ミニワークショップ

テストから現場を改善していこう！

～ テスト結果を、もっと活かそうではなイカ!? ～

◆進行：JaSST Kansai実行委員会

◆資料作成：角口勝隆、亀井浩気

2011/07/22

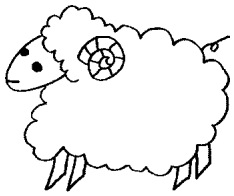
■アブストラクト

- ソフトウェア開発において、テストで不具合が発生することは常である。しかし、毎回テストにより不具合を潰すだけでは、効率は上がらない。不具合を低減させて効率を上げるには、テスト結果を開発へフィードバックし改善を図る必要がある。
- テスト結果から、フィードバックすべき情報とは何か？また、どのように分析をすれば良いのか？そのような声をよく聞くが、本当に必要なのは、「与えられた情報を吟味し、そこから必要な情報を得る”眼”ではないか？」と考える。

■ワークショップの概要

- 本ワークショップでは、テスト実施中に発見された不具合情報をもとに、開発進行中の製品や、次の開発プロジェクトを改善していくことを想定し、以下のテーマでグループワークおよび討論を行う。
  - 不具合情報をODC（直交欠陥分類）で分類
  - 開発対象製品の問題傾向を着眼点（影響度および作り込み要因等）に沿って掘り下げ／選定し、必要なアクションを考察
  - 開発プロセスの問題傾向を着眼点（どの工程で、どのような不良を見逃したのか等）に沿って掘り下げ／選定し、必要なアクションを考察
- 現場で発生する不具合を低減させたいソフトウェア技術者は、是非とも参加頂きたい。

■目次

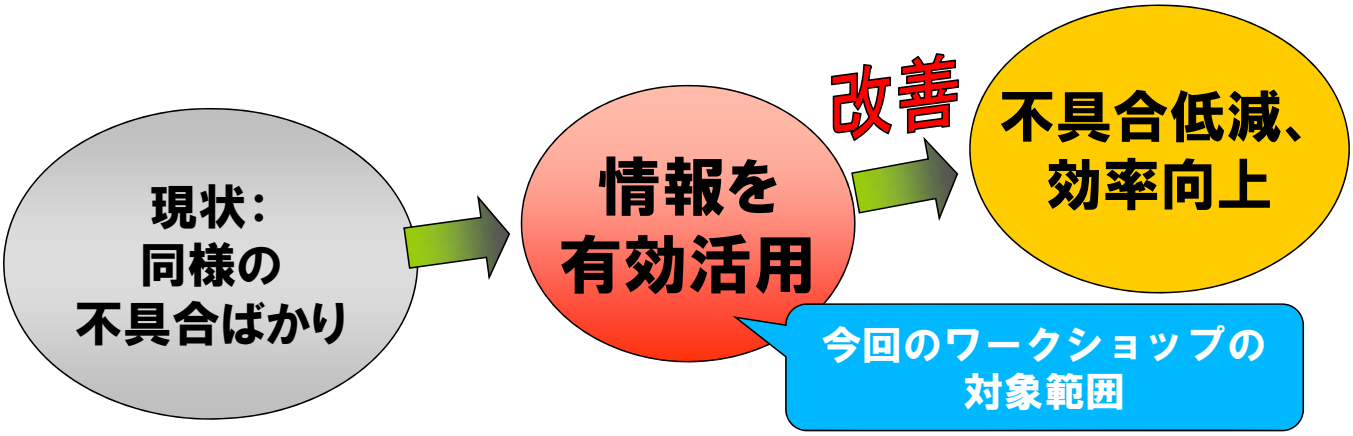


1. オープニング
2. ワークショップのシチュエーション
3. ODCの活用による不具合分類演習
   3.1 演習1：分類をしてみなイカ!?
   3.2 演習2：解決の糸口を探さなイカ!?
   3.3 演習3：過去の過ちを振り返らなイカ!?
4. グループ発表
5. クロージング

■とある現場にて。

- いつも、似たような不具合ばかり。いつまでも減らない。
- 先週対策したのに、また不具合。ちゃんと見直しているの？
- 「見える化」だからと、多くの情報を集めたが、それらの情報が役に立ったようには見えない（活用されていない）

集めた情報を分析し、改善ポイントを見つけたい！！



■情報を有効に活かすには？

- **問題を正確に捉える必要がある。**
- **そのためには、色々な切り口で問題を分類し、整理する必要がある。**
- **整理した情報を吟味し、そこから改善のためのヒントを見つけ出す必要がある。**

複雑に絡まった問題も、ひとつずつ分解すれば、元は1本の糸。

解決の糸口

どうやって、みつける??

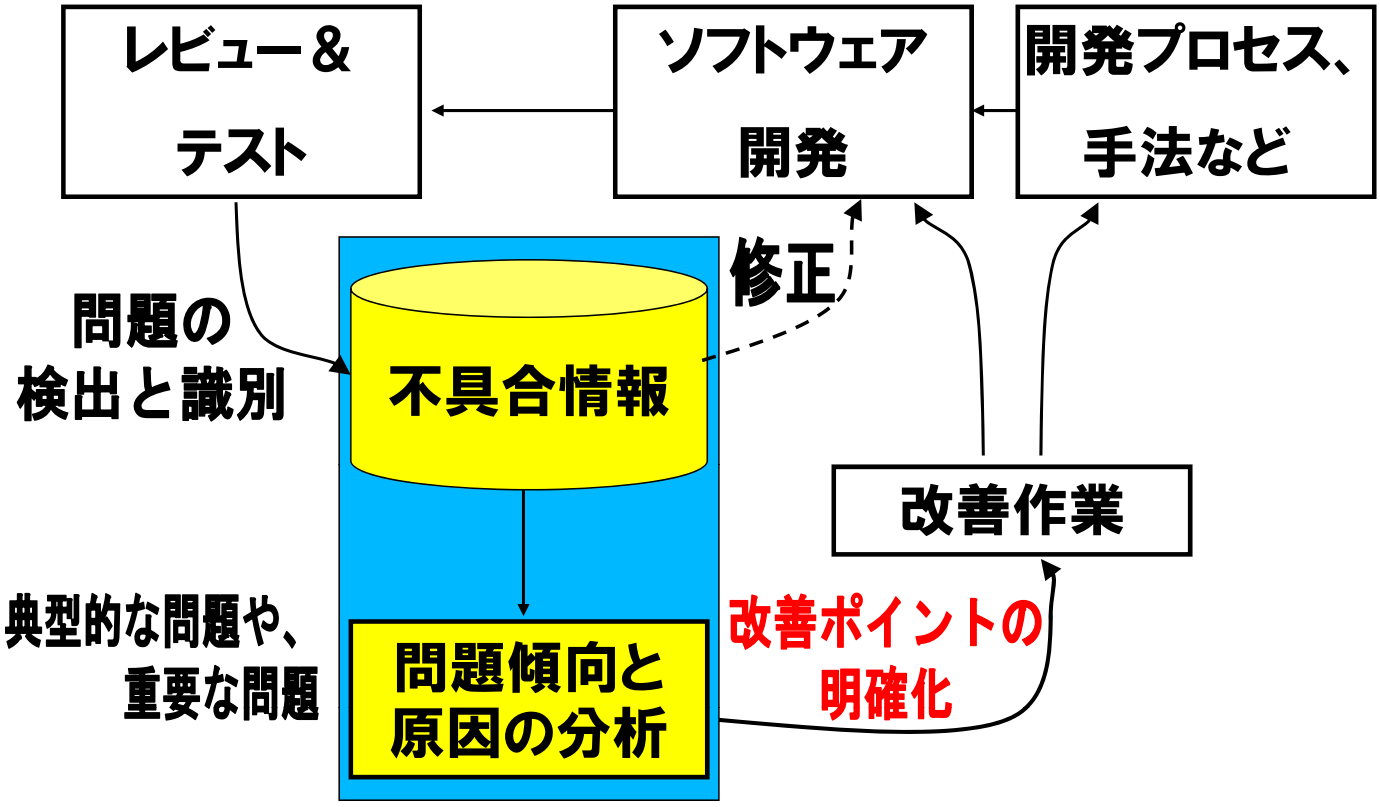
■解決の糸口をつかむには？

- 幾つかの**不具合には同じような傾向がある**といわれている。
- 不具合の**傾向を分析すれば、改善すべきポイントが見えてくる**。
- 改善ポイントに対して**的確に対処すれば、不具合発生率は下がる**はず。

不具合情報の傾向を見つけることが、解決の糸口になる

～解決の糸口

■不具合分析から改善までのフロー



■目次

1. オープニング

2. ワークショップのシチュエーション

3. ODCの活用による不具合分類演習

3.1 演習1：分類をしてみなイカ!?

3.2 演習2：解決の糸口を探さなイカ!?

3.3 演習3：過去の過ちを振り返らなイカ!?

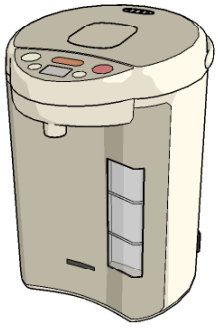
4. グループ発表

5. クロージング

■開発中の製品について

- 話題沸騰ポット
- 胡麻印まほうびん(株)社が開発中の新製品
- 手動(手押し式)のポットを電子化し、ボタンで操作可能とする
- 特に安全性(信頼性)が求められる

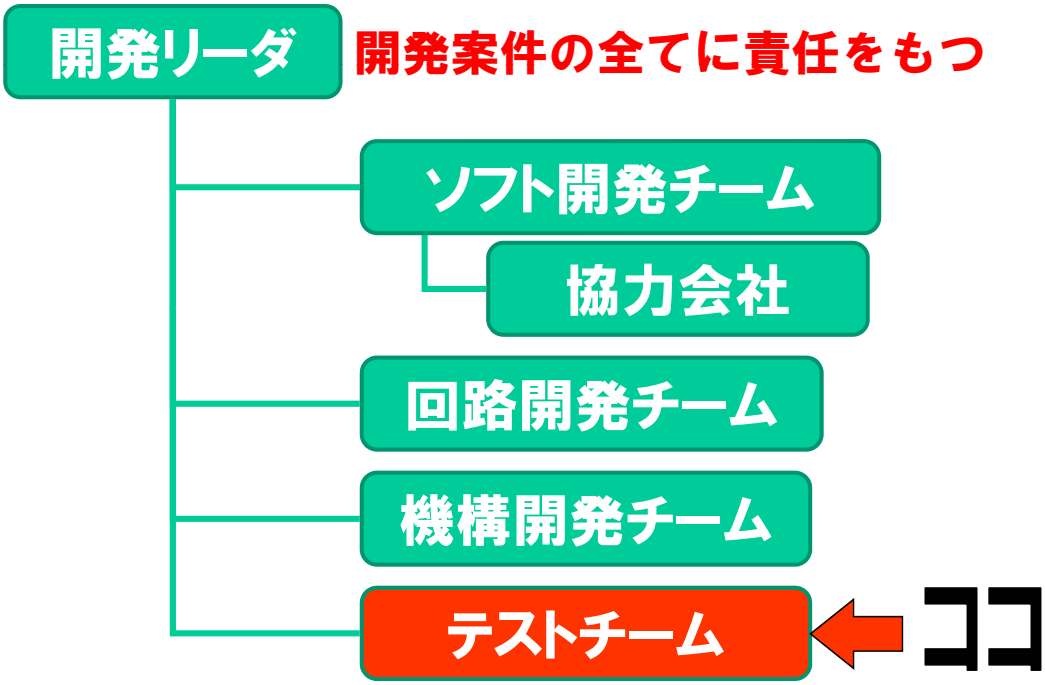
エアーポンプ式 → **電子制御による電動ポンプ式**



参考：話題沸騰ポット要求仕様書 (GOMA-1015型) 第7版

http://www.sessame.jp/workinggroup/WorkingGroup2/POT_Specification_v7.PDF

プロジェクトの組織体制

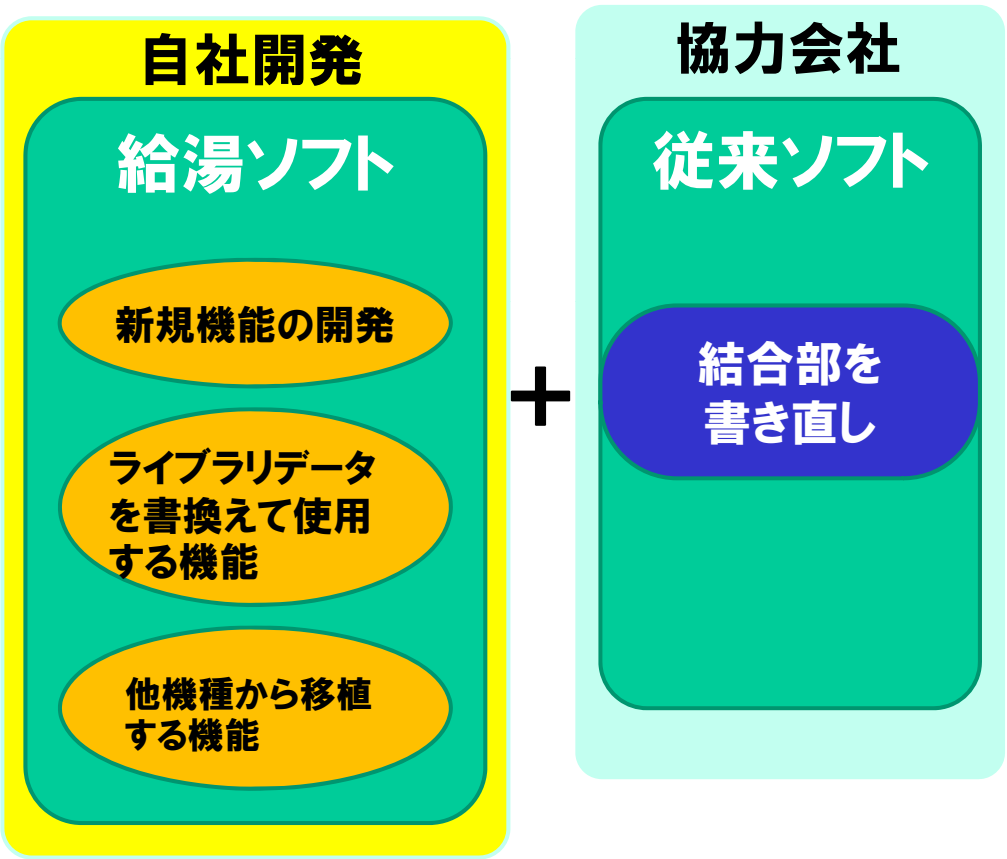


・開発とは独立したテストチーム
・システムテストを実施
・テスト結果を開発リーダーに報告

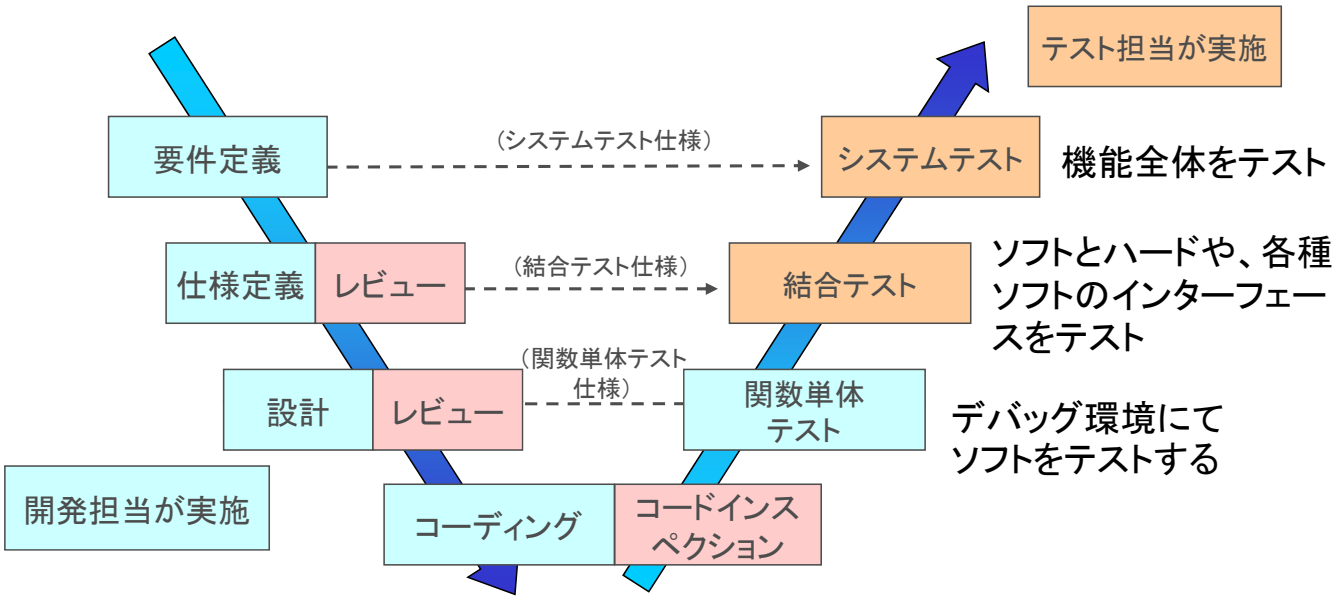
■あなたの立場と状況

- 製品のテストチームに所属。
  テスト結果を踏まえて、開発リーダーへ品質状況を報告する。
- 話題沸騰ポットのシステムテスト工程の、テスト項目の消化が，半分を過ぎたあたり。日程および不良摘出数の進捗も、計画通り。修正件数の進み具合も計画通り。
  → 定量的には計画通りであり、品質に問題は発生していないと判断できる。
- しかし、不具合内容を見て、定性的にも検証をしなければ正確な判断はできない。
  → 情報を有効活用するために、ODCで不具合を分類しよう！

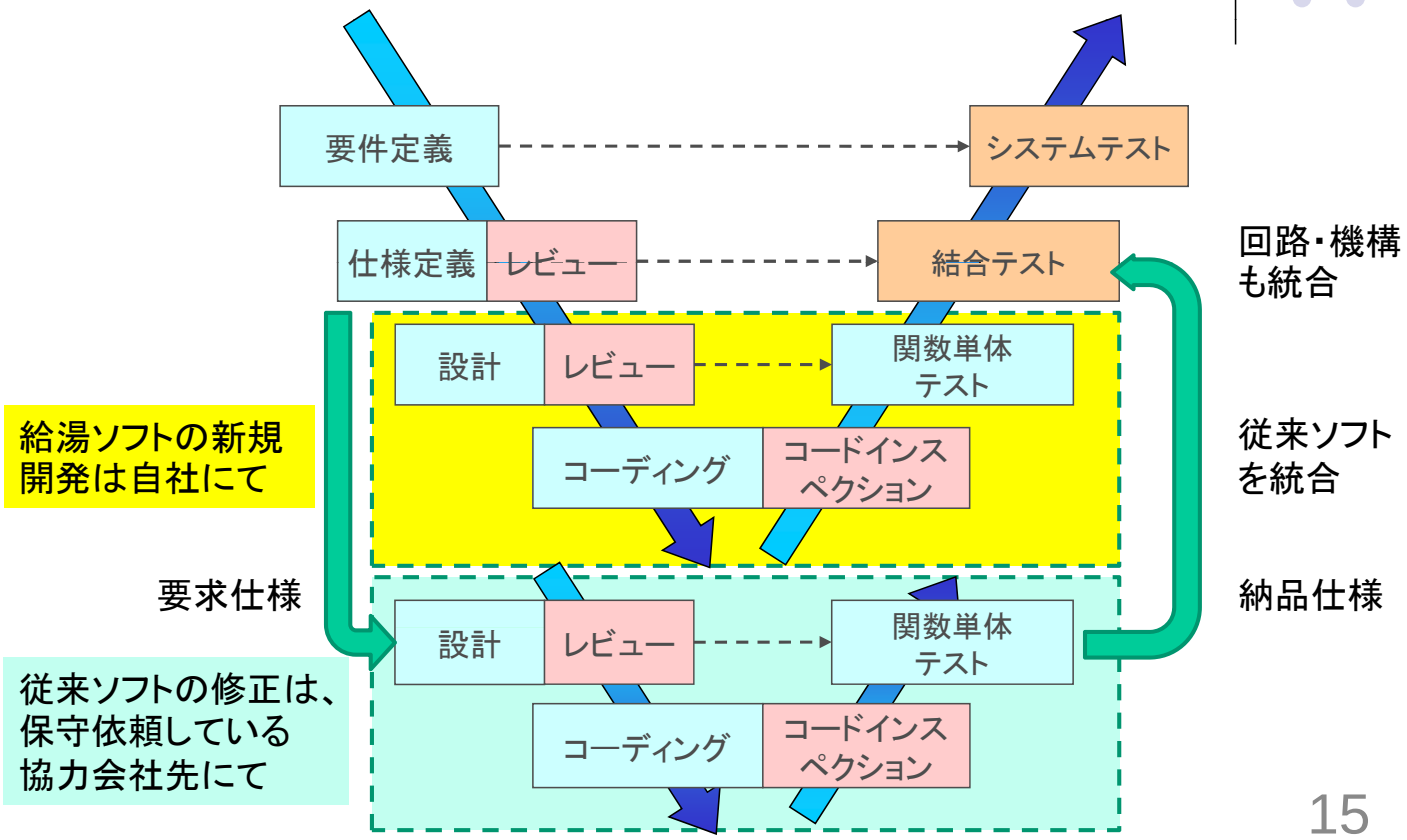
話題沸騰ポットの ソフトウェア開発イメージ



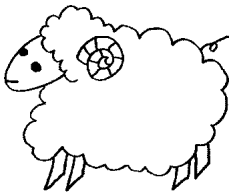
部署で採用している開発プロセス



今回の開発プロセス



■目次



1. オープニング
2. ワークショップのシチュエーション
3. ODCの活用による不具合分類演習
   3.1 演習1：分類をしてみなイカ!?
   3.2 演習2：解決の糸口を探さなイカ!?
   3.3 演習3：過去の過ちを振り返らなイカ!?
4. グループ発表
5. クロージング

Orthogonal Defect Classification（ODC：直交欠陥分類法）とは？

- 迅速にソフトウェアの欠陥を、「意味」で捉える方法
  - 統計的解析でモデル化を可能とする
  - 健康診断における血液検査のようなもの
  - 不具合数や信頼度成長曲線などでは露見しない問題が発見でき，多角的な評価が可能になる

17

ODC分類の構成

- 次の2つの構成と、8つの分類項目がある

構成内容	項目	意味
欠陥報告時（Opener Section）	Defect Removal Activities	どこのプロセスで発見したか？
	Triggers	どのような状況で発生するか？
	Impact	どのような影響を及ぼすか？
欠陥修正時（Closer Section）	Target	どこが悪かったのか？
	Defect Type	何を間違ったのか？
	Qualifer	どのように間違ったのか？
	Age	どの時点で間違ったのか？
	Source	誰の物が間違ったのか？

18

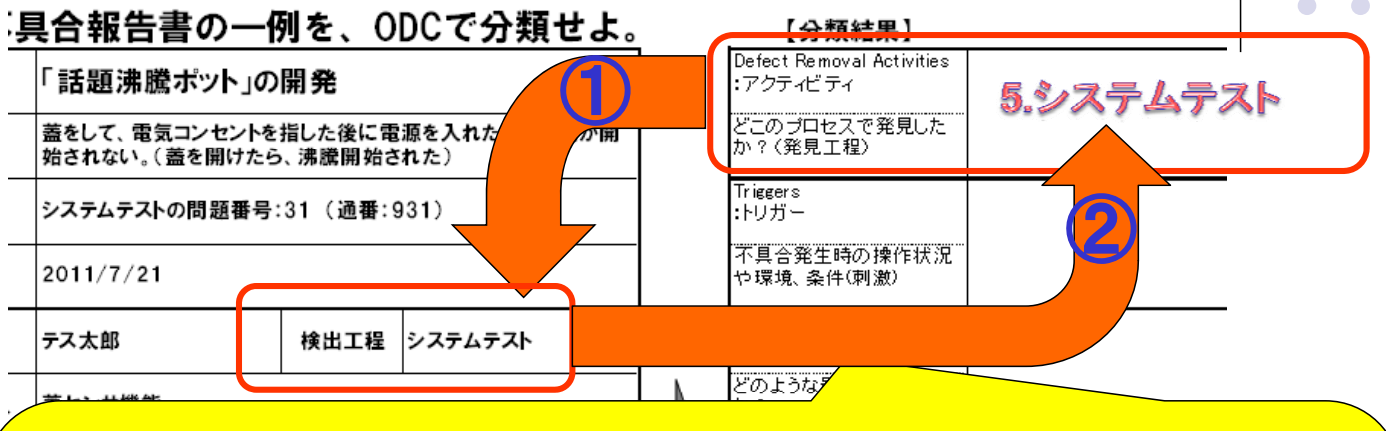
演習１：
ODCの分類を、やってみよう！

- ①不具合は、どんな特徴があるのか？
  - 各個人で、不具合１件をODCで分類してみる
    - 尚、テスター側が記載するのは、欠陥報告時（Opener Section）である。欠陥修正時（Closer Section)は通常記載しないが、体験するという意味合いで、取り組んでください
- ②不具合の分類結果について、グループでディスカッション

資料1と、演習1の用紙を準備ください。

合図があれば、始めてくださーい。

課題①各人で、不具合をODC分類実施
課題②不具合の分類結果について、グループでディスカッションを実施



- 分類を実施する上でのコツ
  - ①ODCのカテゴリを捉えて、不具合報告書から連想するキーワードを探す
  - ②資料1の分類一覧から、一番近い分類項目を埋める
- どうしても分類がわからない場合
  - 細部にこだわらずに、直観的に答える
    (気になるなら、「？」などのフラグを立てておく)

演習1の回答例

不具合分類の結果（例）その１

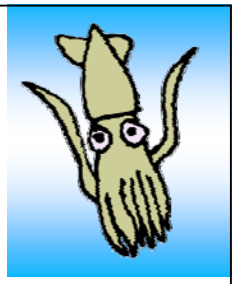
Defect Removal Activities :アクティビティ	5.System Test（システムテスト）
どこのプロセスで発見したか？（発見工程）	「検出工程」から抽出した。
Triggers :トリガー	10.Simple Path(通常使用)
不具合発生時の操作状況や環境、条件(刺激)	通常利用での不具合であり、上記が一番近いと判断した。
Impact :影響	6. Reliability(信頼性)
どのような影響を及ぼすか？	「9. Requirements(必要条件)」と迷ったが、蓋を開ければ沸騰するので、完全に要件を満たさない訳ではなく、上記が一番近いと考えた。
Target :ターゲット	3.Code（コーディング）
修正の対象。どこが悪かったのか？	「修正箇所・Ver.」から、修正対象がコーディングであると、判定した。

演習1の回答例

不具合分類の結果（例）その２

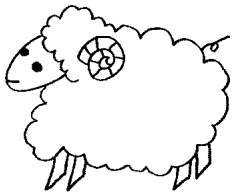
Defect Type :欠陥の種類	3. Alg/Method(アルゴリズム／実装方法)
何を間違ったのか？	不具合詳細から、「ロジックの間違い」というキーワードを捉えて、上記と判定した
Qualifer :不具合の属性	2. Incorrect(不正確(誤り))
どのように間違ったのか？ なぜ欠陥が入り込んだのか？	不具合詳細から、「間違い」というキーワードを捉えて、上記と判定した
AGE :新規性	1. Base(従来)
どの時点で間違ったのか？	開発イメージの図では、蓋センサーは開発対象とされていない。従来のソフトウェアロジックが、ハードの変更にマッチしないと推定。
Source :ソース(出所)	1. Developed In-House(自社開発)
発生源はどこか？	主な原因分類から、自社開発ソフトというキーワードを捉えて、上記と判定した

［現場で運用する際の注意事項］こうすれば、いいのではなイカ！？



- 分類に関する、関連者のベクトル合わせが肝心
  - 分類する人の主観的な結果とならないよう、事前の意識合わせを行う必要がある。（分類開始前のブリーフィングを実施しておく）
- プロジェクトの途中では、分類項目は変更しない
  - 積み上げた分類結果が、崩れる可能性がある

■目次



1. オープニング
2. ワークショップのシチュエーション
3. ODCの活用による不具合分類演習
   - 3.1　演習１：分類をしてみなイカ!?
   - 3.2　演習２：解決の糸口を探さなイカ!?
   - 3.3　演習３：過去の過ちを振り返らなイカ!?
4. グループ発表
5. クロージング

◆演習２ 解決の糸口を探そう（ODCのメリット）

ODCのように直交している分類の場合

- それぞれの属性が、5W1Hのように独立しており、お互いに影響を与えない。
- 分類した項目について、視点（属性）を替えて、さらに分類できる。
- 属性の内訳をカウントすることで、問題の有無や偏りを把握することができる。

その他、直交していない分類の場合

- 属性間に相関関係があると、分類した項目の視点（属性）を替えるには、それぞれの影響をたどって分類をし直す必要がある。（煩雑である）

演習２：プロダクトの問題傾向を考察

- テストチームは、テスト項目の半分を消化した時点で、以下を把握する。
  - 不具合の報告状況から、現時点での品質状況を確認する
  - 残り半分のテスト項目を実施可能か判断する
  - 残り半分のテスト項目を実施したのち**出荷可能か判断**する
- そのためには、出荷後の**影響を把握したい。**
  - **→ ODCの分類項目のImpact（影響度）を評価する。**

■現在の状態を調査しよう！

- システムテストで、出荷に大きな影響を及ぼす不具合が発生していないか、現在の状態を調査しよう！
  - 不具合の影響には、どんな傾向があるのか？
    →「影響度：Impact」で内訳をカウント

資料2と、演習2-①の用紙を準備ください。

課題①不具合を「影響度：Impact」で内訳をカウント

演習２：プロダクトの問題傾向を考察

影響度：Impact	分類される問題番号を、記載する。	件数	評価(ワーストに×)
記載例1	#2、#4、#5	3件	
記載例2	(なし)	0件	
記載例3	#1、#3、#6、#7、#8	5件	×(悪)
1. Installability (インストール)			
2. Serviceability (保守性)			×
3. Standards (標準)			
4. Integrity/Security (完全性／セキュリティ)			

問題番号で、カウント

偏りがある部分を、考察対象として選定する

グループ内で、コード番号を読み上げる人と、問題番号を記載する人を分担すれば、スムーズです

■現在の状態を調査しよう！

- 「影響度：Impact」の内訳をカウントした結果、
  - 不具合の影響には、どんな傾向があるのか？
    - 影響度：もっとも重要度の高い、Reliability（信頼性）で11件（34.3%）の不具合が発生している。

演習２－①：プロダクトの問題傾向を考察（回答例）

影響度：Impact	分類される問題番号を、記載する。	件数	評価(ワーストに×)
1. Installability（インストール）			
4. Integrity/Security（完全性／セキュリティ）			
5. Migration（マイグレーション）			
6. Reliability（信頼性）	……	11	×

１１件（34. 3%）と、もっとも多い。他の分類項目でも、調査が必要！

■さらに、詳細調査しよう

- 影響度のReliability（信頼性）で、11件（34.3%）の不具合が発生していた。
  - 期日までにリリースするように、品質を確保するには、どうすれば良いのか？
    - システムテスト工程の半分の時点では、まだまだ不具合が発生する可能性がある。（先行きの見通しをつけたい）
    - テストで摘出してから不具合を修正するより、先に弱点である部分を把握し対処する。
- 他に傾向が無いか、分類項目を変えて、傾向を把握しましょう！
- 選定した中から、不具合の詳細を確認し、傾向を「考察」しよう。

課題②他に傾向が無いか、分類項目を変えて、傾向を把握しましょう！選定した中から、不具合の詳細を確認し、傾向を「考察」しよう。

演習２：プロダクトの問題傾向を考察

影響度：Impact	分類される問題番号を、記載する。	件数	評価(ワーストに×)
記載例1	#2、#4、#5	3件	
記載例2	(なし)	0件	
記載例3	#1、#3、#6、#7、#8	5件	×(悪)
1. Installability (インストール)			
2. Serviceability (保守性)			×
3. Standards (標準)			
4. Integrity/Security (完全性／セキュリティ)			

問題番号で、カウント

偏りがある部分を、考察対象として選定する

他の分類項目で、評価をしてみる。

必要により、問題番号を元に問題点の詳細を確認する

演習2の回答例(影響度×ターゲット)

プロダクトの問題傾向の考察結果

①影響度：もっとも重要度の高い、Reliability (信頼性) で11件 (34.3%) の不具合が発生している。

→他の分類項目でも、傾向をみてみる。

②上記①の中で、「ターゲット」で絞り込んだ結果、コーディングでの問題が8件 (72.7%) 発生している。

→設計不良ではなく、実装段階での不良が多い傾向にある。

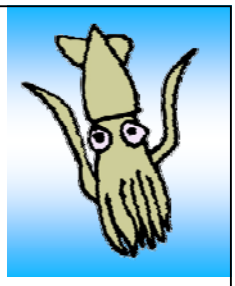
→絞り込んだ8件に対して、問題点の詳細を確認し、詳細分析を行う。

演習2の回答例（信頼性×他の分類項目）

プロダクトの問題傾向の考察結果

分類項目	集計結果	件数	考察
Triggers： どのような状況で発生するか？	1. Design Conformance（仕様適合）	6	従来からの古いソースコードが原因となって、コーディング実装段階での抜け漏れが多い傾向にある。 どのソースコードに、どのような問題が発生しているのか特定するために、観点を定めたコードインスペクションの実施を検討する。
Target：どこが悪かったのか？	3. Code（コーディング）	8	
Defect Type：何を間違ったのか？	3. Alg/Method（アルゴリズム／実装方法）	6	
Qualifer：どのように間違ったのか？	1. 抜け漏れ　および 2. 不正確（誤り）	各5	
Age：どの時点で間違ったのか？	1. Base（従来）	9	
Source：誰の物が間違ったのか？	3. Outsourced（アウトソーシング）	8	

［ポイント］
こうすれば、いいのではなイカ！？



- 情報をカテゴリごとに分類し、細分化することで、見えなかったものが見えてくる。
- 抽象化して情報量を減らせば、見えなかったものが見えてくる。
- 問題発生傾向に偏りがある部分が、特に「テコ入れ」すべき改善ポイント。
  →選定した問題の詳細について、さらに考察する。

■目次

1. オープニング
2. ワークショップのシチュエーション
3. ODCの活用による不具合分類演習
   3.1 演習1：分類をしてみなイカ!?
   3.2 演習2：解決の糸口を探さなイカ!?
   3.3 演習3：過去の過ちを振り返らなイカ!?
4. グループ発表
5. クロージング

演習３－１：過去の過ちを振り返ろう（プロセスの問題傾向を考察しよう）

- **開発上の問題点を振り返る場面にて。**
- **プロセス上の問題が無かったのか、振り返りたい。**
- **不具合の作り込み工程と、摘出工程が同じであるのが、理想。本当にそうなっているのか、現状を把握したい。**

→「トリガー」を元に、アクティビティを辿ってみよう。

トリガーとアクティビティの関連性

IBMリサーチのHPでは、ActivityとTriggerの対応付けの一例が提示されています。今回は、この一例に沿った対応付けを利用します。

Activity	Triggers
Design review	• Design Conformance • Logic/ Flow • Backward Compatibility • Lateral Compatibility • Concurrency • Internal Document • Language Dependency • Side Effect • Rare Situations
	• Design Conformance • Logic/ Flow • Backward Compatibility

Triggerの傾向から、見逃し工程（本来発見されるべき工程）を推測してみる。

課題①トリガーから、アクティビティで辿って、「本来発見すべき工程」を導き出す。

アクティビティ / トリガー	1.Design Rev（デザインレビュー）	2.Code Inspection（コードインスペクション）	3.Un（ユ
1.Design Conformance（仕様適合）			
2.Logic/ Flow（論理／フロー）			
3.Backward Compatibility（旧バージョン互換性）			
4.Lateral Compatibility（一般的互換性）			
5.Concurrency（並行性）			
6.Internal Document（内部文書）			
7.Language Dependency（言語依存性）			
8.Side Effect（副作用）			
9.Rare Situations（レアケース）			
10.Simple Path（通常使用）			
11.Complex Path（複合条件での使			

Age	Source	問題番号	欠陥場所	本来発見されるべき工程
どの時点で間違ったのか？	発生源はどこか？			
1. Base（元から）	4. Ported（ポーティング）	15	沸騰処理	1.Design Rev（デザインレビュー）
1. Base（元か	1. Developed House（自発）	16	ヒーター処理	2.Code Inspection（コードインスペクション）
	eused n ary（ライブからの再）	17	表示処理	
		18	沸騰処理	
			処理	

①トリガーから、アクティビティをたどって、

②本来発見されるべき工程を、求める。

資料3と、資料2を用意ください

［回答例とポイント］

資料３：アクティビティと、トリガーの

トリガー ＼ アクティビティ	1.Design Rev（ﾃﾞｻﾞｲﾝﾚﾋﾞｭｰ）	2.Code Inspection（ｺｰﾄﾞｲﾝｽﾍﾟｸｼｮﾝ）	3.U（ユ
1.Design Conformance（仕様適合）			
2.Logic/ Flow(論理／フロー)			
3.Backward Compatibility(旧バージョン互換性)			
4.Lateral Compatibility(一般的互換性)			
5.Concurrency(並行性)			
6.Internal Docum			
7.Language Depe 性)			
8.Side Effect(副作			
9.Rare Situations			
10.Simple Path(通			
11.Complex Path(複 用)			
12.Coverage(カバレッジ)			
13.Variation(バリエーション)			

Activities	Trigger	Target	
どこのプロセスで発見したか？（発見工程）	不具合発生時の操作状況や環境、条件(刺激)	修正の対象。どこが悪かったのか？	本来発見されるべき工程
5.System Test（システムテスト）	1.Design Conformance（仕様適合）	2.Design（仕様・設計）	1.Design Rev（デザインレビュー）
		3.Cod ーディング）	1.Design Rev（デザインレビュー）
	件	2.Des 設計）	3.Unit test（ユニットテスト）

①トリガーから、アクティビティをたどる

資料3では、「仕様適合」のトリガーに対応する、本来発見されるべき工程は「デザインレビュー」か「コードインスペクション」である。

② 「デザインレビュー」か「コードインスペクション」か迷った場合、ターゲット（修正対象）を確認すると、本来発見されるべき工程がわかる。

39

■不具合を、早い段階で低減させるには？

- **分析の結果、前工程で発見されるべき不具合がシステムテストで摘出されているようである。**
- **今後のプロジェクトでは、不具合を早い段階で低減させたい。そのためには、どうすれば良いのか？**
  - **なぜ、不具合を見逃したのか？**
  - **少なくとも、不具合の作り込み工程と同じ工程で不具合を検出するにはどうすればいいのか？**

40

■演習３－２の課題

●対象範囲について

演習３－１から導きだした、「見逃し工程（本来発見されるべき工程）」で、**もっとも対策効果が高い工程**は、どこか？

●実施すべき課題について

その中で、ODC分類のうち、どれを用いれば、**具体的に実施すべき課題を導き出せる**のか？

グループで検討した後、上記分類で問題を選定し、考察してください。

課題①「見逃し工程（本来発見すべき工程）」で、もっとも対策効果が高い工程は、どこか？
課題②具体的に実施すべき課題とは？

演習３－２：不具合を早い段階で低減させる施策を考察

見逃し工程	分類される問題番号を、記載する。	件数	評価（ワストに×）
1.Design Rev（デザインレビュー）			
2.Code Inspection（コードインスペクション）			
3.Unit test（ユニットテスト）			
4.Function Test（機能テスト）			
5.System Test（システムテスト）			

評価が「×（最も悪い）」分類に対して、さらに絞り込む分類を検討して、集計を実施。

	分類される問題番号を、記載する。	件数	評価（ワストに×）
1			

考察結果をグループ発表いただきます。

なぜ、その分類を選んだのかも、発表してくださいね！

①主張：言いたいことの要点。良いのか？悪いのか？	XXXには、XXXの傾向があり、特に問題が集中している。
②根拠：事実に基づいたもの。（仮定とは、識別すること）	XXXとXXXで集計の結果、XXX件（XX%）と、最も比率が高い。
③提案：改善に向けて実施すべき課題	XXXXについて、より詳細を確認し、重点的に見直し実施する必要がある。

■目次

1. オープニング
2. ワークショップのシチュエーション
3. ODCの活用による不具合分類演習
   3.1 演習1：分類をしてみなイカ!?
   3.2 演習2：解決の糸口を探さなイカ!?
   3.3 演習3：過去の過ちを振り返らなイカ!?
4. グループ発表
5. クロージング

4． グループ発表

- パネラーの紹介（敬称略、50音順）
  - 宿口雅弘氏(JaSST Kansai実行委員)
  - 幡山 五郎氏(オムロンソーシアルソリューションズ)
  - 細川宣啓氏(日本IBM)
  - 森崎修司氏(奈良先端科学技術大学院大学／静岡大学)
- グループ発表

  （演習2、演習3について、当日発表いただきました！）

■目次

1. オープニング
2. ワークショップのシチュエーション
3. ODCの活用による不具合分類演習
   3.1　演習1：分類をしてみなイカ!?
   3.2　演習2：解決の糸口を探さなイカ!?
   3.3　演習3：過去の過ちを振り返らなイカ!?
4. グループ発表
5. クロージング

■クロージング

・不具合分類により、問題解決の糸口を掴むことはできたでしょうか？

・多くの不具合に対して改善を図る場合、不具合の傾向を掴み選定することで、重要度の高い部分から効率的に改善を図ることができます。

・テスト項目消化率や、摘出不良数など、数値だけの評価だけではなく、不具合内容を定性的 および 定量的に評価することで、見えなかった問題が見えてきます。

・本日体験頂いたことを現場でも活用いただければ幸いです。皆さまの現場での、今後の活躍を祈念します。

■参考文献など

◆IBMリサーチ
http://www.research.ibm.com/softeng/ODC/DETODC.HTM

◆ソフトウェアの欠陥予防
－ テストより確実な品質改善法 －
（日経BPソフトプレス 社、ISBN-13: 978-4891005979 ）

◆成功に変える！失敗力トレーニング
（TAC株式会社、ISBN:978-4-8132-3066-3）

◆ソフトウェアの品質管理を定量的に行うために
第23回ソフトウェア生産における品質管理シンポジウム チュートリアル、小笠原秀人氏
http://www.juse.or.jp/software/pdf/23spc_2.pdf

◆組込みソフトウェア管理者・技術者育成研究会（SESSAME）
～話題沸騰ポット要求仕様書（GOMA-1015型）第7版～
http://www.sessame.jp/workinggroup/WorkingGroup2/POT_Specification_v7.PDF