テストを軸としたソフトウェア品質の改善

ソフトウェアテストシンポジウム新潟 2011

2011/2/18（金）

電気通信大学 大学院 情報理工学研究科
総合情報学専攻 経営情報学コース
西　康晴

© NISHI, Yasuharu

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
- もろもろ
  - TEF: テスト技術者交流会 / ASTER: テスト技術振興協会
  - WACATE: 若手テストエンジニア向けワークショップ
  - 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

© NISHI, Yasuharu

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1700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

© NISHI, Yasuharu

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1600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10373名の受験者・5694名の合格者
    - » Advanced Level（テストマネージャ）を2010年8月に開始
  - 各地でソフトウェアテストの教育を行っている
    - » テストのスキル標準（test.SSF）をIVIAと共同で開発している
    - » 札幌や大分などでテストの教育の実績がある
    - » 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
  - テスト開発方法論などの先端技術を研究開発している：智美塾

© NISHI, Yasuharu

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

© NISHI, Yasuharu

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

SQiP
Software Quality Profession

© NISHI, Yasuharu

Safeware翻訳プロジェクト

- 2009年11月に翻訳版を刊行した
  - 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）
  - 翔泳社 発行

第1部　リスクの性質
- 第1章　近代社会におけるリスク
- 第2章　コンピュータとリスク
- 第3章　事故の階層的考察
- 第4章　事故の根本原因
- 第5章　ヒューマンエラーとリスク
- 第6章　自動化システムにおける人間の役割

第2部　システム安全への序論
- 第7章　システム安全の基礎
- 第8章　システム安全の基本

第3部　定義とモデル
- 第9章　用語
- 第10章　事故とヒューマンエラーモデル

第4部　セーフウェアプログラムの要素
- 第12章　システムとソフトウェア安全プロセス
- 第13章　ハザード分析
- 第14章　ハザード分析モデルと技法
- 第15章　ソフトウェアハザードと要求分析
- 第16章　安全性のための設計
- 第17章　ヒューマンマシンインタフェースの設計
- 第18章　安全性の検証

付録
- 付録A. 医療機器：Therac-25の歴史
- 付録B. 航空宇宙：アポロ13号、DC-10型機、チャレンジャー号
- 付録C　化学産業：セベソ、フリックスボロー、ボパール
- 付録D　原子炉事故：ウィンズケール、スリーマイル島、及びチェルノブイリ

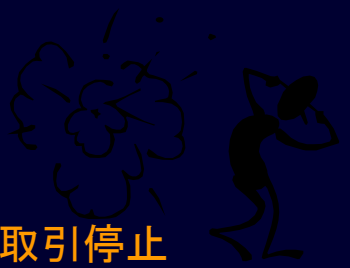
© NISHI, Yasuharu

講演の流れ

- ソフトウェアの低品質による経営リスク
- 受託ソフトウェア組織の基本戦略：高カイゼン戦略
- （少し進んだ）組込みソフトウェア開発現場の悩み
- ソフトウェア品質に対する日本的な考え方
- カイゼンし続けるソフトウェア組織のイメージ
- ソフトウェア品質保証技術の発展
- 開発現場を「融合」しよう
- Wモデル：開発者が「最もよいやり方で考える」ことに近づけていくプロセス
- テスト「道」

© NISHI, Yasuharu

ソフトウェアの品質は低いと言わざるを得ない

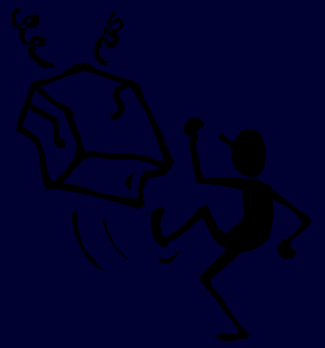


- エンタープライズ系システム
  - メガバンクの情報システム統合に伴う品質事故は記憶に新しい
  - 東京証券取引所の情報システムは度重なる品質事故
    - » 売買システムの処理増強に伴う品質事故により全銘柄で午前中の取引停止
    - » ジェイコム株誤発注事件では品質事故により400億円超の損害賠償請求訴訟
  - 羽田空港の航空管制システムの品質事故は30万人以上が足止め
  - 成功するプロジェクトはわずか26.7%（日経コンピュータ）
- 組込み系システム
  - 国内の携帯電話：65万台が回収・無償交換、131億円にのぼる損害
  - ドイツの高級車のブレーキシステム：68万台がリコール
  - 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

© NISHI, Yasuharu

ソフトウェアの低品質による経営リスク

- ソフトウェアの低品質による失注・損害リスク
  - 低品質に起因する手戻り多発による失注・利益損失
  - 低品質に起因するメンタルトラブルなどによる人財損失
- ソフトウェアの低品質による訴訟リスク
  - ソフトウェアの低品質などにより訴訟が相次いでいる
  - 訴訟になると裁判費用や対応人員など多額のコストがかかり、信用も大きく低下する可能性がある
- ソフトウェアの低品質によるメディアリスク
  - サービス事業者の責任が前面に出るべきにも関わらず、受託ソフトウェア企業名が報道されるケースが相次いでいる



© NISHI, Yasuharu

ソフトウェアの低品質による失注・損害リスク

- 低品質に起因する手戻り多発による失注・利益損失
  - 低品質に起因する高コスト化・長納期化
    ・低精度の見積もりと高い安全係数による失注
  - 低品質に起因する信用損失による失注
  - 手戻り作業増大による開発コストの超過
  - マネジメント作業増大による管理コストの超過
  - 謝罪訪問増大による営業コストの超過
  - 準備不足のアウトソーシング・オフショアによる失敗コストの超過
- 低品質に起因するメンタルトラブルなどによる人財損失
  - 過大労働からのメンタルトラブルによる人財損失
  - 達成感欠如からのメンタルトラブルによる人財損失
  - 開発現場の人間関係悪化による人財損失
  - 教育不足・ノウハウ蓄積不足からの成長感欠如による人財損失

技術者としての「やりがい」が全くない
牢獄のような現場になってしまう

© NISHI, Yasuharu

ソフトウェアの低品質などによる訴訟リスク

- ソフトウェアの低品質などにより訴訟が相次いでいる
  - 「情報システムやソフトウェアに関連する訴訟の数は確実に増加している。東京地裁だけでも、年間に20件を越す訴訟が起きていることが分かった。」
    - » 東京地裁・コンピュータ専門の民事調停委員／保科好信氏
    - » 日経コンピュータ2004/10/18号
- 訴訟になると裁判費用や対応人員など多額のコストがかかり、信用も大きく低下する可能性がある
  - みずほ銀行×東京証券取引所（係争中）
  - スルガ銀行×日本IBM（係争中）
  - ムトウ×東洋ビジネスエンジニアリング（調停・ゼロ和解）
    - » http://itpro.nikkeibp.co.jp/article/NEWS/20080319/296631/
  - エイチ・エス証券×大和総研（訴訟に至らず和解？）
    - » 以下全て日経コンピュータ2004/10/18号から
  - スペースリンク×アジアパシフィックシステム総研（調停・他ベンダで再構築）
  - 白寿生化学研究所×東京NTTデータ通信システムズ（調停・稼働せず）
  - アルソア×三井情報開発（調停・請求金額上限以上の解決金）

© NISHI, Yasuharu

ソフトウェアの低品質によるメディアリスク

- サービス事業者の責任が前面に出るべきにも関わらず、受託ソフトウェア企業名が報道されるケースが相次いでいる
  - 東京証券取引所→富士通
    - » システムを開発した富士通については「実損が出れば損害賠償請求を検討する」と述べた。(http://headlines.yahoo.co.jp/hl?a=20080722-00000027-maip-brf)
  - JR他→日本信号
    - » SuicaとPASMOに対応した自動改札機が起動しない不具合が発生
  - JR東海→東芝
    - » 東海道新幹線自動改札機でトラブル
  - 川崎市→富士通
    - » 富士通が処理をミス、高齢者医療で誤送付と誤表記
  - 愛媛県→愛媛電算
    - » 後期高齢者医療制度：県内147人分の保険料、プログラムミスで天引きできず
  - 富山市→インテック
    - » 富山市：国保納入通知書でミス1087通発送
  - 電力6社・日本原子力研究開発機構→日立
    - » 原発配管の強度評価に誤り　17基でプログラムミス

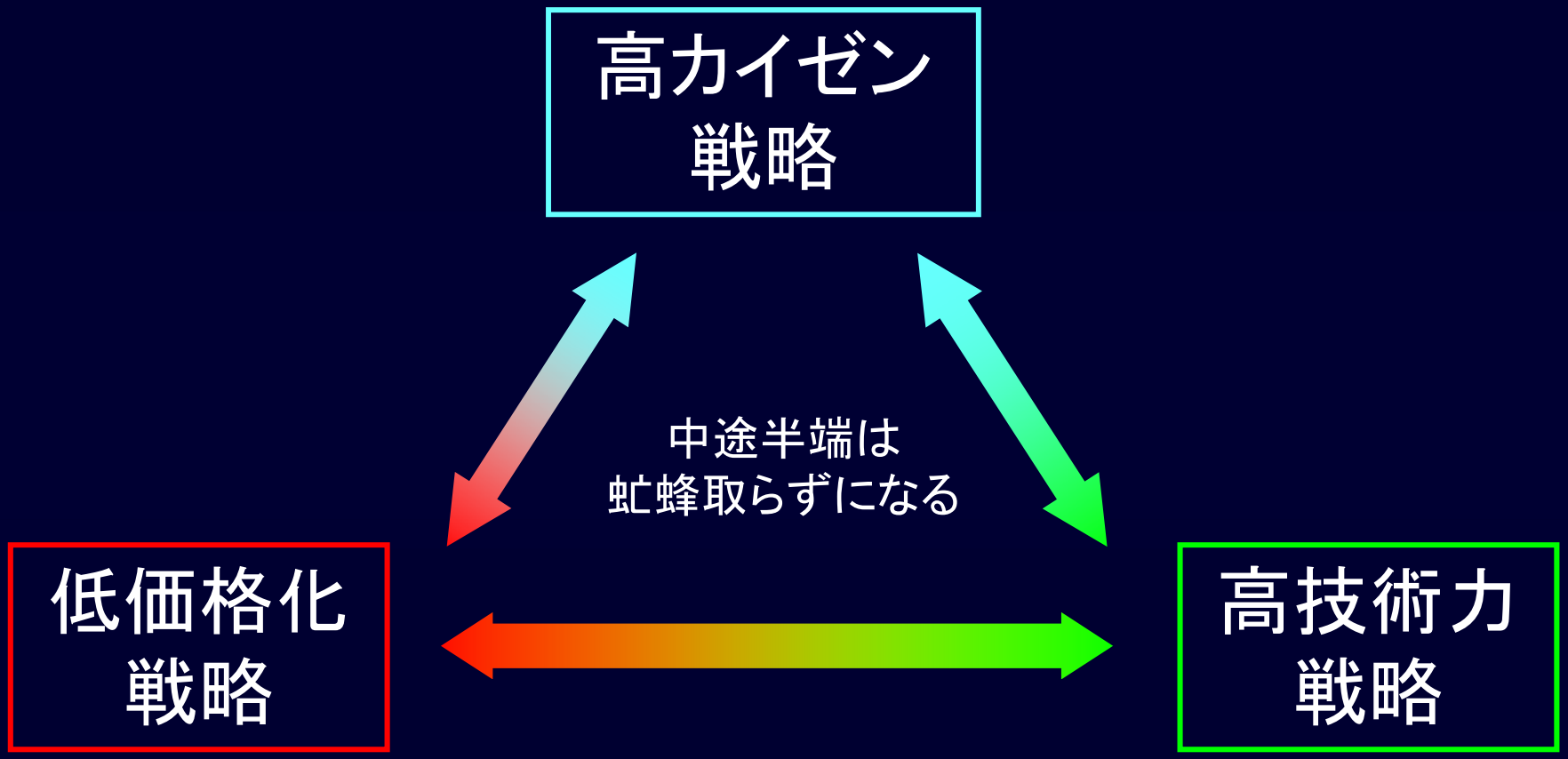
© NISHI, Yasuharu

講演の流れ

- ソフトウェアの低品質による経営リスク
- 受託ソフトウェア組織の基本戦略：高カイゼン戦略
- （少し進んだ）組込みソフトウェア開発現場の悩み
- ソフトウェア品質に対する日本的な考え方
- カイゼンし続けるソフトウェア組織のイメージ
- ソフトウェア品質保証技術の発展
- 開発現場を「融合」しよう
- Wモデル：開発者が「最もよいやり方で考える」ことに近づけていくプロセス
- テスト「道」

© NISHI, Yasuharu

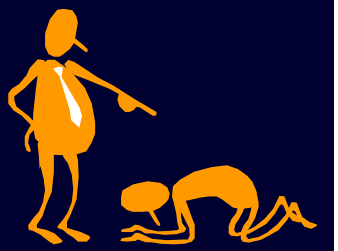
受託ソフトウェア組織の経営ポジショニング



© NISHI, Yasuharu

低価格化戦略

- キーワード:「人材の世界最適調達」
  - 例)デルコンピュータ
- 徹底した低価格化と大規模化の戦略
  - 低コストの人材を調達し、競争力のある価格を提示する
    - » 国内派遣労働者かオフショア人材の調達が主になる
    - » 調達コストが高く、削減するとスキル・モチベーション・規模などに如実に表れる
  - 薄利多売のために高コストでの営業が必要となる
    - » 規模拡大のためのコストと、謝罪コストの両方が増えていく
  - 低スキル人材を早急に活用するための高コストでの技術フレームワーク・マネジメントシステムの開発が必要となる
    - » 他社と差別化できるフレームワーク・ツール・マニュアルの開発が必要になる
- より大きな企業の下請けになりやすく、結局のところ価格競争で疲弊しやすい
  - 差別化するのがとても難しい戦略なので価格競争になりやすい
    - » 資格取得は基礎であって差別化ではない
  - 元請けの状況や景気に左右されやすく稼働損リスクが意外に高い
  - 社内が疲弊するのは簡単だが、力を付けるのはとても難しい



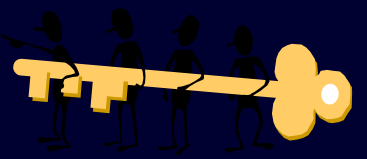
© NISHI, Yasuharu

高技術力戦略

- キーワード:「オンリーワン」
  - 例)インテル、シマノ
- 高い技術力によって顧客をリードする高価格化戦略
  - 高コストの人財によって差別化可能な技術を提供し、プレミア価格を提示する
    - » 顧客が頭を下げてくるほどの技術力を持つ
  - 初期調達コストは非常に高いが、評判が立てば調達コストも営業コストも下がる
    - » 腕の良いエンジニアは腕の良いエンジニアと働きたい
    - » 高い技術力はクチコミで評判になり、腕の良いエンジニアは広告塔になる
    - » 満足度が高いので謝罪コストが低く済む
  - 高スキル人財を活用するためのオープンなマネジメントスタイルが必要となる
    - » アットホームかつフラットで技術重視のマネジメント・人事評価システムと、余裕のあるワークスタイルを実現する必要がある
- 経営陣の「度量」があるうちは、そこそこの規模で高利益率を保持できる
  - 高技術力戦略を現在採っていない企業がシフトするのは難しい
  - 人財のばらつきが大きくなるとマネジメントが急に難しくなるため、大規模化は頭打ちになる
  - 人財の独立をフロー的に上手くマネジメントする必要がある

© NISHI, Yasuharu

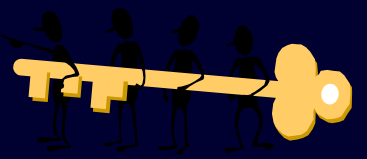
高カイゼン戦略



- キーワード:「周りを巻き込んで成長し続ける」
  - 例)トヨタ(グループ)
- 自社のみならず顧客やパートナーもカイゼンすることにより、価格以上の付加価値を与える戦略
  - 顧客に製品を提供するだけでなく、顧客の業務プロセスのカイゼンも提供する
    - » お宅に頼むとうちのエンジニアが成長する、と顧客に言わせる
  - 高コストの人財や調達が必要とは限らないので、価格をそれほど上げずに済む
    - » 顧客満足度が高いためリピートオーダーや社内展開が多く営業コストが下がる
    - » 顧客の業務に深く入り込むため乗り換えられにくく、単価の高い業務へのシフトがしやすくなる
    - » パートナーと共に成長するため、パートナーが優先的に良い人財を回してくれやすい
  - 常に悪さの情報を共有し、原因を分析し、カイゼンし、未然防止につなげ、目先の利益にとらわれず品質を高め組織力を向上する、という「ぶれない」組織カルチャーやマネジメントスタイルを持つことが必要となる
    - » マネジメントコストがかかるように見えるが、実はノウハウ蓄積コストなので見返りが大きい
    - » ミスをしたエンジニアを責めず、むしろ適切な未然防止活動は高く評価すべきである

© NISHI, Yasuharu

高カイゼン戦略



- カイゼンを進め続け品質を高め続ける「覚悟」を
  経営陣が持ちトップが徹底し続けられれば、
  そこそこ高い利益率で安定した成長を見込むことができる
  - カイゼンは「今日よりも明日はもっと良い仕事をする」という原則なので、
    現在の技術力が低くても高カイゼン戦略を始めることができる
    - » 何かの取り組みをすれば割とすぐに達成できるというものではなく、
      腰を据えて長期的な視点で徹底的に進める覚悟が必要である
  - 品質を高めようとすると手戻りが少なくなるため、
    コストが下がり納期が短くなるという原則を全員が理解する必要がある
    - » 高品質をムダな監査、ムダなテストやレビュー、ムダなドキュメントで
      達成しようとしてはいけない
    - » 悪さを把握し原因を潰そうとするため当初はコストがかかるように見えるが、
      手戻りが減るので作業効率を向上でき納期を短縮できる
  - 「カイゼンしている暇があったら利益の出る仕事をしろ」などという
    ミドル・トップが１人でもいると、高カイゼン型の組織カルチャーは育たない
    - » 利益を無視して些細なバグにこだわるという意味ではない

© NISHI, Yasuharu

受託ソフトウェア組織の基本戦略

「カイゼンによる品質向上」は
非常にロバストな経営戦略である

© NISHI, Yasuharu

講演の流れ

- ソフトウェアの低品質による経営リスク
- 受託ソフトウェア組織の基本戦略：高カイゼン戦略
- （少し進んだ）組込みソフトウェア開発現場の悩み
- ソフトウェア品質に対する日本的な考え方
- カイゼンし続けるソフトウェア組織のイメージ
- ソフトウェア品質保証技術の発展
- 開発現場を「融合」しよう
- Wモデル：開発者が「最もよいやり方で考える」ことに近づけていくプロセス
- テスト「道」

© NISHI, Yasuharu

（少し進んだ）ソフトウェア開発現場の悩み

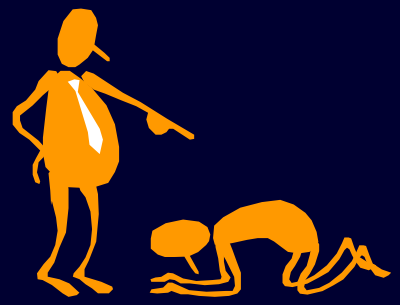
- 一生懸命開発しているが、一向によくならない
  - 製品やシステムを開発することに追われ、
    自分達の技術力が高まっているとは感じられない
    - » 現場は「これでよいのか」と思いながらも開発を進めざるを得ない
    - » 上司に聞いても「それしか方法が無い」もしくは「先送りだ」と言われる
    - » 元請けはイマイチな仕様ばかり出してくるのに、
      こちらの話を聞いてくれるわけではない
    - » 協力会社は頑張ってくれているが、仕事をこなすだけである
  - 様々なプラクティスを導入しているが、
    全体としてよくなっている実感が無い
    - » モデルベース開発、SA/SD、OO/UML、SPLE、XDDP、直交表によるテスト、
      ETSSによるスキル開発、CMMI、Automotive SPICE
    - » むしろ開発が重くなってしまった
    - » むしろ縦割り的になってしまった
    - » むしろ形式的で中身を伴わなくなってしまった

何だか
バラバラ

© NISHI, Yasuharu

開発現場がバラバラになる原因

- 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



© NISHI, Yasuharu

高カイゼン戦略を阻害する悪しき「受託根性」

- 悪しき受託根性
  - お客様の言ったことを完璧にこなすのが私たちの仕事です
    - » お客様の言わないことはやりません
    - » お客様に言われるまでやりません
    - » お客様の指示がおかしくても、指摘しません
  - 私たちの技術は、お客様のドメイン技術に詳しいことです
    - » 確かに、お客様のドメイン技術に精通していることは重要
    - » 受託側にはバッドノウハウが溜まりがちであることも事実
    - » ソフトウェア開発者としての技術にプライドを持っているとは限らない

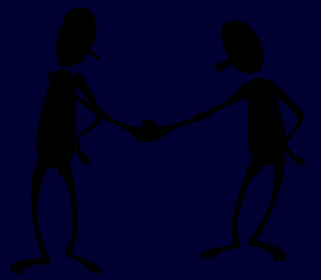
悪しき受託根性の強い組織に
支えられていると
品質は低い

© NISHI, Yasuharu

高カイゼン戦略を支える「フォロワーシップ」

- フォロワーシップ
  - 貢献力
    - » お客様の言ったことを粘り強く完璧にこなす能力
    - » お客様が働きやすいように表現したり立ち回る能力
    - » 自分達の作り方・自分達の詳細仕様の書き方をカイゼンする能力
  - 提案力（批判力）
    - » お客様にとってもっと良い業務や開発を常に提案し続ける能力
    - » お客様が言うことや判断することを先読みする能力
    - » お客様の仕様の書き方を（上手に）カイゼンする能力

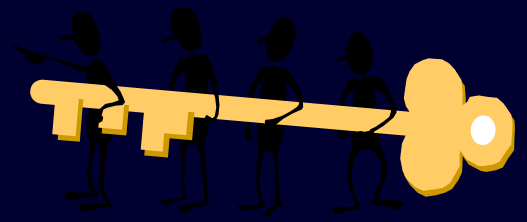
フォロワーシップの高い組織に
支えられていると
品質は高い



© NISHI, Yasuharu

受託ソフトウェア企業の高カイゼン戦略

- 自分達のカイゼン
  - 自分達の作り方をカイゼンする
    - » 複数の現場での経験を交換し、たくさんのノウハウをお客様に提供する
    - » 社外の技術動向を常に取り入れ続け、常に最新かつ王道の技術を身につける
  - 自分達の詳細仕様の書き方をカイゼンする
    - » 自分達が受託側になることで得た経験を活かして上手な委託をする
    - » 自分達に残す技術と競争力をしっかり考え抜く
- お客様のカイゼン
  - お客様よりも速く効果的な水平展開をする
    - » 悪さの知識を分析しパターン化しておく
  - お客様の仕様の書き方をカイゼンする
    - » 悪い仕様を分析してパターン化しておく
  - お客様の検収力も同時にカイゼンする
    - » お客様が検収時に着目すべき観点を洗い出し、抜け漏れの無い検収設計を行う



- パートナーのカイゼン
  - パートナーの仕事の進め方をカイゼンできるようになってもらう
    - » パートナーの技術力が向上するようにカイゼンの進め方を伝え、一緒に仕事をしながらやってみせて、できるようになってもらう
    - » 自分達がパートナーにやってみせられる程のカイゼン力が必要になる

© NISHI, Yasuharu

講演の流れ

- ソフトウェアの低品質による経営リスク
- 受託ソフトウェア組織の基本戦略：高カイゼン戦略
- （少し進んだ）組込みソフトウェア開発現場の悩み
- ソフトウェア品質に対する日本的な考え方
- カイゼンし続けるソフトウェア組織のイメージ
- ソフトウェア品質保証技術の発展
- 開発現場を「融合」しよう
- Wモデル：開発者が「最もよいやり方で考える」ことに近づけていくプロセス
- テスト「道」

© NISHI, Yasuharu

ソフトウェア品質に対する欧米的な考え方

- PMBOK: Project Management Body Of Knowledge
  - 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: SoftWare Engineering Body Of Knowledge
  - 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

© NISHI, Yasuharu

品質とコスト・納期はトレードオフであるという誤解

- 品質に対する欧米的な考え方
  - 顧客の要求に対する合致度といった「開発の結果」である
    - » PMBoK:「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」(ASQ)
    - » SWEBOK:よく読むと、品質の定義を巧妙に避けているが、要するに顧客満足度
  - 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
    - » 顧客の要求の達成どころか、
      自分の仕事の言い訳にしか使わない輩もいる

- 品質とコスト・納期はトレードオフであるというのは誤解である
  - 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

© NISHI, Yasuharu

ソフトウェア品質に対する日本的な考え方

- SQuBOK: Software Quality Body Of Knowledge
  - 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

© NISHI, Yasuharu

ソフトウェア品質に対する日本的な考え方

- 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
    - » ブランド力、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

© NISHI, Yasuharu

SQuBOKにみられ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

© NISHI, Yasuharu

SQuBOKにみられ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

© NISHI, Yasuharu

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

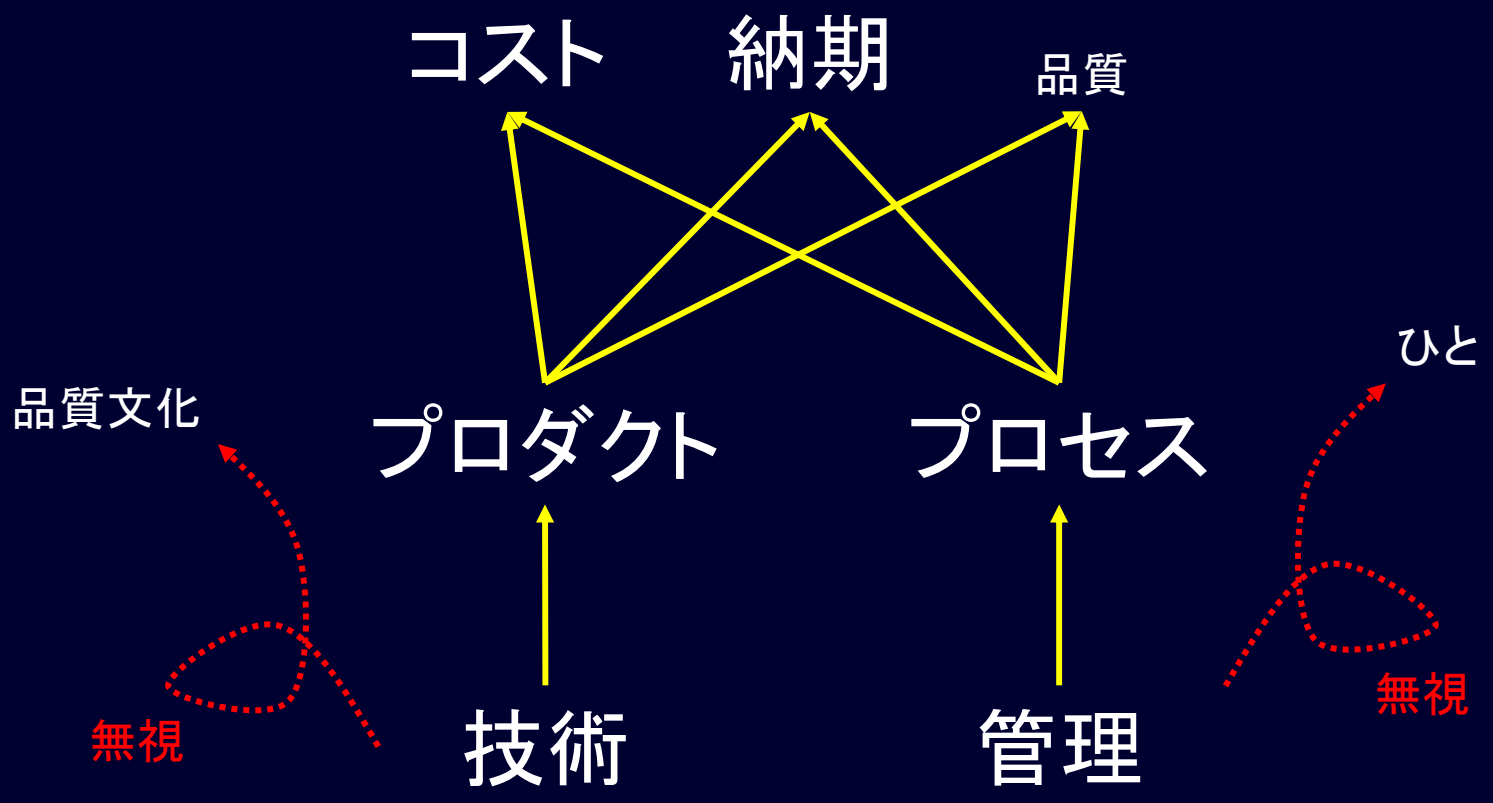
© NISHI, Yasuharu

講演の流れ

- ソフトウェアの低品質による経営リスク
- 受託ソフトウェア組織の基本戦略：高カイゼン戦略
- （少し進んだ）組込みソフトウェア開発現場の悩み
- ソフトウェア品質に対する日本的な考え方
- カイゼンし続けるソフトウェア組織のイメージ
- ソフトウェア品質保証技術の発展
- 開発現場を「融合」しよう
- Wモデル：開発者が「最もよいやり方で考える」ことに近づけていくプロセス
- テスト「道」

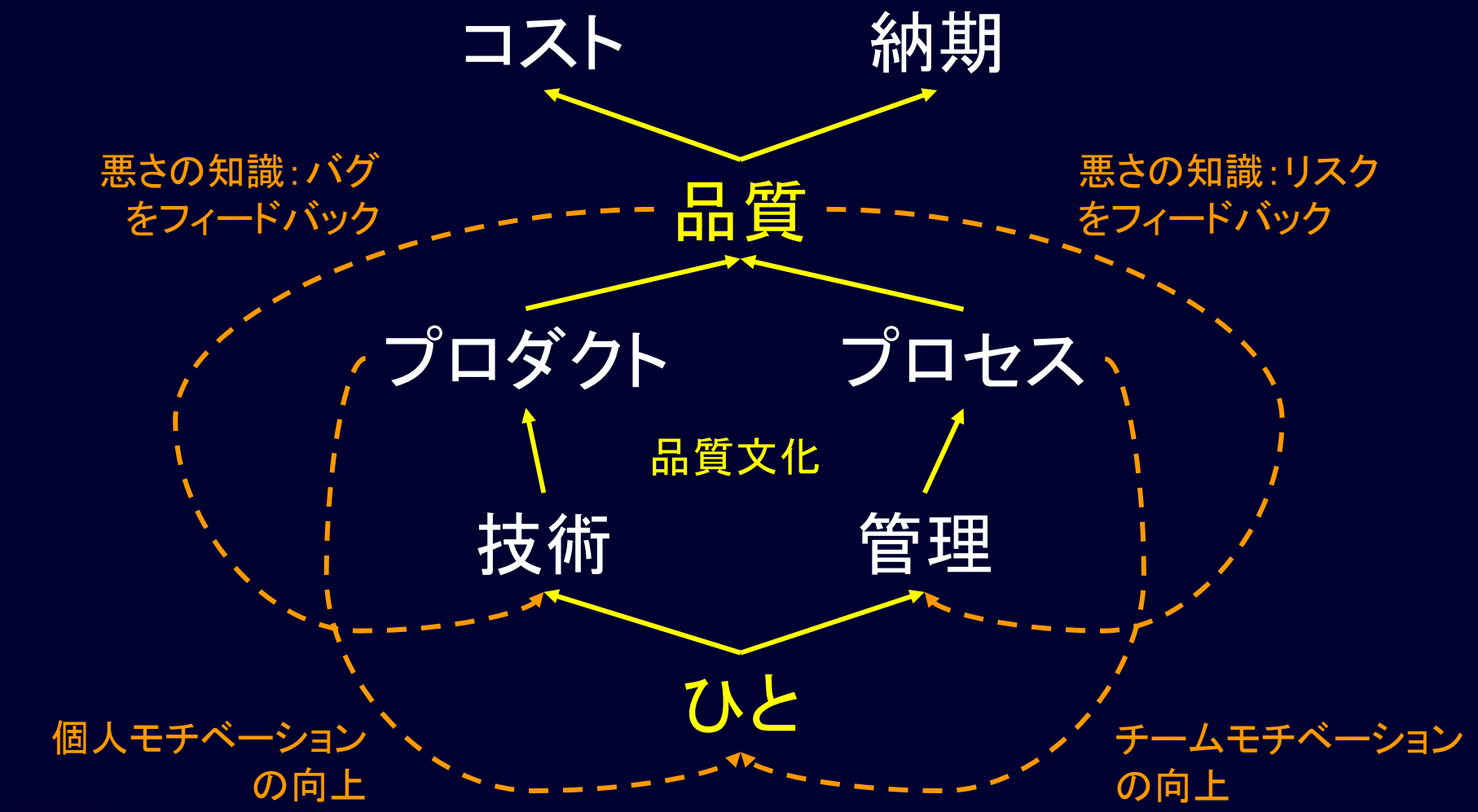
© NISHI, Yasuharu

カイゼンしないソフトウェア組織のイメージ



© NISHI, Yasuharu

カイゼンし続けるソフトウェア組織のイメージ



© NISHI, Yasuharu

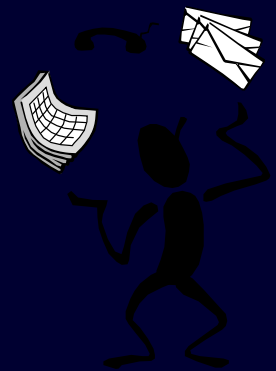
講演の流れ

- ソフトウェアの低品質による経営リスク
- 受託ソフトウェア組織の基本戦略：高カイゼン戦略
- （少し進んだ）組込みソフトウェア開発現場の悩み
- ソフトウェア品質に対する日本的な考え方
- カイゼンし続けるソフトウェア組織のイメージ
- ソフトウェア品質保証技術の発展
- 開発現場を「融合」しよう
- Wモデル：開発者が「最もよいやり方で考える」ことに近づけていくプロセス
- テスト「道」

© NISHI, Yasuharu

ソフトウェア品質保証技術の発展

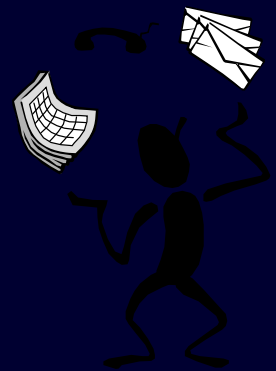
- ソフトウェア品質保証技術とは
  - お客様に安心して使って頂けるような製品を提供するためのすべての活動
    - » プロダクトとプロセスが特定された要求に合致しているかどうかという十分な確信を提供する、といったたぐい活動ではない
- ハードウェアの品質保証技術（品質管理：QC）の発展
  - SQC: Statistical Quality Control / 統計的品質管理
    - » 工場での大量生産のバラツキを減らすための活動群（統計が必須ではない）
  - TQC/TQM: Total Quality Control (Management) / 全社的品質管理
    - » 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



© NISHI, Yasuharu

ソフトウェア品質保証技術の発展

- 技術の発展の順序：癒着→剥離→融合
  - 癒着期
    - » ごく少数の開発者だけが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
  - 剥離期
    - » 技術を発展させるため、技術ごとに専門分化する
    - » 分割統治原則に従うので、技術の体系化やプロセスの整備が容易になる
    - » 安直な分割統治を行うと分割された個々の技術しか目に入らなくなるため、全体像を見失いやすく、縦割り化・硬直化・官僚化が起きやすい
  - 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
      - ・ 融合のコンセプトが重要になる
    - » 技術を融合することで、余計な工数をかけずに見通しの良い開発を行う



© NISHI, Yasuharu

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

© NISHI, Yasuharu

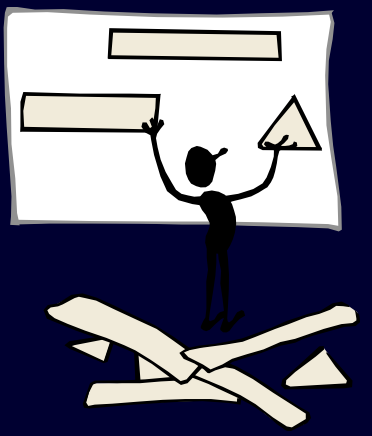
ソフトウェア品質保証技術の発展：剥離期

- 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

© NISHI, Yasuharu

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



© NISHI, Yasuharu

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
    - » 外部の技術や力やツールで最初から完璧なモノを作ろうとする
    - » 万能のレビューやテストの方法があると思いこむ
  - 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
  - どんな要求品質なのか、どんな分析・設計・実装の品質なのか、
どんなテストを設計しているのか、どんなマネジメントなのか、
などを考えずに品質指標を決めようとする

© NISHI, Yasuharu

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だと思っている
    - » 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

© NISHI, Yasuharu

剥離期の弊害

- 取り組み
  - 何のために測ろうかと考えずに、まず測ってみようとする
    - » 現場の問題を見ずにデータだけで語ろうとし、現場に負担がかかりアレルギーとなる
  - 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

© NISHI, Yasuharu

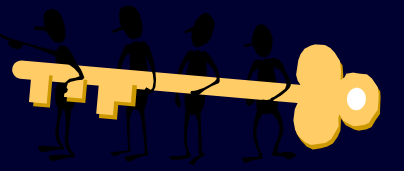
講演の流れ

- ソフトウェアの低品質による経営リスク
- 受託ソフトウェア組織の基本戦略：高カイゼン戦略
- （少し進んだ）組込みソフトウェア開発現場の悩み
- ソフトウェア品質に対する日本的な考え方
- カイゼンし続けるソフトウェア組織のイメージ
- ソフトウェア品質保証技術の発展
- 開発現場を「融合」しよう
- Wモデル：開発者が「最もよいやり方で考える」ことに近づけていくプロセス
- テスト「道」

© NISHI, Yasuharu

開発現場を「融合」しよう

- 悪しき受託根性から、組織の融合へ
  - お客様、上司、部下、パートナーを先読みし、どんどん改善していく
  - 社内の基準やプロセスを盲信せず、基準やプロセスそのものを改善していく
- 品質とコスト・納期のトレードオフから、目標の融合へ
  - 品質とコスト・納期との関係をトレードオフではなく、
    全て同時に達成すべき目標だと再認識する
    - » 品質を上げようとするとコストが下がり納期が短くなる、と体得する
  - 品質とコスト・納期とを同時に達成するには「技術力」が重要であり、
    そのためにはきちんとした技術投資を継続的に行う必要がある
    - » 現場が自分達で手戻りやムダ作業を減らせるよう投資すれば、
      目標を全て同時に達成できるだけでなく、中長期的にどんどん現場がラクになっていく
- 安直な分割統治から、技術の融合へ
  - 技術は一つであるということを理解し、
    異なるように見える技術を結びつける「のり」となる技術を鍛える
    - » グレーボックステスト、不具合分析、レビュー
    - » Wモデル、他工程配慮



© NISHI, Yasuharu

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見通しの良い開発を行う
  - フィードバック・フィードフォワード（悪さの見通し）
    - » 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（由来と行く末の見通し）
    - » 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

© NISHI, Yasuharu

融合期に目指す姿：日本的品質管理

- 標準化・仕組みづくり
  - 標準は常にベストプラクティスであり、明日には超える対象である
    - » 守る標準ではなく攻める標準、従う標準ではなく乗り越えるための標準
    - » 標準は現場自身、もしくは現場以上に現場を分かっている人が発信する
  - 品質をプロセスで作り込む
    - » プロセスの質を向上させることが
      どう品質の向上につながるかを考え抜き、観測し、フィードバックする
    - » プロセス標準の遵守だけをチェックしても、形骸化するだけで何の意味もない
  - フィードバックを軸に上流で品質を作り込む
    - » 失敗を早期に発見して迅速かつ適格な原因分析と対処を行い、
      フィードバックするとともに水平展開して技術力を向上し
      上流の弱点を潰し続けることで、上流をカイゼンする
    - » フィードバック無しで上流を改善しようとしても上手くいかない
  - 失敗による学習を推奨するが、失敗の繰り返しは許容しない
    - » 学習：PDCA，改善，本質把握
    - » マネジメントシステムが自身を改善するためのサブシステムを
      有したシステムになっている

© NISHI, Yasuharu

融合期に目指す姿:日本的品質管理

- 組織と文化、考え方
  - 品質管理は全員参加である
    - » 全員参加により品質文化を醸成し、前向きなモチベーションを全員が持つ
      - ・ 失敗した「ひと」を責めない
    - » 「次工程はお客様」の思想によって、全体最適な組織行動様式を備える
      - ・ 他人をフォローする、よってたかってレビューする、部下を指導する
    - » 常に開発作業そのものが教育になり育成が可能になる
      - ・ 全員管理者、全員部下、情報共有、自律、先読み、
        上司が一格落とす覚悟、ガマン、目配り
  - 組織トップが品質にコミットメントし、品質第一の施策を採る
    - » コストダウン・納期低減は品質向上の後に行う

© NISHI, Yasuharu

融合期に目指す姿：日本的品質管理

- 組織と文化、考え方
  - 改革は改善の先にあると知っている
    - » 長期的なカイゼンのプランを持っているからこそ、方向性の異なる技術に投資できる
    - » 改革「案」は一部の天才だけでも作れる
    - » 改革は、全員が進歩しようという気概が無ければタネは見つからないし所詮浸透しない
  - 長期的で大局的、しかし泥臭く愚直に施策を実践する
    - » 良い家族型経営になっている
    - » 5ゲン主義（現地・現物・現実＋原理・原則）を常に地でいく
  - 現象に囚われず問題を見る
    - » 見通しを良くすることで複雑さに対処する

© NISHI, Yasuharu

融合期に目指す姿:日本的品質管理

- 取り組み
  - データ「で」語っている
    - » 現場の問題を見てデータで語り、事実を踏まえて
      合理的に議論することにより、適切な施策を打ち、個人を責めない
    - » 個人差や能力、「ついうっかり」に帰着させてはいけない
  - トラブルの予測と予防を行うことで問題を発生させず、
    常に前向きの仕事をする
    - » 応急処置ではなく、再発防止でも満足せず、未然防止を目指す
      - ある敏腕なマネジャーの言葉:「問題はカイゼンできて当たり前。
        どう未然防止するかがマネジャーの腕の見せ所であり醍醐味である」

© NISHI, Yasuharu

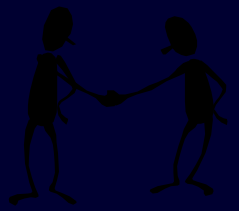
融合期に目指す姿：日本的品質管理

- 取り組み
  - 重点指向でまずやってみるという組織行動様式になっている
    - » 60点でよいからまず始め、少しでよいから成果を出し、カイゼンすることでゴールに近づく習慣をつける
  - 不具合分析を軸にして、開発とレビュー・テストとカイゼン・ノウハウ蓄積とOJTを同時に実施する
    - » 別々に行うと「本業が忙しい」と言い訳をするようになる
  - 「ひと」を中心に据えて取り組みを整理し、実践している
    - » 小集団活動は全員参加によるカイゼンの場であり、グループワークによる相互啓発の場でもある
      - 問題解決や困難直面に対する姿勢・立ち居振る舞いの伝授
      - 部課長層の理解，指導，支援が重要
  - カイゼンは技術力が向上し現場に余裕をもたらす楽しいプラクティスである、と思っている

© NISHI, Yasuharu

融合期に目指す姿：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
- 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たい
  - 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



© NISHI, Yasuharu

他工程配慮を業務に埋め込む

- 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
  - 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

© NISHI, Yasuharu

自工程完結型開発の達成へのステップ

- 組込みソフトウェア開発における自工程完結の問題
  - トラブルの原因工程が特定できていないため、改善が進まない
  - 自工程改善のやり方が分からないため、改善が進まない
  - 「きちんとやらない方がよい圧力」が働いたり、
    他工程から測定結果や悪さの情報をもらっていないため、改善が進まない
  - 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ので、未然防止できない
- 自工程完結型開発の達成へのステップ
  - トラブル分析による原因工程の特定の仕組みの構築
    - » 質のよい「なぜなぜ分析」の体系化が必要となる
  - トラブル原因工程内での改善（自工程改善）の仕組みの構築
  - トラブル原因工程外からの改善（他工程配慮）の仕組みの構築
    - » Wモデルなど
  - トラブル防止のための考慮点の
    工程俯瞰図（ソフトウェア版QC工程表）の作成

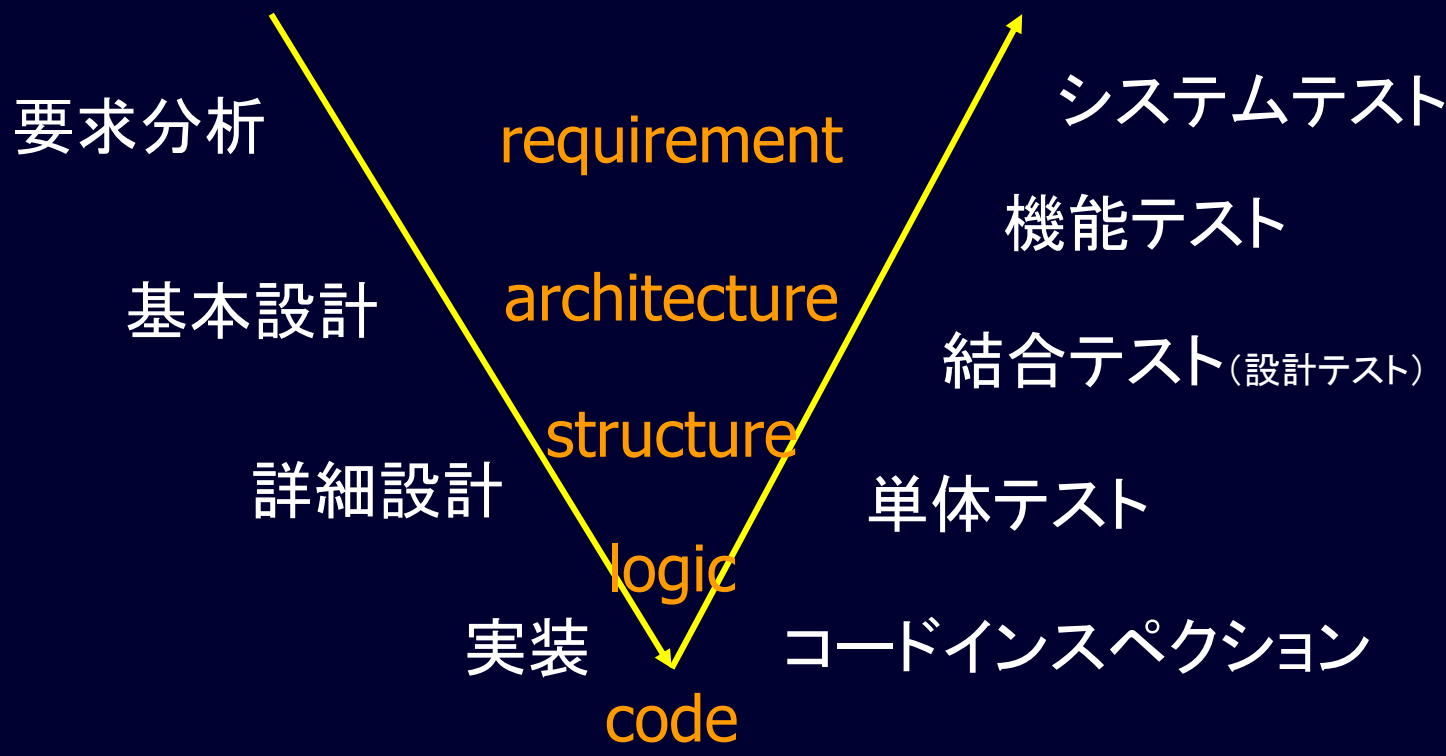
© NISHI, Yasuharu

講演の流れ

- ソフトウェアの低品質による経営リスク
- 受託ソフトウェア組織の基本戦略：高カイゼン戦略
- （少し進んだ）組込みソフトウェア開発現場の悩み
- ソフトウェア品質に対する日本的な考え方
- カイゼンし続けるソフトウェア組織のイメージ
- ソフトウェア品質保証技術の発展
- 開発現場を「融合」しよう
- Wモデル：開発者が「最もよいやり方で考える」ことに近づけていくプロセス
- テスト「道」

© NISHI, Yasuharu

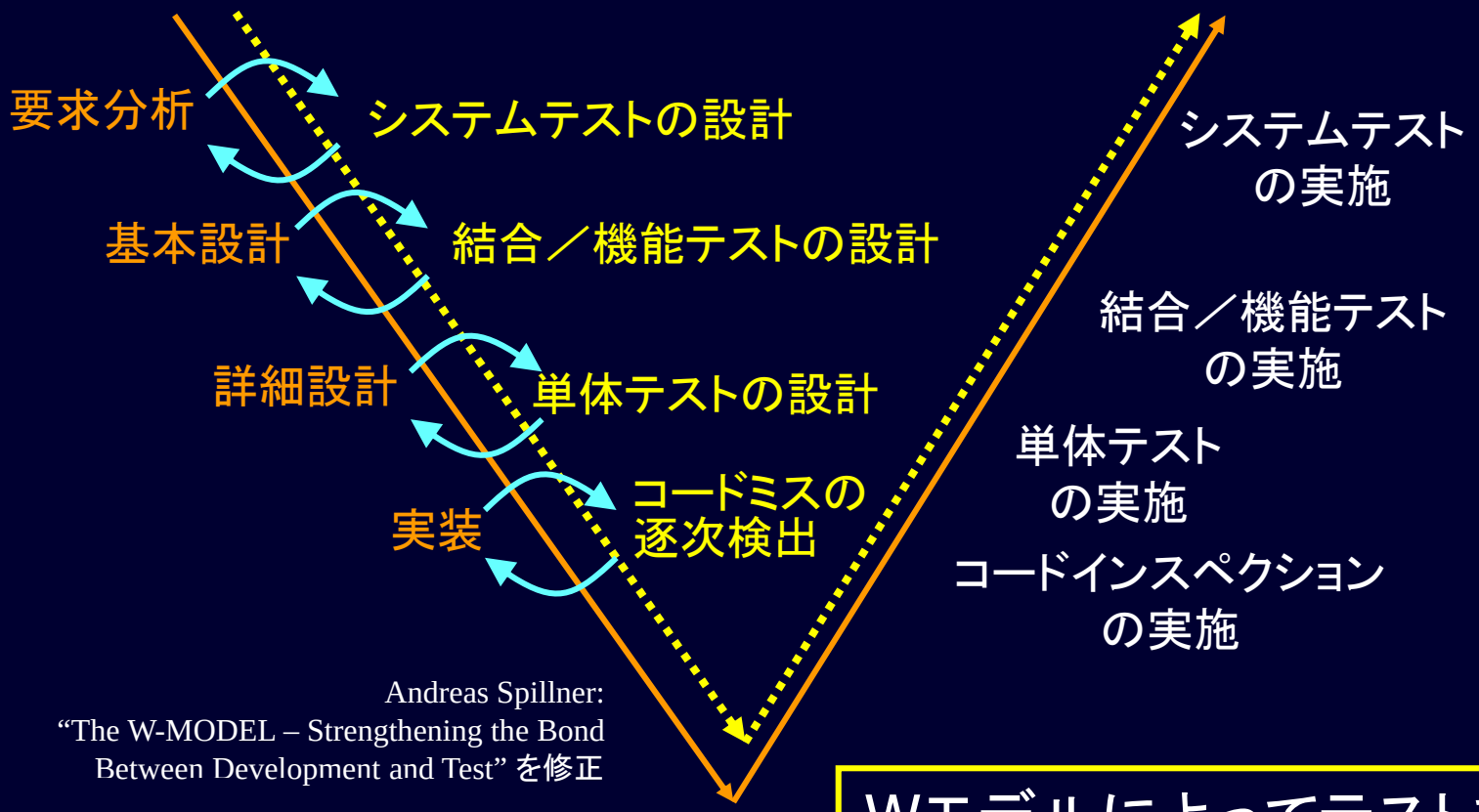
分割統治的プロセスの例：Vモデル



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

© NISHI, Yasuharu

他工程配慮的プロセスの例：Wモデル



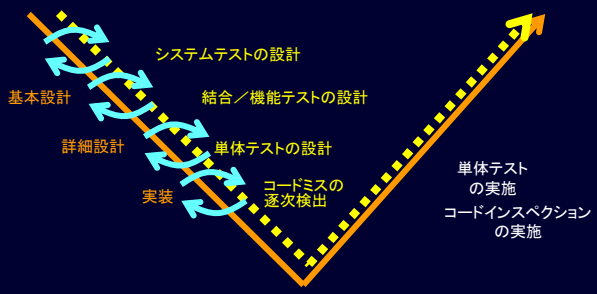
Andreas Spillner:
"The W-MODEL – Strengthening the Bond Between Development and Test" を修正

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

© NISHI, Yasuharu

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



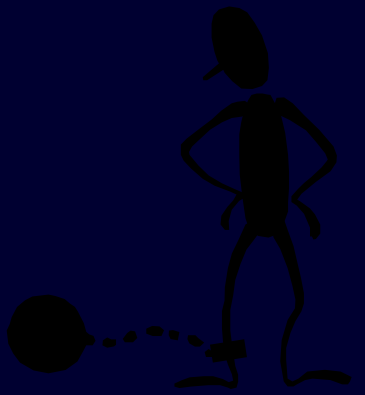
- 進んだテスト技術
  - 組み合わせテスト設計、Critical Combination Analysis、禁則分析
  - テスト観点の抽出・整理・体系化
  - 不具合モード分析・後工程不具合リスク分析
  - リスクベースドテスト
  - グレーボックステスト
  - 探索型テスト
- 開発全般に関わるテスト技術
  - テスト容易性設計
  - 出荷判定基準の明確化

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

© NISHI, Yasuharu

クラシック(古典的だが基本的)なテストのパラダイム

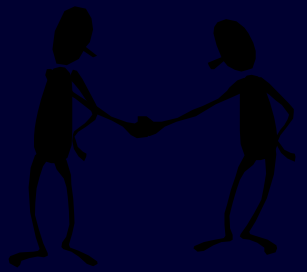
- Vモデル
  - 単体テスト／結合テスト／機能テスト／システムテスト／受け入れテスト
  - 回帰テスト
- ホワイトボックステスト／ブラックボックステスト
  - ホワイトボックステスト＝ソースコードテスト≒制御パステスト(パスカバレッジ)
  - ブラックボックステスト＝外部仕様テスト
    ≒機能網羅テスト＋境界値テスト＋デシジョンテーブルテスト
- システムテストカテゴリ
  - ボリュームテスト／ストレージテスト／高頻度テスト／ロングランテスト
  - 構成テスト／両立性テスト／データ互換性テスト
  - 操作性テスト／セキュリティテスト・・・
- プロセス無しのテスト
  - 線表引きとテスト項目作成とテスト実施と不具合報告
  - 信頼度成長曲線



© NISHI, Yasuharu

Wモデル：上流とテストが協調し、上流を改善する

- 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



© NISHI, Yasuharu

Wモデルにはモダンなソフトウェアテストが必要

- クラシックなソフトウェアテストのパラダイム
  - テストは実施するもので、検算のようなものである。だから簡単だ。
  - テストはテストケースを実施してバグを見つけるものである。
  - テストはバグを見つけるものだ。だから仕様への適合が第一だ。
  - テストはソースコードベースのホワイトボックスと
    ユーザ・仕様ベースのブラックボックスの2つがある。
  - テストは下流でバグを見つける活動である
  - テストケース数は爆発するので網羅できない

- モダンなソフトウェアテストのパラダイム
  - テストは設計するもので、分析も必要である。だから難しくクリエイティブだ。
  - テストは実施でもバグを見つけるが、設計の過程でもバグを見つける。
  - テストは製品・システムの総合的な品質を評価するものである。
    だから「よいシステムとは何か」を分析することが第一だ。
  - ホワイトとブラックを上手く融合したグレーボックステストが重要だ。
    そしてコードやユーザ、仕様、バグなどのテスト観点の考慮が重要だ。
  - テストは上流から開発と並行してバグを見つけていき、バグを予防する活動である
  - テストケース数が爆発しないよう、どこまでどう網羅するかを検討するのが重要だ

© NISHI, Yasuharu

Wモデルの効果：タイプ1

- 単純に作業順序を入れ替えるだけだが、
  上流でテストを設計することで細かく考えることができ、
  抜け漏れ矛盾を減らすことができる
  - 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
    - » 開発者は開発時に主要な仕様や設計を考えるに留まり、
      実は細かくは考えていないことが多い
    - » 仕様や設計が複数の組織やバージョンに由来し、整合性を取れていないことがある
    - » テストに頼る風土の組織では、一貫性すらきちんと確認していないことがある
- テスト設計の質を少しだけ向上することができる
  - 仕様や設計を行った直後にテストを設計するような手順にすると、
    極めて単純なテストの抜け漏れだけは防ぐことができる
    - » テストの抜け漏れが何でも防げるわけではない

© NISHI, Yasuharu

Wモデルの効果:タイプ1の注意点

- テスト設計をきちんと行っていない組織では効果が出ない
  - 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
    - » 何をどれくらい網羅し、どういう理由でどのくらい間引くかを
      明示していない組織では、抜け漏れ矛盾を見つけることができない
- レベル1に留まったままだと逆に問題が起きやすい
  - 開発上流で膨大なテストケースを記述することになるため、
    仕様変更・設計変更時に膨大なテストケースの書き直し作業が必要となり、
    工数超過を起こしてしまう
    - » 仕様の書き方をそもそも改善しないといけない
    - » テストケース生成ツールで自動生成させたくなるが、
      自動生成させると抜け漏れ矛盾が見つかりにくくなる
    - » 結局のところ「テスト設計とは何か」をきちんと考えないといけない

© NISHI, Yasuharu

Wモデルの効果：タイプ2

- 開発組織がいま持っている観点できれいに作る（汚く作らせまいとする）よう圧力をかけることができる
  - 開発者は「もっとすっきりさせたいが時間がない」というジレンマを抱えている
    - » 上流で（組み合わせ）テスト項目数を概算することで「もっとすっきりさせないと、テストで膨大な時間がかかる」と圧力をかけることができる
  - （組み合わせ）テスト設計を上流で行うことで、仕様や設計、コードがすっきりし、不具合を減らすことができる
    - » 組み合わせテスト設計で結合度の低下や依存関係の存在をきちんと把握するので開発時に予期していない副作用（組み合わせ不具合の芽）を防ぐことができる
    - » 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
- テスト設計の質を向上することができる
  - すっきり作ってあれば依存関係を追いやすく例外事項が少ないため、工数を増やさずに抜け漏れを少なくできる

© NISHI, Yasuharu

Wモデルの効果：タイプ3

- 開発組織がいま持っていないような多面的な観点から気をつけて作ることができる
  - 開発者はあらゆる観点から検討しているようで、採用した開発のやり方に固有の観点抜けが出てしまう
    - » 例）制御系の開発では状態遷移の検討の抜けが発生しやすい
  - テスト設計の際に観点を俯瞰的・多面的に列挙し整理することで、開発者が気づいていない観点を示唆することができる
    - » テスト技術者は過去の不具合の分析などにより、抜けやすい観点を知っている場合がある
    - » 開発者は開発対象に必要な観点を絞り込むように頭を使うが、テスト技術者はテスト対象を俯瞰的・多面的に捉えようとする
  - 必ずしもテストケースを詳細に記述する必要はない
- テスト設計の質を向上することができる
  - 開発時に検討した観点をテストで共有することで、特に内部構造に関する

© NISHI, Yasuharu

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

© NISHI, Yasuharu

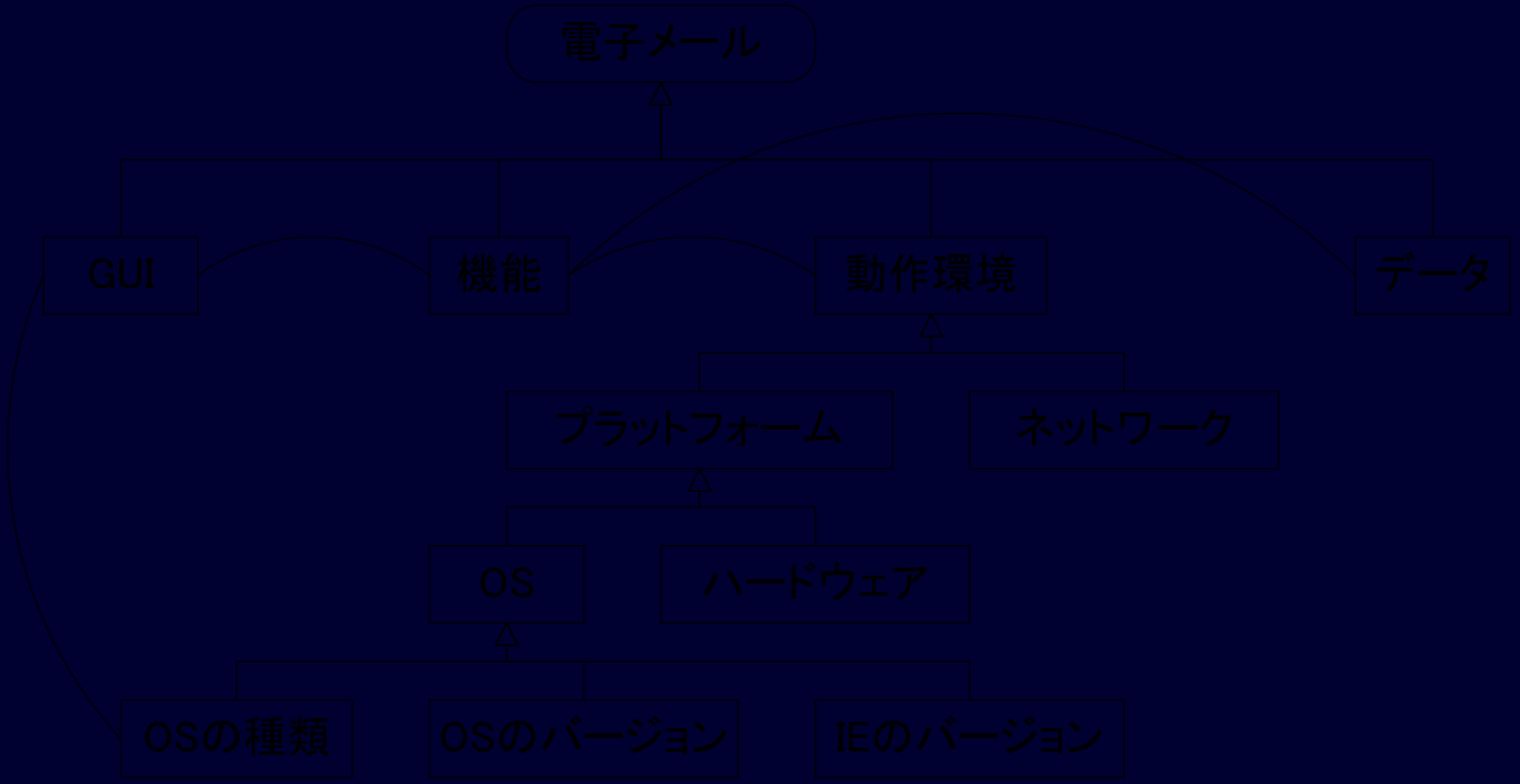
組込みのテストの設計で考慮すべき観点の例

- 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

© NISHI, Yasuharu

テスト観点モデルの例



© NISHI, Yasuharu

テスト開発ライフサイクルは進化しつつある

テスト設計（or テスト計画 or テスト実施準備）

テスト実施

テスト項目の抽出

テスト
要求分析

テスト要求の
獲得と整理・
テスト要求
モデリング

テスト
アーキテクチャ
設計

テストアーキテクチャ
モデルの検討

テスト
詳細設計

テスト技法の
適用による
テストケースの
列挙

テスト
実装

具体的な
テスト手順・
テストスクリプトの
記述

テスト
実施

© NISHI, Yasuharu

Wモデルの効果：タイプ4

- 開発組織が忘れがちな、要求分析・設計・実装工程の「由来」と「行く末」を常に意識し、両者の乖離を防ぐことができる
  - 開発者は意外に「なぜそういう設計をしているのか」や「こういう設計変更を行うとシステム全体としてはどういう振る舞いをするのか」などを考えずに開発しているため、「由来」と「行く末」が乖離してしまう
  - テスト設計の際に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、などを常に考えながら作るようになる
  - テスト設計の際に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、などを常に考えながら作るようになる
  - 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
- テスト設計の質を向上することができる
  - テスト目的を常に意識するため、心配だからという理由のテストが減っていく

© NISHI, Yasuharu

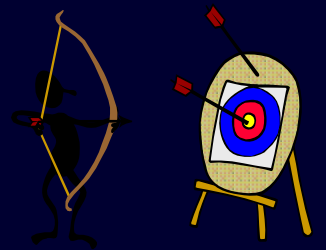
Wモデルの効果：タイプ5

- 開発組織の持っている弱点を明確にし防ぐことができる
  - 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
- テスト設計の質を向上することができる
  - 不具合が作り込まれている確率の高いところからテストすることができる

© NISHI, Yasuharu

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
    - » 当該工程だけでなく、後工程でバグを作り込む原因となりうる部分もパターン化する
  - 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



© NISHI, Yasuharu

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

© NISHI, Yasuharu

Wモデルの進化：フィードバック時間の減少

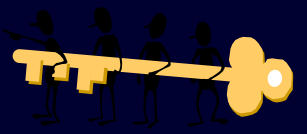
- Wモデルが効果をあげるように
  テスト技術を進化させ前倒しすることで
  開発上流から品質の作り込みを行っていく

1. 開発者が開発（仕様策定、設計、実装）を行った後で、
   すぐにテスト技術者がテスト設計を行いフィードバックする
2. 開発者が開発を行った後で、
   すぐに開発者が自らテスト設計を行いフィードバックする
3. 開発者が開発を行うと同時に、
   テスト技術者が独立にテスト設計を行いフィードバックする
4. 開発者が開発を行うと同時に、
   テスト技術者が開発者とワイガヤしフィードバックしながら
   テスト設計を行う
5. 開発者が開発を行いながら、
   開発者が自らテスト設計を行い両者を同時に改善していく
6. 開発者が開発を行う前にテストについても思索を深め、
   はじめからバグの無い開発を行う

© NISHI, Yasuharu

Wモデルの目指すべき姿

- 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
- 開発者が「最もよいやり方で考える」ことに近づけていくプロセス
  - テストについて深く洞察することで、テストを実施することなく品質を高められるようになる
  - 開発者の気づきのフィードバックサイクルが極限まで小さくなり、開発予測力が高まっていく
    - » リスクベースドテストをきちんと運用できる技術力が備わっていく
  - よい開発をするという点において、開発者とテスト技術者のゴールは同じである



© NISHI, Yasuharu

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

© NISHI, Yasuharu

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

© NISHI, Yasuharu

講演の流れ

- ソフトウェアの低品質による経営リスク
- 受託ソフトウェア組織の基本戦略：高カイゼン戦略
- （少し進んだ）組込みソフトウェア開発現場の悩み
- ソフトウェア品質に対する日本的な考え方
- カイゼンし続けるソフトウェア組織のイメージ
- ソフトウェア品質保証技術の発展
- 開発現場を「融合」しよう
- Wモデル：開発者が「最もよいやり方で考える」ことに近づけていくプロセス
- テスト「道」

© NISHI, Yasuharu

Boris Beizer のテスト「道」

フェーズ0　－　テストはデバッグの一部である

フェーズ1　－　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　－　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　－　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　－　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

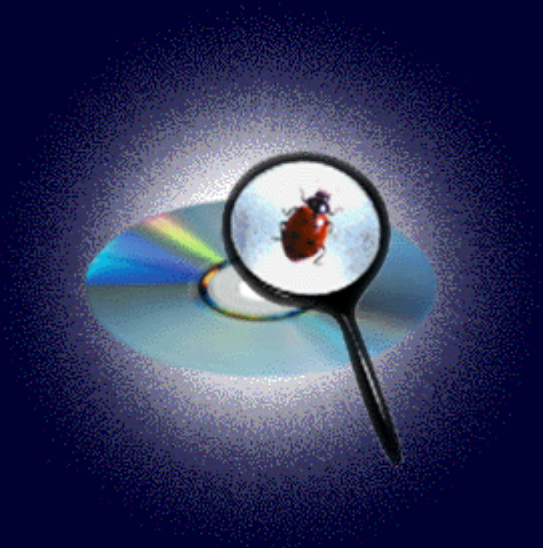
© NISHI, Yasuharu

講演のまとめ

- （受託）ソフトウェア組織にとって
  高カイゼン戦略は最もロバストな経営戦略である
  - しかし（少し進んだ）ソフトウェア開発現場は、
    一生懸命開発しているが、一向によくならないと感じている
- 融合型の組織を目指すことで、状況を打開しよう
  - 品質とコスト・納期のトレードオフから、目標の融合へ
  - 悪しき受託根性から、組織の融合へ
  - 安直な分割統治から、技術の融合へ
- Wモデルを目指してテストを上流と協調させることで、
  仕事をするだけでノウハウが飛び交い
  駆使されどんどん蓄積される
  競争力の高い組織に進化させていこう

© NISHI, Yasuharu

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

© NISHI, Yasuharu