

人的要因とレビュー見逃し確率についての検証

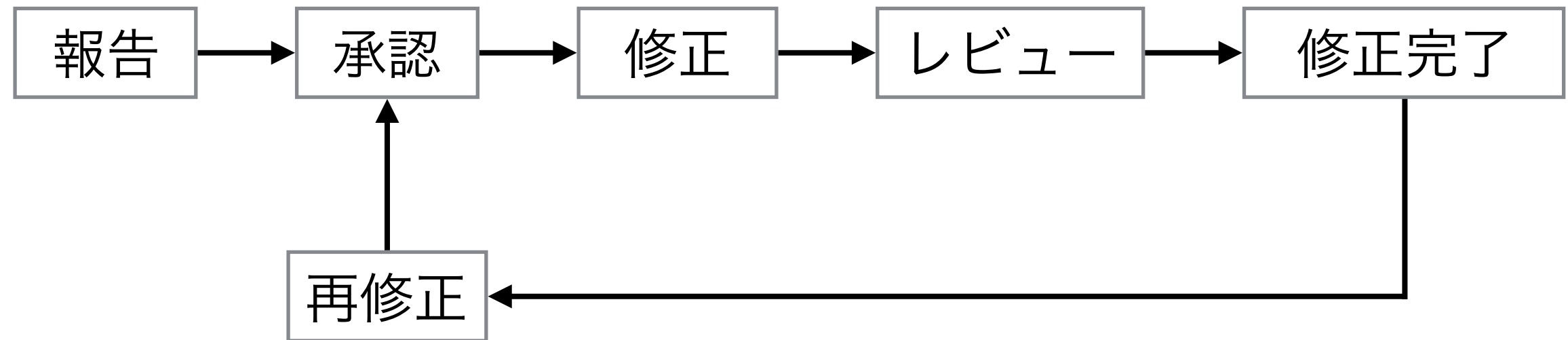
60180081 築山文香

2017/05/24

背景

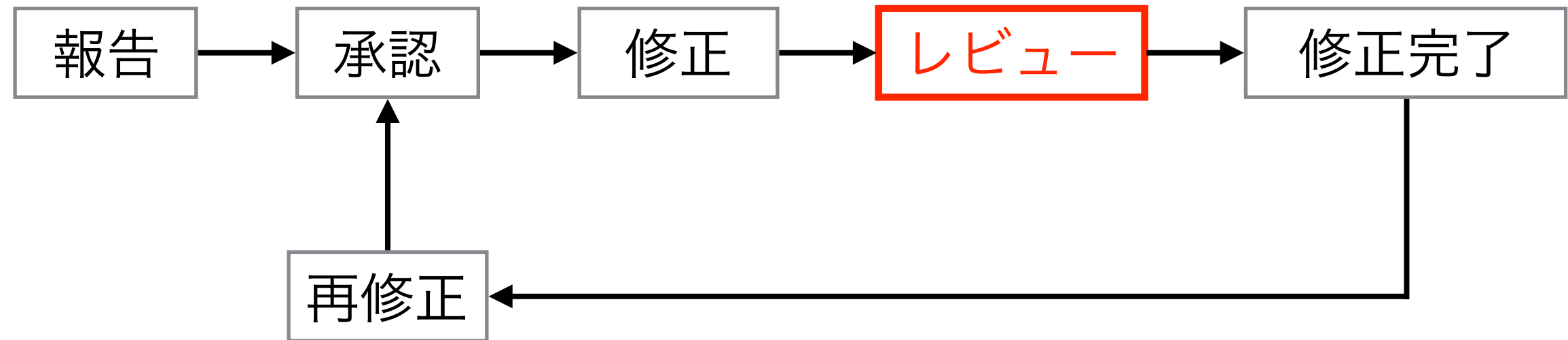
- ◎ ソフトウェア開発工程の80%がソフトウェア保守に使われていると推定されている
- ◎ 不具合修正プロセスにおいて、再修正は大きなコストとなる

不具合修正の状態遷移概要



不具合修正時間の長期化

研究の目的



- ◎ 誰が不具合修正パッチのレビューに関わったか
- ◎ **この人がいたらレビュー見逃しのバグが減る**
 - ▶ 個人
 - ▶ 個人 × 個人（組み合わせ）

レビュー見逃しの定義

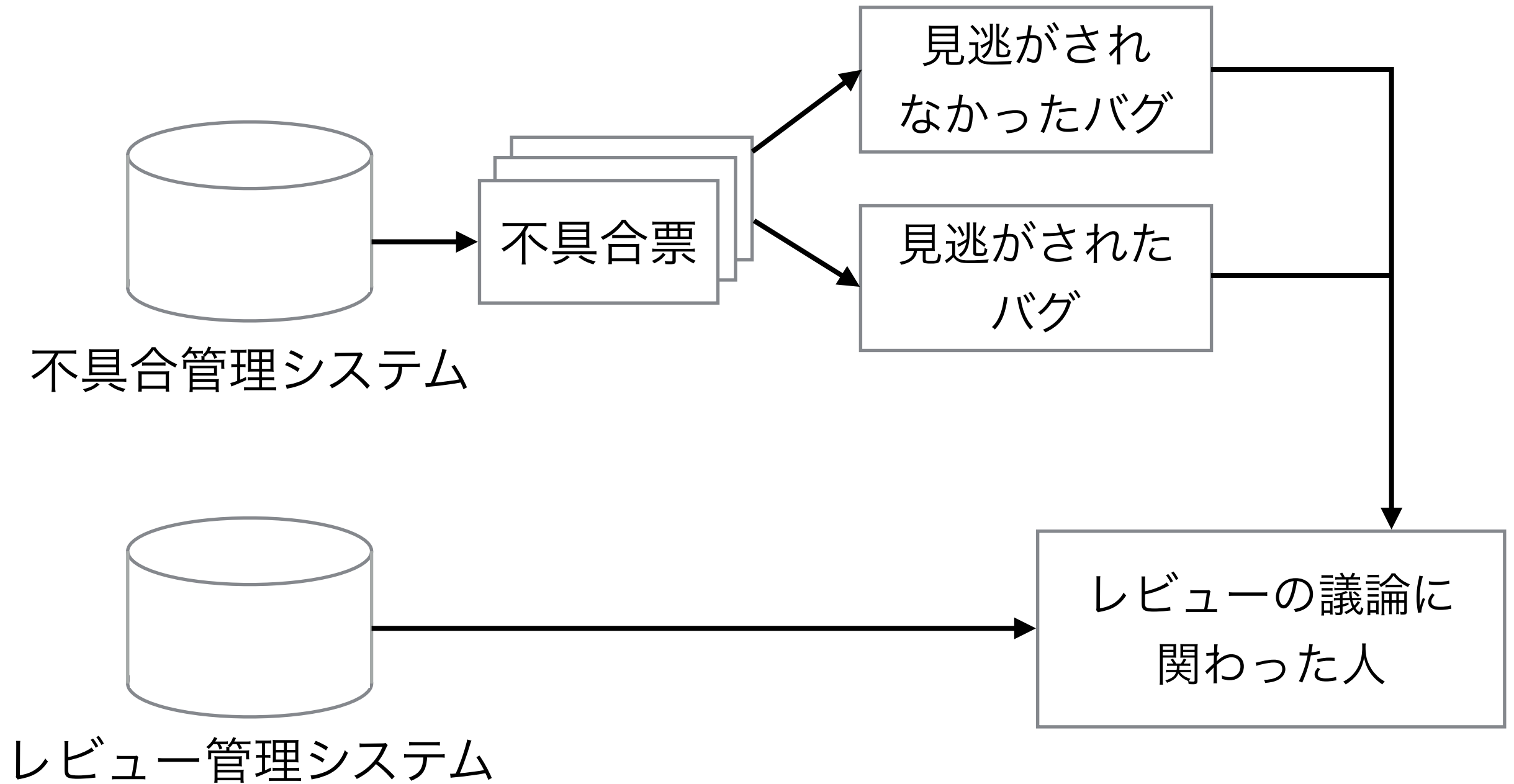
- ◎ SZZアルゴリズムでレビュー時に不具合が埋め込まれたかを判定[1]
 - ▶ 不具合修正 / 機能変更 → レビュー
 - ▶ レビュー時に埋め込まれた ≡ レビュー見逃し

[1] Investigating Code Review Quality: Do People and Participation Matter?

関連研究

- ◎ レビューの参加数, どんな人がレビューしたか (経験, 社会的地位? など) を見る研究はある

アプローチ概要



学習方法

- ◎ レビューを 見逃した / 見逃がさなかった 不具合, それぞれの母数に従いランダムに抽出
- ◎ レビュー議論に参加した個人それぞれにベイズ理論 (?) を適用
 - ▶ 個人のレビュー見逃しの確率を出す

評価

- ◎ k-分割交差検証を適応
 - ▶ $k = 10$?
 - ▶ $k-1$ を学習データ, 1をテストデータとして, テストデータを変えながら k 回学習して評価

レビューの議論に
関わった人
(最終コミット
した人含む)

レビューを見逃す
確率

分析

- ◎ まずは, コミットした個人の確率
- ◎ 次に, レビューに参加した個人の確率
- ◎ 最後に, レビューに関わった 個人×個人 の確率
 - ▶ 例
 - Aさんがコミット, Bさんが議論にいる
 - CさんとDさんが議論にいる

追加分析

- ◎ レビュー議論に関わった人はどんな人か
 - ▶ 関わった人を抽象化する
 - どのような観点（不具合の種類）で不具合検証を行っているか
- ◎ どのような人がレビューの議論に参加すると良いか
 - ▶ 経験
 - ▶ 不具合をレビューする頻度

基本分析TODO

- ◎ 見逃しバグと個人の確率に関してはわりと（まだ）簡単？
 - ▶ データ取得
 - ▶ SZZアルゴリズム
 - ▶ ベイズ理論
 - ▶ 個人名を入れて, 不具合見逃し率を出す
- ◎ 前提
 - ▶ レビューした個人名（ユーザー名）が特定できる
 - 名寄せ…？
- ◎ 不具合票 × コミットデータ

追加分析TODO

- ◎ 追加分析まで考えると、取得するメトリクスを考慮する必要性？
 - ▶ 個人の属性を考慮したい
 - ▶ 後で考える