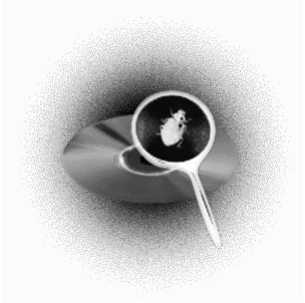
テストの改善、テストによる改善



ソフトウェアテストシンポジウム（JaSST）2009東海
2009/10/16（金）
電気通信大学 電気通信学部 システム工学科
西　康晴

自己紹介

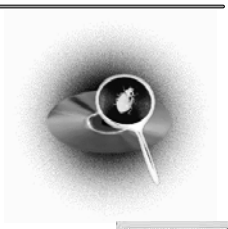
- 身分
  - ソフトウェア工学の研究者
    - » 電気通信大学 電気通信学部 システム工学科
    - » ちょっと「生臭い」研究／ソフトウェアテストやプロセス改善など
  - 先日までソフトウェアのよろず品質コンサルタント
- 専門分野
  - ソフトウェアテスト／プロジェクトマネジメント／QA／ソフトウェア品質／TQM全般／教育
- 共訳書
  - 実践ソフトウェア・エンジニアリング／日科技連出版
  - 基本から学ぶソフトウェアテスト／日経BP
  - ソフトウェアテスト293の鉄則／日経BP
  - 基本から学ぶテストプロセス管理／日経BP
- もろもろ
  - TEF: テスト技術者交流会 / ASTER: テスト技術振興協会
  - SESSAME: 組込みソフトウェア管理者技術者育成研究会
  - SQiP: 日科技連ソフトウェア品質委員会
  - 情報処理学会 ソフトウェア工学研究会 / SE教育委員会
  - ISO/IEC JTC1 SC7 WG26（ISO/IEC29119 ソフトウェアテスト）
  - 経済産業省 組込みソフトウェア開発力強化推進委員会

ASTER

SESSAME

SQiP
Software Quality Profession

TEF: Software Testing Engineer's Forum



- ソフトウェアテスト技術者交流会
  - 1998年9月に活動開始
    - » 現在1600名強の登録
    - » MLベースの議論と、たまの会合
  - http://www.swtest.jp/forum.html
  - お金は無いけど熱意はあるテスト技術者を無償で応援する集まり
  - 「基本から学ぶソフトウェアテスト」や「ソフトウェアテスト293の鉄則」の翻訳も手がける
    - » ほぼMLとWebをインフラとした珍しいオンライン翻訳チーム
  - 技術別部会や地域別勉強会が実施されている
    - » プリンタ、Web、AVなど
    - » 東京、関西、九州、東海、札幌など
    - » TestLinkというオープンソースツールの日本語化部会もある

ASTER: Association of Software Test EngineeRing

- ソフトウェアテスト技術振興協会
  - テストを軸にして、ソフトウェア品質向上に関する教育や調査研究、普及振興を行うNPO法人
    - » 2006年4月に設立／理事・会員は無給
  - ソフトウェアテストシンポジウム（JaSST）を開催している
    - » 実行委員は手弁当／参加費は実費＋α
    - » 毎年4Qに東京で開催／今年はのべ約1500名の参加者
    - » 今年は大阪・四国・札幌・九州・東海でも開催／会場はほぼ満席
  - ソフトウェアテストの資格試験（JSTQB）を運営している
    - » Foundation Levelは5784名の受験者・3191名の合格者
    - » Advanced Levelを近い将来に開始
  - 各地でソフトウェアテストの教育を行っている
    - » 札幌や大分などで実績がある
    - » 勉強会の開催などを支援することで、地場の産業振興の定着を図る
  - アジア各国とテスト技術の交流（ASTA）を行っている
  - ソフトウェアテスト開発方法論などの先端技術を研究開発している

ASTER

SESSAME: 組込みソフトの育成研究会

- 組込みソフトウェア技術者管理者育成研究会
  - Society for Embedded Software Skill Acquisition for Managers and Engineers
  - 2000年12月に活動開始
    - » 200名強の会員／MLベースの議論と、月イチの会合
  - http://www.sessame.jp/
- 中級の技術者を10万人育てる
  - PCソフトウェアのような「そこそこ品質」ではダメ
    - » 創造性型産業において米国に劣り、コスト競争型産業でアジアに負ける
    - » ハードウェアとの協調という点で日本に勝機があるはず
  - 育成に必要なすべてを開発する
  - オープンプロダクト／ベストエフォート
    - » 文献ポインタ集、知識体系（用語集）、初級者向けテキスト、スキル標準など
    - » 7つのワークグループ：組込みMOT・演習・MISRA-C・ETSS・子供・高信頼性
  - セミナーだけでなく、講師用セミナーも実施

SQiP：Software Quality Profession

- 名称：
  - 日本科学技術連盟・ソフトウェア品質委員会
- 目的
  - SQiPは、ソフトウェア品質技術・施策の調査・研究・教育を通じて、実践的・実証的なソフトウェア品質方法論を確立・普及することにより、ソフトウェア品質の継続的な向上を目指す
- 3つの視点
  - ソフトウェア品質・実践・普及啓蒙
- 主軸とする活動
  1. 実践的・実証的なソフトウェア品質方法論の確立
  2. ソフトウェア品質方法論の普及促進・資格認定
  3. ソフトウェア品質向上のための国際協力の推進
- 活動方針
  1. ソフトウェア品質追究の重要性訴求
  2. 日本での実践的・実証的ソフトウェア品質方法論の形式知化
  3. グローバルな視野での活動
  4. 新しい課題へのチャレンジ

Safeware翻訳プロジェクト

- 2009年11月に翻訳版が刊行
  - Nancy Leveson（MIT）著
  - 翻訳チーム：松原友夫・片平真史（JAXA）・吉岡律夫（日本機能安全）・青木美津江（電気通信大学）・西康晴（電気通信大学）
  - 翻訳社 発行

多くの組織ではテストの改善が進んでいない

- そもそも改善しようとしていない
- 何をすればいいか分からない
- 何かしているが、テスト工程に効果が出ていない
- テスト工程に効果は出ているが、開発全体に効果は出ていない

テストの改善を進めるにはどうすればよいか

- そもそも改善しようとしていない
  - 改善しようとする
- 何をすればいいか分からない
  - テストの技術とテスト改善の技術を身につける
- 何かしているが、テスト工程に効果が出ていない
  - 効果的なテストに必要な考え方を身につける
- テスト工程に効果は出ているが、
  開発全体に効果は出ていない
  - テストによって開発全体の改善に役立てるために
    必要な考え方と技術を身につける

講演の流れ

- 効果的なテストに必要な考え方
  - 説明責任と持続的革新
- テスト改善の技術・テストの技術
  - 実態駆動型テスト改善
  - モデル駆動型テスト改善
  - クラシックなテストの技術・モダンなテストの技術
- テストによって開発全体を改善する考え方・技術
  - テストによって開発全体を改善する考え方：他工程配慮
  - 不具合モード分析による開発・レビュー・テストの改善
  - 組み合わせテスト分析による開発の改善
  - 他工程配慮的プロセスの例：Wモデル

効果的なテストに必要な考え方

- 説明責任
  - テストとは、ステークホルダーに
    製品やシステムの価値を説明することである
  - そのためには、以下のことを説明できないといけない
    - » 説明の目的と説明相手：ミッションとステークホルダー
    - » 説明範囲：全体像と、それに対する説明内容
    - » 説明単位や説明時期、説明順序など
    - » 実際にテストを進めるために行う技術上・管理上の選択の想定結果と選択理由
- 持続的革新
  - テストとは、開発組織やテスト組織の能力を
    持続的に革新するための作業である
  - そのためには、以下の能力を備えていないといけない
    - » 技術の徹底
    - » 個別技術力の向上
    - » 全体技術力の向上
    - » 向上能力の維持・向上

説明責任を支える考え方

- 説明の目的と説明相手
  - ミッション：何のためにテスト結果を用いるのか
  - ステークホルダー：そのためには、誰にテスト結果を伝えればよいか
- 説明範囲
  - それで全部なのか、そのうちのどれくらいなのか
    - » 空間的全体像
    - » 時間的全体像
    - » 技術的全体像
    - » 制約・前提
- 説明単位や説明時期、説明順序など
  - テストすべき範囲がどういう構造をしているのか、どう分ければきれいなのか
- 選択の想定結果
  - ある技術上・管理上の選択をした際に得られる
    価値とリスク、納期やコストはどのくらいか、何を犠牲にするのか
- 選択理由
  - なぜその技術上・管理上の選択をしたか

説明責任を確保する技術

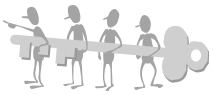
- 説明の目的と説明相手
  - テストもしくはテスト組織の位置づけを定義し直す必要がある場合もある
    - » バグ出しなのか保証なのか
    - » 開発なのかPMなのか営業なのか経営層なのかお客様なのか
- 説明範囲
  - それで全部なのか、そのうちのどれくらいなのか
    - » テスト分析モデル：テストすべき観点を全て挙げて整理し、テストする範囲を明示する
    - » テストプロダクトライン：テストスイートをどのくらい流用して何世代どう使い回すのか
    - » モデルベースドテスト・不具合モードテスト
      ：テスト技術をモデルで整理したり、勘や経験を見える化する
    - » テスト前提分析
      ：間引きに必要な「前提」を明らかにし、後で成立していることを「確定」する
- 説明単位や説明時期、説明順序など
  - テストすべき範囲がどういう構造をしているのか、
    どう分ければきれいなのか
    - » テストアーキテクチャ：テストレベルやテストタイプに分割する

説明責任を確保する技術

- 選択の想定結果
  - ある技術上・管理上の選択をした際に得られる
    価値とリスク、納期やコストはどのくらいか、何を犠牲にするのか
    - » 品質保証アーキテクチャ
      ：社会におけるビジネスの位置づけ
      ＋ビジネスにおけるシステム・製品品質の位置づけ
      ＋システム・製品品質におけるテストの質（選択結果）の位置づけ
    - » 品質モデル：テスト対象の価値はどういう特性・機能から構成されるか
    - » インテグリティレベル・リスクベースドテスト
      ：テスト対象のどこをどの程度の価値・リスクにしなくてはいけないか、
      それぞれのテストはどこの価値・リスクをどの程度説明するのか
- 選択理由
  - なぜその技術上・管理上の選択をしたか
    - » 論理的整合性・妥当性／バランス・トレードオフ／実績／勘と経験
    - » 技術駆動型定量化：用いる定量的数値に必ず技術的根拠とストーリーを入れ込む
    - » 組織モデル
      ：どういう組織構造・責任分担・インセンティブ・情報流通・方向性にすると、
      どういう想定結果に結びつき、どういうマネジメントリスクが発生するか

持続的革新を支える考え方

- 技術の徹底
  - 徹底
    - » 現場・現実・現物に対して徹底し、改善の取り組みが現場と経営とお客様を必ず幸せにするようにする
    - » 原理・原則に対して徹底し、できるはずなのに自分たちの技術ではできないことを洗い出し、知恵を絞る
  - 丸投げ感の排除
    - » 目的や理由が腹に落ちない指標や標準には従わず、自分達が納得できるよう交渉し行動する
- 個別技術力の向上
  - 既存技術改善（自工程改善）
    - » ムダ取り・ダメ取り
    - » 置き換え・自動化
    - » 分担の最適化・事前検討
    - » 重点化
  - 新規技術獲得／パラダイムシフト



持続的革新を支える考え方

- 全体技術力の向上
  - 指標的融合
    - » トレードオフと思いがちだが同時に達成できる複数指標（例：QCD）を考える
  - 技術的融合
    - » ある技術（例：テスト）を高めることで別の技術（例：設計）も高まるようなことを考える
  - 工程的・組織的融合
    - » 他工程配慮：他の組織や人のためになることを積極的に助ける仕組みを考える
  - 時間的融合
    - » 効果・効率が上がるような順序変更を考える
    - » ある時点の最適化が中長期的な最適化になるよう考える
- 向上能力の維持・向上
  - 個々の技術者の問題意識の向上
  - 現場での継続的改善
  - 個々の技術者による技術調査、他社・他分野の学習
    - » ソフトウェア組織はハード以外の他分野の学習が足りない
    - » ハードウェア生産はテスト改善の一部の参考にしかならない

講演の流れ

- 効果的なテストに必要な考え方
  - 説明責任と持続的革新
- テスト改善の技術・テストの技術
  - 実態駆動型テスト改善
  - モデル駆動型テスト改善
  - クラシックなテストの技術・モダンなテストの技術
- テストによって開発全体を改善する考え方・技術
  - テストによって開発全体を改善する考え方：他工程配慮
  - 不具合モード分析による開発・レビュー・テストの改善
  - 組み合わせテスト分析による開発の改善
  - 他工程配慮的プロセスの例：Wモデル

テスト改善の技術

- 実態駆動型テスト改善
  - 実際のテスト漏れ、工程内検出バグ、テストスイートなどから
    テスト分析・設計をリバースし、テストを改善していく方法
  - 難しいことをわざわざ勉強しなくても適用できるが、
    改善の方向性をしっかり考えることが必要になる
  - ひたすらカバレッジ率を上げて個別最適に突き進む罠に陥りやすい
- モデル駆動型テスト改善
  - テスト改善モデルを軸に、
    （多くはプロセスレベルで）あるべき姿と現状とのギャップ分析を行い、
    そのギャップを埋めるようにテストを改善していく方法
    - » TPIやTMMiが有名
  - テスト改善モデルの思想や方向性、技術、用語などを
    しっかり理解する必要がある
  - 改善を進めても現場の問題が解決されない罠に陥りやすい

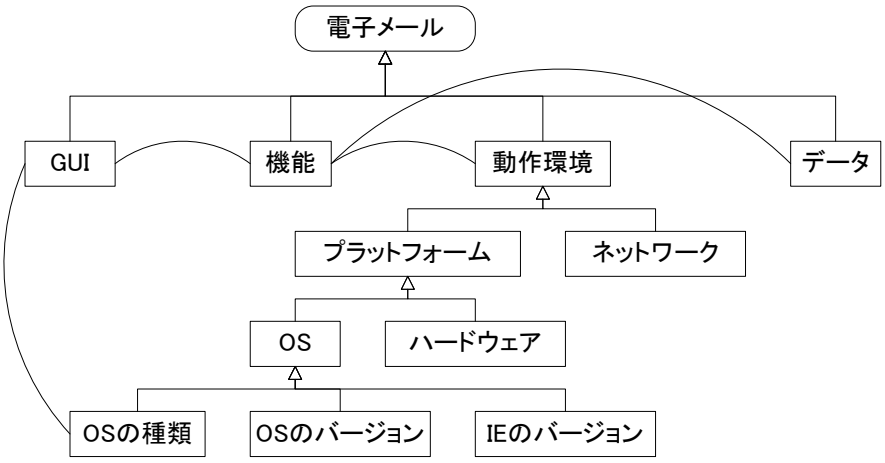
実態駆動型テスト改善の例：Reverse VSTeP

- テスト観点モデルのリバースモデリングによる改善
  - 実際のテストケースを分析して、
    どういうテスト観点でテストしようとしているかを列挙・整理し改善する
  - 実際に見逃したバグや検出されたバグ、テストケース外で検出されたバグを
    分析して、どういうテスト観点が足りなかったかを列挙・整理し改善する
- カバレッジ分析による改善
  - 見逃したバグを分析して、テスト観点不足、低カバレッジ基準、特異点、
    工数不足（不適切なリスク値）などの原因を明らかにし、
    テスト観点やカバレッジ基準、不足した前提、
    リスク値判定基準などを改善する
- テスト観点モデルのリファクタリングによる改善
  - NGTで記述されたテスト観点モデルのリンク（親子関係）に着目し、
    テスト観点が網羅的に詳細化できているかどうかをレビューし、補完する

実態駆動型テスト改善の例：Reverse VSTeP

- 不具合モード分析による改善
  - 見逃したバグや検出されたバグを分析しパターン化して、
    テスト観点を改善する
- ディレイ分析によるテストアーキテクチャの改善
  - 見逃したバグや検出されたバグを分析して、
    実際のテストレベルやテストタイプを
    リバースされたテスト観点モデルにマッピングし、
    アーキテクチャ分割をレビューし改善する
- 傾向分析によるリスク値の改善
  - 見逃したバグや検出されたバグを分析して、
    各テストケースのリスク値を改善する

NGTによるテスト観点モデルの例

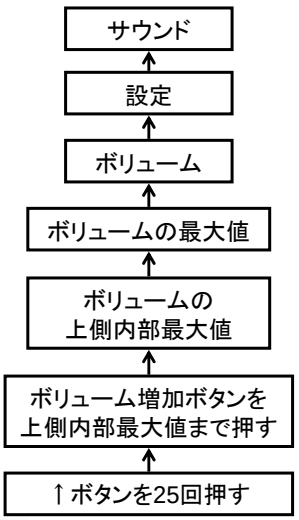


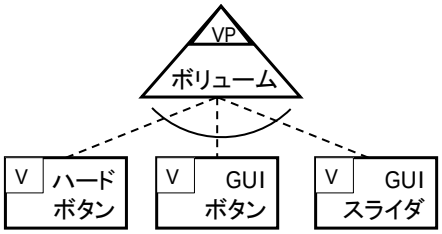
モデリングしないとテストの再利用が難しい

大項目	中項目	小項目	テスト ケース
サウンド	設定	ボリューム	↑ボタンを 25回押す

- テストケースを再利用できるだろうか？
  - 次の製品では以下の仕様変更が起こる可能性がある
    - » ボリューム調整のUIの変更
    - » ボリューム調整の手段の増加
    - » ボリューム範囲の変更
    - » デフォルトのボリューム値の変更
  - 「25回」の意味が記述されていないので、意味不明のままテストケースを再利用する羽目になる
    - » 仕様変更に追従できず、テストケースを再設計せざるを得ない
    - » 仕様変更に追従できず、新たに増えた仕様のテストが漏れてしまう
    - » テストケースが本来の狙いを果たさなくなるため、バグを検出できない
    - » 意味不明のテストケースが増えていき、保守（削除）できなくなってしまう

テスト観点モデルによるテストの再利用性の改善





- テストモデルに「テスト設計意図」や「テスト設計理由」を入れ込むことでテストケースを再利用しやすくなる
- 上流のモデルにテストのモデルを対応させておくと、そもそも再利用しやすいテストを設計しておくことができる
  - 上流からテストをモデリングすると、テスト容易性を考慮するようになり分析や設計がスッキリしていく

Software Testing

モデル駆動型テスト改善の例：TPIのKey Area

1. テスト戦略
   - A: 単一高位レベルテスト
   - B: 高位レベルテストの組み合わせ
   - C: 高位レベルテストに低位レベルテストor評価の組み合わせ
   - D: 全てのテストレベルと評価レベルの組み合わせ
2. ライフサイクルモデル
   - A: 計画、仕様化、実行
   - B: 計画、準備、仕様化、実行、完了
3. 関与の時点
   - A: テストベースの完成時点
   - B: テストベースの開始時点
   - C: 要求定義の開始時点
   - D: プロジェクトの開始時点
4. 見積もりと計画
   - A: 実証された見積もりと計画
   - B: 統計的に実証された見積もりと計画
5. テスト仕様化技法
   - A: 非公式の技法
   - B: 公式の技法
6. 静的テスト技法
   - A: テストベースのインスペクション
   - B: チェックリスト
7. 尺度
   - A: プロジェクト尺度（成果物）
   - B: プロジェクト尺度（プロセス）
   - C: システム尺度
   - D: 組織尺度（複数システム）
8. テストツール
   - A: 計画ツールと制御ツール
   - B: 実行ツールと分析ツール
   - C: テストプロセスの自動化強化

Software Testing

モデル駆動型テスト改善の例: TPIのKey Area

9. テスト環境
   - A: 管理され制御された環境
   - B: 最適環境におけるテスト
   - C: 要求に即応できる環境
10. オフィス環境
   - A: 適切かつタイムリーなオフィス環境
11. コミットメントと意欲
   - A: 予算と時間の割り当て
   - B: プロジェクト組織に統合されたテスト
   - C: テストエンジニアリング
12. テスト役割と訓練
   - A: テスト管理者とタスク
   - B: (公式の)手法的、技法的、機能的支援、管理
   - C: 公式の内部品質保証

13. 方法論の展開
   - A: プロジェクトで固有
   - B: 組織で共通
   - C: 組織で最適化、研究開発
14. コミュニケーション
   - A: 内部コミュニケーション
   - B: プロジェクト内コミュニケーション(欠陥、変更管理)
   - C: テストプロセスの品質についての組織内コミュニケーション
15. 報告
   - A: 欠陥
   - B: 進捗(テストと成果物のステータス)、アクティビティ(コスト、時間、マイルストーン)、欠陥と優先度
   - C: リスクと提言、尺度による実証
   - D: ソフトウェアプロセス改善特性への提言

モデル駆動型テスト改善の例: TPIのKey Area

16. 欠陥管理
   - A: 内部欠陥管理
   - B: 柔軟な報告機能による広範な欠陥管理
   - C: プロジェクト欠陥管理
17. テストウェア管理
   - A: 内部テストウェア管理
   - B: テストベースとテスト対象物の外部管理
   - C: 再利用可能なテストウェア
   - D: テストケースに対する追跡性システム要求
18. テストプロセス管理
   - A: 計画と実行
   - B: 計画、実行、監視、調整
   - C: 組織における監視と調整

19. 評価
   - A: 評価技法
   - B: 評価戦略
20. 低位レベルテスト
   - A: 低位レベルテストライフサイクルモデル(計画、仕様化、実行)
   - B: ホワイトボックス技法
   - C: 低位レベルテスト戦略

注意点:
- 鵜呑みにしない
- 現実・現場の問題点を見る
- 改善を回すことが重要

Tim Koomen, Martin Pol著、高野寛監訳: "テストプロセス改善 − CMM流実務モデル" を邦訳

モデル駆動型テスト改善の例: TMMi

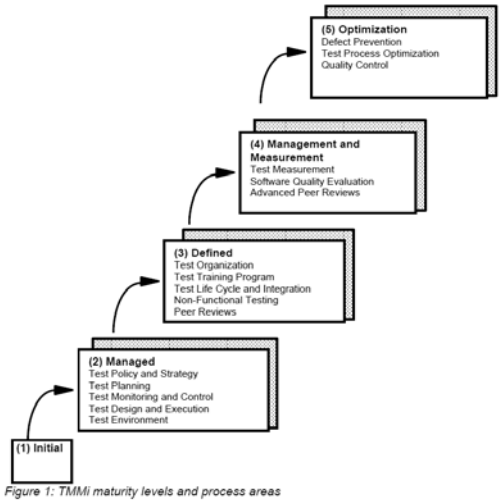


Figure 1: TMMi maturity levels and process areas

モデル駆動型テスト改善の例: TMMi

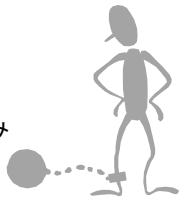
1. 実施されたレベル
2. 管理されたレベル
   - テストのポリシーと戦略
   - テストの計画
   - テストの監視と制御
   - テストの設計と実施
   - テストの環境
3. 定義されたレベル
   - テストの組織
   - テストの教育訓練
   - テストのライフサイクルと統合
   - 非機能テスト
   - ピアレビュー
4. 定量的に管理されたレベル
   - テストの測定
   - ソフトウェアの品質評価
   - 進んだピアレビュー
5. 最適化しているレベル
   - 欠陥予防
   - テストプロセスの最適化
   - 品質管理

注意点:
- 鵜呑みにしない
- 現実・現場の問題点を見る
- 改善を回すプロセスが重要

TMMi Foundation: "Test Maturity Model Integration (TMMi) Version 2.0" を邦訳

クラシック（古典的だが基本的）なテストの技術

- Vモデル
  - 単体テスト／結合テスト／機能テスト／システムテスト／受け入れテスト
  - 回帰テスト
- ホワイトボックステスト／ブラックボックステスト
  - ホワイトボックステスト＝ソースコードテスト
    ≒制御パステスト（パスカバレッジ）（＋状態遷移テスト）
  - ブラックボックステスト＝外部仕様テスト
    ≒機能網羅テスト＋境界値テスト＋デシジョンテーブルテスト
- システムテストカテゴリ
  - ボリュームテスト／ストレージテスト／高頻度テスト／ロングランテスト
  - 構成テスト／両立性テスト／データ互換性テスト
  - 操作性テスト／セキュリティテスト・・・
- プロセス無しのテスト
  - テスト実施の自動化
  - テストプロセスはテスト実施のみ、もしくはテスト設計と実施のみ
  - 信頼度成長曲線



モダンなテストの技術

- 組み合わせテスト設計
  - 直交配列表を用いたテスト、All-pair法など
- ピンポイントテスト設計
  - 不具合モード、バグパターンなど
- モデルベースドテスト・グレーボックステスト・テスト自動設計
- TLCP（テストライフサイクルプロセス）、テスト開発
  - 例）テスト分析 → テストアーキテクチャ設計 → テスト詳細設計
    → テスト実装 → テスト実施 → テスト結果報告
  - テスト計画にテスト設計は含まれない
- テスト分析モデル・テストアーキテクチャモデル
  - NGT、FV表、ゆもつよマトリックス、マインドマップなど
- テスト開発方法論
  - NGT/VSTeP、HAYST法、ゆもつよメソッド、Tiramis、CFD法など
- テストと開発の融合
  - Wモデルなど

講演の流れ

- 効果的なテストに必要な考え方
  - 説明責任と持続的革新
- テスト改善の技術・テストの技術
  - 実態駆動型テスト改善
  - モデル駆動型テスト改善
  - クラシックなテストの技術・モダンなテストの技術
- テストによって開発全体を改善する考え方・技術
  - テストによって開発全体を改善する考え方：他工程配慮
  - 不具合モード分析による開発・レビュー・テストの改善
  - 組み合わせテスト分析による開発の改善
  - 他工程配慮的プロセスの例：Wモデル

開発全体を改善する考え方：自工程完結

- 自工程完結＝自工程改善＋他工程配慮
  - 「品質を各工程で造り込み、後工程に最高品質の仕事を手渡す」ことである
    - » http://www.toyota.co.jp/jp/ir/library/annual/pdf/2008/p08_15.pdf
    - » エンジニアのポテンシャルを極限まで引き出すことに他ならない
  - 自工程完結には、自工程内だけでできる改善がある
    - » トラブルの原因の特定や改善に至るストーリーの構築は意外に難しい
  - 自工程完結には、他工程から配慮されて可能になる改善がある
    - » 分割統治や管理過多、外注過多によって視野狭窄が発生している
    - » 「きちんとやらない方がよい圧力」が働くため改善しにくい場合がある
    - » 他工程でないと測定できない／測定しにくい指標によって改善しにくい場合がある
    - » 他工程から悪さの情報をもらわないと改善しにくい場合がある
    - » どういうトラブルが発生し、どの工程でどの工程のために
どういう考慮をしておかなくてはいけないかが分からないと、未然防止できない
  - 他工程配慮をどのマネジメント階層で行うかは、
その組織のマネジメントポリシーによって異なる
    - » 欧米型組織：マネジメント層で他工程配慮を行う
    - » 日本的組織：現場で他工程配慮を行う

開発全体を改善する考え方：他工程配慮

- 他工程配慮とは
  - 品質を各工程で造り込むため、他の工程と組み合わせたり、他の工程が積極的に指標や情報を渡して改善に協力すること
    - » 上工程配慮：自工程の悪さにつながる上工程の指標や悪さを伝える
    - » 下工程配慮：下工程の悪さにつながる自工程の指標や悪さを伝える
    - » 横工程配慮：類似工程の悪さにつながる自工程の指標や悪さを伝える
  - 自分達のお客様（発注側）もパートナー（再委託先）も他工程である
  - 他工程配慮によって、開発者・開発組織の視野狭窄を乗り越える
  - 他工程配慮が成熟している組織はイノベーションを起こす可能性も高い
- 他工程配慮による融合
  - 指標的融合
    - » トレードオフと思いがちだが同時に達成できる複数指標（例：QCD）を考える
  - 技術的融合
    - » ある技術（例：テスト）を高めることで別の技術（例：設計）も高まるようなことを考える
  - 工程的・組織的融合
    - » 他工程配慮：他の組織や人のためになることを積極的に助ける仕組みを考える
  - 時間的融合
    - » 効果・効率が上がるような順序変更を考える
    - » ある時点の最適化が中長期的な最適化になるよう考える

他工程配慮の無い現場の特徴

- 悪しき受託根性
  - 何かに従っていれば仕事をしたことになると思っているし、上司もそういう指示をするし、従っていれば評価が下がらない
    - » お客様や上司の言うことに従っていれば仕事をしたことになると思っている
    - » 社内の基準、プロセスに従っていれば仕事をしたことになると思っている
    - » パートナーや部下を改善する場合には基準を厳しくすればよいと思っている
- 品質とコスト・納期のトレードオフ
  - 品質は単なる開発の結果であり、コストや納期とトレードオフだと思っている
    - » 品質を上げようとするとコストがかかり納期が延びる、と誤解してしまっている
    - » 上流はバグを作ってしまうのが当然だと思っており、レビューやテスト、監査でバグを見つけることでしか品質を上げられないでいる
    - » 本質安全よりもまず機能安全を考えてしまう
- 安直な分割統治
  - 異なる技術を専門家毎に高度化しようと思っている
    - » 設計とテスト、PMとSEPG、開発と教育
    - » Vモデル、コストダウンのための第3者検証

他工程配慮を支える考え方：TQM/SQuBOK

- 「品質」
  - 仕事の質，サービスの質，情報の質，工程の質，部門の質，人の質，システムの質，会社の質など，これら全てを含めた「質」
- 「品質保証」
  - お客様に安心して使って頂けるような製品を提供するためのすべての活動
    - » プロダクトとプロセスが、特定された要求に合致しているかどうかという十分な確信を提供する活動ではない

- 「改善」
  - 不十分でもとにかく動き出して全員が今より高いところを目指してプロセスそのものを改善しながら進める
    - » 欧米流の、プロセスを定義してその通りに実行していることを確認することではない
- 「品質第一」
  - 品質を高めることによって手戻りや作業のムダを省き，継続的な納期の短縮やコストダウンにつなげていく
    - » 納期を厳守するために必要な作業を省くのではない

他工程配慮を支える考え方：TQM/SQuBOK

- 「品質の作り込み」
  - より上流で“悪さ”の知識を子細に活用し障害を排除していく
    - » ただ単にきちんと作る、という意味ではない
- 「次工程はお客様」
  - 他の工程を助けるような改善を進め，全体最適へと向かっていく
- 「人間性尊重」
  「組織活性化」
- 「小集団活動」

- 「5ゲン主義」
  - 現場・現物・現実
  - 原理・原則

- 「全員参加」
  - 品質管理はスタッフだけではなく、現場も経営陣も一丸となって進めなければならない
  - 現場の関係者全員が納得してベクトルをあわせ，目標達成のために取り組む
  - 現場中心のため隅々まで改善が行き届くとともに，全員が自ら考えて行動する組織を構築できる

他工程配慮をテストプロセスに埋め込む

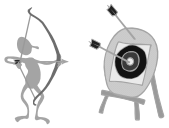
- 自工程完結には他工程配慮が不可欠である
  - 他工程配慮を業務に埋め込むような仕掛けが重要になる
- 開発プロセスに関する他工程配慮の仕掛けの例
  - Wモデル
    - » テストを開発に配慮させる仕掛け
    - » テストファーストやTDDも同様の仕掛けである
  - 不具合分析・不具合モード分析
    - » “きちんとした”不具合分析
    - » 不具合分析から根本原因を導き出しパターン化する仕掛け
  - 組み合わせテスト分析（Critical Combination Analysis）
    - » 組み合わせテストを開発に配慮させる仕掛け
  - テスト容易性設計
    - » 開発をテストに配慮させる仕掛け
  - テスト分析・設計モデルの記述と要求・設計へのフィードバック
    - » テスト観点を体系的・網羅的に把握することで、分析・設計での考慮漏れを可視化し防止する

講演の流れ

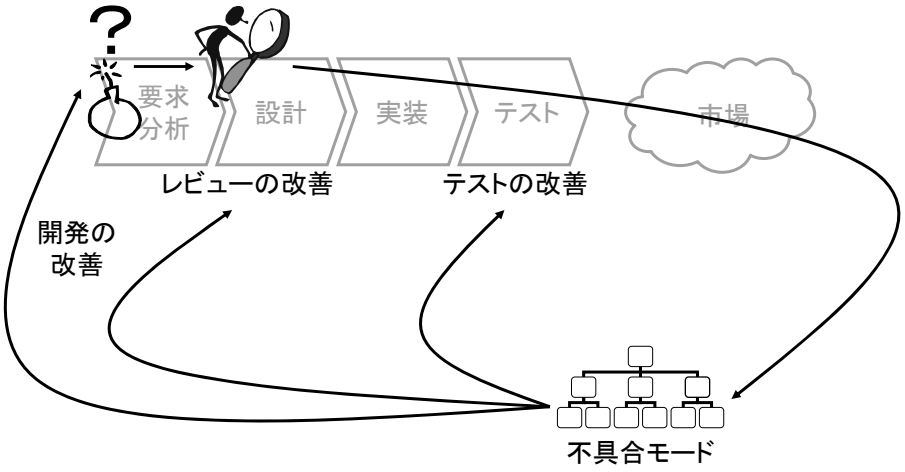
- 効果的なテストに必要な考え方
  - 説明責任と持続的革新
- テスト改善の技術・テストの技術
  - 実態駆動型テスト改善
  - モデル駆動型テスト改善
  - クラシックなテストの技術・モダンなテストの技術
- テストによって開発全体を改善する考え方・技術
  - テストによって開発全体を改善する考え方：他工程配慮
  - 不具合モード分析による開発・レビュー・テストの改善
  - 組み合わせテスト分析による開発の改善
  - 他工程配慮的プロセスの例：Wモデル

不具合モードによるピンポイントテストの設計

- 不具合が作り込まれそうな「弱点」(不具合モード)を狙って
  ピンポイントでテストを設計する方法
  - 過去の不具合を蓄積・分析し、不具合が作り込まれそうな箇所を推測することで、
    ピンポイントにテスト設計を行う
    - » よく発生する、同じようなメカニズムで作り込まれたと思われる
      不具合を集めて分析することで、そのメカニズムを推定し、
      仕様や設計、コードのパターンを導き出してテストを設計する
    - » 不具合分析のレベルが低いと、効果的な不具合モードを得ることができない
  - テスト設計だけでなく、レビューの指摘項目の設計や
    開発ガイドラインの改善にも活かすことができる
    - » テストと開発のコミュニケーションを密にするツールにもなる
    - » 自分たちの開発の弱点を得られるので、
      効果的かつ効率的にプロセス改善を行うことができる
  - テストケースあたりの不具合の検出率は向上するが、
    網羅するわけではないので品質保証には適さない
    - » 「またやっちゃった」という不具合を防ぐことができる
    - » 限られた時間で多数の不具合を検出するための手法に過ぎず、
      テスト漏れを防ぐ手法ではない
  - 不具合分析(なぜなぜ分析)の成熟度が低いと上手くいかない

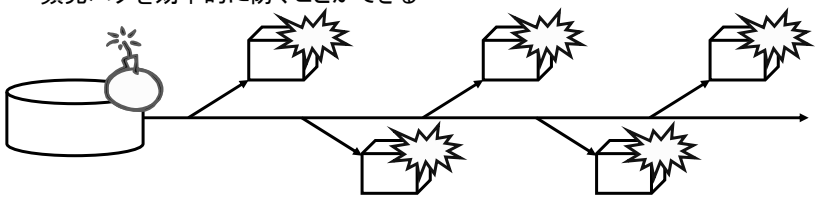


不具合モード分析による開発・レビュー・テストの改善



派生開発・SPLEにおける頻発バグを防ぐ

- 派生開発・SPLEでは特定の種類のバグが派生品をまたいで頻発することがある
  - 母体に不具合モードもしくは不具合モードの芽があるため、派生品を開発する度にバグを作り込んでしまう
    - » 不具合モード「例外」：法規対応、多仕向地、機能複合型製品
    - » 母体の設計にそもそも難がある
    - » 仕様書群の構造そのものに不具合モードがあったりもする
  - 母体の不具合モードに着目してテストを設計することで頻発バグを効率的に防ぐことができる



不具合分析のアンチパターンと効果の薄い対策

- Vaporization（その場限りの簡単なミスとして片付けてしまう）
  - コーディングミス：スキルを向上させるよう教育を行う
  - 考慮不足：しっかり考慮したかどうかレビュー時間を増やし確認する
  - うっかりミス：うっかりしていないかどうかレビュー時間を増やし確認する
- Exhaustion（不完全な実施や単なる不足を原因としてしまう）
  - しっかりやらない：しっかりやったかどうかレビュー時間を増やし確認する
  - スキル不足：スキルを向上させるよう教育を行う
  - 経験不足：経験を補うためにスキルを向上させるよう教育を行う
- Escalation（上位層に責任を転嫁してしまう）
  - マネジメントの責任：トップを含めた品質文化を醸成する
- Imposition（外部組織に責任を転嫁してしまう）
  - パートナー企業の責任：パートナー企業を含めた品質文化を醸成する
- Culturalization（文化的問題に帰着してしまう）
  - 品質意識／品質文化の欠如：全社的な品質文化を醸成する

講演の流れ

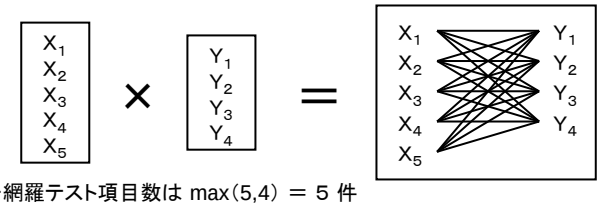
- 効果的なテストに必要な考え方
  - 説明責任と持続的革新
- テスト改善の技術・テストの技術
  - 実態駆動型テスト改善
  - モデル駆動型テスト改善
  - クラシックなテストの技術・モダンなテストの技術
- テストによって開発全体を改善する考え方・技術
  - テストによって開発全体を改善する考え方：他工程配慮
  - 不具合モード分析による開発・レビュー・テストの改善
  - 組み合わせテスト分析による開発の改善
  - 他工程配慮的プロセスの例：Wモデル

ブラックボックスによる組み合わせテストの限界

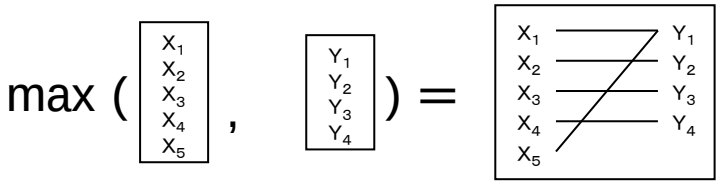
- 全ての組み合わせを網羅すると、組み合わせバグは必ず検出できるが、テスト項目数が爆発する
  - 単一（1段）で網羅すると、テスト項目数は少なく済むが、組み合わせバグを検出できるとは限らない
    - » テストが無限に思えると、思考停止を招いてしまう
    - » 要求分析で、何段の組み合わせまで保証したいかをしっかり考えておく
- 直交配列表を使うと、2段に関する組み合わせバグを少ないテスト項目数で必ず検出できる
  - 3段以上の組み合わせバグについては検出できるとは限らない
    - » テストしたい組み合わせが分かっていれば割り付けられる可能性が高い
  - 複数の水準は、複数の因子を割り付けるか、多水準直交表を用いる
  - 禁則については補完を行うか、All pair法などを用いる
    - » きちんと対応するにはHAYST法を用いる
- あくまでブラックボックスの保証型テストに有効
  - 組み合わせバグの原因である内部の依存関係には着目していない
  - グレーボックスで内部構造を考慮した方が効率的になる場合が多い

グレーボックステストによる依存関係の間引き

- 組み合わせ網羅テストと単一網羅テスト
  - 単機能テスト項目5件の機能Xと、単機能テスト項目4件の機能Yがある
    - » 組み合わせテスト項目数は 5×4 ＝ 20 件

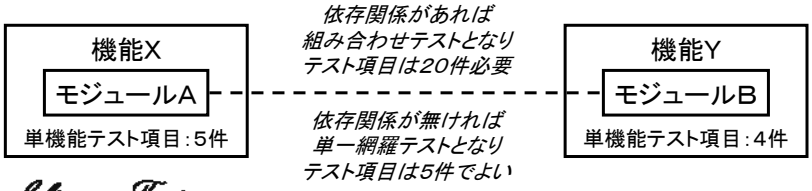


- » 単一網羅テスト項目数は max(5,4) ＝ 5 件



グレーボックステストによる依存関係の間引き

- グレーボックスで依存関係を分析しながらテストを設計すると、機能Xと機能Yの組み合わせテストを構造的にレビューし判断して間引くことができる
  - 組み合わせが必要なのは、機能Xの処理に応じて、機能Yの処理が変わるから
    - » 機能を構成するモジュール間に依存関係があるから
  - 構造的にレビューして、依存関係が無かったり、影響伝播が抑えられていることが保証できれば、単一網羅テストで十分となる
    - » ブラックボックステストに少しホワイトボックス的観点が入るのでグレーボックステスト
    - » 実務的には完全に間引くわけではなく、優先度を落とすことになる



組み合わせテスト分析による開発の改善

- ソフトウェア設計とテスト設計を協調・並行して行うことで
  組み合わせが激増する「ホットスポット」を特定し
  設計そのものを改善することでテスト工数を抑える
  - 不必要な依存関係を抑えることで、
    テスト工数の爆発やTestabilityの低下につながる設計を防止する
    - » ホットスポットは設計の複雑さが集中する部分であり、
      組み合わせによるテスト工数が激増する部分である
    - » 製品設計時にモジュール性を考慮した方がよいことは皆知っているが、
      モジュール性を減らした時にどうなるかを定量的に把握したことはない
  - ホットスポットを改善しないと、派生開発の度に組み合わせテストを行う羽目になる
- 製品設計時にテスト工数を考慮してトレードオフを行う
  - Wモデルによって上流の設計時に
    テスト工程も考慮しながらトータルでトレードオフを行う
    - » 通常の製品設計時のトレードオフでは、テストのことなど頭にない
    - » 製品設計時には「楽だけどちょっと複雑」、テスト時には「工数爆発」
    - » 製品設計の工数がほんの少し増えても、テストで激減すればトータルで得

講演の流れ

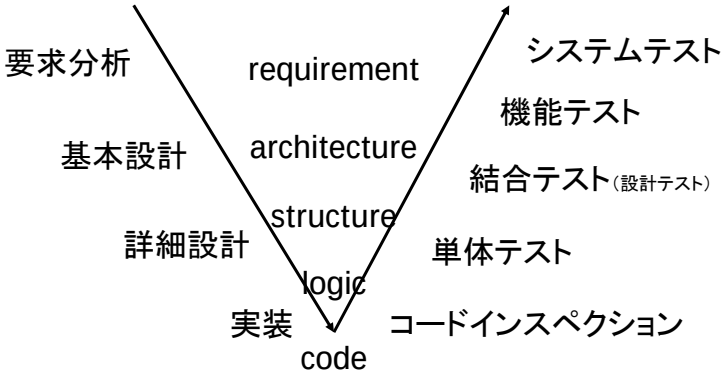
- 効果的なテストに必要な考え方
  - 説明責任と持続的革新
- テスト改善の技術・テストの技術
  - 実態駆動型テスト改善
  - モデル駆動型テスト改善
  - クラシックなテストの技術・モダンなテストの技術
- テストによって開発全体を改善する考え方・技術
  - テストによって開発全体を改善する考え方：他工程配慮
  - 不具合モード分析による開発・レビュー・テストの改善
  - 組み合わせテスト分析による開発の改善
  - 他工程配慮的プロセスの例：Wモデル

上流と協調し、上流を改善するようにテストする

- 成熟度の高い組織は、テストと上流が協調している
  - 取りあえずテストの人員や工数を増やそうとするのではなく、自分たちのテストのムダやモレをしっかり分析しようとする
  - その結果、わざわざ下流まで待たなくても検出できる不具合がかなりあることが分かる
  - ビルドして初めてテストを考えるのではなく、上流からテスト設計を行っていくことでバグを防ぐ
    - » 上流からのテストは、単なる要求の裏返しではない
  - 上流からテストの品質を上げようとすると、テストは無限にならず、出荷時の品質リスクも考えられるようになる
  - 「上流からの品質の作り込み」の本質は、ノウハウの循環の活性化である

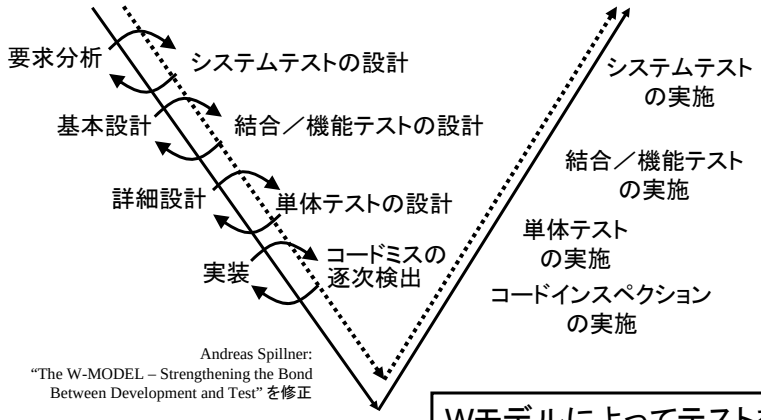
- 「見通しのよい」開発ができるようになる
  - 品質の高いソフトを開発している組織は、自分たちがやっていることの意味合いや位置づけ、全体像を把握し、どうなるかを概ね推測できる
  - 開発やマネジメント/プロセスだけでなく、レビューやテストなど評価系の意味合いや位置づけ、全体像をしっかり把握する必要がある

分割統治的プロセスの例：Vモデル



Vモデルでは上流とテストが分断され、プロダクト的な悪さの知識がフィードバックされない

他工程配慮的プロセスの例：Wモデル

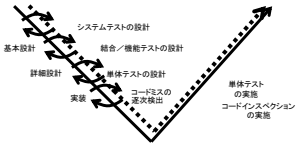


Andreas Spillner:
"The W-MODEL – Strengthening the Bond Between Development and Test" を修正

Wモデルによってテストを前倒しし、
プロダクト的な悪さの知識を
フィードバックする

Wモデルを支えるテスト技術

- 基本的なテスト技術
  - 体系的なテスト技法の活用、期待結果の導出
  - 網羅テストの段階的詳細化
  - テストレベルの設計もしくは解釈
- 進んだテスト技術
  - 組み合わせテスト設計、Critical Combination Analysis、禁則分析
  - テスト観点の抽出・整理・体系化
  - 不具合モード分析・後工程不具合リスク分析
  - リスクベースドテスト
  - グレーボックステスト
  - 探索型テスト
- 開発全般に関わるテスト技術
  - テスト容易性設計
  - 出荷判定基準の明確化



テストの改善をきっかけにして、
そもそもバグを作り込みにくい
開発プロセスに改善していく

Wモデルの効果：レベル1

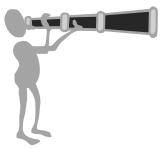
- 作業順序の入れ替えによりムダを低減できる
  - テスト設計をきちんと行うことで検出できる不具合を、
    下流まで遅らせることなく上流で検出することにより、
    不具合の伝播や増殖、不要な設計変更などのムダを省くことができる
    - » テスト設計をきちんと遂行できる能力が必要である
    - » テスト設計の技術やノウハウをレビューに活かすことによる
      レビュー強化活動だと考えることもできる
  - アドホック型やコピー＆ペースト＆モディファイ型のテスト設計では、
    テスト設計時に不具合が検出できることが少ないため、
    テスト設計を上流にシフトしても効果は少ない
    - » 「テストレベルの設計」もしくは「テストレベルの解釈」も必要である
    - » Wモデル導入の前提にVモデルの定着が必要と思われがちなのは、
      テスト（項目）設計とテストレベル設計が必要だからである

Wモデルの効果：レベル2

- 設計をきれいにし、テストに深みを与えることができる
  - 組み合わせテスト設計を上流で行うことで、
    仕様や設計、コードの改善が可能になる
    - » 結合度の低下・凝集度の向上や依存関係の把握により、
      組み合わせ不具合（副作用）が低下する
    - » 不必要な依存関係や禁則を持った仕様を減らすことができる
  - テスト容易性を検討でき、作り込むことができる
    - » テスト設計しやすいような仕様・設計・コードは、
      可読性が高く複雑度が低いうえに、ふるまいを予測しやすい
  - グレーボックステストによって、
    組み合わせテスト項目数を低減することができる
    - » ブラックボックステストでは把握できない機能間の独立性を
      内部構造の検討によって把握し、
      組み合わせる必要のないテスト項目を間引くことができる

Wモデルの効果：レベル3

- 要求分析・設計・実装工程で、「由来」と「行く末」を常に検討し、両者を一致させることができる
  - 網羅型テストの段階的詳細化を行うことにより、由来（要求や制約、根拠）をイメージバックしながら開発できるようになる
    - » どのような要求群から成り立つのか、どのような制約がありうるのか、を常に考えながら作るようになる
    - » 当該工程の作業の根拠を常に考えるようになる
  - 出荷判定基準の明確化や期待結果の導出を行うことにより、行く末（出荷基準やトリアージ順位、ふるまい）をイメージフォワードしながら開発できるようになる
    - » どのような出荷基準になっているのか、どのようなトリアージ順位になっているのか、を常に考えながら作るようになる
    - » 当該作業成果物のふるまいを常に考えるようになる
  - 由来と行く末が乖離する開発に歯止めをかけることができるようになる



Wモデルの効果：レベル4

- 個々の工程の能力を常に進化させ続けることができる
  - 下流で不具合モード分析を行って当該工程の弱点をフィードバックしてもらうことにより、頻発する不具合を防止した開発ができるようになる
    - » 下流での不具合モード分析により、自分たちが作り込みやすい不具合のパターンやドメイン特有の作り込みやすい不具合のパターンを把握し、最初から気をつけて開発することができる
    - » 「上流からの品質の作り込み」とは、必要なノウハウを必要な時期に必要なだけ駆使することであり、最初から矛盾の無いものを作って矛盾の無いように自動生成することではない
  - 上流で後工程不具合リスク分析を行って、上流で気付いていた「心配事」を当該工程にフィードフォワードしてもらうことにより、分かっていたのに起きてしまった検討漏れを減らすことができるようになる
    - » 上流での後工程不具合リスク分析により、工程が進んだ後に不具合になるリスキーな仕様・設計・コードの情報をもらうことができ、リスキーな部分に常に気をつけて開発することができる

Wモデルの効果:レベル5

- 個々の工程で系統的に探索を行うことができる
  - 当該工程で見つかった不具合の原因が当該工程にある場合、探索型テストを行って、似たような原因を探索的に推測し、同様の原因で発生する他の不具合を検出できる
    - » 不具合モードなど不具合の原因をパターン化しておけると、系統的に工程内の探索できるようになる
  - 当該工程で見つかった不具合の原因が上流にある場合、探索的に上流までさかのぼって原因を特定(・修正)し、同様の原因で発生する他の不具合を検出できる
    - » 上流までさかのぼることにより、同様の原因で発生する他モジュールの不具合を検出することができる
  - 当該工程でリスクベースドテストを行い、想定した不具合の作り込みやすさや致命度に応じて、信頼性設計(フェイルセーフ、フェイルソフト、フェイルソフトリー)を必要なだけ行うことができる
    - » ハザード解析で最も難しい、ソフトウェア不具合というハザードの解析ができる

Wモデルの目指すべき姿

- 「必要な品質の保証を必要な時に必要なだけ行う」開発プロセス
  - テストすべき観点や開発の弱点が工程毎に分かるため、プロダクトの要求品質に合わせた品質保証プロセスの設計が可能になる
    - » ソフトウェア版QC工程表
  - 考慮不足のテスト観点や不具合モードを把握することができるため、プロダクトの作られ方に合わせた品質保証プロセスの進化が可能になる
    - » 探索型品質保証プロセス
  - 現在のプロセスに傾きすぎたソフトウェア品質保証プロセスを、プロダクト側に揺り戻すことができる
- 全ての工程でノウハウを生みだし共有し駆使する開発プロセス
  - ノウハウの抽出・蓄積・共有・駆使・改訂がプロセスに埋め込まれるようになる
  - 開発チームが開発進行中に自分達でプロセス改善をするようになる
  - 幅と遠さの両方で、開発者の見通しをよくすることができるようになる
    - » 開発者は一般的に分割統治・詳細隠蔽の原則で発想することが多いので、テスト設計者と一緒に設計検討をすると見通しがよくなることが期待できる
  - ソフトウェア開発者の多能工化が進む

Wモデルの導入へのアプローチ

- プロセス改革アプローチ
  - スタッフ部門や上級管理職が主導でWモデルのプロセスマニュアルを作り、各部門に展開して導入する
    - » 私見だが、多分うまくいかない
- レビュー強化アプローチ
  - レビューを強化しようという運動として導入する
    - » 私見だが、レビューの指摘項目の設計の強化にはつながらず、結局プロセスをいじって終わってしまうと思う
- 工程順序変更アプローチ
  - テスト設計を改善し、テスト設計時に不具合を見つけてもらうことで、下流のテスト設計まで不具合検出を遅らせるムダを意識してもらい、工程順序を変更する圧力を自発的に発生させる
    - » 定着しやすいが時間がかかる
- 設計ノウハウ蓄積・活用アプローチ
  - 設計者同士のノウハウ共有という運動として不具合モードを蓄積し、それを全工程で活かすようにする
    - » すぐに実行できるが、開発者が忙しいと停滞しやすい

Wモデルの導入へのアプローチ

- 技術の発展は癒着→剥離→融合の段階を経ることを意識し、剥離させずに融合させようとしても定着しないことを理解する
  - 自分達は何を分かっており何を分かっていないのかをきちんと理解したうえで、適切な技術やノウハウを駆使できるようにする
    - » 癒着：個々の技術を意識せず、したがって技術成熟度が上がらないまま、様々な技術を暗黙的かつKKDで組み合わせて利用している状態
    - » 剥離：個々の技術を意識して分割統治することで技術成熟度を上げる状態
    - » 融合：複数の技術を意識的に組み合わせることでムダを省き全体最適を実現している状態
- 自分たちがきちんと仕事をすれば責任は無い、というカルチャーやコンセプトで導入すると、Wモデルは順序入れ替え以上の効果を及ぼさない
  - Wモデルは（静的な）プロセスモデルというよりも、プロセス進化のモデルであることを理解する
    - » 次工程はお客様、TOC、自工程完結などのコンセプトの理解が重要となる

講演の流れ

- 効果的なテストに必要な考え方
  - 説明責任と持続的革新
- テスト改善の技術・テストの技術
  - 実態駆動型テスト改善
  - モデル駆動型テスト改善
  - クラシックなテストの技術・モダンなテストの技術
- テストによって開発全体を改善する考え方・技術
  - テストによって開発全体を改善する考え方：他工程配慮
  - 不具合モード分析による開発・レビュー・テストの改善
  - 組み合わせテスト分析による開発の改善
  - 他工程配慮的プロセスの例：Wモデル

テストの改善を進めるにはどうすればよいか

- そもそも改善しようとしていない
  - 現場に行って問題を目の当たりにし、改善したくなる衝動を持つ
- 何をすればいいか分からない
  - 自分達が何をできていて何をできていないか、を把握する
    - » 今やっているテスト、今出しているバグ、今行っている開発を虚心坦懐に見直すことが必要
  - 自分達の背丈に合った改善を身の回りからできる範囲で始める
    - » 小さなフィードバックループを頻繁に回すことで他工程配慮クセを付ける
    - » テスト改善モデルにいきなり傾倒しない
- 何かしているが、テスト工程に効果が出ていない
  - 説明責任と持続的革新という2つの考え方を身につける
- テスト工程に効果は出ているが、開発全体に効果は出ていない
  - 他工程配慮によって、指標的・技術的・工程的・組織的・時間的に融合した開発組織を目指し改善を進めていく
  - 常に「現場がどう良くなるか」という視点で進めていく
  - 最初からバグを作り込まない開発を目指す

Boris Beizer のテスト「道」

フェーズ0 ー テストはデバッグの一部である

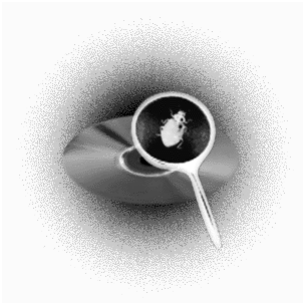
フェーズ1 ー テストの目的は、
ソフトウェアが動くことを示すことである

フェーズ2 ー テストの目的は、ソフトウェアが動かないことを
示すことである

フェーズ3 ー テストの目的は、何かを証明することではなく、
プログラムが動かないことによって発生する
危険性をある許容範囲までに減らすことである

フェーズ4 ー テストは行動ではなく、テストをしないで品質の
高いソフトウェアを作るための精神的訓練である

ご清聴ありがとうございました



電気通信大学 西 康晴
http://blues.se.uec.ac.jp/
nishi@se.uec.ac.jp