特別テーマ　活動報告

ODC分析実践のための
技術者育成コンテンツの実証実験

～　ODC分析実践の２つの壁を乗り越えるための**実践コース**　～

2025/3/14

ODC分析研究会　運営委員
杉崎眞弘

講師紹介：杉崎　眞弘　日本科学技術連盟　ODC分析研究会　運営委員

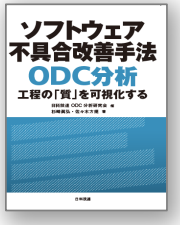
■プロフィール

会　社　名　SUGIシステムズエンジニアリング　代表
経　　歴　日本ＩＢＭ株式会社
　　独立行政法人 情報処理推進機構 (IPA)

・研究会への参加動機
IBM時代にDr. Ram Chillaregeから直伝を受けたODC分析のコンセプトとその分析理論、Know-Howを、もっともっと広く技術者の方がたに役立ててもらえるよう、普及に貢献したい。

・ODC分析の現状課題
「知る人ぞ知る」手法からの脱却を図る。

・これから学ぶ方々への期待
単に分析手法を学ぶだけでなく、そこに裏打ちされる開発プロセス、品質検証についての「心」と「確信」を共有することにも目を向けていきたい。

👉 

(ISBN978-4-8171-9713-9)

ODC分析の生い立ちと展開

■ 原典 (*1)

Orthogonal Defect Classification – A concept for In-process Measurement

Ram Chillarege, Inderpal S. Bhandari, Michael J. Halliday, etc., IBM Thomas J. Watson Research Center
IEEE Transactions on Software Engineering Vol .18, No. 11, Nov. 1992©IEEE

■ ODC分析手法の確立・社内講師育成 → IBM社内の品質検証・プロセス改善手法として展開

■ IBM社外のプロジェクトへの適用拡大

- 海外公開事例：
  Bellcore (AT&T Bell研究所）自社分析Toolへの組込み成果
  NASA ８プロジェクト（飛行システム）での改善効果
- 日本国内での適用支援 (2003 ~)
  電機、重工業、複写機、デジタル機器、医療機器、携帯電話など多くの企業プロジェクトに適用

IBMとして書籍を残さなかった・・・
新たに学ぶ手立てがない・・・

知る人ぞ知る手法？

■ **日科技連　ODC分析研究会発足 (2017/5~)**　**2024年度 第6期活動完了　2025年度 第７期準備中**

■ **ODC分析の解説書出版 (2020/8)**

「ソフトウェア不具合改善手法 ODC分析」
～ 工程の「質」を可視化する ～
日科技連出版　(ISBN978-4-8171-9713-9)

2025年度 年4回開催予定

■ **日科技連　ODC分析の基礎セミナー開催**（2021/07以降年2回開催 **これまで通算8回 + 企業向け1回 開催済**）

■ 「ODC分析の基礎」セミナーのカリキュラム

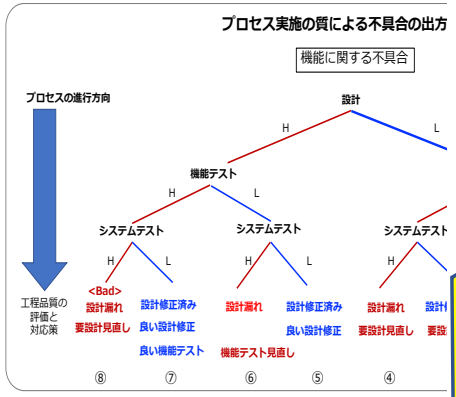
(途中、適時休憩を入れます。)

	時間	講義内容
第1日目	9:30 ~ 12:00	**セクション 1：概論「ODC分析とは」** ～ ソフトウェア開発実施の「質」を見える化する ～ ■ 開発現場でよくある判断基準 ■ ソフトウェア開発の見える化に必要なこと ■ 既存の不具合分析と異なるODC分析のアプローチ
	12:00 ~ 13:00	昼食休憩
	13:00 ~ 17:00	**セクション 2：ODC分析のコンセプト** ～「不具合を抑制する」～ ■ 「不具合を抑制する」ことによる改善への道筋 ■ 不具合について見えてきたこと ➢ 開発プロセスの観点 ➢ 不具合の持つ属性の観点 ➢ プロセス実施の質と不具合の関係 ■ ODC分析とその評価の「やり方」
第2日目	9:30 ~ 12:00	**セクション 3：ODC分析の適用事例研究** ■ 不具合属性にもとづいた分析・評価の事例考察 ■ 設計品質から示唆される開発の「やり方」への指摘事例 ■ テスト工程での適用効果事例
	12:00 ~ 13:00	昼食休憩
	13:00 ~ 17:00	**セクション 4：ODC分析評価の理論的裏付け** ■ 開発プロセスの定義と不具合との関係 ■ 開発プロセスの定義とODC分析との関係 **セクション 5：ODC分析に関わる開発プロセスについての考え方** ■ 開発プロセスの「心」について ■ 開発プロセスにおける「検証」について ■ 改善策策定の「やり方」事例(DPP)

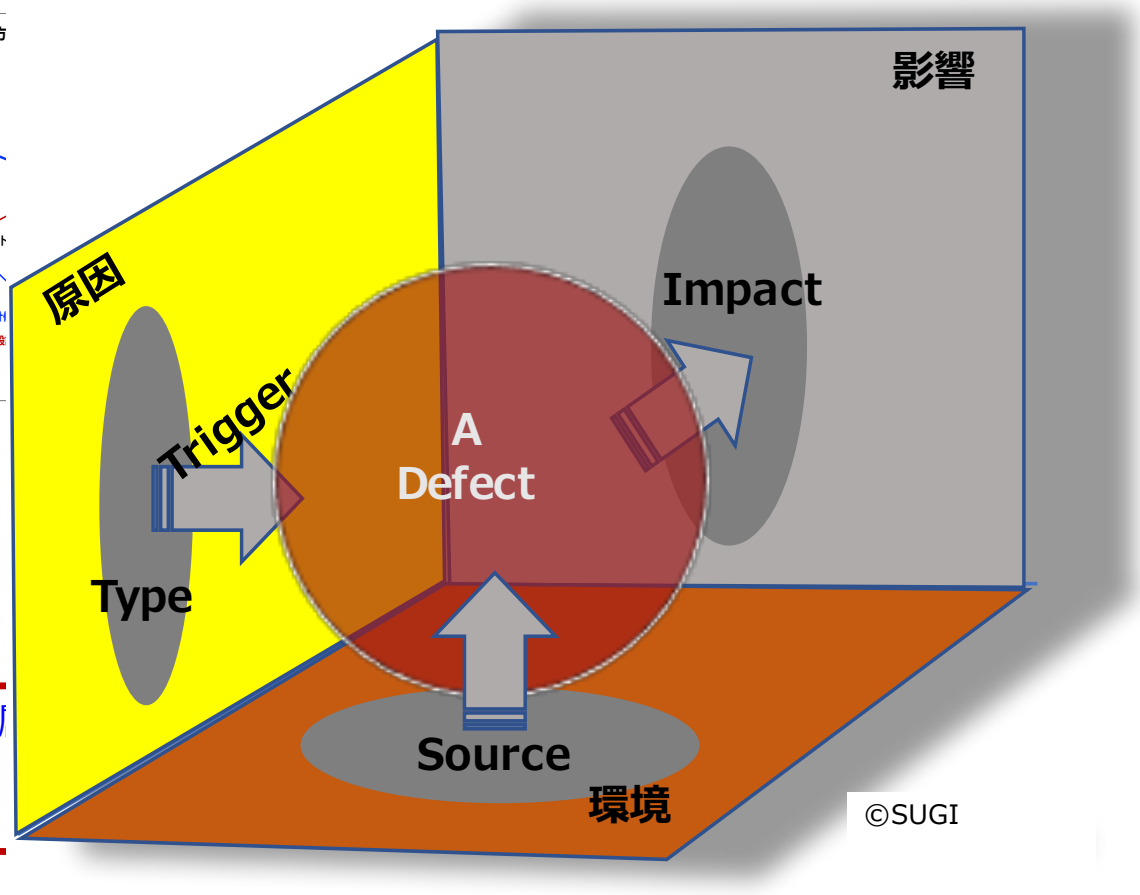
■ ODC分析のコンセプト

● 不具合について見えてきたこと　（３つの観点からの気づき）

① プロセス実施の観点



② 不具合属性の観点



©SUGI

③ プロセス実施の質と属性の出方の観点

工程	基本設計	詳細設計	コード	単体テスト	機能テスト 統合テスト	システムテスト
			X	X		
		X	X	X		
			X	X	X	
lize		X				X
		X	X	X	X	X
	X				X	

ー（検証項目）	経験	（新人/見習）	（製品開発経験）	（プロジェクト経験）	（システム開発経験）	Platform）（熟練/権威）
性を検証		×	×	×	×	×
仕様書						
仕様書						
キテクチャ設計書						
理解と検証			×	×	×	×
手順						
食合い						
生存性						
ョンとの互換性			×			
ムとの互換性					×	
ス						×
一貫性			×	×	×	×
依存性		×	×			×

気づきからの発案

不具合を

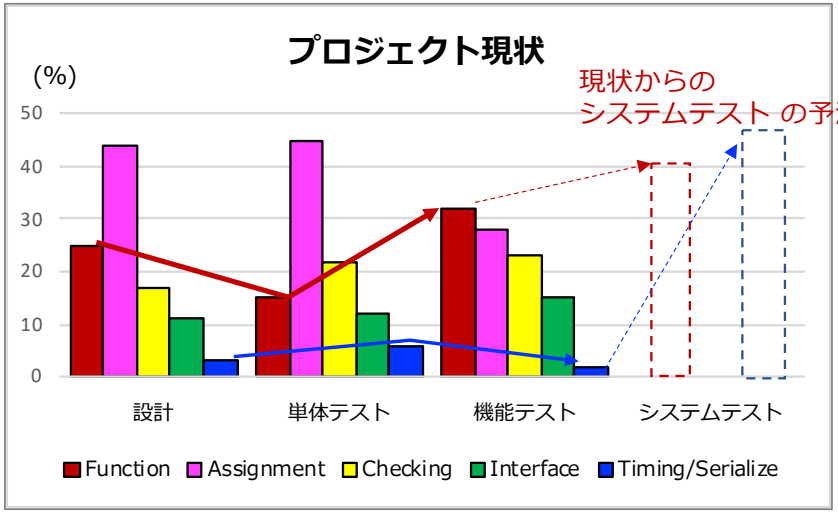
できるのでは・・・

■ ODC分析のコンセプト

発案：ODC分析の実施と評価の「やり方」

- ここの不具合を４つの属性の観点で分類し、各属性に分類された件数の集計結果を可視化（グラフ化）する。
- 可視化（グラフ化）された属性の分布・推移を**開発プロセスの期待**と比較し、その差異を見つけ出す。
- そして、**その差異の分析から示唆される必要改善策**を導き出す。

現状 = 何かおかしい？



「属性分布のあるべき姿」と現状との
“ズレ”を発見し
示唆される改善点を考察する

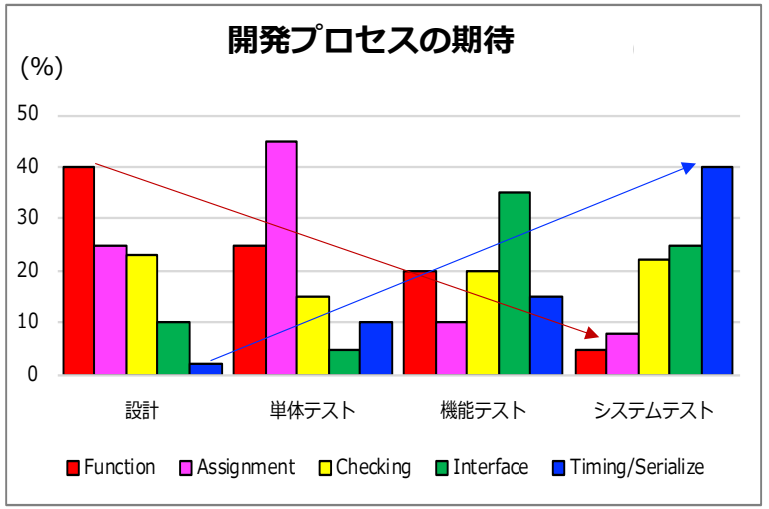
↓

示唆される工程への**改善策を**
導き出して、実施する

↓

狙い：開発プロセスの期待に
近づくことで“効果あり”

あるべき属性分布の姿＝開発プロセスの期待
プロセスシグネチャー



ODC分析とは（一言で言うと）

工程実施の妨げとなる**不具合を低減するために** (Defect control)

不具合の出方を分析することで

工程実施の「やり方」の質が見える化でき

必要なアクションが示唆される

定量的に

ソフトウェア不具合の定量的分析手法である。

■ これまでの研究会での事例研究から気づいたこと

➢ 第一の壁：「属性は何をつければ良いか？」・・・ **不具合分類や分析理論の理解不足、自己流**

分類の前提課題

- 工程、プロジェクトを跨いで混在する不具合を分類しようとしている・・・　分類単位不明 = 分類の意味なし
- 「とりあえずトリガー分析から始める」・・・　４つの属性は何のためにあるのか？ 手近に始めても事足りない
- 👉 **設計レビューの記録がないので**テスト工程だけでも・・・　え～っ！！諸悪の根源にたどり着けない

特に課題

- 不具合記述レベルがバラバラ・・・　不具合管理はできているか？　そもそも3大管理は回っているか

➢ 第二の壁：「属性分布から何がわかるの？」・・・ **属性分布が示すことが読み取れない = あるべき姿は？**

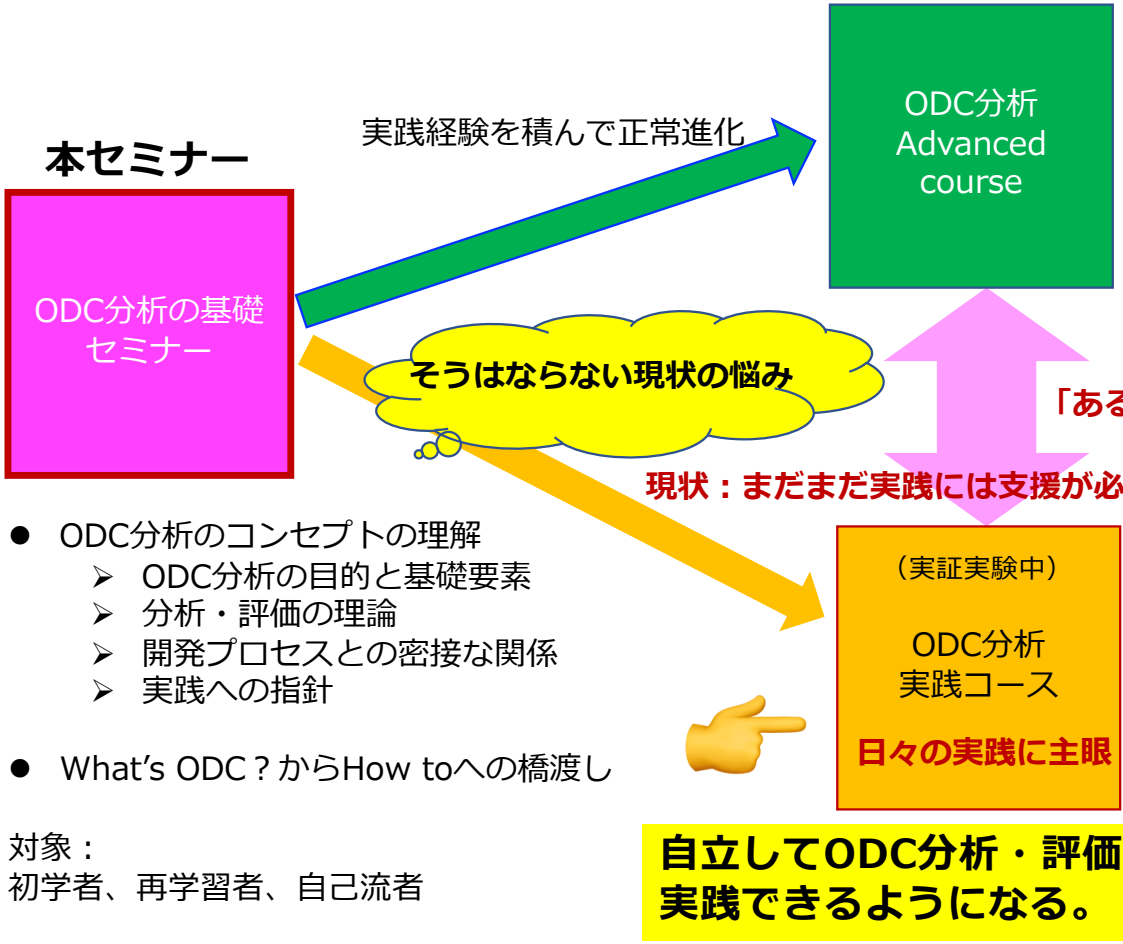
評価の実施課題

- 認識：実施すれば自動的に答えが出るものではない　**観察と考察が必要**
- 適用する開発プロセスの工程目的、作業定義、方法論を理解しているか？・・・ 何を目的にODC分析？
- あるべき属性分布のイメージはできているか？　・・・「何かおかしい」から「何**が**おかしい」への**洞察力**

ODC分析の基礎理論を一度は勉強しておきましょう！

気づき：ODC分析の学習ステップ

ODC分析の組織内定着・継承と分析・評価の研究を深化させる。



- ODC分析のコンセプトの理解
  - ODC分析の目的と基礎要素
  - 分析・評価の理論
  - 開発プロセスとの密接な関係
  - 実践への指針
- What's ODC？からHow toへの橋渡し

対象：
初学者、再学習者、自己流者

当初想定しているAdvanced course

- 分析・評価の「やり方」の深化
  - 分類結果の可視化バリエーション
  - 評価の着眼点
  - 改善策を導き出す洞察力の鍛錬
  - 説得力のある評価提示の仕方
  - 改善策の組織的策定・実施 (DPP)

対象：
実践経験者、品質管理者、組織責任者

- ODC分析の**「２つの壁」**を乗り越えるため
  - 第一の壁：**属性は何をつければ良いか？**
    → 分類の正確性を確保するには
  - 第二の壁：**属性分布から何がわかるの？**
    → 評価の仕方、着眼点を鍛える
  - 組織的な開発の**「やり方」の見直し**
    → プロセス実施のガバナンス点検
- 加えて
  - 必要な改善策を見出す力（洞察力）とは

対象：
自己流による挫折者、悩みを抱える実施者

実践コースのテーマ

■ 認識しているODC分析実施への２つの壁 を乗り越える

聞こえてくること：ODC分析は何か良さそう・・・でも、やろうとすると

疑問1
実際にやるには、どうしたらよいの?

疑問2
過去プロジェクトのバグを分類したが、あってる？

疑問3
たぶん適切に分類できた。どう分析して何をフィードバックすればいいの?

経験的に実施には次の**「２つの壁」**があると認識している。

第一の壁：「属性は何をつければ良いか？」

第二の壁：「属性分布から何がわかるの？」

再認識してもらいたいこと：

ODC分析は手段であって目的ではない。

す

👉 **実施すれば自動的に答えが出るものではない。**

実施結果が何を意味するか考察できる力（洞察力）を鍛えるべし

改善ができなければ意味がない

2024年度　特別テーマ

「ODC分析技術者育成のための実践コース」実証実験

概要

研究会として
有効性の実証実験

目的：ODC分析実施に際して必要な実践的知識・Know-Howを身につける

手段：　講義：これまで研究会で検討してきた実践に必要な知識・経験値を<u>カリキュラム</u>として提供する
WORKSHOP：<u>自社の不具合データ</u>を元に、ODC分析を実施し、その「やり方」を見直し・改善する

期待される成果：<u>自立して</u>ODC分析を実施できるようになる

参加者のメリット
プロジェクトに直結
した分析ができる

2024年度参加者：大手機器メーカーのシステムテスト部門の方々

「ODC分析技術者育成のための実践コース」実証実験カリキュラム(1H)（適時調整有り）

1H達成目標：自力で不具合属性の付与判断と属性分布の可視化ができるようになる。

	1H/2H	研究内容のテーマ	WORKSHOPのテーマ
準備回	1H	講義：ODC分析の理解の共有 • ODC分析についての共通理解の重要性 • ODC分析実施における２つの壁 ➢ なぜ不具合を前にして悩むのか？	WORKSHOP開催のガイダンス ① WORKSHOPの進め方 ② ODC分析に必要な準備の確認（各自） ③ 対象プロジェクトと管理の説明（各自）
第2回		講義：**第１の壁を乗り越える-1**：個々の不具合への属性付与 （属性は何をつければ良いか？） • 不具合属性定義の再確認 • 不具合属性の付与の手順と役割分担 (不具合分類の妥当性検証) • バリデーターについて	**WORKSHOP-1：不具合を分析・分類してみましょう** ① 不具合記述を読み解く：不具合の属性を見極める （属性の決め手となるキーワードを探る） ② 不具合管理のルールと不具合顛末記述の可読性 ③ 講評
第3回		講義：**第１の壁を乗り越える-2**：不具合の属性分布を可視化 (属性分布をどう表現したら良いか？) • 属性の集計に関する鉄則 • 事例から見る属性分布の表現の仕方	**WORKSHOP-2：不具合属性を可視化してみましょう** ① 日々の属性集計のやり方 ② 属性分布の可視化（グラフ化）個別作業と分類の見直し ③ 講評
第4回 **特別講義１**		**特別講義1：** **ソフトウェア不具合はなぜ起こる？** ➢ 不具合をちゃんと認識していますか？ VerificationとValidation ➢ 品質要求の変容という見方 **ODC分析とUSDMの親和性について** • USDMによる機能展開と設計レビュー • USDMを用いて不具合属性付与と集計	④ 各自の属性分布結果レビュー（発表と意見交換） ⑤ 効果的な表し方についての意見交換 ⑥ WORKSHOP-2の講評と2Hへのガイダンス

「ODC分析技術者育成のための実践コース」実証実験カリキュラム(2H)（適時調整有り）

2H達成目標：属性分布が示唆することが読み取れ、必要改善策が議論できるようになる。

	1H/2H	研究内容のテーマ	WORKSHOPのテーマ
第5回	2H	講義： 第２の壁を乗り越える-1：属性分布が示唆すること • 開発プロセスと不具合属性の関係 • プロセスシグネチャーについて	WORKSHOP-3：不具合属性分布を議論してみましょう ① 各自の適用する開発プロセスの概要説明 ② WORKSHOP-2で作成した各自の不具合属性分布について示唆されることと自身の見解をまとめる
第6回		講義：第２の壁を乗り越える-2：プロセスの期待に応える • プロセスの期待との差異の意味すること • 差異の原因を洞察する力	③ 各自がまとめた不具合分布と示唆されることの発表 ④ それに対する参加者それぞれの見解・意見を議論する ⑤ WORKSHOP-3の講評
第7回		講義：第２の壁を乗り越える-3：示唆される改善点と改善策策定 • 組織を巻き込む改善策策定 • 改善策策定のための事例 (DPP) 全課程を通しての講評と個別相談会受付	WORKSHOP-4：必要改善策を策定しよう（グループ討議と発表） ① WORKSHOP-3での議論をもとに必要改善策を策定し発表 ② 各発表に対する意見交換 ③ WORKSHOP-4の講評
第8回 特講2		特別講義：開発プロセスの心 ➢ 開発プロセスモデルの意味 ➢ 「やり方の質」の差が不具合になる ➢ 開発プロセスにおける検証	
第9回		講義：これまでの講義内容の振り返りと質疑応答 • 属性付与のポイント • 属性分布から示唆を読み取るには • 示唆されることを改善に結びつける	WORKSHOP-5：属性付与の妥当性確認 ① Step9不具合リストでの属性見直しの妥当性確認 ② 迷いやすい属性選択のポイント
第10回		講評：Workshopの振り返り これまでコメントしたStep5/7/9報告資料の再確認と次のステップへの提言	

第一の壁：「属性は何をつければ良いか？」を乗り越える

不具合属性付与の精度を上げる

1Hに予定する講義内容　第一の壁：属性は何をつければ良いのか？

講義１　ODC分析の理解の共有
(6/14)

1-1 ODC分析についての共通理解の重要性

1-2 ODC分析実施における２つの壁

1-3 なぜ不具合を前にして悩むのか？

講義２　第１の壁を乗り越える-1
(7/19)　個々の不具合への属性付与（属性は何をつければ良いか？）

2-1 不具合属性定義の再確認

2-2 不具合属性の付与の手順と役割分担

2-3 不具合分類の妥当性検証
- バリデーターについて

講義３　第１の壁を乗り越える-2
(8/23)　不具合の属性分布を可視化 (属性分布をどう表現したら良いか？)

3-1 属性の集計に関する鉄則

3-2 事例から見る属性分布の表現の仕方

特別講義 1
(9/26)　ソフトウェア不具合はなぜ起こる？
- 不具合をちゃんと認識していますか？
- VerificationとValidation
- 品質要求の変容という見方
- 設計品質　入り口で押さえるという考え方

ODC分析とUSDMの親和性について
- USDMによる機能展開と設計レビュー
- USDMを用いて不具合属性付与と集計

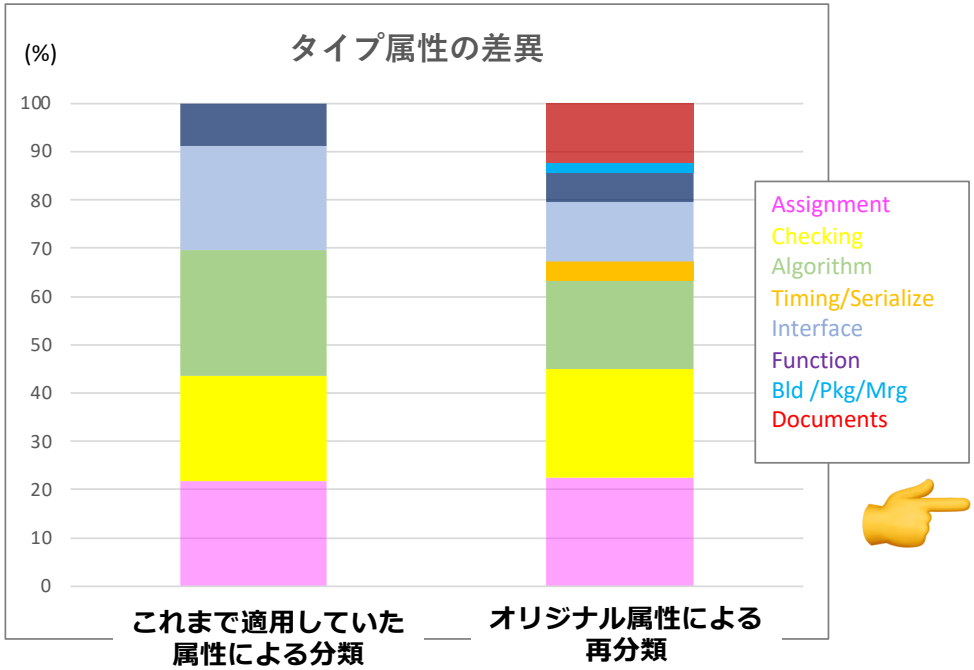
● 不具合属性付与の差異

不具合リストに付与されているタイプ属性を客観的に不具合記述のみから再度分類してみた。

◆ タイプ属性

属性付与に差異が生じている、
- **オリジナルで定義されたタイプ属性とは異なる属性が用いられている。（ v5.2なるものの属性が用いられている）**

Version N-1



- **差異が生じる理由**
  現行適用しているタイプ属性には
  - Timing/Serialize
  - Bld/Pkg/Mrg
  - Documents

  の分類がないため、他の属性に含まれていた。
  オリジナルでは GUIはInterfaceに含まれる。

分類後の属性分布評価において「然るべき属性分布」であるプロセスシグネチャーとの比較評価になるので、オリジナル属性の適用が重要です。

● タイプ属性がオリジナル属性定義と異なる属性が使われている。[*1]

現状　タイプ属性		オリジナル定義　タイプ属性	
10_値の代入/初期化		Assignment	値の割り当ての誤り、あるいは欠如
20_値のチェック		Checking	パラメータまたは条件文におけるデータの比較検証の誤り、あるいは欠如
30_アルゴリズム		Algorithm	アルゴリズムの誤り、または欠如
40_共有リソースのアクセス制御		Timing/Serialize	共有リソースの制御順に関する誤り、または欠如
50_インタフェース/メッセージ	80_GUI	Interface	ユーザー間、モジュール間、コンポーネント間、プロダクト間、またはH/WとS/Wの間のコミュニケーションに関する誤り、又は欠如
70_機能/クラス		Function	機能性、ユーザー・インターフェイス、あるいはグローバル・データ構造の誤り、または欠如
		Bld/Pkg/Mrg	ビルド・プロセス、ライブラリー・システム、または変更/バージョン管理に関する不具合
		Documentation	設計書、ユーザー・ガイド、導入マニュアル、プロローグ又はコード・コメントの誤り、または欠如
60_関連付け			

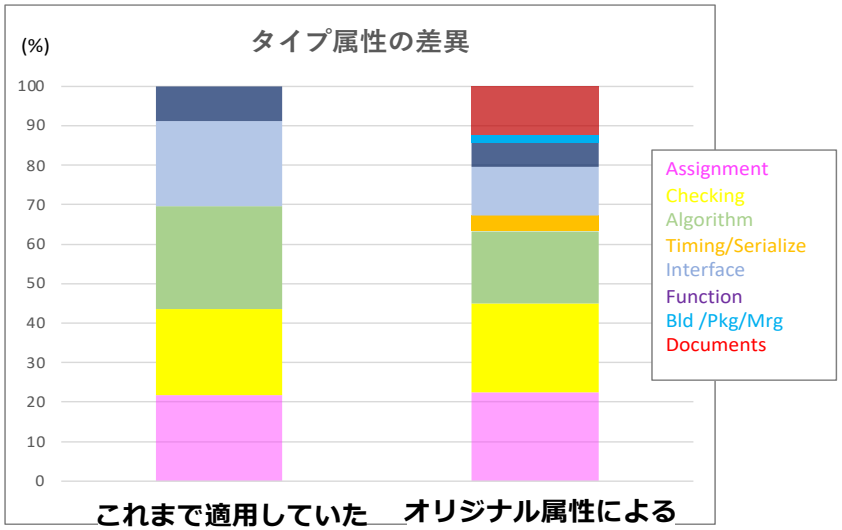
➢ **オリジナルのODC分析では理論的裏付けのある属性が定義されている。　今回オリジナル定義に読み替えて再分類した。**

オリジナルの定義

*1: Orthogonal Defect Classification – A concept for In-process Measurement
Ram Chillarege, Inderpal S. Bhandari, Michael J. Halliday, etc., IBM Thomas J. Watson Research Center
IEEE Transactions on Software Engineering Vol .18, No. 11, Nov 1992　©IEEE

タイプ（Defect Type）	意味の説明 （実際の不具合の分類は、修正者により修正された時点でなされる）
Assignment （値の設定）	値の割り当ての誤り、あるいは欠如
Checking （条件分岐）	パラメータまたは条件文におけるデータの比較検証の誤り、あるいは欠如
Algorithm （アルゴリズム）	アルゴリズムの誤り、または欠如
Timing/Serialize （タイミング・順序）	共有リソースの制御順に関する誤り、または欠如
Interface （インターフェース）	ユーザー間、モジュール間、コンポーネント間、プロダクト間、またはH/WとS/Wの間のコミュニケーションに関する誤り、又は欠如
Function （機能性）	機能性、ユーザー・インターフェイス、あるいはグローバル・データ構造の誤り、または欠如
Bld/Pkg/Mrg （ビルド・パッケージ・結合）	ビルド・プロセス、ライブラリー・システム、または変更/バージョン管理に関する不具合
Documentation （機能関連ドキュメント）	設計書、ユーザー・ガイド、導入マニュアル、プロローグ又はコード・コメントの誤り、または欠如
（注）	上記タイプのバリューを一つ選択するとともに、”missing 欠如”、または”incorrect 誤り” によるものかを識別子として付記しておく。

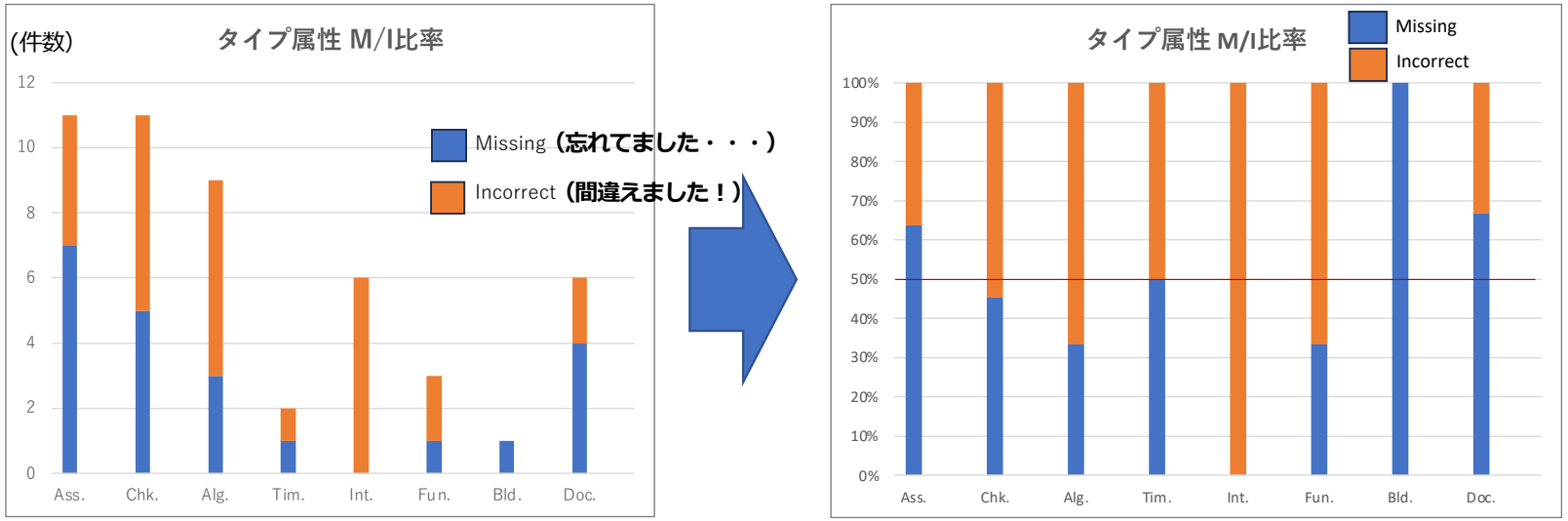
Version N-1



● タイプ属性分類において、M/Iが付与されていない（分類がされていない）

前述のオリジナルタイプ属性に基づいて、結合テストの不具合リストを再分類して、M/I分析を行った。

Version N-1



MS講評として：

- 結合テスト結果にしては、総じて**Missing (忘れてました・・・）の不具合が各タイプ属性において50%前後ある。**
  示唆されることは、
  - 直接には実装漏れで発見された不具合ではある。
  - 機能仕様書あるいは詳細設計書の記述漏れが原因であることから**（Doc属性 M 67%)**、詳細レビューが不十分を示唆。
  - そのことは、Algorithmの割合の多さからも裏付けられる。**Algorithmが多いのは、詳細設計レビューの不足を示唆する。**

● トリガー属性がオリジナル[*1]定義と異なり、テスト工程視点の考慮なく分析に使用されている

現状　テスト工程で適用しているトリガー属性

10_基本操作（標準状態）
15_基本操作（オプション等の変更した時）
20_操作順序に依存
30_複数機能の相互作用
40_負荷・ストレス
・・・
70_システム構成

対応

オリジナル[*1]のトリガー属性として定義されている然るべき工程

単体テスト	機能テスト（統合テスト）	システムテスト
Simple Path Coverage 単純パスのカバレージ		
Combination Path Coverage 組み合わせパスのカバレージ		
Test Coverage テスト項目のカバレージ	**Test Coverage** テスト項目のカバレージ	
Test Sequencing テスト実行順	**Test Sequencing** テスト実行順	
Test Interaction 相互間でのテスト	**Test Interaction** 相互間でのテスト	
Test Variation テストの多様性	**Test Variation** テストの多様性	
Side Effect 縁バグ	**Side Effect** 縁バグ	**Side Effect** 縁バグ
		Workload Volume/Stress 負荷テスト/ストレス・テスト
		Normal Mode 正常系テスト
		Recovery/Exception 復旧テスト/例外テスト
		Startup/Restart 起動時/再起動時テスト
		Hardware Configuration ハードウェア組み合わせテスト
		Software Configuration ソフトウェア組み合わせテスト

MSコメント

➢ レビューおよびテスト工程では、
それぞれに検証視点が定義されているはず。
現状混在したトリガー属性で評価して何が示唆されるか疑問である。

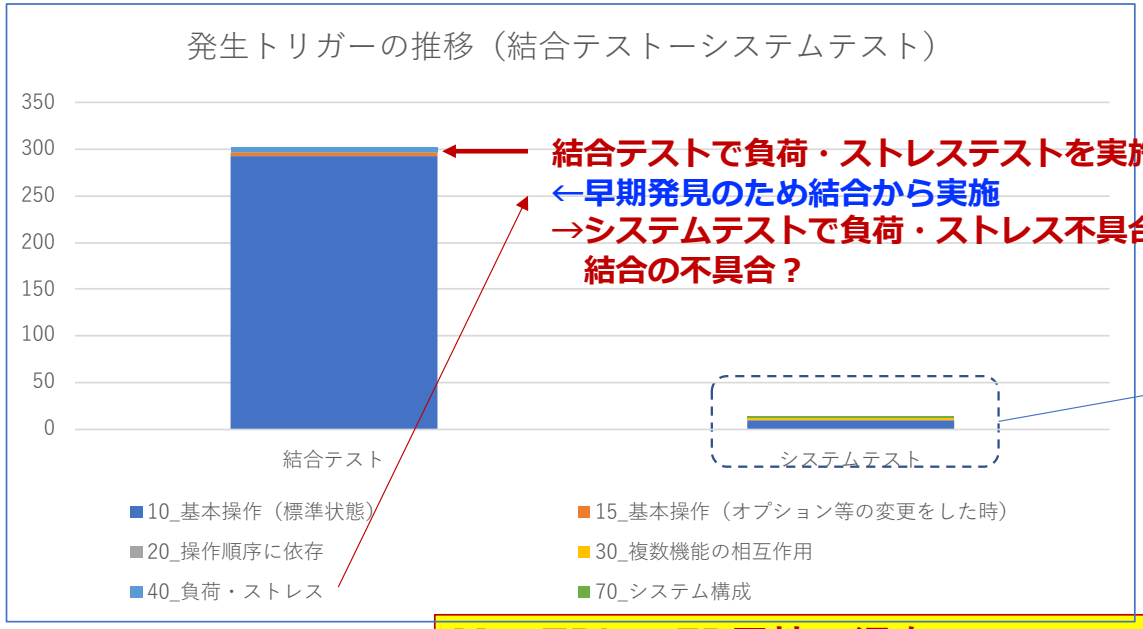
👉 **トリガー属性についても、
各工程で妥当な検証観点になっているかを
評価するためオリジナルのトリガー属性を
適用されることをお勧めします。**

オリジナルの定義

*1: Orthogonal Defect Classification – A concept for In-process Measurement
Ram Chillarege, Inderpal S. Bhandari, Michael J. Halliday, etc., IBM Thomas J. Watson Research Center
IEEE Transactions on Software Engineering Vol .18, No. 11, Nov 1992

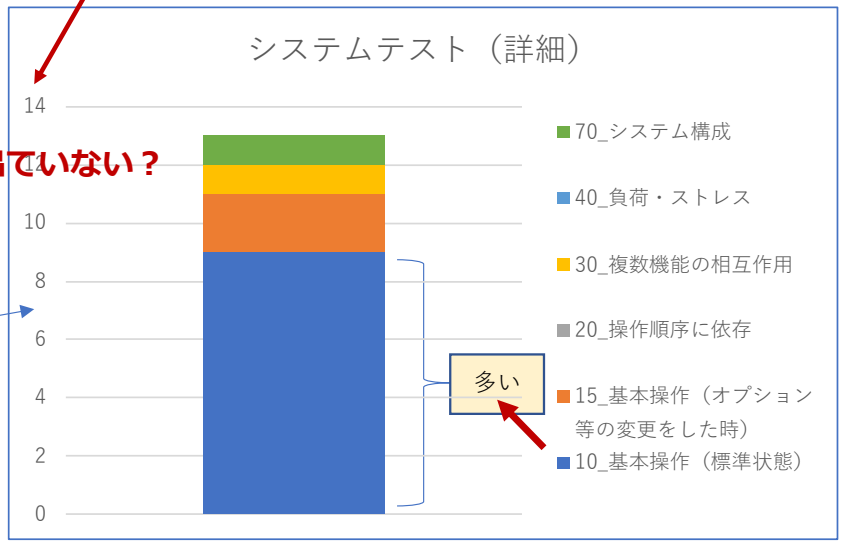
- **トリガー属性がオリジナル[*1]定義と異なり、テスト工程視点の考慮なく分析に使用されている**
  - **発生トリガーの推移（結合テスト・システムテスト）からテストの習熟度を確認する。**

Version N-1



結合テストで負荷・ストレステストを実施？
←早期発見のため結合から実施
→システムテストで負荷・ストレス不具合が出ていない？
結合の不具合？

%表示にすると拡大の必要なし

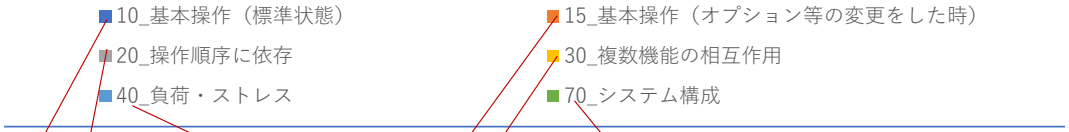


「考察」
95%以上の不具合を結合
工程でも発見されているの
ただし、結合テストで発
いると考えます。

MS: TRIGGER属性の混在：
オリジナルの定義では、結合テストとシステムテストとでは適用Trigger属性は異なる。
システムテストでも基本操作不具合が大部分を占めるが、結合と同質の不具合かどうかは
トリガー属性からはわからない。改善のためには、タイプ属性での分析が必要。

◆ トリガー属性の混在

● 結合テストーシステムテストで適用されているトリガー属性



● ODC分析の定義するトリガー属性

テスト観点・目的に応じて「然るべき」トリガー属性を区別している。

単体/統合テストでのトリガー	事象説明
Simple Path Coverage 単純パスのカバレージ	単純パスを一通り実行するテストを実施していた。
Combination Path Coverage 組み合わせパスのカバレージ	すべての条件分岐にもとづいて、可能なパスを一通り実行するテスト
Test Coverage テスト項目のカバレージ	それぞれの機能単位でのテスト（あらかじめ作成したテストケースの実行範囲）
Test Sequencing テスト実行順	異なる機能それぞれの実行順を考慮したテスト
Test Interaction 相互間でのテスト	相互作用をみるテスト
Test Variation テストの多様性	異なる入力項目（無効な入力を含めて）で実行するテスト
Side Effect 縁バグ	テスト目的外での、予期しない振る舞いの発生

システムテストでのトリガー	事象説明
Workload Volume/Stress 負荷テスト/ストレス・テスト	システム・リソースの限界付近、上限値/下限値でのテスト
Normal Mode 正常系テスト	システム・リソースの限界内での正常系テスト（ユーザーシナリオ）
Recovery/Exception 復旧テスト/例外テスト	例外処理/エラー処理テスト、それに伴う復旧処理テスト
Startup/Restart 起動時/再起動時テスト	通常システム/サブシステムの初期化/再起動テスト
Hardware Configuration ハードウェア組み合わせテスト	サポートしているH/Wとの組み合わせテスト
Software Configuration ソフトウェア組み合わせテスト	サポートしているS/Wとの組み合わせテスト

● 不具合属性付与の精度向上について 成果

Version N_結合テスト MS比較

Microsoft Excel ワークシート

いまいち・・・

Version N+1 不具合分析 MS比較

Microsoft Excel ワークシート

Good!

- ただし、AlgorithmとFunctionの区別についてはさらなる理解の深化を期待します。

Algorithm: 処理ロジックの「誤り」か「欠如」

Function: 要求機能の実現度合いが不完全な「誤り」か要求機能の設計・実装の「欠如」

ヒント

講義資料2より

■ 不具合記述の中から属性選定のキーワードを見つける

	タイプ副属性
①「～初期化されていなかった」「～の値が設定されていなかった（間違っていた）」	→ Assignment
②「～の条件が抜けていた（間違っていた）」「～の無効な条件文が設定されていた」	→ Checking
③「～処理のやり方が要求に合っていない」「冗長な処理になっていた」	→ Algorithm
④「～処理の手順が要求に合っていない」「～処理の完了までの待ちの時間が不足（設定していない）」	→ Timing/Serialize
⑤「APIの３つのパラメータのうち２つしか設定していなかった」「～の入力フィールドがプロテクトされている」	→ Interface
⑥「要求機能が作動しない」「要求されている機能が存在しない」	→ Function
⑦「仕様書の記述が要求と異なっている（無い）」「仕様書の記述が曖昧な（誤解を招く）表現になっていた」	→ Documentation

主語と動詞に着目 仕様記述のUSDM化が有効 USDMは、またの機会に・・・

第6期 ODC分析研究会 活動報告

2024 特別テーマ

不具合#	ステータス	調査：発生条件・原因	確認：確認結果・内容	ODC不具合タイプ	MS 見直し
101146	完了			70_機能/クラス	Assignment / M
101143	完了			70_機能/クラス	Assignment / M
100827	完了			50_インタフェース/メッセージ	Build/merge / M 詳細設計不十分
100801	完了			10_値の代入/初期化	Checking / M 詳細設計　　対象外処理が仕様漏れならFunction/ M
100214	完了			30_アルゴリズム	Checking / M　F→Cの変換を無条件におこなっていたのではないか？
100209	完了			30_アルゴリズム	Checking / M　F→Cの変換を無条件におこなっていたのではないか？
100206	完了			30_アルゴリズム	Checking /M 詳細仕様の M
100164	完了			80_GUI	Interface / I
100151	完了			80_GUI	Function / I　過去の仕様の実装
100091	完了			99_非不具合	Document / M 仕様書の修正もれ
100025	完了			99_非不具合	Document / M 設計を正とするなら仕様書の記載欠如
100024	完了			99_非不具合	Document / M 設計を正とするもデフォルトの意味になっていない、No problemでいいのか？
99569	完了			80_GUI	Interface / I
99545	完了			30_アルゴリズム	Assignment /M 初期化の欠如
99532	完了			20_値のチェック	Document / M 仕様書の更新漏れ
99457	完了			10_値の代入/初期化	Timing/Serialize /I 処理順番ではないか？
98937	完了			80_GUI	Assignment /M フィルター初期化の欠如
98850	完了			10_値の代入/初期化	Assignment /M 初期化の欠如 あるいはInterface / M 最大数表示の漏れ（出すの意味不明）
98818	完了			70_機能/クラス	Algorithm / I　重複時間処理の変更
98726	完了			10_値の代入/初期化	Document / I 仕様書「人検知」機能の削除漏れ
98723	完了			30_アルゴリズム	Timing/Serialize /M 前処理の完了確認欠如
98681	完了			50_インタフェース/メッ	Document / I 仕様書の記述誤り
98675	完了			70_機能/クラス	Checking /M タッチチェックイベントの検知欠如と理解
98672	完了			20_値のチェック	Function / I　Re-saize eventの有効な処理不足？

Version N

Version N+1

成果

分析研究

第一の壁：「属性は何をつければ良いか？」を乗り越える

不具合属性付与についての講評

- ４つの属性の意味の理解の深化が精度を向上させている。（単語の意味でなく、適用される具体的な不具合事象で理解すること）
- Version N-1, Nの時点に比べて、Version N+1の不具合リストのタイプ属性付与結果を見るに、**属性の意味と見分け方を再確認**することで、タイプ属性付与の精度は向上している。
（MSの見解と一致しているという観点で）
- より効果的な分析をするには、1件の不具合に対してオリジナル**４つの属性全ての観点**で付与・分類することが必要です。

  特に不具合修正者の方には、タイプ属性の**M/I識別**と**ソース属性**の付与をお願いしたい。
  今後の展開として、分類後属性分布をもとにしたプロセス改善への分析評価の仕方において必要となります。

第二の壁：「属性分布から何がわかるの？」を乗り越える

属性分布から示唆されることを読み取る

2Hに予定する講義内容　第二の壁：属性分布から何がわかるの？

講義 4　属性分布が示唆すること
(10/24)

4-1 開発プロセスと不具合属性との関係

4-2 プロセスシグネチャーについて

つづく

講義 5　プロセスの期待に応えるとは？
(11/22)

5-1 プロセスシグネチャーについて（つづき)

5-2 プロセスの期待との差異の意味すること

5-3 差異の原因を洞察する力

(12/20)

講義 6　示唆される改善点と改善策策定

6-1 組織を巻き込む改善策策定

6-2 改善策策定のための事例 (DPP)

(1/24)

講義 7　これまでの講義内容の振り返りと質疑応答

- 属性付与のポイント
- 属性分布から示唆を読み取るには
- 示唆されることを改善に結びつける

(2/21)

講義 8　Workshopの振り返り
これまでコメントしたStep5/7/9報告資料の
再確認と次のステップへの提言

(3/14)　**報告会**

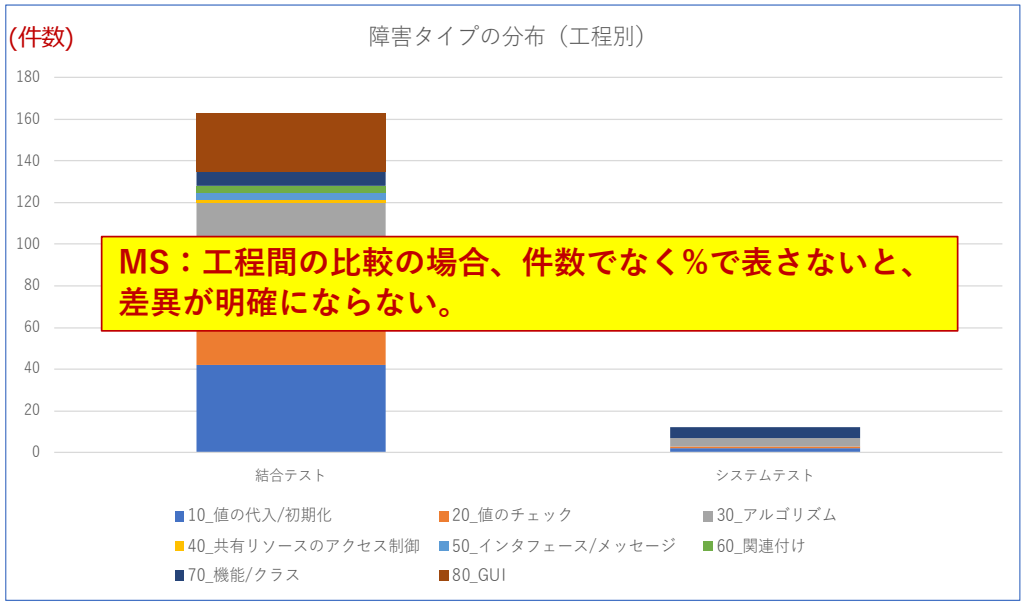
第5回　特別テーマ　Workshop資料

Version N-1 ~ N+1 までの分析評価の「やり方」「見方」の見直しと進歩

■ タイプ属性の分布[工程別]（結合テスト・システムテスト）

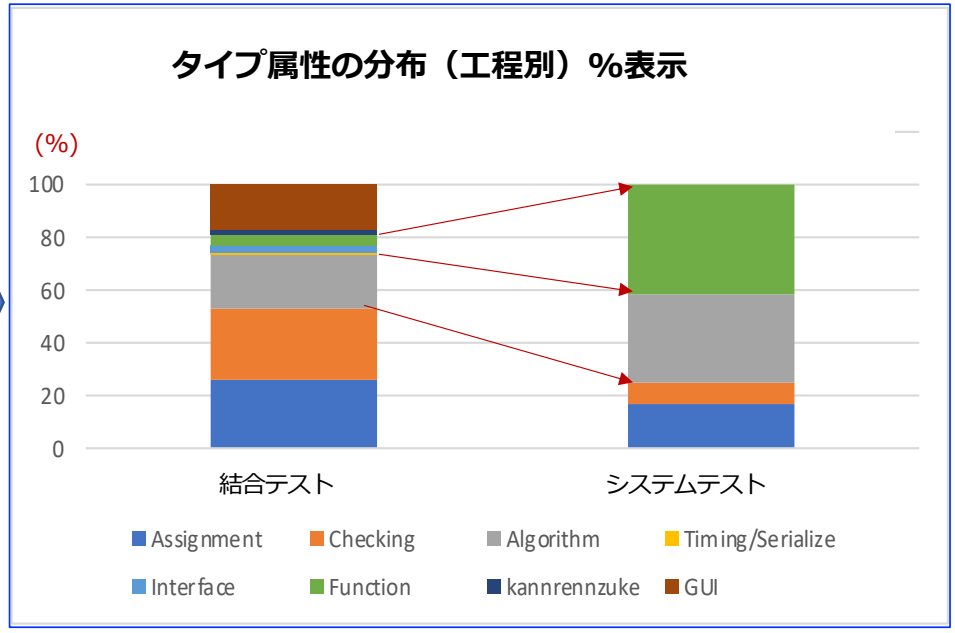
- 各工程で、然るべき不具合が検出されていることを確認する。

Version N-1



MS：工程間の比較の場合、件数でなく%で表さないと、差異が明確にならない。

属性分布を件数でなく、%で表示すると差異が顕著になる。



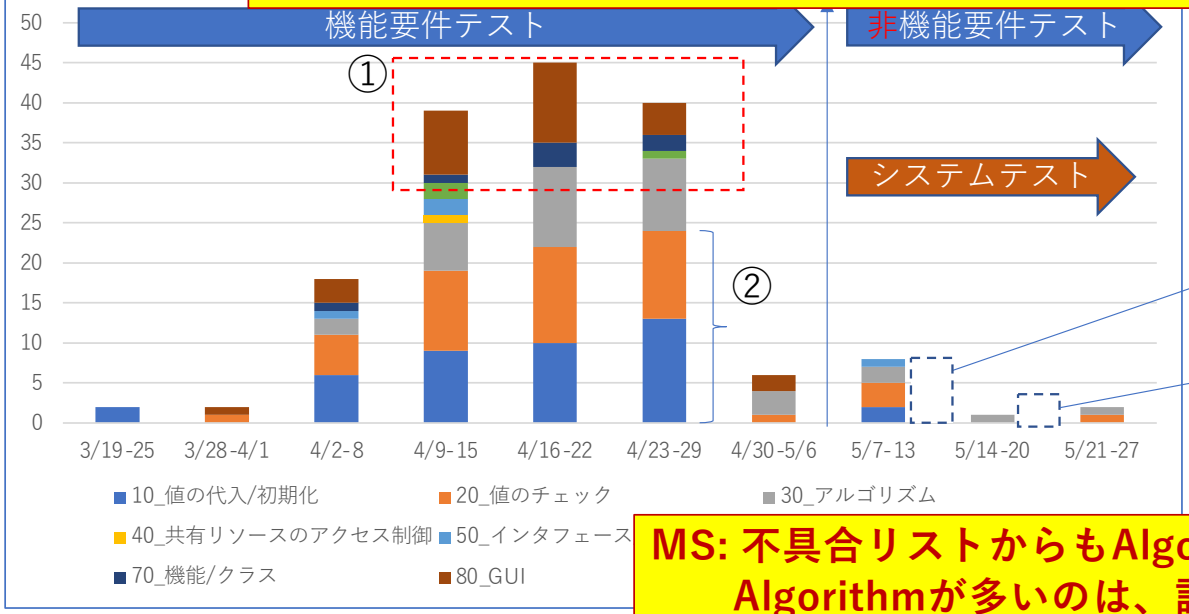
MS：システムテストでFunctionとAlgorithmが急増している。詳細設計の不十分さが露呈している。

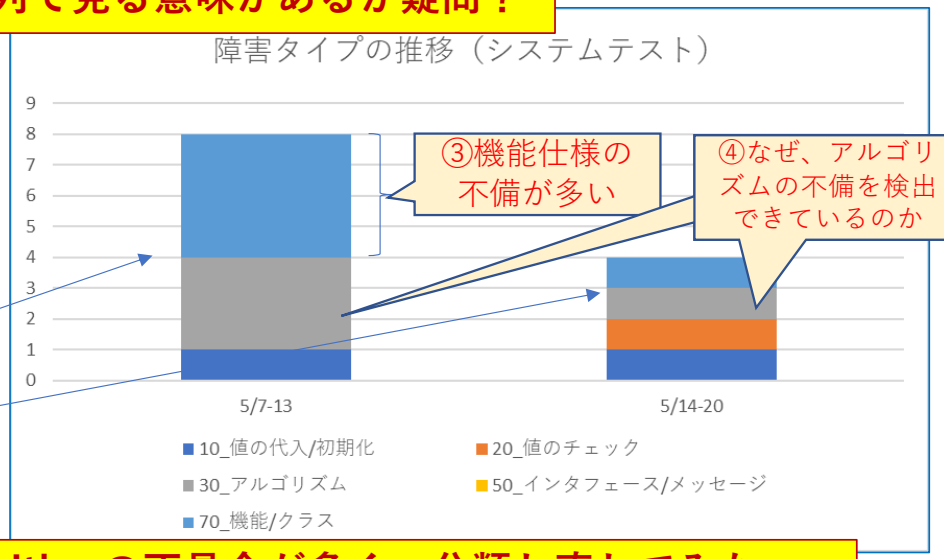
■ タイプ属性の分布[工程別]（結合テスト・システムテスト）

- 障

結合テスト工程内で、テストケースは何サイクル回すのか？
サイクル毎の期間で推移を見るなら成熟度評価に意味がある。
そうでないなら（1サイクルしか回さない）、時系列で見る意味があるか疑問？

Version N-1



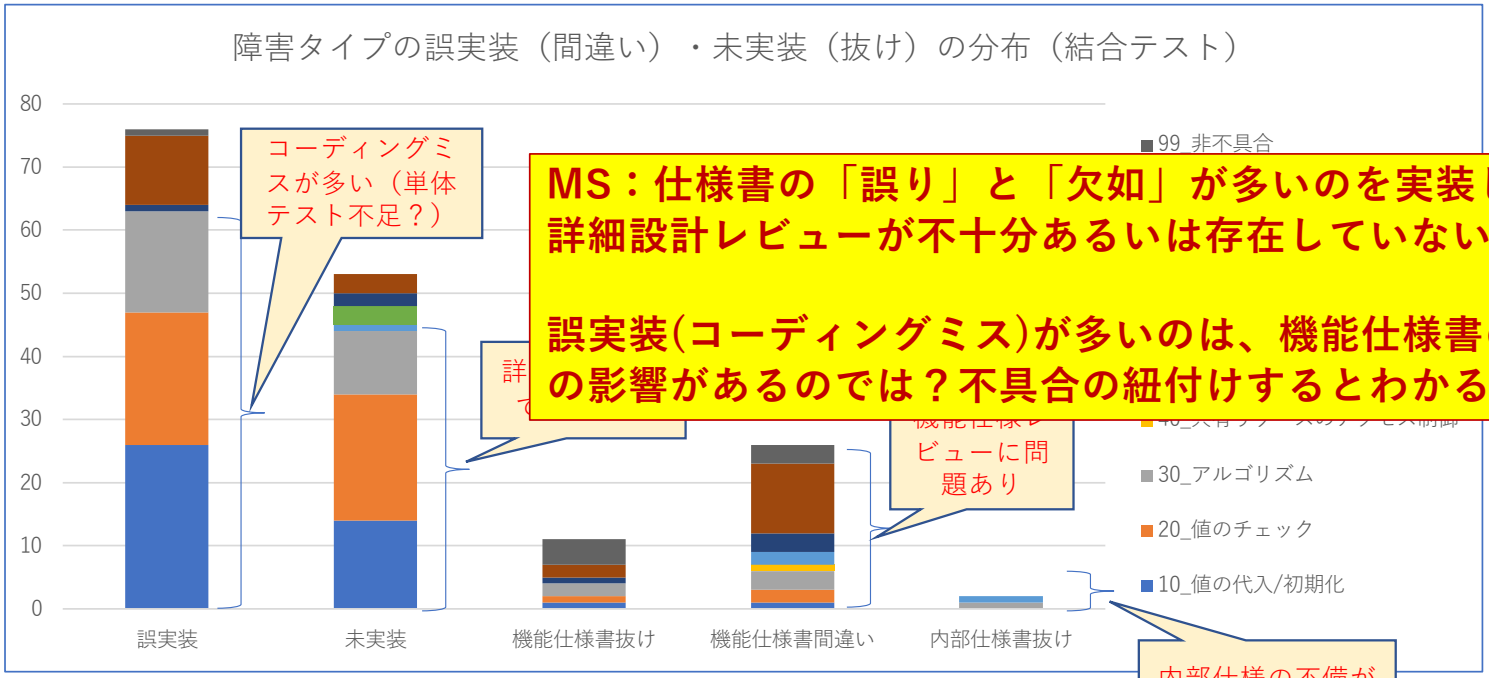


MS: 不具合リストからもAlgorithmの不具合が多く、分類し直してみた。
Algorithmが多いのは、詳細設計レビュー不足が原因と判定される。
特にAlgorithmのMissingが多いのは詳細レビューに原因がある。
Type属性分析には必ずM/Iの識別子をつける必要がある。

「考察」
①GUIの不具合は結合テストで出しきれ
②「値の代入／初期化」、「値のチェッ
結合テストはある程度機能していると考えます。
③システムテストでは機能仕様の不備（抜け・間違い）が多くなっており、結合テストで十分検出（指摘）できていない。
④なぜ、アルゴリズムの不備を検出できているのか。前工程では発見が困難なのか？（結合テストでは試験条件、組合せが足らないのか？）

■ 障害タイプごとの障害実装タイプより、設計プロセスの弱点を推測する。

Version N-1



MS：仕様書の「誤り」と「欠如」が多いのを実装して気付くのは詳細設計レビューが不十分あるいは存在していない証拠

誤実装(コーディングミス)が多いのは、機能仕様書の抜け、間違いの影響があるのでは？不具合の紐付けするとわかる。

「考察」
　実装ミスの場合でも、「詳細設計の誤り、抜け」ではなく実装だけを修正していることが殆どなのに違和感を感じます。詳細設計が十分できていますか？

MS: その通り！
違和感でなく、明らかに「やり方」に欠落がある。

■ 障害タイ…いることを確認する。

MS: システムテストでFunctionおよびCheckingが各々1/4も出るのは明らかに前工程の検証不十分の現れ。
レビュー項目、単体テスト項目の見直しが必要。

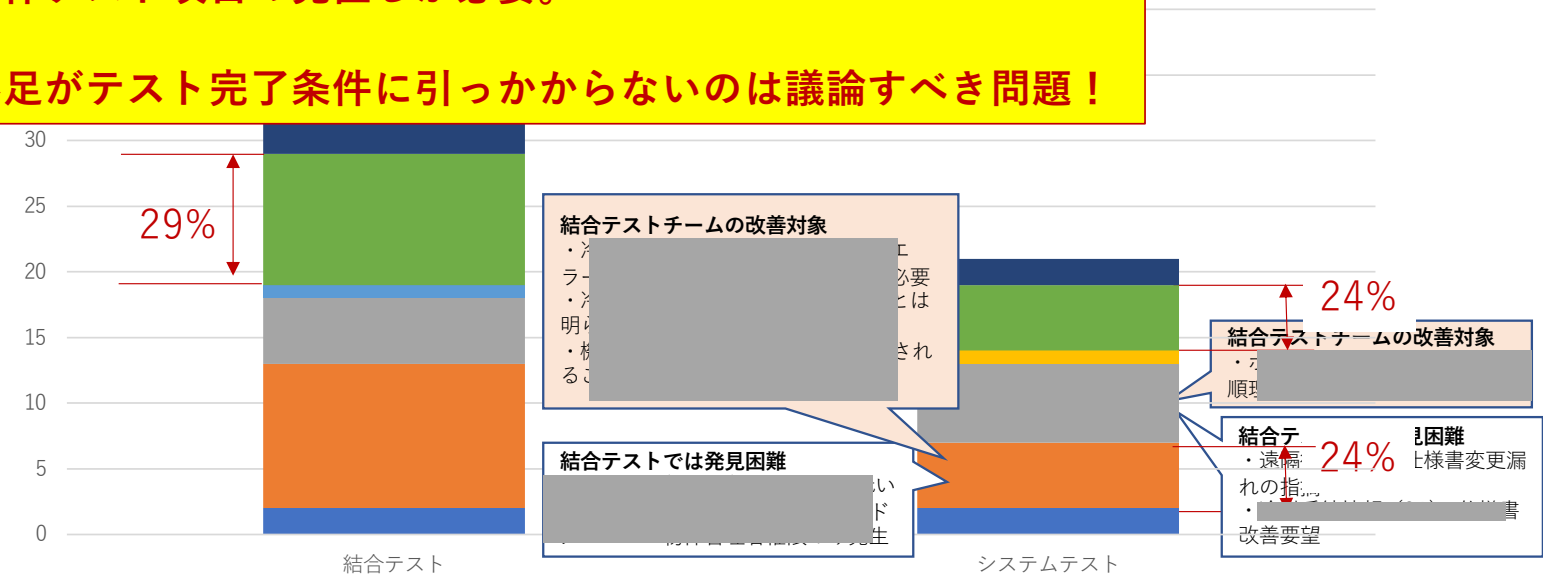
また、試験条件不足がテスト完了条件に引っかからないのは議論すべき問題！

Version N

結合テスト
①機能テスト
②システム結合テスト

↓

①機能の正確性、十分性
②インターフェース（機能間連携）、
（処理の）アルゴリズム
が見つかるべき



MS：工程間の比較の場合、件数でなく%で表さないと、差異が明確にならない。

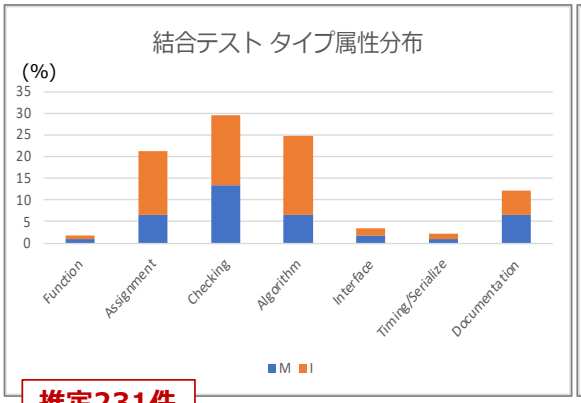
「考察」
障害タイプの分布が、結合テストとシステムテストでは大きな違いは見られなかった。
→「10_値の代入初期化、20_値のチェック」などの**実装単純ミス**や「70_機能／クラス」など**設計に関わる重大ミス**が結合テストで検出しきれていない。アルゴリズムに関しては、準正常系（誤って同じエアネットアドレスを設定した室外機など）の**試験条件が不足しているために見逃していると推測される。**

WORKSHOPを通じて
実践コース参加前（Version N-1, N)までの分析・評価の「やり方」の改善点を議論して
現行 Version N+1 での分析・評価を行うと・・・

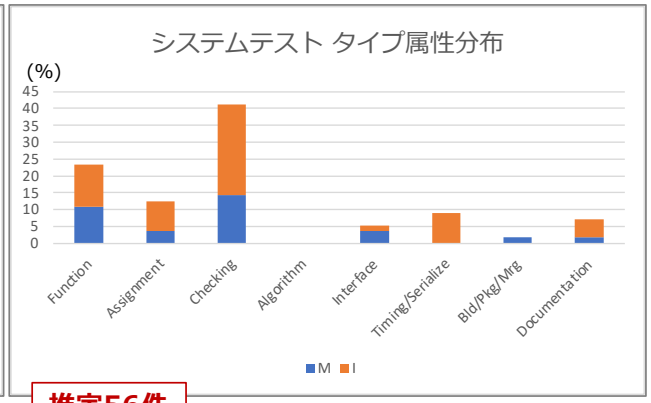
2．システムテストで発見した不具合分析（ODCタイプ属性）（1/2）

所見①③ タイプ属性分布が明らかに然るべき属性分布から乖離している。

Version N＋1



推定231件



推定56件

タイプ属性と開発プロセス工程との関係 （第5回講義テキスト参照）

工程 / タイプ属性	基本設計	詳細設計	コード	単体テスト	機能テスト 統合テスト	システムテスト
Assignment			X	X		
Checking		X	X	X		
Algorithm			X	X	X	
Timing/Serialize		X				X
Interface		X	X	X	X	X
Function	X				X	

自社でのシステムテスト評価

【課題】
- 不具合多い→開発工程からの不具合流出の削減
- 結合テストでのテスト漏れの削減

MS コメント

・結合テストの分布から見て、Assignment, Checking, Algorithmが多いのは、
単体テストの分布に近い。
このことは、**詳細設計レビュー/単体テストの不足、あるいは実施されていないことが示唆される。要裏付け**

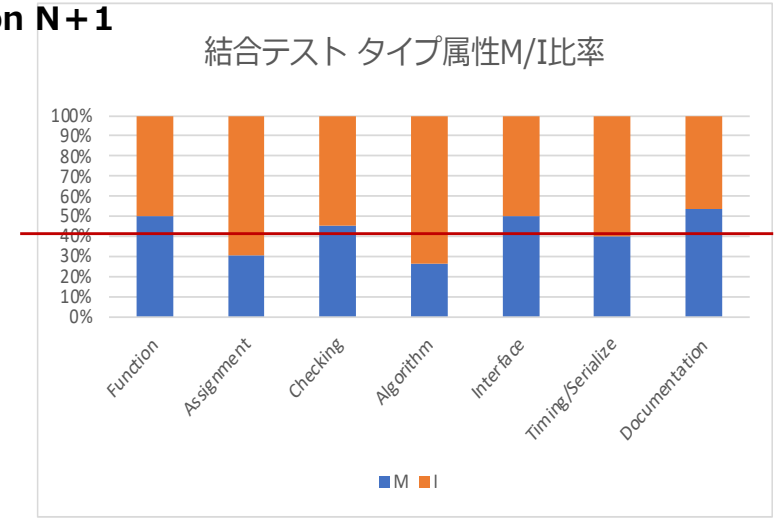
・結合テストからシステムテストに移行して、Function, Checkingの割合が増えているのは、
- 結合テストのカバレージが適切でないことが疑われる。→ **結合テストのカバレージを裏付け確認する必要がある。**
- あるいは、属性を選定した人が異なるのか、属性の理解が異なることが疑われる。**要確認**
- また、結合でAlgorithmが多かったのに、システムテストで0件なことも属性付与の適切さが疑われる。

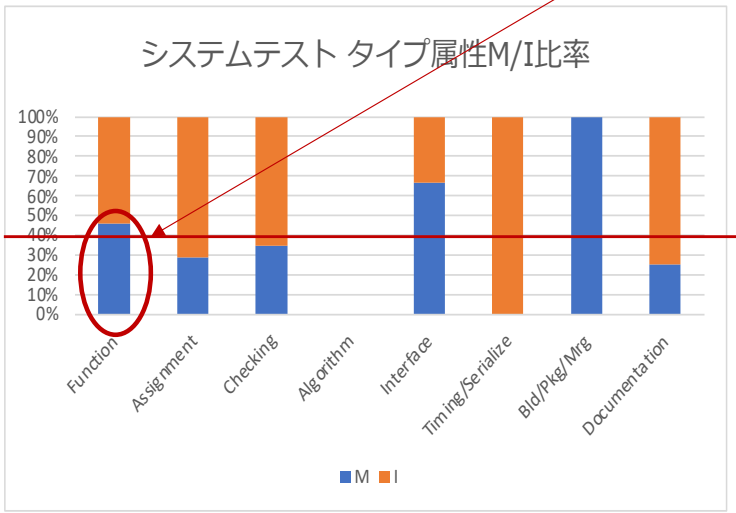
2．システムテストで発見した不具合分析（ODCタイプ属性）（1/2）

所見② 結合/システムテストという工程後半のテストにも関わらずMissingが多い。

特にFunctionのMissingは許せない。

Version N＋1





（Bld/Pkg/Mrgは除外）

MSコメント

・結合/システムテストでのMissingの出方を比較すると、だいたい４０%がMissingで結合で抑えきれずに“漏れ”が“漏れ”ている。
このことは、もともと仕様から“漏れ”ているから最終工程まで“漏れ”ていることを示唆している。
→ **Missingの不具合を仕様と紐付けて仕様の“漏れ”の事実を確認する必要がある。**（少なくとも結合工程でそのことを気づくべき。）

・ここから改善に結びつけるには、**設計レビューでなぜ“漏れ”が指摘できていなかったかを追求すべき。**
所見①にあるように特に詳細設計レビューの実態の的確性、十分性を検証すべき。

・さらに要求そのもののからの“漏れ”がある場合は、開発プロセスの要求工程での「やり方」の妥当性にも及ぶ。要裏付け

Version N+1での総評：不具合タイプ：Functionの内容 →機能仕様設計･レビューに課題あり 20024/12/24

1-1. 要求・要件を抜けなく設計できているか（網羅性）	
①要求からの展開漏れ、間違い	[redacted]
②要求としては明確に定義していないが当たり的仕様の展開漏れ、間違い	[redacted]
1-2. 要求を満たす妥当な設計になっているか（妥当性）	
①例外条件の検討漏れ	[redacted]
②要求の妥当性についての検討不十分	[redacted]
③要素間の仕様が不整合	[redacted]
1-3. その他	
①データモデル標準化チームとの連携不足	データモデルの定義を要件とした場合、データモデルの変更に追従できていなかった。(#107936)

MS：着眼点 Goodです！

MS: Version N+1の分析結果の総評

成果

1. タイプ属性の分類精度について
   - 分類精度は以前Versionに比べてかなり向上していると感じます。
     理由：結合／システムテストでの属性分布のシグネチャーが、絵に描いたような要求・設計工程でのレビューの漏れを示唆する単体テストのシグネチャーを示すことができているからです。
   - ただし、AlgorithmとFunctionの区別については、属性をつけた人の属性理解の違いによるのではないかと疑問が残ります。
     タイプ属性の選定で迷った場合、修正者から不具合修正理由・修正内容を特定してもらい判断すべき。

2. 分析評価と改善点抽出への着眼点について
   - 当初、前半のタイプ属性分類の示し方について、数値データを逆算して%分布で属性分布を再作成して示唆される問題点を考察したところ、
     - 要求・設計レビューでも漏れ、あるいは不十分さが示唆される
     - 加えて詳細設計レビュー・単体テストの不十分さで漏れが発見できず結合/システムテストでMissingとして出てきているという示唆を得た。
   - そのことが、読み進めた後半にある御社指摘事項・改善要求の内容がMS評価と一致していることが確認できた。Goodです！
   - タイプ属性分布から示唆されることの読み取りが的確（要求からの展開漏れ、間違い）にできていると考えます。

3. 改善策策定への改善点裏付け調査について
   - 具体的な技術事項についてはコメントできませんが、裏付け調査として
     - 要求分析・設計レビューでなぜ漏れが出たかの体制、やり方、レビュー観点などをTrigger属性、Source属性で分析するとより課題が浮き彫りになり、改善要求に説得力が増すと考えます。

■ 「実践コース」の実証実験についての講評

成果

- 策定した「実践コース」カリキュラムと現物ベース（実不具合データ）でのWORKSHOPは有効であった。
  - 講義内容として「基礎コース」から抽出したエッセンスのみであっても、ODC分析の理論的理解は深化できる。
  - 自社での不具合の**実データ**を使うことで、WORKSHOPでの分類・分析・評価の議論がより**具体的、実践的**にできる。
  - WORKSHOPを通して、不具合の根元が前工程のテストの不十分さだけでなく、分析結果から**示唆される問題点**から設計レビューの不十分さの**気づきに確信**を持たれ、**改善**に動かれる機運になってきたことは大きな成果であった。

再認識してもらいたいこと：

ODC分析は手段であって目的ではない。

す

👉 **実施すれば自動的に答えが出るものではない。**

実施結果が何を意味するか考察できる力（洞察力）を鍛えるべし

改善ができなければ意味がない

END OF DOCUMENT