ソフトウェアの信頼性とプロセス

菊野 亨
大阪大学 大学院情報科学研究科
情報システム工学専攻

講演の概要

1. 信頼性向上への対策
2. 品質予測への取り組み
3. 2つのケーススタディ

1. 信頼性向上への対策

- プログラム書法 (1980年代)
- SRGM (1980年代)
- テスト法／レビュー法 (1990年代)
- 開発プロセス (1995年ー)
- テスト設計／モデル検証 (2005年ー)

(私の年表)

プログラム書法

- 1960年代　職業的なプログラム作り
- 1970年代　構造化プログラミング
  データ抽象技法
- 1980年代　オブジェクト指向プログラミング
  オブジェクト指向設計技法

信頼性の定義

- プログラムが仕様に定められた機能をいかに漏れなく，かつ正しく実現しているかを表す品質を信頼性と呼ぶ．更に，プログラムが暴走するなどの異常事態を引き起こさないことも含める．

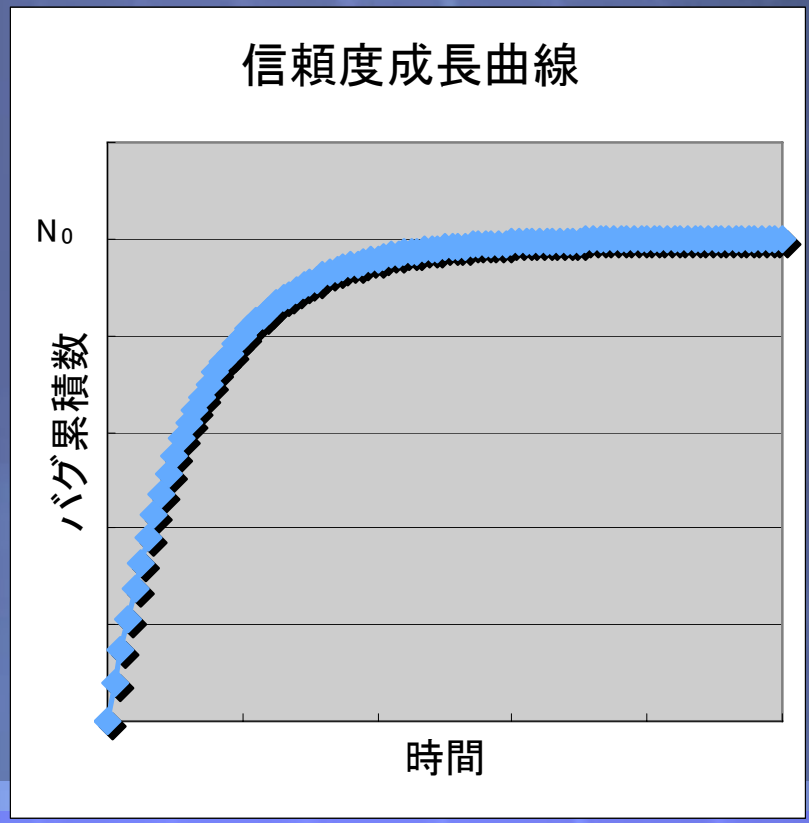
  玉井，三嶋，松田「ソフトウエアのテスト技法」共立出版

SRGM

- 指数形ソフトウエア信頼度成長モデル

$$m(t) = a(1 - e^{-bt})$$

- グラフ



テスト法／レビュー法

- ホワイトボックステスト（構造テスト）
- ブラックボックステスト（機能テスト）
- コードレビュー
- ウォークスルー
- デザインレビュー

  - バグ分析
  - レビュー工数の目安

開発プロセス

- ライフサイクルモデル
  - ウォーターフォールモデル
  - スパイラルモデル
- プロセスプログラミング
  - プロセス改善(As-is mapの記述, Should-be mapの記述)
  - プロジェクトシミュレータの開発

プロセス改善

- プロセス成熟度モデル
  - CMU/SEI　CMMI
- 5段階のレベル
  - レベル1　初期プロセス
  - レベル2　反復可能プロセス
  - レベル3　定義されたプロセス
  - レベル4　管理されたプロセス
  - レベル5　最適化プロセス

テスト設計／モデル検査

- テストカバレージにこだわる傾向
  - テストケースの生成法
  - 直行表，ペアワイズテスト
- 記号モデル検査
  - SMV（CTLモデル検査）
  - SPIN（LTLモデル検査）

2. 品質予測への取り組み

ベイズ識別器を用いた品質などの予測

Quoted from

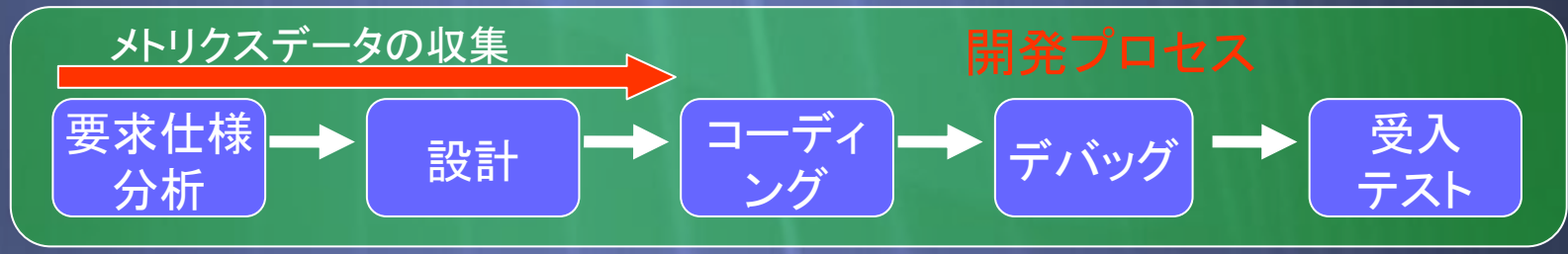
Seiya Abe, Osamu Mizuno, Tohru Kikuno, Nahomi Kikuchi, and Masayuki Hirayama, ``Estimation of Project Success Using Bayesian Classifier,'' In Proc. of 28th International Conference on Software Engineering (ICSE2006), pp.600-603, May 2006.

データの収集

- データの収集方針
  - はじめから目的を考えてデータの収集を行う.
  - SECとの綿密な打ち合わせの後，具体的なデータ収集を実施
  - 29プロジェクトからデータが集まる
- ゴール
  - 品質(Q), コスト(C), 期間(D)について，よい結果を残すか否か

ソフトウェアメトリクス

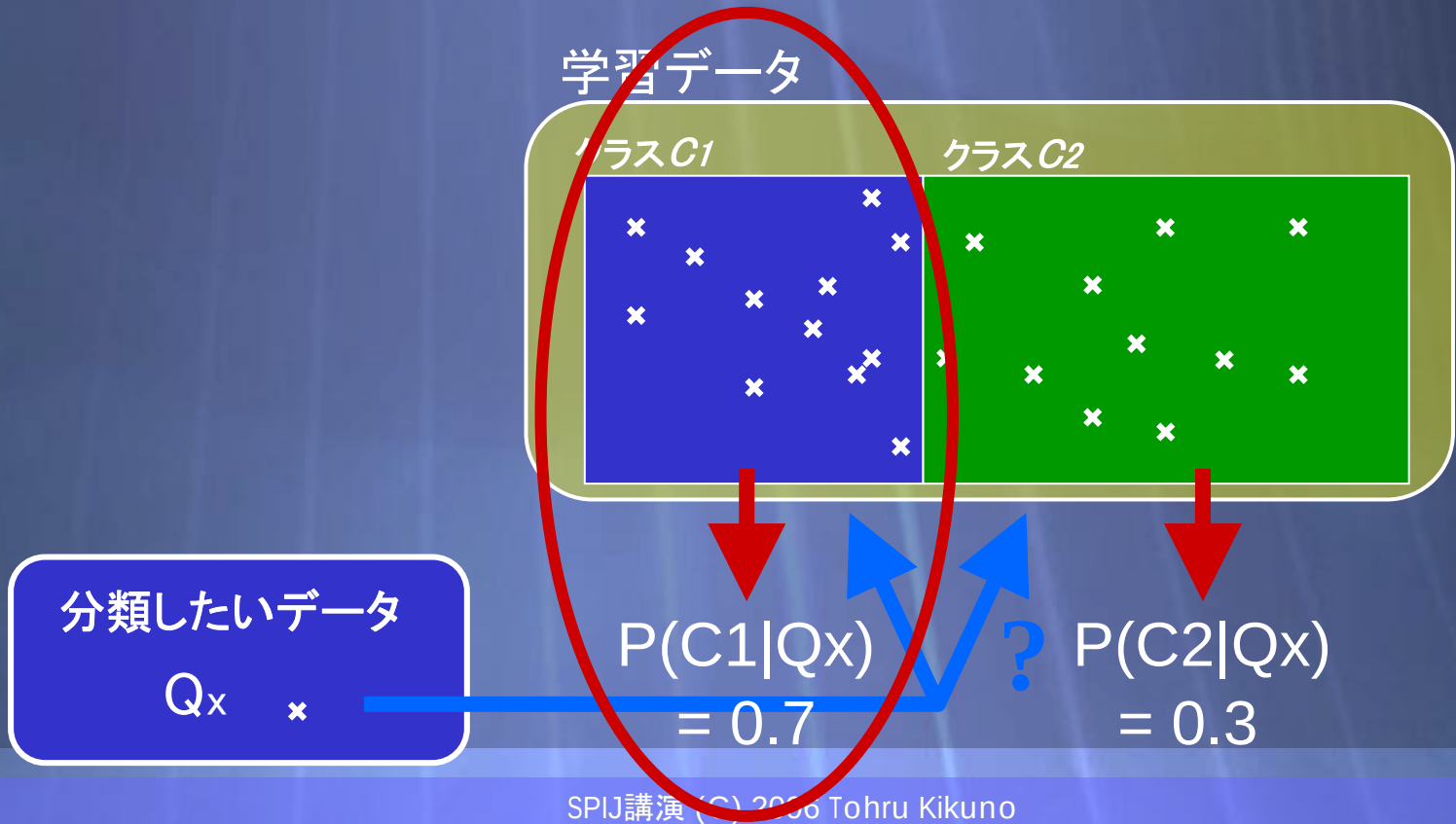
- メトリクス＝開発プロセスの状況を反映する指標（尺度）
  - 品質，コスト，工期と深い関係がある．



- 5つのカテゴリに分類される29のメトリクス（ 5段階評価）
  1. 開発プロセス: 例）システムの複雑度
  2. プロジェクト管理: 例）全体計画の正確さ
  3. 組織: 例）要求定義の顧客の参加度
  4. 人的要因: 例）開発者のスキルと経験
  5. 外乱: 例）要求の揮発性

ベイズ識別器の考え方

- カテゴリカルデータの分類を行う手法
  - 学習データからクラス分けを計算
  - 新規データが所属すべきクラスの決定



ベイズ識別器

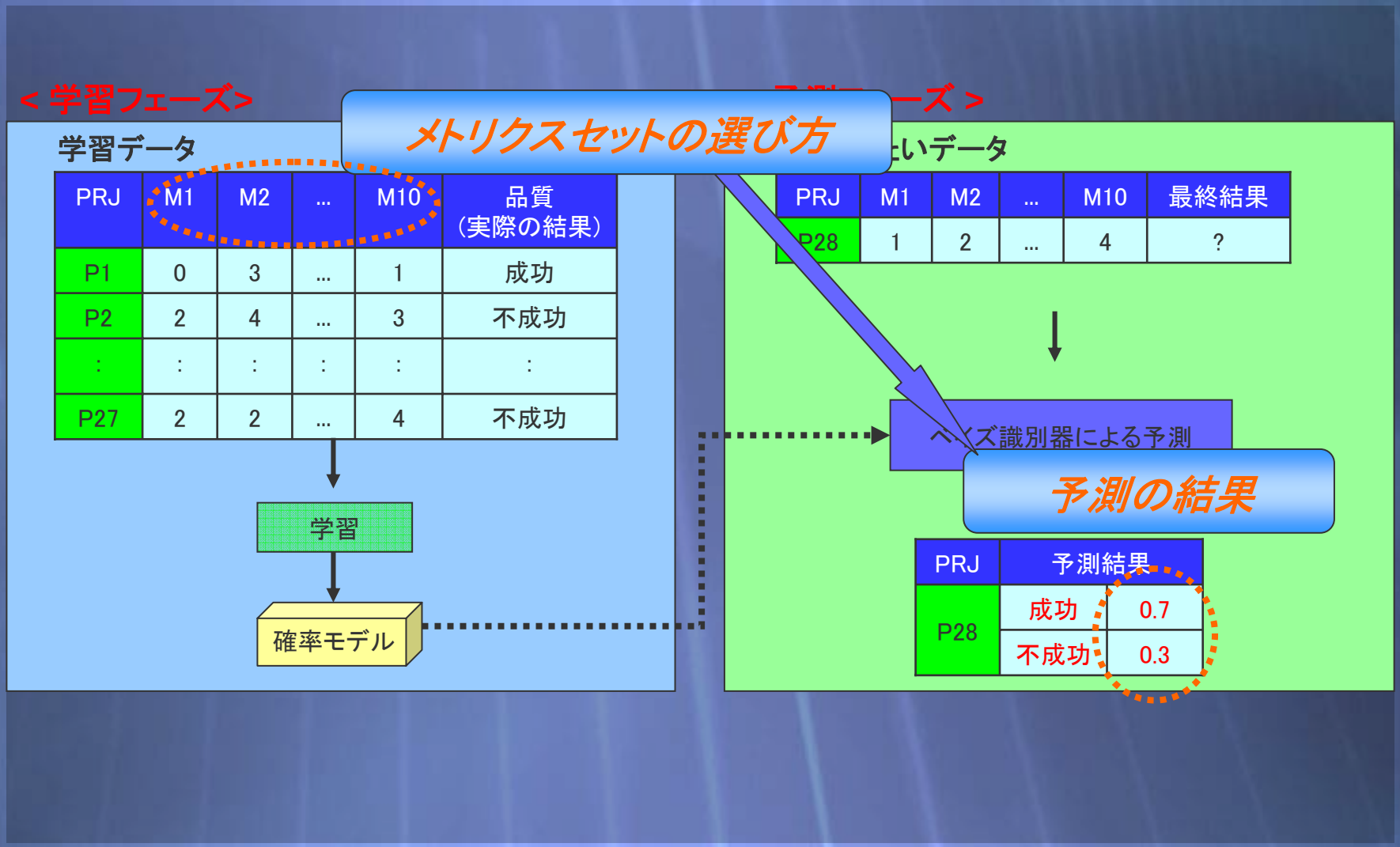
- 過去のデータからこれから起こりうる事象の確率を計算
  - クラス： $C_i \ (1 \le i \le n)$
  - 各データの持つ属性値： $D = (d_1, d_2, \cdots, d_m)$
  - データ $Q_x = (d_1 = x_1, d_2 = x_2, \cdots, d_m = x_m)$ が クラス C_i に属する 確率

$$P(C_i \mid D = Q_x) = \frac{\prod_{j=1}^{m} P(d_j = x_j \mid C_i)}{P(D = Q_x)} P(C_i)$$

学習データから計算可能

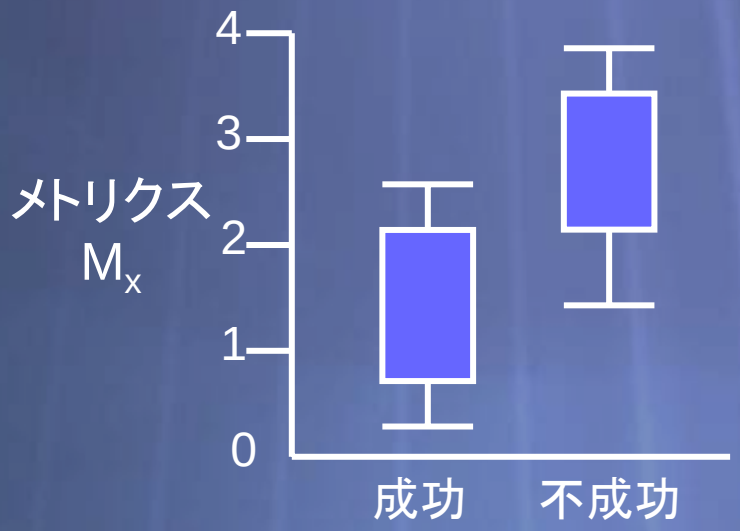
定数

ベイズ識別器による予測

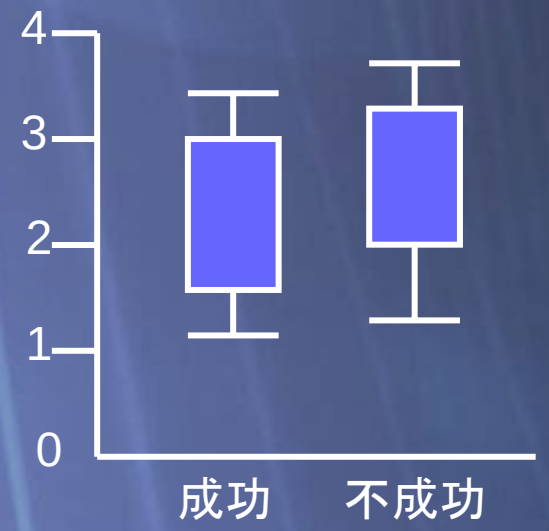


統計的検定によるメトリクスの選択

- 統計的検定によりデータの中から有効だと考えられるメトリクスを選択
  - 各メトリクス毎に，成功・不成功のクラス（グループ）間で，データの分布に差の有無を考える．



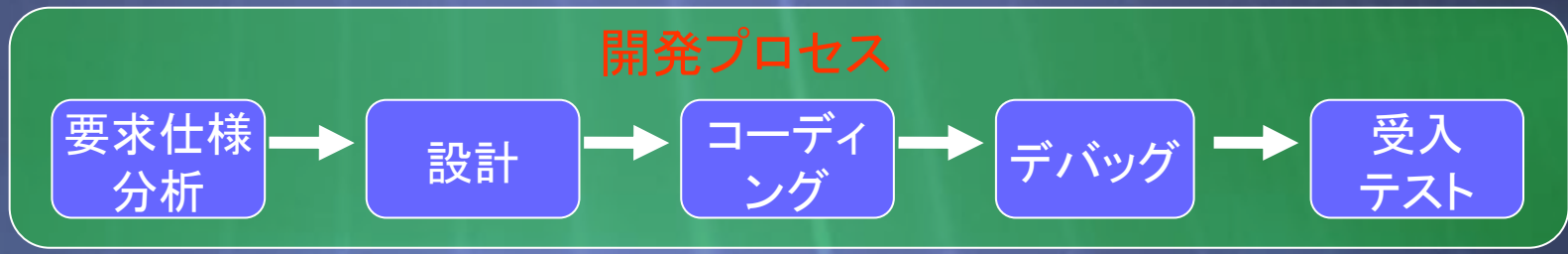
データの分布に差がある
→ **予測に有効**



データの分布に差がない
→ **予測に有効でない**

プロジェクトコンテキスト

- 企業のある組織で開発された以下の28プロジェクト
  1. 特定顧客向けのシステム統合
  2. 高信頼性のシステムが求められる.
  3. すべて同じ開発プロセスで開発された.



  4. プロジェクトのサイズはすべて中規模以上
  5. この組織がすべてのプロジェクト成果物の品質について責任をもつ.

成否の定義

- 視点毎に閾値を設け，成功・不成功を定義
  - 品質： 受入テストにおける発見不具合数
  - コスト： 計画値と実測値とのずれ
  - 工期： 計画値と実測値とのずれ

Project ID	視点		
	品質	コスト	工期
P1	不成功	成功	成功
P2	–	不成功	成功
P3	成功	成功	成功
:	:	:	:
:	:	:	:
P26	成功	成功	成功
P27	成功	成功	不成功
P28	成功	成功	–

- データ数
  - 品質： 24プロジェクト
    - 成功：20，不成功：4
  - コスト： 27プロジェクト
    - 成功：19，不成功：8
  - 工期： 26プロジェクト
    - 成功：14，不成功12

統計的検定

- Wilcoxonの順位和検定
  - 対立仮説H_1：「2つのグループの分布に差がある．」
  - 帰無仮説H_0：「2つのグループの分布に差がない．」
- 視点毎に選択されたメトリクスセット一覧
  - いずれの視点にも選択されないメトリクスが存在

	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15	M16	M17	M18	M19	M20	M21	M22	M23	M24	M25	M26	M27	M28	M29
品質	○								○									○	○	○					○	○	○	○	○
コスト						○								○							○							○	
工期							○		○	○									○		○	○			○		○		

予測結果

品質

実際の結果	予測結果	
	成功	不成功
成功	20	0
不成功	3	1

予測精度

$$\frac{20+1}{24} = 0.875$$

コスト

実際の結果	予測結果	
	成功	不成功
成功	14	5
不成功	3	5

予測精度

$$\frac{14+5}{27} = 0.704$$

工期

実際の結果	予測結果	
	成功	不成功
成功	11	3
不成功	3	9

予測精度

$$\frac{11+9}{26} = 0.769$$

GOTO 26

別の手法：専門家によるメトリクスの選択

- 各メトリクスがどの視点の予測に有効かを，専門家が**知識と経験に基づいて**選択
  - 豊富な開発経験とメトリクスの十分な知識を持つ複数の専門家
- 視点毎に選択されたメトリクスセット一覧
  - すべてのメトリクスはいずれかの視点に有効

	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15	M16	M17	M18	M19	M20	M21	M22	M23	M24	M25	M26	M27	M28	M29
品質							○	○	○	○	○	○	○						○					○	○		○	○	
コスト												○		○															○
工期	○	○	○	○	○	○	○								○	○	○	○	○	○	○	○	○		○	○			

ベイズ識別器による予測結果

品質

実際の結果	予測結果	
	成功	不成功
成功	16	4
不成功	3	1

予測精度

$$\frac{16+1}{24} = 0.708$$

コスト

実際の結果	予測結果	
	成功	不成功
成功	14	5
不成功	8	0

予測精度

$$\frac{14+0}{27} = 0.519$$

工期

実際の結果	予測結果	
	成功	不成功
成功	7	7
不成功	6	6

予測精度

$$\frac{7+6}{26} = 0.500$$

予測結果の比較(1)

精度	品質	コスト	工期
専門家	0.708	0.519	0.500
統計的検定	0.875	0.704	0.769

- 統計的検定で選ばれたメトリクスによる予測精度が大幅に向上した.
  - 選択されたメトリクスセットの違い
- 品質:
  - 不成功のクラスのデータ数が少ないため, 予測が成功に傾き, 精度が良く見える.
    - 成功:20件, 不成功:4件

予測結果の比較(2)

精度	品質	コスト	工期
専門家	0.708	0.519	0.500
統計的検定	0.875	0.704	0.769

- コスト:
  - 両方法で選択されたメトリクスの個数は同程度
    - 専門家:3個, 検定:4個
  - 検定ではコストの予測に有効なメトリクスの抽出に成功
- 工期:
  - 専門家により選択されたメトリクスの数に対して, 実際に予測に有効なメトリクスの割合は少ない.
    - 専門家:18個, 検定:8個

まとめと今後の予定

- まとめ
  - ベイズ識別器によるプロジェクトの成否の予測に有効なメトリクスセットを選択
  - 統計的検定で選択されたメトリクスセットによる予測では予測精度が大幅に向上した.
- 今後の予定
  - ほかのプロジェクトコンテキストでの適用
  - 対象プロジェクトと使用したメトリクスの関連付け
  - 不成功の回避策の考慮

3. 2つのケーススタディ

利益予測に基づくテストプロセスの改善事例

Quoted from

Toshifumi Tanaka, Keishi Sakamoto, Shinji Kusumoto, Ken-ichi Matsumoto, and Tohru Kikuno, ``Improvement of Software Process Description and Benefit Estimation,'' In Proc. 17th International Conference on Software Engineering (ICSE-17), pp.123-132, April 1995.

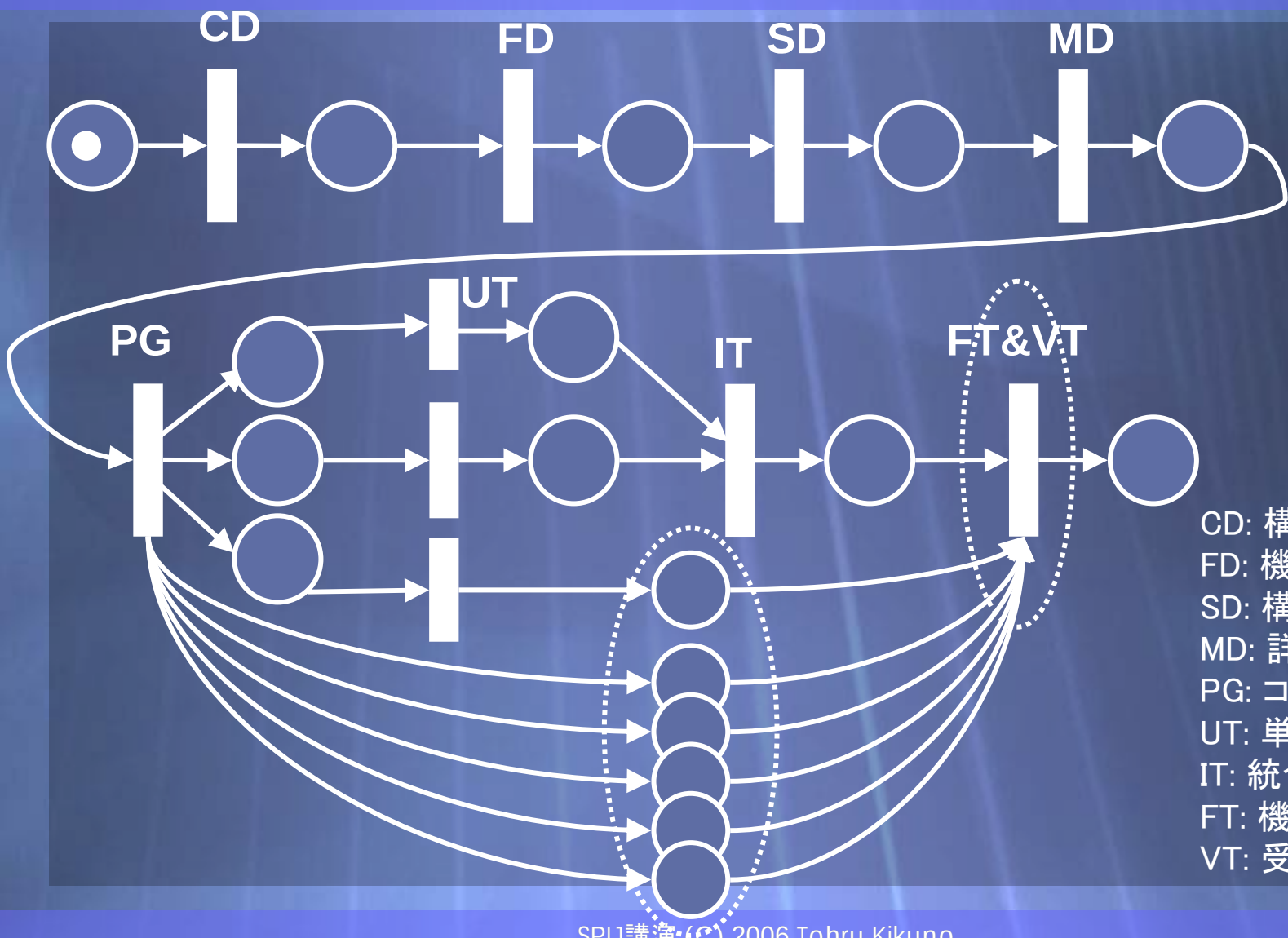
プロセス改善の枠組み

- 現状プロセスの記述
- 現状プロセスの分析
- 改善計画の作成
- 利益予測
- 改善計画の実行
- 改善計画の評価

対象ソフトウェア開発プロジェクト

- 機器組込型のソフトウェア開発
  - ほぼ，機能や規模の同じ開発が続く
  - 客先からの注文がいろいろ異なる
- あるプロジェクトPR1で
  - プロジェクトが混乱状態に陥った.
  - その改善計画を立案したい
  - ペトリネットでプロセスの記述を行い，改善案も記述する.

現状プロセスの記述

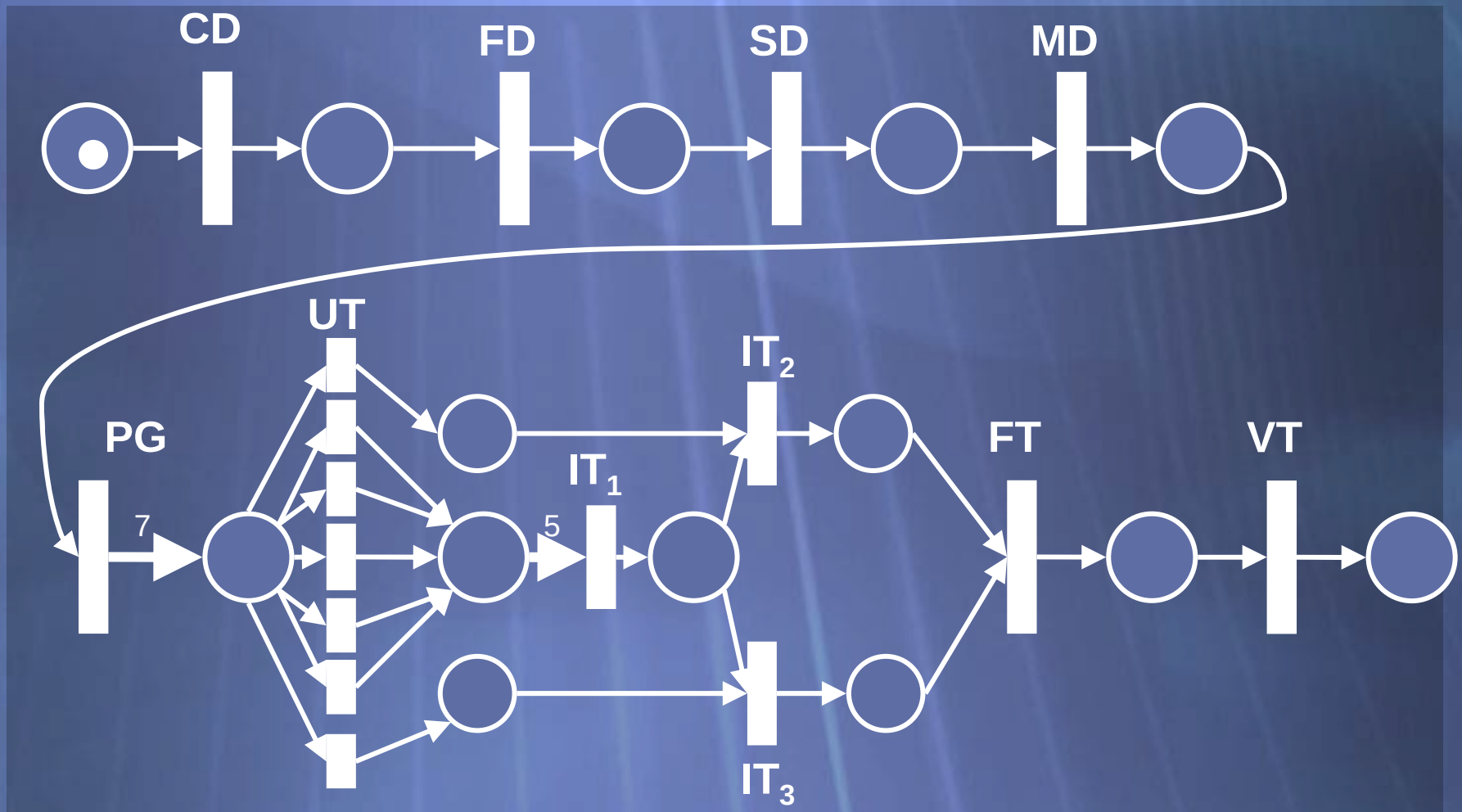


CD: 構想設計
FD: 機能設計
SD: 構造設計
MD: 詳細設計
PG: コーディング
UT: 単体テスト
IT: 統合テスト
FT: 機能テスト
VT: 受入テスト

問題点

- PG (コーディング)の直後にほとんどのモジュールがFT & VTへ流れている.
- そもそもFT & VTが並行に実行されてしまっている.
  - 図には書いていないが更に複雑なプロセスがFT & VTの内部に存在する.

改善プロセスの記述



改善効果の予測

- 改善後のプロセスで改善前と同様の作業を行うことを考えて，効果の予測を行う．
- プロジェクトPR1（実際には問題が発生したプロジェクト）について改善を実施したと仮定した時の不具合数と工数を予測
  - 各工程での発見不具合数を理想値を使って配分
  - その不具合数に基づき，工数を予測

PR1に関するシミュレーション

PR1 Original (実績)	CD,FD,SD,MD	PG	UT,IT	FT	VT	その他	計
不具合(個数) (%)		14 (4.5)	0 (0)	219 (69.7)	81 (25.8)		314 (100.0)
工数 (人日) (%)	399 (31.4)	134 (10.5)	111 (8.8)	448 (35.3)	151 (11.9)	27 (2.1)	1270 (100.0)

PR1 Improved (シミュレート)	CD,FD,SD,MD	PG	UT,IT	FT	VT	その他	計
不具合(個数) (%)		165 (52.6)	83 (26.4)	50 (15.9)	16 (5.1)		314 (100.0)
工数 (人日) (%)	399 (31.4)	155 (14.3)	199 (18.4)	227 (21.0)	76 (7.0)	27 (2.5)	1083 (100.0)

約10%の工数削減

改善効果の評価

- PR1での改善をふまえたプロジェクトPR2を実施
  （PR2は改善されたプロセスを実施）
- PR2では品質・工数の削減に成功
  - では，元のままで実施していたらどうなっていたのかをシミュレーションで調べてみる．

PR2に関するシミュレーション

PR2Original (シミュレート)	CD,FD,SD,MD	PG	UT	IT	FT	VT	その他	計
不具合(個数) (%)		0 (0)	0 (0)	(18.7)	312 (70.1)	50 (11.2)		445 (100.0)
工数 (人日) (%)	303 (25.5)	59	0	127	425 (35.7)	244 (20.5)	33 (2.8)	1192 (100.0)

FT工程での不具合発見の大幅な減少

PR2Improved (実績)	CD,FD,SD,MD	PG	UT	IT	FT	VT	その他	計
不具合(個数) (%)		141 (31.7)	78 (17.5)	(18.7)	93 (20.9)	50 (11.2)		445 (100.0)
工数 (人日) (%)	303 (28.9)	75 (7.1)	5.4 (5.1)	127 (12.0)	215 (20.5)	244 (23.2)	33 (3.2)	1051 (100.0)

約12%の工数削減

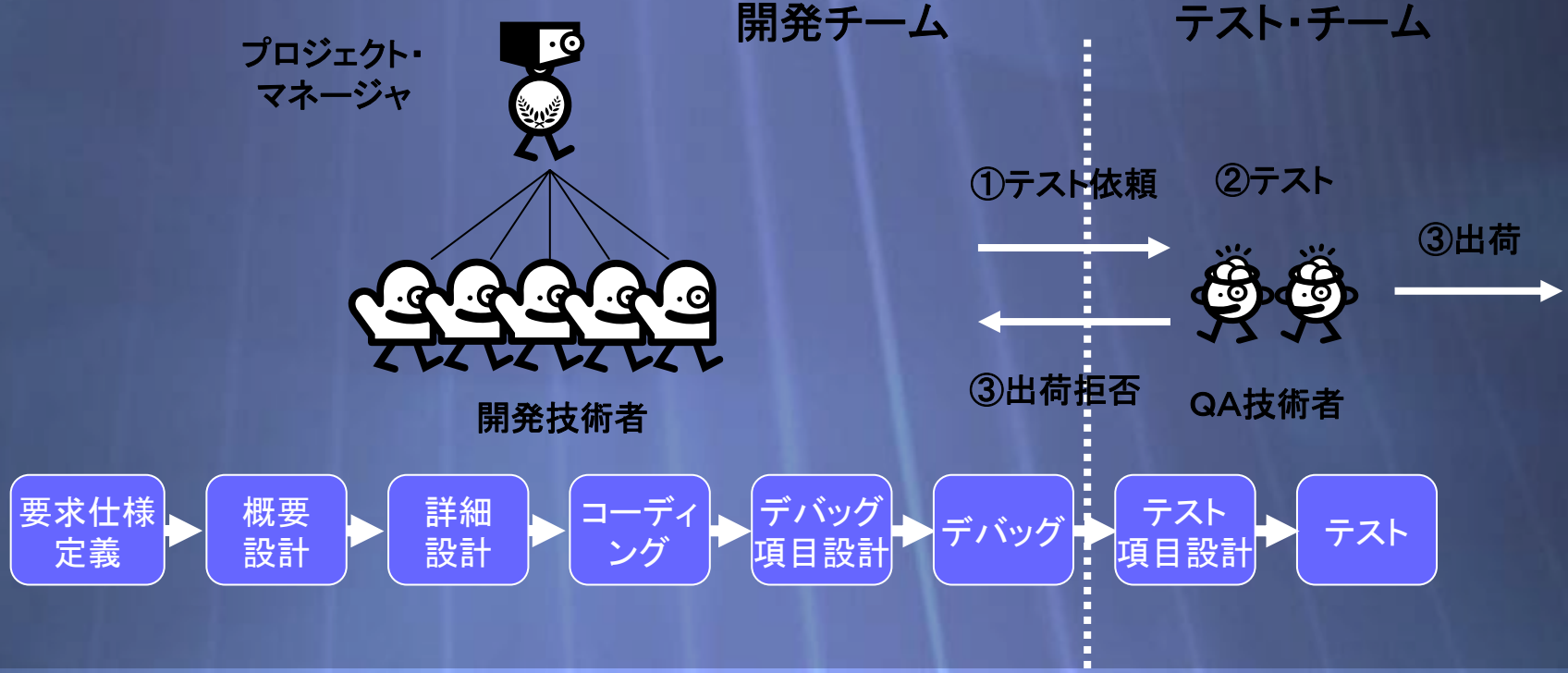
テスト実施プロセスの改善事例

Quoted from

Tsuneo Yamaura, Ph.D. dissertation (2006).

第3世代テストプロセス

- 開発プロジェクトとは独立にテストチームがある.
  - 「プログラムにはバグがある」との性悪説でテストが進む.
  - テストの技術や経験を備えたプロ集団.

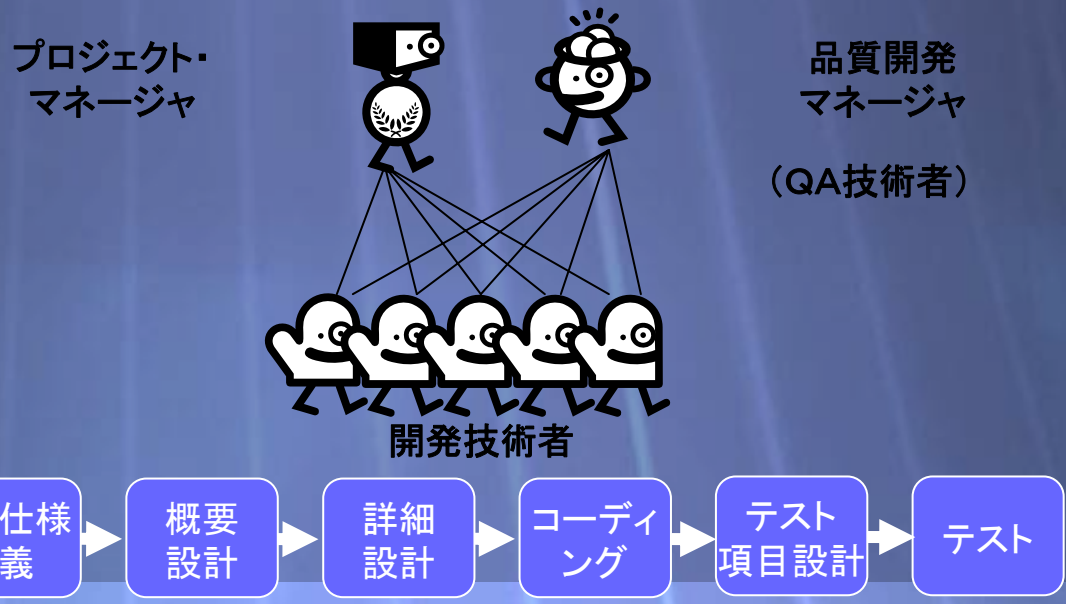


第4世代テストプロセス(外的な効率改善)

- 第3世代テストプロセスの限界を超えてテストプロセスの効率を向上させるため，第4世代テストプロセスを提案.
  - 開発プロセスの期間，必要工数の削減.
  - 第4世代テスト・プロセスの提案.
    - “Fourth Generation Test Process,” SEC Journal No.5, pp.6-15, January, 2006

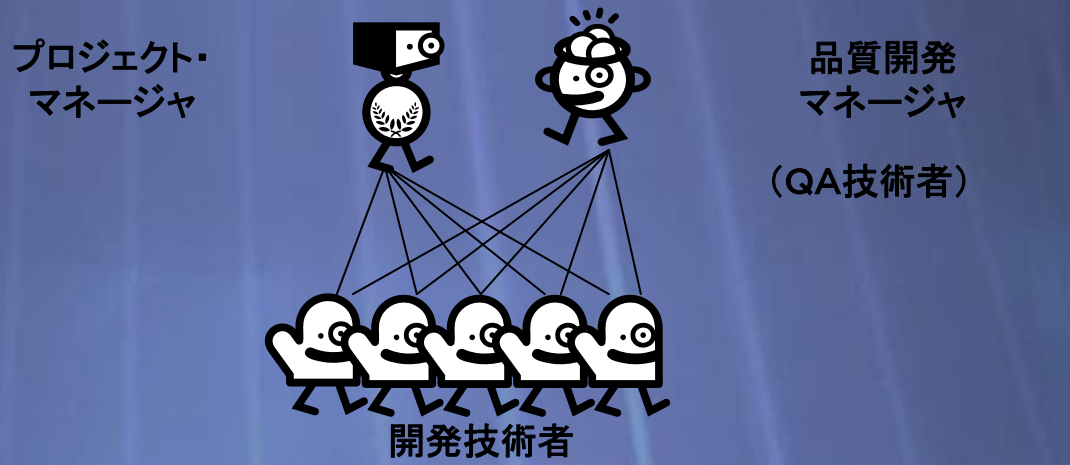
第4世代テストプロセスの概要 (1)

- 第3世代のデバッグプロセスとテストプロセスを1つにまとめる.
  - ⇒ デバッグ・プロセスの期間分が短縮できる.
  - QA技術者 (品質開発マネージャ) が, 開発者を指揮・監督して品質制御プロセスを実施する.



第4世代テストプロセスの概要 (2)

- 開発技術者が実際の品質制御プロセス(テストの設計，作成，実行)を実施 .
- 品質開発マネージャが品質制御プロセス(デバッグプロセスとテストプロセス)を指揮・制御.
- 品質開発マネージャは品質管理の最高責任者として開発プロジェクトに関与.



第4世代テストプロセスの評価（総合評価）

- 総合評価
  - 5つのメトリクスによる評価

(1) 開発期間の短縮
(2) テスト資源の削減
(3) 生産性の向上
(4) 品質の向上
(5) 教育，士気など

適用プロジェクト概要

- Webベースの通信系アプリケーション．記述言語はC++．
- プロジェクトAは新規開発．プロジェクトB～Mは，プロジェクトAのバージョンアップ．（プロジェクトAは，特異値として，評価対象から除外）

プロジェクト		開発規模（KLOC）	全工数(人週)	開発期間(週)	開発人員(人)
プロジェクトA	（3世代その1）	103.3	456	40	12
プロジェクトB	（3世代その2）	4.6	20.5	13	2
プロジェクトC	（3世代その3）	5.6	25.5	13	3
プロジェクトD	（3世代その4）	5.9	33	16.5	3
プロジェクトE	（3世代その5）	5.0	22	16.5	3
プロジェクトF	（4世代その1）	15.1	63	15	4
プロジェクトG	（4世代その2）	12.0	52	18	3
プロジェクトH	（4世代その3）	10.1	41	13	3
プロジェクトI	（4世代その4）	15.1	57	19	3
プロジェクトJ	（4世代その5）	6.6	26	10	3
プロジェクトK	（4世代その6）	5.8	23.5	10	3
プロジェクトL	（4世代その7）	8.4	34.5	17	2
プロジェクトM	（4世代その8）	6.2	24.5	11	2

詳細評価: 開発期間の短縮

- 全開発プロセスに占める品質確保プロセス(デバッグ＋テスト)の割合で評価.

テストプロセス	品質確保工数(全体に占める割合)平均
第3世代	44.8 %
第4世代	35.1 %

有意水準1%で有意

テストの資源削減（その1）

- テスト項目1件あたり，何件のバグを摘出できるかで評価.

テストプロセス	1テスト項目の平均バグ摘出率
第3世代	6.78 個
第4世代	8.85 個

有意水準1%で有意

テストの資源削減（その2）

- ソースコード1KLOCあたり，何件のテスト項目を作成したかで評価．

テストプロセス	1KLOCあたりの平均テスト項目数
第3世代	130.5件/KLOC
第4世代	93.8件/KLOC

有意水準1%で有意

生産性の向上

- １人週の工数あたりで生産したソースコード生産ステップ数で評価.

テストプロセス	1人週あたりの平均ソースコード生産ステップ数
第3世代	0.21KLOC/人週
第4世代	0.25KLOC/人週

有意水準1%で有意

品質

- 1KLOCあたりのバグ件数で評価.
  - ⇒　平均値では改善されているが，統計的有意性は保証されない.（ただし，悪くはなっていない）

テストプロセス	1KLOCあたりの平均バグ検出数
第3世代	8.9件/KLOC
第4世代	8.1件/KLOC

有意水準10%でも有意ではない（p=0.28）

第4世代テストプロセスのまとめ

- 開発期間の短縮
  - ⇒ 効果あり
- テスト資源の削減
  - ⇒ 効果あり
- 生産性の向上
  - ⇒ 効果あり
- 品質の向上
  - ⇒ 平均値は改善されたが，
    統計的な有意性は明確ではない．
- 教育，士気など
  - ⇒ 効果あり

まとめ

1. 信頼性向上への対策

   ソフトウェア信頼性向上へのこの20年間の取り組みを大学の研究室という窓口を通して振り返ってみた．テストに始まり，プロセスに移り，再びテストに戻ってきている．

2. 品質予測への取り組み

   ソフトウェア開発プロジェクトが抱える危険について，ベイズ識別器を適用して予測する試みについて述べた．限定的ではあるが一応の成功を収めることができた．

3. 2つのケーススタディ

   これこそが我々の研究室の礎であり，今後も適用事例を増やしたいと考えている．