System Creates

DevJTestでQCDの同時達成を目指せ

～時代の変化の中でのテストプロセスも変えよう～

JaSST’13 Tokai基調講演

2013/11/15

株式会社 システムクリエイツ
代表取締役 清 水　吉 男
http://homepage3.nifty.com/koha_hp
shimz@nifty.com
派生開発推進協議会　代表
http://www.xddp.jp

本日のお話しの内容

1. テストとプロセス改善の歴史
2. テストでできること／できないこと
3. バグを見つけるよりも減らせ
4. プロセス改善の勘違い？
5. 自分の人生を生きる

XDDP：eXtreme Derivative Development Process の略．派生開発に対応するように開発したプロセスです．（参考文献②）
USDM：Universal Specification Describing Manner の略．要求仕様がモレない表現方法として開発したものです．（参考文献①）
PFD　：Process Flow Diagram の略．合理的なプロセスを設計するためのツールでDFDをアレンジしたものです。

日本での

1. テストとプロセス改善の歴史

時代の要請

間違ったプロセス改善への取り組み

テスト作業の時代的変化

時代は「QCDの同時達成」を求めている

- 「QCD」を達成する方法・技術は既に習得したはず
  - Q・・・1980年代に確立した（はず）
  - D・・・1990年代前半に競った
  - C・・・1990年代後半に競った

2000年・・・「QCD」の同時達成の方法を競う時代へ

でも実現していない！

- 「Faster, Cheaper, Worse」（確かに早いし安い、でも品質は悪い）
  - 2008年、4WCSQでのハンフリー氏の講演テーマ

ビジネスの勝敗が「ソフトウェア」で決まる時代

- 『ソフトウェアでビジネスに勝つ』(W. ハンフリー)
  - ソフトウェアが他社との違いや製品の価値を作る
  - 同じ機能でも、その見せ方や使い勝手などで“価値”を高める

- 機器の機能を動かすだけのソフトに“価値”はない
  - H／W部品のほとんどは調達品
  - デジタル時代にあって機能の優位性は長続きしない

QCD+「α」で競争に勝つ

新しい時代の要請

GEの方針転換

GEは2013年5月29日にカリフォルニアで開かれた「D11カンファレンス」「かつてGEはソフトウェアは他社に任せて、自社はハードの世界に集中すべきと考えていたが、方針を変えた」（日経ビジネス2013.7.1号より引用）

- 「GEの方針転換」から何を読み取るか
  - ビジネスの勝敗はソフトウェアで決まることに気付いている
  - 製品に「価値」を付けるのはソフトウェアであることに気付いている
  - 航空機のエンジンであっても、今日、適切な製造プロセスと指導があれば、どこでも作れる　→安く作れるところで作る

何のためにプロセスを改善するの？

① バグなどの混乱を鎮めて品質と生産性を向上させるため

- 無秩序なソフトウェア開発では品質は確保できない

② 市場の要求（QCD）の変化に対応するため

- 市場の要求が変化する背景・・・
  - 競争している企業の勝つための仕掛け
  - 生活様式の変化などによる顧客（ユーザー）の嗜好性の変化
  - ベースとなる技術の進歩
  - 市場の変化を読み取って新しい設計技術が提供される
  - 社会の状況変化を受けて新しい要求が発信されることがある

世の中は無常

プロセスを変化させて対応するはずが・・・ ??

QCDに対する競争の始まり

- 1970～80年代・・「品質」に対して強い要求
  - 当時、日本製品の品質に対して世界は脅威を感じた
  - ただし、後工程の「テスト（試験）」によって獲得したもの
    - 当時のソフトの規模は小さい

品質は「プロセス」で確保する！

「Quality is Free」（フィリップ・クロスビー）

- 「最初から正しく仕事をすれば高品質は確保できる」
- 品質コスト（予防コスト＋評価コスト＋失敗コスト）
  ＝ものごとを正しく行わないことの代償

- プロセスで品質を確保するためにいろいろな方法が考案された
  ISO、6Σ、CMM（後にCMMI） ・・・

プロセスで「品質」を確保する方法

- 「品質」と「生産性」を同時に確保する**「5つの普遍的原則」**
  - 1980年代に考案、当時は「DFD」を使っていた

① 今回の要求を実現できる合理的な開発アプローチを設計できる
② プロセスを適切に実行できるソフトウェアエンジニアリングの技術を持つ
③ 決めたプロセスを整然と実行できる（シミュレーション）
④ 途中の状況の変化に対してプロセスを変化させて「代案」を考え出す技術
⑤ バグなどから原因プロセスを改善する技術

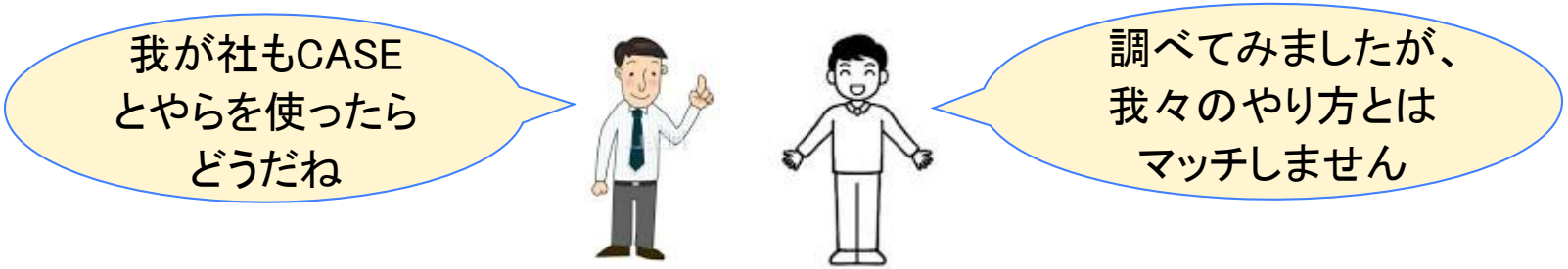
QCDの向上 ⇒ 新しい方法 / 新しいツール ＋ プロセスを工夫することで納期やコストの要求にも対応できる

QCDの要求に対する日本の対応 (1)

プロセスで品質を確保する方法が前提

1990年代に入って「納期」と「コスト」の競争が始まる

- 90年代前半：納期の短縮要求が追加
  - 世界・・・期間の短縮を支援するツールが開発された
    - CASE, EA, MS-Project, ソースコードの自動生成などが出てきた



  - 日本・・・人海戦術で対応して一人の分担を狭くした
    - 当時、バブルで新規採用が膨れていた

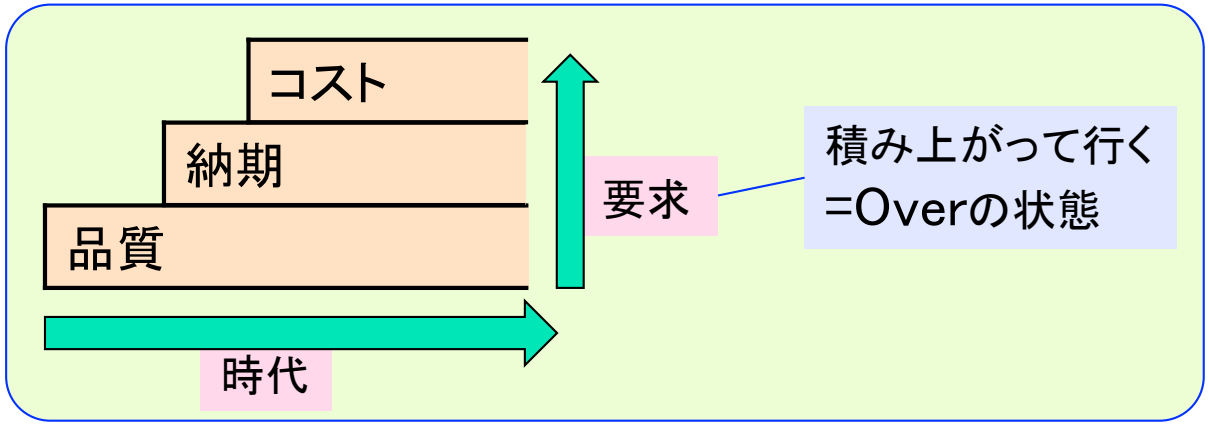
取り返しのつかない
状態へ

QCDの要求に対する日本の対応 (2)

- 90年代後半：コストの削減要求が追加
  - 製造部門と一緒に「オフショア」へシフト
    - 国内のソフト会社もオフショア先とコストの叩き合いにあって疲弊
  - 要員は残ったままで「オフショア」の中継作業を担当
    - 入社以来一度も「設計」をしていない

品質や納期は確保できたのか？

この後、コストを圧迫する要因になる



- 2000年代：QCDの同時達成要求
  - さらに「QCD ＋ α」が求められている

どうやって対応するつもりか？

テスト作業の時代的変化

- 小規模で設計者がテストをした時代
  - 自分がテストしやすいように設計する ➡ 作り方の工夫へ
- 規模の増大に伴って専門職化
  - 大量採用も役割の分化に拍車
  - 入社時に「コース」分け
    - 設計者はテストのことを知らない
    - テスト技術者は設計のことを知らない

➡ 無駄な作業が増える

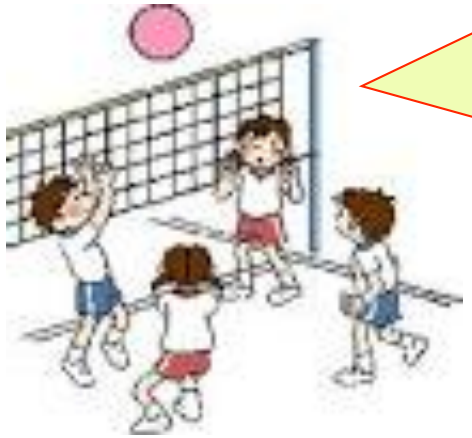
H／W設計者	プログラムがないと動かない 動作テストをソフトウェア技術者に依存
ソフトウェア設計者	機能を実現することしか考えていない テストのやりやすさを考慮していない
テスト技術者	作り方が読めないので設計者に注文できない バグを逃がさないためにテストの「網」を強化

誰も自立しない！

2. テストでできること／できないこと

なぜ、品質と生産性は相反するのか？

派生開発のテストは？



- 仕切りが「ネット」なら、相手の動きを読んで少人数で対応できる。
- 仕切りが「ベニヤ板」なら、どこから飛んできても対応できるように、大勢で備えるしかない。

品質と生産性が相反してしまう？

- 通常のテストで検出されるのは動作させた結果のバグ
  - 要求仕様との食い違い
  - 要求仕様には書かれていないが常識的な間違い

要求仕様をベースにテストケースが作られていることが多い

このテストの網では、
川の水に混じる「ゴミ」が拾えただけ

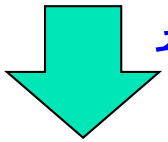
バグの検出漏れ発生 → テストの強化

テストの網を強化すれば、拾える「ゴミ」が増えるが・・・

- バグそのものが減らなければ、テスト工数が増え続ける
- 逆にソースコードの劣化が進行して、バグが発生しやすくなる

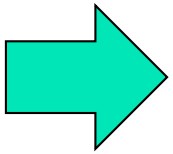
派生開発ではテストも混乱

- 派生開発の特徴
  - 競争に勝つための機能追加と仕様変更が行われる
  - 変更に伴って思わぬところで影響がでる・・・デグレード
- 工数が足りない
  - 派生開発に適さないプロセスでテストの期間を使ってしまう



その結果

- バグが残ったままリリースされる



市場から拒否される可能性もある

テスト作業の強化へ

工夫もなく闇雲なテストへ

- 新規開発 ▶ 要求仕様書に基づいてテスト仕様を作った
- 派生開発 ▶ テスト仕様を改訂するには変更箇所の情報だけでは不足

- 変更箇所に関する情報がないため、テストの範囲を絞り込めない
  - どこをどのように変更したのか？
  - どの機能と関係しているのか？
  - 変更したのはどのような作り方をされた箇所か？

情報がないのか、
情報を入手する
工夫がないのか？

- その結果、テストの範囲を必要以上に広げるしかない
  - 人海戦術で日程の不足を補う

コストの増加

それでQCDの同時達成は？

- テストでいくらバグを見つけても、根本的にバグが作り込まれないように改善できていない
  - 同じようなバグを繰り返す
  - テストの網が強化されて工数が増える
  - 市場はQCDの達成を要求しているが、満たされない

プロセス改善の目的ではなかったのか？

私、作る人 ⟷ 私、テストする人

お互いに距離が開いている状態ではいつまでたっても解決しない

思慮もなく作られたプログラムに紛れ込んだバグを、期限内にすべて見つける方法はあるか？

Over

3. バグを見つけるよりも バグを減らせ

設計技術者と連携しよう

相互の情報共有

QCDの同時達成を目指せ

Dev Join to Test

たくさんバグを見つけた人が偉いのか？

テスト技術者は、テストで**バグを見つける**ことは第1の仕事

- バグをたくさん発見した人が評価される？
  - だったら、テスト情報を秘密にする！

得意気になっているのは部門長だけ？

ユーザー（顧客）が求めていることか？

- バグの検出件数をノルマにしたら何が起きるか？

バグを作り込まないために

- バグの作り込みを繰り返さないように誘導する方がよい
  - バグのパターンを公開する
  - 作り込まない方法を指導する
  - テスト情報を公開する

QCDの同時達成に繋がる！

これを評価する組織文化に変える必要がある

フロントローディングへ

- 川に流れ込む水を、その発生源で対応する
  - 家庭からの排水
  - 工場からの排水
  - 農業での使用制限

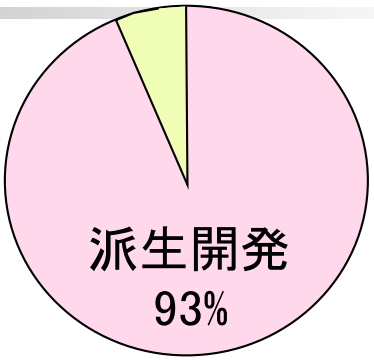
ゴミや汚染物質の種類が異なる

それぞれの場面で対応した方が効率が良い

大部分は派生開発

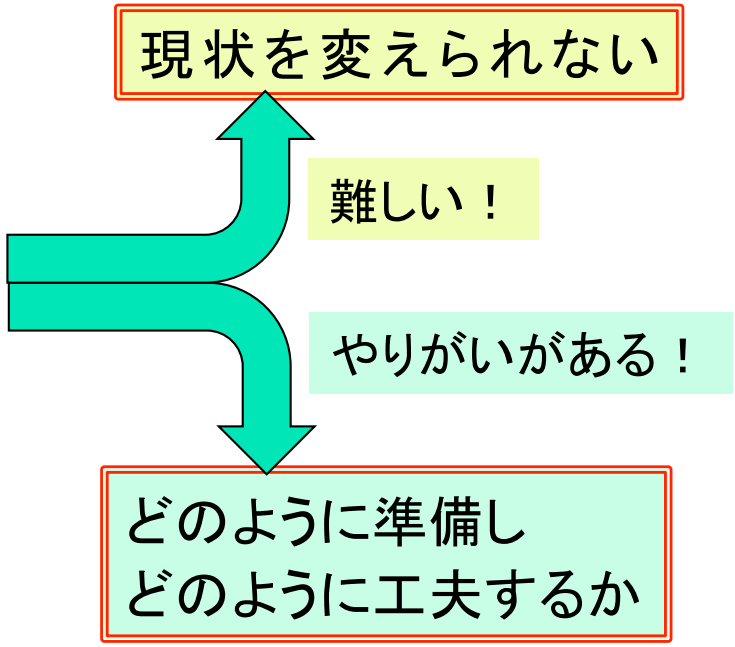
- 今日、開発案件の大部分は派生開発

- 派生開発で開発担当とテスト担当で協調して取り組む価値がある



ある企業の1部門のケース

- 厳しい場面も
  - 納期が短い
  - 変更規模などが多様
  - 「部分理解」の中での作業



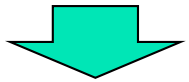
多くの人の問題は、
自分(の可能性)を
信じていないことだ！

「XDDP」は変更の情報が見えている

- 派生開発の最大の特徴
  - 「変更前は問題なく稼働していた」という事実

最初から

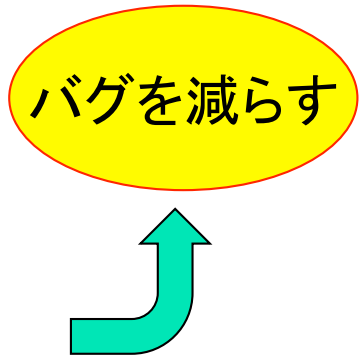
- 「XDDP」は派生開発においてQCDの同時達成が可能な方法として考案したもの
- 変更情報は「before / after」情報として書き出している
  - 変更前のどの部分をどのように変更するのか
  - 「変更3点セット」に最小限の変更情報は見えている
- すべてのバグはこの変更箇所、変更方法に起因する



XDDPに合った効果的なテスト方法を考案する余地がある

「before / afoter」から見えること

- 「before / after」を表現する目的は？
  - 対部分理解・・・・担当者が変更しようと考えた「変化点」を明らかにすることで、回りの人に気付いてもらうことが狙い
  - 表現することで自分でも気付くことがある
- 気付いて欲しいことは？
  - 変更箇所・内容は依頼者の意向と合っているか？
  - その変更の仕方は適切なのか？
  - その変更に関連する仕様を持つ箇所はないのか？
  - その変更で影響がでることはないのか？



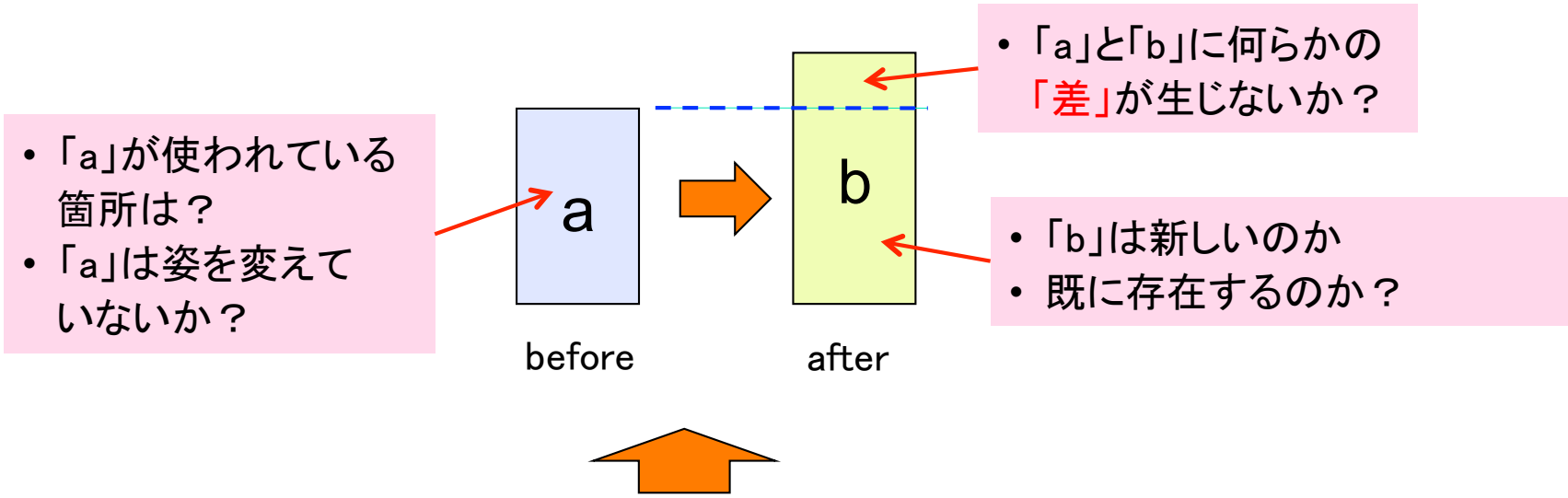
ソースコードの部分を見せられても担当者以外の人には判断できないが、変更の意味や目的を文章で記述されていることでレビューに参加できる

変更で変わる箇所は？

- バグを生むのは・・・
  - ① 「before」から「after」への変化
  - ② 「before」と「after」の「差」

過去のバグから学習できる

依頼・・・「a」の処理を「b」の処理に変更する



変更要求仕様には、担当者が気付いていることが見える

テスト範囲を絞り込む方法は？

- 派生開発では、変更情報からテスト範囲を絞り込むことが重要になる
- 現実には・・・
  - 新規開発時に「派生開発」の場面を想定して設計されていない
  - 派生開発の中で無秩序にソースコードを変更してきた

本来の変更箇所
機能的関連で変更する箇所
作り方の影響で変更する箇所

XDDPの「変更3点セット」で、これらの様子が見える

協同作業

変更箇所とテスト範囲の関係を整理する

テスト範囲を絞り込めるようにプログラムを整理する

派生開発の場から始めよう

- 市場や顧客の要求を反映しながら、製品やシステムをリリースし続ける必要がある

派生開発を制覇して「次」に繋げるしかない

如何にバグを
埋め込まないか

如何にして効率的に
バグを検出するか

4. 新しい取り組みへの注意

プロセス改善の注意

形骸化への対応

バグを減らすプロセス改善へ

- 共通の手順
  - 問題(を示すデータ)の収集と整理
  - 問題が生じた原因プロセスを特定
  - 「課題」の形成
  - 対応策の考案
    - バグを見つける方法
    - バグを埋め込まない方法
  - 実施して効果を確かめる
  - 結果を論文形式にまとめる

「PFD」が有効

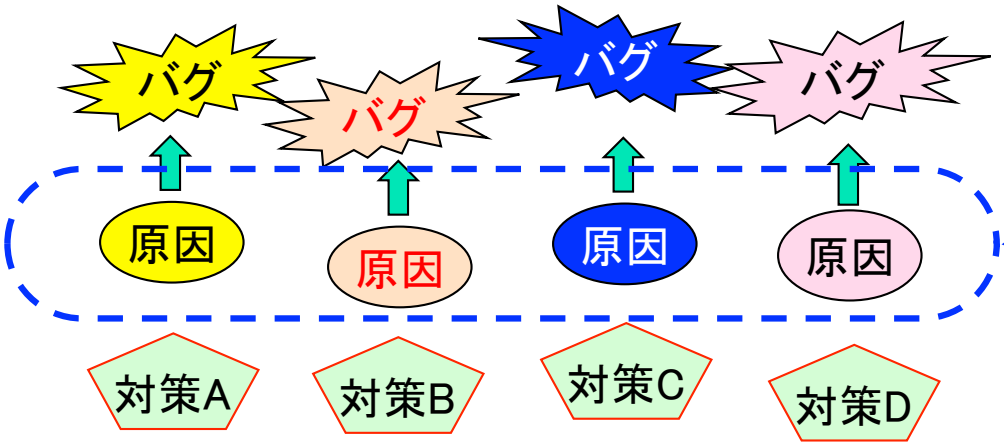
形骸化防止策 ①

新しい方法を導入する際に、それがどの問題を解決するのかを明確にしておく

- 2種類の対応策
  - ① 現状のプロセスや定義書(ガイドラインを含む)を変更(改訂)する
  - ② 新しい方法を導入する

「銀の弾丸」の意味

- 一つの取り組みで「多くのバグを減らす方法」はない
  - それぞれのバグには「それぞれの原因」があり、「それぞれの対策」がある



一般に、まとめて効く対応策はない。
ただし、プロセスに起因する問題は、いくつかまとめて効果を出すことがある。

- 取り組みが行き詰まるもう一つの背景
  - 解決できそうな問題を特定しないまま新しい方法に取り組む
  - 自組織の「強み」と組み合わせていない

取り組みが形骸化する問題

- コンサルティング期間が終わって数年後

〇〇さん、久しぶり！
その後、現場の様子は？

「形骸化」ってどう
いう状態なの？

「before / after」
は表現されている
の？

あの時はありがとうございました。
実は、だんだんと取り組みが
形骸化しているんですよ。
お恥ずかしい話しです

「変更3点セット」は
書けているのです
が・・・

一応は「before」と
「after」で書かれて
います・・・

これが派生開発推進協議会
を作った理由です

Association for Facilitation of Rational Derivational Development
AFFORDD

取り組みが形骸化する理由

- 新しく取り組んだ方法の「本質」を認識していない
  - 「形」から入ることは必ずしも間違ってはいない
  - ただし、その「本質」を理解していないと先祖帰りする
    - 適切に理解できていない場面に遭遇したとき
    - 前後のプロセスと整合性が取れなくなったとき
    - 時間的に急がされたとき
  - 「先祖」＝うまくできていなかった時のやり方 ←ポケットにある

「本質」に気付かないのは、「本質」を考えようとしないから

「本質」を考えよう

形骸化防止策 ②

System Creates

- うまく行かないことがあれば「原典」に戻ろう
  - 今まで気付いていない「何か」に気付く
- 「本質」を考えて、文章に書き出そう
  - なぜ、「理由」を書くのか？
  - なぜ、変更内容を「文章」で書くのか？
  - なぜ、「before / after」を書くのか？
  - なぜ、「3種類」の成果物にするのか？
  - なぜ、ピアレビューで「肯定意見」を出すのか？

自身で納得する

人に説明できる

単に「変更前の状態」と「変更後の状態」を書けばよいわけではない

「理由（ニーズ）」の方は変化しにくい

他には？

手法の背後にある「文化」の違いに注意

- ピアレビューが個人攻撃になってしまう
  - 「肯定意見を出す」ことを勧められていることに気付いていない？
  - 「肯定意見」にどのような意味があるのか？
  - 「レビュー」に対する文化の違いがある？

国によっては先ず「褒める」ことが文化になっているため、
文献には「肯定意見」を出すことの重要性を協調していない

「構造化ウォークスルー」の本には「肯定意見」の記述は「1ヶ所」しかない

なぜ「CMM（TR-24）」のピアレビュー
でこれを推奨しているのか？

効果をデータで裏付ける

形骸化防止策 ③

System Creates

- 取り組み前と取り組みの後の違いを「データ」で示す
- もともと、新しい取り組みをする前に
  - 現状の問題を分析する・・・ここで取り組み前のデータが得られる
  - そこから取り組むべき「課題」を特定する
  - 課題の解決に効果的と思われた方法と取り組み方を検討する
- 実際に取り組んでみて
  - 結果を分析する・・・ここで取り組み後のデータが得られる
  - 効果の検討で「本質」を押さえていたかどうかを検証する
- 結果をまとめておく
  - 何が「本質」なのか・・・・これは外さない！
- 多くの組織では、はじめに「解決策」がある

5. 自分の人生を生きる

テストエンジニアとして、ソフトウェアエンジニアとして
胸を張れる人生を生きよ！

テストは知的作業

- テストエンジニアは知識労働者である
  - 「知識労働者」と「ブルーカラー労働者」の違いを認識する
- 「事実に基づいた推理」を駆使する仕事である
  - 間違いを犯しやすい人がいる
  - 問題を犯しやすい場面がある
  - このバグの背後に何があるのか？
- 間違いを犯すのは「人」の思考であり行動である
  - 行動のパターンがある
  - 「人」を知る
- 他の領域の技法を組み合わせる
  - テスト技術だけでは「人」が見えない

テストエンジニアの知見

身に染みついたプロセス？

肯定眼を身につけよ

- テストエンジニアの利点は？
  - 多くの設計エンジニアの成果物を見る機会がある
- テストエンジニアの陥る危険
  - 人の欠陥ばかりを見るようになる
  - 「ピアレビュー」で「肯定意見」を出すことの本質を知ること

- 50点の成果物でも部分的に「良いところ」はある
  - この設計方法はバグが入り込みにくいのでは？
  - このような仕様の捉え方をすれば隙間が生じにくいのでは？

「肯定眼」でものごとを見続けることで誰にも負けない自分を作る

時代を読む

- 最後に「幸せな人生」だったと言えるように
  - ソフトウェア無しでは成立しない社会
  - 活躍する場はたくさんある
  - ただし、技術レベルなどの条件を満たす必要がある

人の運の善し悪しは、時代に合わせて行動できるか否かにかかっている
（マキアヴェリ）

参考文献

- 参考文献
  - ①「＜改訂第2版＞要求を仕様化する技術・表現する技術」（技術評論社）
  - ②「『派生開発』を成功させるプロセス改善の技術と極意」（技術評論社）
  - 「SEの仕事を楽しくしよう」（SRC）
  - 「わがSE人生に一片の悔いなし」（技術評論社：新書）

USDM

XDDP