RICOH

# 不具合を分析して設計を駆動する

山田　大介
株式会社リコー
ソフトウエア研究開発本部
ソフトウエア工学研究センター

RICOH

## はじめに

ソフトウエア開発プロセスは、ウォーターフォールプロセスから反復プロセスへと変化しつつあります

それに伴い、テストの時期と方法も従来スタイルからの脱却が必要になってきています

レビュー／インスペクション／テストを、いつ、どのように、実施するのか、という課題もある反面、

テスト結果を次の反復に活かすことも可能となります

本セッションでは「テストで設計を駆動する」というコンセプトを紹介します

RICOH

## 目次

- 反復プロセスでのテスト
- 効果的かつ効率的なレビューとテスト
- 小刻みなプロセス改善
- 設計／QA／SEPGの連係
  （QAは「品質保証」、SEPGは「プロセス改善」）

RICOH

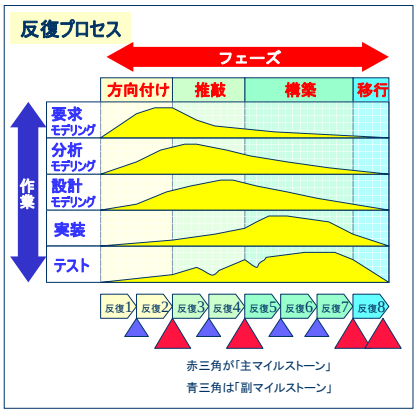
## 反復プロセスとテスト

反復の特徴と課題

RICOH

## 反復プロセスとテスト

- 反復プロセスではプロジェクトの最初からテストを行う
- 何をテストするの？
- どうやってテストするの？
- 効果と効率のバランスは？
- テスト結果を次の反復に活かすには




RICOH

## 反復の特徴

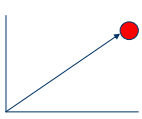
- 反復毎に計画を再設定
  - 最終ゴールが徐々に明確に
  - 反復毎に改善施策が打てる？
- 開発は線形に進まない
  - 前半と後半で進み方が異なる
  - フェーズごとにテストの方法とボリュームを変える？
- 作業は直列ではない
  - 設計とテストが繰り返される
  - テストを先行させる設計ができる？

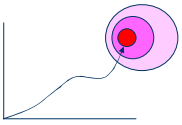
⇒品質の高いソフトウエアを作り出す可能性は高くなったが、その分、考慮すべきことが多くなった

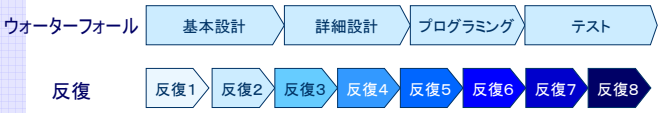
RICOH

## 反復毎に計画を再設定

- ウォーターフォール
  - 最初に計画立案
  - 1パスですべて終了
- 反復
  - 反復毎に計画立案
  - 複数回の反復を経て終了







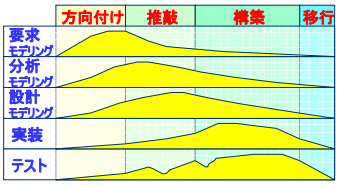

RICOH

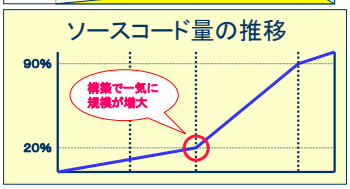
## 開発は線形に進まない

- ソースコードの成長に特徴あり

⇒構築フェーズの突入時にギアチェンジ

- 推敲フェーズまでは、モデルの骨格作り
  - 推敲フェーズ完了時点で20%のソースコード
- 構築フェーズ以降は、骨格に対する肉付け
  - 構築フェーズ完了時点で90%のソースコード




RICOH

## 作業は直列ではない

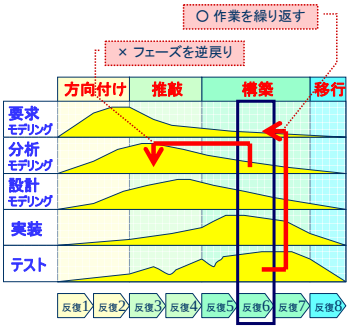
- 作業を繰り返すのであればOK

⇒洗練化

- フェーズを逆戻りするのはNG

⇒手戻り

- 洗練化なのか手戻りなのかを意識して、各作業を組み立てる
  - テスト主導の設計に近づく




RICOH

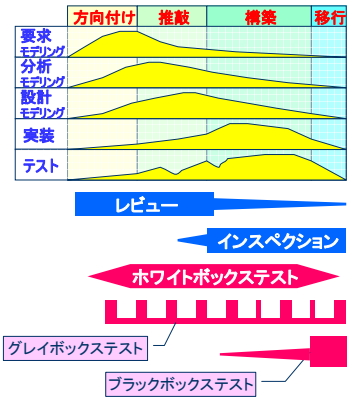
## 効果的かつ効率的なレビューとテスト

レビュー
インスペクション
グレイボックステスト

RICOH

## いつ、なにを実施するのか

- ウォークスルーは、トレーニング
- レビューは、コンセンサスの形成
- インスペクションとテストは、品質改善
- それぞれ目的を明確にして、作業量を決定




RICOH

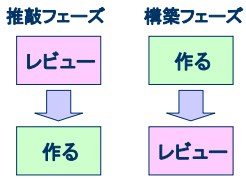
## レビュー／インスペクション／テスト

- 前半はレビューやウォークスルー
  - 設計ドキュメントをレビューせよ
  - 設計方針を固める
  - 設計者同士の理解共有を促進
- 後半はインスペクション
  - ソースコードをインスペクションせよ
  - 障害を探して、高品質なものづくり
- グレイボックステストで骨格の確認
  - スカスカでもテストせよ
  - 骨格が崩れていないかを監視する

RICOH

## レビュー

- 推敲フェーズまでは、
  レビューファースト
  - ドメインに素直なモデル骨格の作り込み
- 構築フェーズでは、
  レビューラスト
  - モデル骨格が崩れていないかの検証




- 構築フェーズの骨格検査
  1. 骨格矯正レビュー
  2. やや肥満レビュー
  3. 慢性症状レビュー

モデルで可視化

RICOH

## TIPS:レビューが必要な症状一覧

- 骨格矯正
  - 骨格がもともとずれていた
  - 無理な肉付けにより骨格がずれていく
- やや肥満
  - ひとつのクラスが肥大化する
  - ひとつのクラスの状態数が増加する
  - クラスの数が増加する
- 慢性症状
  - 毎回モデル骨格自体を検討してしまう
  - 機能追加のたびに考え込んでしまう
- 自覚症状が無い場合の検知方法が大切

RICOH

## インスペクションの特徴

- レビューよりも厳密なプロセス
  - 設計の欠陥を検出
  - 設計者同士ではなく、インスペクションリーダが運営
  - 「必要に応じて実施」ではなく、「計画的に実施」

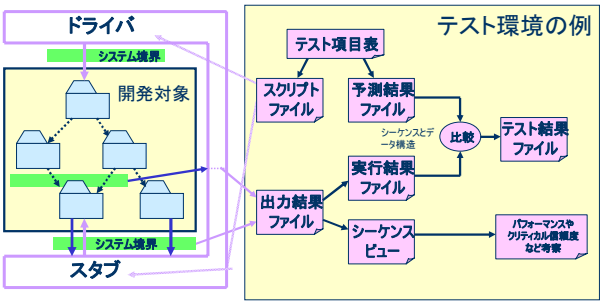

- 直交障害分類により、障害を分析し、プロセス改善

RICOH

## グレイボックステスト

- モデル骨格を維持する
  - システム境界と重要なサブシステム間インターフェイス

実装がスカスカでもテストする




RICOH

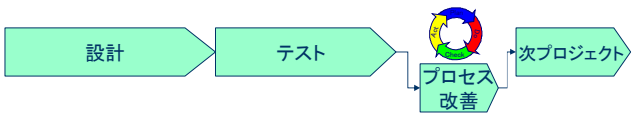
## 小刻みなプロセス改善

回想会
直交障害分類

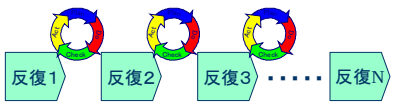
RICOH

## 小刻みなプロセス改善

- 重たくて、長いプロセス改善ループ




- 小刻みなプロセス改善ループ
  - 品質データを分析することにより設計を駆動



RICOH

## プロセス改善ループを小刻みに回す

- 反復毎の回想会
  - 反復毎に振り返り、次の反復に進む
  - プロセスや仕組みを改善
  - 全員参加することが大切


- 直交障害分類での軽いプロセス改善ループ
  - 障害データは貴重な情報源
  - 障害データを分析することにより設計を駆動する
  - どのような属性を取得するか自体をテーラリング

RICOH

## 回想会

- 反復毎に「回想会」を実施
  - 反復の良かったこと／悪かったこと、を振り返る
- 「反省会」だと後ろ向きな印象
  - 多くの人がうつむき加減
  - 何を言われるのだろうという受身の姿勢

RICOH

## TIPS：回想会

- 全員参加
  - “強い“マネージャは時には欠席してもらう
- 否定や非難をしない
  - 建設的な発言
- みんなでより良いプロジェクトにする、という意識
  - 素直に振り返る

RICOH

## 障害処理票の役割

- 品質の管理 (Quality Control)
  - 迅速かつ確実な解決への進捗管理
  - ソフトウェアの品質が各ステップで定量的かつ正確に管理・把握できること
- プロセス改善 (Process Improvement)
  - 障害を分析し根本的な原因を改善する
  - 未然防止のための仕組みを構築する

⇒直交障害分類による品質データの蓄積と分析

RICOH

## 直交障害分類

- 障害の特性を記録しておく

| 属性 | 定義 |
|---|---|
| 検出工程 | 障害を検出した活動 |
| 混入工程 | 障害が混入した工程 |
| 重要度 | 危険／重大／軽微 |
| 問題分類 | 障害／質問／意見 |
| 原因パターン | 障害が混入した原因<br>要求仕様の誤解／インターフェイスの不整合／設計ミス／Typo |
| 対策パターン | 対策方法<br>設計変更／ソースコード改修／割り切り |

RICOH

## 直交障害分類

- 完全に分類できること
  - どのような障害に対しても属性値が見つかる
- 属性にダブりが無いこと
  - 属性の項目が直交している
- 属性値が一意に選択できること
  - ひとつの選択肢がみつかる
- 属性値が多すぎないこと
  - 7±2以内に収める
- 属性と属性値の説明があること
  - あいまい性を排除

RICOH

## 直交障害分類シート(例)

| 属性 | 定義 | 記述者 | 属性値 | 説明 |
|---|---|---|---|---|
| 検出工程 | 障害を検出した活動 | 障害検出者 | 要求レビュー | |
| | | | 設計レビュー | |
| | | | ソースコードレビュー | |
| | | | 単体テスト | |
| | | | システムテスト | |
| | | | 商品テスト | |
| 混入工程 | 障害が混入した工程 | ロギングミーティング | 要件定義 | |
| | | | 基本設計 | |
| | | | 詳細設計 | |
| | | | プログラミング | |
| | | | 障害修正 | |

RICOH

## TIPS:直交障害分類

- 軽い仕組みを作る
  - 記入に時間がかからない
  - 選択肢に迷わない
- プロジェクトの特性に応じてテーラリング
  - 必須とオプションの属性を設定
  - プロジェクトの特性に応じて取捨

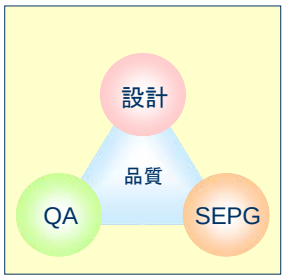
RICOH

## 役割と連係

RICOH

## 設計／QA／SEPGの連係

- 反復プロセスでは作業とフェーズが分離しているため、より密接な連係が必要
- 分散型
  - 作業の順次処理が前提
  - 組織間の壁
- 集中型
  - 人員のスキルが高い前提
  - 専門スキルの壁




RICOH

## 3者間の思惑？

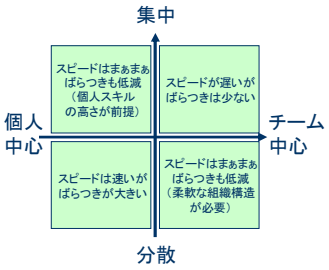
- 設計とQA
  - 設計⇒最後に品質を見られても。こっちは頑張っているのに。
  - QA⇒上流から品質を見ていきたいが、ドキュメントが読めない。そもそもドキュメントがない。
- 設計とSEPG
  - 設計⇒どれだけの効果が有るのかわからない。時間も取れないし。
  - SEPG⇒やり始めなければ改善は始まらない。とにかく測って欲しい。
- QAとSEPG
  - QA　⇒SEPGは大変だね
  - SEPG　⇒QAは大変だね

⇒もっとお互いを知って、協調しあうところからスタート？

RICOH

## 組織(構造)と個人(体質)

- 柔軟な組織構造と個人の体質改善、が必要
- 柔軟な組織構造
  - 集中と分散の間の適切な位置へ
- 個人の体質改善
  - 少数精鋭からチームの相乗効果を発揮できる体質へ
- 今の位置取りを把握し、次なる展開へ



RICOH

## まとめ

- 反復プロセスでのテスト
  - 反復によるメリットを活かそう
- 効果的かつ効率的なレビューとテスト
  - いつ、何を、どのようにテストするのかの戦略を作ろう
- 小刻みなプロセス改善
  - 主体性をもって取り組もう
- 設計／QA／SEPGの統合
  - スピード間のある連係をしよう

RICOH

## 最後に

- 可視化と定量化で、設計を駆動
  - （設計の）可視化　　：見ながら⇒作る
  - （成果物の）定量化　　：作って⇒測って⇒改善
  - 設計で品質を作りこもう
- 設計／QA／SEPGの連係により品質の高い“ものづくり”
  - 設計の次がテスト、なのではない
  - 今こそ、お互いの信頼関係が大切

RICOH

## ご清聴ありがとうございました