SS2009　テストWG

■日程

6月18日（木）WG内の討論　9:00～18:00

6月19日（金）午前中終了後に成果物を事務局に提出（事務局配付のUSBに格納）

■参加者

秋山さん、秋元さん、田中（一）さん、岩田さん、小池さん、田中（英）さん、
西さん、佐々木さん、安達（記録）

■ポジションペーパ

１．秋元さん：ISOL

北海道生まれ、札幌勤務　北洋銀行業務

札幌ではシンポジウムがほとんどないので参加

産学連携対応ED ヨードンの論文：昔から何も変わっていないのでは

会社では JSTQBを受けましょうモードになっている

２．田中さん：IXナレッジ

佐々木さんと同じ会社

2次3次オンライン

その後受け入れテストの部隊長になった

受け入れドメイン知識をどう習得するのか

３．岩田さん：日本ナレッジ

昨年から現職・東京本社：第三者検証が仕事

オープンソースのテストツールを使いこなす・内部に広める役割

社内で第三者テストを実施している組織は少ない

開発者と異なる観点でテストを行うのが大事

SWに限らず、カーナビ・業務アプリなどもテスト対象

４．小池さん：日立製作所・生産技術研究所

SW/HWいろいろと経験。

テスト対応では、コードの中味の問題が大きいと感じていた。

上流の対応：口では上流で品質を作りこむと言っているが実はあまりやれていない

検証も大事だが、よいものを作るには基本的な動作が必要

CMMIなどを使ってプロセス改善を実施

SS岡山でレビュープレゼンを実施したのが最初

100 年に一度の不景気：プロセスに投資するべき

５．田中さん：電盛社
前回 SS も参加。不具合の分析の議論が非常に勉強になった。
半導体製造装置の開発から現職に。担当した製品の市場で不具合が多かった。
上流工程のレビューを強化しようと取り組んでいる
レビューはやっていて、指摘件数・指摘内容の分類を見ているが、効果が見えていないレビューアのスキルがまだまだ、

６．西さん：電気通信大学
不況のためコスト切り詰め。
筋肉もぜい肉も切り詰める組織／筋肉を強めようという組織

７．佐々木さん：アイエックスナレッジ
Web システム開発・何でも屋：保守・運用もやっている
品質が人に依存している：スキルが高い人がやると問題なし、以外はぐだぐだ
全体を責任持っている人がいない

８．秋山さん：富士ゼロックス
昨年初回：事例分析を実施
今年は現場が抱える具体的な問題を取り上げて取り組めればと思います。

９．安達：HBA
SS 初参加
現場が抱える具体的な問題で参加者の共通的なものを取り上げて取り組めればと思います。

＜議論したいテーマ＞
１．秋元さん
・上流での品質確保：細川さんのセッションもあったが、他の方法などを知りたい
★テスト担当の育成／権限分離
・30～40％工数がテスト：工数をどう低減するか

２．田中さん
・ドメイン知識をどう習得するのか
・モチベーションを維持する方法

３．岩田さん

・ツールの有効活用について

標準仕様・標準テストがほとんどない

機能テストツールは高額　１ライセンス：20～100 万なので使えない　など制約が多い

・テストエンジニアの差別化できるスキル習得：ツールの駆使が課題

４．小池さん

・市場に出すタイミングを逃せられない：納期が重要／厳しい

コストも厳しいのでレビューとテストが削減される→まずいと分かっているのにやめられない

どうテストケースを作るのかが重要

分からない人たちにどうアプローチするのか・実績データで示すことが重要だが、いまひとつしっくりこないものが多い

気づき・ヒントなどがあれば

５．田中さん

・上流工程で不具合を食い止める方法

テスト技術、スキルの差は分かっているが取り込まない・・・なぜか?

・レビューの効果向上

・改善への温度差、やらされ感がぬぐえない

６．西さん

・リスクベースドテストをしたい。担当者数名しかいないなのでテストを減らしたい。

ローテーションしていくのでどうやって引き継ぎするのか

・ユーザ：ベンダー／マネージャ：開発者　などどの関係でも同じ状態

欧米化が止められない中で、どうしていくのがよいのか

「モノを作る」をコアに考えていく必要があるのではないか

中長期的視野で戦略を考える必要があるのではないか

参加組織の言：「数字を決めて、数字で見ればよくなる」

品質作りこみガイド：仕様書の頁数からテスト項目を設定

欧米ライクのマネジメントが進むほど空洞化しているのではないか

ユーザも気がつきつつある。

有名メーカ系アップデート、Ver アップでいろいろと修正する必要がある

ユニケージ手法：Linux・Cshell で無印良品のシステム再構築・自らメンテできる

国の批判をしない、大企業の批判をしないメディアも多い

他人に価値観をゆだねている：責任を取らなくて済むから楽

７．佐々木さん

・SS に来ないような人、スキルアップに取り組まない人が多いのが課題
　何とかして成し遂げるメンタル面を持っていないとだめかと
　そういう人たちは、W モデルだの何だのを言っても、やってもダメ
　実務だけしか目を向けようとしない
　ペナルティなし、がんばっても変化しない→意気込みが高まらない原因
　職人気質ではない／机の上に技術書がない現場が多い
　すごいと思った本／講演がないんじゃないか
　・コードコンプリート
　・ピープルウェア
　・実践ソフトウェアエンジニアリング（プレスマン）
　・ソフトウェアアーキテクチャー
　・福岡正信さん：自然農法の本　「無」１・２・３
　・人月の神話
　・デスマーチ
　・デマルコ
　・ヨードン
　　サインをするときのポイントも違う

８．秋山さん

・共通の課題があればそれに取り組めればよいです。
・重要品質問題の発生を防止するテスト方法：なかなかない
・オーダーメード SW：ソリューション系
　例：文書管理システム　既成部品あり→カスタマイズにより問題も発生する
　エンプラ系の Agaile、テスト方法

■テーマ１：上流工程で不具合を食い止める方法（**テスト・レビューの成果向上）**
欠陥をいかに上流工程で検出するか

・レビュー強化対応中。数か月やっているが効果が見えていない。
実施方法がばらばらだったが、客先作業で事前にオフラインレビューを実施。
そのあとオンラインレビュー
・要件メモ＝要件定義を含めて詳細を徹底的に詰める・上流ですべてやりきる

・インスペクション対応の事例：小池さん
アプレイザからの評価結果でレビューが弱いといわれた
レビューの現状と問題点
事前配付・個別レビュー後に・・・理想だが時間がないのでできない
説明に時間をかけすぎ
議論が発散→欠陥の見落とし
自分流のプロセスを定義／レビュー支援システム／拒絶反応を和らげる対応
＜ポイント＞
レビューアが事前に個別レビューの結果チェックにより優先度付与
→レビューの効率向上
議事録記載（木構造）＝指摘範囲－対策－不具合　の階層構造で
→これらが以降のノウハウになる
不良現象コード　設計要素コード　重要度
レビュー効率（欠陥数／時間）
基本設計工程の欠陥摘出密度（欠陥数／P）
詳細設計工程の欠陥摘出密度　　など改善できたことを定量的に把握できている
→ただし、これらの数値に説得力がない
一方、データを取っていれば、着目点などの教育などに使える
単純にレビューを実施するのではなく、やり方や傾向・結果を情報化して使えるようにしていくことが重要

現在展開中。全面適用ではない。
Q1. 欠陥摘出数などが人事考課に使われないのかな？
A1. 使うなと言っている

Q2. 計画に工数がかかるところをどう説得したのか
A2. 1：10：100 を説明して理解を示した組織に適用した
CMMI 適用でプロセスデータを取る組織が増えているのでそこに適用推奨
Q3. 設計品質が向上したことを確認していますか？
A3. 工程毎に欠陥摘出密度が上がったのは定量的に把握している。
レビュー実施前までに欠陥がなくなっている、減っているかどうかは確認していない。

→ノウハウが共有され、プロジェクト要員の流用率が低くても品質が保持されるなどの効果が分かればよいのではないか
例：XDDP：保守理由なども含んで残していこうとしている

Q4. 記載の粒度などは決まりがあるのか
A4. いきなり抽象化するとぼけてしまうので、まずは詳細に記載していくのがよい
→チェックリストにするのが一つの方法ではないか

にしさん
レビューチェックリストをどう作るのかの研究過程にある
・具体例（仕様表現）
　電源オフの際にはデフォルト設定に戻す
・具体例の背景や経緯
　祖語の具体的な
・リスク潜在単位で分解：文章単位　単機能単位
・不良リスク観点
・リスクポイント：仕様表現　　例：完了と終了

効果的なチェックリストのポイント
（1） 具体的な実績情報をそのまま蓄積しておく（個別情報）
　　どんな時に、どんな風になど
（2） 意味のある単位で集約して抽象化する（組織共有情報）
　　3つ実例を出すと理解できる
（3）使うまでに一つ一つ具体化して使う（個別活用）

＜抽象化の方向性が重要＞
機能として抽象化、現象としての抽象化はうまくいかない
★こんな設計やこんな仕様だと間違いやすい　という抽象化がよいのではないか

きっかけは「バグレポート」：テストオペレータがちゃんと書けないとね
　→新入社員がテスト実施をやることが多いが、レポートが書けないとノウハウとして残せなくなる
抽象化の過程で設計者などにヒアリングしながら確認していく
よくこんな間違いするよね？→どうしてなんだろう？
抽象化対応を通じてノウハウ共有しつつ、関係者に気づきを与えること
チェックリストを作ることよりもむしろ、間違いやすいところ、注意すべきポイントなど

を関係者で共有していく、認識を付与していくことが重要

なぜなぜ分析→パターン化＝分類：カテゴリに分けることが重要
実際には集約すること
単体テストを検討する→Ｃ１にする？Ｃ２にする？など結論に直結させたがる

分析した結果、みながうれしくなる内容にすることが重要なのに、面白くない・使えない情報にしたらおしまいの場合が多い。
→自動車会社：あらゆる場面でなんで？どうして？をみんなで問いかけている
　なぜなぜ分析の本質を毎日毎日あらゆる局面で問いかけ続けているからなぜなぜ分析がうまい、質が高い（“それで何がうれしいんだろうか？”）
どこにいっても一貫している：
５ゲン主義：現地・現物・現実／原理・原則を考える

応急処置的な対応に終始しないために、なんでそういうやり方をしなかったのか？今後どうやっていくのか？を問いかけ、考えてもらい、本質的な対応に結び付ける。
バグだけに閉じずにありとあらゆる場面でそれらを考え、行動する。

Q1. このチェックリストの使用単位は？
A1. プロジェクト単位です。

Q2. 34 個？
A2. 現時点での洗い出し状態
　　数としてもよい

DRBFM：重要度に応じてみていこうという考え方
　FMEA 部品の故障パターンから問題を洗い出す手法／ポイント付与が面倒
　自動車会社では変化点管理として使われている
　故障モードがうまく使われている組織は少ない
　JAXA ではうまく使っている／担当者がいろいろなことを知っているので、役立つ故障モードを集められている

・QI は統計値ベースでやっているが、統計情報だけからやると成果が異なる
・議事録記載（木構造）＝指摘範囲－対策－不具合　の階層構造で

→これらが以降のノウハウになる

・テストケース作成は見積もりに入れていないことが多い。優秀な PM は考慮している。
リスクマネジメント工数に入れている場合もあるが、プロジェクト開始時に一度実施して，あとは監視しないこともある

★仕事をしていること、プロセス改善を行うこと、教育すること、が一緒にできる方法がチェックリスト→スキルアップにつなげる
・ふりかえりを行っても内容が何かを付け足すことでおわりになることが多い

・客先に報告するために分析して何かをすることを決める→でもそのあとホントに実施して、効果が出たのか誰も見ていない
ここまで 12:40
（昼食タイム）
14:20～
レビューツール Muses：レビュープロセスの中で機械的にできるところを抽出して作成
・MS-office コメント機能による指摘抽出・コメントマージ
・指摘対応箇所の表示
・議事録作成　など
→インスペクション運営を効率的にするための機能
日立ソフトから発売：ReviewCoordinator
欠陥の種類をどう分類するかを組織で決める必要がある（重要）
レビュープロセス改善ソリューションとして売る予定

日本ではこの手のツールが売れないことが多い
開発に必須だと信じ込まされているものが売れる。
コードインスペクションが必須・国が進めています、の戦略で QAC が売れている。

40 分その場事前チェック：事前チェックをやってこない人たちが多い
暗黙の 2 時間ルールの中で 40 分間だけ個人チェックをやってから再開することで，事前チェックをやらないよりは，品質を向上させることができる．ツール利用によるノウハウの一つ．

Q1.コメントを付けることへの抵抗は？
A1．電子的にコメントを付けるのは時間をかけるとやってくれている（慣れの問題）

秋元さん
暗黙知をどのように抽象化して使えるようにしていくのかが重要なのではないか

上流で何かしらモデリング・図などを書く→最初からちゃんとやる方法を考える
設計書を書きながらテストケースを一緒に考えてみるのもいいかもしれない。

１．解釈を共有する→テストケースではどうなのか？
２．形式化手法：要件ドキュメントに事前条件・事後条件を設定する

青木さんのやり方
PM3 頃に毎日その日の成果物を提出させる→以降作業者は自由に
青木さんは成果物の内容確認→フィードバックを提供する
小さく回す・コミュニケーションを強くして、技術的フィードバックを返してあげる
ものの考え方、進め方を共有していくことが重要
受け取る側が吸収する能力がないと、そしてリーダがちゃんと見てあげる能力が必要
分割しすぎるのもダメ：切り出し方、粒度が適切であれば OK
ある塊、まとまりのある成果物を作ってもらうことがポイント

ではプロセスと基準で青木さんとおなじことができるのか？
ペアプロ標準があっても、すぐ効果が出るようなものではない。

基準までしか対応しない組織　　→　不具合が多い
　　→丸投げ組織
基準外も含めてちゃんと対応する組織　→　不具合が少ない

基準を出すとそれ以上にならない
顧客の要求のとおり

「数字のとおりにやろうとする組織」を「数字だけではないところまでちゃんとやる組織」に持っていくためには

数字のとおりにやろうとする組織
・ある意味で達成感を得るため
・仕事の目的になっている／本当の目的を忘れているのでは

・本当のゴールを分かっていない
・組織が変われば個人も変わるのではないか
・数値があれば分かりやすい

・ケルビン卿の言葉：計測できなければ改善はできない　がまかり通っている
・３人の石工・石仕切り・家族養う・教会を立てている
・問題解決力よりもむしろ問題検知力、発見力が重要
・データは取っているが活用できていない
・定量データか？定性データか？→定性データが重要ではないか
・データで使えるもの＝分母が明確に把握できているもの
　行数・工数、信頼性曲線など分母が明確ではないものはあまり使えないデータ
・青木率＝青木さんレビュー実施成果物量／（その日のレビュー成果物×人数）
　他の人ではできないのではないか
・青木プロセスは記述できないのではないか→計測できない
　じゃあ、どうしたらよいのか？
・組織文化：腕の良い人が品質保証に来る組織、腕の悪い人が品質保証に来る組織
　開発→テスト→品質保証→開発→テスト→品質保証・・・・・・・
・細川さんいわく：QA 部門がなくなることが目標
・分けられない組織もあるので、その場合も考えたい
　バッドノウハウばかりが蓄積される

どこで役に立つスキルなのか？

ガバナンスの話で、保守する人、テストする人／開発する人、運用する人
悪意があり犯罪行為につながる　ということがないとすると？
効率やスキルを考えるとひとりでやっても大丈夫だと思う。
開発した人しかわからない欠陥をかぎわけることもある。

＜分けたほうがよい＞
・ユーザ視点が出やすい
・出来上がってすぐ市場に出さないで済む→致命的な問題が発生しずらい
　開発者の思い込みで触らないところがある
・一つのクライテリアまでに持っていく QG に対するモチベーションが得られるのではないか　→テストチームからスクリプトを渡して作りこみ時点で注意できる

・テストを真剣に考えてくれる、スキルアップに努めてくれる
　自らコードを書いてよくある間違いテストしてみるなどの努力もやってきたが、テストだけやれればよいと狭く考える人も一方でいるのがデメリット
　→開発との間でローテーションをすると・・・→開発に移ると急に偉くなった勘違いもある
・自社開発分で問題が発生した事例から社内でもテスト部隊を独立した経緯あり
　→事故の発生抑制になっている：〇〇を通過したという形式的な対応も大事ではないか

文化の違いは大きい／その文化によって対応方法が異なる
　・品証と仲が悪い組織　→　分けたままでよいのではないか
　　品証に文句を言われないように，自分たちで品質を高めようという意識があれば，それはそれで運用すればよい.

　・品証と連携して対応する組織　→
組織を分けた際に、ちぇんと理由や趣旨を説明することも重要
作る製品によって対応方法が異なるのでは
→問題発生時の金額＝責任感・緊張感を高める
　規制を超えて考えるマインドを育てることが大切
マインドをプラクティスに埋め込まれているのがよいのでは。

ペアプロの成功要因
２名の特性によって成果が変わる／相互のアプローチがかみ合わないと
組合せ方が難しい／成功した話しか見ない

ペアデバッグはうまくいっている
バグを２名（間違えた人＋１名）で対応させる
開発した人が直すと、同じようにミスしてしまう
思い込みや論理性の欠如、まずさを排除する効果＋まずさの情報の共有ができる

職人に食いつきの良い素人を付けると、職人が持つノウハウをどんどん吸収していく
ごちゃごちゃしている全体　→　一度分解していく　→　その分解された要素を再整理してみる

大事なことは部署（手段の一つではある）を分けることではなくて、役割として分ける、別の役割の方と一緒にノウハウを共有するなどが重要。

・製品による
・組織の長の考え方
・顧客（特定ユーザ（厳しさ）・一般市場）による、顧客に合わせる
　→顧客との契約に依存する欧米型では品質が上がらない
　　常に端っこのギリギリを狙っていると、真ん中すら狙えなくなる
　　自分の基準、譲れないものを持っているか
・ユーザ系組織とベンダー系組織では違う
　ユーザ系組織は外部発表を聞きに行かない、質問をしない
　→メーカを呼んで説明してもらうから／メーカはサービスの一環として対応する
　　最近はガバナンスの関係で、１社だけではだめなのでこれまで無関係の組織にも聞くようになってきている
・システム子会社（エンプラ）が・・・御用聞き体質になりがち
　実測データなども活用して運営するノウハウも、その気もない

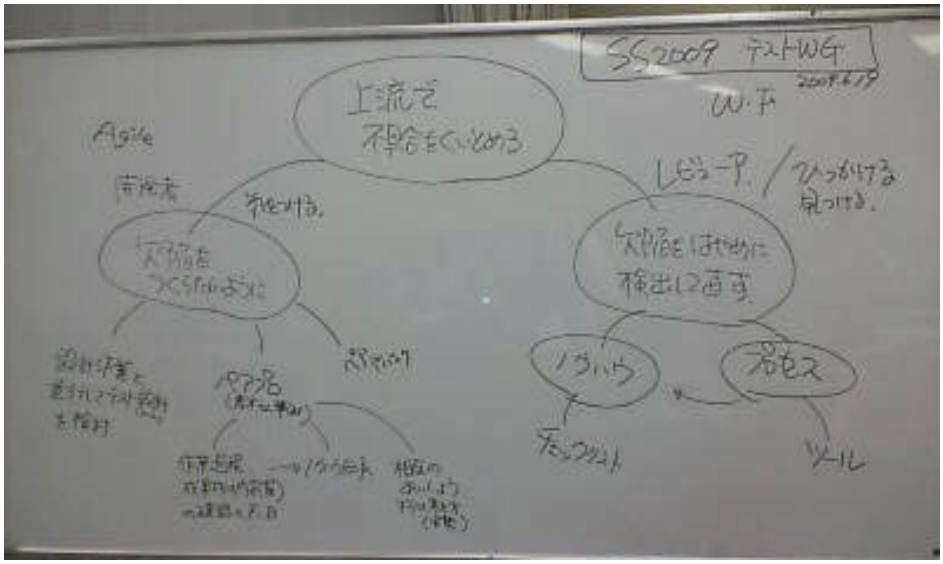
※分けていても、場が共有できている（場所が近くにあるとか)、喫煙所が同じなどの場合は分けていない状態に近くなる
　→距離的な近さだけではないかもしれない
　→顔を見ることで状態が分かるという面もある

＜分けない方がよい＞
・上流から一貫して対応できるから
　・関連情報、スキルの伝達・共有がスムーズ／上流で何をやっているのかわからないと検証できないことも多い
・分けると開発側がテスト側に甘えて丸投げ対応することもあるから
・しくじっちゃったけどばれてない、自分が感じているまずさ加減を持ちつつ通過した後でも対応できる可能性が高い

分けたとしても、何かの手段で解消できる要素があれば“分けない”方にシフトできる

コミュニケーション
フォーマルなもの・インフォーマルなもの
・何の？　よいこと、わるいこと（どこが悪いのか、どこが悪くなりそうか)
　兆候は誰かが気付いているけど・・・言うとやらされるから言えない、言わない
　レビュー終了後に感想を聞くのもよいかも→構えずに話せる場が必要
・どうやって？　誰と？　いつ？

＜結論＞
今回の討議の全体像



お互い（分ける、分けない）のメリットを考えて、ある過程を経て共有するべきことを共有したうえで分ける、分けないを決める／
社内標準などで決まっているからではなくて、こういう理由で、こういう根拠でここまでを分ける、ここまでは分けないと判断し、関係者で認識共有する。

プロダクト QA だけではなく、プロセスも同様に作りこみの仕方を確認する（例：レビューでプロセスも確認する）なども必要。第三者検証チームが受け入れる際も、規模あたりの欠陥数などの数値データだけではなく、どのように対応してきた結果そうなっているのかなども確認して受け入れる必要あり。→プロセス QA：中味を伴ってちゃんとやったかの確認をする。
どちらかではなく、両方を見て判断する。

２日目
9:00～１日目のおさらい：全体整理
10:00～とりまとめ＋スキルアップについての討論
～11:00 まで
現状復帰
「各自のＰＰ」「２日間の成果物」をＵＳＢに格納し提出する。
以降昼食

13:00～クロージングキーノート