ソフトウェアレビューの本質から考える品質保証

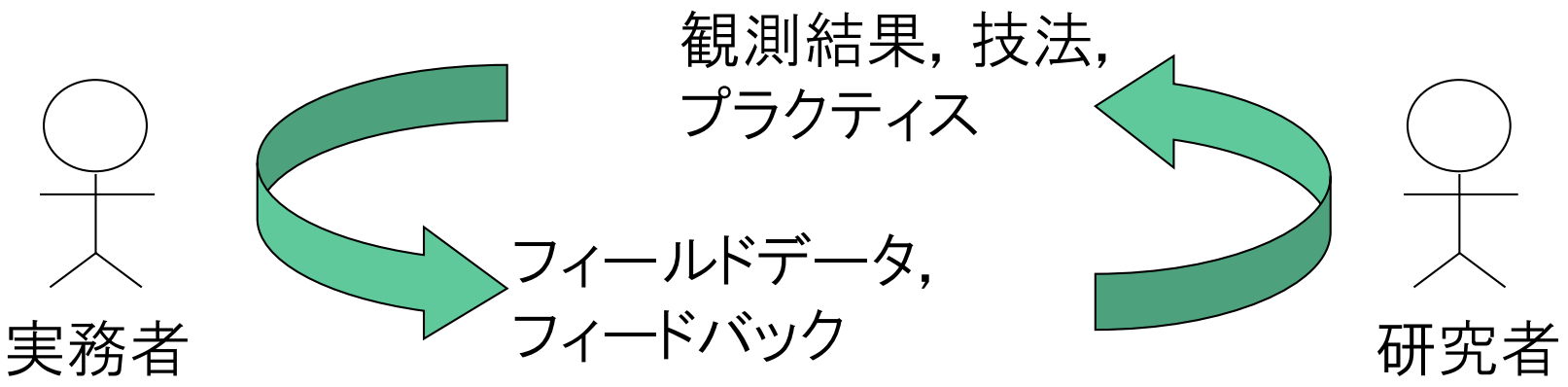
レビューの原理、原則と事例から読み解く

森崎　修司

名古屋大学 大学院情報学研究科

研究活動の方針 ― 森崎 修司

- 実務として役に立ち、研究として価値があるテーマ設定を目指している。
  - 実証的ソフトウェア工学
  - 研究が実践に新たな気づきを提供する。
  - 実践が研究の発展につながる。



名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

アジェンダ

- 前提
  - 本セッションでのソフトウェアレビューの定義
  - 本セッションでお伝えしたいこと
- レビューの効果
  - 効果を高めるための原理
  - 不具合の修正工数の調査
- レビューで検出すべき問題
  - コンテキストに応じた問題種別
  - 問題種別と検出シナリオ
  - 問題種別の設定方法
- 効率化にむけて
  - レビューの局所化
  - 役割分担

名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

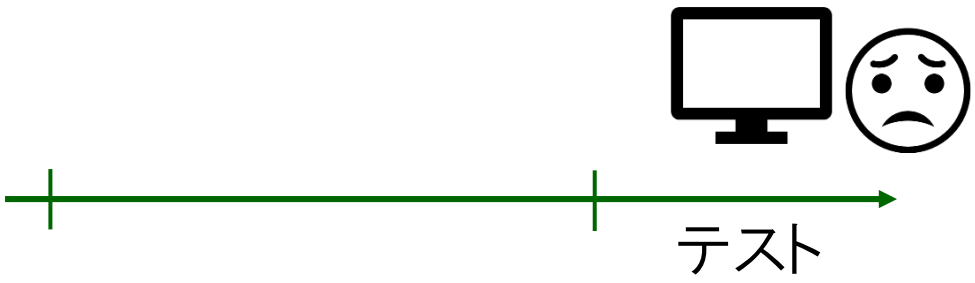
ソフトウェアプロダクトレビューの定義

- プログラムが動作する前に中間成果物に問題がないかを目視で確認する静的解析技法
- 目的
  - 不具合の修正コストの低減
  - 状況把握と承認
  - 保守性向上
- インプット
  各種ドキュメント（ソースコードを含む）
- アウトプット
  インプットのドキュメントに含まれる問題点のリスト

（本セッションでの「問題」は潜在的なバグ、欠陥を含む）

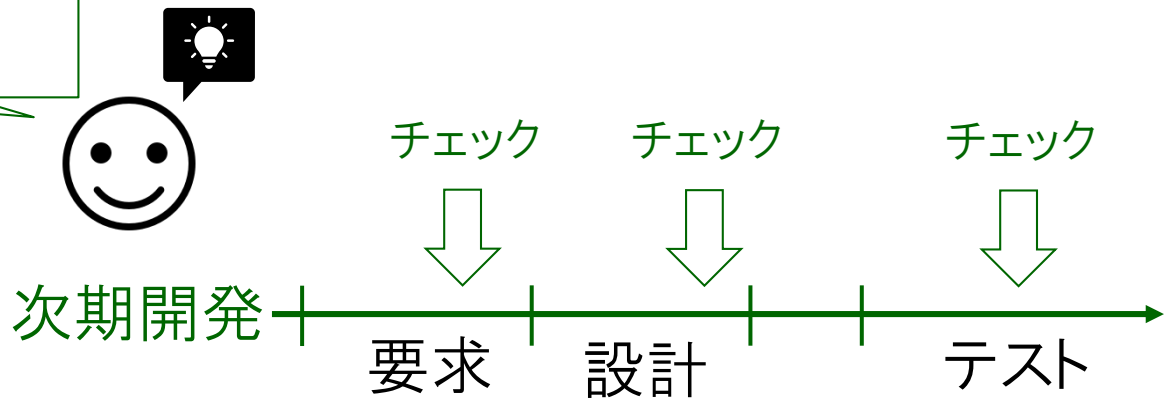
© 2020 Shuji Morisaki

本セッションの総括 1/4



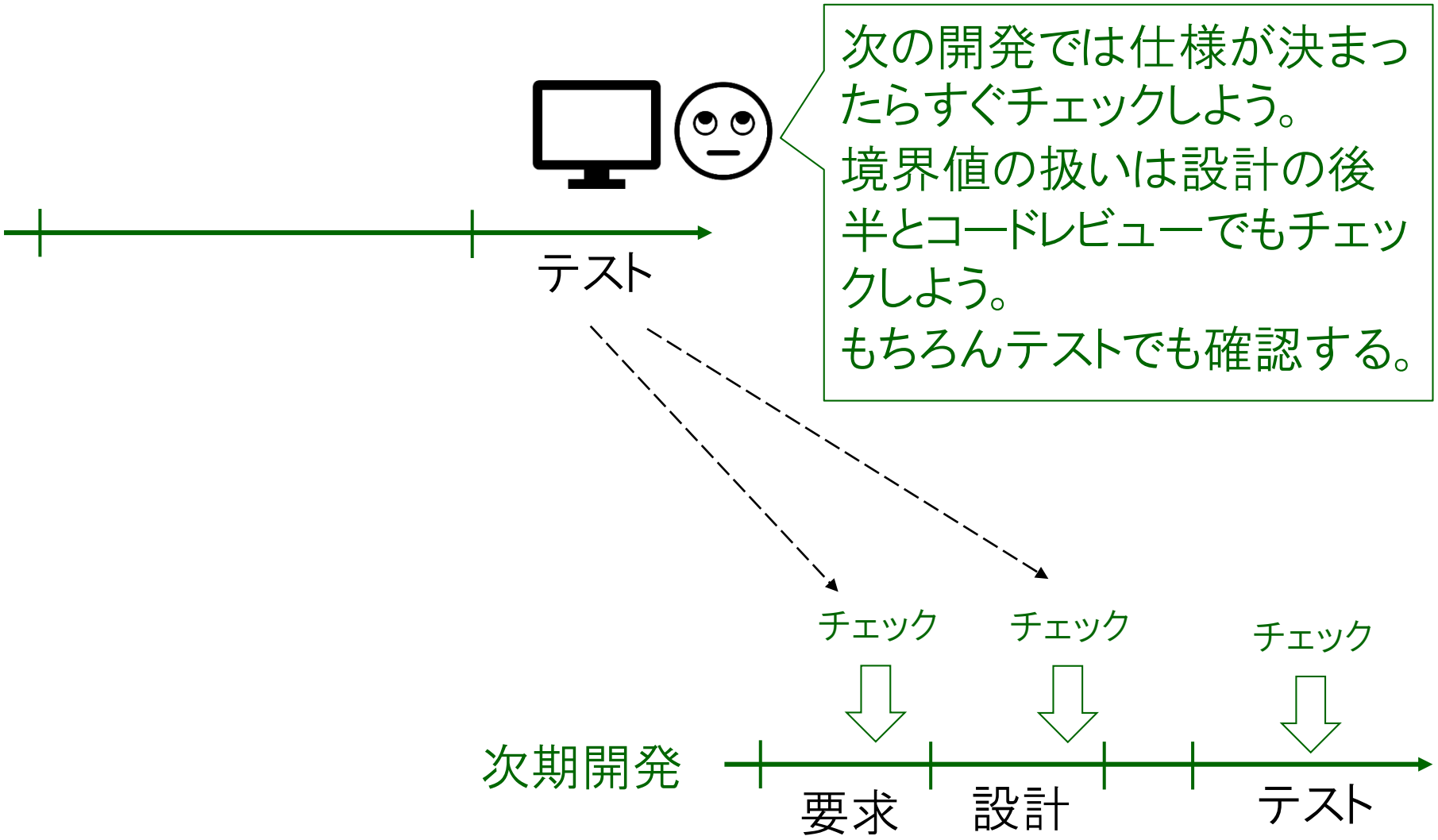
こんな仕様じゃユーザが納得しないよ…
境界値の扱いが曖昧な仕様でプログラムはもっとひどい…
早い段階からチェックしていれば…

前倒しできるものは全部前倒ししてチェックすれば絶対によくなる！



名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

本セッションの総括 2/4

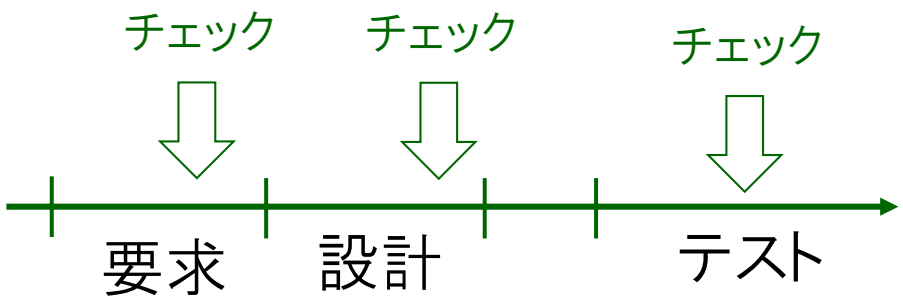


© 2020 Shuji Morisaki

本セッションの総括 3/4

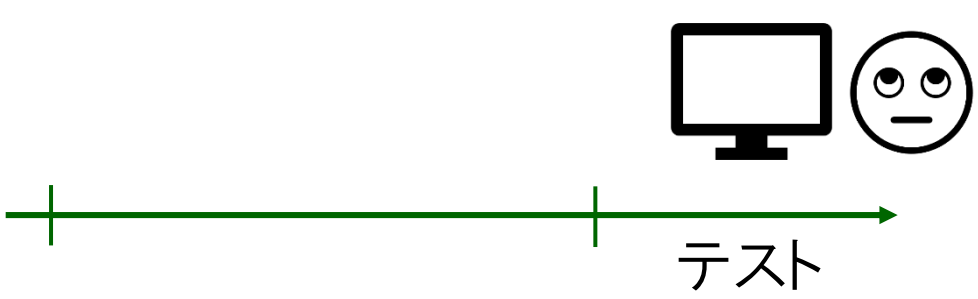
実際に前倒しすると･･･

前倒しするとコストが増える…
しかも前倒ししてチェックして指摘しておいてもその後にバグが混入する…



© 2020 Shuji Morisaki

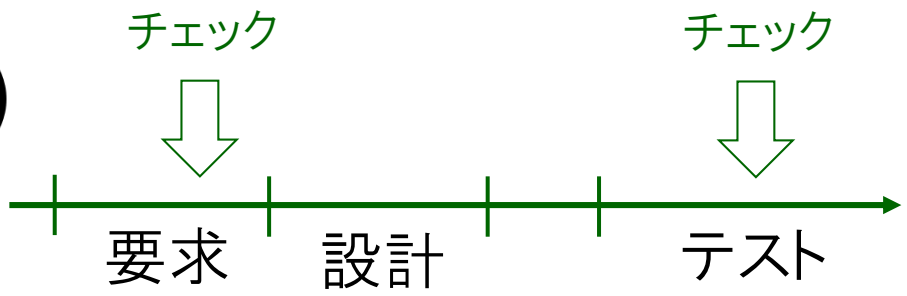
本セッションの総括 4/4



次の開発では仕様のこの部分は手戻りが大きいから、次から早期に確認しよう。
境界値は他と比べると手戻りが小さいからテストで確認しよう。

効果のあるタイミングでレビュー

効果があるもののみ早期に確認するので、適正なコストで実施できる。



名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

実際には…

仕様ができたら、ユーザが関与するXX部のチェックをさせてほしいんです。テストで不備が見つかると手戻りが大きいから

他にもチェックしたいことあるんだけど・・・

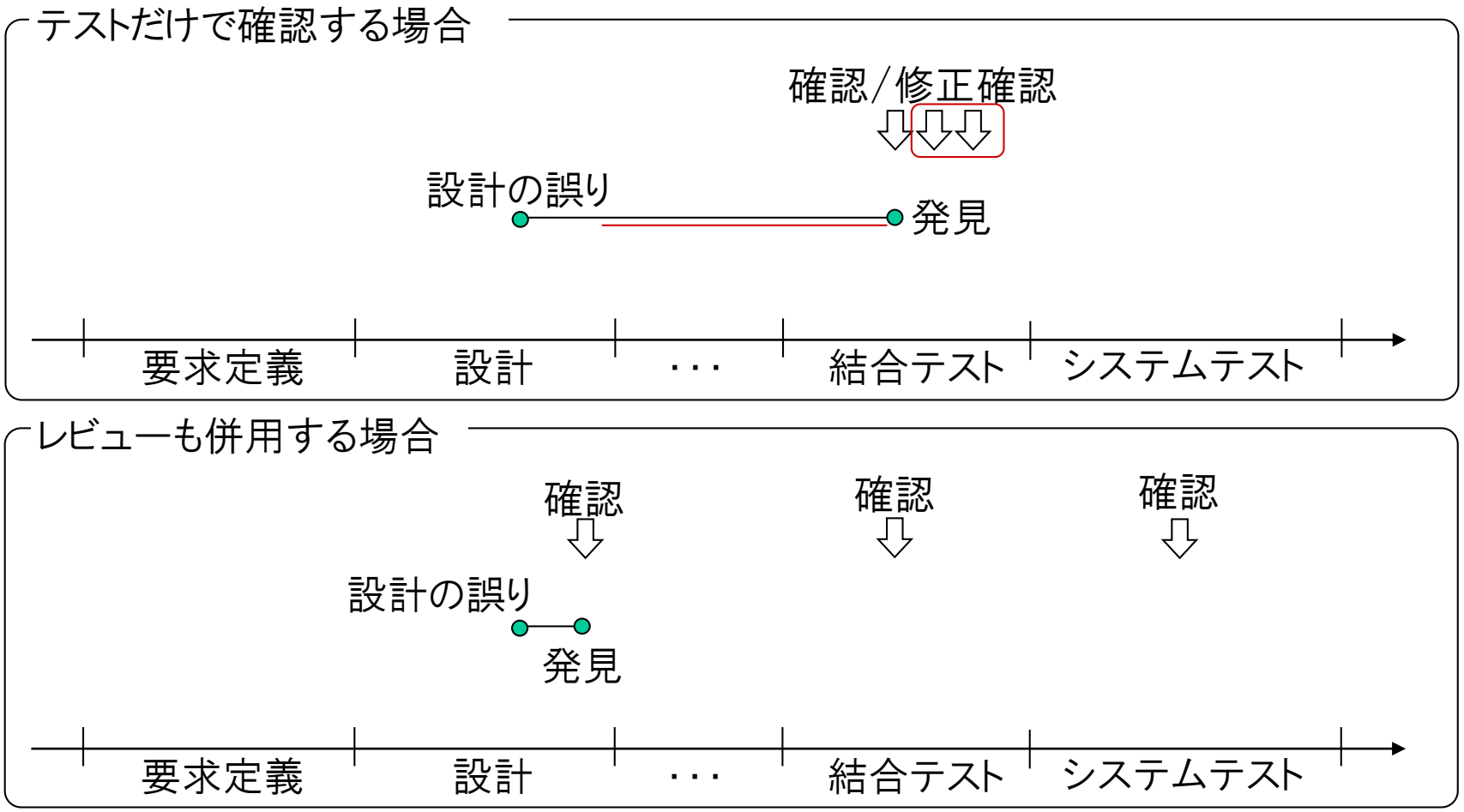
本セッションでは人間力で解決することを前提にして、この部分は別の機会に…。

名古屋大学 NAGOYA UNIVERSITY

© 2020 Shuji Morisaki

レビューの効果

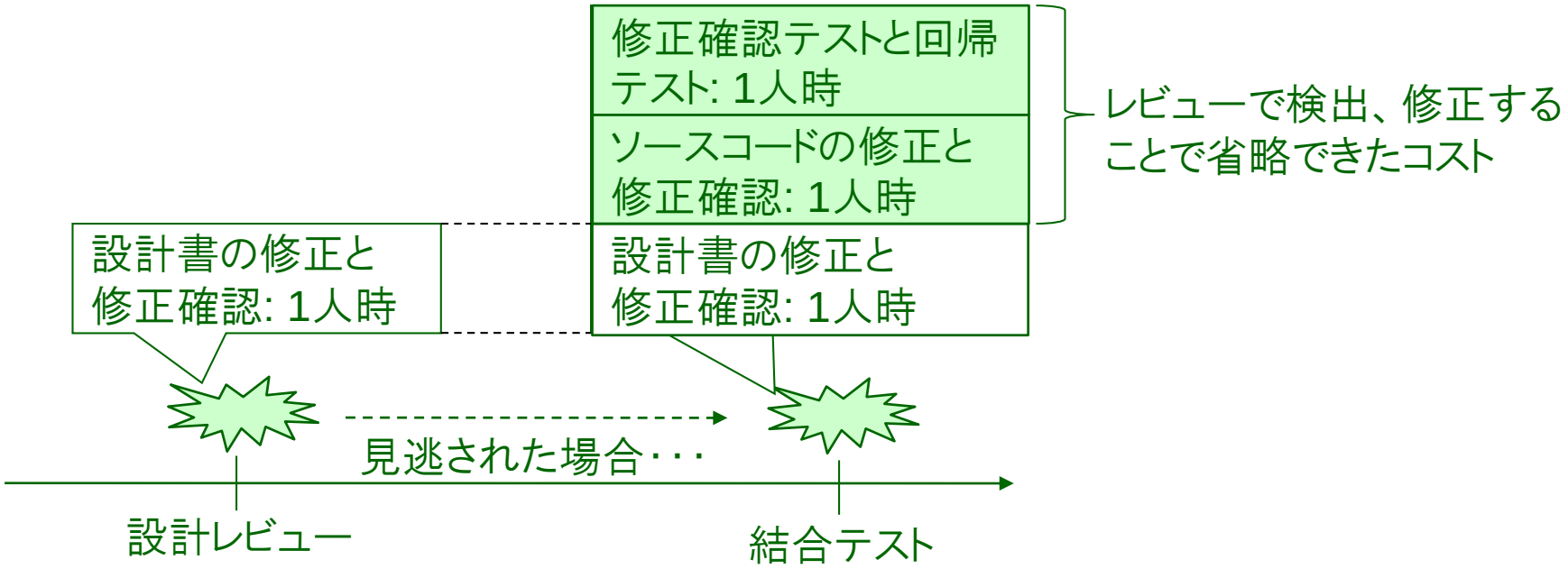
- テストでの修正確認コストの低減
- 誤りに基づいた設計や実装コストの低減



名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

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検出、修正できると、効果が得られる。

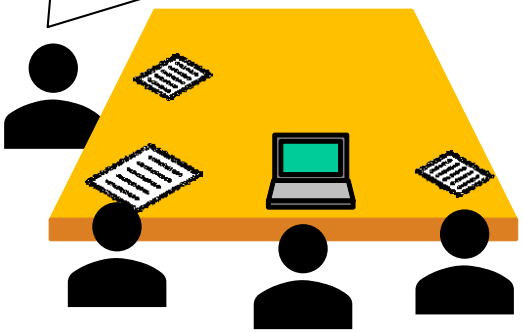


名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

レビューのコスト対効果

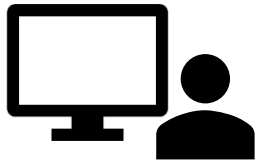
- レビューのコスト効果を得るためには、
  「レビュー実施コスト＜レビューにより省けたコスト」
  でなければならない。

ログ出力の “File could not open”は “File not found”のほうが適切ですね



レビュー

ログ出力の “File could not open”は “File not found”が適切だな。



テスト

名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

事例1: 不具合の修正工数の調査

- プロジェクト概要
  - 交通情報を中心としたセンサネットワークシステムのサーバサイドロジック
  - C/C++で330KLOC（流用込み）、約1年
  - 複数企業によるウォータフォール型開発
- バグレポート
  - 収集フェーズ：単体～総合試験
  - 記録バグ件数 約1,300
  - 組織内でバグと承認されたもの以外は入っていない。
- 調査方法
  - 修正工数が大きくなるような不具合の属性の組合せと対策を考える。

© 2020 Shuji Morisaki

事例1: 不具合情報(バグレポート)の属性

変数名	取りうる値
修正工数	修正に必要となった工数(人時)
混入工程	混入，修正，発見された工程 (基本設計，詳細設計，製造/単体試験，結合試験，総合試験)
発見工程	
修正工程	
機能	機能名(9機能のいずれか)
発見が遅れた理由	レビュー未実施，レビュー指摘もれ，修正確認漏れ，工程間引継ぎ，コミュニケーション不足，試験項目抽出もれ，テスト計画に含まれていない，環境が整わずテスト未実施，結果確認ミス
優先度	高，中，低
重要度	高，中，低
再現性	常に，たまに，一度だけ

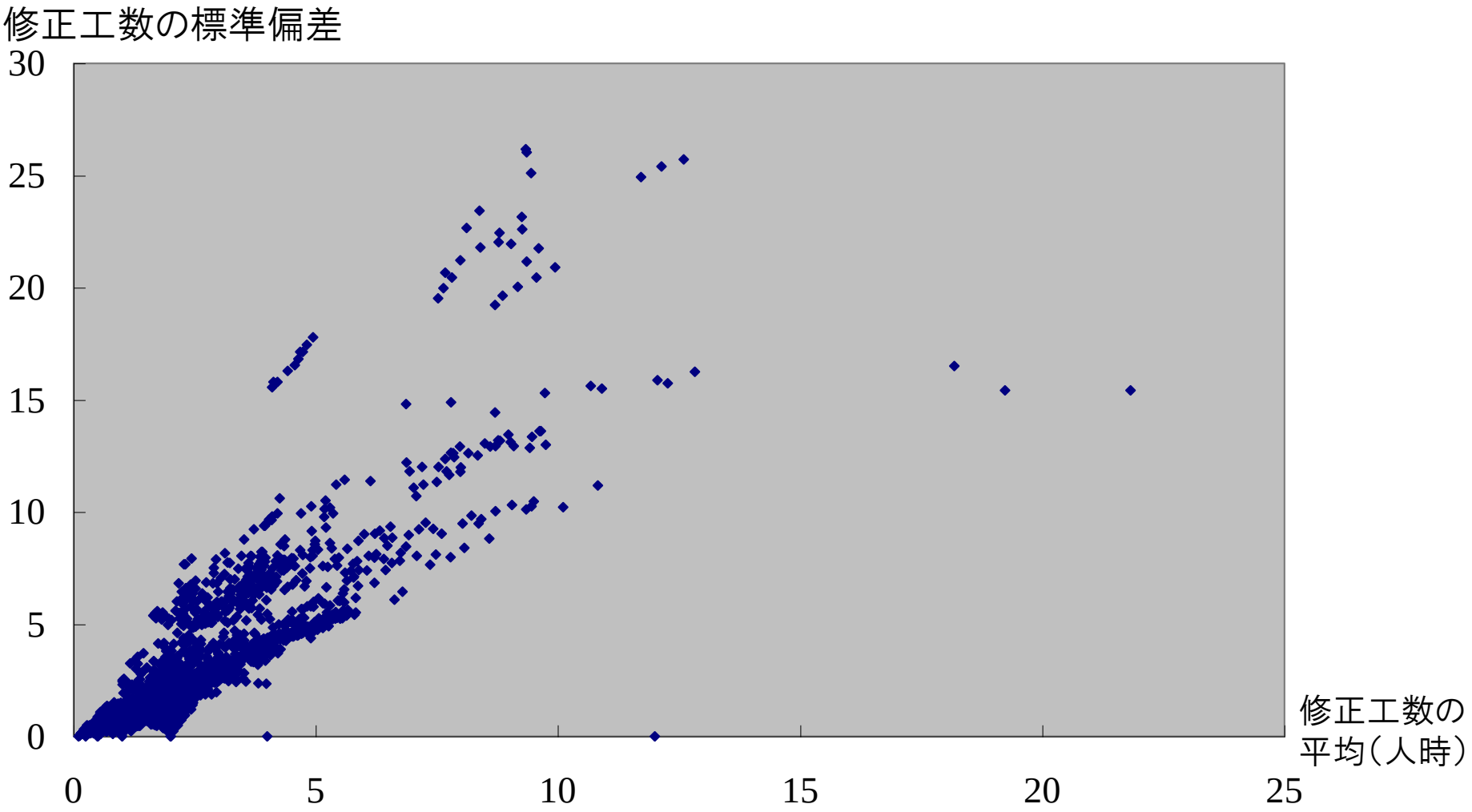
組合せの調査: (発見工程 = 総合試験) かつ (再現性 = たまに)
を満たす不具合は修正工数の平均 4時間、標準偏差=4.2

出典: 森崎、門田、玉田、松村、松本:”Defect Data Analysis Based on Extended Association Rule Mining”, Proceedings of International Workshop on Mining Software Repository, pp.17-24(2007)

名古屋大学 NAGOYA UNIVERSITY

© 2020 Shuji Morisaki

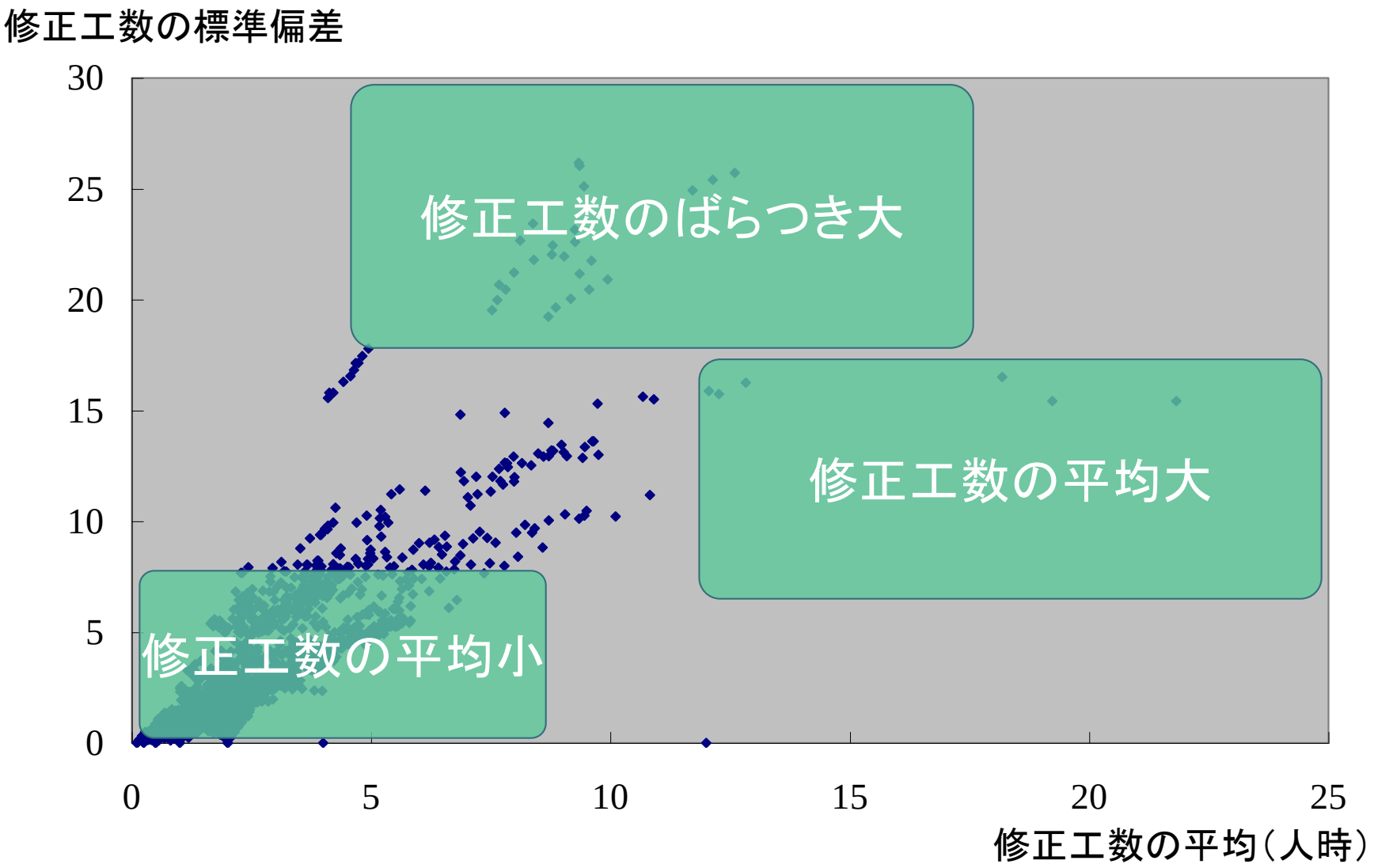
事例1: 不具合の属性の組合せ(6件以上の組合せ)



出典: 森崎、門田、玉田、松村、松本:”Defect Data Analysis Based on Extended Association Rule Mining”, Proceedings of International Workshop on Mining Software Repository, pp.17-24. http://se.naist.jp/~morisaki/publications/#i-200705

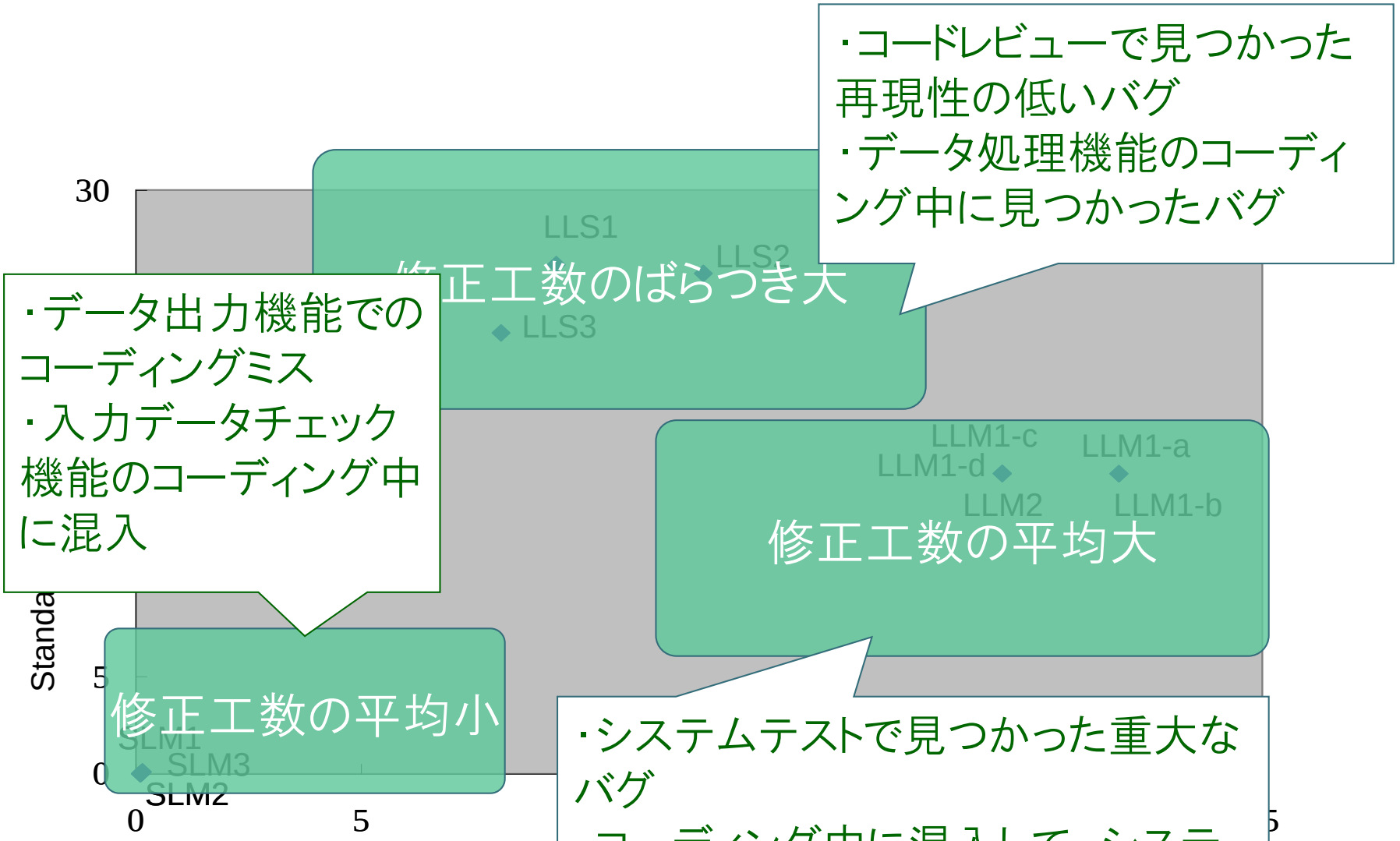
名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

事例1: 属性の組合わせの分布



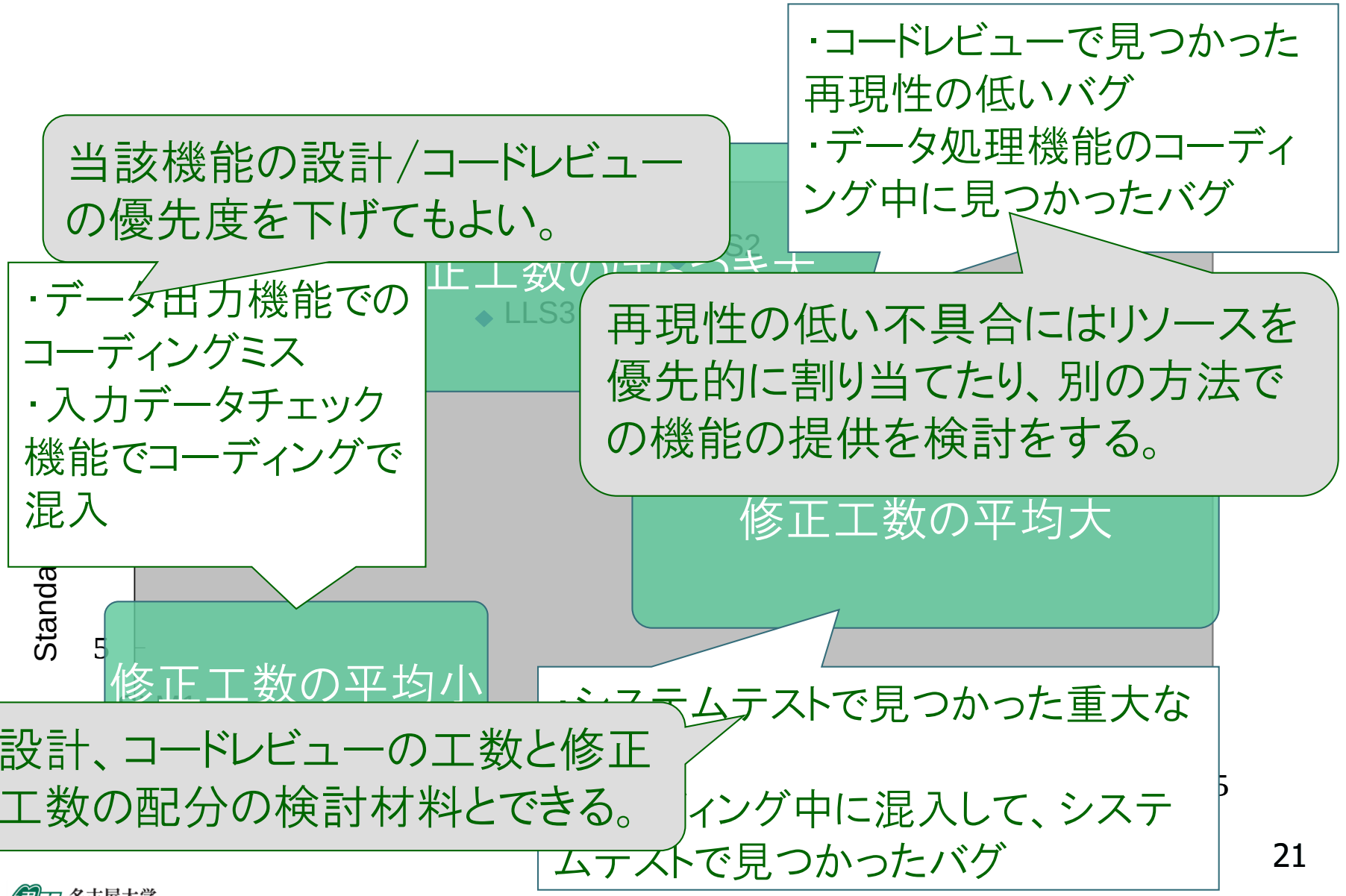
名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

事例1: 不具合の属性の組合せの例



名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

事例1: 次期開発での改善への活用例



名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

アジェンダ(再掲)

- 前提
  - 本セッションでのソフトウェアレビューの定義
  - 本セッションでお伝えしたいこと
- レビューの効果
  - 効果を高めるための原理
  - 不具合の修正工数の調査
- レビューで検出すべき問題
  - コンテキストに応じた問題種別
  - 問題種別と検出シナリオ
  - 問題種別の設定方法
- 効率化にむけて
  - レビューの局所化
  - 役割分担

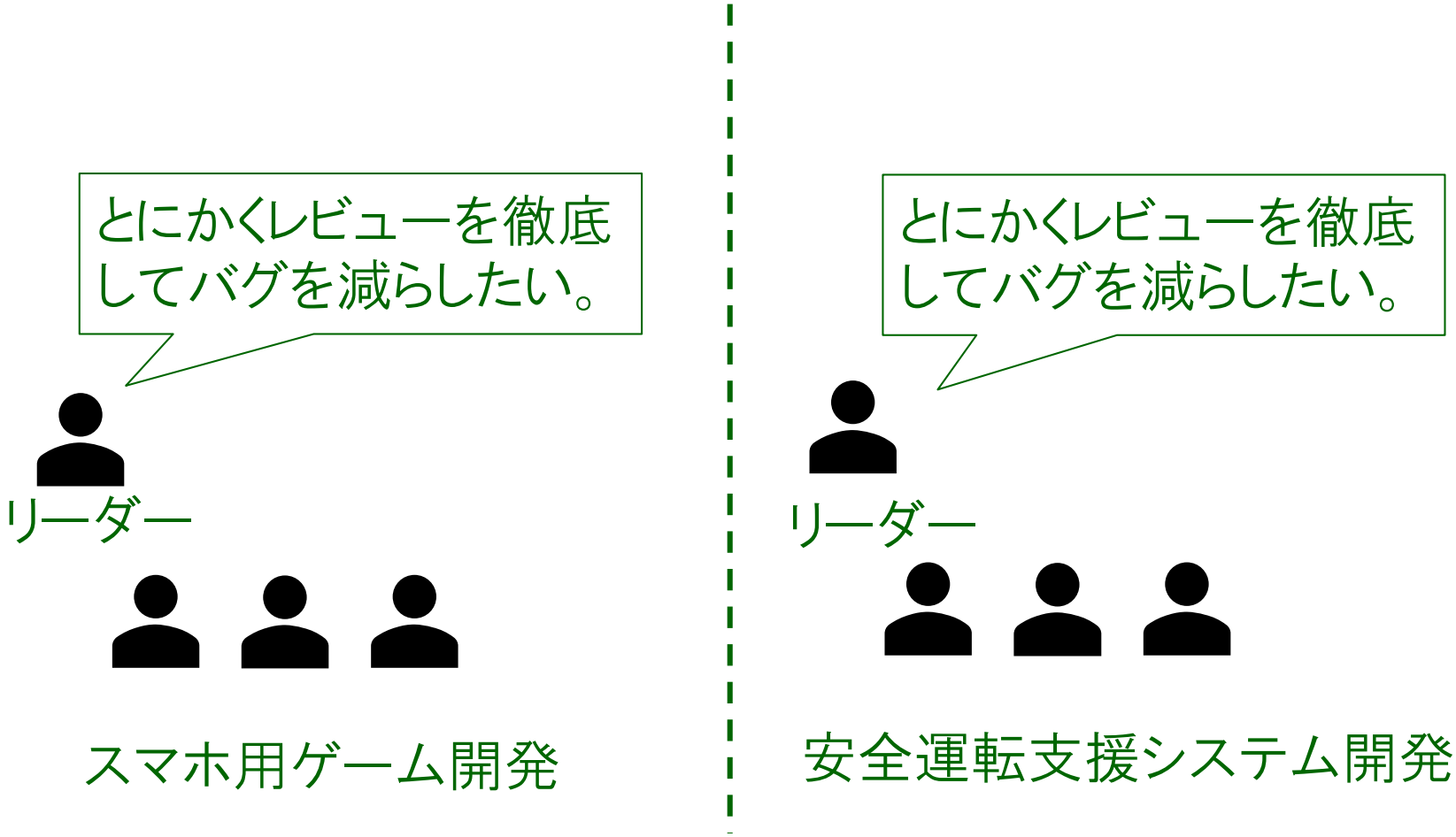
名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

レビューで検出すると効果の高い問題

- 「レビューの効果が高くなる問題を検出してください」と言われてもすぐには思いつかない。
- 典型的と言えるものは少ない。
  - 性能、リソース問題
  - 様々な部分に波及するデータ定義の変更
  - システムの前提や要求の考慮漏れ
  - セキュリティ
- 大半はコンテキストに依存する。
  - プロダクトに求められる品質
  - リリース形態（時期や回数）
  - プロダクトやプロジェクトのコスト吸収力
  - レビューアのスキル

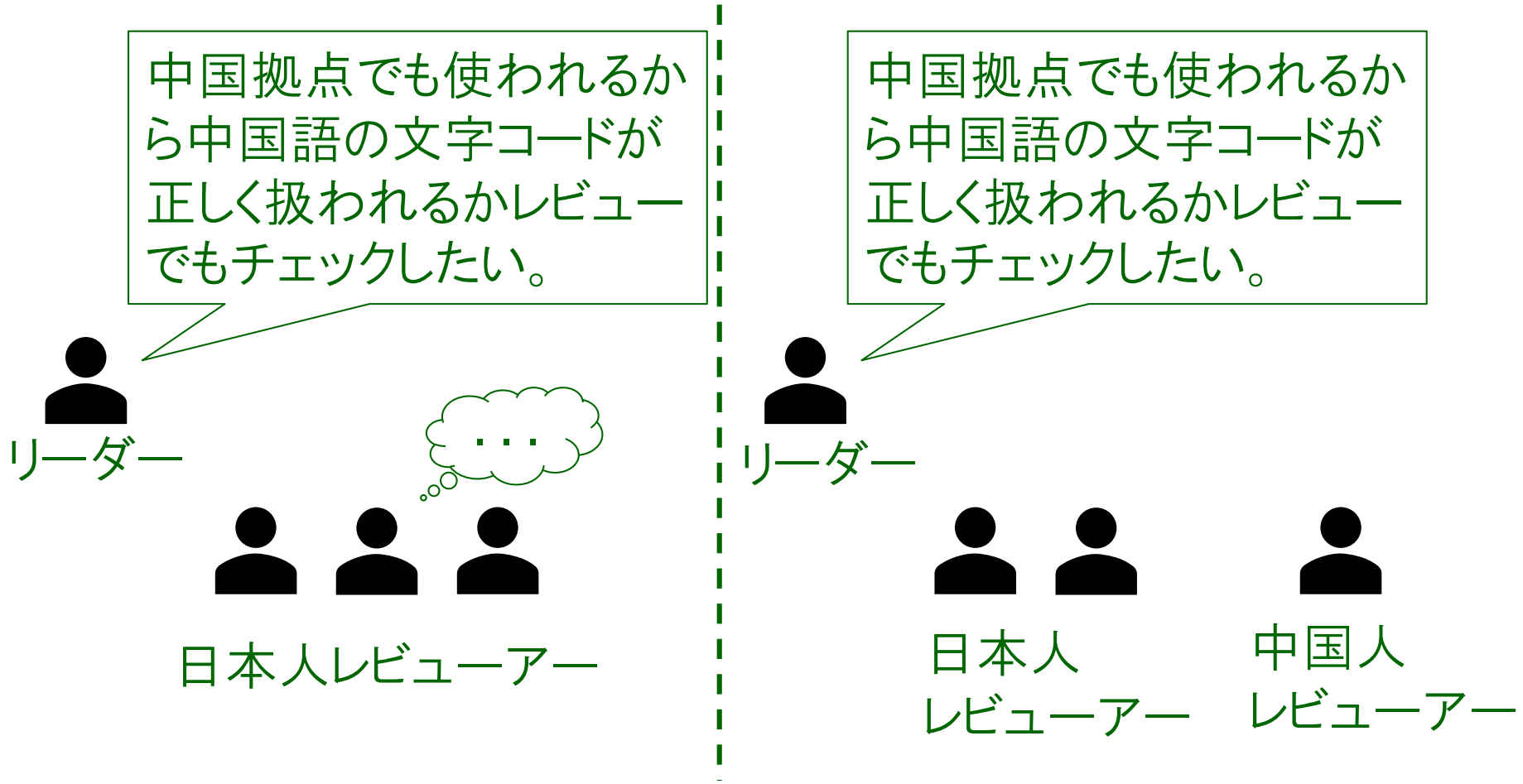
名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

コンテキストの考慮 – 求められる品質



名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

コンテキストの考慮 - レビューアーのスキル



名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

効果が高くなる問題と検出シナリオ

- 検出すべき問題種別
  検出して効果のある問題をタイプに(一般化)したもの
- 検出シナリオ
  設定した問題種別の問題をみつけるための大まかな手順

問題種別と検出シナリオの例

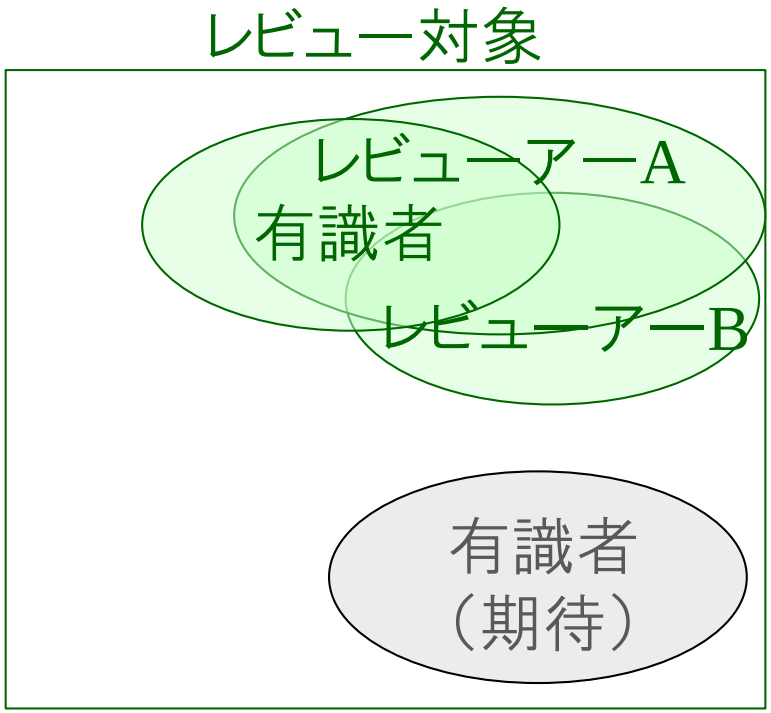
問題種別	検出シナリオ
連携システムとの不整合	連携システムのインタフェース仕様書を確認し、データ定義(データ種別やデータ長)と実行順序(実行の前提条件)の不整合がないか確かめる。
前提条件の誤り	インタフェース定義を確認し、記載されている振る舞いと入出力の前提が期待通りか確かめる。

出典: A. Porter, L. Votta and V. Basili, "Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment," IEEE Transactions on Software Engineering, Vol. 21, No. 6, pp. 563–575 (1995)

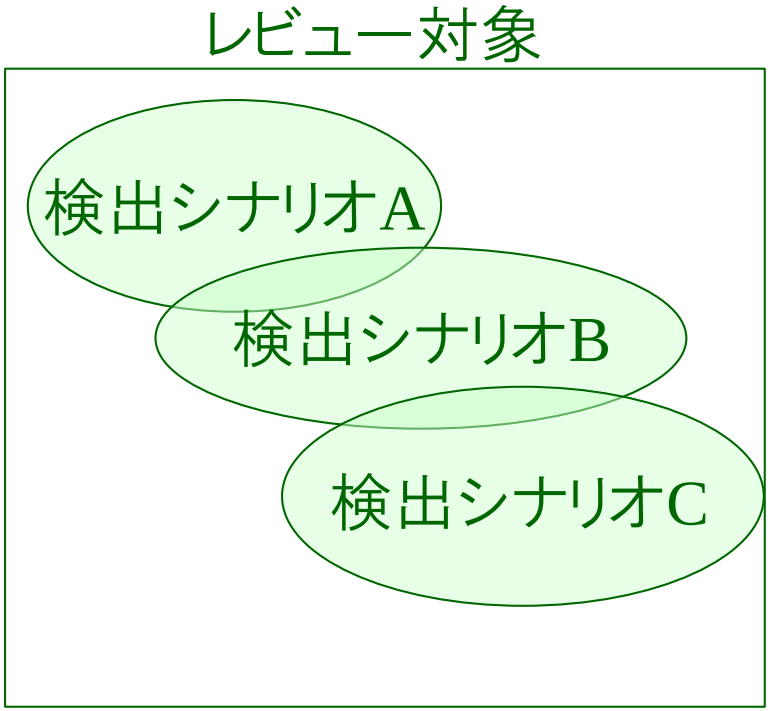
© 2020 Shuji Morisaki

問題検出のイメージ

- レビューでどのような問題を検出したいかを合意するために問題種別と検出シナリオを決めておく。



指示や合意をしないレビュー
（アドホックレビュー）



シナリオを使ったレビュー

名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

検出シナリオを使ったレビューのメリット

- 集中できるので漏れが少なく、適正な時間でチェックできる。
- 何が確かめられていて何が確かめられていないか明確になる。
- レビューアーのスキルが足りないときでも事前に割り切れたり、他の方法でカバーできる（最悪でも覚悟できる）。
- 検出シナリオをレビューアーで分担すれば、効率的になる。

名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

事例2: セキュリティ問題に限定したコードレビュー

- 目的: セキュリティ問題をみつけるための検出シナリオでレビューしてもらい、効果と効率を確かめる。
- 技術者向けイベントにおいて試行を実施した。
  - 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問題検出してもらう。
  - 100名程度の技術者に協力いただいた。

名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

事例2: セキュリティに限定したコードレビューの結果

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%
セキュリティ以外の指摘	91件（24.6%）
レビュー速度（行/時）	3,860行 （Technical review, 問題検出、Java、コメント込み）

既存文献でのレビュー速度（参考）

文献名	公開年	速度（行/時）
Fagan M. “Design and code inspection to reduce errors in program development”, IBM Systems Journal vol. 15, no. 3	1976	700～900行（問題検出）、 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）

© 2020 Shuji Morisaki

検出すべき問題の設定方法

- レビューで検出すべき問題はコンテキストによって異なる。
- 類似問題から設定
  - 過去に検出された問題から決める。
    - レビューで検出された問題
    - レビューで見逃され、テストで発見された不具合
  - アーキテクチャ、要素技術、体制で起こりがちな問題を想定して決める。
- 保証対象から設定
  - 対象システムに求められることや対象システムの価値を損なう問題がないか確かめる。

名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

過去に検出された問題から設定

- 類似の問題を探して一般化する。
- 一般化した問題を検出すべき問題種別とし、それを検出するためのシナリオを作る。

例）オンラインショッピングサイトの過去の設計レビューで指摘された問題のうち「カード」を含む指摘

指摘ID	指摘内容
042	Webから注文された場合の**カード**決済用のタイムゾーンが定義されていない
049	**カード**決済時の分割払のオプションがない
081	16桁以外の**カード**番号が想定されていない

森崎 修司，間違いだらけの設計レビュー，日経BP(2015)

名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

過去に検出された問題から設定

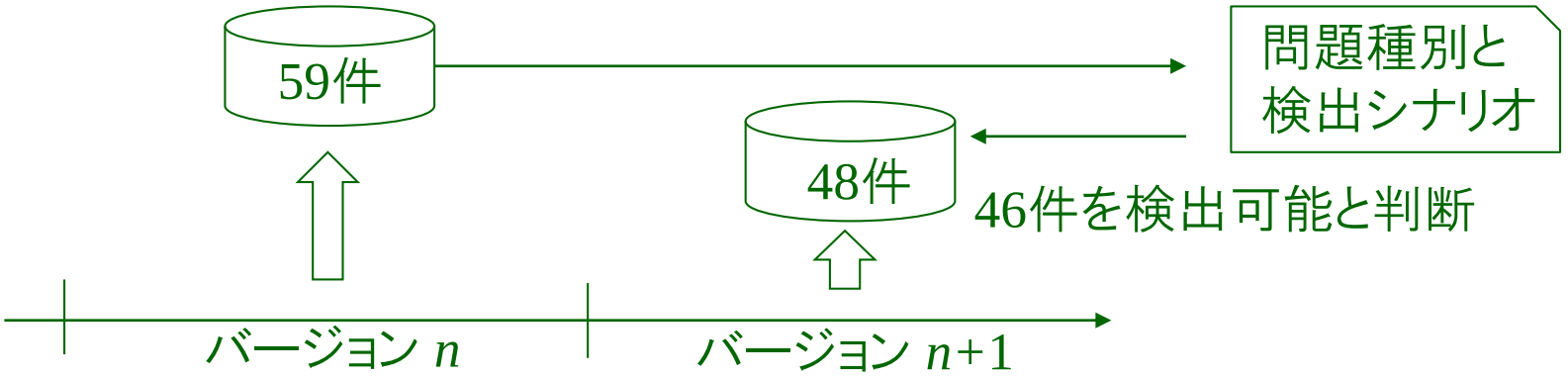
- 例）過去の指摘リストのうち「カード」を含む指摘
  - 049と081から問題種別「カード決済に必要な情報や条件が揃っていない」を設定する。

指摘ID	指摘内容
042	Webから注文された場合の**カード**決済用のタイムゾーンが定義されていない
049	**カード**決済時の分割払のオプションがない
081	16桁以外の**カード**番号が想定されていない

名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

事例3: 過去の指摘リストから検出シナリオ作成

- 対象: 生産システムの要求仕様書のレビュー指摘
- 深刻度「大」の問題59件を一般化した。
  - 頻出キーワード(「カウンタ」「イベント」)を含む指摘から9件の検出シナリオを作成した。
- 過去の指摘48件(深刻度「大」)が検出できていたかを確認した。
  - 46件が検出可能と判断した。



出典: K. Kasubuchi, S. Morisaki, A. Yoshida and C. Ogawa, An empirical evaluation of the effectiveness of inspection scenarios developed from a defect repository, IEEE International Conference on Software Maintenance and Evolution, pp. 439-448(2015)

名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

事例4: 不具合情報を使った問題種別の設定

- プロジェクト
  - デジタルメディア処理ソフトウェア開発
  - C/C++で650KLOC（流用込み）、約3.5年
  - 派生開発
- 不具合
  - 発見フェーズ 単体～総合試験
  - 分析対象不具合件数 約350件
  - バグ以外の要求事項等は含まれない。

出典
森崎, レビュー効率化にむけた産学連携の取組み, ソフトウェアプロセス改善カンファレンス 2009
http://www.jaspic.org/event/2009/SPIJapan/session4A/4A1.pdf
森崎 修司, 久保 匡, 荻野 利彦, 阪本 太志, 山田 淳, 過去の不具合の修正工数を考慮したソフトウェアレビュー手法, 電子情報通信学会論文誌 D Vol.J95-D No.8 pp.1623-1632(2012)

名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

事例4: 対象とした不具合管理票の項目

変数名	値
修正時間	修正に必要となった時間（手動で入力）
担当者	テスト担当者名（18名）
サブシステムカテゴリ	5種類（ミドルウェア、アプリケーション等）
サブシステム名	53種類（ミドルウェア、機能名等）
原因種別	設計ミス、設計漏れ、調査漏れ、調査ミス、外部設計漏れ、外部設計ミス、内部設計漏れ、内部設計ミス、コーディング漏れ、コーディングミス
修正期間	解決日と発生日の差（休日含む）
検査種別	正常系、異常系、限界系、その他
発生種別	新規バグ、内在バグ、既知バグ、その他

© 2020 Shuji Morisaki

事例4: 抽出した不具合属性の組合せ(抜粋)

- 該当する不具合件数、修正期間の平均、標準偏差が大きいものに着目

ルール	該当件数	修正期間平均(対全体平均)	修正期間標準偏差(対全体標準偏差)
MW(P)サブシステムのバグで内部設計ミスが原因	3.8%	6.62 (1.26)	1.26 (0.73)
担当-015 が検出した異常系のテストでみつかったバグ	3.5%	6.33 (1.32)	1.50 (0.73)
原因種別が内部設計漏れのバグ	10.2%	5.68 (1.18)	2.09 (0.62)
サブシステムMW(F) で異常系のテストで見つかったバグ	4.4%	5.67 (1.18)	2.29 (1.02)
サブシステムFW-00 か MW(P)-001	14.0%	5.62 (1.17)	2.11 (1.04)

出典: 森崎, レビュー効率化にむけた産学連携の取組み, ソフトウェアプロセス改善カンファレンス 2009
http://www.jaspic.org/event/2009/SPIJapan/session4A/4A1.pdf

名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

事例4: 属性の組合せから設定した問題種別

- 注目した属性の組合せに当てはまるバグの現象説明欄等から問題種別を設定

⇩

設計レビューで優先して検出すべき問題種別(優先問題種別)として表中色塗りの項目を選定

問題種別	件数
例外処理	20
パラメータ初期化/クリア	8
特殊入力データ/パラメータ	8
排他処理	5
内部シーケンス	5
ユースケース	4
アルゴリズム	3
バッファオーバーフロー	2
データ領域管理	2
状態遷移	2
条件分岐	2
IF(戻り値)	2
操作ミス	1

© 2020 Shuji Morisaki

事例4: 試行結果

- 設定した検出シナリオにより予想修正工数が大きくなった。

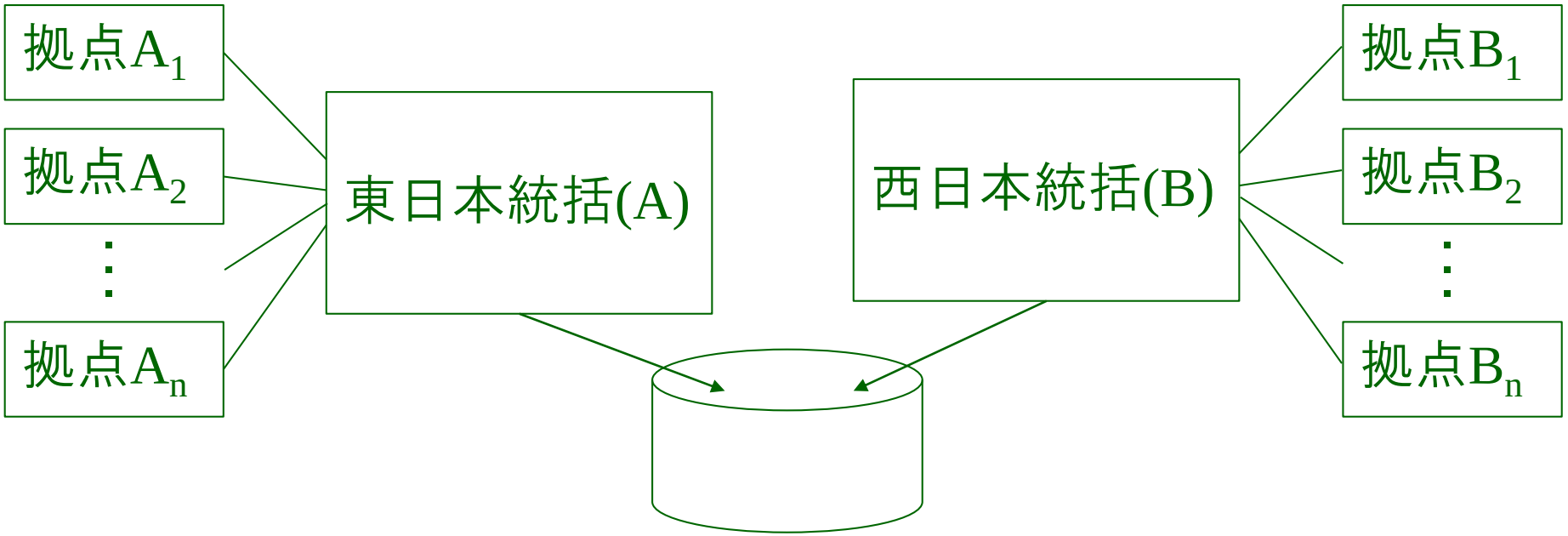
	試行1（サブシステム1）		試行2（サブシステム2）	
	提案手法	通常	提案手法	通常
レビュー工数（人時）	3.0	6.0	7.7	9.0
検出問題数の合計	8	6	11	4
優先問題種別に該当する件数	6	(2)	10	(3)
問題を見逃した場合の修正工数の合計（見積り: 人日）	15.5	6.5	14.4	5.0

出典1: 森崎，久保，荻野，阪本，山田，過去の不具合の修正工数を考慮したソフトウェアレビュー手法，電子情報通信学会論文誌 D Vol.J95-D No.8 pp.1623-1632(2012)
出典2: 検証II 定量的な観点絞り込みの効果 http://itpro.nikkeibp.co.jp/article/COLUMN/20120808/415155/

名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

類似のアーキテクチャから問題種別を設定

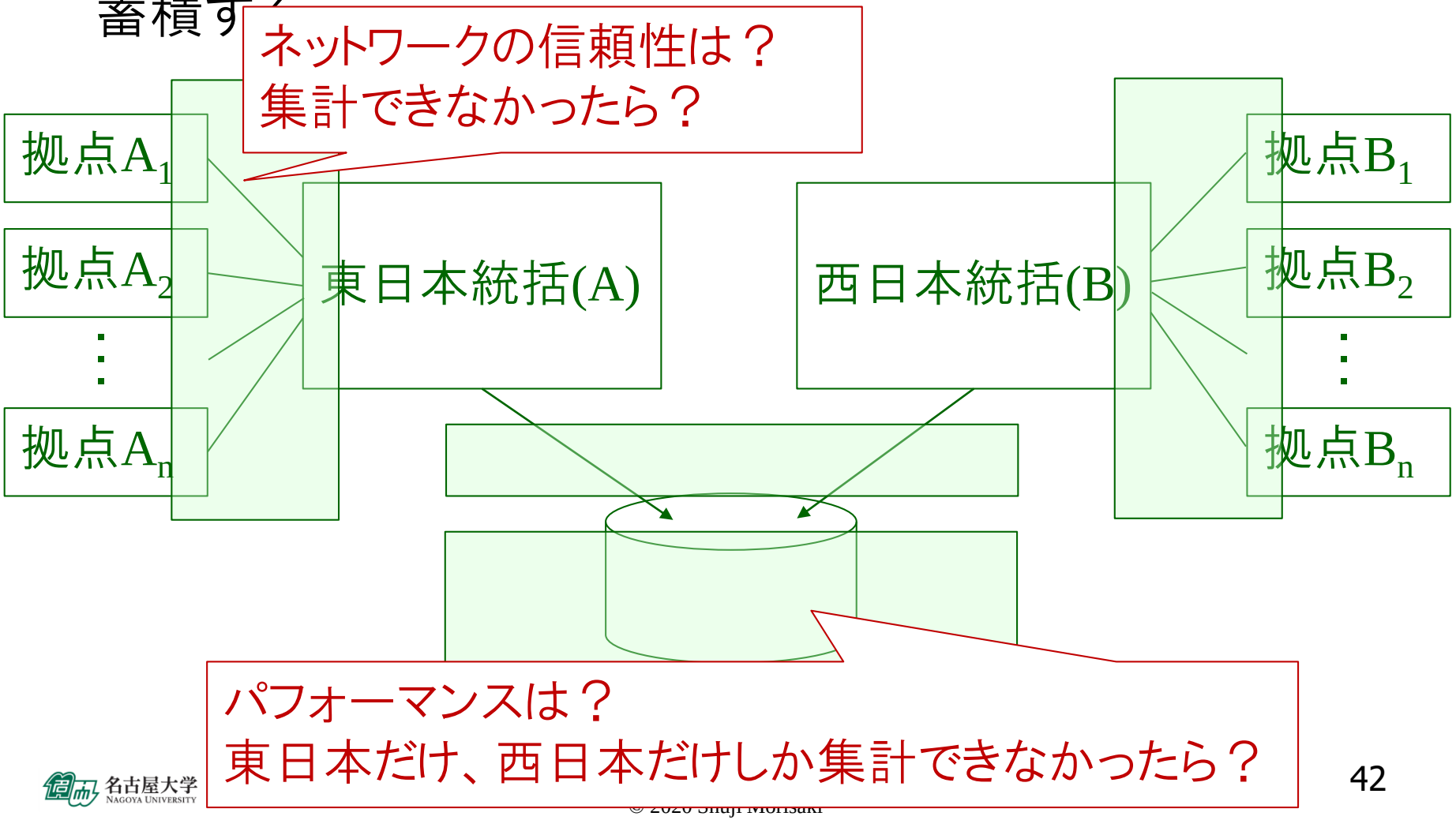
- 前日の売上げから翌日の出荷量を決めるシステムがある。
- 東日本統括サーバ、西日本統括サーバで集めた日次の集計結果を集約してデータベースに蓄積する。



© 2020 Shuji Morisaki

類似のアーキテクチャから問題種別を設定

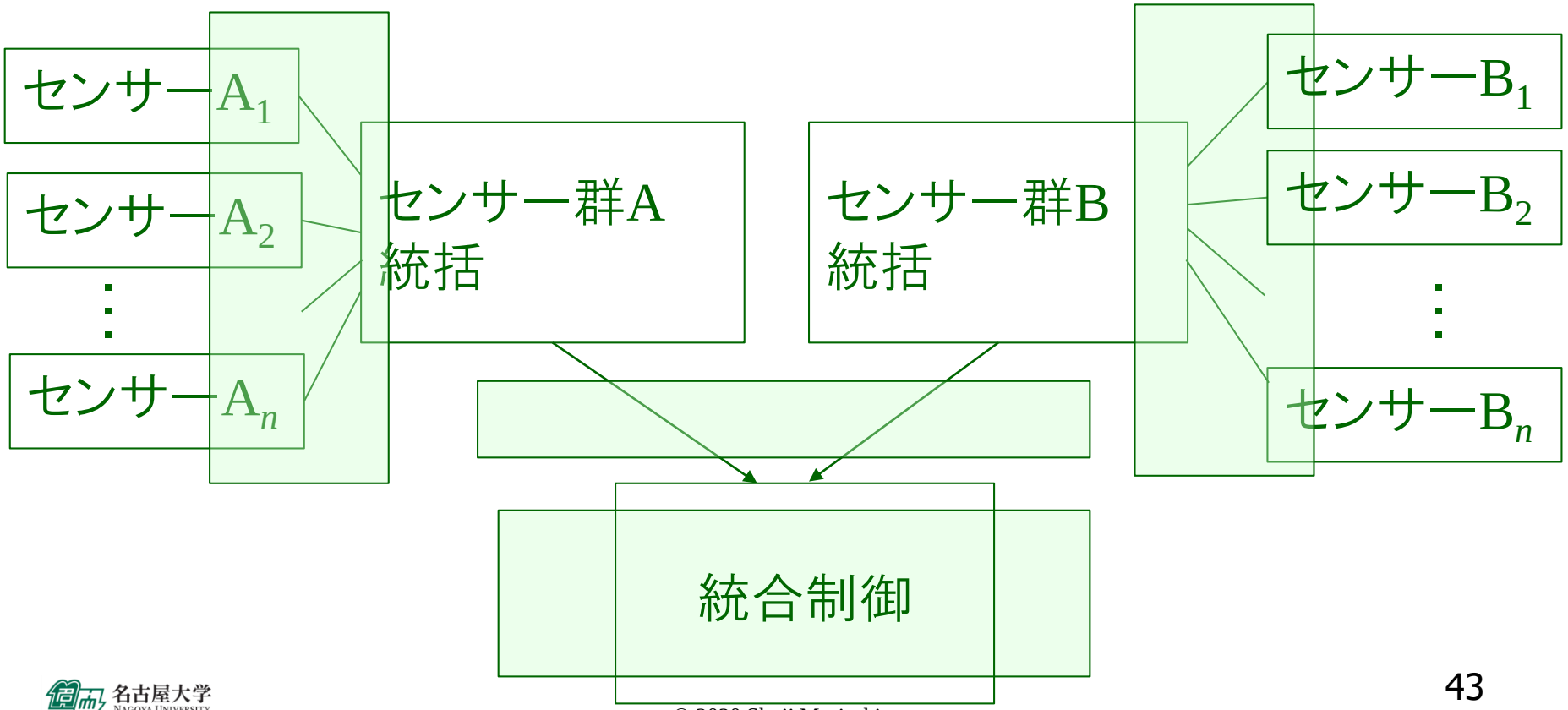
- 東日本統括サブシステムAで集めた日次の集計結果と西日本統括サブシステムBで集めた集計結果をデータベースに蓄積する



© 2020 Shuji Morisaki

類似のアーキテクチャから問題種別を設定

- リアルタイム性が高いセンサー群Aとリアルタイム性が中程度のセンサー群Bの情報を使って、認識をする。
- 確認すべきところは前のページまでの売り上げシステムと似ている。



© 2020 Shuji Morisaki

保証対象から問題種別を設定

- ゴール指向(Goal-oriented) レビュー
  - ゴール指向要求分析を元にしている。
    個々の要求には達成したい目的があり、目的を整理することで要求の漏れを減らし、優先順位をつけやすくなる。
  - ゴールを保証対象として問題種別を設定する。
- 目的の例
  - オンラインショッピングサイト
    - 短時間で買い物する。
    - 実店舗に移動しなくても買いたい。
    - 実店舗と比較して在庫切れが少ない。
    - 他のサイトと値段を比較しながら、決めたい。

M. Wakimoto, S. Morisaki, A Case Study of Requirements Ambiguities and Goal-oriented Focused Requirements Specification, In proc. of 4th International Conference on Enterprise Architecture and Information Systems, pp. 1-6 (2019)

© 2020 Shuji Morisaki

保証対象の設定例

- 対象: 架空の公共交通機関の座席予約システム
- システムの特徴
  - インターネット、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。
- システムが保証すべきこと
  - 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。

出典: 森崎 修司, 間違いだらけの設計レビュー 改訂版 (2015)

名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

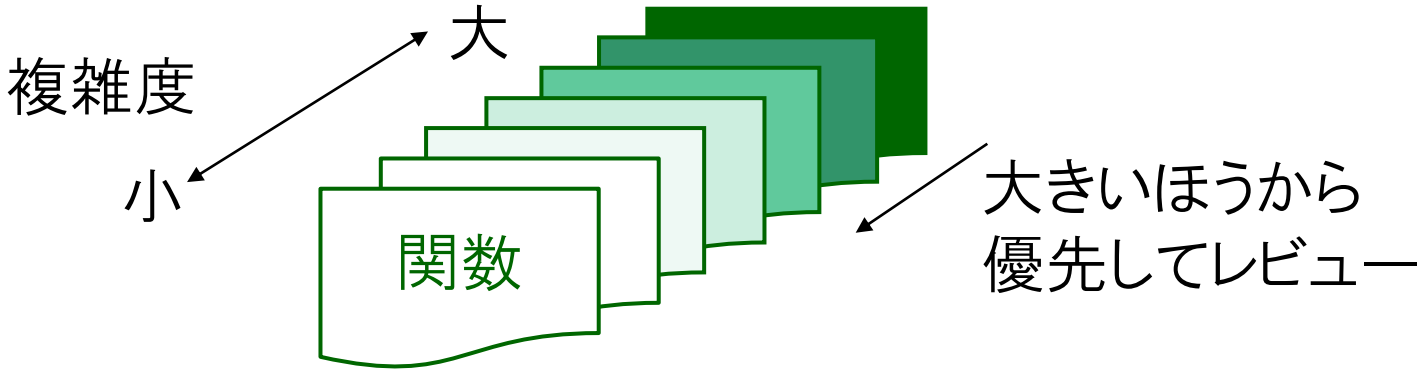
アジェンダ(再掲)

- 前提
  - 本セッションでのソフトウェアレビューの定義
  - 本セッションでお伝えしたいこと
- レビューの効果
  - 効果を高めるための原理
  - 不具合の修正工数の調査
- レビューで検出すべき問題
  - コンテキストに応じた問題種別
  - 問題種別と検出シナリオ
  - 問題種別の設定方法
- 効率化にむけて
  - レビューの局所化
  - 役割分担

名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

事例5: ソースコードメトリクスを用いた局所化

- 分岐等で複雑なモジュール(関数)を選んでレビューを重点化する。
  - 複雑なモジュールをソースコードメトリクスで選ぶ。
  - ソースコードメトリクスはツールで計測できる。
- 通信システム(商用)のソースコードを対象とし、全体の40%の関数を重点化すれば欠陥を検出できることを確かめた。

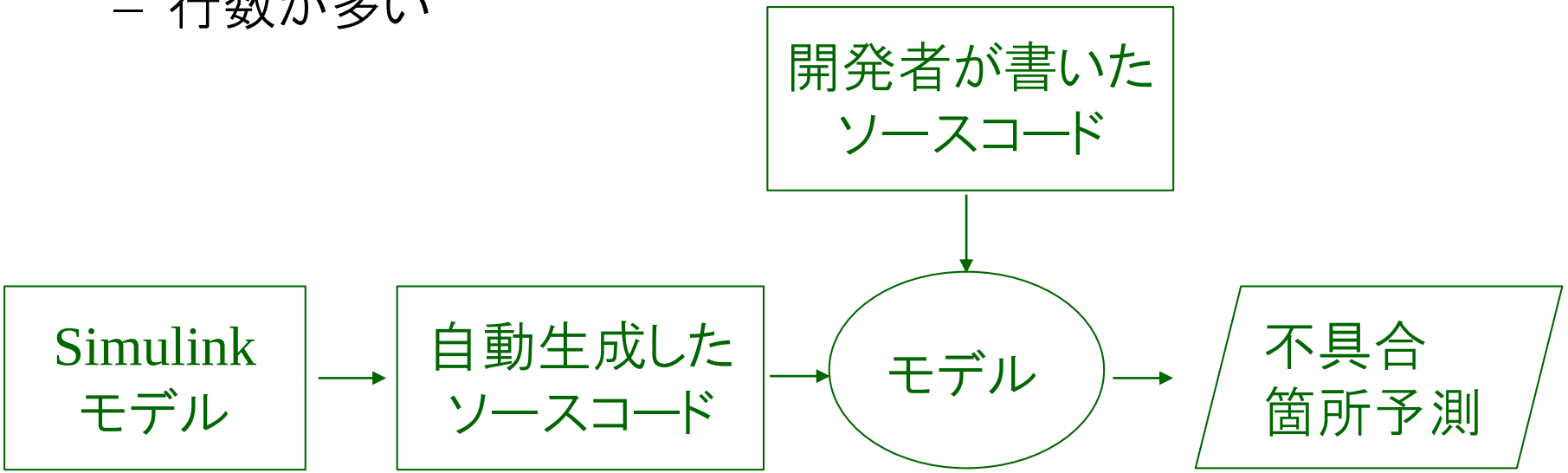


出典: N. Kasai, S. Morisaki, and K. Matsumoto, Fault-Prone Module Prediction Using a Prediction Model and Manual Inspection. In *Proceedings of the 2013 20th Asia-Pacific Software Engineering Conference* , *pp.* 106-115 (2013)

© 2020 Shuji Morisaki

事例6: 設計モデルのメトリクスを用いた不具合予測

- Simulinkモデルで自動生成したソースコードのメトリクスを計測して、不具合箇所を予測する。
- 開発者が書いたソースコードで学習した統計モデルを使って、自動生成したソースコードの不具合の予測ができた。
  - Halsted Volumeが大きい
  - 行数が多い



出典: A. Harald, State of the Art Software Development in the Automotive Industry and Analysis upon Applicability of Software Fault Prediction, PhD Thesis Graz University of Technology (2016)

名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

事例7: 過去の不具合の調査

- 目的: 業務に特化した問題種別が設定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

© 2020 Shuji Morisaki

事例7: 対象不具合情報の属性(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

名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

事例7: 業務(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)

名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

事例7: 不具合の一般化の支援にむけて1

- 自由記述に含まれる単語により、日付に関する不具合を識別できるか適合率と再現率を求めた。
  - 不具合情報の「現象」「原因」「対策」欄を対象とし、日付に関する単語「日」「期間」「期限」を少なくとも1つ含むものとする。
  - 適合率: 84.9% $= \frac{|\text{自由記述欄に「日」「期間」「期限」を含む不具合}|}{|\text{目視で確認した日付に関する不具合}|}$
  - 再現率: 68.9% $= \frac{|\text{目視で確認した日付に関する不具合}|}{|\text{自由記述欄に「日」「期間」「期限」を含む不具合}|}$

名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

事例7: 不具合の一般化の支援にむけて2

- 不具合情報の各欄に含まれた名詞のうち上位20件を集計した。
  - 日付以外にも業務観点の欠陥種別を特定できるだろうという熟練者の見解を得た。

「現象」欄			「原因」欄			「対策」欄		
順位	単語	件数	順位	単語	件数	順位	単語	件数
6	日	88	8	金	34	13	ファクター	47
9	取引	61	9	残	33	13	拠点	43
15	銘柄	48	10	拠点	34	16	異動	32
19	残高	44	11	解除	30	18	通貨	31
19	検閲	44				20	解除	30

名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

さらなる情報

- 書籍、セミナー
  - 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）
  - セミナー「間違いだらけの設計レビュー」（2020/5予定）
- 記事
  - 連載「ソフトウェアレビュー入門」
    連載「レビューがうまくいかない理由」
    @IT（アットマークアイティ）
    http://www.itmedia.co.jp/keywords/software_review.html http://www.atmarkit.co.jp/ait/series/6367/
  - 連載「失敗しない設計レビュー」システムの価値から問題種別の設定
    日経SYSTEMS 2018年4～9月号
  - 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/

名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki

まとめ

- 品質保証活動においてどのような問題を確かめるべきかレビューを題材に紹介した。
  - 不具合の修正工数の調査結果をつかって、レビューにおいて検出することでコスト効果が高まる問題を紹介した。
  - テストで見つかる欠陥のうち、どのようなものをレビューで見つけておくべきか事例で紹介した。
- レビューで検出すべき問題をどのように設定するかを紹介した。
  - 類似問題と事例2件
  - 保証すべき点
- 効率化を目指したレビューの局所化と役割分担の事例を紹介した。

名古屋大学 NAGOYA UNIVERSITY
© 2020 Shuji Morisaki