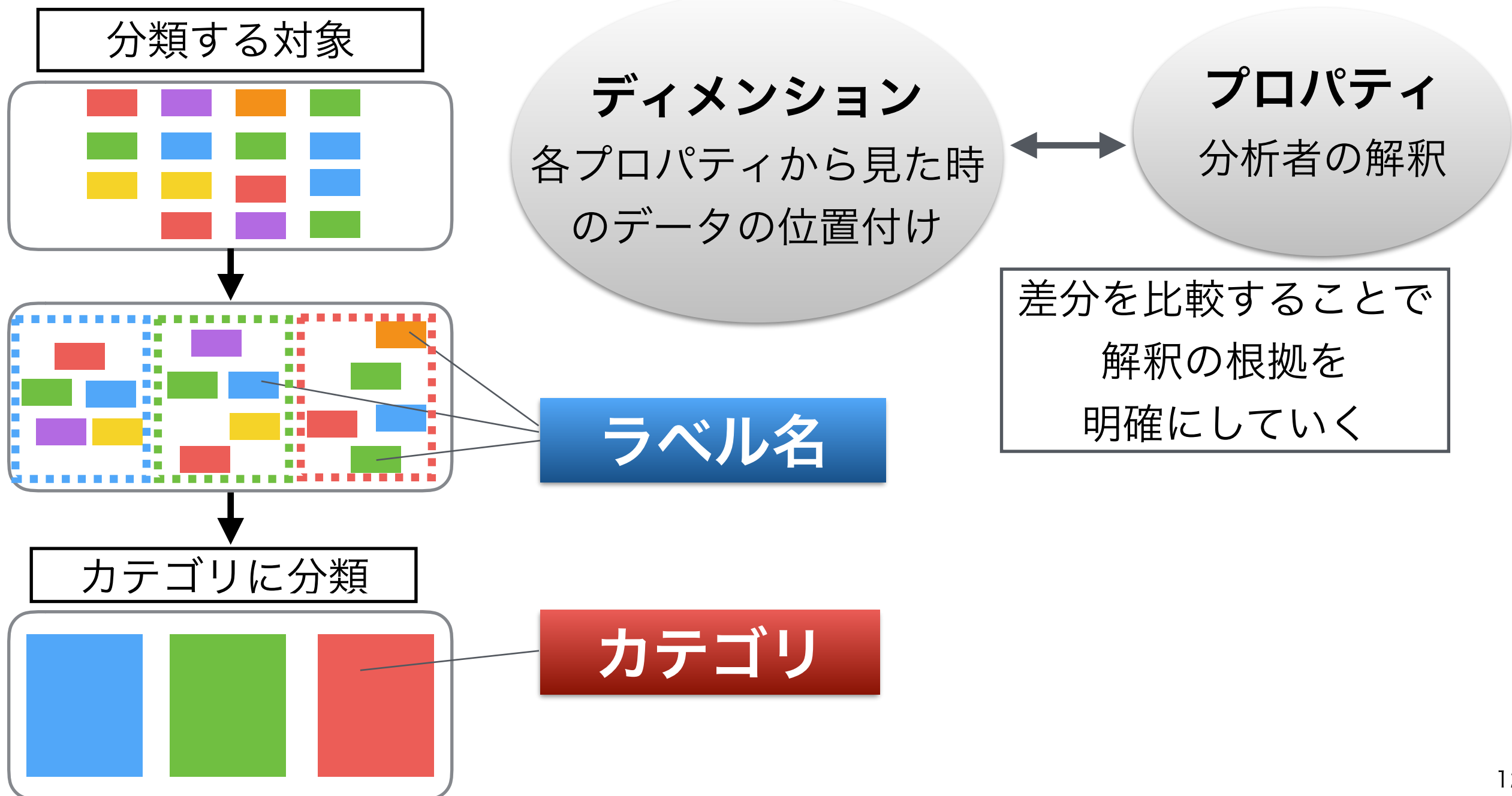
現在のアプローチの流れ



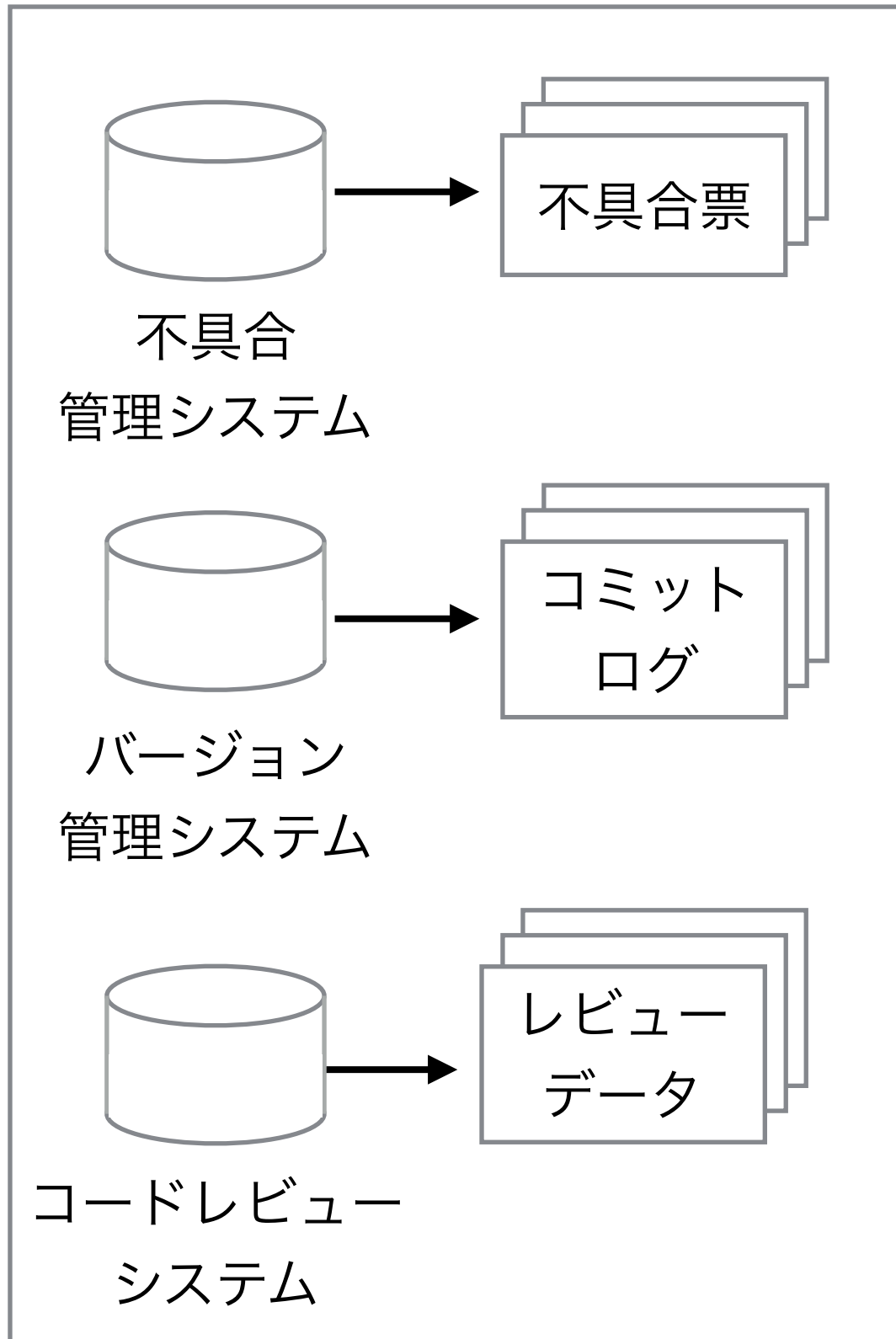
欠陥の特徴分類方法

◎ グランデッドセオリー

- ▶ 定性的分析手法
- ▶ 得られたデータからカテゴリに分類する



データの取得：Mozilla.Firefox

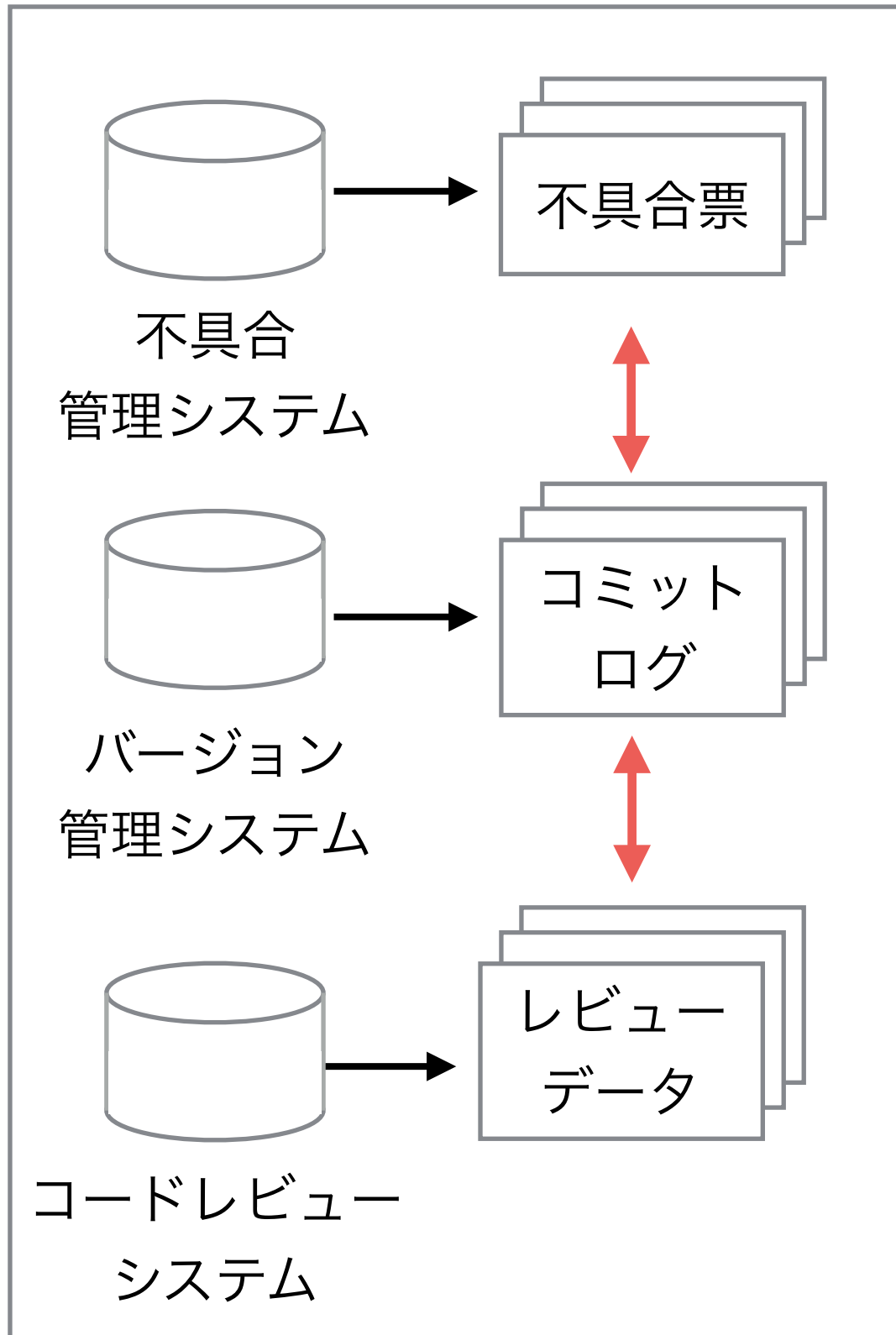


Bugzilla

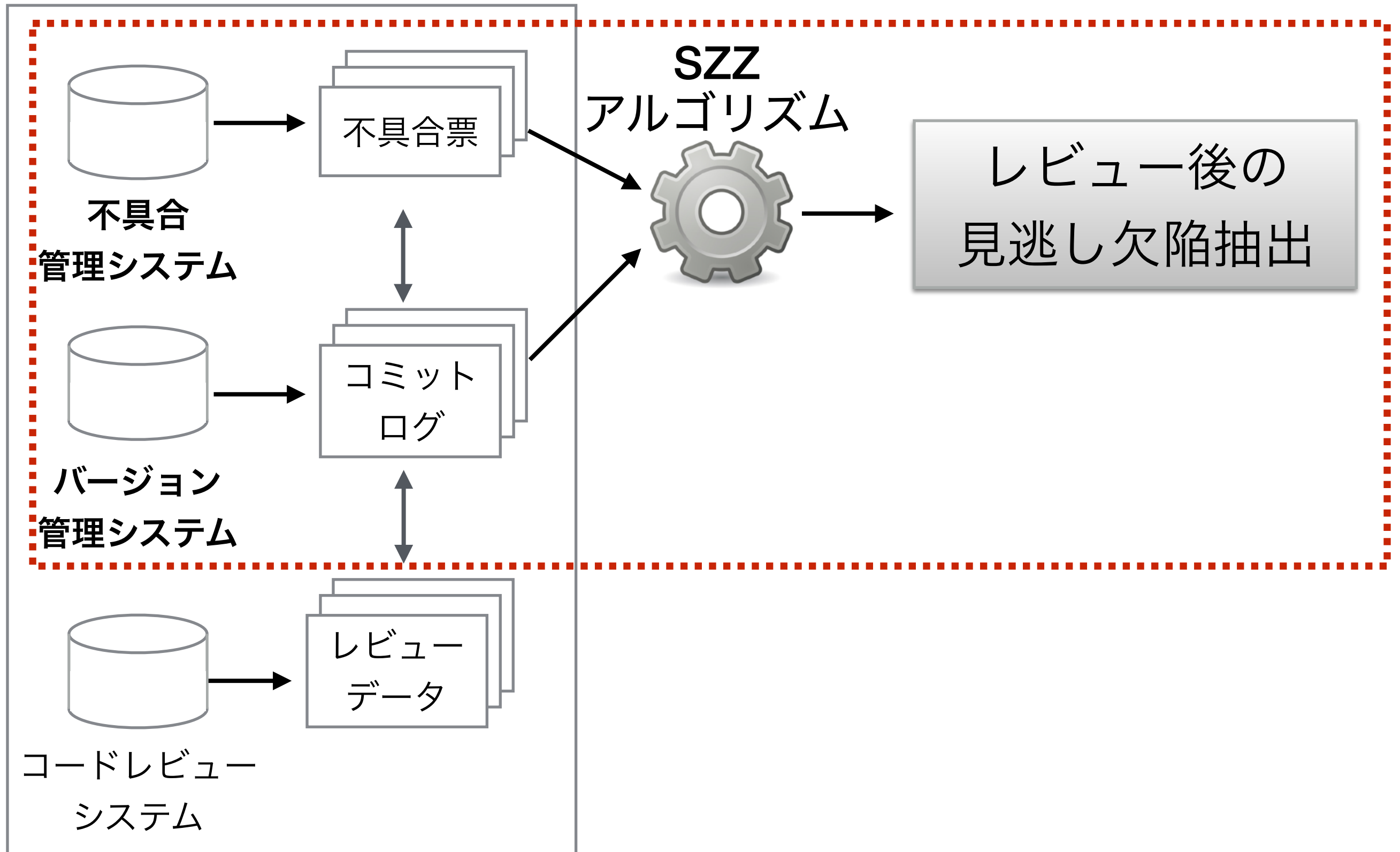
Mercurial (Git.Gecko-dev)

ReviewBoad

データの取得

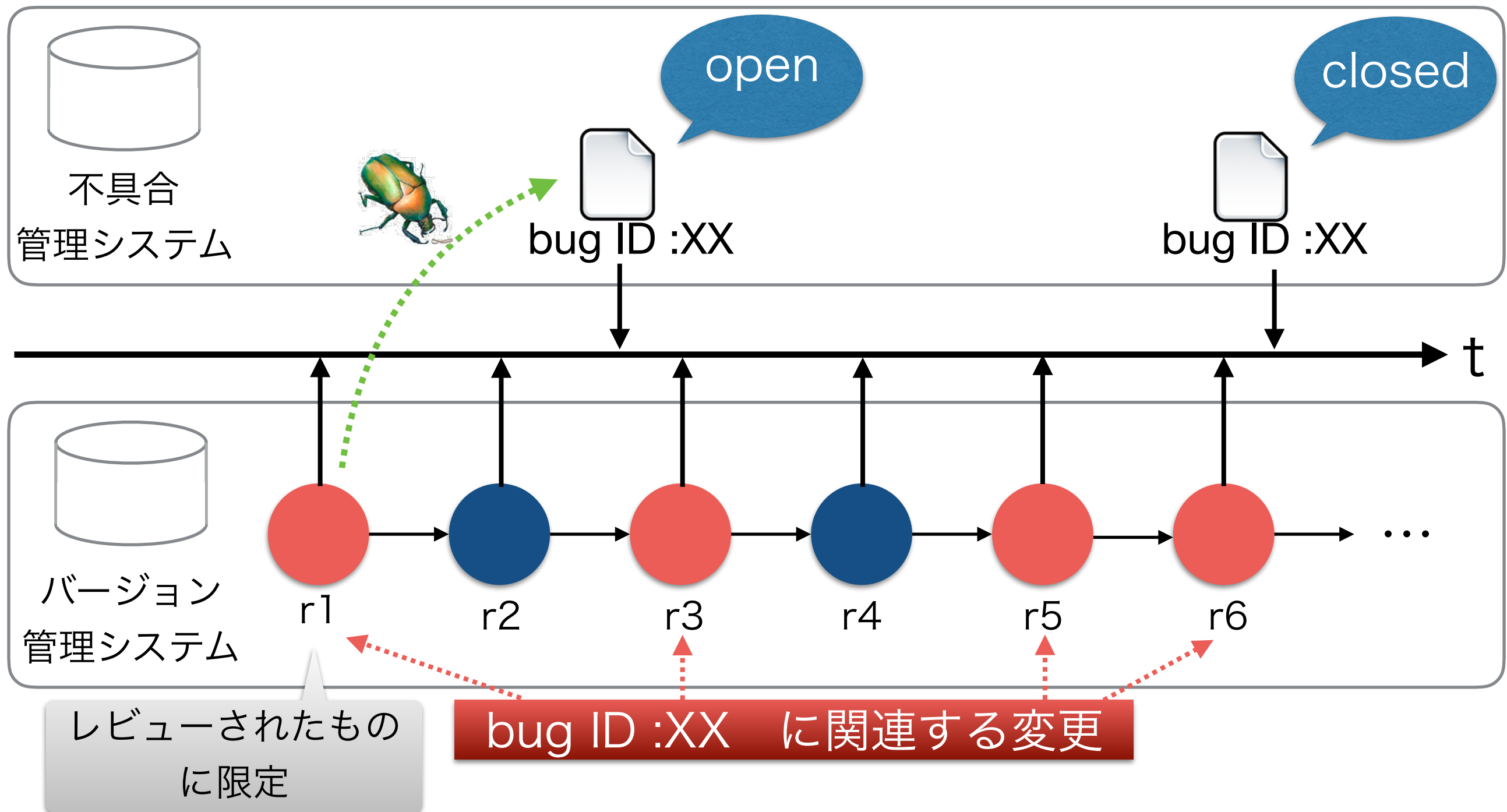


データの取得



見逃しバグの特定方法[6]

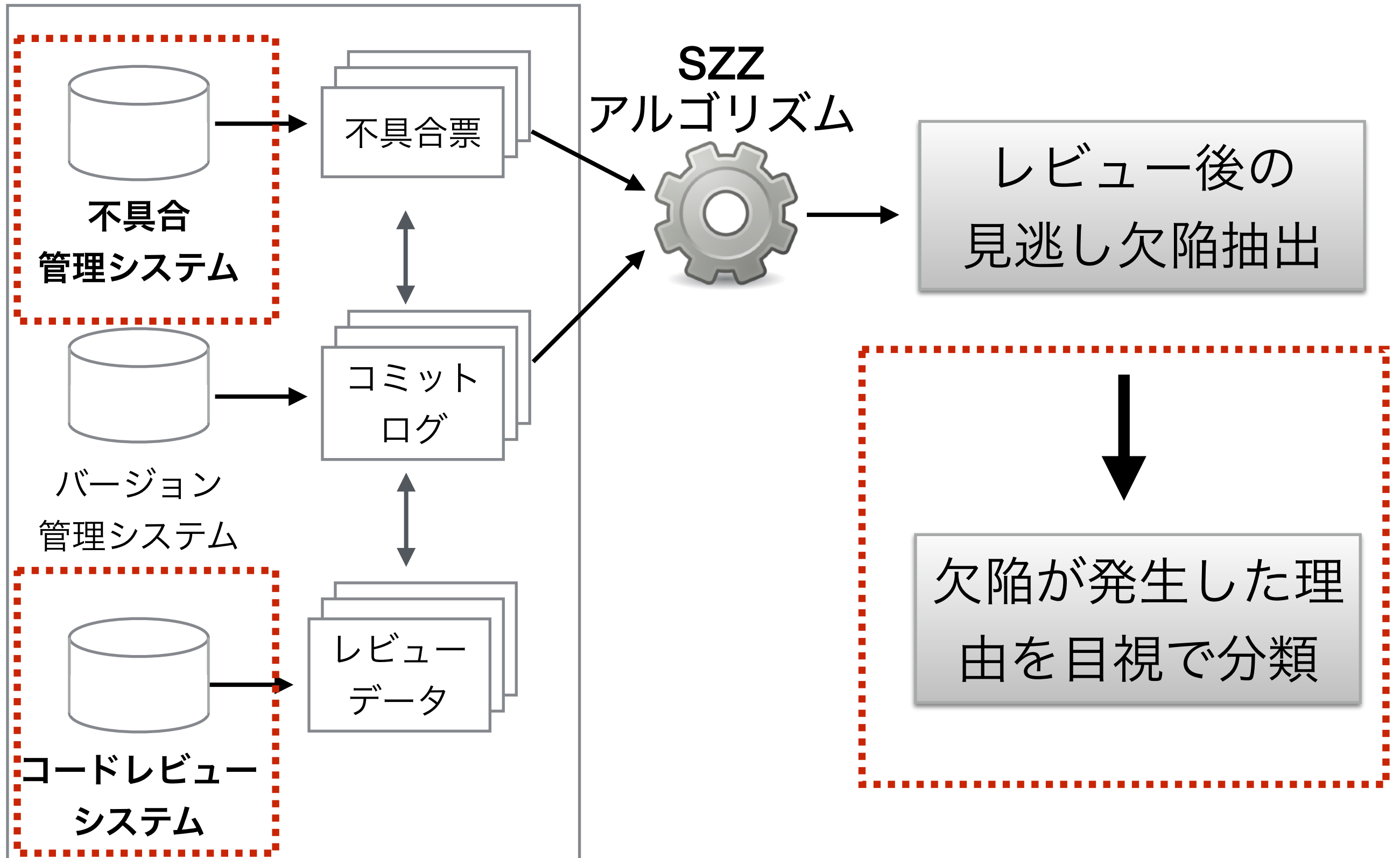
◎ SZZアルゴリズム



SZZアルゴリズムを使う理由と懸念事項

- ◎ そのレビューで不具合が混入したことを特定できる
- ◎ 不具合とコミットの関係は必ずしも1対1ではない
 - ▶ 他対他も考えられるが, 本研究では入り口としてSZZアルゴリズムを使用
 - 分析すべきデータの足がかりを見つける
 - ▶ その後, 他対他の場合などの詳細は目視で確認する

データの取得



分析項目

- ◎ 定量的分析

- ▶ 定性的分析の補助として行う
- ▶ 先行研究の知見を分析に組み込む

- ◎ 定性的分析

- ▶ 定量的分析を元に, 目視で因果関係を調査する

進捗

- ◎ データ取得
 - ▶ バグデータ ○
 - あるらしい
 - ▶ レビューデータ △
 - 今日取得開始します
 - ▶ コミットデータ ×
 - SZZ終了後

- ◎ データ理解
 - ▶ お盆のうちに…

- ◎ サーベイ
 - ▶ データ取得中（お盆）にします