ソフトウェア欠陥メトリクスの意味と活用方法を考える

東洋大学経営学部　野中　誠

組込みシステム技術に関するサマーワークショップSWEST15

2013年8月23日

自己紹介

野中　誠

- 所属
  - 東洋大学 経営学部 経営学科　准教授
- 背景
  - 工業経営／経営システム工学，ソフトウェア工学，品質マネジメント
- 主な学外活動
  - 日本科学技術連盟 SQiPソフトウェア品質委員会 委員長
  - 日本SPIコンソーシアム 外部理事
  - （旧）IPA/SEC 定量的管理基盤WG 主査，定量データ分析WG委員（『データ白書2012-2013』）
  - 組込みシステム技術協会実装品質強化WG メンバー
  - 情報サービス産業協会SPES企画WGエキスパート
  - 国立情報学研究所 特任准教授 （トップエスイー講座，メトリクス講義担当）
- 主な著書
  - 野中誠・小池利和・小室睦著 『データ指向のソフトウェア品質マネジメント』 日科技連出版社 (2012)
  - 野中誠・鷲崎弘宜訳 『演習で学ぶ ソフトウェアメトリクスの基礎』 日経BP社 (2009)
- 研究教育
  - ソフトウェア品質マネジメント（メトリクス）
  - 情報システムと経営の関わり
- 大学での担当授業
  - 情報管理論，品質マネジメントなど

データ指向のソフトウェア品質マネジメント（デート本）

メトリクス分析による「事実にもとづく管理」の実践

日経コンピュータ NIKKEI COMPUTER

2012年11月8日号
p.125　書評

ソフトウエア品質メトリクスで、日本における気鋭の研究者である野中誠氏による解説書。
ソフトウエアの品質データ（メトリクス）は、その収集自体が目的化してはならないが、その分析は開発現場ではなくてはならないものである。
本書はその収集・分析方法、用いるべきモデルなどを、統計学の初歩的解説とともに述べている良書である。

日経SYSTEMS システム開発の現場力を磨く

2012年11月号
p.110　書評

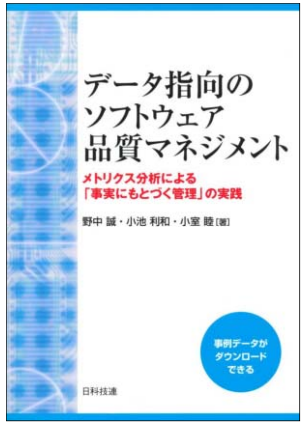
ソフトウエアの品質を高めるには、開発の過程で問題をいち早く把握し、適切な対策を打つことが不可欠である。それには、問題を可視化することが大事だ。
本書は、事例を基に、品質データの測定項目や分析手法を解説している。ソフトウエア品質に関わる多様なデータを測定し、分析を繰り返すことは容易ではないが、分析手順を詳しく説明している。
事例を読者自身の開発に当てはめて実践してみれば、必要な知識やスキルを身に付けやすいだろう。

- 品質メトリクス分析の実践的入門書
- 11の分析事例
- データと分析手順の解説をダウロードできる（同じ分析手順を手元で行える）

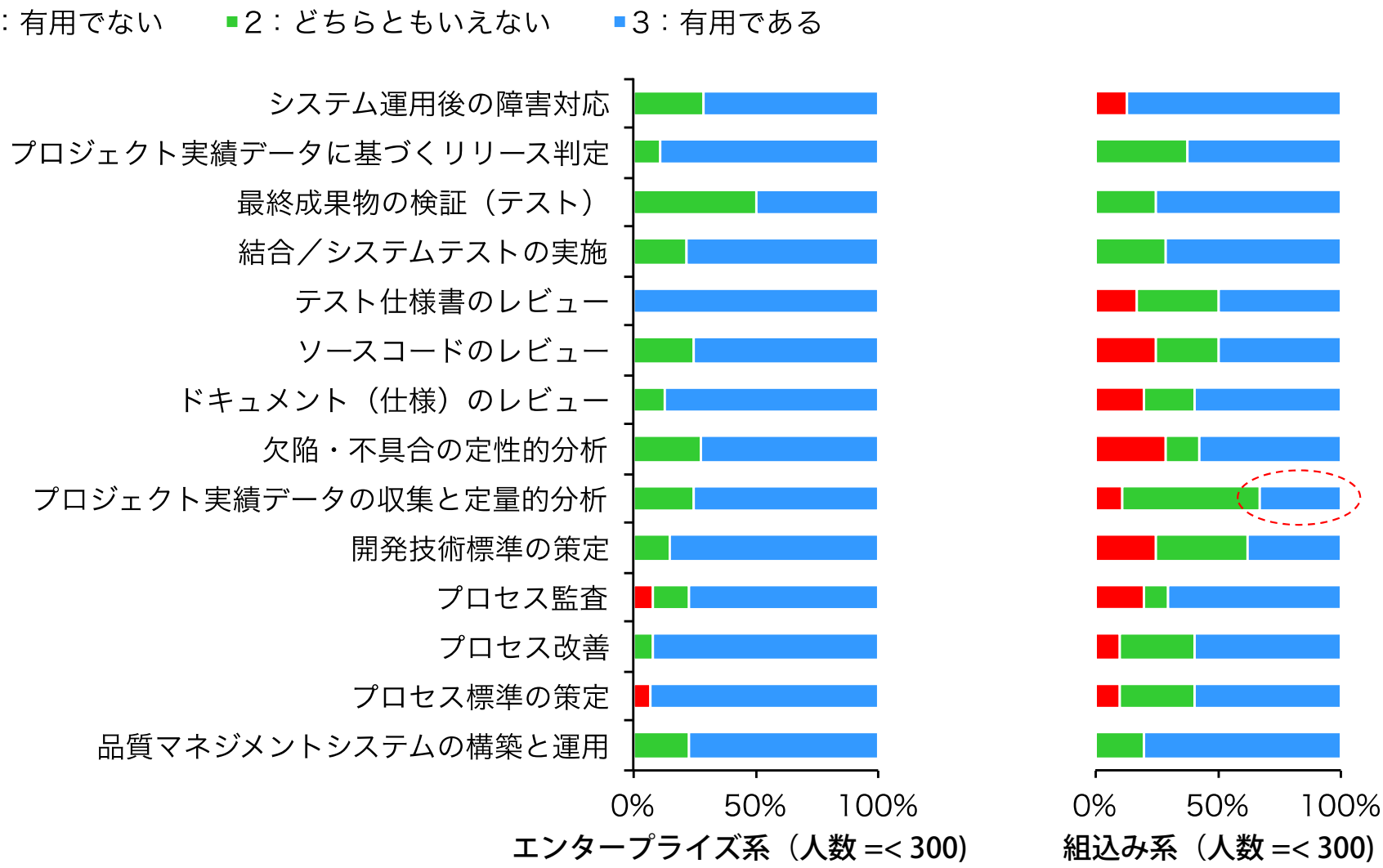
主要目次

野中　誠（東洋大学）
小池 利和（ヤマハ）
小室　睦
（元 日立ソリューションズ、
現 富士フイルムソフトウエア）

2012年9月19日発行
定価　3,780円（税込）
ISBN978-4-8171-9447-3



品質部門の支援業務に対する、開発部門の評価（開発対象別）



- 組込み系で、「プロジェクト実績データの収集と定量的分析」の有益度評価が低い
- 品質部門が集めた実績データを、開発部門にうまくフィードバックできていないのが原因？

市場流出欠陥のベンチマークデータ

日・米・欧・印のパフォーマンス比較 （Cusumano *et. al.*, 2003）

項目	インド	日本	米国	欧州他	合計 or 平均
調査対象のプロジェクト数（件）	24	27	31	22	104（合計）
新規コード行数（KLOC, 中央値）	209	469	270	436	374
出荷後の欠陥密度（中央値）	0.263	0.020	0.400	0.225	0.150

皆さんの組織での市場流出欠陥は、このデータに比べてどうですか？
そもそも「出荷後の欠陥密度」は、信頼できるメトリクスですか？

Cusumano, M., *et. al.,* "Software Development Worldwide: The State of the Practice," *IEEE Software*, Nov/Dec 2003, pp.28-34.

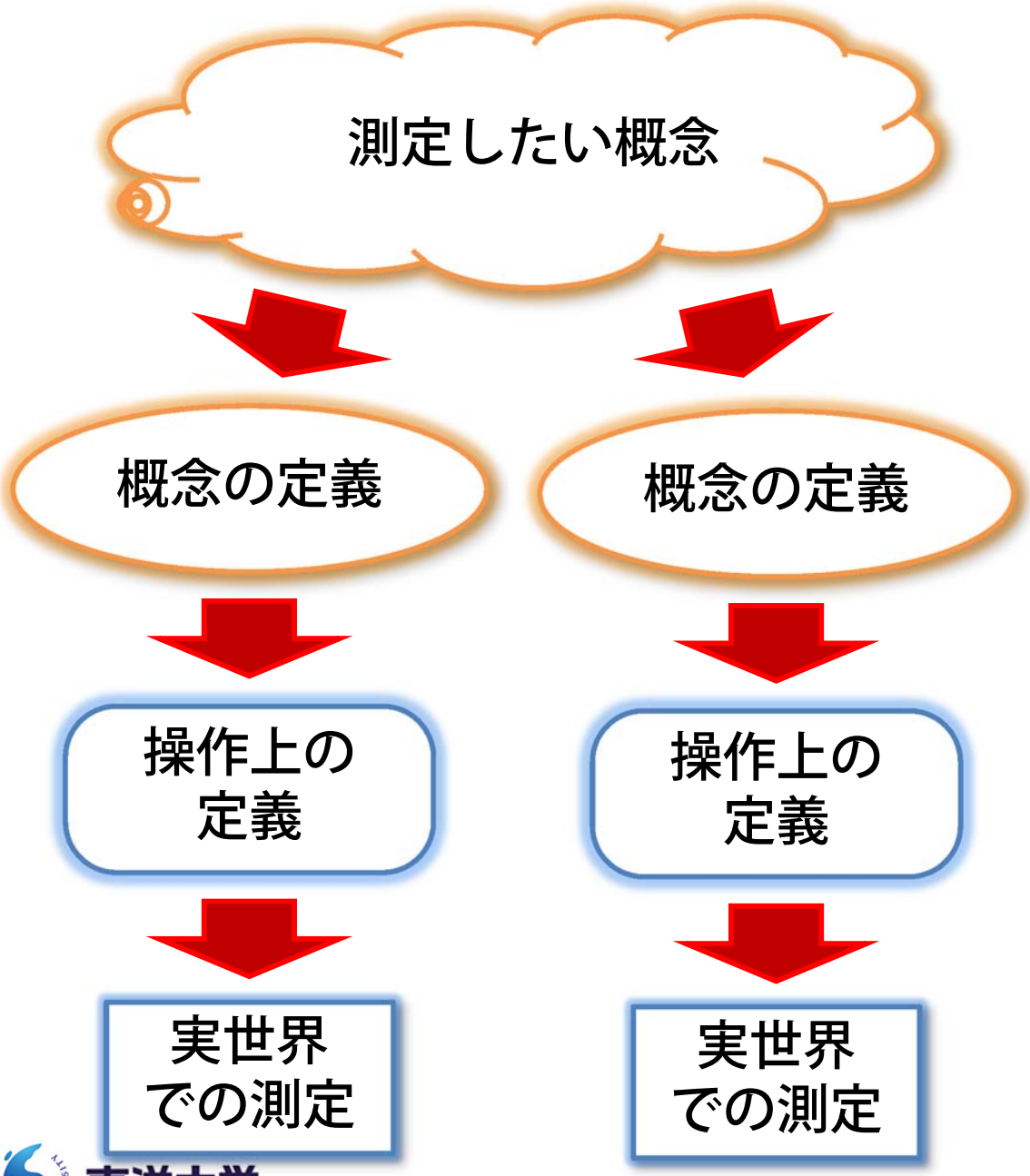
なぜ定量的管理が必要なのか？ Informed decision指向

- 「必要な情報」を得た上での意思決定のために
  ソフトウェア／プロジェクトといった抽象的・複雑な対象について、
  そのマネジメントにおける様々な意思決定に役立つ情報を、
  一貫した方法で獲得し、論理的な判断を下し、リスクを把握し、
  そして、前へ前へと進んでいきたい

- 「必要な情報」の獲得・伝達手段として
  - 主観的情報も意思決定の際に用いている（説明の論理性を判断）
  - 主観的情報は、情報収集の一貫性、効率性、説明性に問題がある
  - メトリクスを定めることで、必要な情報を、効果的に、一貫して収集できる
  - 組織やメンバーに説明可能な根拠情報として利用できる

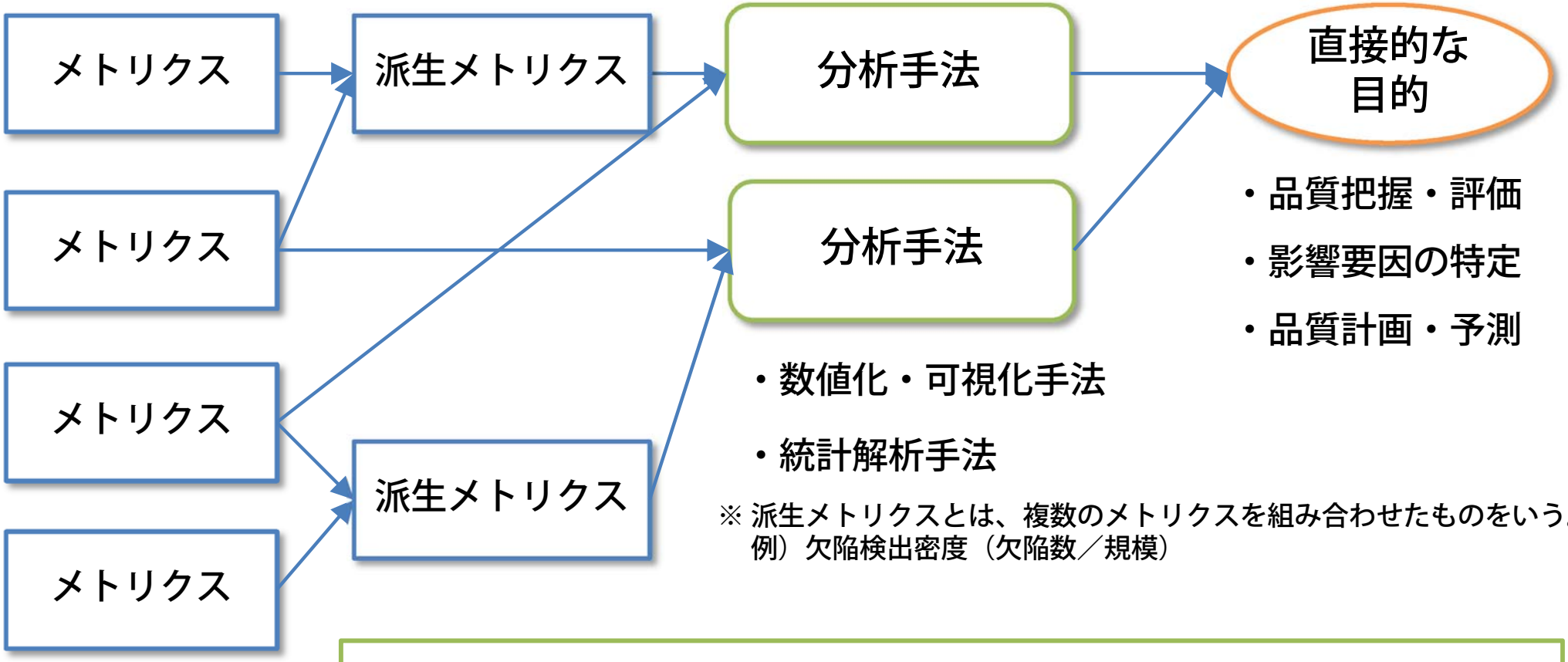
定量的管理に取り組まない合理的理由は、限定的である

メトリクスの限界を知り、informed decision に役立てる



- 実世界で測定できることは、測定したい概念の一部に過ぎない
- 複数のメトリクスで、測定したい概念を多面的に捉える

目的に合った分析手法と適切なメトリクスを用いる



※ 派生メトリクスとは、複数のメトリクスを組み合わせたものをいう。
例）欠陥検出密度（欠陥数／規模）

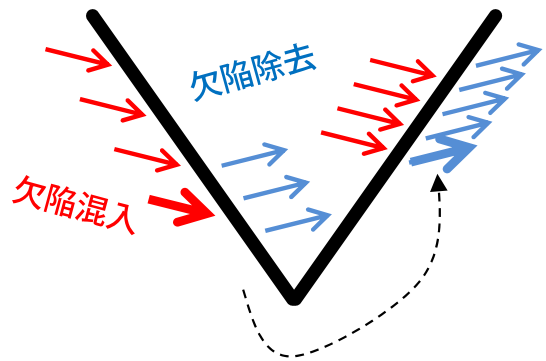
- 測定の信頼性
- 測定の妥当性

- 「品質の把握・評価」には、一見、単純と思えるような数値化や、可視化の工夫で十分なことが多い
- 「影響要因の特定」や「予測モデルの構築」には、統計解析手法が適していることが多い
- インプットとなるメトリクスの信頼性と妥当性に大きな問題があると、どのような分析手法を用いてもダメな結果しか得られない

上流工程までの累積欠陥除去率の目標値に関して(1)

例：「上流工程までの累積欠陥除去率 75% を目標としましょう」

開発プロセス（V字モデル）



上流工程までの欠陥除去数：　3件
テスト工程での欠陥除去数：　5件
欠陥除去数の総数：　8件

完了したプロジェクトについて、
目標の達成状況を把握したい

「欠陥除去数の総数」を分母としたときの、
上流工程での累積欠陥除去率は 3/8 = 37.5% である

- 正しい「上流工程までの累積欠陥除去率」は、明らかに 3/4 = 75%である
- 上流工程までの累積欠陥除去率を事後に正しく評価するには、
  下流工程で除去した欠陥の混入工程を把握しておく必要がある

欠陥データマトリクスと欠陥除去率

作込み / 除去	要求定義	機能設計	詳細設計	ｺｰﾃﾞｨﾝｸﾞ	単体テスト	結合テスト	ｼｽﾃﾑテスト	運用	合計	DRE
機能設計 ｲﾝｽﾍﾟｸｼｮﾝ	49	681							730	74.4%
詳細設計 ｲﾝｽﾍﾟｸｼｮﾝ	6	42	681						729	61.3%
コード ｲﾝｽﾍﾟｸｼｮﾝ	12	28	114	941					1095	54.8%
単体テスト	21	43	43	223	2				332	36.7%
結合テスト	20	41	61	261	―	4			387	67.1%
ｼｽﾃﾑテスト	6	8	24	72	―	―	1		111	58.1%
運用	8	16	16	40	―	―	―	1	81	
合計	122	859	939	1537	2	4	1	1	3465	97.7%

例：詳細設計ｲﾝｽﾍﾟｸｼｮﾝの DRE
DRE = 729 / (122+859+939 – 730)

プロセス全体の DRE

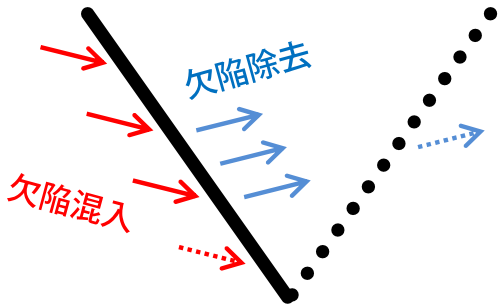
欠陥除去能力の低い工程を特定し、重点改善対象の候補とする

Laird, L. M. and Brennan, M. C., *Software Measurement and Estimation: A Practical Approach*, John-Wiley and Sons, 2006 (野中誠, 鷲崎弘宜訳：演習で学ぶソフトウェアメトリクスの基礎, 日経BP社, 2009).
Kan, S. H., *Metrics and Models in Software Quality Engineering*, 2nd ed., Addison-Wesley, 2002 (古山恒夫, 冨野壽監訳, ソフトウェア品質工学の尺度とモデル, 共立出版, 2004).

上流工程までの累積欠陥除去率の目標値に関して(2)

例：「上流工程までの累積欠陥除去率 75% を目標としましょう」

開発プロセス（V字モデル）



上流工程での欠陥混入数： 3件 （認識できている範囲で）
上流工程での欠陥除去数： 3件

いま、上流から下流工程への移行判定をしようとしている

上流工程での累積欠陥除去率は100% … 目標達成できた？

- 上流工程が終わった時点では、上流工程で混入した欠陥のうち、上流工程で除去できたものしか把握できない
- 「目標値 75%」を達成できたかどうかは、テストを終えないと （正確には、出荷後の一定期間が経過しないと） 分からない
- 上流工程が終わった時点で目標値を達成できたと判定するためには、欠陥混入数の予測値を得ておかなければならない

欠陥数の予測例：重回帰分析の適用

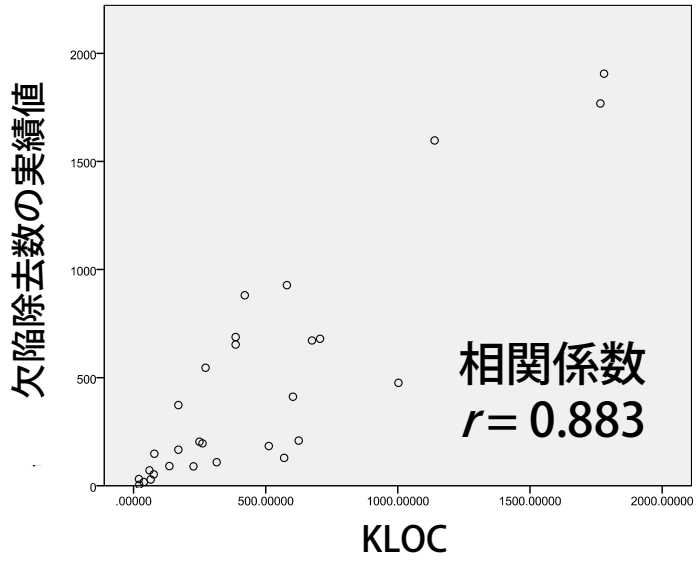
分析対象のデータ（$N=29$）

区分	記号	メトリクス
定量データ	Defects	欠陥検出数
	KLOC	ソースコード行数
定性データ（5段階評価）	D1	開発メンバーの経験
	P5	ステークホルダーの関与度
	P6	顧客とのコミュニケーション

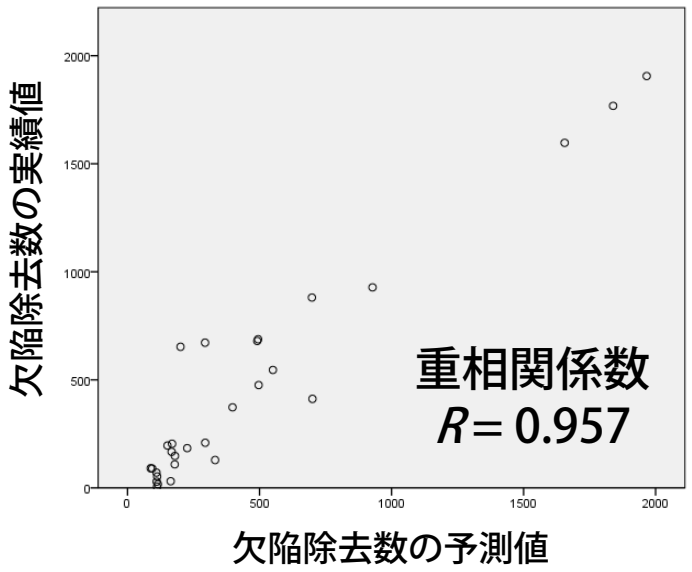
適切な統計手法を用いることで、
欠陥数を高い精度で予測できる
モデルを構築できる

ただし、「精密に誤るよりも…」には留意

規模と欠陥数の関係



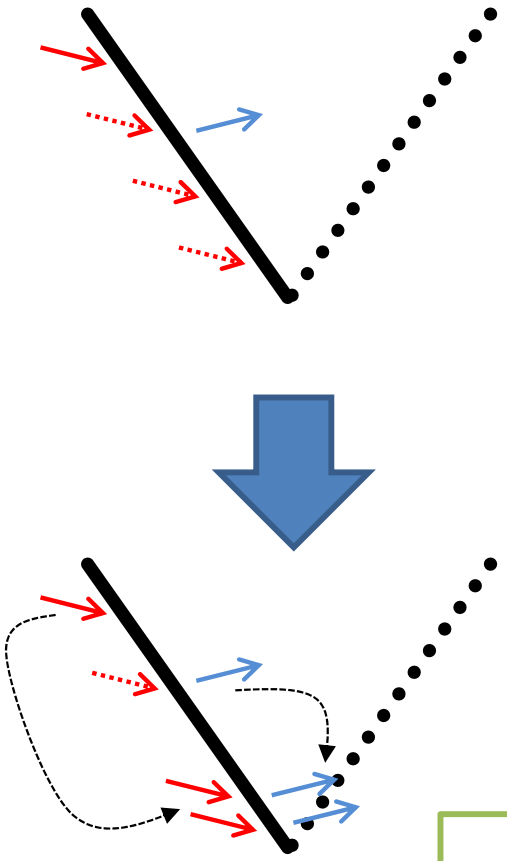
重回帰分析：
Defects = f(KLOC, D1, P5, P6)



（データの出典）Fenton, N., Neil, M., Marsh, W., Hearty, P. and Radlinski, L.: On the effectiveness of early life cycle defect prediction with Bayes Nets, *Empirical Software Engineering*, pp. 499–537, Springer, June 2008. http://www.springerlink.com/content/kg06211161305k1t/ よりダウンロード可能
野中（2011）「計数データとしてのソフトウェア欠陥の予測」『ウィンターワークショップ2011・イン・修善寺 論文集』pp.111-112, 情報処理学会

上流工程までの累積欠陥除去率の目標値に関して(3)

例：「上流工程までの累積欠陥除去率 75% を目標としましょう」



上流工程での欠陥除去数：　1件
予測された欠陥混入数：　4件
累積欠陥除去率の予測値：　1/4＝25%

いま、上流から下流工程への移行判定をしようとしている

欠陥の粒度は、開発・技術部門に委ねているとする

「少なくとも、欠陥をあと2件見つけないと次に進めません」と、開発・技術部門に指示した

出荷を急ぐ開発・技術部門が取り得る対応
- 除去済み欠陥を3件にスプリットする （！）
- 見かけ上の累積欠陥除去率は、3/4＝75% （！！）

欠陥の粒度のガイドラインを定めておく必要がある

IEEE標準は、欠陥の数え方に関するガイドラインとなるか？

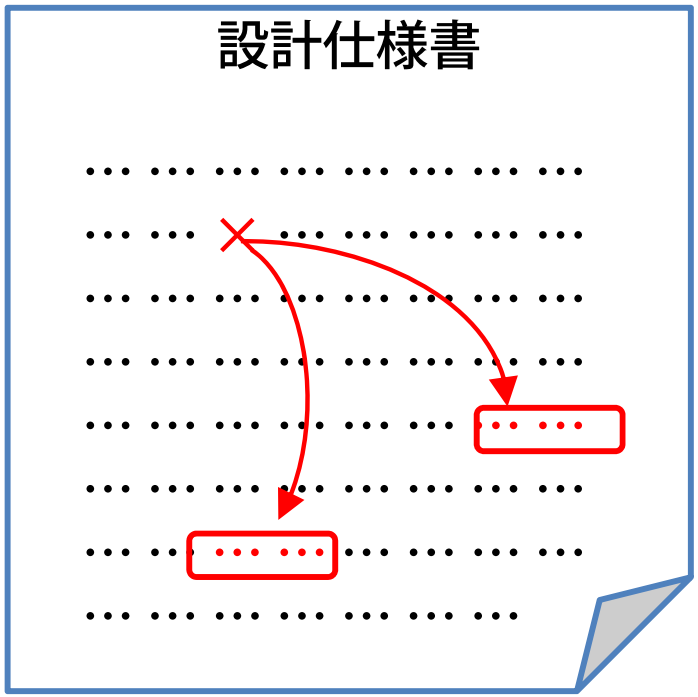
用語	定義（筆者訳）	出典のIEEE標準
欠陥 (defect)	障害(fault)または故障(failure)のいずれかを言及できる一般的用語。	982.1-2005
障害 (fault)	(1) ハードウェアデバイスまたはコンポーネントの欠陥(defect)。 (2) コンピュータープログラム内の，不正確なステップ，プロセス，またはデータ定義。一般には，エラー(error)やバグ(bug)をこの意味に用いる。	610.12-1990(R2002)
故障 (failure)	システムまたはコンポーネントが，指定された性能要求の範囲内で要求された機能を果たせないこと。	610.12-1990(R2002)
エラー， 誤り (error)	(1) 正しい結果と，観測された結果の相違。狭義のerror。 (2) 不正確なステップ，プロセス，またはデータ定義。狭義のfault。 (3) 不正確な結果。狭義のfailure。 (4) 不正確な結果を生み出した人間の行為。狭義のmistake。	610.12-1990(R2002)
不正 (anomaly)	要求仕様，設計文書，ユーザ文書，標準，または誰かの認識もしくは体験など，これらに基づく期待から乖離した状態。	1044-1993(R2002) 1028-2008
	文書において，またはソフトウェアもしくはシステムの運用において観測された，期待から乖離したものすべて。ただし，期待は，検証済みのソフトウェア製品，リファレンス文書，または指定された振る舞いを記述したその他の情報源に基づく。	829-2008

欠陥1件の粒度について、IEEE標準はとくに言及していない

Q1：何件の欠陥と数えるか？

設計レビューで、設計仕様書の誤記を発見した。

当該個所1個所を修正したのち、これに伴って変更を要する2個所（設計仕様書内）を修正した。



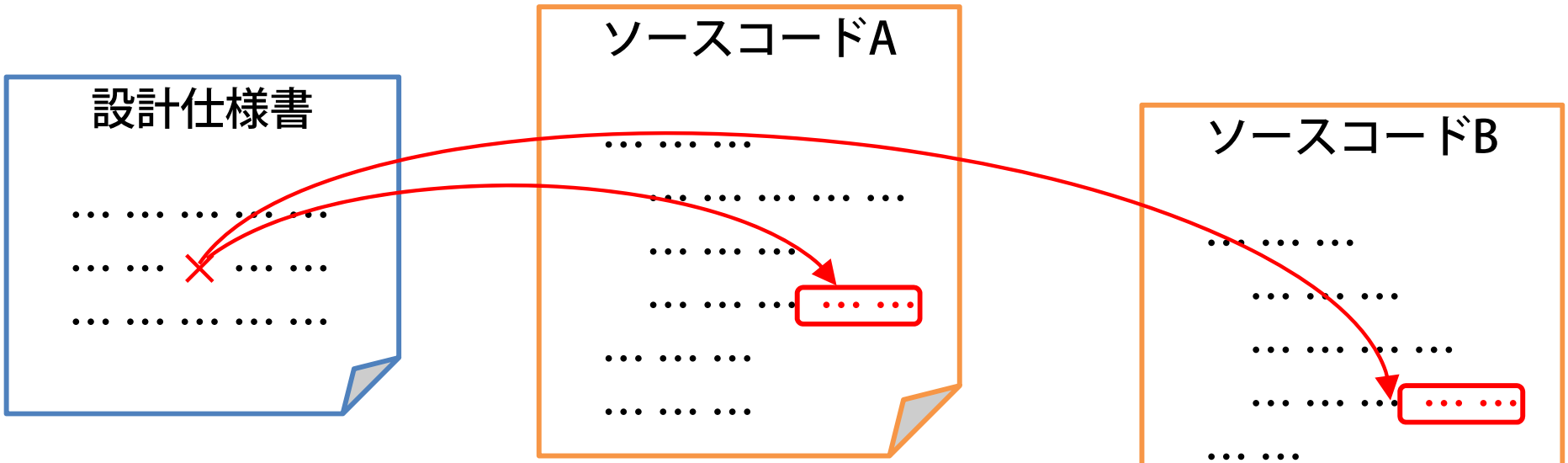
この場合、何件の欠陥としてカウントしますか？

1？ 2？ 3？ それ以上??

Q2：何件の欠陥と数えるか？

結合テストで、プログラムの振る舞いの問題を発見し、
原因が設計仕様にあることが分かった。

設計仕様書の当該個所1個所を修正したのち、
これに伴って変更を要するソースコード2個所を修正した。

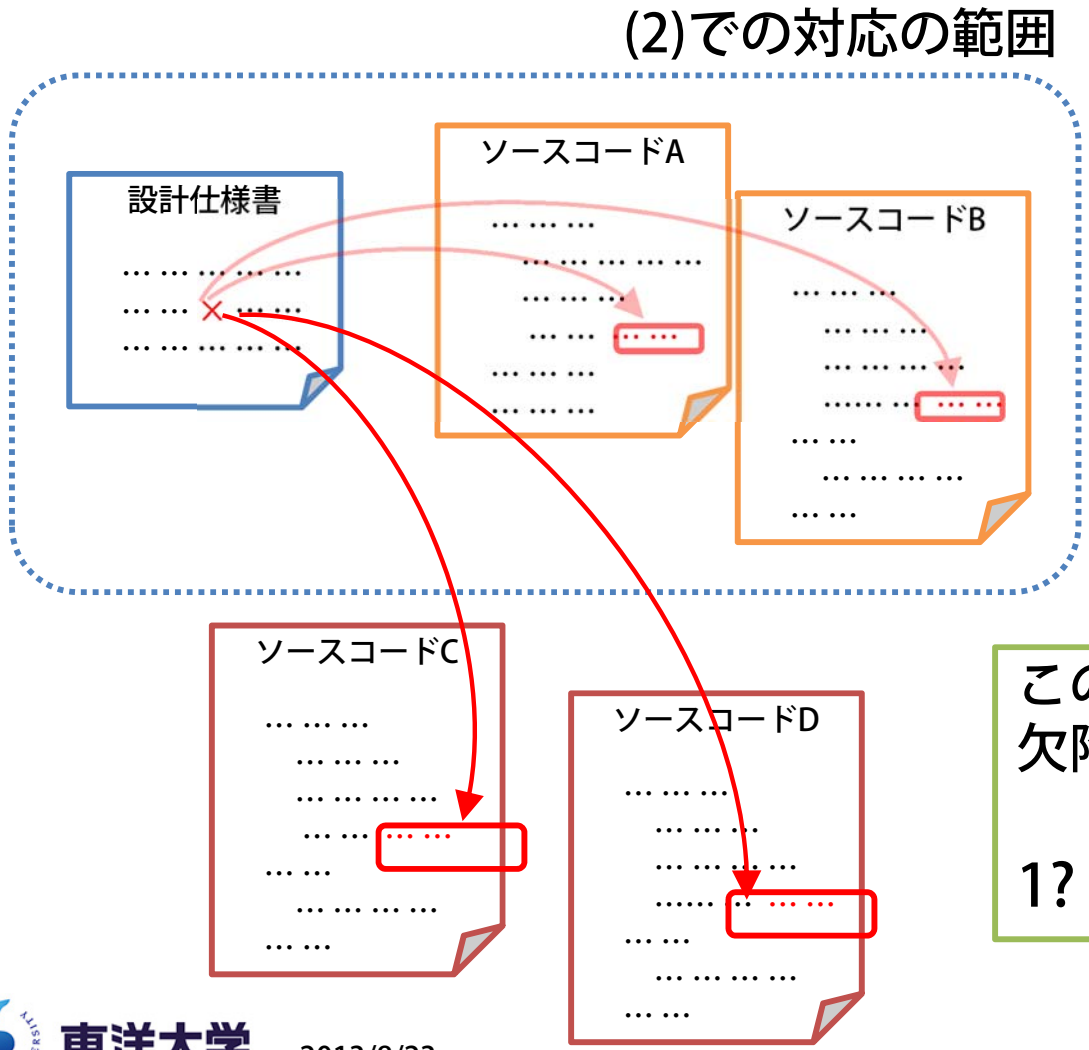


この場合、何件の欠陥としてカウントしますか？

1? 2? 3? それ以上??

Q3：何件の欠陥と数えるか？

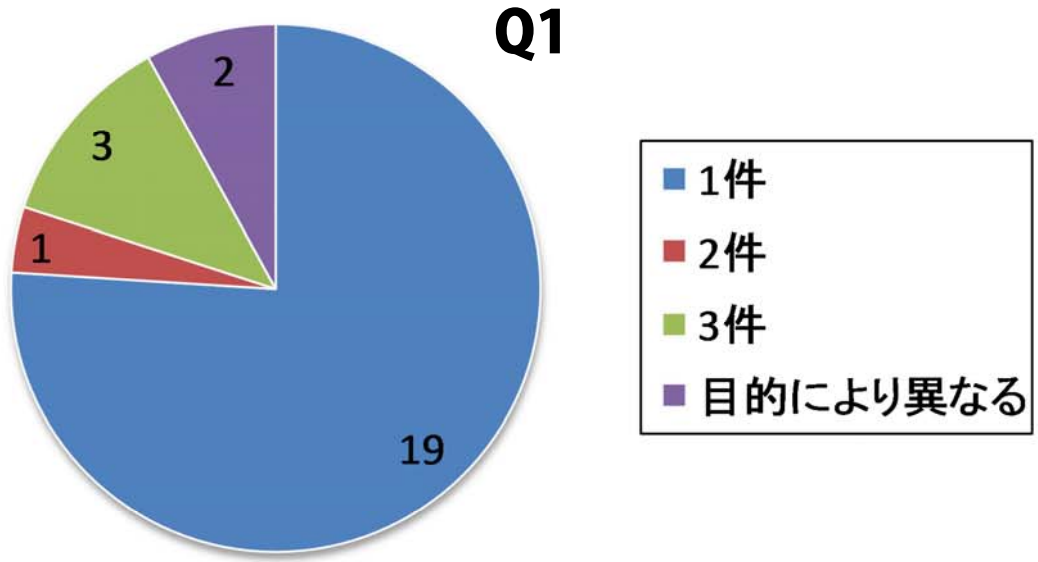
（続き）その後のシステムテストで、
(2)と同じ対応をし忘れたソースコード2個所を発見し、
これを修正した。

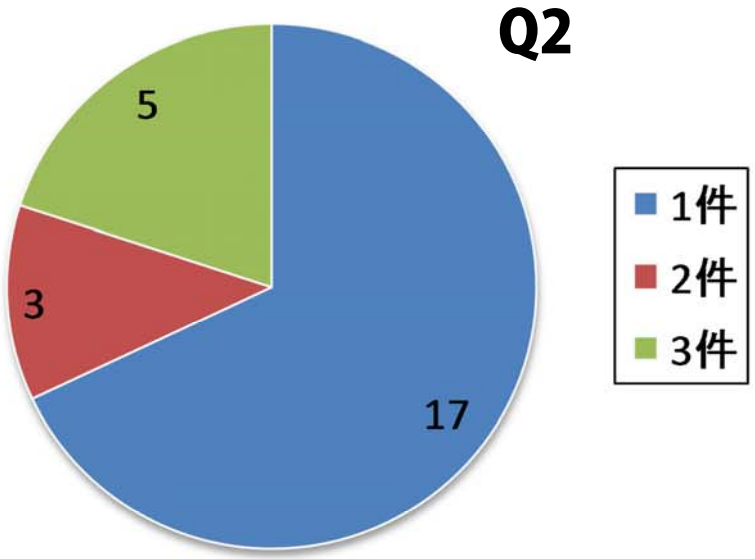


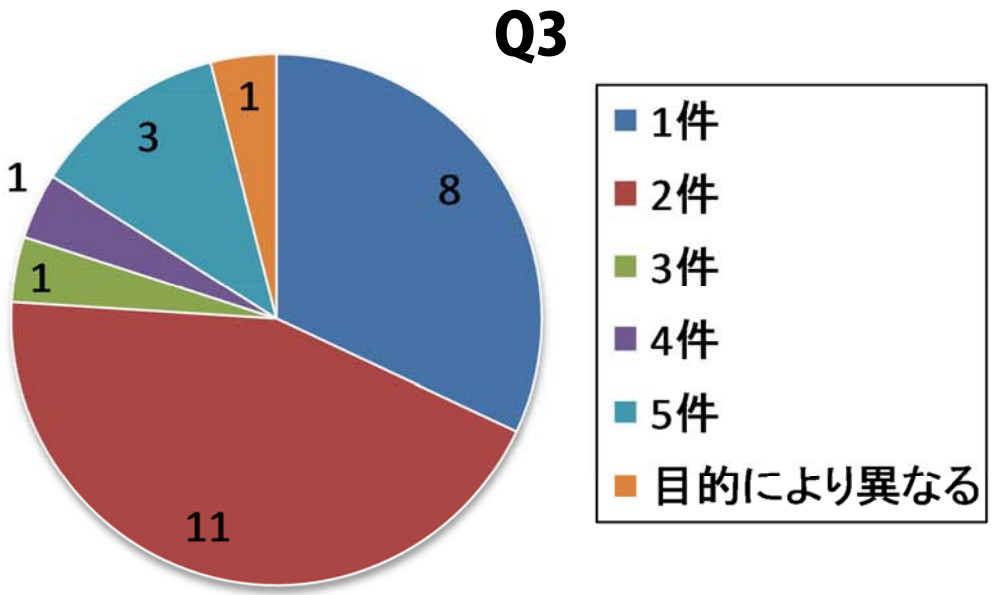
この場合、全体として何件の
欠陥としてカウントしますか？

1? 2? 3? 4? 5? それ以上??

参考：SQiPシンポジウム2010メトリクスSIGでのアンケート結果







- 数え方は組織によって様々なので、このグラフのように結果がばらつくのは仕方ない
- しかし、あなたの組織内でこれだけばらついていたら、欠陥のデータは意味をなすだろうか？

野中（2010）「ソフトウェア品質の定量的管理における曖昧さ－ソフトウェア欠陥測定の原則－」『経営論集』pp.99-109，東洋大学経営学部

ソフトウェア欠陥1件の数え方の例

※ 開発プロセス全体で欠陥の粒度に一貫性を持たせ、
欠陥除去工程のパフォーマンスを正しく評価したい場合

原則 ： プロダクトの原因部分で集約して１件と数える

- 原因部分が1個所のみの場合 → 1件
  - 要求仕様書の記述に誤りが1個所あり、要求レビューでこれを除去した
- 原因部分の修正が、ほかの個所に影響する場合 → 1件
  - 仕様の問題個所を修正した後に、関連して修正しなければならない仕様書の記述2個所を修正した … Ｑ１、Ｑ２のパターン
- 原因が成果物ではなくプロセスにある場合 → 修正個所を別々にカウント
  - Ｑ３の例は、デバッグプロセスにおける修正漏れという問題のために、2個の欠陥が新たに入り込んだと解釈する
  - Ｑ２で識別した１件と、Ｑ３で新たに識別した２件を合計する → ３件

野中（2010）「ソフトウェア品質の定量的管理における曖昧さ－ソフトウェア欠陥測定の原則－」『経営論集』pp.99-109，東洋大学経営学部

欠陥の分類：欠陥の数を扱う前に考えなければならないこと

機能性欠陥
(functional defects)

大規模欠陥	機能性の欠如、大幅な追加・修正が必要
サポート	ライブラリの欠陥
タイミング	マルチスレッド環境で生じる問題
チェック	妥当性確認ミス、不正な値の検出ミス
リソース	データ、変数、リソース初期化、解放などの誤り
ロジック	比較演算子、制御フロー、計算などの誤り
インタフェース	ライブラリ、デバイス、DB、OS等とのインタフェース不整合

発展性欠陥
(evolvability defects)

構造	コード構造
視覚的表現	コードの読みやすさ
ドキュメンテーション	コードの説明

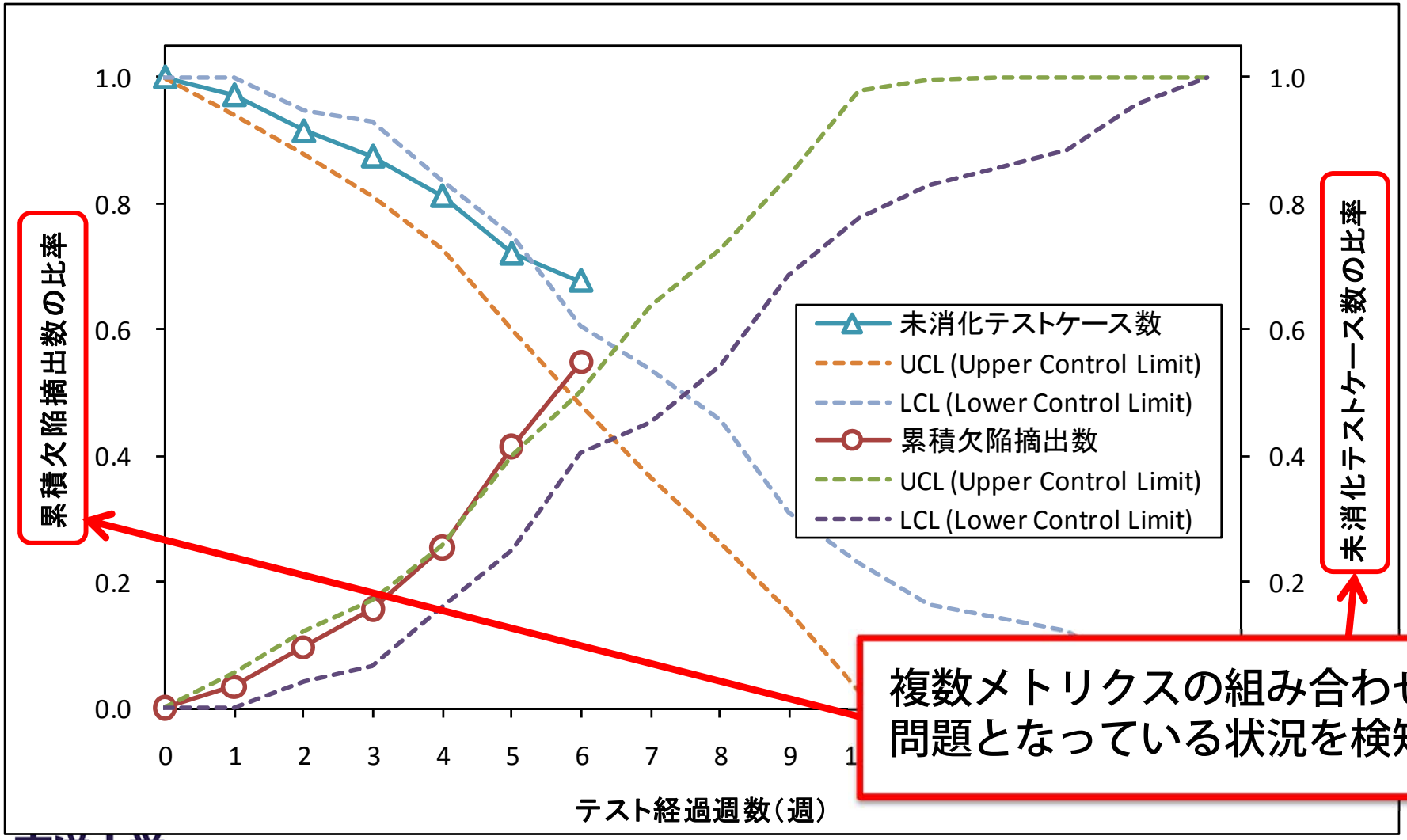
コードインスペクション指摘の約70%は、発展性欠陥を検出
(Mäntylä, 2009)

- 欠陥の定量的管理の前提として、欠陥の分類を考えておく必要がある
- 欠陥除去のフロントローディングを議論するときは、機能性欠陥に着目する
- 将来の機能性欠陥につながる発展性欠陥は、早い段階で芽を摘んでおく
- 欠陥の分類に着目して、プロジェクトの状況を把握する

Mäntylä, M. V. and Lassenius, C., "What Types of Defects Are Really Discovered in Code Reviews?," *IEEE Trans. Softw. Eng.*, vol. 35, no. 3, pp. 430-448, 2009.

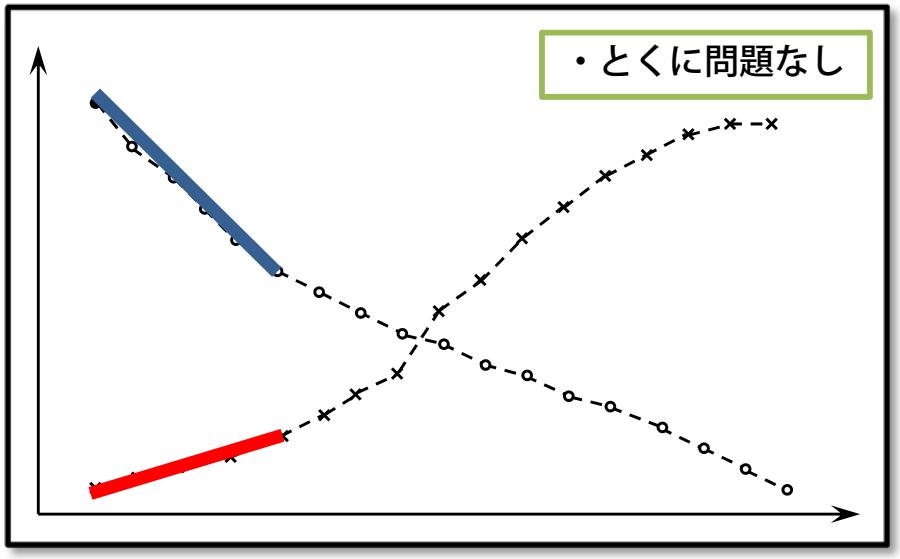
テスト工程の管理

- 次のテスト管理図から、どのような状況が読み取れるか？
- その状況を生み出した原因として何が考えられるか？
- その原因特定のために、どのようなドリルダウンをすればよいか？

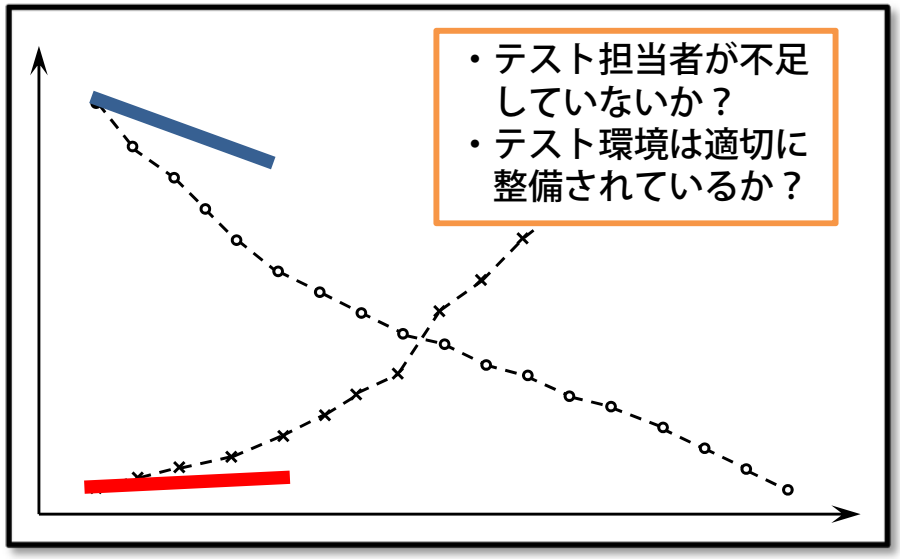


テストの進捗：テスト管理図のパターン

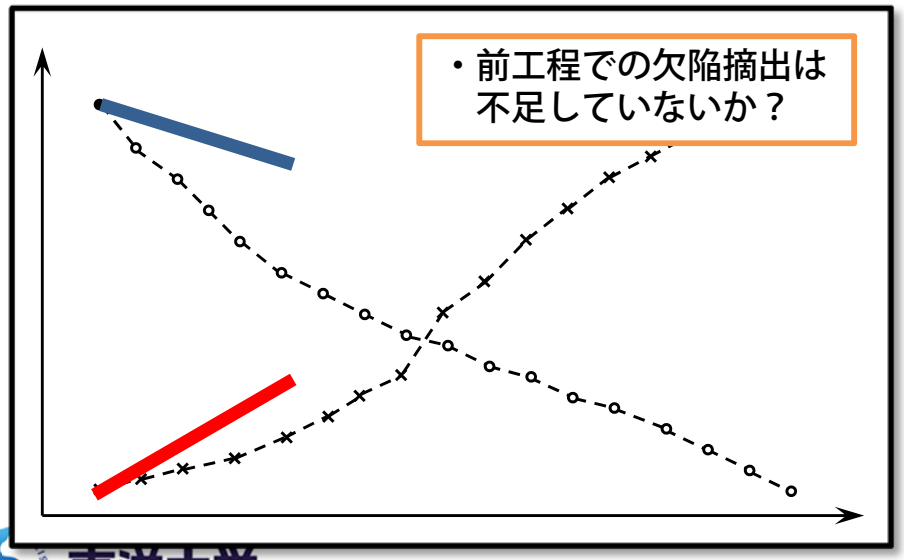
計画通りに進行



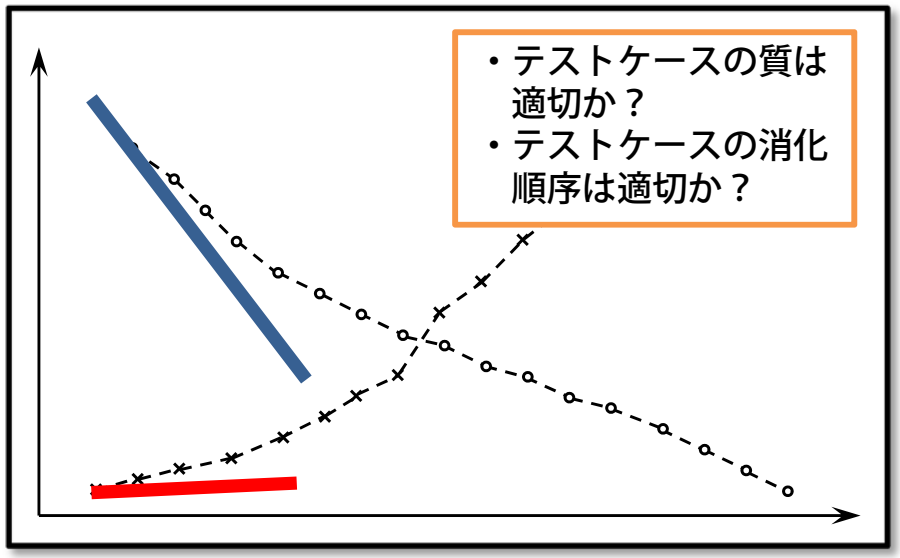
テストケース消化停滞・欠陥摘出不足



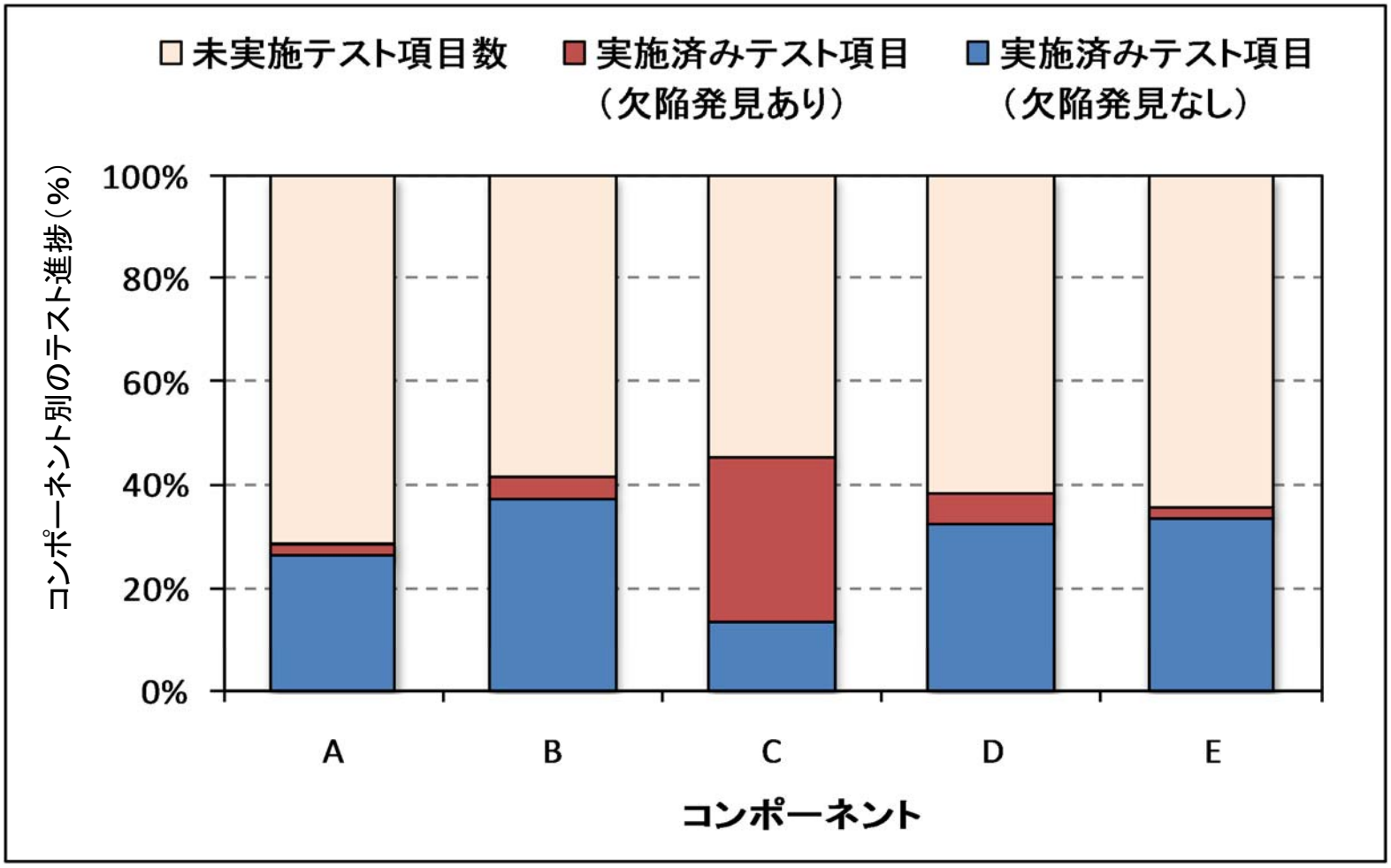
テストケース消化停滞・欠陥多発



テストケース急速消化・欠陥摘出不足



コンポーネント別のテスト進捗

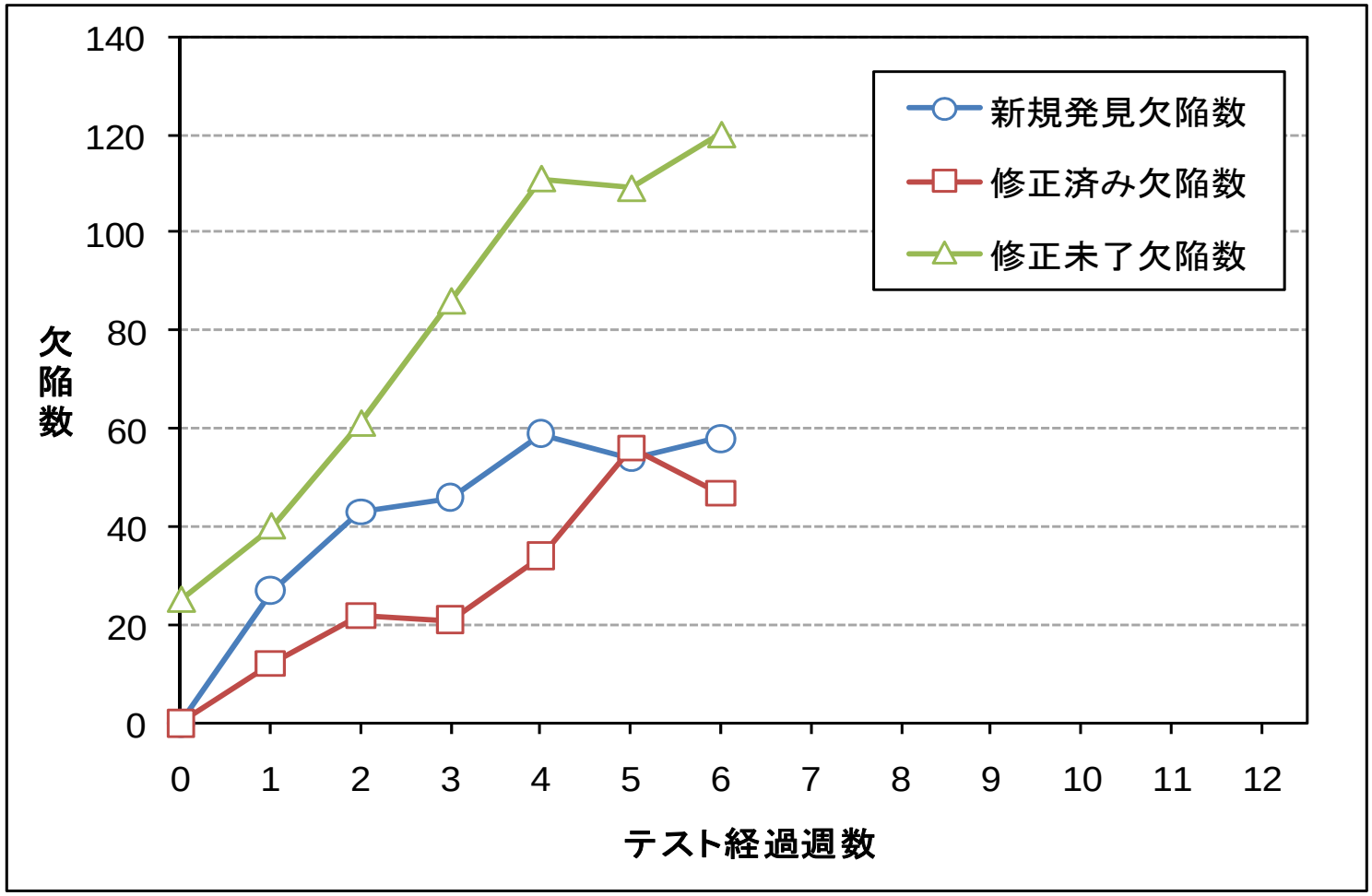


第6週での状況

- コンポーネントCから、多くの欠陥がテスト工程で見つかっている
- コンポーネントCのレビューが適切ではなかったと考えられる

参考：L. M. Laird and M. C. Brennan, *Software Measurement and Estimation: A Practical Approach*, John-Wiley and Sons, 2006.
（野中・鷲崎訳：演習で学ぶソフトウェアメトリクスの基礎、日経BP社（2009））

欠陥バックログ



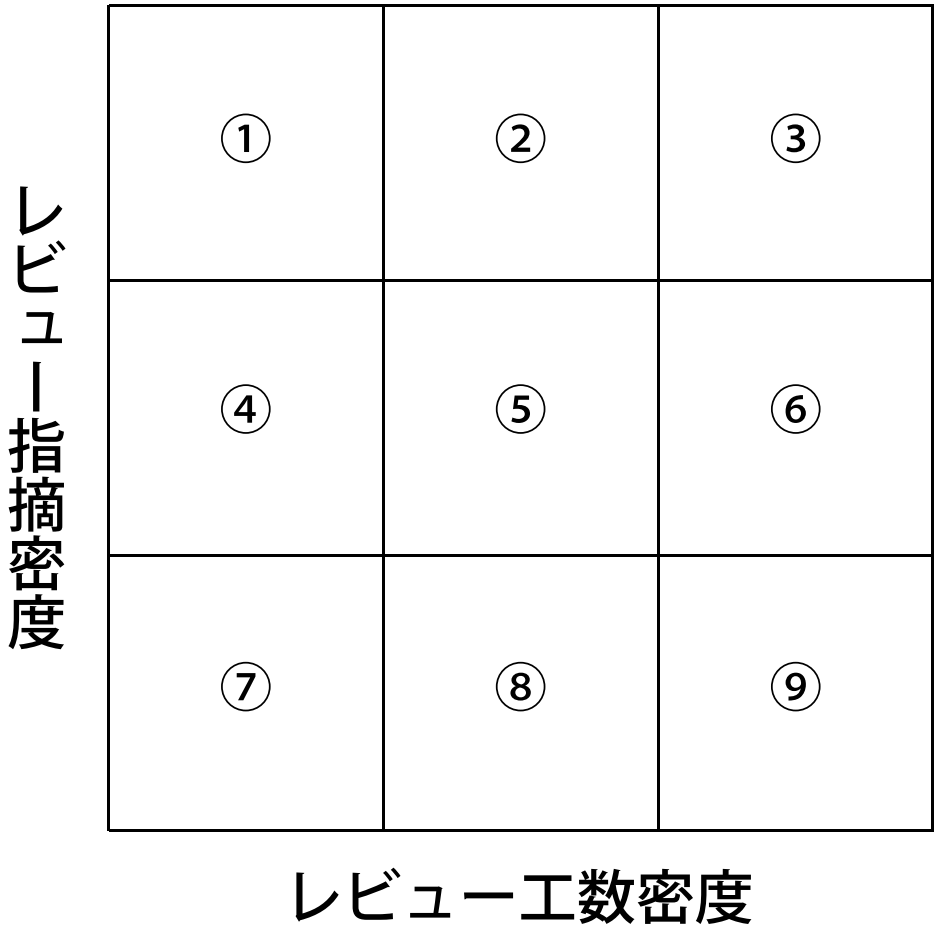
第6週での状況

テストで発見した欠陥の修正に手間取り、テスト進捗を停滞させる原因になっていると考えられる（テストとデバッグが同一チームの場合）

参考：L. M. Laird and M. C. Brennan, *Software Measurement and Estimation: A Practical Approach*, John-Wiley and Sons, 2006.
（野中・鷲崎訳：演習で学ぶソフトウェアメトリクスの基礎、日経BP社（2009））

ゾーン分析：レビューメトリクスの組合せによる判断

- レビュー指摘密度：規模当たりのレビュー指摘件数（件／規模）
- レビュー工数密度：規模当たりのレビュー工数（工数／規模）



問題がないと思われるのは？ なぜ？

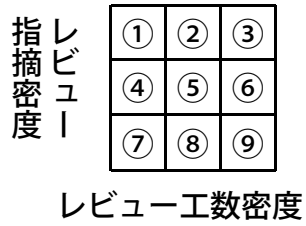
品質が悪い可能性があるのは？ なぜ？

品質が良い可能性があるのは？ なぜ？

品質を判定できないのは？　なぜ？

問題があるなら、どのような対策が必要？

ゾーン分析の判定例



『続・定量的品質予測のススメ』

評価	
⑤	一応、品質は良好
⑥	レビューの進め方、体制の点検が必要
⑧	レビューの進め方、体制、漏れの点検が必要
⑨	レビュー効率が悪いため、レビューの進め方、体制、漏れの点検が必要
②	設計不良のため、前工程設計不良、検討不足の点検が必要
③	設計不良のため、前工程設計不良、検討不足の点検が必要
①	レビュー不足のため、レビューの進め方、体制、 漏れの点検が必要
④	レビュー不足のため、品質不良、設計・レビュー点検が必要
⑦	レビュー不足のため、追加レビューで指摘増となる可能性がある

SQiP2012 『SQiP品質保証部長の会 からの情報発信』

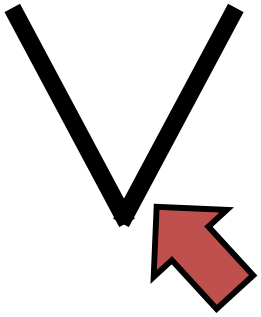
評価	
⑤	問題は発生していない
⑥	
⑧	品質が良い可能性がある
⑨	
②	品質が悪い可能性がある → 成果物を見直して再レビュー
③	
①	
④	
⑦	品質を判断できる状況にない → 追加レビューが必要か否かを判断

プロダクトメトリクスにも目を向けよう

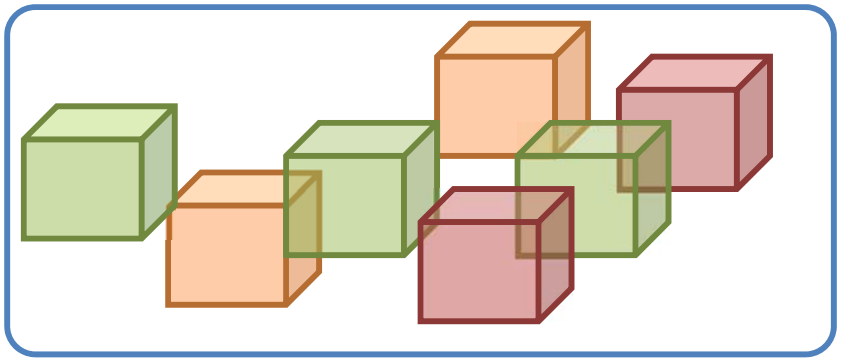
- **スタッフ部門が着目するのは、多くの場合、プロセスメトリクス**
  - 例） レビュー工数、レビュー指摘欠陥数、テスト密度、…
  - プロセスに着目すること自体は悪くなく、むしろ推奨すべき
    → プロセスを制御してプロダクト品質を制御するのは品質管理の基本
  - しかし、プロセスだけを見て、プロダクトを見ないのはいけない
    → レビュー工数が足りない、レビュー指摘数が足りない、…が足りないと言われても、「どの部分に着目して対処すべきか」の解にならない
    → プロセスメトリクスは、分解能に限界がある

- **ソフトウェア開発技術の道理を踏まえた改善を推進しよう**
  - 凝集度、結合度、複雑度などは、70年代、80年代から言われ続けているソフトウェアエンジニアリングの基本
  - これらの特性を測定するメトリクスは90年代までに数多く示されていて、ツール化されているものも多い
  - 保守性を疎かにしていると、後でたいへんな「ツケ」となって回ってくる

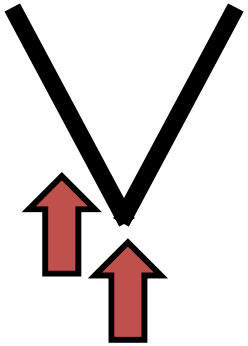
fault-proneモジュール予測のイメージ



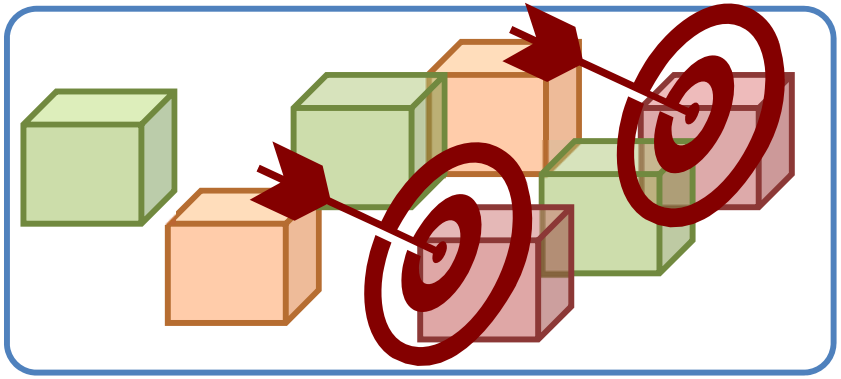
ソフトウェア開発プロセス
（V字モデル）のコーディング／
単体テストが終わった時点で、



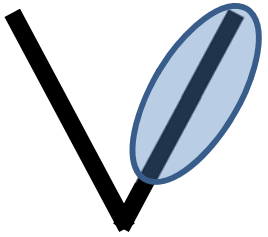
欠陥がありそうな度合いによって
モジュールを色分けして、



ソースコード等から測定可能な
プロダクトメトリクスや
プロセスメトリクスを用いて、



優先的／重点的にレビューしたり
テストしたりすることで、
レビューやテストの効率／効果を
高めたい



機能テスト／システムテストで
欠陥が見つかりそうなモジュールを、

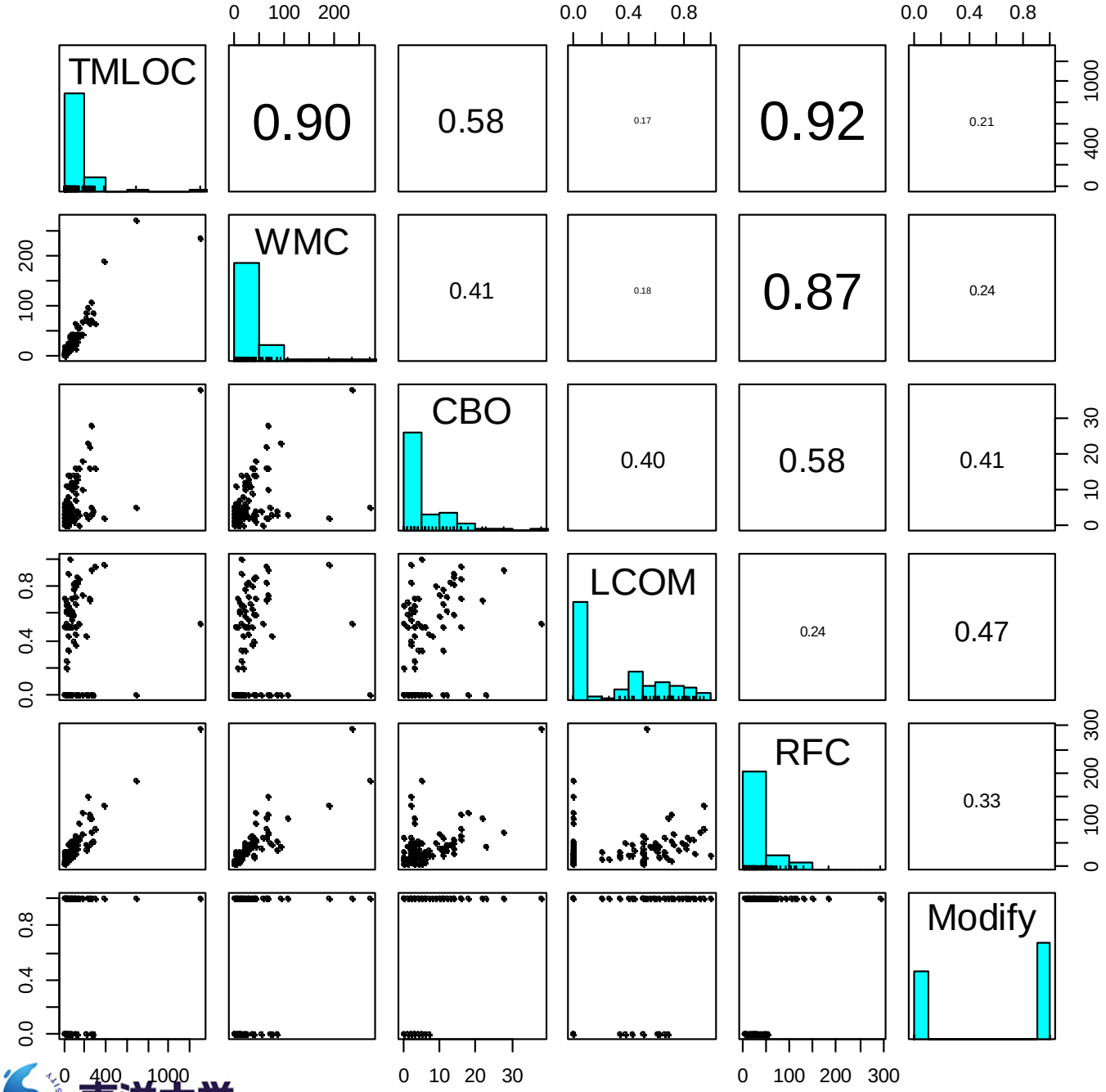
例）Twitter４Jのメトリクス分析（クラスファイル単位で測定）

メトリクス	記号	説明
総メソッド行数	TMLOC	Total Method Lines of Code 各メソッドのソースコード行数の合計
重み付きメソッド数	WMC	Weighted Methods per Class 各メソッドのサイクロマチック複雑度の合計
結合度	CBO	Coupling Between Object Classes 結合関係にある他クラスの数
凝集度の欠如	LCOM	Lack of Cohesion in Methods 属性に対してクラス内メソッドが網羅的に アクセスしていない度合い
応答処理の多さ	RFC	Response For a Class 応答に用いるメソッド呼出しの種類の数
修正の有無	Modify	リリース後に欠陥修正が生じたか否か （0: なし　1: あり）

注1）これらの無料ツールを用いて測定
- Eclipse Metrics plug-in 1.3.6, http://sourceforge.net/projects/metrics/
- Chidamber & Kemerer Java Metrics, http://www.spinellis.gr/sw/ckjm/

注2）LCOMにはさまざまなバリエーションがあるが、ここではHenderson-Sellersの定義を用いた
Henderson-Sellers, B., *Object-Oriented Metrics,* Prentice-Hall, 1996.

測定値が得られたすべてのクラスファイルの散布図行列



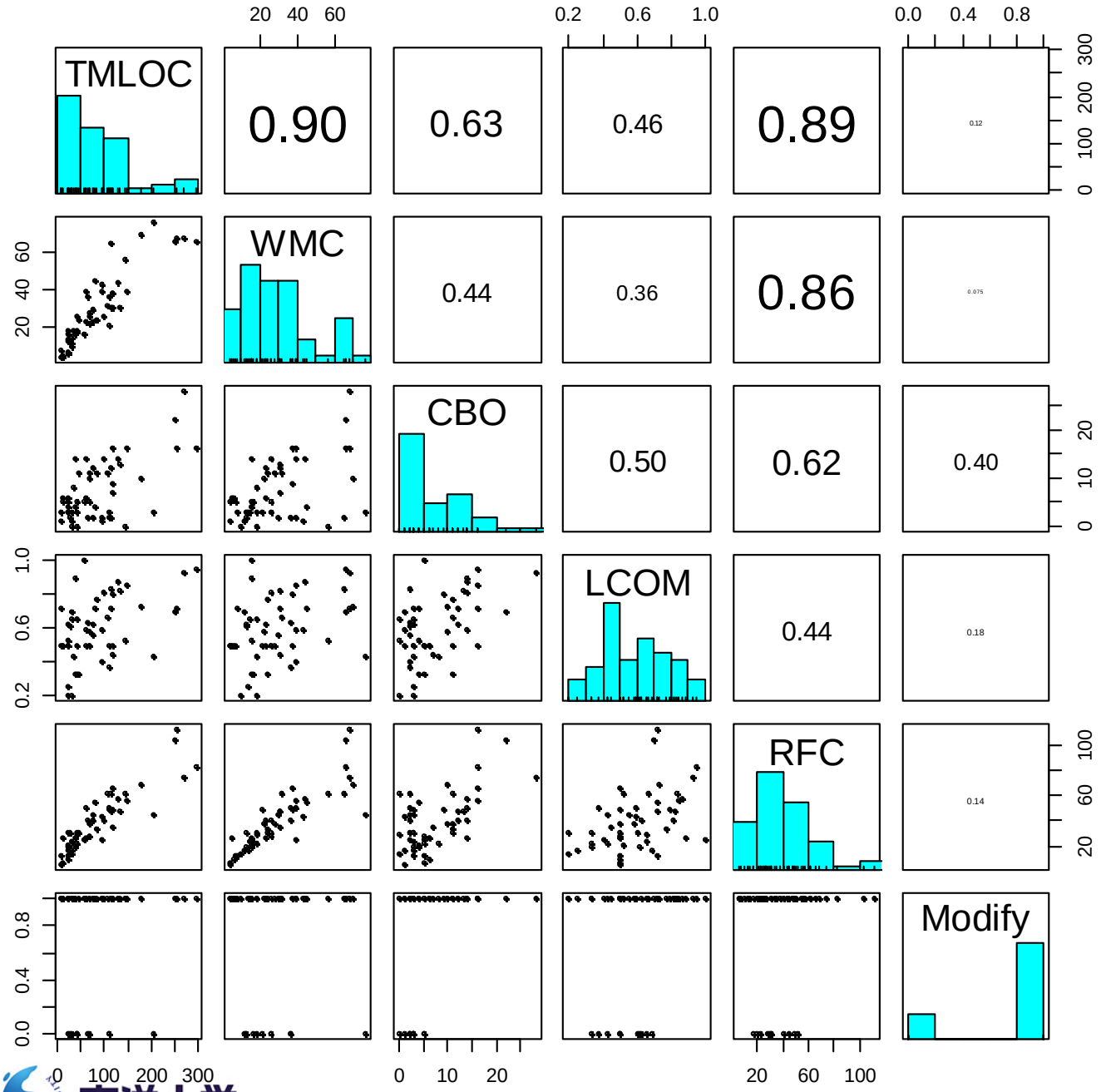
n = 102

詳細な分析の前に、
まず何を考えますか？

R の psych ライブラリの
pairs.panels関数で作図

```
pairs.panels(
    data,
    smooth=FALSE,
    density=FALSE,
    ellipses=FALSE,
    scale=TRUE
)
```

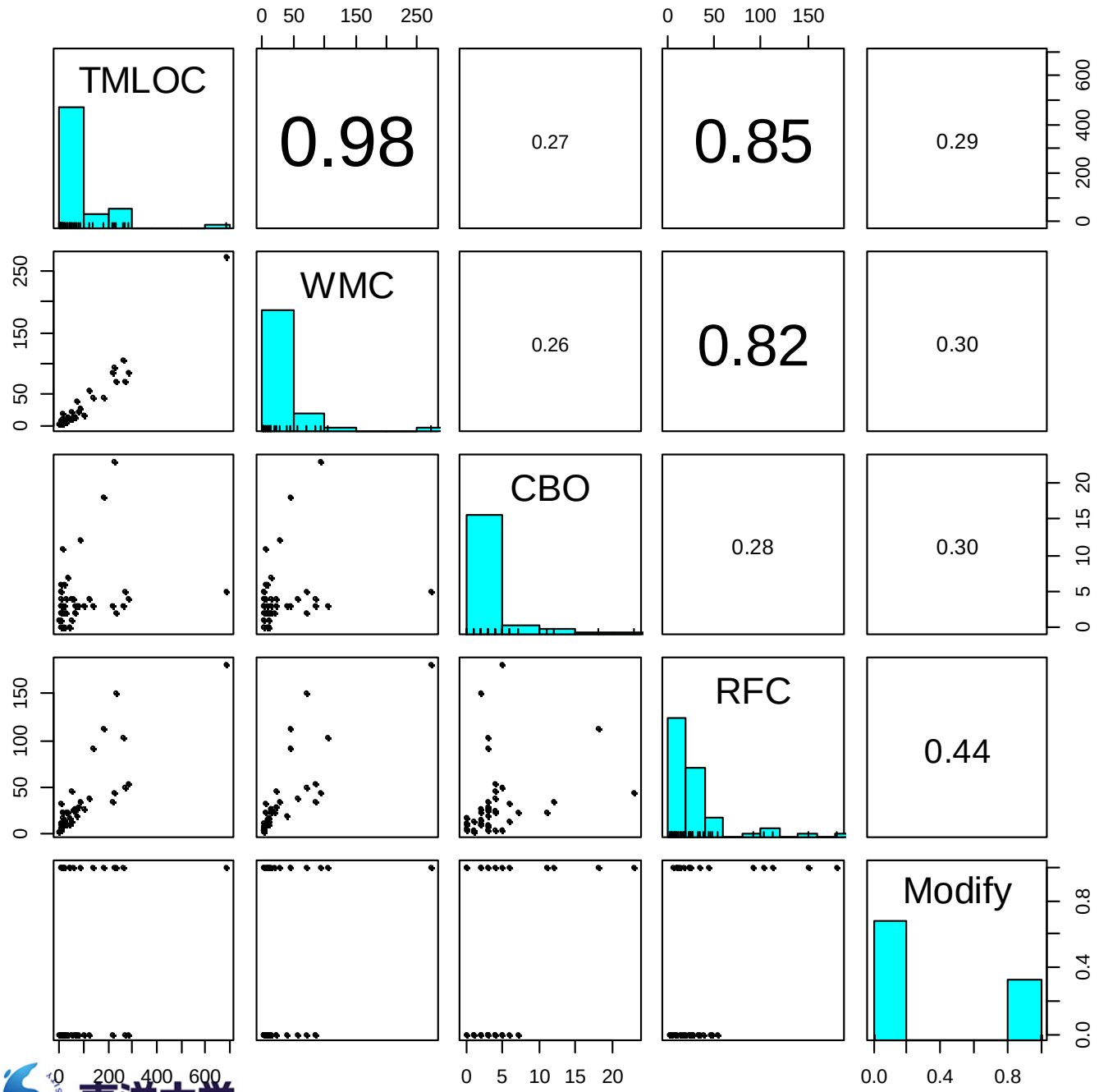
外れ値を除外し、「ある条件」で絞りこんだデータ



n = 53

この散布図行列から
何が読み取れますか？

外れ値を除外し、「ある条件」で絞りこまれなかったデータ



n = 47

この散布図行列から
何が読み取れますか？

分析によって得られた知見（分析対象のソフトウェアについて）

- TMLOCが380行以上という大きいクラスは3個あり，これらはすべてリリース後に欠陥修正があった（欠陥修正率3/3＝1.0）。
- LCOM＝0のクラスは46個あり，このうちリリース後に欠陥修正があったのは15個であった（欠陥修正率15/46＝0.33）。
- 一方，LCOM＞0のクラスは53個あり，このうちリリース後に欠陥修正があったのは42個であった（欠陥修正率42/53＝0.79）。
- LCOM＞0のクラス53個について，CBOの値が中央値の5以下では欠陥修正率が17/28＝0.61であったのに対して，中央値より大きい場合では欠陥修正率が25/25＝1.0であった。
- LCOM＝0 のクラス46個について，RFCの値が中央値の18.5以下では欠陥修正率が6/23＝0.26であったのに対して，中央値より大きい場合では欠陥修正率が9/23＝0.39であった。

レビューやテストの重点化を進めるにあたって，
このようなデータから得られた客観性の高い知見は
プロセス改善の推進力になる

品質データ分析は改善の推進力

- 品質マネジメントの基本原則は、ソフトウェアにも適合する
  - 手法をそのまま導入するのではなく「基本原則」を当てはめることが大事

- ソフトウェア開発において、事実にもとづく管理は容易でない
  - データの測定・収集プロセスが、人に大きく依存している
  - 目的と整合性のあるメトリクス分析をし続けることが難しい

- 品質データ分析は、品質に効く （長期的に）
  - 品質向上（または悪化）のメカニズムを解明する
  - データが示す客観的事実の力を活用し、改善の推進力とする
  - データを手がかりに原因を特定し、アクションをとり、効果を確認する

  そして、
  - 再発防止のためにプロセスを改善し、失敗プロジェクトを撲滅する
  - 顧客価値の向上につながる活動に注力し、顧客の満足度を高める

*“I‘d rather be **vaguely right** than **precisely wrong**.”*

精密に誤るよりも、漠然と正しくありたい

– John Maynard Keynes

- 私たちのメトリクス利用は 「精密に誤って」 いないか？（自戒の念も込めて）
  - 理論モデルの洗練や予測精度の向上が目的になっていないか

- 「漠然と正しい」 情報が得られているか？
  - 意思決定に十分役立つ精度の情報が得られているか
  - 正しい意思決定に役立つ、正しい方向感の情報を生み出しているか