測定可能なプロセスを対象とした フォーマルメソッド導入に関する提案

九州大学 大学院システム情報科学研究院
荒木・日下部 研究室

発表概要

- 背景
- 目的と方針
- フォーマルメソッドの導入例
- 結果と考察
- まとめと今後の課題

背景

フォーマルメソッドの課題

- (一部の高信頼性システムなど) 限定的な導入事例
  - 定量的な効果が分かりにくい
  - 具体的な導入方法がよくわからない
  - 人材の育成・確保

  - マネジメントの理解が得にくい

- 具体的な導入の指針と効果を定量的に示すことが必要

目的と方針

目的と方針

- 目的
  - 測定可能な個人(チーム)プロセスがあれば，個人(チーム)プロセスへのフォーマルメソッドの導入指針が立ちやすい，という仮定を検証
    (ログは手掛かりの宝庫では?)
- 方針
  1. ベースラインとなるプロセスのデータを測定する
     - 作業時間，欠陥データ（欠陥の種類・内容・修正時間）
  2. プロセスデータを分析しフォーマルメソッド導入を検討(例VDM)
  3. 導入前後のプロセスデータを比較し評価を行う
  4. 測定可能な個人(チーム)プロセス：(例PSP)
     - 作業内容が定義されておりプロセスの分析に必要なデータが収集可能
     - 個人(チーム)が要求仕様を受け取ってモジュール開発を行うプロセス

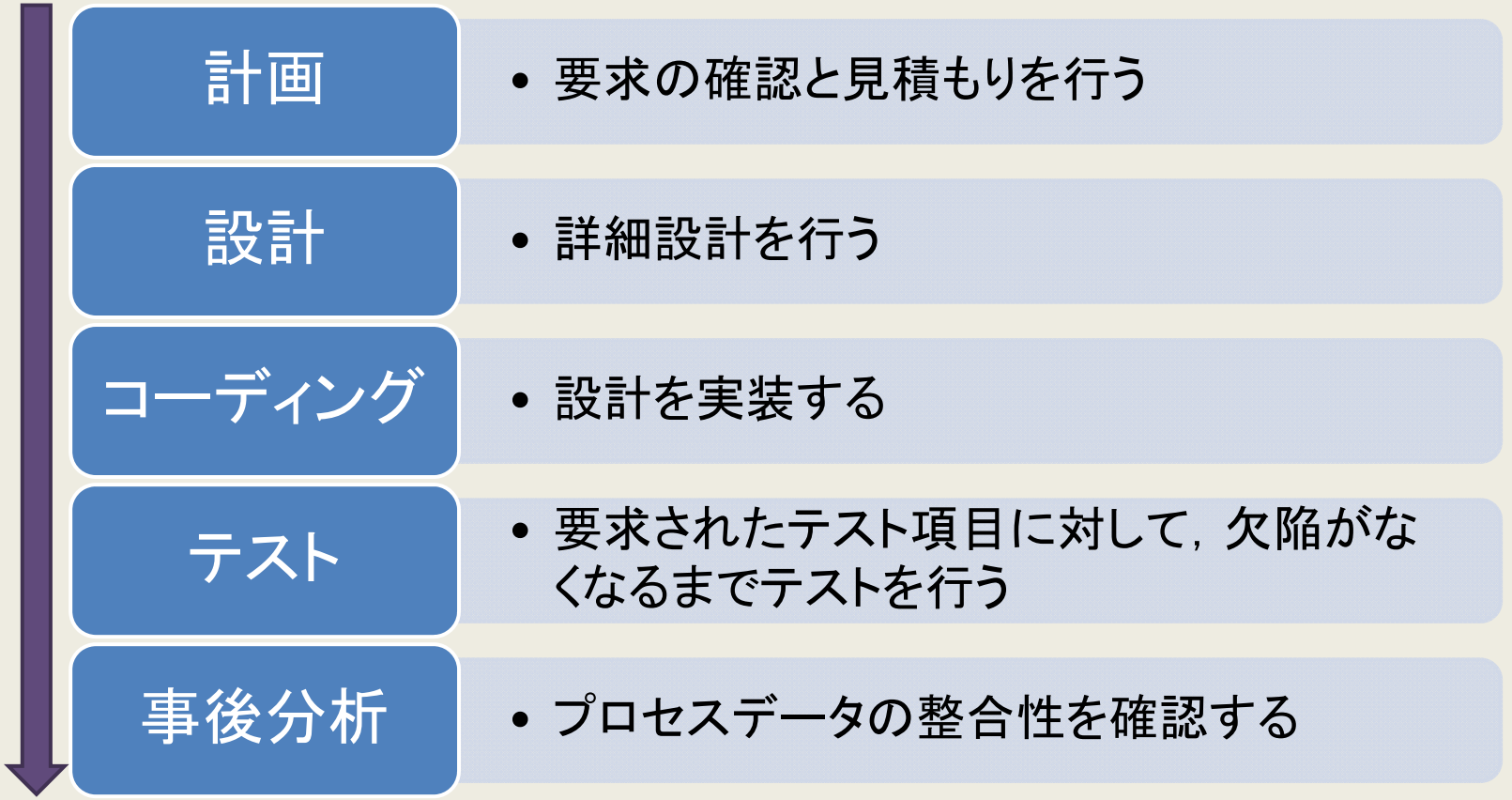
実施の方法

- プロセスデータの測定に用いた方法
  - CMU/SEIで開発されたPersonal Software Process（PSP）*1を用いてプロセスデータの測定を行う
- 測定対象（ベースラインプロセス）
  - PSPの演習課題を学生(修士)が行った事例を対象とした

課題番号	課題内容	実施プロセス
1	LinkedList	ベースライン プロセス
2	プログラムの規模（LOC）カウンタ	
3	PROBE 見積もり値の計算	
4	相対規模の計算	
5	数値積分	提案プロセス1
6	関数P(x) の積分値がp になるx の値の計算	提案プロセス2

*1 Personal Software Process, PSPはカーネギメロン大学のサービスマークである

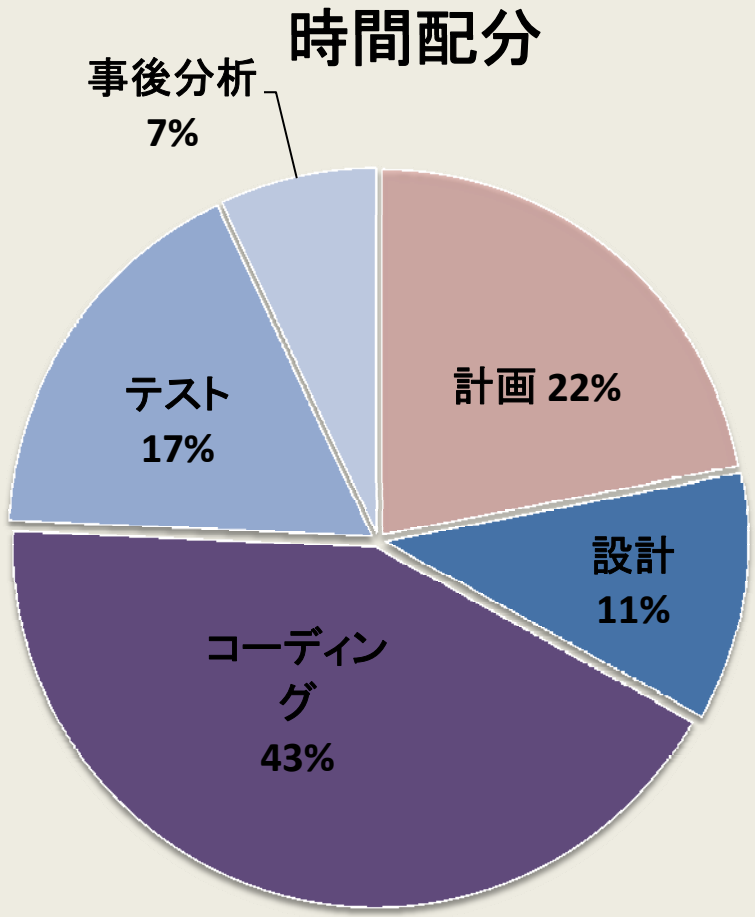
ベースラインプロセス



- 設計にはクラス図とシーケンス図を使用
- 実装にはJavaを使用

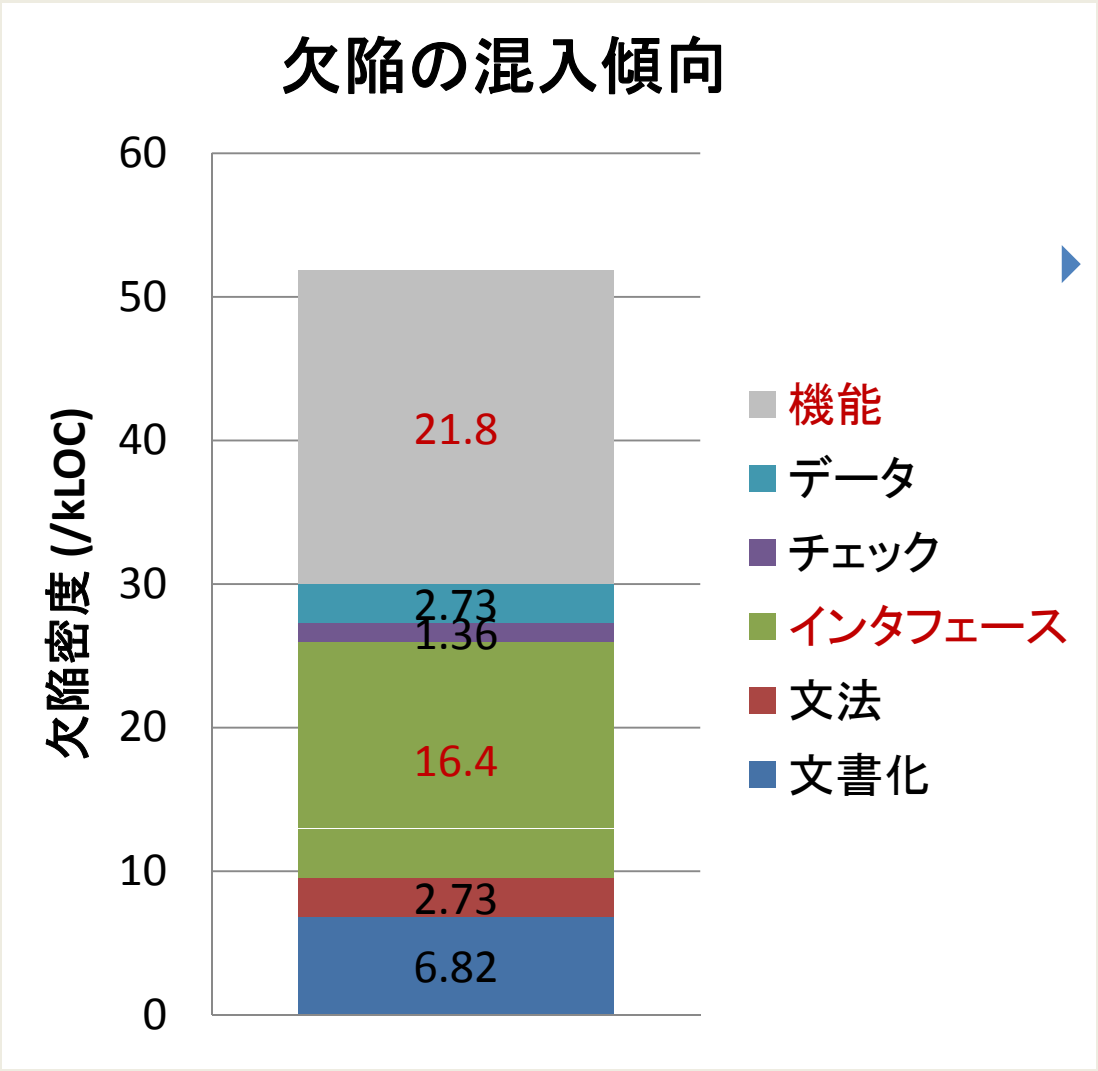
フォーマルメソッドの導入方法

ベースラインプロセスのデータ(時間配分)



- 設計にかける時間が短い
- コーディングに多くの時間を費やしている

ベースラインプロセスのデータ(欠陥データ)



- 機能欠陥・インタフェース欠陥が多く発生している
  - 今回はこの2つの欠陥に着目する

着目した欠陥データの細分化

- 欠陥の原因に基づいてさらに小分類に分ける

インタフェース型(I型)	機能型(F型)
詳細化が足りていないことに起因する欠陥(I-1)	繰り返し処理に関する欠陥(F-1)
その他(I-2)	実装間違いに関する欠陥(F-2)
	条件に関する欠陥(F-3)

- 平均修正時間

欠陥型	平均修正時間(分)
I-1	15.8
I-2	1.6
F-1	10.3
F-2	6.8
F-3	3

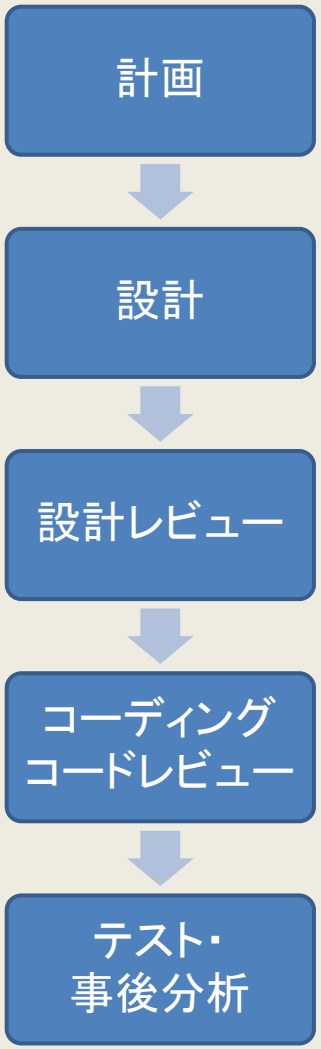
■ターゲットとする欠陥

- I-1
- F-1
- F-2

導入方針

- 出来るだけシンプルな導入法
  - Light-weightなフォーマルメソッドの利用を考える
- 着目した欠陥の作り込み防止を主に考える
  - 必ずしも万能な方法を考えるわけではない
- ツールを活用
  - VDM Toolsによる自動チェックを活用する方法を考える

導入対象



- PSP 2.0の作業プロセス（左図）にVDMを導入することを考えた
- 変更点
  - 設計
    - UMLベースの設計
    - VDM++を用いた設計
- レビュー
  - チェックリストによる確認に加え設計レビューでツールの機能を活用

導入方法の提案1

- ベースラインプロセスに追加したこと
  - チェックリストに基づくレビューを追加
- 対処する欠陥
  1. インタフェース - 詳細化が足りていないことに起因する欠陥
  2. 機能 – 繰り返し処理に関する欠陥
- 対処方法
  1. 依存関係のあるメソッドについて陽な仕様記述を行う
  2. シーケンスへの処理を設計段階で厳密に記述し，確認する
- 作業内容
  1. 依存関係のある部分について陽に仕様を記述し，ツールで構文・型検査を行う
  2. シーケンス処理を陽に記述し，処理が正しいことを確認する

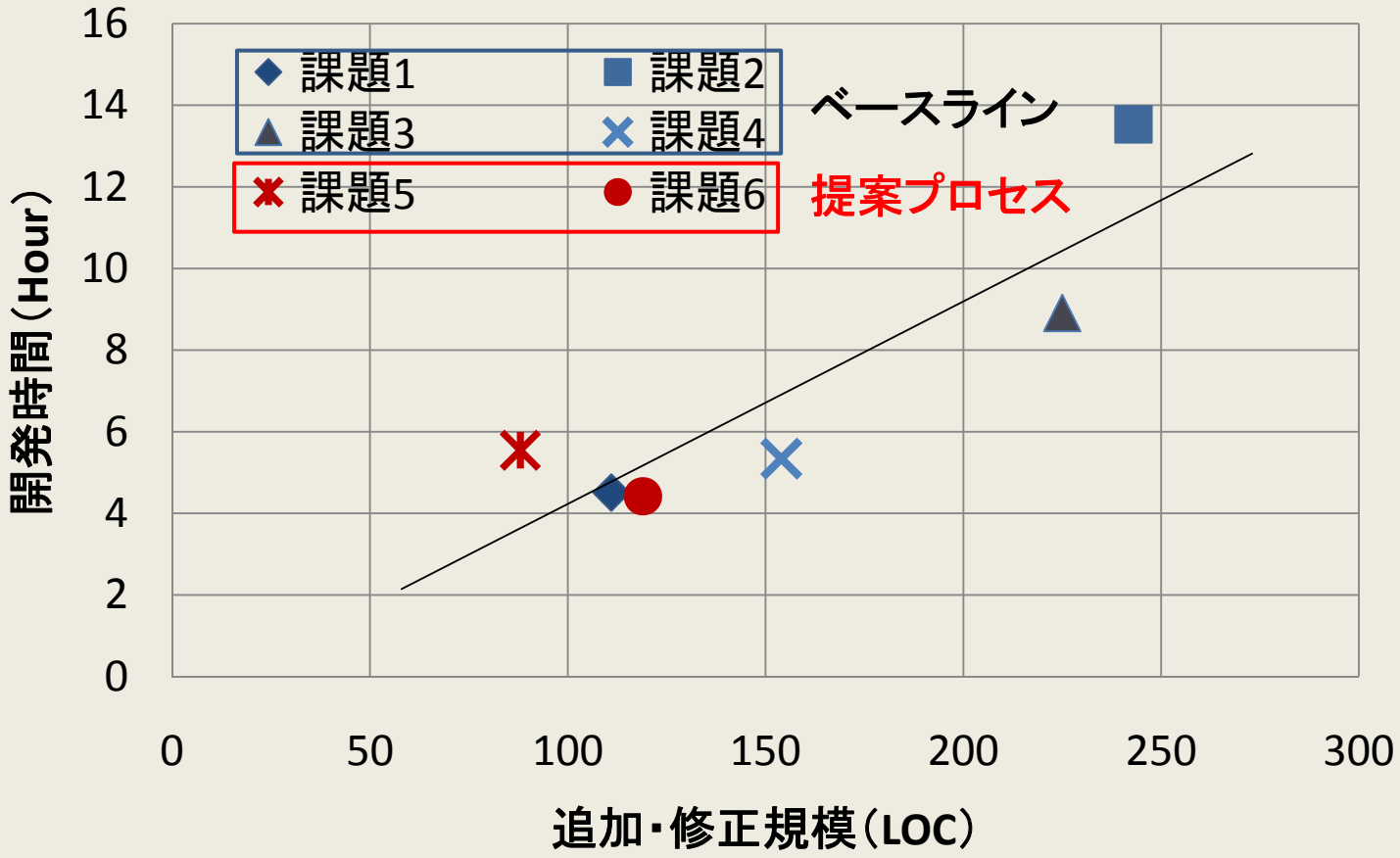
導入方法の提案2

- 導入方法1に追加したこと
  - 仕様アニメーションによる設計検証を追加
- 対処する欠陥
  1. 実装間違いに起因する欠陥
- 対処方法
  1. 仕様アニメーションによる設計の妥当性確認を行う
- 作業内容
  1. 導入方法1に加えて設計フェーズで検査したい処理を実行可能なレベルまでVDM++記述の詳細化を行う
  2. 設計レビューで仕様アニメーションによる動作確認を行い，必要があれば修正を行う

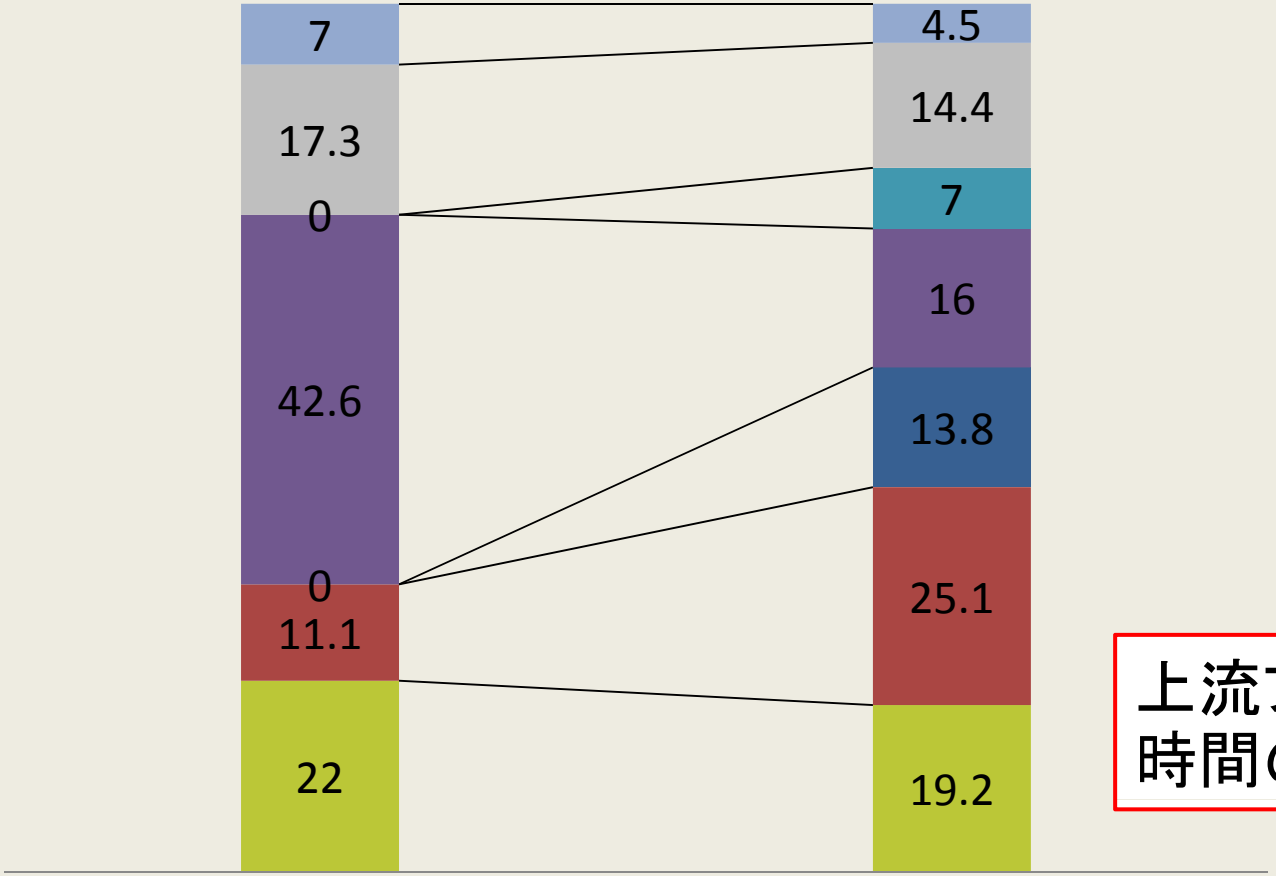
結果と考察

生産性の変化

- 導入前後で生産性の低下は見られなかった

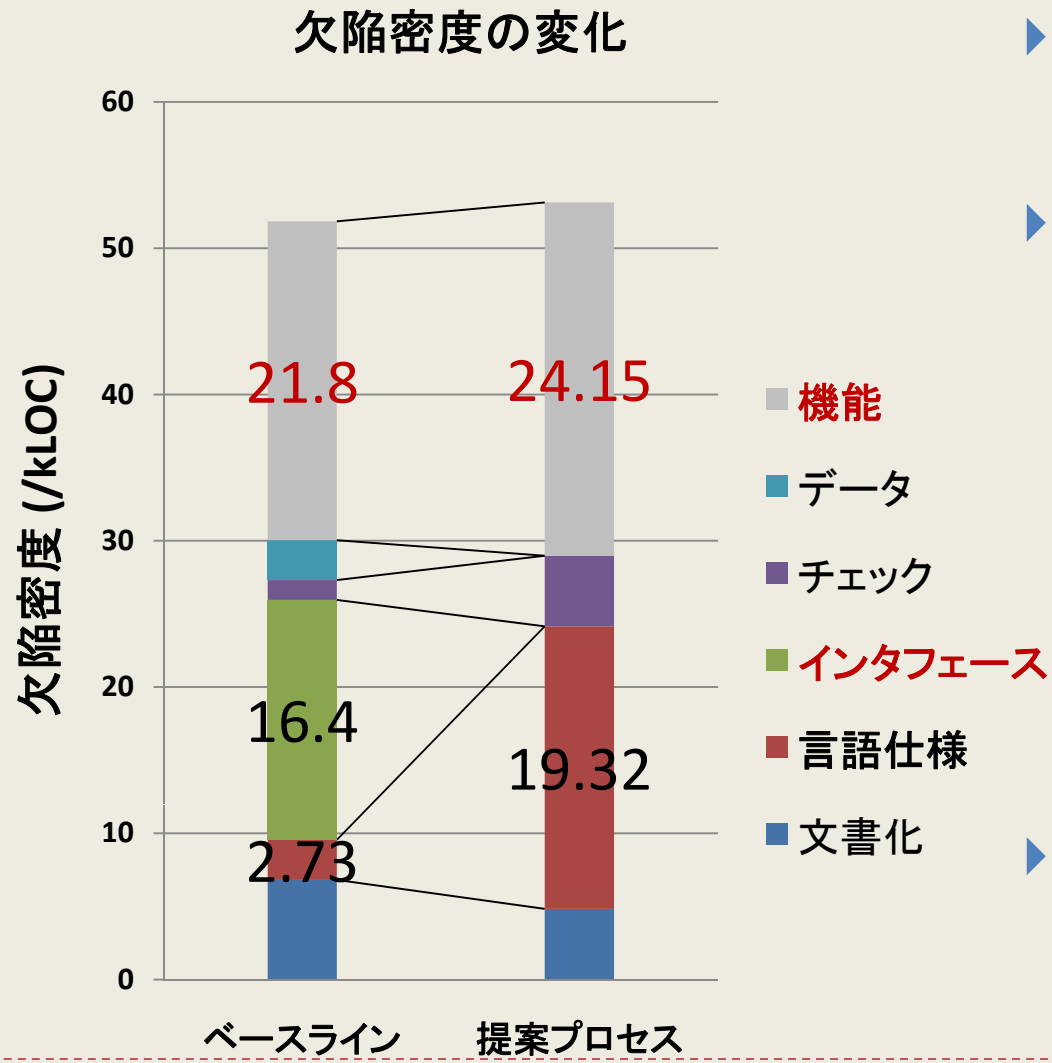


時間配分比率の変化



上流フェーズに作業時間の重心が移った

欠陥の密度の変化



- インタフェース型
  - なくなった
- 機能型欠陥
  - ベースライン
    - 87.5%がテストで除去されていた
  - 提案プロセス
    - ほとんどが設計レビューで除去
    - 20%のみがテストで除去
- 全体の欠陥密度
  - 変化していない

まとめと今後の課題

まとめ

- 概要
  - 欠陥型に着目したプロセスのデータ分析に基づくVDM++による仕様記述の導入方法の提案と効果の評価を行った
- 結果
  - 提案したプロセスでは，インタフェース型欠陥の作り込みを減らし，機能型欠陥を上流で除去できた
  - 導入による生産性の低下は見られなかった
  - 仕様アニメーションによる妥当性確認の効果測定は今後の課題
- 自動ツールに過度に頼らず，プロセス改善を通した品質向上に有効(教育的には好ましい)