システム連携時代のソフトウェアレビューの課題と対策

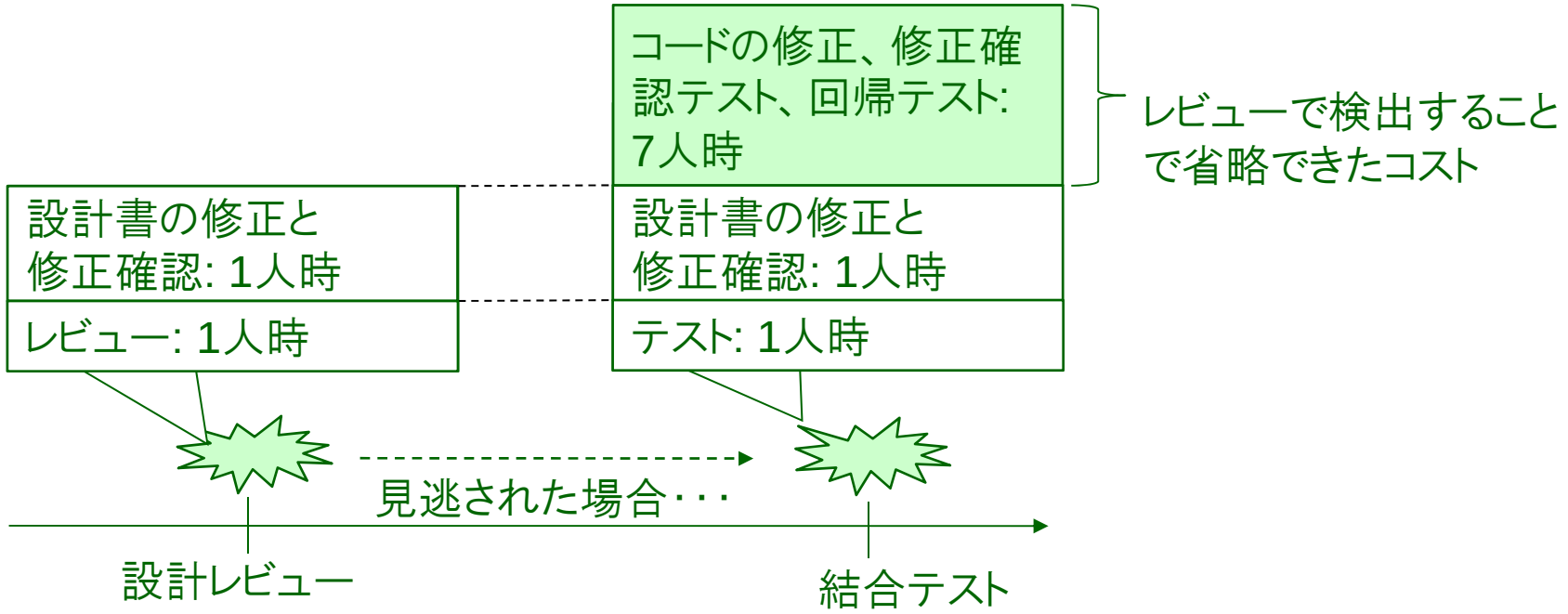
森崎　修司
名古屋大学 大学院情報科学研究科

ソフトウェア／システムの評価技術とレビュー

- 評価技術
  - 静的解析: プログラムを動作させずに成果物、中間成果物を評価する。
    - レビュー
    - ソースコード解析
  - 動的解析: プログラムを動作させながら成果物を評価する。
    - テスト
    - プロファイリング
- レビュー
  - 成果物や中間成果物を目視で確認し、問題がないか確認する。
  - 活動の自由度が高く、結果が場の設定やレビューアに依存しやすい。

レビューの効果が得られる例

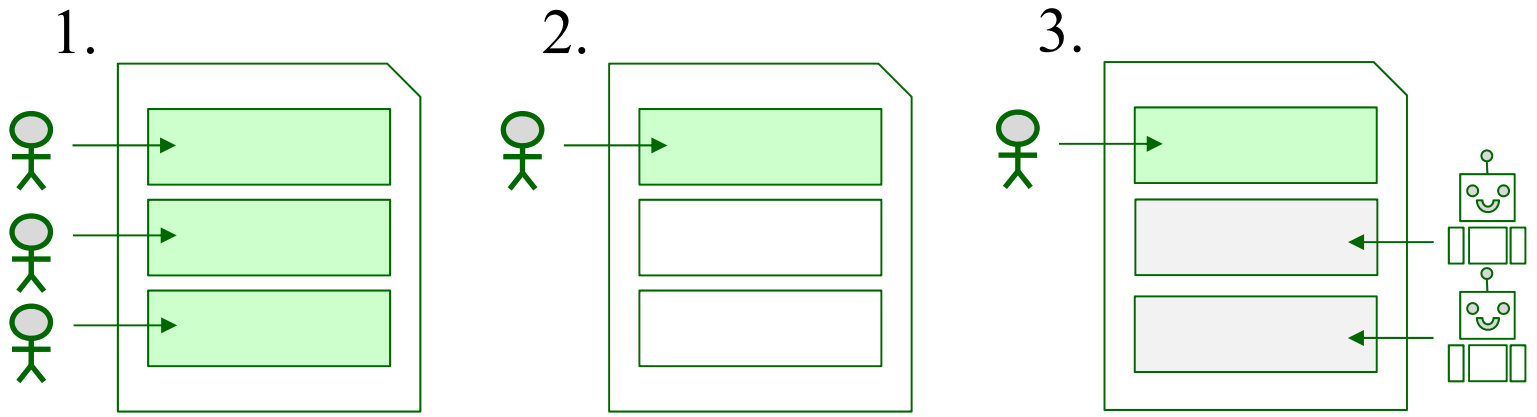
- レビューで見逃してテストで発見された場合に、修正コストが大きくなる欠陥を発見できると、効果が得られる。



参考: http://www.itmedia.co.jp/im/articles/1008/19/news090_3.html

システム大規模化、システム連携に伴う課題と対策

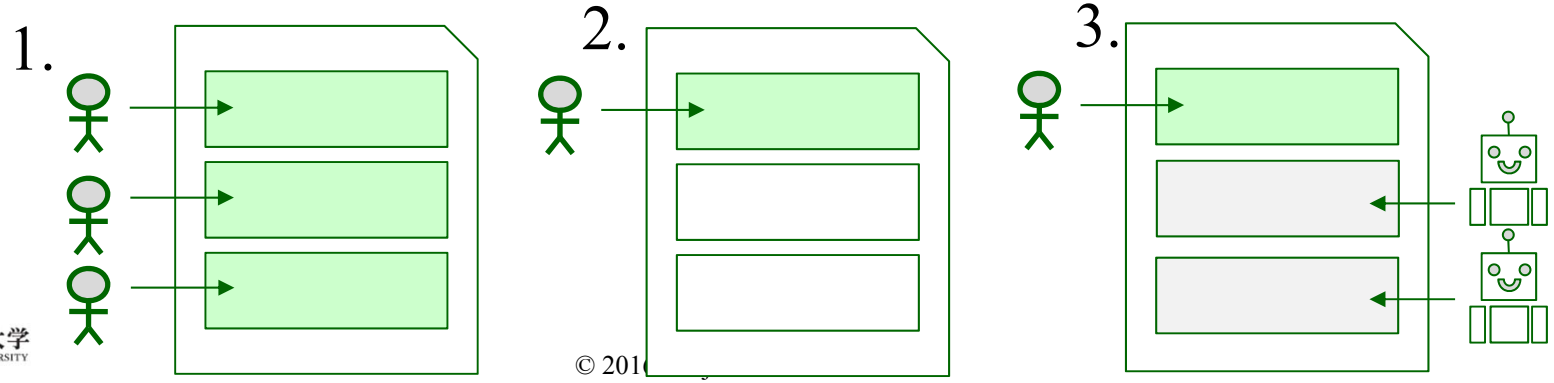
- 課題
  - 大規模化、連携に伴って1人のレビューアーではカバーできなくなる。
- 対策
  1. 複数人のレビューアーで分担する。
  2. 検出する欠陥を絞り込む。
  3. 一部自動化、形式化する。



システム大規模化、システム連携への対策の障壁

実際に対策しようとすると簡単にはできない。

1. 複数人のレビューアーで分担する。
   - 複数人でレビューしても同じような欠陥しか検出できない。
   - どうやって分担するか明らかではない。
2. 検出する欠陥を絞り込む。
   - 何をあきらめて何を検出するか決めるのは簡単ではない。
   - どういう基準で決めるか明らかではない。
3. 一部自動化、形式化する。
   - 何を自動化し、形式化すればよいか明らかではない。



障壁の克服へむけて

- 特定の着眼点に限定したレビューの訓練をする。
  - あれこれ指摘せず、絞り込めるようにする。
- 役割や着眼点で分担して、欠陥検出する。
  - 開発に携わる役割で確認（Perspective-based reading）
  - 業務観点での過去の欠陥の抽象化
- 価値の高い部分からレビューする。
  - ユースケースで重み付け（Usage-based reading）
  - ビジネス価値の分析
- 一部自動化、形式化
  - Fix cache
  - Fault-proneモジュール予測

着眼点を限定できるか実証的に評価

- 着眼点の限定、時間短縮可能かを実証的に評価した。
- 実証評価をイベントの一環として実施した。
  - オンラインショッピングのためのサーバサイドプログラム（Java 4000行程度）
  - 認証の一貫性、パスワード再発行、虚偽のアカウント発行が可能となっていないかだけを欠陥検出してもらう。
  - 協力者: 100名程度の実務者

着眼点をセキュリティに限定した結果

全指摘数（参加人数）	370件（93名: 1名あたり3.98件）
正答率	76.1%
観点以外の指摘	91件（24.6%）
レビュー速度（行/時）	3,860行 （Technical review, 問題発見、Java、コメント込み）

既存文献でのレビュー速度

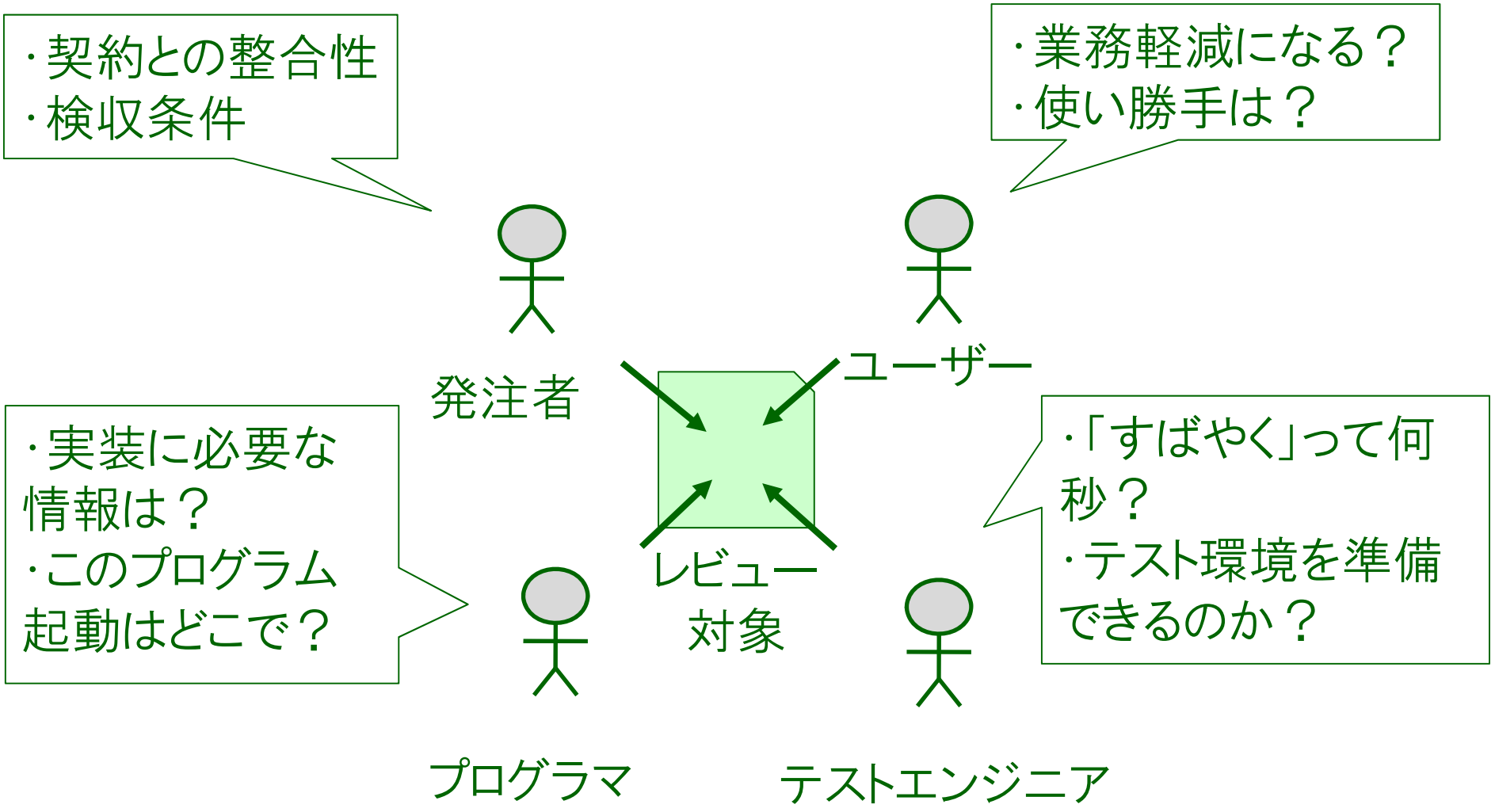
文献名	公開年	速度（行/時）
Fagan M. “Design and code inspection to reduce errors in program development”, IBM System Journal vol. 15, no. 3	1976	700～900行（問題発見）、 500～700行（問題指摘）、 （インスペクション、 COBOL, コメント行含まず）
Introduction to the Team Software Process TSPi	1999	200行以下 （インスペクション）
IEEE 1028 Software Reviews and Audits	2008	100～200行 （Technical reviews）

開発に携わる役割による分担（PBR）

- PBR: Perspective-based reading*
- ソフトウェアに関わる役割を定義し、役割を前提においた検出方法を合意しておく。
- 役割の例: ユーザー、テストエンジニア、プログラマ
- 役割は必ずしも実際の立場でなくてよい。
  例）プロジェクトリーダーがユーザーの役割を担当してもよい。

* V.R. Basili, S. Green, O. Laitenberger, F. Lanubile, F. Shull, S. Sørumgård, and M.V. Zelkowitz, The Empirical Investigation of Perspective-based Reading. Journal of Empirical Software Engineering, vol. 1, no. 2, pp.133-164(1996)

Perspective-based reading 役割と分担の例



業務観点での過去の欠陥の抽象化

- 目的: 業務に特化した欠陥に抽象化できるか実際の不具合情報と熟練者へのインタビューによって確かめる。
- 対象ソフトウェア
  - 用途: 大手銀行の外貨有価証券の取引入力、記帳管理
  - 開発期間: 2年(基本設計～リリース、ウォータフォール)
- 不具合情報
  - 当該ソフトウェアのレビューでは，既に汎用的な欠陥タイプでの欠陥検出が完了している.
  - 収集フェーズ: 結合テスト(サブシステム間の結合)
  - レビューで検出できたと判断された不具合: 488件
    (不具合の原因分析でレビューに原因があると判断されたもの)
  - 入力項目数: 28項目

出典: 森崎 修司, 松本 健一, "業務観点でのレビューを目指した不具合情報の分析", 情報処理学会研究報告 ソフトウェア工学研究会, vol. 2013-SE-179(35), pp. 1-8(2013)
2013年度情報処理学会山下記念研究賞

対象不具合情報のデータ項目（一部抜粋）

項目	形式	説明
原因分析	選択肢（33項目）	「レビュー不備」「レビューア不十分」「レビュー観点不備」等
対応工数	数値	修正に要した工数
重大度	選択肢（A, B, C）	A（全体に大きな影響）, B（特定の機能が使用不可）, C（機能は使用不可）
現象	自由記述	不具合の現象
原因	自由記述	不具合の原因
対策	自由記述	不具合の修正方法
テストイベント	選択肢（28項目）	テストの種別
機能カテゴリ	選択肢（14項目）	機能の分類

出典: 森崎 修司, 松本 健一, “業務観点でのレビューを目指した不具合情報の分析”, 情報処理学会研究報告 ソフトウェア工学研究会, vol. 2013-SE-179(35), pp. 1-8(2013)
2013年度情報処理学会山下記念研究賞

業務観点(日付)に関係する不具合の割合

- 日付に該当する不具合は修正工数の平均がそれ以外の不具合の修正工数の平均よりも大きかった。
- 日付に該当する不具合は重大度Bの割合が大きかった。

分類	重大度			修正工数の平均
	A	B	C	
日付に関する不具合	2 2.3%	59 68.6%	25 29.1%	2.10
日付に関する不具合ではない	4 1.0%	190 47.4%	207 51.6%	1.68
全件(「原因」にレビューを含む)	6 1.2%	250 51.2%	232 47.5%	1.75
不具合情報全体	2.9%	39.8%	57.3%	2.21

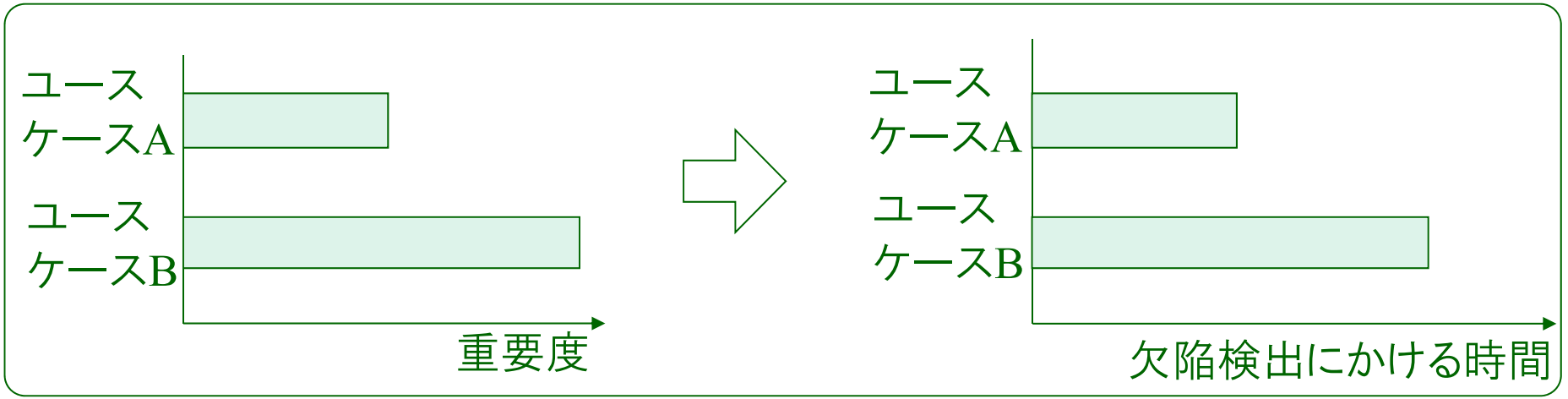
出典: 森崎 修司, 松本 健一, "業務観点でのレビューを目指した不具合情報の分析", 情報処理学会研究報告 ソフトウェア工学研究会, vol. 2013-SE-179(35), pp. 1-8(2013)

障壁の克服へむけて（再掲）

- 特定の着眼点に限定したレビューの訓練をする。
  - あれこれ指摘せず、絞り込めるようにする。
- 役割や着眼点で分担して、欠陥検出する。
  - 開発に携わる役割で確認（Perspective-based reading）
  - 業務観点での過去の欠陥の抽象化
- 価値の高い部分からレビューする。
  - ユースケースで重み付け（Time controlled reading）
  - ビジネス価値の分析
- 一部自動化、形式化
  - Fix cache
  - Fault-proneモジュール予測

Time controlled reading

- ユースケース別に欠陥検出するUsage-based readingを拡張している。
- ユースケースを列挙し、重要度等の優先順位を付ける。
- 優先順位の高いユースケースに時間を多く割当てて、ユースケース別に欠陥検出を試みる。



Kai Petersen, Kari Rönkkö, Claes Wohlin, “The impact of time controlled reading on software inspection effectiveness and efficiency: a controlled experiment, Proc. of International Symposium on Empirical Software Engineering and Measurement, pp. 139-148(2008)

ビジネス価値の分析による優先順位付け

- 対象: 公共交通機関(企業D)の座席予約システム
- システムの特徴
  - インターネット、企業Dの窓口、旅行代理店の端末といった連携システムから座席を予約できる。
  - 標準的な予約手順があり、詳細な状態遷移の定義やそれに対応した再利用できるテストがある。
  - 予約システムは長期にわたって運用する。
- システムの価値
  - 予約と対応する座席に過不足がない。
  - 短時間で予約でき、乗客が代理店やインターネットから座席の空きを確認しながら予約できる。
  - システム内部の予約の手順が変更されず長期にわたって保守できる。

森崎 修司:「間違いだらけの設計レビュー 改訂版」日経BP, p. 75 (2015)

ビジネス価値の分析による優先順位付け

- システムの価値
  - 予約と対応する座席に過不足がない。
    → 予約、解約、参照の間で守られるべき条件が守れているかを確認する。
  - 短時間で予約でき、乗客が代理店やインターネットから座席の空きを確認しながら予約できる。
    → 性能要求が損なわれる可能性のある部分を抽出しておき、テストで確認する。
  - システム内部の予約の手順が変更されず長期にわたって保守できる。
  - → 予約手順が実質的にかわっていないかを確認する。対応する設計やコードの変更が必要ないかを確認する。

森崎 修司:「間違いだらけの設計レビュー 改訂版」日経BP, p. 75 (2015)

変更履歴を用いた欠陥予測

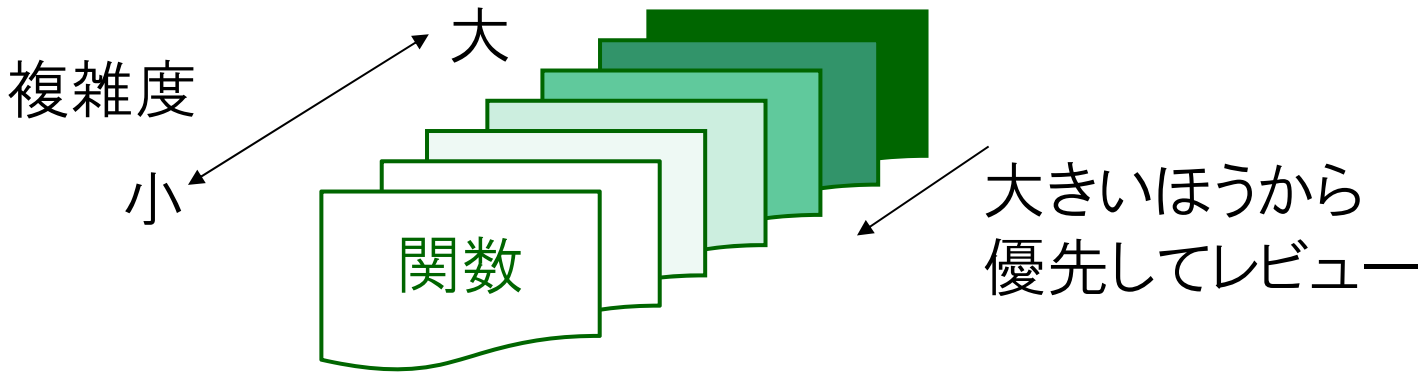
- FixCache*[1]: 過去に修正された不具合と同じような不具合が再発する。
  - 7つのオープンソースプロジェクトの変更履歴約200,000件で実証
  - 「メモリキャッシュ」のアルゴリズムを用いて不具合の可能性の高いソースコードファイルを選出
  - 選出した全体のソースコードファイルの1割で73～95%の精度で不具合を予測できた。
- Googleでのソースコード更新履歴に基づく不具合予測に応用されている。*[2]

*1: Kim S., Zimmermann T., James E. W., Zeller A., “Predicting Faults from Cached History” In Proceedings of the 29th international conference on Software Engineering , p. 489-498.

*2: Lewis C., Ou R. “Bug Prediction at Google” http://google-engtools.blogspot.jp/2011/12/bug-prediction-at-google.html

ソースコードメトリクスを用いた選択

- 全てのソースコードをレビューできない場合に、重点化するソースコードをツールを用いて選ぶ。
  - ツール: ソースコード静的解析ツール
  - 選択方法: サイクロマティック複雑度が大きい関数を選ぶ。
- 商用のソースコードを対象とし、最大で全体の40%の関数を目視することで効果が得られることを確認した。
  (AUC: Area Under the Curveによる精度比較)



N. Kasai, S. Morisaki, and K. Matsumoto, Fault-Prone Module Prediction Using a Prediction Model and Manual Inspection. In *Proceedings of the 2013 20th Asia-Pacific Software Engineering Conference , pp.* 106-115(2013)

名古屋大学 NAGOYA UNIVERSITY

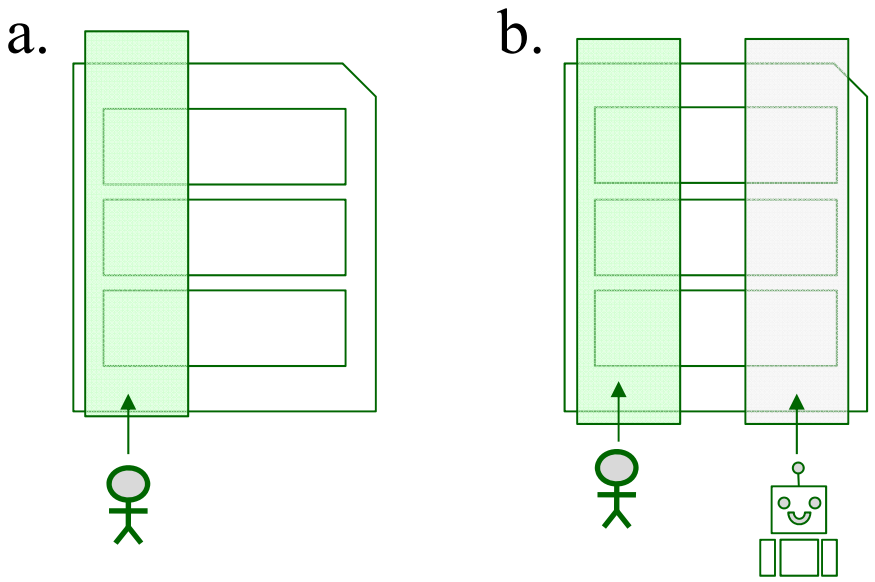
まとめ

- レビューの定義と効果を示した。
- レビュー対象であるシステム大規模化とシステム連携がもたらす課題と対策を紹介した。
  - 特定着眼点での欠陥検出が可能であることを実証的に評価し、分担につながることを確認した。
  - 検出すべき欠陥の分類方法と事例を紹介した。
  - 検出すべき欠陥の優先順位付けの方法と事例を紹介した。
  - レビューの一部自動化にむけた方法と事例を紹介した。

今後の課題

a. 特定アスペクトでの表記によるレビューの効率化

b. a. に加え特定アスペクトを用いた一部自動化



書籍、記事

- 書籍
  - 「なぜ重大な問題を見逃すのか？間違いだらけの設計レビュー改訂版」（日経BP, 2015）
- 記事
  - 「ソフトウェアレビュー入門」
    @IT（アットマークアイティ）
    http://www.itmedia.co.jp/keywords/software_review.html
  - インスペクションの世界
    ThinkIT http://thinkit.co.jp/book/2009/03/01/66
  - 「提言 昔ながらのレビューから脱却しよう」
    日経ITpro http://itpro.nikkeibp.co.jp/article/COLUMN/20120808/415157/
  - 「コードレビューの道具、使っていますか？」
    IBM developerWorks
    http://www.ibm.com/developerworks/jp/opensource/library/j_os-codereview/ (日本語)
    http://www.ibm.com/developerworks/library/os-toolset/ (英語)

名古屋大学 NAGOYA UNIVERSITY