日科技連 ODC分析研究会 第2期報告会資料

活動報告(3)
ODC教育ガイドライン

Rev.0.71
2019/03/20

日科技連ODC研究会 運営委員
瀬能 芳幸
キヤノン株式会社

トピック

1. 教育ガイドラインチームの議論
2. ODCとは
3. ODC導入ガイド
4. ODC分類ガイド
5. ODC分析ガイド

1. 教育ガイドラインチームの議論

- 教育ガイドラインチームメンバーの構成

  プラント制御　組込みソフト　Web・ITシステム

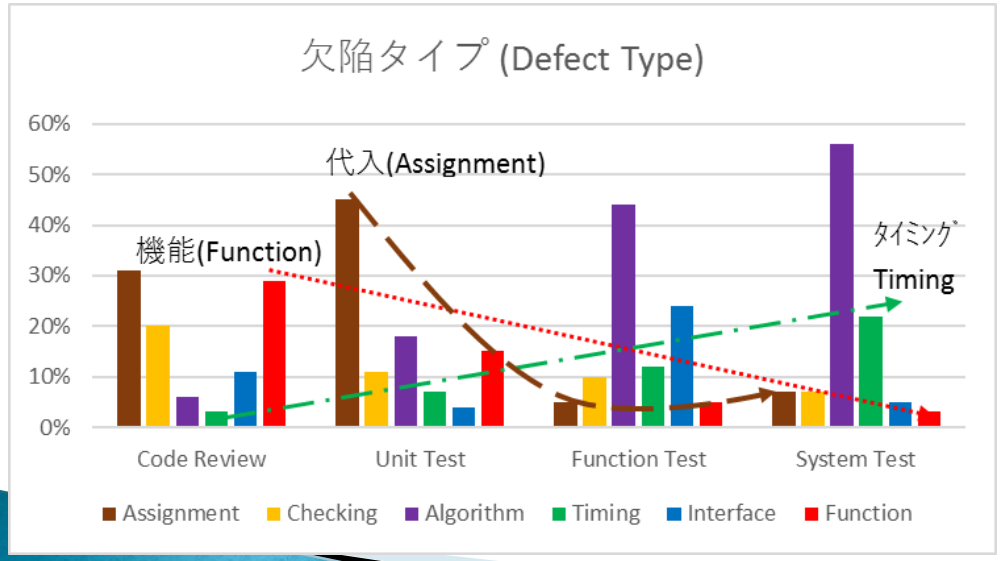
- 以下の様な議論がありました。
  - ODCを現場に導入するにはどうしたらよいか？
  - そもそも効果は？
  - ODC分類ガイドが欲しい。
  - ODC分析ガイドが欲しい。
  - テストを担当しているが、テストチームからどんな報告ができるか？
  - 導入するにはどんな苦労があるか？どんな準備が必要か？

2. ODCとは

ODCはバグ分析の一手法である。1992年IBMワトソン研究所のRam Chillarege氏により発表された。

信頼度成長曲線	直交欠陥分類	根本原因解析
SGRM	**ODC**	**RCA**
(Software Reliability Growth Model)	(Orthogonal Defect Classification)	(Root Cause Analysis)
● 収束を見る	● 即状況把握できる	● 真因を特定する
■ 具体的対策ない	■ 習得・導入が大変	■ 時間がかかる

Signature

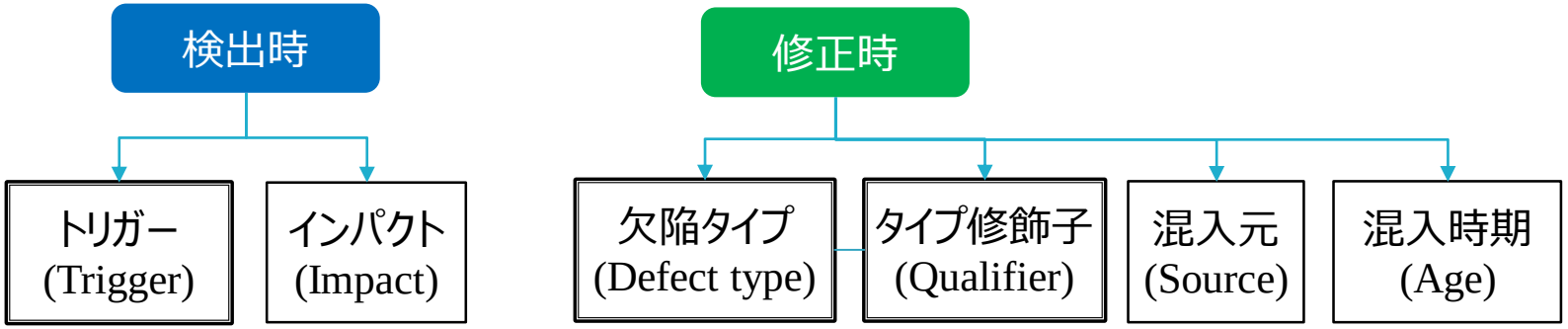


ODCは開発プロセスに着目し、あるべき状態と現在の状態を比較し、どのプロセスに問題があったのか目星をつける。タイムリーに対策を打つことができ、プロセス改善に役立つ。

2.1 ODC概説

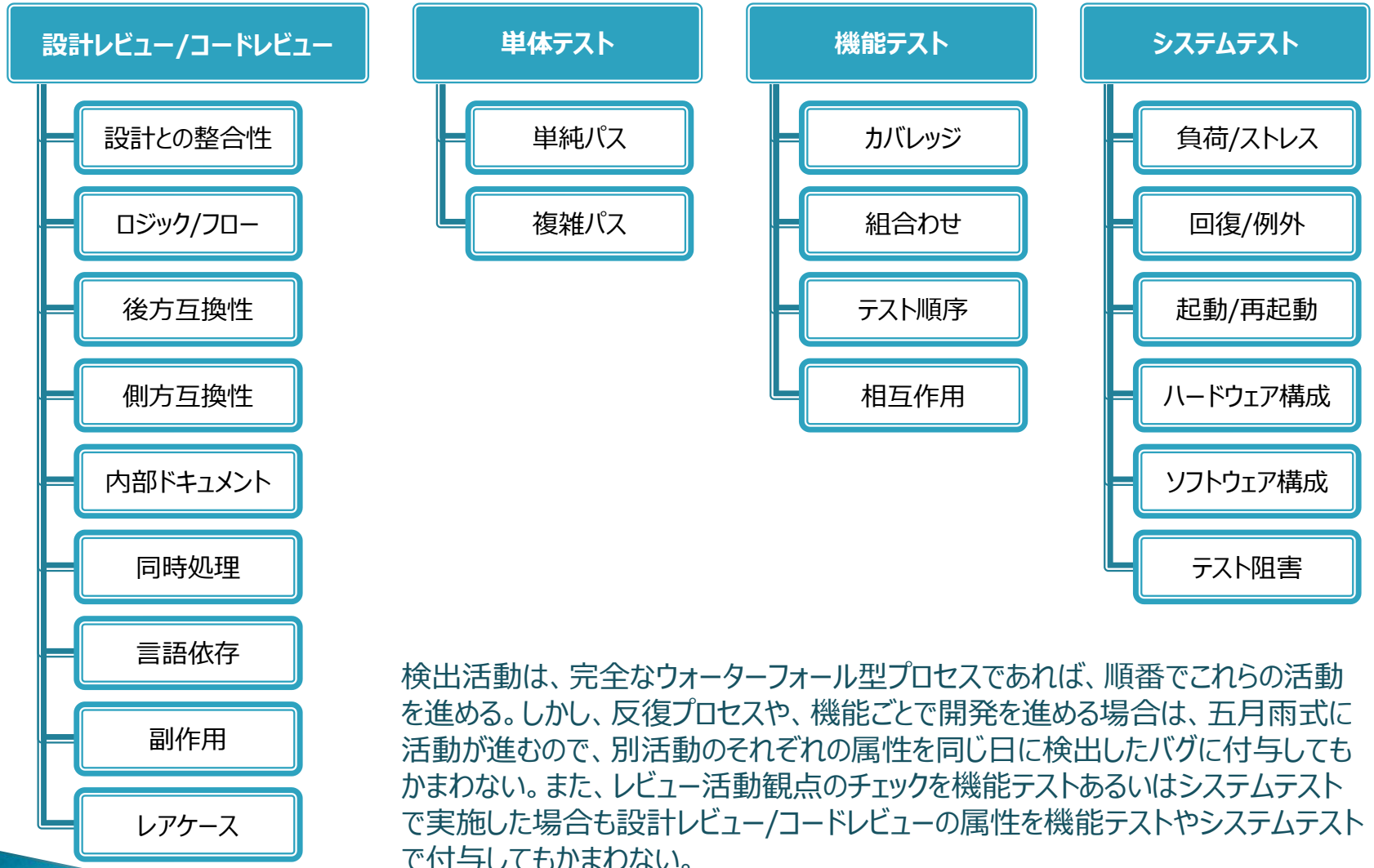
ODCはバグを複数の属性で分類し、フェーズごとや時系列でグラフ化しあるべき割合と比較することで、現課題の目星をつけ、対応策により改善する手法である。

- 欠陥タイプ(Type)とトリガー(Trigger)が代表的なODC属性である
- 検出時にトリガー属性を付与する
- 修正時に欠陥タイプ属性を付与する



2.2 ODC概説–トリガー属性

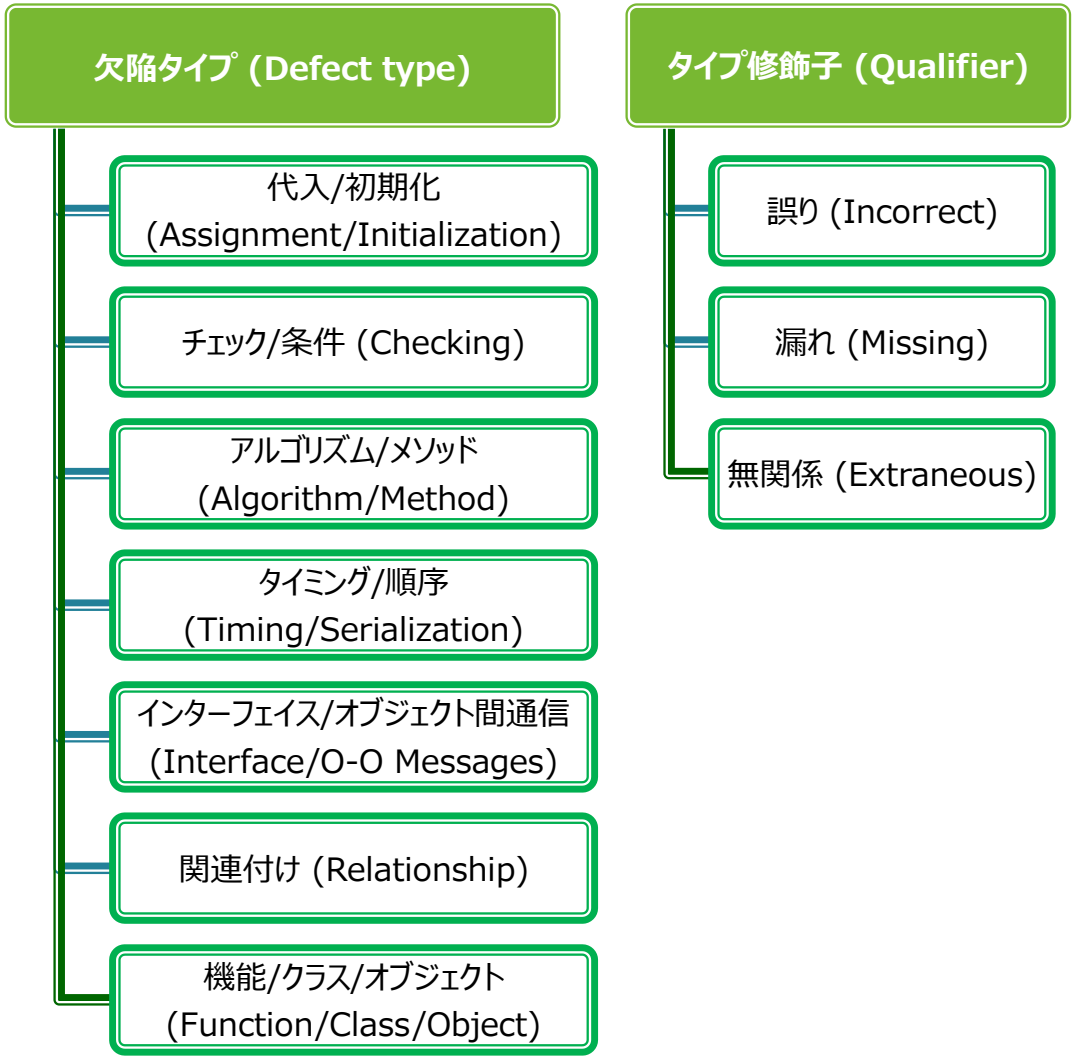
検出活動ごとに属性があり、どの活動を行っていて検出したかで決定する



検出活動は、完全なウォーターフォール型プロセスであれば、順番でこれらの活動を進める。しかし、反復プロセスや、機能ごとで開発を進める場合は、五月雨式に活動が進むので、別活動のそれぞれの属性を同じ日に検出したバグに付与してもかまわない。また、レビュー活動観点のチェックを機能テストあるいはシステムテストで実施した場合も設計レビュー/コードレビューの属性を機能テストやシステムテストで付与してもかまわない。

2.3 ODC概説–欠陥タイプ属性

欠陥タイプ属性は、どのように修正したかで属性を決定する。



3. ODC導入ガイド

ODCはIBMで開発され、ITT, AT&T, Motorolaなどで活用されている。
運用するには、ODC導入計画の策定が必要！

分類フローを参考にしても実際の
正解率は

**from 50%
to 70%**

分析を正しく行うには少なくとも
正解率は

**80%
or more**

必要！

- ネット上の情報を参考に分類し、グラフ化しても開発状況を読み解けない。
- 正しく分類しないと、分析できない。
- バグが多い場合（2000~6000件）、有識者で分類すると大変！

3.1 ODC導入ガイド–導入計画

導入する際、いきなりすべての属性を活用するなどと、欲張らないことが大切。テストに懸念がある場合は、トリガー属性から、設計・実装など品質作込みに懸念がある場合は、欠陥タイプ属性とはじめるなどスモールスタートがよい。

導入計画例：

日程	イベント	内容
第1週目	キックオフ	ODC概要の説明 自部門の製品課題の解決を目標とする
第2週目	過去バグ分析例説明会	過去の同系列のバグ分析を行いODC分析による課題があきらかになることを説明する。 次週からODCトレーニングを開始することを担当者に通知し、過去バグのODC分類を20件程度依頼する。
第3週目	第1回トレーニング	指導者と過去のバグをODC分類の答合せを行い、異なる部分を説明し合う。ODC分類を20件程度依頼する。
第4週目	第2回トレーニング	指導者と過去のバグをODC分類の答合せを行い、異なる部分を説明し合う。ODC分類を20件程度依頼する。
第5週目	第3回トレーニング	指導者と過去のバグをODC分類の答合せを行い、異なる部分を説明し合う。
第6週目	実際の製品への導入計画説明	バグ管理システムのどの欄にODC属性を記述するか決め、ODC属性付与の役割を定める
テスト中盤	現状の説明とODC分析例の説明	実際に活用し、課題抽出および分析・対策を実施する

3.2 ODC導入ガイド–分類トレーニング

ODCの分類トレーニングは実際のバグを分類するのが一番の近道である。

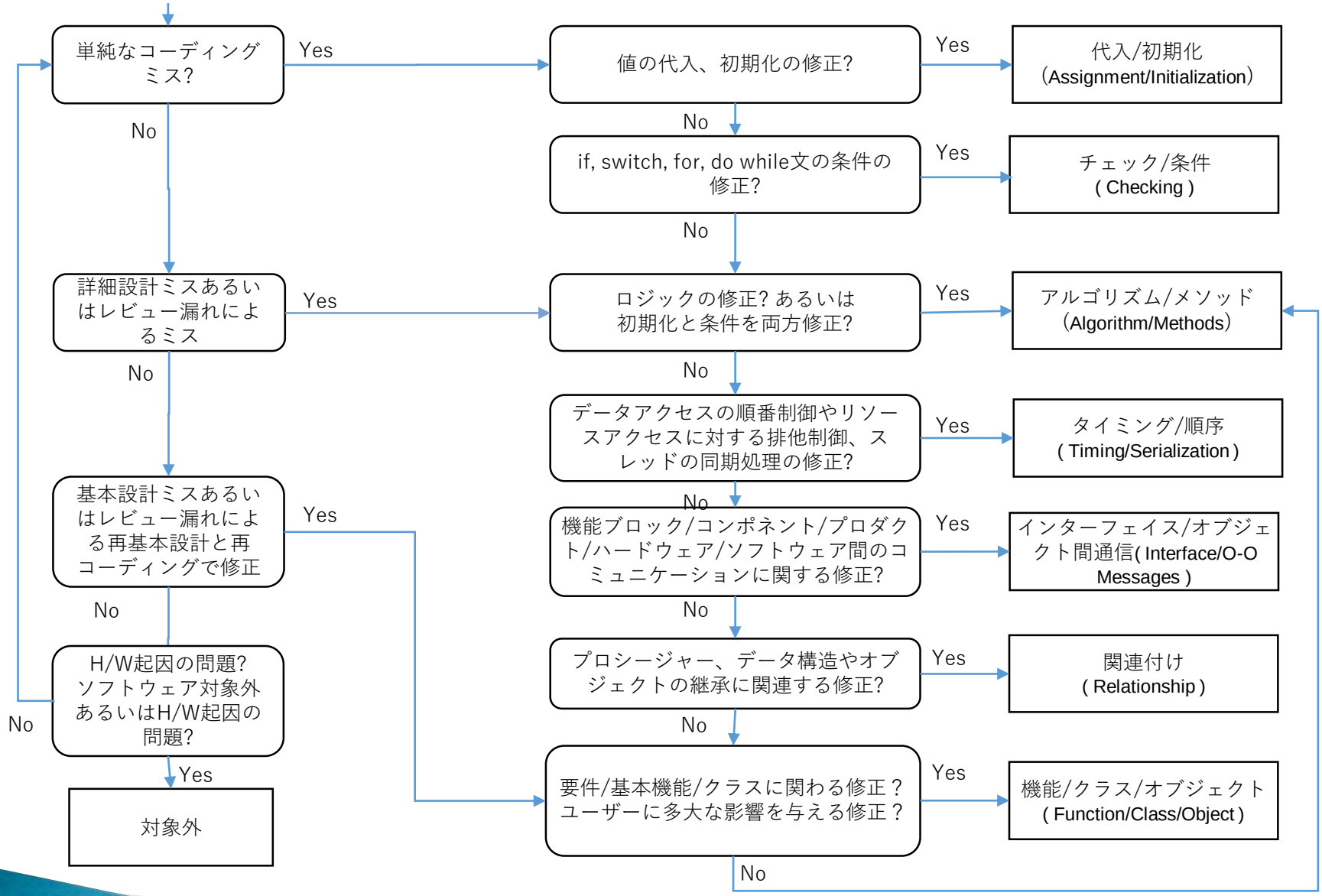
以下の様に実際のバグをあらかじめ分類し、見解が異なる部分を、どうして自分はそう思ったかを議論する。設計メンバーは、**実際の修正コードがどうなっているのかを説明の上**で、最終的には有識者の意見に集約するように行う。

				Aさん分類		Bさん分類		有識者分類	
番号	タイトル	原因	対策	欠陥タイプ	不備形態	欠陥タイプ	不備形態	欠陥タイプ	不備形態
01	アイコンがグレーでDisableのように見える	画像のみ非活性にしていた	ボタンと画像を非活性となるよう属性を修正	チェック	不正	初期化	不正	初期化	不正
02	スキャン中にキャンセルするとダイアログが表示される	原稿台と紙読込みの判定がまちがっていた	判別プログラムを修正した	チェック	漏れ	アルゴリズム	不正	チェック	不正
03	両面印刷時にカウント表示が正しくない	両面時ページカウントを+1としてカウントしていた	両面印刷時は印刷全体枚数が偶数場合1/2奇数の場合整数部を1/2+1と修正	機能	不正	チェック	不正	アルゴリズム	不正
04	大量印刷後に再び印刷するとメモリーリーク	2回目の印刷時ハンドルがなく、メモリ解放できていない	ハンドル情報がなくてもメモリ解放するように変更	機能	漏れ	初期化	不正	アルゴリズム	不正
05	印刷ジョブが多重の場合、連続スキャンになってしまう	２つのスレッドが同時に処理されている場合、スキャン状況が不正になっていた	多重処理の場合スキャン情報が排他するように修正	タイミング	不正	アルゴリズム	不正	タイミング	不正

4. ODC分類ガイド–欠陥タイプ属性

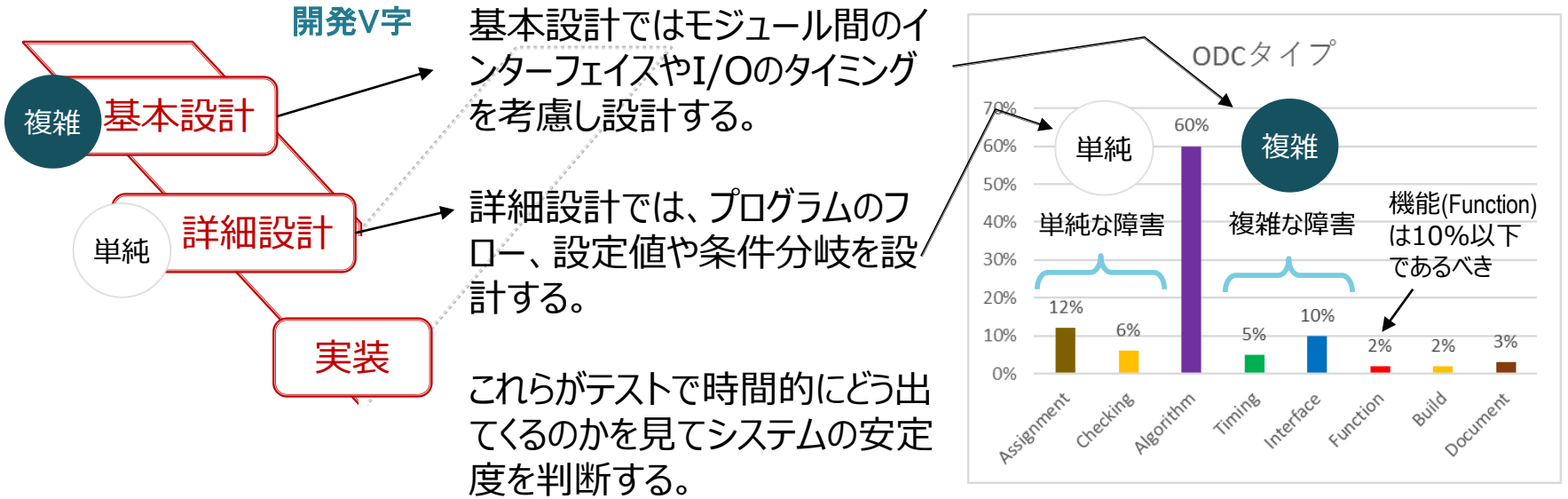
欠陥タイプ（Type）	説明（Description）
代入/初期化 （Assignment/Initialization）	(変数・定数への)値の代入の誤り、または初期化の誤り(Incorrect)、または欠如(Missing)
チェック/条件 （Checking）	If文などの条件判定文での、引数、データに対する判定の誤り、または欠如 (ただし、引数やデータの取得に関数が用いられる場合はアルゴリズムとする。)
アルゴリズム/メソッド （Algorithm/Method）	アルゴリズムの誤り、または欠如
タイミング/順序 （Timing/Serialization）	データアクセスの順番制御やリソースアクセスに対する排他制御、スレッドの同期処理の誤り、または欠如
インターフェイス/オブジェクト間通信 （Interface/O-O Messages）	機能ブロック間、コンポーネント間、プロダクト間、又はハードウェアとソフトウェアの間のコミュニケーションに関する誤り、または欠如
関連付け （Relationship）	プロシージャー、データ構造やオブジェクトの継承()に関連する誤り、または欠如
機能/クラス/オブジェクト （Function/Class/Object）	要求仕様との不整合、機能性、ユーザインターフェイス又はグローバルデータ構造の誤り、または欠如

4.1ODC分類ガイド–欠陥タイプ分類フロー(例)



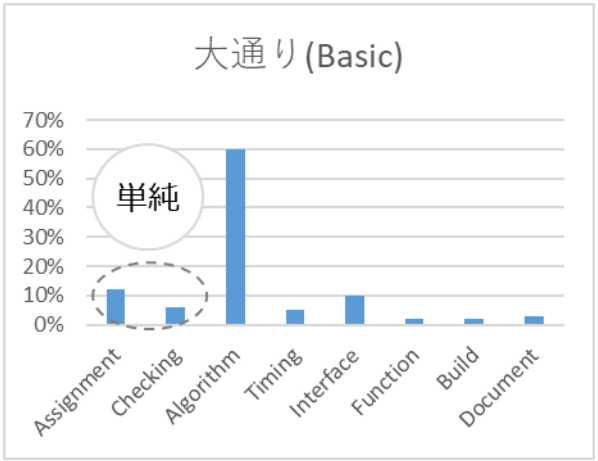
5. ODC分析ガイド–基本的な考え方

ウォーターフォール型の場合、大きく捉えると基本設計から詳細設計の不備またはレビューの不備により起きる「複雑なバグ」は、インターフェイス、タイミングとして分類される。詳細設計から実装にかけての不備によるバグは、「単純なバグ」として、初期化、チェックに分類される。

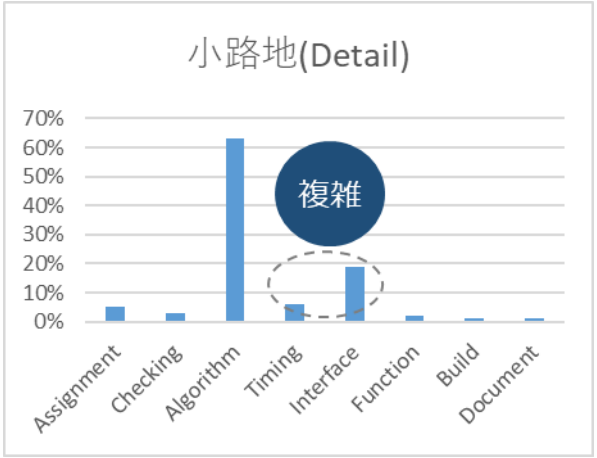


5.1 ODC分析ガイド–基本的な考え方

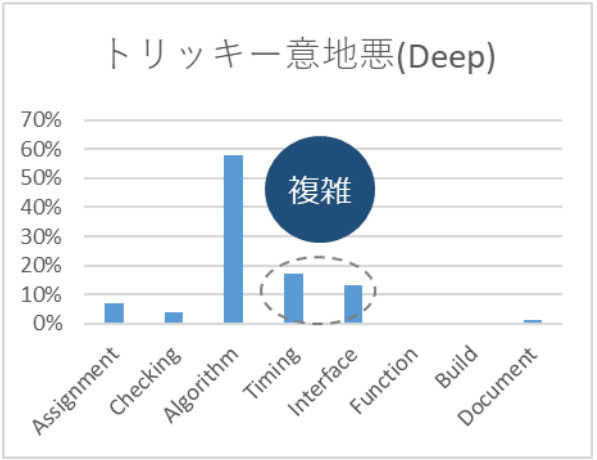
テスト全体を大きな流れで考えると、大通り(Basic), 小路地(Detail), トリッキー/意地悪(Deep), 最終的には収束を迎える.



大通り(Basic)テストでは単純なバグがでる。



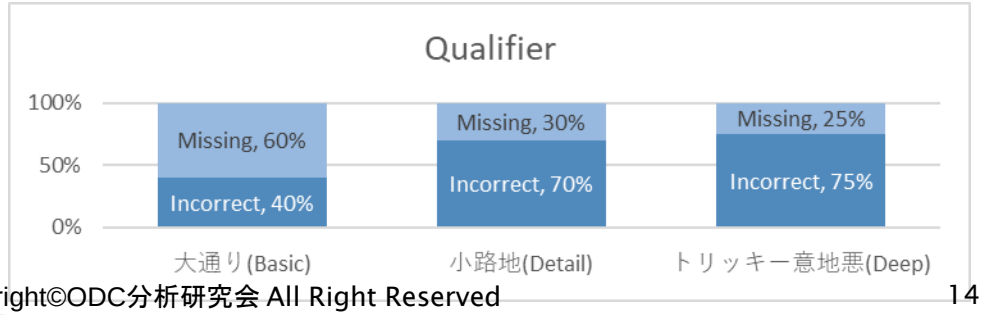
小路地(Detail)テストでは複雑なバグのうち、Interfaceが中心に出る。



トリッキ-/意地悪(Deep)テストでは複雑なバグのうち、Timingが出る。

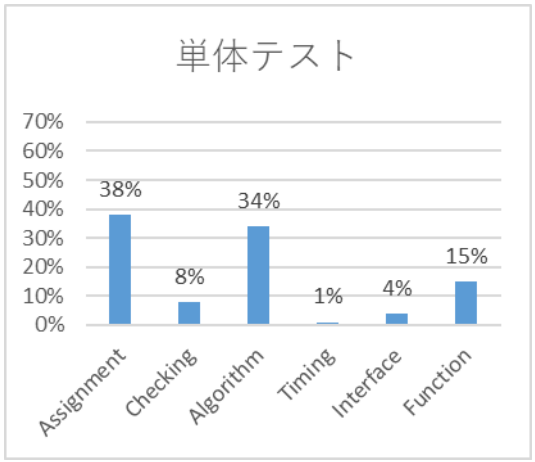
最終的には数的には減少するがAlgorithmが残るというパターンを基本として理解すると捉えやすい。

漏れ(Missing)は20~30%に減少して収まる。

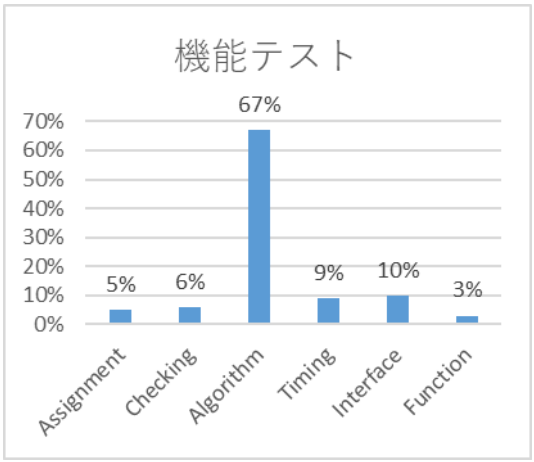


5.2 ODC分析ガイド–テスト工程別欠陥タイプ比率

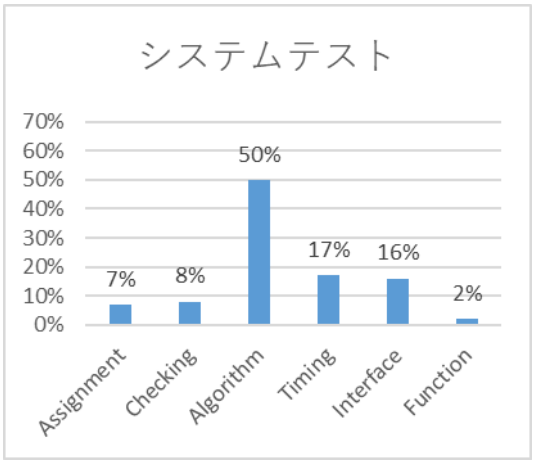
大規模開発の一例として複合機能コピー機のよい結果例を示す。



実際には単体テストでのバグ管理がされていることが稀なので「初期化」が38%を占めるグラフは珍しい。「機能」が15%を占めているが単体テストなので問題ではない。



「機能」が3%である。10%以上の場合は赤信号と考え、問題点を分析する。「初期化」が5%で理想形である。単体テストが完全な例は少ない。10から20%であれば単体テスト不備の分析が必要。「タイミング」「インターフェース」が10%出ている。全結合テストが適切に実施できている。



「機能」は2%で安定している。

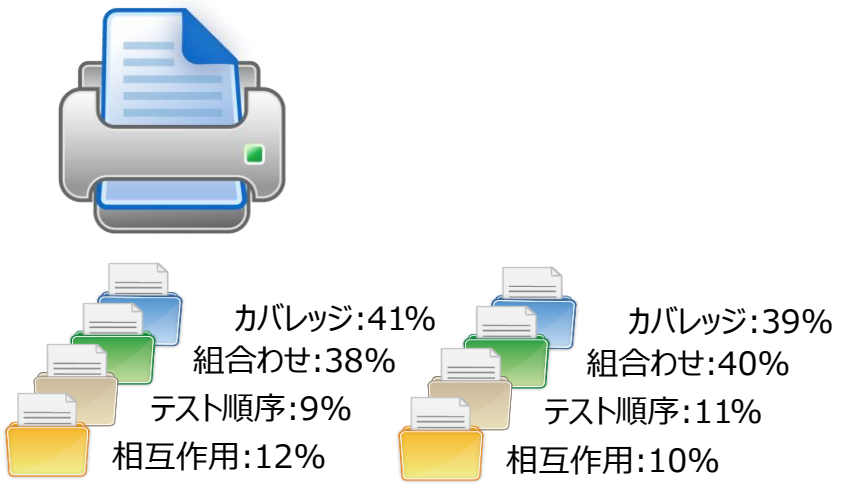
負荷や例外などのDeepなテストが実施できており、「タイミング」「インターフェイス」が16から17%を占めている。「初期化」や「チェック」も機能テストより若干増えている。

システムテストを時系列でみると「タイミング」「インターフェイス」も終盤は収束傾向がみられるべきである。

5.3 トリガー分析

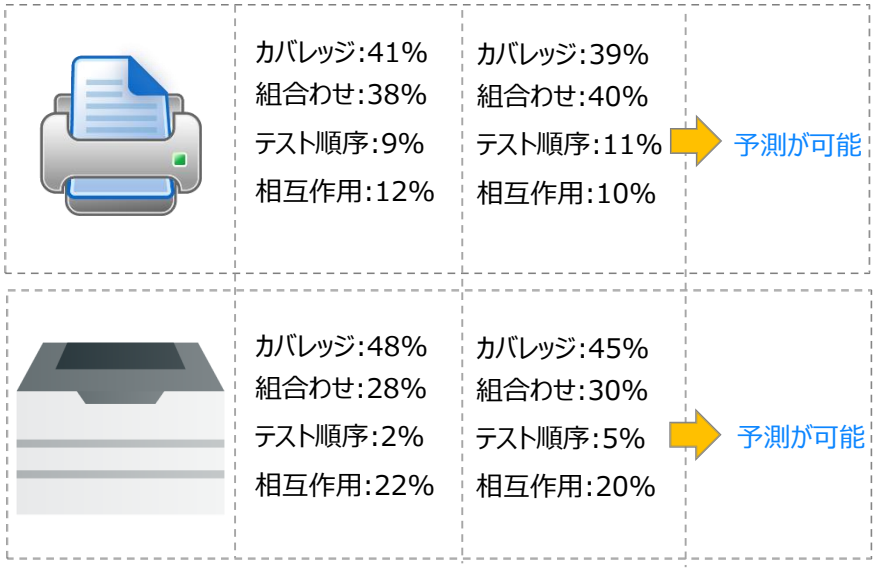
- 派生開発では、追加機能分のテストケース追加は必要だが、一連のテストケース群は再利用できる。従ってトリガー属性の分布もある程度固定される。

テストケースでトリガーが決まる



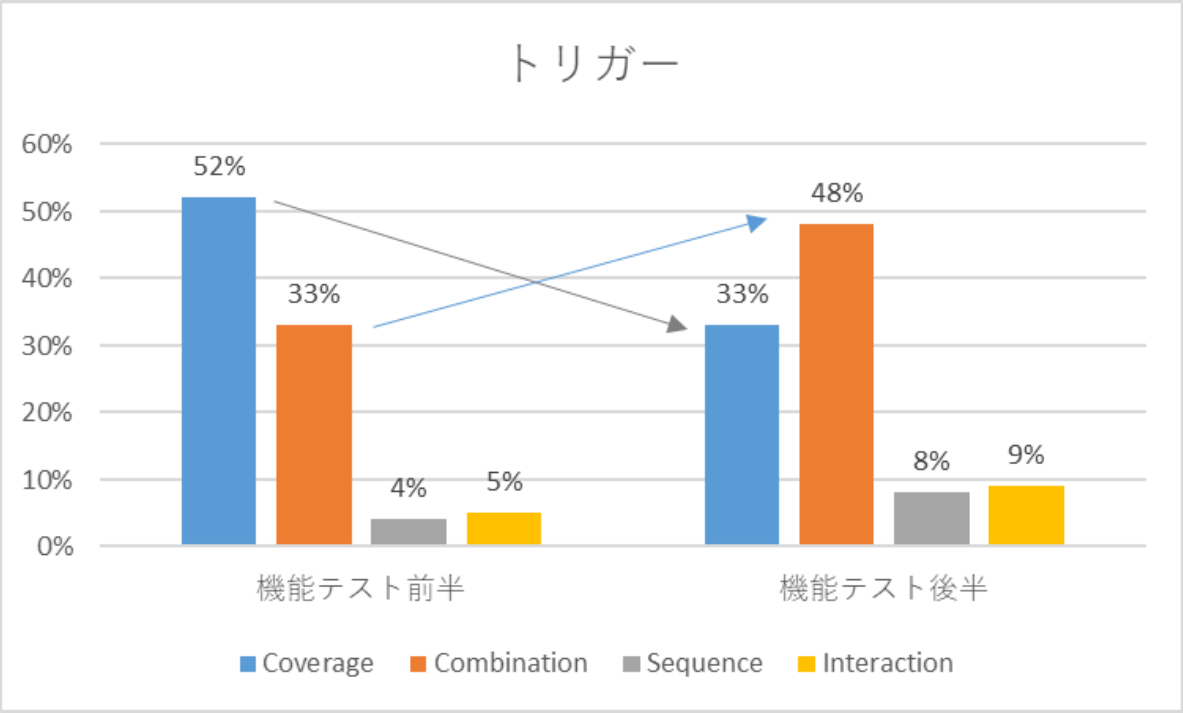
- 同系統の製品のトリガー分布を蓄積し、それらと比較することでより詳細な分析をすると精度が高いトリガー分析が可能である。

製品系列で傾向を分析



5.3.1 標準的な傾向

機能テストのトリガーはカバレッジが50%以上の高い割合を示し、組合せが30%強であり、後半、カバレッジは減少し、組合せがカバレッジを上回る。テスト順序や相互作用は多くの占めず、後半の方が増加傾向となるのが良い傾向とされる。



ご清聴ありがとうございました。