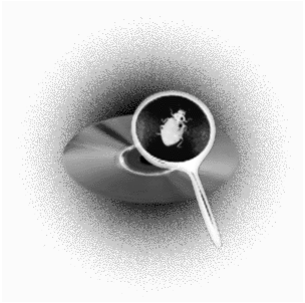
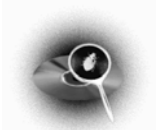
ソフトウェア品質保証技術の動向



ソフトウェアテストシンポジウム四国2008
2008/6/27（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

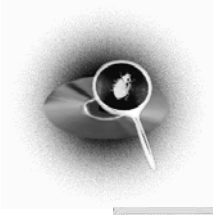
- 身分
  - ソフトウェア工学の研究者
    - » 電気通信大学 電気通信学部 システム工学科
    - » ちょっと「生臭い」研究／ソフトウェアテストやプロセス改善など
  - 先日までソフトウェアのよろず品質コンサルタント
- 専門分野
  - ソフトウェアテスト／プロジェクトマネジメント／QA／ソフトウェア品質学／TQM全般／教育
- 共訳書
  - 実践ソフトウェア・エンジニアリング／日科技連出版
  - 基本から学ぶソフトウェアテスト／日経BP
  - ソフトウェアテスト293の鉄則／日経BP
  - 基本から学ぶテストプロセス管理／日経BP
- もろもろ
  - TEF: テスト技術者交流会
  - 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 副委員長
  - 情報処理学会 アクレディテーション委員会 ソフトウェアエンジニアリング分科会
  - 経済産業省 組込みソフトウェア開発力強化推進委員会



SESSAME

SQiP
Software Quality Profession

TEF: 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1400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
  - "JaSST:ソフトウェアテストシンポジウム"も開催している
    - » 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・札幌・博多でも開催／会場は満席
  - 「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

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/

SESSAME

- 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します。
- 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
  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

SQiP Software Quality Profession

講演の流れ

- ソフトウェアの品質や信頼性は低い
- 品質とは
- SQuBOKに垣間見える日本的品質管理の特徴
- ソフトウェア品質保証技術の発展：癒着期、剥離期
- テストに着目した融合期の技術
- Wモデルの導入／テスト容易性設計
- 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
- 不具合モードによるテスト・レビュー・開発の改善
- テスト「道」

日本のソフトウェア開発は品質が良い？

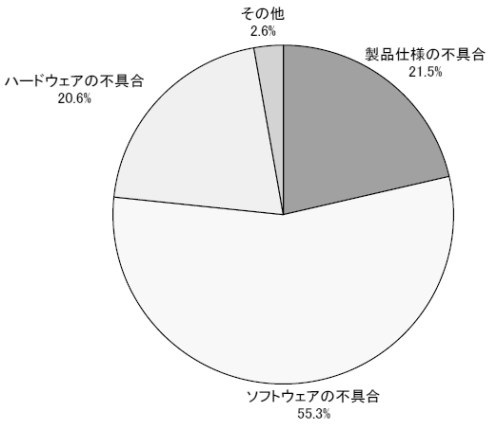
Performance data

	India	Japan	US	Europe & other	Total
Number of projects	24	27	31	22	104
Median output[1]	209	469	270	436	374
Median defect rate[2]	.263	.020	.400	.225	.150

1. No. of new lines of code / (avg. no. of staff × no. of programmer-months).
2. No. of defects reported by customers in 12 months after implementation / total source LOC. We adjusted this ratio for projects with less than 12 months of data.

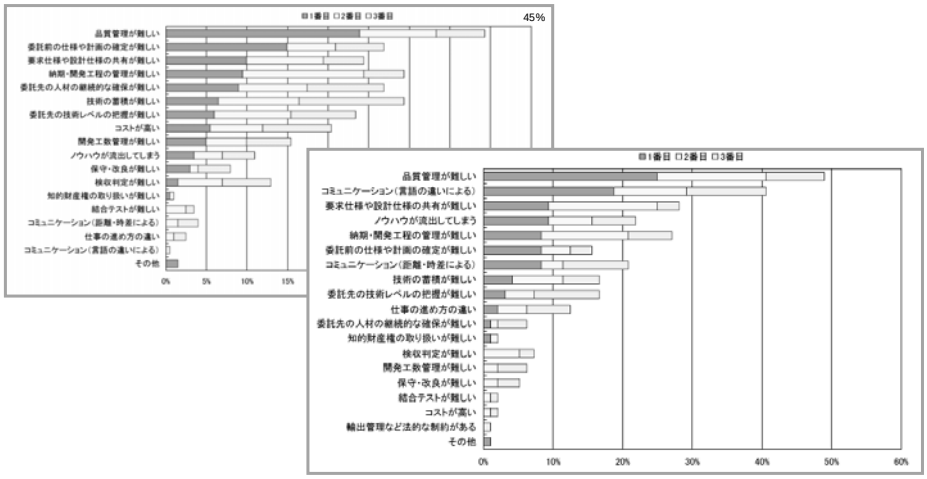
Michael Cusumano, et.al:
"Software Development Worldwide:
The State of the Practice"
IEEE Software, Vol.20.Issue 6, pp.28-34(2003)

製品出荷後の設計品質問題の主な原因



「2006年版組込みソフトウェア産業実態調査報告書:
経営者・事業責任者向け調査」より

外部委託の課題



「2006年版組込みソフトウェア産業実態調査報告書:
経営者・事業責任者向け調査」より

ソフトウェアの信頼性は低いと言わざるを得ない

- エンタープライズ系システム
  - メガバンクの情報システム統合に伴う品質事故は記憶に新しい
  - 東京証券取引所の情報システムは度重なる品質事故
    - » 売買システムの処理増強に伴う品質事故により全銘柄で午前中の取引停止
    - » ジェイコム株誤発注事件では品質事故により400億円超の損害賠償請求訴訟
  - 羽田空港の航空管制システムの品質事故は30万人以上が足止め
  - 成功するプロジェクトはわずか26.7%(日経コンピュータ)
- 組込み系システム
  - 国内の携帯電話:65万台が回収・無償交換、131億円にのぼる損害
  - ドイツの高級車のブレーキシステム:68万台がリコール
  - 鉄道:ATCの誤作動により1ヶ月に50回以上の速度超過、19件の緊急停止措置
  - ダム:ゲートが開き、総量約2万立米の貯水が流出
- 安全性が重視されるシステムが増えていくが. . .
  - 自動車はX-by-wireになり、ITSにより道路状況を運転に反映させるようになる
  - 軌道式公共交通機関は無人になっていくかもしれない
  - 人間を殺傷する動力を持つ家庭用ロボットが、事前訓練を受けずに動かされる

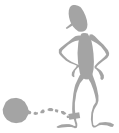
組込みソフトウェアの受難

- 機能要求の増大
  - 制御の複雑化
  - ソフトウェアによる機能の増加
    - » 携帯電話≒デジカメ≒コピー機？
- CPU/ECU 数の増加
  - 比例してソフトウェアの数も増加
- ネットワークによる協調動作
  - 自動車の重量の1割は通信ケーブル
  - TCP/IPの搭載によるセキュリティの考慮
- プラットフォームが多様
  - チップもOSも多種多様で当分統一されない
- 階層の増加
  - ファームウェアからOS＋ミドル＋アプリへ
- PCアプリの発生
  - PC上で組込み機器を管理／ソリューションの提供も

ついこないだまで
リレーに毛の生えた
ものだったのに．．．

⇩

気が付くと
数百万行にのぼる



組込みソフトウェアの受難

- 開発期間の短縮
  - 携帯機器は半年サイクル
- 信頼性要求の劇的な向上
  - 財布や鍵になる携帯電話
- 低品質な開発資産の流用
  - ドキュメントの無い開発資産
  - グローバル変数を多用した設計
- しかし開発体制は旧来どおり
  - 体制が整う前に複雑化・大規模化し
    プロジェクトがデスマーチ化してしまう
  - ソフト部隊とハード部隊の連携に乏しい
- 品質事故・納期遅延・赤字プロジェクトが
  多発している
  - 携帯、家電、OA、車載、ME、ダム．．．

ついこないだまで
たった1人で
作ってたのに．．．

⇩

気が付くと
数十人のプロジェクトに

講演の流れ

- ソフトウェアの品質や信頼性は低い
- 品質とは
- SQuBOKに垣間見える日本的品質管理の特徴
- ソフトウェア品質保証技術の発展：癒着期、剥離期
- テストに着目した融合期の技術
- Wモデルの導入／テスト容易性設計
- 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
- 不具合モードによるテスト・レビュー・開発の改善
- テスト「道」

代表的なソフトウェア関連の知識体系

- 全般的な知識体系
  - PMBOK(PMI)
    - » プロジェクトマネジメント全般
  - SWEBOK(IEEE/ACM)
    - » ソフトウェアエンジニアリング全般
  - SQuBOK(JUSE+JSQC)
    - » ソフトウェア品質全般
      ・日本的品質管理
- 試験・教育向け知識体系
  - 大学向けカリキュラム標準
    - » CCSE2004(ACM)
    - » J07-SE/IS/CE(IPSJ)
  - 資格向けシラバス
    - » CSQEシラバス(ASQ・ソフトウェア品質)
    - » 基礎テスト資格・応用テスト資格向け
      シラバス・用語集（ISTQB/JSTQB・テスト）
    - » CSTP/CTMシラバス（IIST・テスト）
- 特定技術向け知識体系
  - EABoK（MITRE）
  - CMBoK/CM-Wiki（CMC Media）
  - PSPBoK（CMU-SEI）
  - FMBoK?（NII？）
- 辞典・用語集（ABC順）
  - ソフトウェア工学大辞典
    - » 1994年出版：“Encyclopedia of Software Engineering”の和訳
    - » 解説が充実している
  - ソフトウェア品質管理ガイドブック
    - » 1990年出版：日本で作成
    - » 解説が充実している
  - IEEE std. 610.12-1990
    - » 定義のみ

ソフトウェア品質に対する位置づけの違い

- PMBOK
  - 品質とは、「本来備わっている特性がまとまって、要求事項を満たす度合い」である（ASQ）
    - » 8章冒頭
  - プロジェクト品質マネジメントプロセスには、品質計画、品質保証、品質管理がある。品質管理 ― 特定のプロジェクト結果が適切な品質規格に適合しているかどうかを判断するために、結果を監視し、不満足なパフォーマンスの原因を除去するための方法を特定する
    - » 8章冒頭
- SWEBOK
  - よく読むと、品質の定義を巧妙に避けている
    - » ユーザ要求に対する適合性（Crosby）、利用に対する適合性において優れたレベルを達成する（Humphrey）、顧客満足度（IBM、MB賞）、本質的特性の集成体が要求を満たす度合い（ISO9001-00）
    - » 11章序説
  - SQMプロセスは、与えられたプロジェクトにおいて、より良いソフトウェア品質を確証することを助ける。SQMプロセスは、その副産物として、全ソフトウェアエンジニアリングプロセスの品質表示など、マネジメントに対する一般的な情報を提示する
    - » 11.2/ソフトウェア品質マネジメントプロセス

ソフトウェア品質に対する位置づけの違い

- SQuBOK
  - 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」
    - » 1.1.1.7/品質の定義：石川馨
  - 組織を長期的・安定的に存続させるには，組織の活動の主たるアウトプットである製品・サービスを顧客に提供し，それによって対価を得て，そこから得られる利益を再投資して価値提供の再生産サイクルを維持することが必須である。そのためには，組織が提供する製品・サービスが長期的に幅広い顧客に満足を与え続けなければならない。これを実現するための武器が「品質のマネジメント」である。
    - » 1.2/品質のマネジメント

「品質」は技術力そのものであり、
組織の持続的な強みの源泉である

日本的品質管理の考え方

- 品質とは
  - 単なる開発の結果ではなく、組織が目指すべき軸である
    - » 欠陥、価値（円）、満足度、喜び、導き
    - » ブランドカ、ヒット商品を海、感受性や予測が必要である
  - 組織における技術・マネジメント・ひとの持続的成長の源泉である
    - » 顧客、経営、現場の3者を同時によくする
  - 現場が楽になるように品質向上活動をする
    - » 中長期的な成長のプランを持ち、成果を現場に見えるように出すことで現場の信頼を得て継続的にカイゼンする
    - » 自組織の弱点を把握して、適切な解決策を選びカスタマイズしたり、本質的な解決策を設計する
    - » 絶えざる目的指向が備わるようになる
  - ロバストな経営指標である
    - » 大きな失敗は無い

品質を上げようとすると
コストは下がり納期は短くなる

日本的品質管理ではないもの

- 品質とは
  - 不具合の数という開発の結果である
    - » 測定すれば分かる
  - コストや納期などと同列の、もしくは後回しの、単なる指標である
  - 品質向上活動によって現場が楽にならない
    - » 行き当たりばったりの活動をしていたり、流行を追っている
    - » 自組織の弱点を把握せず解決策に目を奪われている
    - » 目的を見失い現場の信頼も失う
  - 経営指標ではない
- 品質とコスト、納期の関係
  - 品質を上げるとコストが上がり納期は長くなる
    - » 品質は検査や監査で確保する
  - 最適品質を狙い、過剰品質は避けるべきである

講演の流れ

- ソフトウェアの品質や信頼性は低い
- 品質とは
- SQuBOKに垣間見える日本的品質管理の特徴
- ソフトウェア品質保証技術の発展：癒着期、剥離期
- テストに着目した融合期の技術
- Wモデルの導入／テスト容易性設計
- 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
- 不具合モードによるテスト・レビュー・開発の改善
- テスト「道」

SQuBOKに垣間見える日本的品質管理の特徴

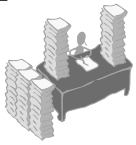
- 「品質」
  - 仕事の質，サービスの質，
    情報の質，工程の質，部門の質，
    人の質，システムの質，
    会社の質など，
    これら全てを含めた「質」
- 「品質保証」
  - お客様に安心して使って頂ける
    ような製品を提供するための
    すべての活動
    - » プロダクトとプロセスが、
      特定された要求に合致している
      かどうかという十分な確信を
      提供する活動ではない
- 「改善」
  - 不十分でもとにかく動き出して
    全員が今より高いところを
    目指してプロセスそのものを
    改善しながら進める
    - » 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

SQuBOKに垣間見える日本的品質管理の特徴

- 「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

コストや納期だけをカイゼンしようとすると失敗する

- コストダウン優先によるデスマーチ化
  - 安物技術の導入→品質の低下→手戻り作業の発生→残り予算減少による、さらなる安物技術の導入プレッシャーの増大
    - » オフショア、アウトソーシング単価低減、安物コンポーネント購入など
  - 結局、赤字プロジェクトになってしまう
- 納期達成優先によるデスマーチ化
  - 工数短縮による必要作業の省略→品質の低下→手戻り作業の発生→残り工期減少による、さらなる納期プレッシャーの増大
    - » 設計軽視、レビュー削減、単体テスト削減、ドキュメント削減など
  - 結局、納期遅延になってしまう



コストを下げ納期を短くするには
品質を上げて手戻りを減らすことが
王道であり早道である

講演の流れ

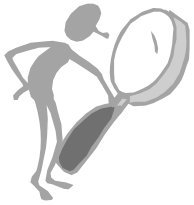
- ソフトウェアの品質や信頼性は低い
- 品質とは
- SQuBOKに垣間見える日本的品質管理の特徴
- ソフトウェア品質保証技術の発展：癒着期、剥離期
- テストに着目した融合期の技術
- Wモデルの導入／テスト容易性設計
- テスト分析・設計モデルの記述と
  要求・設計へのフィードバック
- 不具合モードによるテスト・レビュー・開発の改善
- テスト「道」

ソフトウェア品質保証技術の発展

- ソフトウェア品質保証技術とは
  - 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、といったたぐい活動ではない
- ハードウェアの品質保証技術（品質管理：QC）の発展
  - SQC: Statistical Quality Control / 統計的品質管理
    - » 工場での大量生産のバラツキを減らすための活動群（統計が必須ではない）
  - TQC/TQM: Total Quality Control (Management) / 全社的品質管理
    - » 工場での品質管理の考え方を設計や営業（経営）に適用するフレームワーク
  - 生産の品質管理の技法と設計の品質管理の技法とは、
    考え方は同じだが技法はかなり異なる
    - » 生産の品質管理→設計の品質管理→企画の品質管理という発展もしている
- 技術の発展の順序：癒着→剥離→融合
  - 癒着期
    - » 開発者だけがKKDで品質を確保している
      - 全てのノウハウが頭の中にある
      - 品質向上に関するコミュニケーションが難しい
    - » 規模が大きくなるとニッチもサッチもいかなくなる
    - » ノウハウの伝達が難しいので、人数も対象も増やせないし
      伝承できる世代も延ばせない

ソフトウェア品質保証技術の発展：剥離期

- Ｖ＆Ｖ技術の剥離：何が癒着していたか
  - バグを作り込みやすい仕様・設計・コードのパターン
    - » バグを作り込みやすいコードのパターン：コーディング作法／MISRA-C
  - 何が達成されれば完成したことになるのか、を考えること
    - » ソフトウェア品質特性・品質モデル／ISO/IEC9126
    - » ソフトウェアメトリクス
  - プロジェクトの失敗のパターン
    - » プロジェクトリスクと兆候
- Ｖ＆Ｖ技術の剥離：剥離して得られた技術
  - 不具合分析
  - リスク分析
  - レビュー・インスペクション・テスト
  - 測定・メトリクス／エンピリカル、GQM（ゴール・クエスチョン・メトリクス）
  - プロジェクトリスクマネジメント

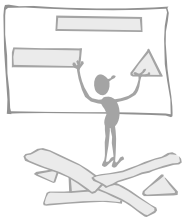


ソフトウェア品質保証技術の発展：剥離期

- プロセス改善技術の剥離：何が癒着していたか
  - カイゼンの考え方・進め方
    - » 品質サイクル：PDCA、DMAIC
    - » プロセス成熟度：SPA/SPI（標準化＋測定＋改善）
    - » 振り返ること
    - » 目標と現状のギャップを調べて追いつくこと
  - 物事の進め方、人の育て方
  - 何を伝えるか、どう伝えるか、どうやる気を出すか
- プロセス改善技術の剥離：剥離して得られた技術
  - プロセス改善モデル（CMM/CMMI、SPICE/ISO15504、TQC/TQM）
  - 標準化（プロセス標準、ライフサイクルモデル、ISO/IEC12207）
  - 落ち穂拾い、ポストモーテム、振り返り
  - 教育・OJT
  - QCサークル
  - ベンチマーキング、AS-IS/TO-BE分析

ソフトウェア品質保証技術の発展：剥離期

- マネジメント技術の剥離：何が癒着していたか
  - 思った通りに進める方法
    - » 計画→トレース→対策
    - » 実施確認
- マネジメント技術の剥離：剥離して得られた技術
  - プロジェクトマネジメント、PMBOK
  - 監査
  - 構成管理・変更管理・不具合管理
  - 要件管理・要件トレーサビリティ
  - 目標値管理・信頼度成長曲線
  - 文書化



剥離期の弊害

- 標準化・仕組みづくり
  - 標準を守れば品質が上がる
    - » 誰でも同じように仕事のできる標準が欲しい
    - » 標準はあるがカイゼンは無い
  - 品質をプロセス“だけ”で作り込もうとしている
    - » プロセスの質の向上が製品の質の向上に
      どう寄与しているかが分からない
  - フィードバック無しで上流で品質を作り込む
    - » 外部の技術や力で最初から完璧なモノを作ろうとする
  - 標準を遵守させようとすることによって
    失敗を許容せず、しかし失敗を繰り返す
    - » 現場評価によるフィードバックの無い標準を作り続け
      遵守させるだけの組織になっている

剥離期の弊害

- 組織と文化・考え方
  - 品質管理は品質保証部の仕事
    - » 僕作る人あなた直す人
    - » 犯人捜しをする
  - 事業部長や経営陣は納期優先コスト第一の施策を取る
    - » 口先では品質第一と言うので始末が悪い
  - 管理者と部下の役割を分担し責任範囲を決めることこそがマネジメントだと思っている
  - すぐ「改革」と口にする上層部がいる
  - 短気的で局所的な施策が多い
    - » 悪い成果主義になっている
    - » しかも流行に踊らされ慣れない言葉を振り回す
    - » もしくは現場を無視して全社展開ばかり気にする
  - 問題を見ず現象を見る
    - » 見えないようにすることで複雑さに対処する

剥離期の弊害

- 取り組み
  - データを語ろうとしている
    - » 現場の問題を見ずにデータだけで語ろうとし、データが目的になる
  - モグラ叩きになっている
    - » 応急処置で品質を確保するが再発し、応急処置に追われ前向きな仕事ができない
  - ムダに頭がよい
    - » 完璧主義・減点主義
    - » 議論ばかりして前に進まない
  - 開発とレビューとバグ分析とOJTを別々に実施する
    - » バグ分析の結果がフィードバックされず、組織に蓄積されない
    - » ノウハウDBなどを作ろうとするが、頓挫する
  - 一番肝心な「ひと」を無視している
    - » 小集団活動どころか残業に次ぐ残業でデスマーチになり、メンタルトラブルで離脱するエンジニアが出てしまう
  - カイゼンは余計な押しつけ作業であり現場の負担になる、と思っている

講演の流れ

- ソフトウェアの品質や信頼性は低い
- 品質とは
- SQuBOKに垣間見える日本的品質管理の特徴
- ソフトウェア品質保証技術の発展：癒着期、剥離期
- テストに着目した融合期の技術
- Wモデルの導入／テスト容易性設計
- 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
- 不具合モードによるテスト・レビュー・開発の改善
- テスト「道」

ソフトウェア品質保証技術の発展：融合期

- 剥離した技術を融合することで剥離した弊害を防止し、
  適切にソフトウェア品質保証技術を適用していく
  - 開発とV&V、プロセス改善、マネジメント技術、教育
- 技術を融合することで、
  余計な工数をかけずに上手に作る
  - フィードバック・フィードフォワード
    - » 悪さの知識をフィードバックして、最初から上手く作る
    - » 心配事をフィードフォワードして、みんなで気をつける
  - イメージバック・イメージフォワード
    - » この仕様や設計が生まれた根拠は何かを
      遡って考えて、仕様や設計を改善する
    - » この仕様や設計によってどんな振る舞いになるかを
      先読みして、仕様や設計を改善する

テストに着目した融合期の技術

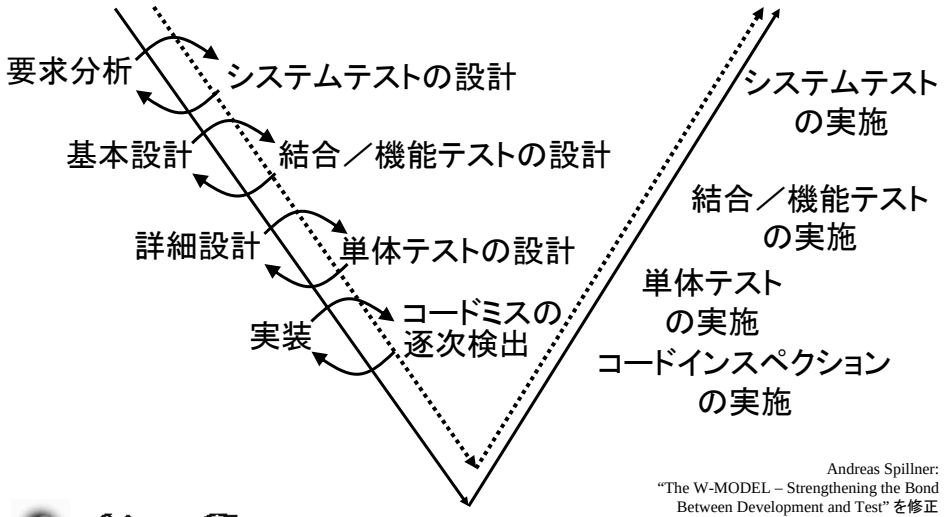
- テストの改善を進めることで、プロセス全体の改善を行っていく
  - Wモデルの導入
    - » 分析・設計・実装とテスト設計とを同時に進めることにより、レビューを改善するとともに、依存関係を減らして設計をスッキリさせる
  - テスト容易性設計
    - » テストが容易になるよう気遣うことで、設計をスッキリさせる
  - テスト分析・設計モデルの記述と要求・設計へのフィードバック
    - » テスト観点を体系的・網羅的に把握することで、分析・設計での考慮漏れを可視化し防止する
  - 不具合モードによるテスト・レビュー・開発の改善
    - » バグが作り込まれやすいところをレビュー・テストしたり、バグを作り込まないように開発ガイドラインを作る

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

講演の流れ

- ソフトウェアの品質や信頼性は低い
- 品質とは
- SQuBOKに垣間見える日本的品質管理の特徴
- ソフトウェア品質保証技術の発展：癒着期、剥離期
- テストに着目した融合期の技術
- Wモデルの導入／テスト容易性設計
- 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
- 不具合モードによるテスト・レビュー・開発の改善
- テスト「道」

Wモデル：上流と同期したテスト設計



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

Wモデルの利点と活用

- なぜWモデルが機能するか
  - 要するにレビュー強化の活動である
    - » テスト設計によって要件・設計の矛盾や一貫性欠如の発見ができる
    - » プロセスワイドなテストファーストである
  - 期待結果の導出によりイメージフォワードができる
    - » 要件・設計の振る舞いをきちんと考慮できる
  - 体系的なテスト設計によるイメージバックができる
    - » テストを設計する際に、仕様を網羅するため元の要求を考えるから、なんでその仕様が存在するかをしっかり考える
  - 網羅するので、頻度の低い処理の組み合わせができる
    - » 多重障害など
  - システム全体のように担当するスコープよりも広い範囲をチェックできる
    - » 切り分けが早すぎて、切り分けの見直しをしていない
  - （組み合わせによる）テストケース数の爆発が定量的に把握できるので、リファクタリングの圧力を高められる
- どうWモデルを機能させるか：無理せず小さく回す
  - まずVモデルの段階で（下流で）、体系的に質の高いテスト設計を行う
  - 下流で、テストを実施せずにテストを設計するだけでバグが見つかるようになる
  - 同じことを単に上流で行えばもっと早くバグが見つかる、という実感を現場が持つ
  - 同じようなところについて、なるべく工数をかけずに上流でテストを設計していく

Testability（テスト容易性）を考慮して開発を行う

- Testabilityを考慮して開発を行う／レビューをする
  - Testability: テスト容易性（DFT: Design For Test）
  - James BachのTestabilitiy要素
    - » 操作性: ソフトウェアがうまく動けば動くほど、テストはどんどん効率的になる
    - » 観測性: 見えるものしかテストできない
    - » 制御性: ソフトウェアをうまくコントロールすれば、テストを自動化し最適化できる
    - » 分解性: テストの適用範囲を制限することにより、より速やかに問題を切り分け、手際よく再テストを行える
    - » 単純性: テストする項目が少なければ少ないほど、テストを速やかに行える
    - » 安定性: 変更が少なければ少ないほど、テストへの障害が少なくなる
    - » 理解容易性: より多くの情報があれば、それだけ手際よくテストができる

上流からTestabilityを考慮して開発すると
品質の高いソフトウェアを作ろうとする圧力が働く

講演の流れ

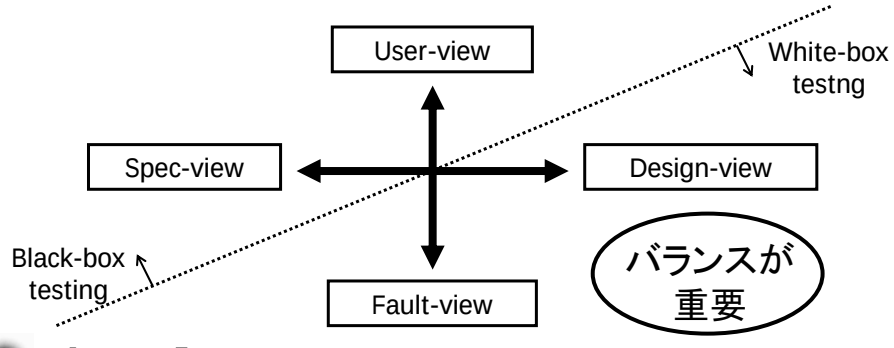
- ソフトウェアの品質や信頼性は低い
- 品質とは
- SQuBOKに垣間見える日本的品質管理の特徴
- ソフトウェア品質保証技術の発展：癒着期、剥離期
- テストに着目した融合期の技術
- Wモデルの導入／テスト容易性設計
- テスト分析・設計モデルの記述と要求・設計へのフィードバック
- 不具合モードによるテスト・レビュー・開発の改善
- テスト「道」

テストの「観点」

- テストには、様々な「観点」が必要だと言われている
  - Ostrandの4つのビュー
    - » ユーザビュー、仕様ビュー、設計・実装ビュー、バグビュー
  - Myersの14のシステムテスト・カテゴリ
    - » ボリューム、ストレス、効率、ストレージ、信頼性、構成、互換性、設置、回復、操作性、セキュリティ、サービス性、文書、手続き
  - ISO/IEC 9126の品質特性
    - » 機能性、信頼性、使用性、効率性、保守性、移植性
- テストの「観点」とは何だろう？ ー テスト対象のモデリング
  - テスト対象の持つ、テストすべき側面
  - テスト対象が達成すべき性質
  - テスト対象（及び含む世界）を、テストの立場からモデリングしたもの
    - » テストする必要が無い側面は、モデリングする必要が無い
    - » 達成する必要が無い性質は、モデリングする必要が無い
  - 抽象的で、階層構造を持つ
    - » 同値分割の包括的なもの

Ostrandの4つの観点

- User-view
  - ユーザが何をするかを考える
- Fault-view
  - 起こしたいバグを考える
- Spec-view
  - 仕様を考える
- Design-view
  - 設計やソースコードを考える



組込みのテストの設計で考慮すべき観点の例

- 機能:テスト項目のトリガ
  - ソフトとしての機能
    - » 音楽を再生する
  - 製品全体としての機能
    - » 走る
- パラメータ
  - 明示的パラメータ
    - » 入力された緯度と経度
  - 暗黙的パラメータ
    - » ヘッドの位置
  - メタパラメータ
    - » ファイルの大きさ
  - ファイルの内容
    - » ファイルの構成、内容
  - 信号の電気的ふるまい
    - » チャタリング、なまり
- プラットフォーム・構成
  - チップの種類、ファミリ
  - メモリやFSの種類、速度、信頼性
  - OSやミドルウェア
  - メディア
    - » HDDかDVDか
  - ネットワークと状態
    - » 種類
    - » 何といくつつながっているか
  - 周辺機器とその状態
- 外部環境
  - 比較的変化しない環境
    - » 場所、コースの素材
  - 比較的変化しやすい環境
    - » 温度、湿度、光量、電源

組込みのテストの設計で考慮すべき観点の例

- 状態
  - ソフトウェアの内部状態
    - » 初期化処理中か安定動作中か
  - ハードウェアの状態
    - » ヘッドの位置
- タイミング
  - 機能同士のタイミング
  - 機能とハードウェアのタイミング
- 組み合わせ
  - 同じ機能をいくつカブせるか
  - 異なる機能を何種類組み合わせるか
- 性能
  - 最も遅そうな条件は何か
- 信頼性
  - 要求連続稼働時間
- GUI・操作性
  - 操作パス、ショートカット
  - 操作が禁止される状況は何か
  - ユーザシナリオ、10モード
  - 操作ミス、初心者操作、子供
- 出荷先
  - 電源電圧、気温、ユーザの使い方
  - 言語、規格、法規
- 障害対応性
  - 対応すべき障害の種類
    - » 水没
  - 対応動作の種類
- セキュリティ
  - 扱う情報の種類や重要度
  - 守るべきセキュリティ要件

非常に多くの観点を網羅的に設計する必要がある

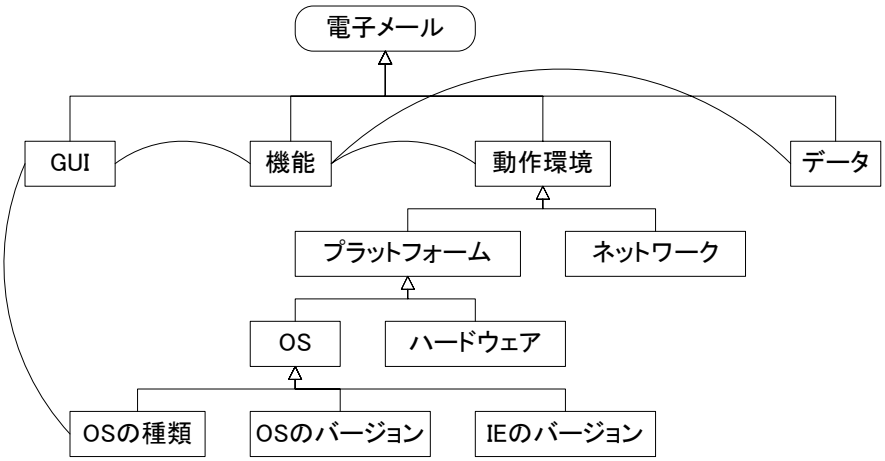
大・中・小項目型テスト設計の問題例

大項目	中項目	小項目	テストケース
機能レベル1	機能レベル2	機能レベル3	機能レベル10
機能レベル1	機能レベル8	機能レベル9	機能レベル10
機能	環境	データ	機能＋環境＋データ
機能	データ	GUI	機能＋データ＋GUI
機能	データ	前提条件	機能＋データ
機能	状態	イベント	状態＋イベント
テストカテゴリ	機能	データ	機能＋データ
組み合わせ	機能①	機能②	機能①×機能②

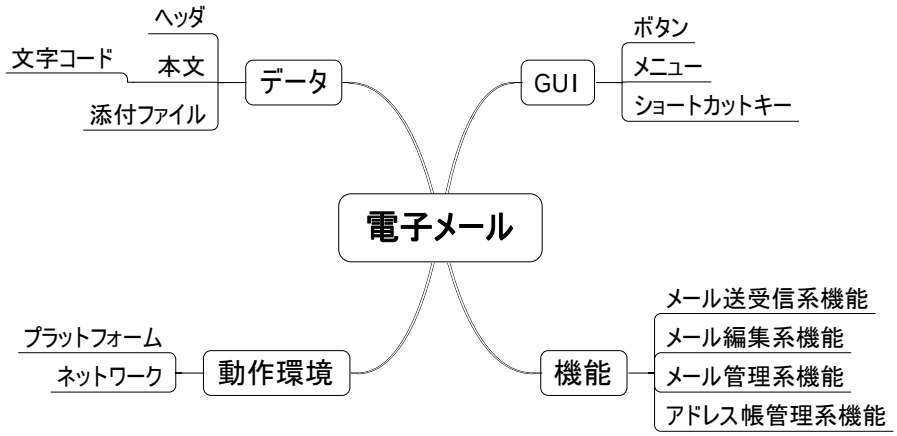
大・中・小項目型テスト設計の問題

- 詳細化のレベルが揃わない
  - 偏った詳細化をしてしまう
  - テストケース群ごとに詳細化の偏りにばらつきがある
- 異なるテスト観点の組み合わせを詳細化だと考えてしまう
  - 機能＋環境＋データ、機能＋データ＋GUI
- テスト設計で考慮する必要の無いものが入ってしまう
  - 機能＋データ＋前提条件
- 異なるテスト観点にぶら下げているので網羅できない
  - 機能＋状態＋イベント
    - » 本来は状態遷移網羅をすべきであり、機能網羅で代用すべきではない
- テスト観点の詳細化を行わない
  - テストカテゴリ＋機能＋データ
    - » 負荷にも色々あるはず．．．。
- 組み合わせテストを押し込んでしまう
  - n元表を使うべきである

テスト分析モデルの例：ツリー風記述（NGT）



テスト分析モデルの例：マインドマップ



改善のためのテスト観点によるテスト漏れの分析

- テスト漏れの原因をテスト観点を軸にして分析し、改善する
  - モデリングの失敗
    - » ビューのモレ／関連のモレ → 要求分析のモレ
    - » 不適切／曖昧なビューの解釈 → 要求解釈のモレ
  - 剪定の失敗
    - » クラス内の網羅基準の緩和のしすぎ → 設計品質の把握不足
    - » ズームアウトのしすぎ→ 設計品質の把握不足
    - » 関連の組み合わせ基準の緩和・削除のしすぎ → 設計の結合度の過多
  - リスク確定の失敗
    - » ソフトウェア設計の内部まで突っ込んで確定しなかった → 設計の把握不足
    - » ソフトウェア設計を基に確定したのに、設計どおり実装されていなかった → 管理ミス
    - » 利用頻度の見積もりを誤った → 要求分析のモレ
  - それ以外の失敗
    - » 不具合が作り込まれる原因にさかのぼって分析しパターン化し、バグの入り込みやすいハザード、分析、設計、実装のパターンリストを作り改善する

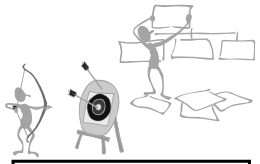
講演の流れ

- ソフトウェアの品質や信頼性は低い
- 品質とは
- SQuBOKに垣間見える日本的品質管理の特徴
- ソフトウェア品質保証技術の発展：癒着期、剥離期
- テストに着目した融合期の技術
- Wモデルの導入／テスト容易性設計
- 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
- 不具合モードによるテスト・レビュー・開発の改善
- テスト「道」

基本的なテストの方針

- 「テストしなければバグは見つからない」
  - 漏れなくテストを行う
  - 漏れなくテストを行うためには、
    思いつきでテストをあげてはならない

網羅



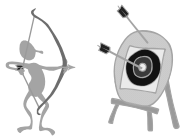
- 「大事なところをテストする」
  - 最もバグが起きそうなところにテストを行う
  - バグが起きてはいけない順にテストを行う

ピンポイント

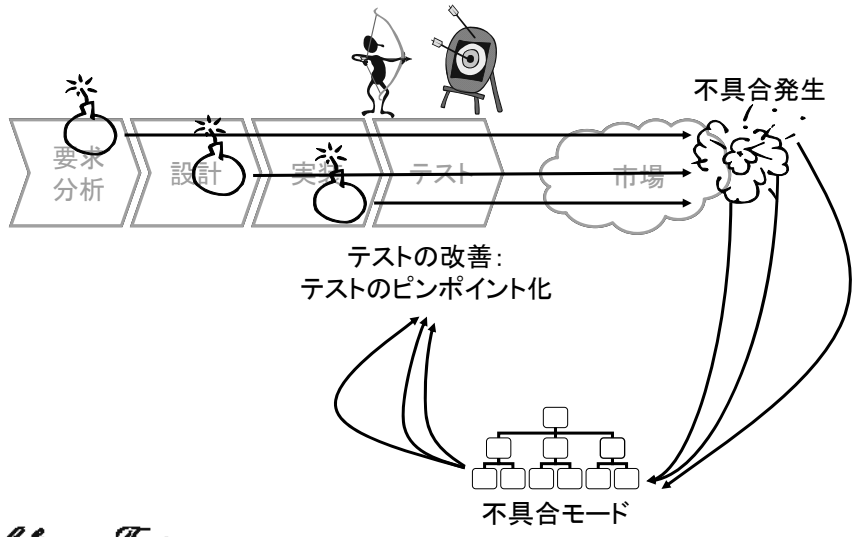
少ない手間で早くたくさん危険なバグを検出する

不具合モードによるフィードバックテスト

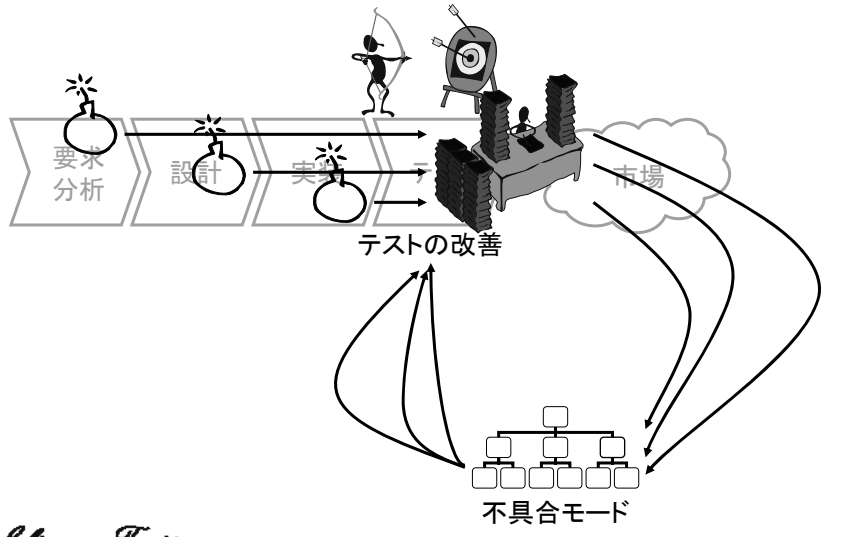
- 不具合が起きそうな「弱点」を狙って
  ピンポイントでテスト項目やレビュー時の指摘事項を設計する
  - 過去に検出された不具合を精査し、同じような原因で発生していると
    推測できる不具合を列挙する
    - » 始めは特定の製品の特定のバージョンに絞った方がよい
  - 不具合の原因となるメカニズムをモデル化する
    - » モデル化されたメカニズムが、再利用のためのテスト資産となる
  - モデルに当てはまる構造を持つ機能のテストを設計する
    - » きちんとレビューやインスペクションを行っている組織では、
      その結果を同じように資産化しておき、フィードバックテストに用いるべきである
- 開発とのコミュニケーションを密にするツールにもなる
  - テスト側から「弱点」(不具合の起きやすい構造)を
    開発にフィードバックできる
  - 開発側が「弱点」を把握しテスト側に伝える
    “フィードフォワードテスト”につながっていく



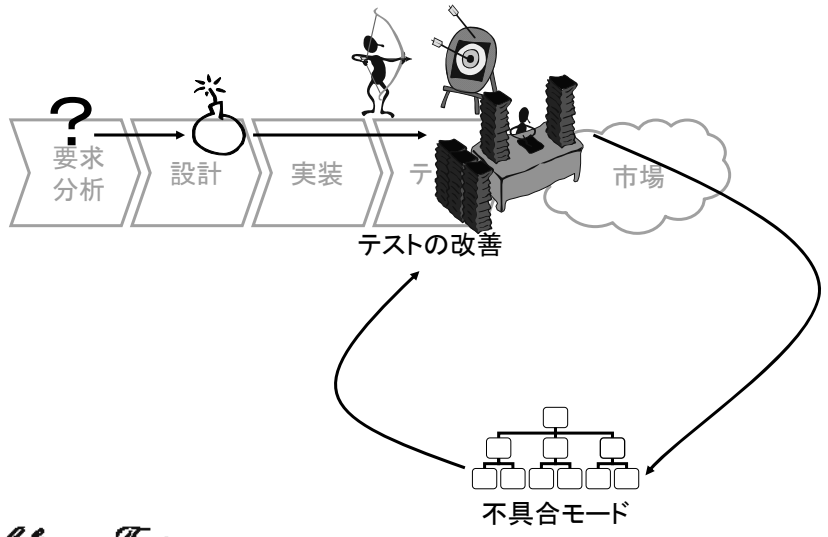
市場に流出した欠陥に対する不具合モード分析



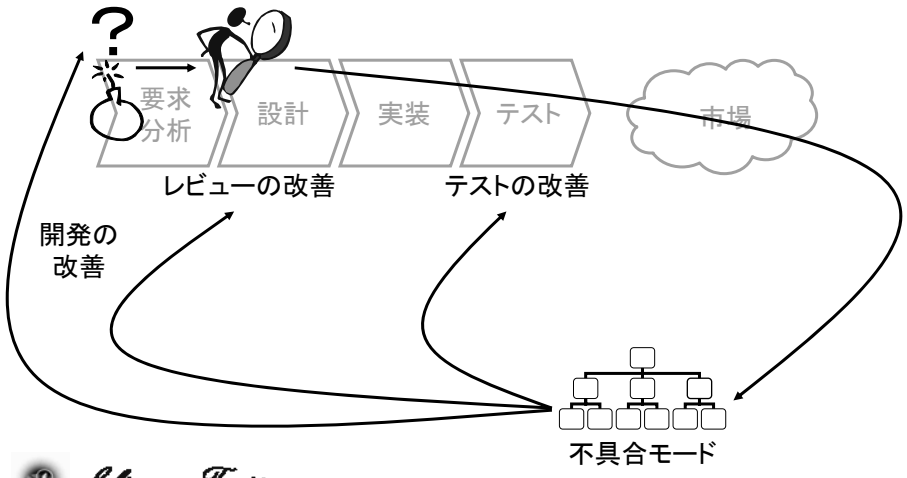
テストで検出した欠陥に対する不具合モード分析



欠陥作り込みリスクに対する不具合モード分析

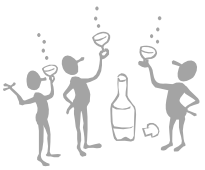


不具合モード分析による開発・レビュー・テストの改善

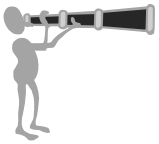


不具合モードによるフィードバックテスト

- フィードバックテストのメリット
  - テスト漏れを防ぐことができる
    - » 2度と同じテスト漏れは起こさないようにできる
  - 上手な「間引き」をすることができる
    - » バグの見つからないテストをうまくサボることができる



- 本当のメリット
  - 自分たちの開発の「弱点」のリストを手に入れることができる
    - » ツール屋さんやコンサルタントからは入手できない
  - バグを分析し自分たちの「弱点」を的確に把握ことで、効果的かつ効率的なプロセス改善を行うことが出来る
    - » プロダクトのバグの分析とフィードバックだけでなく、プロジェクトのリスクの分析とフィードバックも行うべきである



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

- ソフトウェアの品質や信頼性は低い
- 品質とは
- SQuBOKに垣間見える日本的品質管理の特徴
- ソフトウェア品質保証技術の発展：癒着期、剥離期
- テストに着目した融合期の技術
- Wモデルの導入／テスト容易性設計
- 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
- 不具合モードによるテスト・レビュー・開発の改善
- テスト「道」

改善そのもののレベルを判定する

- 改善レベル0：暴飲暴食レベル
  - 全く改善しないレベル
- 改善レベル1：薬漬けレベル
  - 問題点を把握せず標準に闇雲に合わせるだけのレベル
- 改善レベル2：民間療法レベル
  - 感覚的・経験的に問題点を把握し改善するレベル
- 改善レベル3：対症療法レベル
  - メトリクスを測定し問題点を把握しているにもかかわらず、
    問題点の根本原因やメカニズムを理解せずに改善するレベル
- 改善レベル4：治療レベル
  - 問題点の根本原因やメカニズムを把握して改善し再発防止を行うレベル
- 改善レベル5：予防レベル
  - 問題点の根本原因やメカニズムを水平展開して改善し未然予防するレベル
  - 工程の融合を必要とすることが多い

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

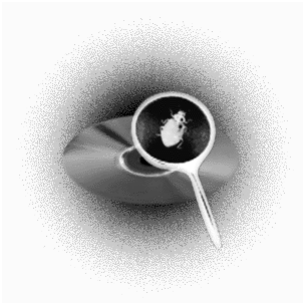
フェーズ1 ー テストの目的は、
ソフトウェアが動くことを示すことである

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