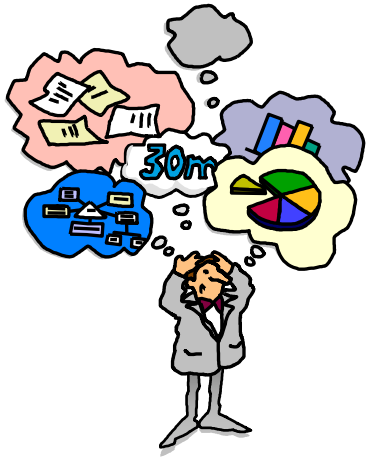
ODC分析研究会　第3期活動報告会資料

ODC分析の基礎

ソフトウェア開発の見える化
- 不具合分析から見えるプロジェクトの健全さ -

(2019年6月13日第３期第一回研究会にて実施)



2020年10月1日

ODC分析研究会　運営委員
杉崎 眞弘

ODC分析の基礎：ソフトウェア開発の見える化
- 不具合分析から見えるプロジェクトの健全さ -

内容：
１．ソフトウェアの出来具合について考える
２．ソフトウェア開発の見える化とは？
３．ODC分析の基礎：不具合分析による開発状況の見える化
４．ODC分析の事例研究
５．ODC分析の適用に際して：プロセス・シグネチャーとODC属性の関係
６．ODC分析の実施ポイント
７．まとめ

■ ODC分析のコンセプト

参照 : Orthogonal Defect Classification – A concept for In-process Measurement

ODC分析のコンセプトは、次の言葉から始まる。

Defect Control is:

A measurement and analysis methodology based on Orthogonal Defect Classification to gain insight and direction for improving software quality.

ソフトウェア開発において、その進捗を妨げる主たる要因は不具合である。
その妨げを低減するには、**「不具合を抑制する (Defect Control)」**ことが重要であると考える。

ここで、**「不具合を抑制する」**とは、
現状このまま推移すると、この先でも起こることが予測される不具合の発生を、事前に然るべき手を打つことで、未然に防ぐ（control: 抑制する）ことである。

そのためには、
- まず、それら不具合を生み出している**「やり方」**を発見する。
- その**「やり方」**を改善するためのアクションを策定し、実施する。

ことが求められ、そのための方法論が必要となる。

すると、左記の言葉は、次のように読める。

「不具合を抑制する (Defect Control)」には、

ソフトウェア品質改善への洞察と示唆を得るための計測方法と分析方法論が必要である。
その元となるのが ODC (Orthogonal Defect Classification)**での不具合の分類法と意味論である。**

- **ODC** (Orthogonal Defect Classification) **の意味**　(原文訳)

“Orthogonal（直交）”という言葉は、数学関係者でない人には聞きなれない言葉かもしれない。幾人かの人たちは、直交する線を頭の中でイメージするかもしれない。
この言葉を使う理由は、
１件の不具合の位置付けを説明することは、直角に交わる壁と床で囲われた部屋の中で、ある物の位置を示すのに壁や床からの位置を使って説明するのに似ているからである。

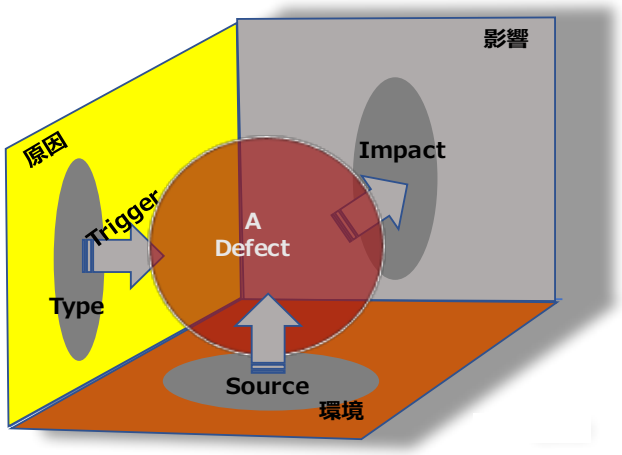


図-3.1-1　１件の不具合とその側面との位置関係

“Defect（不具合）”という言葉は、以前から使われていて、「障害」あるいは「欠陥」のことである。

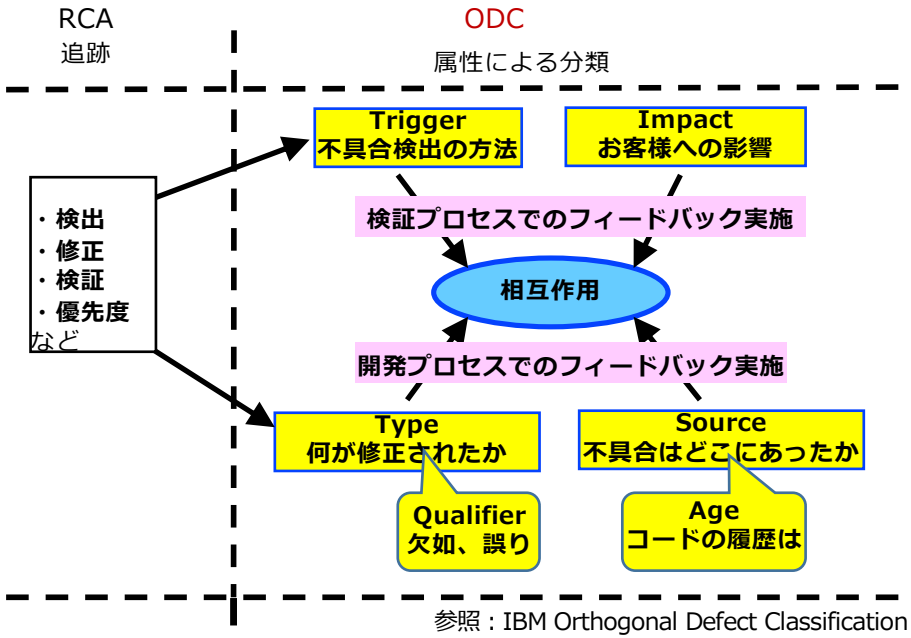
“Classification（分類）”という言葉こそ、この議論の核である。
何かを分類(classify)するとは、種類ごとに振り分けることである。
これまで不具合は研究されてきているが、それらはほぼ市場での信頼性の予測ぐらいのことでしかない。

件の調査・研究の成果として、意味論的に不具合を分類(classify)し、分析する事によって、不具合を抑制(Defect Control)し、適用する**開発プロセスの改善への提言を生み出す**可能性があることを、探り当てることができた。

ODC分析の目的は、開発プロセスの改善にある。

■ ODC分析手法 (ODC不具合属性の説明)

● ODC分析：４つの不具合属性による分類



参照：IBM Orthogonal Defect Classification

なぜなぜ問答を繰り返して、不具合の原因を追求する。

- 多次元の因果関係データをもとに分類・分析する。
- 開発を通じて同一の分析方法でおこなうことで、共通の認識・理解を図る。
- 同一分析方法により開発対象、組織間を通じての整合性を保つことができる。

● ODC分析手法と不具合属性

Classifying by Attributes (不具合属性による分類）を手法とする。

不具合の追跡(tracking)は行わない。追跡には意味論に欠けるからである。
分類(classifying)には設計、テスト、プロセスの問題点を得ることができる。
そのための分類種別を、**属性(attribute)**と呼ぶ。

ODC分析では、以下の**４つの属性**が定義されている。

タイプ (Defect Type)
不具合の意味論理解に影響力のある属性はタイプ(defect type)である。
意味論にもとづいて分類され、プロセスに関する示唆を含んでいる。
この属性の選択肢をバリュー(value)と呼び、**お互いが直交する** 。
この属性の適切なバリューの選択は、通常はその不具合が**開発者**によって解決（修正）された時に明らかになる。

トリガー(Defect Trigger)
不具合が表面化した原因（状況）をトリガー(Defect trigger) と呼ぶ。
その適切なバリューの選択は、**レビューア・テスター**によって行われる。
この属性は**工程固有**となる。

ソース(Source)
不具合が摘出されたコード・プロセスの分類の属性 をソース(source)と呼ぶ。
このバリューの選択は、この不具合を**修正した開発者**による。

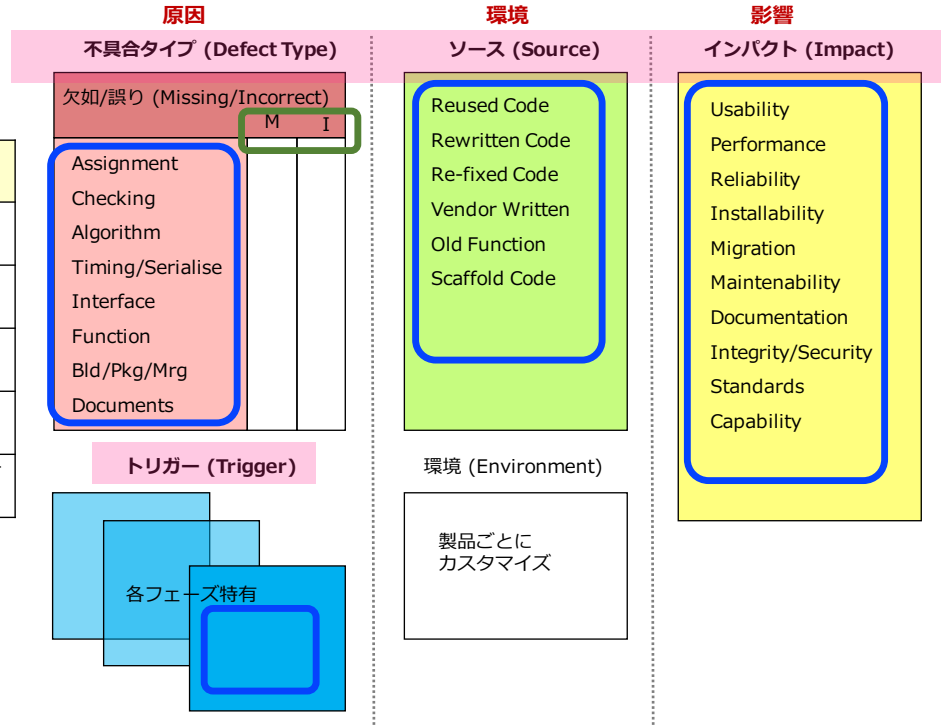
インパクト(Impact)
お客様への影響は、インパクト(impact)属性によって分類される。
品質特性により分類される。 そのバリューの選択は、**レビューア・テスター**がふさわしい。

- **ODC分析：不具合属性の分類について**
  参照：Orthogonal Defect Classification – A concept for In-process Measurement

ODC分析において、不具合属性の分類と分析の観点については、下記の通り。

不具合属性 Attributes	内容	分析の観点
タイプ Type	不具合の種類・性質	本来不具合を検出すべき工程からの漏れの絞込み
トリガー Trigger	不具合が表面化したきっかけ	不具合を検出すべき工程で見直すべき作業内容の絞込み
ソース Source	不具合が含まれていた箇所	品質向上のためにリスクがどの箇所にあるかの絞込み
エイジ Age	不具合が存在していた箇所の履歴	品質向上のためにリスクがどの部分にあるかの絞込み
インパクト Impact	不具合がお客様に与える影響の種類	お客様満足度を向上する要素の絞込み，問題の優先度づけ

- **ODC不具合属性群** (属性と選択肢 Attribute/Value/Dominant)



適用する開発プロセス定義に依存する部分。　参照：IBM Orthogonal Defect Classification

- **ODC分析の適用手順**

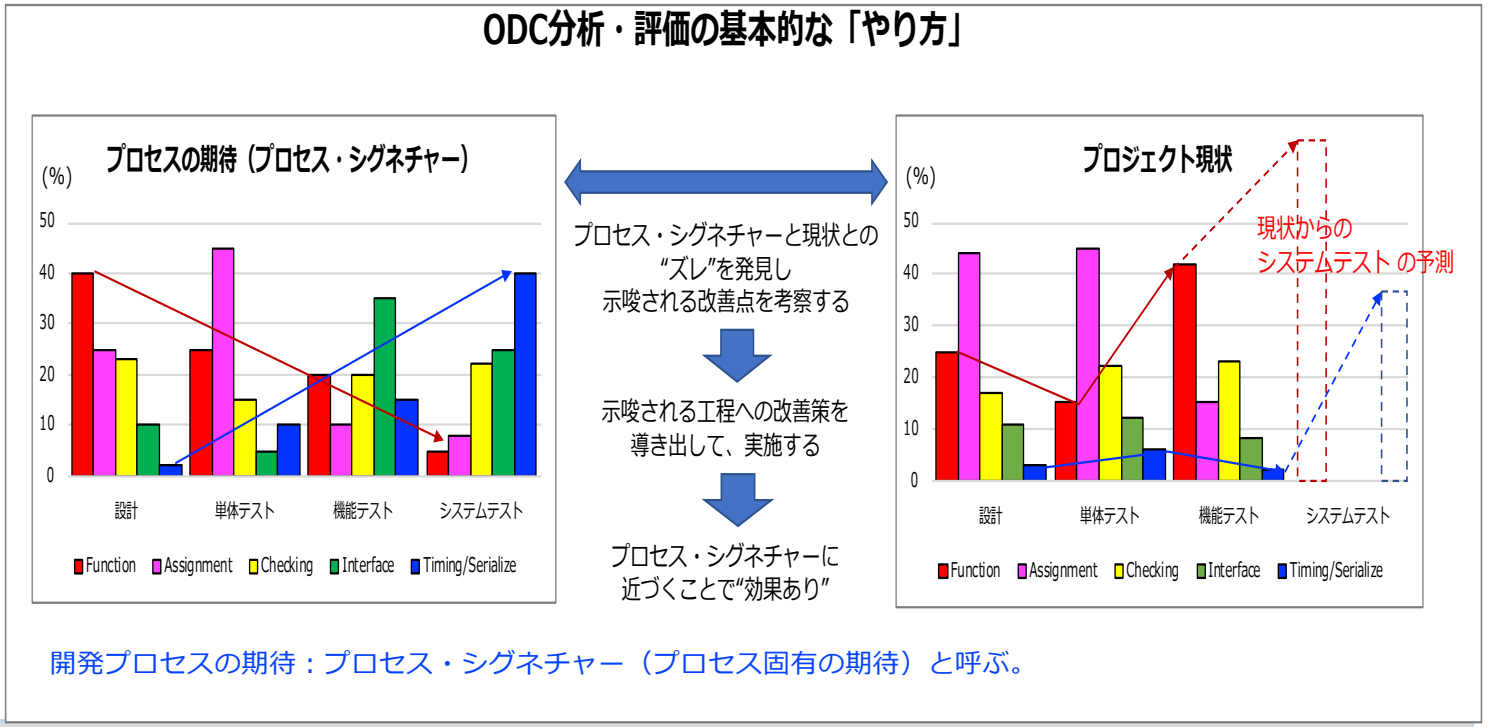
1. 不具合を、不具合属性毎に分類し、その分類結果・分布を集計する。

2. 集計結果とプロセスの期待とを比較・分析する。

3. 発見された”ずれ“の原因から示唆される**必要改善策**を導き出します。

なぜ”ずれ”を問題にするのか？

”ずれ” = 改善のヒント



■ ODC分析の適用に際して（理解しておくべきこと）

● 開発プロセスの特性（シグネチャー）とODC属性分布との関係

開発プロセスの各工程ではそれぞれ目的・主眼、期待が定義されており、その通り実行することで、最終的に全開発工程を通じて、開発対象をあらゆる観点から設計・検証できるように設計されているはずである。

したがって、各工程で検出される不具合も、検証される事項に沿った属性の不具合であるはずである。これをプロセス・シグネチャーと呼び、プロセスの期待である。

例えば、不具合タイプと検出工程の関係は下図のような関係になる。

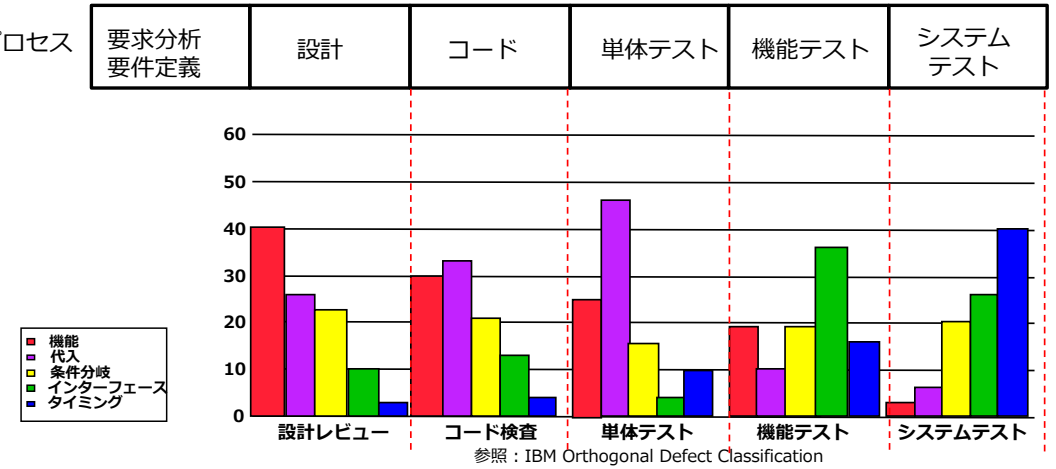
Phase / Defect Type	HLD	LLD	Code	Unit Test	Func Test	Sys Test
Assignment			X	X		
Checking			X	X		
Algorithm			X	X	X	
Timing/Serialize		X				X
Interface		X	X	X	X	X
Function	X				X	

参照：IBM Orthogonal Defect Classification

■ プロセス・シグネチャー

- プロセス・シグネチャーは、開発プロセスに依存した固有の**期待値(こうなるはず)**である。
- プロセス・シグネチャーを定規として、現状とに差異がある場合、「何かおかしい？」とする。
- その「何かおかしい？」を分析することで、そこから示唆される改善策**「何をすべきか」**を見つける。

それが、**ODC分析の志**　である。



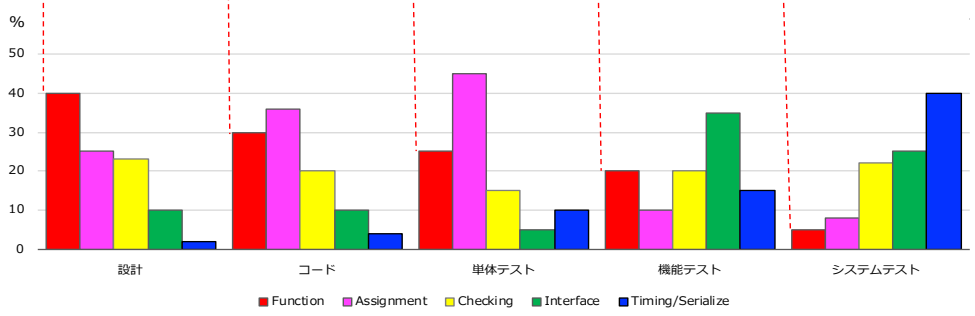
参照：IBM Orthogonal Defect Classification

PART 3: ODC分析の適用に際して（理解しておくべきこと）

- **開発プロセスの特性（シグネチャー）とODC属性分布との関係**

開発工程	要求	設計		コード	テスト		
	要求分析 要件定義	基本設計	詳細設計	コード	単体テスト	統合テスト （機能テスト）	システムテスト
工程目的	要求を分析して 要求仕様書を 作成する	要求仕様を分析し アーキテクチャ分析・ 設計を行い 基本設計書を作成する	基本設計書から 機能単位・構造単位の 詳細設計書を作成する	詳細設計書をもとに コードを作成する	詳細設計書にもとづいて コンポーネント・ 単機能単位でのテストを 行う	基本設計書にもとづいて 求められる機能性の 達成度合いをテストする	要求仕様に求められる 稼動環境・使用範囲に おける 機能要件・非機能要件 をテストする
検証目的	要求に対する 要求仕様書の 実現可能性、妥当性、 を検証する	要求仕様書をもとに 基本設計の妥当性・ 網羅性・ トレーサビリティーを 検証する	基本設計書をもとに 詳細設計書の妥当性・ 網羅性・ トレーサビリティーを 検証する	詳細設計書をもとに、 作成コードが仕様書 どおりかを検証する	詳細設計書をもとに コンポーネント・ 機能単位で仕様どおり 機能するか検証する	基本設計書をもとに 仕様どおりの機能性を 提供しているかを 検証する	要求仕様書をもとに 機能要求、非機能要求 を満たしているか 検証する
予想される (期待される) 不具合	機能性の欠如か誤り、 制約事項の充足性 の不具合	機能分担単位での処理 の動き、振舞いの欠如 あるいは誤りの不具合		設定値や条件分岐の 数字項目、処理順、 APIの呼び出し手順、 制御手順の仕様不具合	設定値や条件分岐の数 字項目、処理手順、 APIの呼び出し手順、 制御手順の実装不具合	システム環境での 機能性、接続性、 処理一貫性、整合性、 妥当性の不具合	機能要求、非機能要求 におけるユーザーの 期待に対する満足度に 関わる不具合
予想される (期待される) 主たる タイプ属性		(Assignment)	Assignment	Assignment			
		Checking	Checking	Checking			
			Algorithm	Algorithm	Algorithm		
		Timing/Serialize				Timing/Serialize	
		Interface	Interface	(interface)	Interface	Interface	
	Function				Function		

期待される
タイプ属性の分布
（プロセス・シグネチャー）

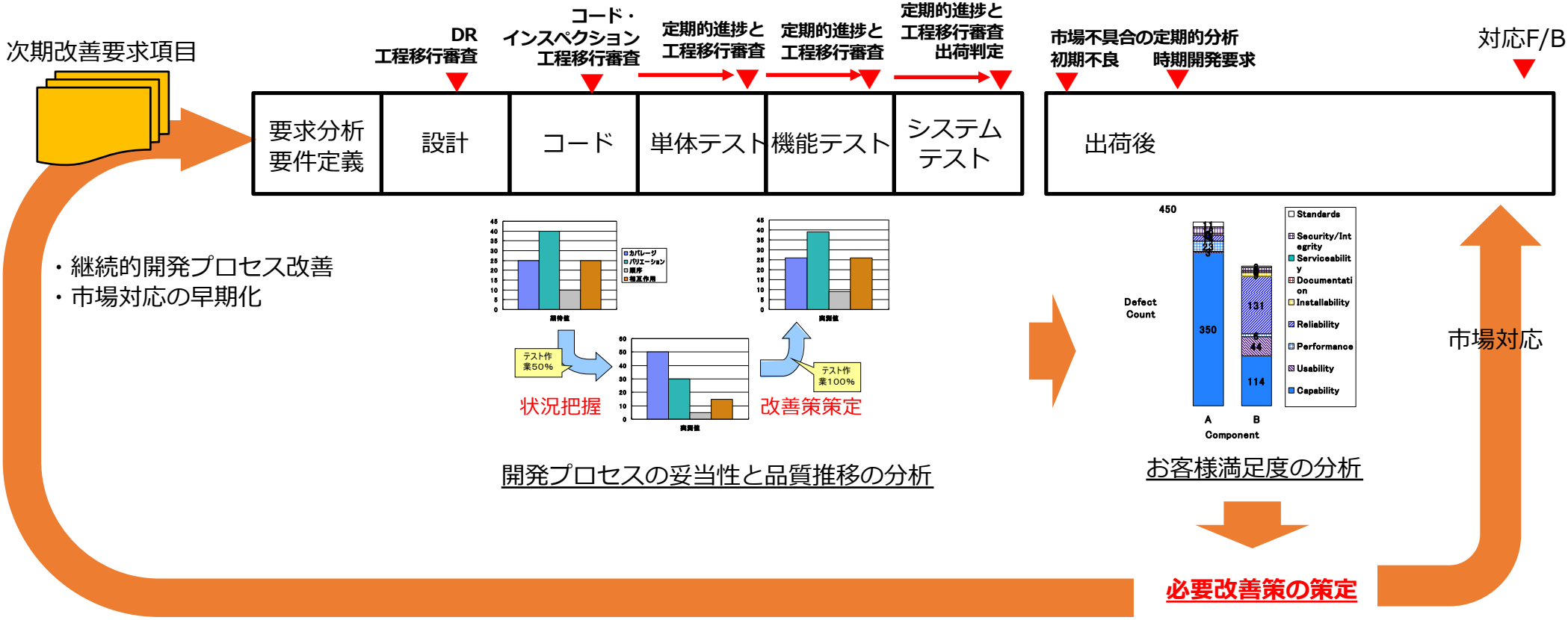


参照：日科技連出版
「ソフトウェア不具合改善手法 ODC分析」

- **ODC分析の効果的な実施タイミング ▼**

ODC分析を定期的、継続的に実施することで、
- 開発プロセス改善の継続
- 市場対応の早期化
- 次期製品の顧客満足度向上

を、図る。

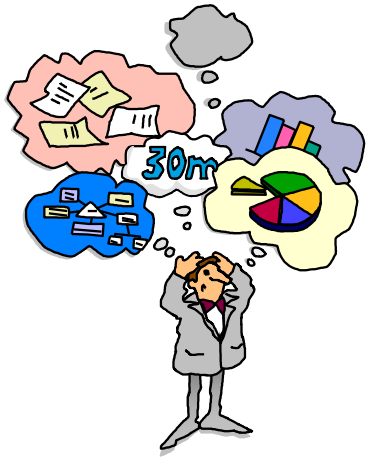


第３期第１回講義アンケートのF/B

ODC分析の基礎

ソフトウェア開発の見える化
- 不具合分析から見えるプロジェクトの健全さ -

2019年7月23日



ODC分析研究会　運営委員
杉崎 眞弘

■ アンケート集計結果

2019年度　第3期ODC分析研究会　特別講義アンケート集計結果

第1回例会「ODC分析の基礎」

2019年6月13日

参加者：34名　　提出者：33名（提出率97%）

4

3

科　目

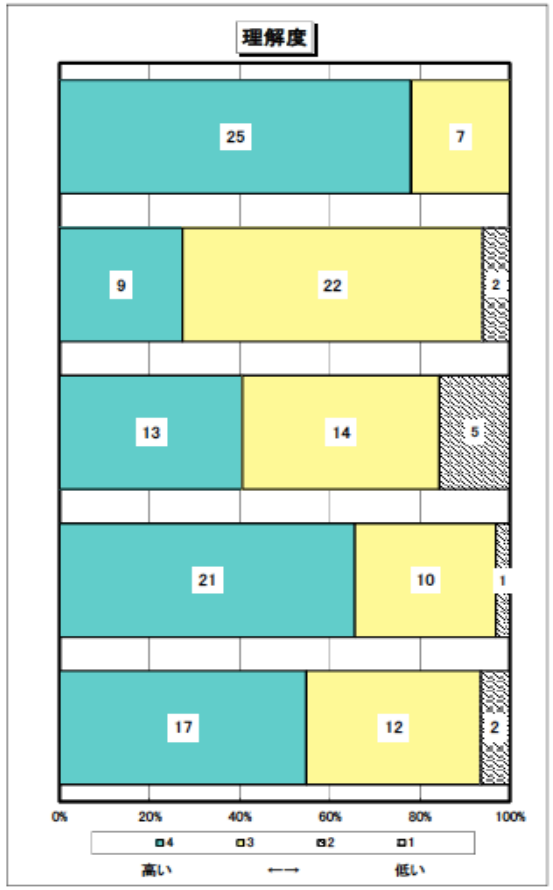
1.ソフトウェアの出来具合について考える
2.ソフトウェア開発の見える化とは？

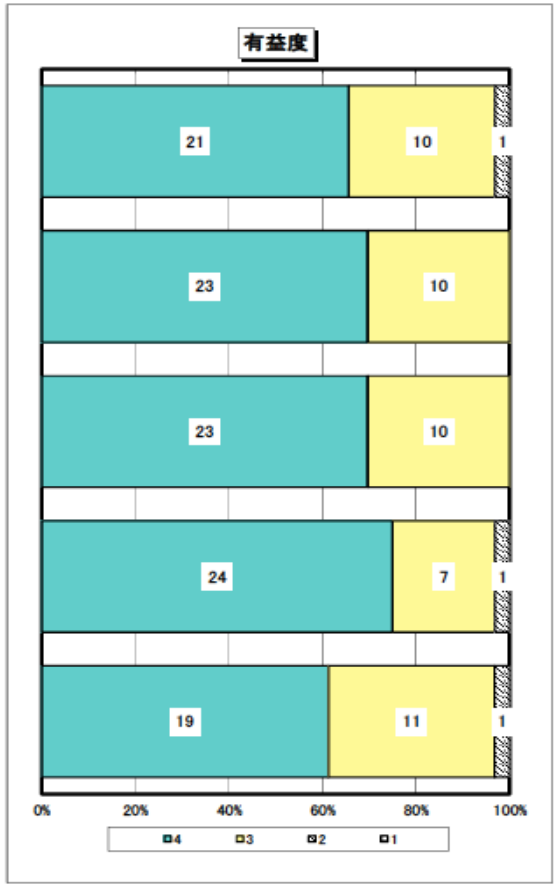
3.ODC分析の基礎

4.ODC分析の事例研究

5.ODC分析の適用に際して

6.ODC分析の実施ポイント
7.まとめ





第３期第１回講義アンケートのF/B

1.ソフトウェアの出来具合について考える
2.ソフトウェア開発の見える化とは？

ソフトウェア工学= ISO/IEC 12207
開発工程の正しい推移＝不具合の出方は減少する

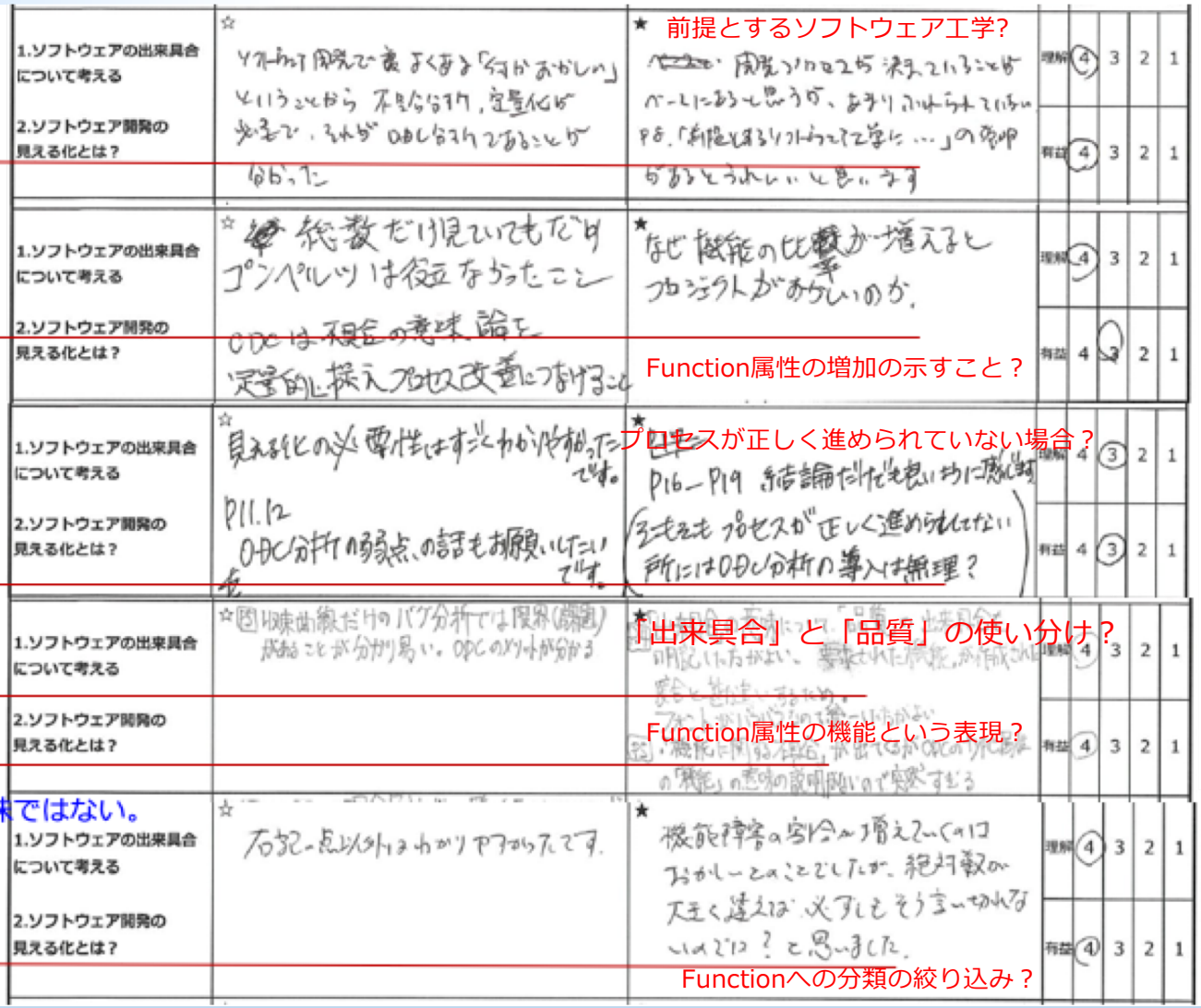
ここでの“機能”とは、ODC属性の”Function“の意
システムテスト工程では前工程で”Function“のタイプの
不具合は、プロセス上、除去されきっている（はず）。

ODC分析は、提要されるプロセスの定義に従って評価
するので、プロセスに従ってに進められていることを
前提とする。そこから、
プロセス定義の不十分さ、実施内容の欠如、未熟さが、
ODC分析で明らかになり、改善につなげる。
とはいえ局面的にも効果的な指摘はできる。

あえて品質でなく「出来具合」と意図は、そこにあり、
要求通りと仕様通り両者を含めて、出来具合とした。

“Function”=機能に関する不具合の機能とは、
機能性に意味で、プロフラムの間違えを一括りにした意味ではない。

同上　”Function“　＜　機能障害で、そのための選択肢
Function”に陥りやすい



3.ODC分析の基礎

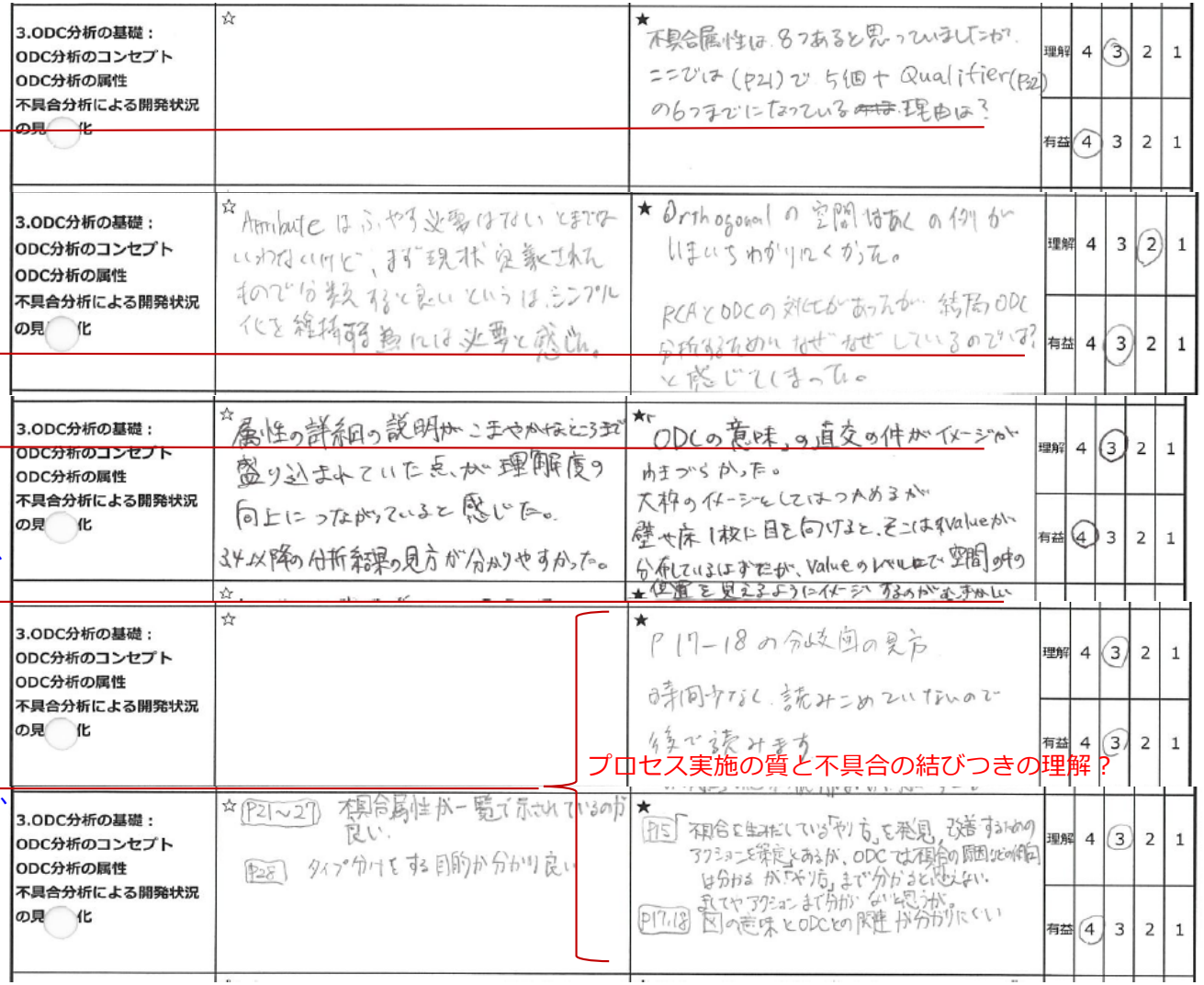
V.5.2?
Attributes：属性（Type, Trigger, Source, Impact）
Value: 各属性の選択肢
Qualifier: 識別子（Missing/Incorrect）

ODCでは、なぜなぜ？問答はしない。
属性定義に従って5~6秒で分類する。

直交とは、相交わらない属性を意味している。

直交する側面からの距離をValueで考えるのは面白いが、
そこまで言及していないと理解する。

プロセスで定義された工程での不具合摘出が、
プロセス完了時にどう結果をもたらすかを示しており、
ODC分析での工程評価の基本的な見方。



3.ODC分析の事例研究

件数より比率の方が、特徴が出やすい

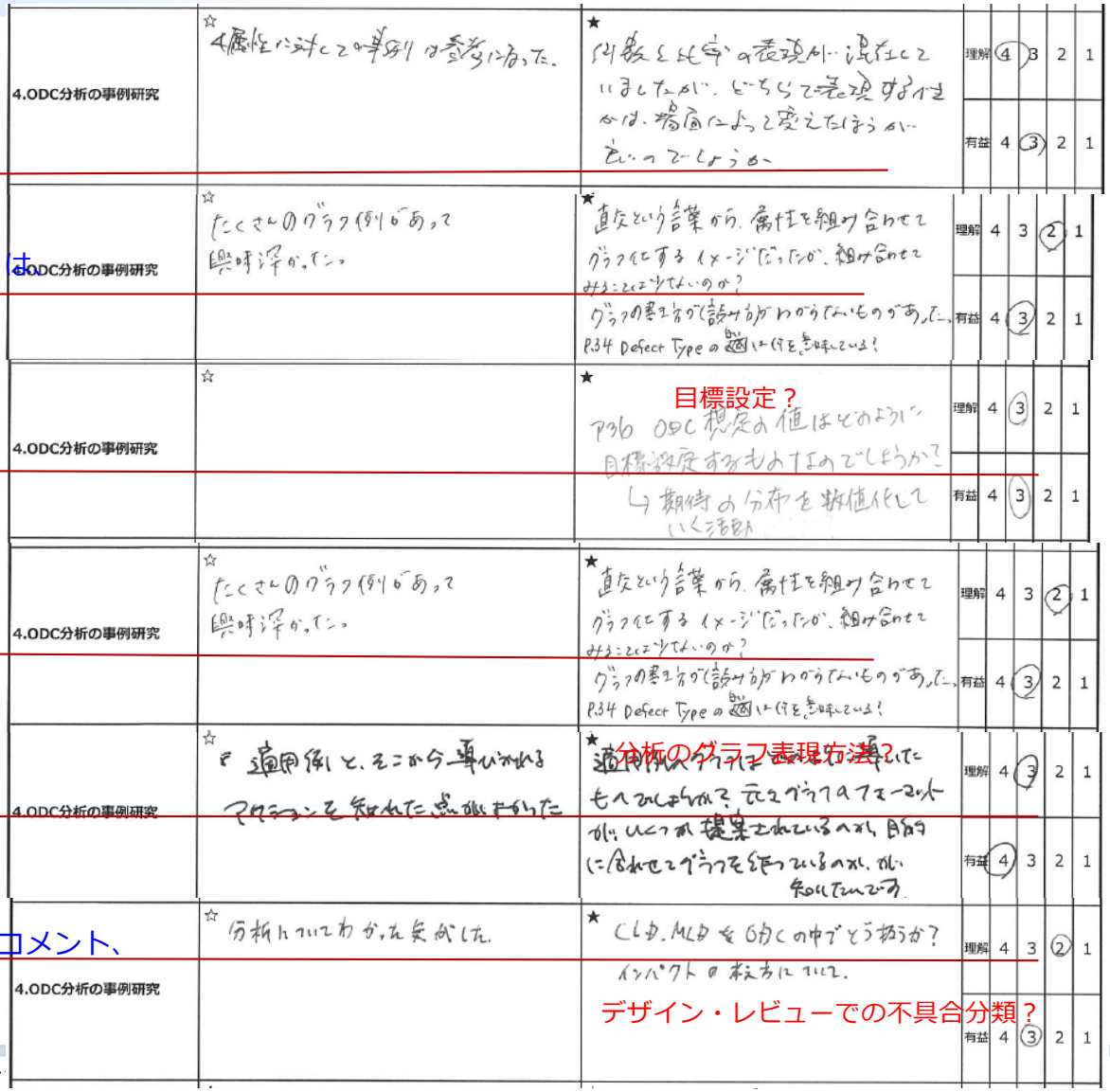
異なる属性での組み合わせ分析（TypeとTriggerなど）は、
特にテストの妥当性検証に使う

プロセス・シグネチャーの値は目標値ではなく、
プロセスとしての分布比率

同上

グラフのフォーマットがあるわけではない
フリーフォーマット

上流から（要求分析、設計レビューなど）レビューコメント、
指摘事項を分類する
非機能要件いついても、同様。

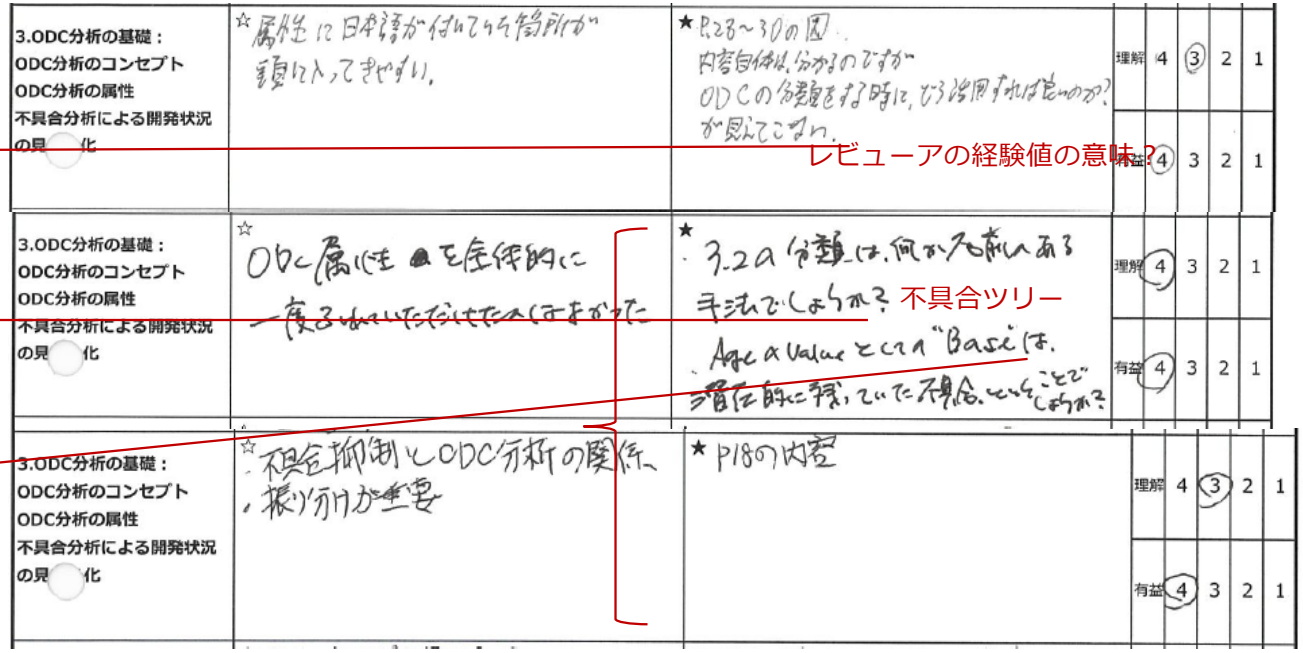


第３期第１回講義アンケートのF/B

上流工程でのレビュー等での指摘事項収集時、
レビュー ア ）質の向上のためのヒント

特に手法でなくく、工程活動の質を
理論的に表したもの

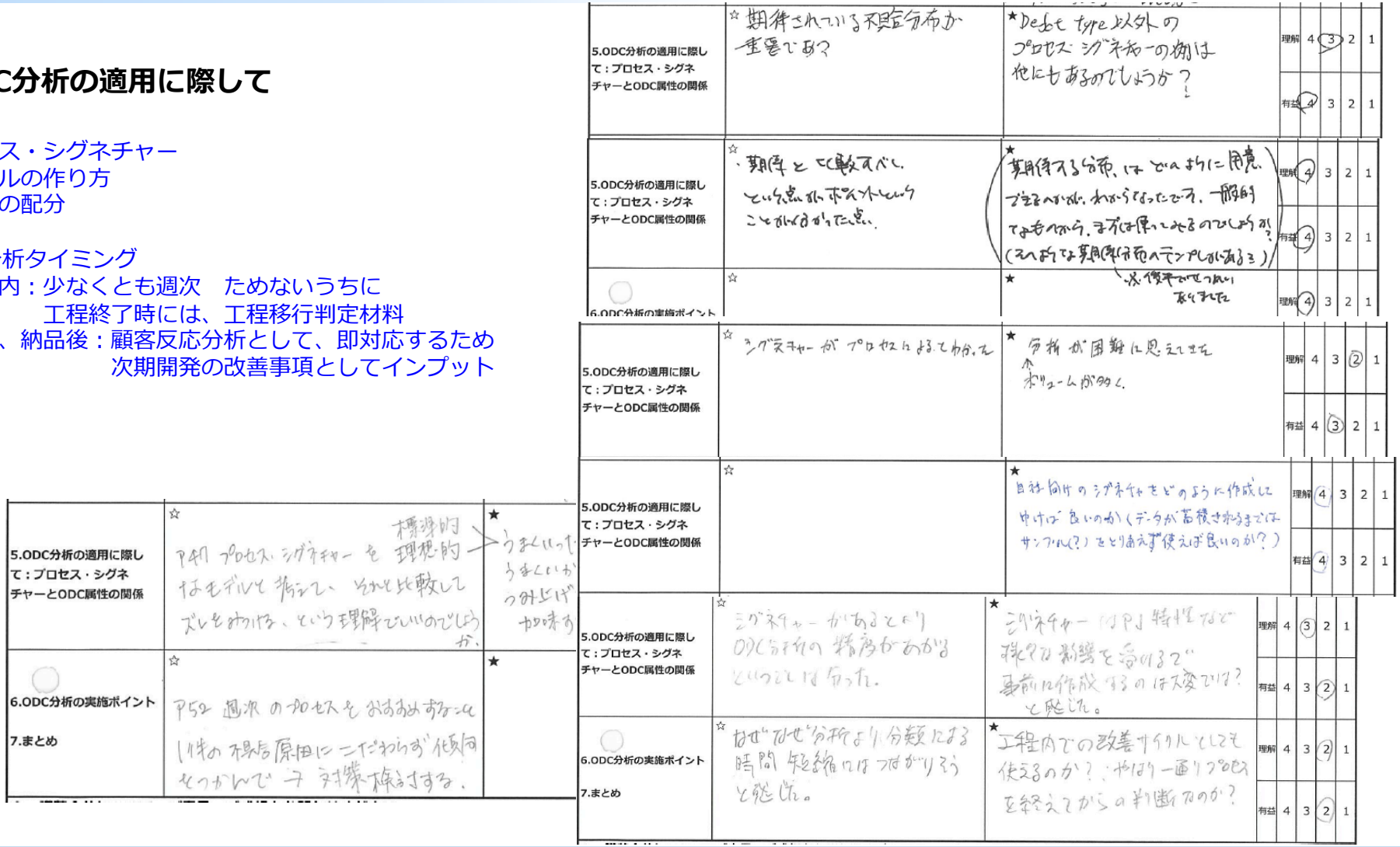
ソース分類でのベース・コードの意味
バージョンアップ、派生開発の際の、ベース・
コードの残存不具合のこと



5.ODC分析の適用に際して

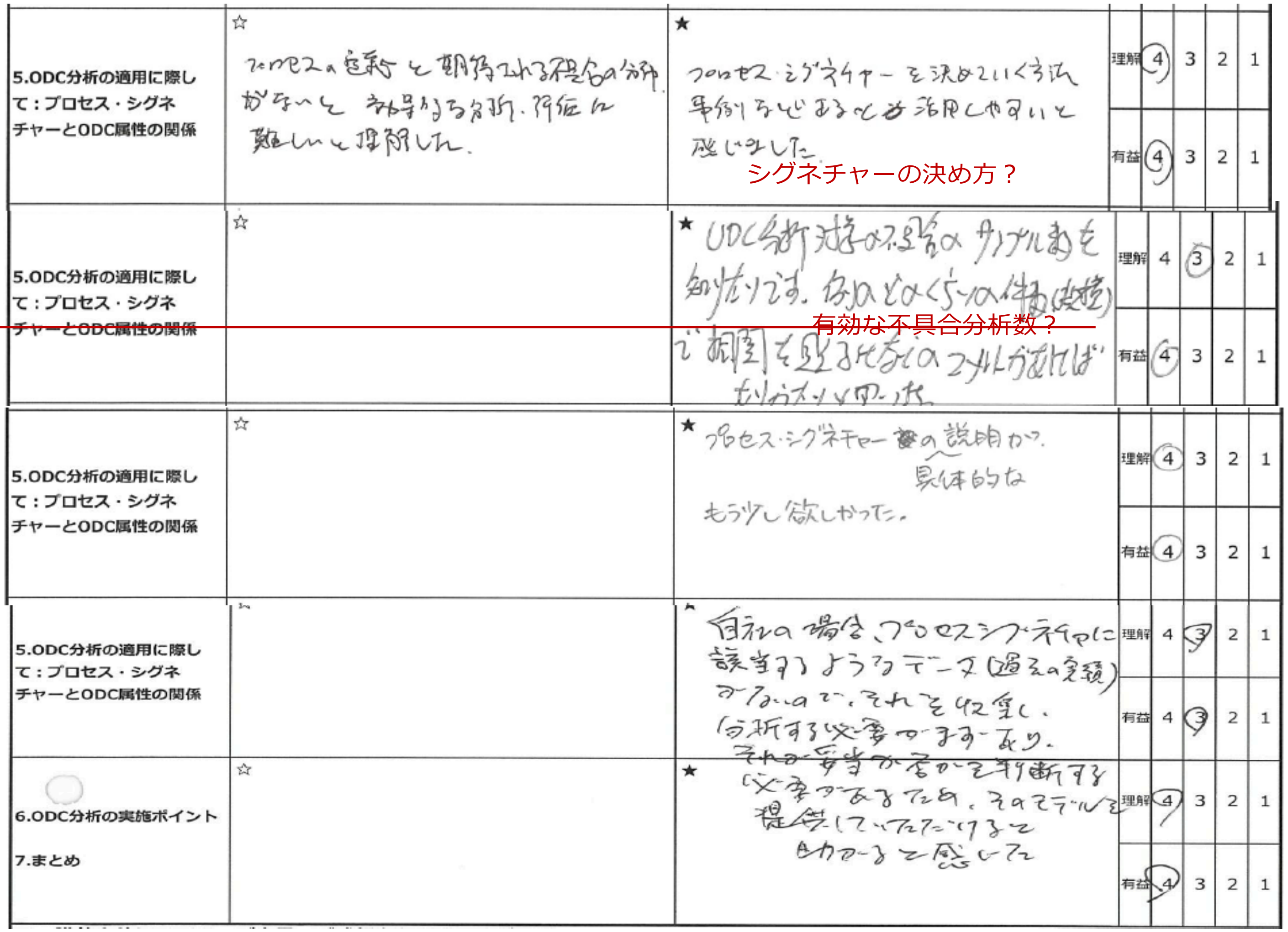
プロセス・シグネチャー
　モデルの作り方
　分布の配分

ODC分析タイミング
　工程内：少なくとも週次　ためないうちに
　　工程終了時には、工程移行判定材料
　出荷、納品後：顧客反応分析として、即対応するため
　　次期開発の改善事項としてインプット



第３期第１回講義アンケートのF/B

対象件数は言及しない。
相対比率で議論するという考え方。



項目	☆	★	評価
5.ODC分析の適用に際して：プロセス・シグネチャーとODC属性の関係	プロセスの定義と期待される結合の分析がないと該当する分析・評価は難しいと理解した。	プロセス・シグネチャーを決めていく方法、事例などあると活用しやすいと感じました。	理解 ④ 3 2 1 / 有益 ④ 3 2 1
5.ODC分析の適用に際して：プロセス・シグネチャーとODC属性の関係		ODC分析対象の不具合のサンプル数を知りたいです。何件ぐらいの件数(実績)で相関を見る[illegible]があればわかりやすいと思った。	理解 4 ③ 2 1 / 有益 ④ 3 2 1
5.ODC分析の適用に際して：プロセス・シグネチャーとODC属性の関係		プロセス・シグネチャーの具体的な説明がもう少し欲しかった。	理解 ④ 3 2 1 / 有益 ④ 3 2 1
5.ODC分析の適用に際して：プロセス・シグネチャーとODC属性の関係		自社の場合、プロセスシグネチャに該当するようなデータ(過去の実績)がないので、それを収集し、分析する必要があります。	理解 4 ③ 2 1 / 有益 4 ③ 2 1
6.ODC分析の実施ポイント 7.まとめ		それが妥当か否かを判断する必要があるため、そのモデルを提供していただけると助かると感じた	理解 ④ 3 2 1 / 有益 ④ 3 2 1

シグネチャーの決め方？

有効な不具合分析数？

6.ODC分析の実施ポイント

自社に適した運用手順のガイドが必要
- 実施のタイミング
- 実施方法
- 運用手順
- 役割分担

開発プロセスにODC分析を組込む
・評価としての認知度・継続性の向上
・役割分担の明確化

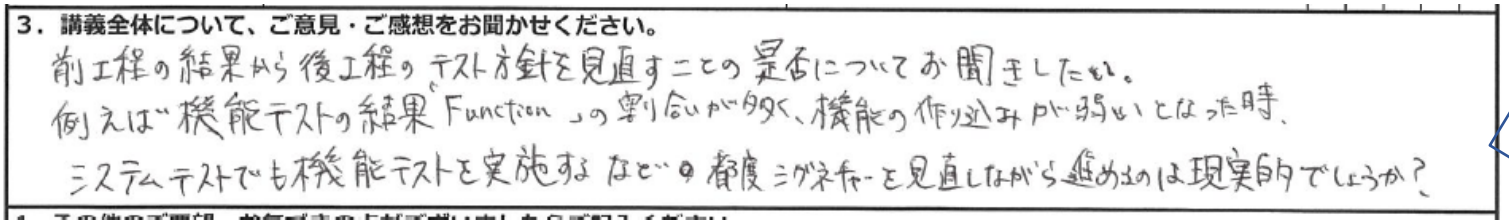
○ 6.ODC分析の実施ポイント 7.まとめ	☆ 成果物のトレーサビリティについては同感であり、理解できた。	★	理解	(4)	3	2	1
			有益	(4)	3	2	1
○ 6.ODC分析の実施ポイント 7.まとめ	☆ ODCはインプロセス、次プロジェクト、両者の改善に役立つP54の共感	★ もう少しくわしく説明して頂きたくと良かったです	理解	4	3	(2)	1
			有益	4	(3)	2	1
○ 6.ODC分析の実施ポイント 7.まとめ	☆	★ 分析するタイミングが前章および初めてはわからなかったが、この章で理解はしたが、毎工程ごとの度に前工程の内容も踏まえながらに行って分析していく必要もあるかも！	理解	4	(3)	2	1
			有益	4	(3)	2	1

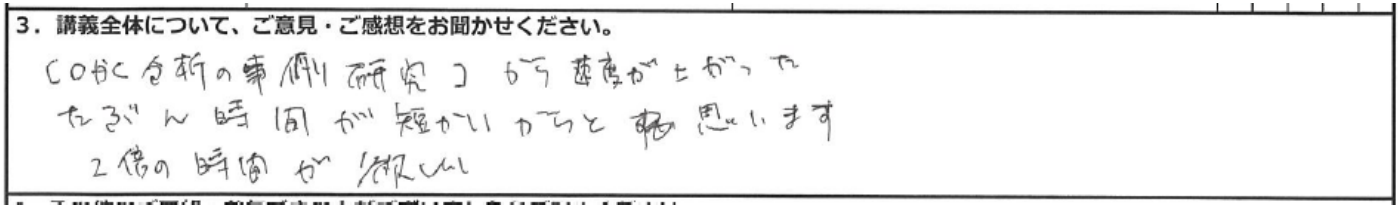
アンケートのまとめ

共通に理解を深めるべきテーマ

- ここでいう開発プロセスについての共通理解　（V字モデルはプロセスではない。プロセス・モデルの一つ）
  開発プロセスという社内での開発標準を明文化したドキュメントを指す。
- ODC属性・選択肢・識別子の定義と区別
- プロセス・シグネチャーの策定方法
- 分析結果の見方、評価の仕方
- ODC分析の運用ガイド　（実施タイミング、日々のデータ収集、分析分担など）

第３期第１回講義アンケートのF/B





工程が終わってからでは後出しジャンケンに見られ、反感も出る。

工程途中で適時ODC分析を実施することで、無い兆候を早期に見つけて、都度注意喚起をして改善していくのが通常のやり方と考える。
しかしながら、
前工程が不十分であった事実に対して、組織としてやり直しを英断するあるいは、対応策を考えることは、組織責任者の度量に必要である。

ただし、よくある次工程（特にシステムテスト）で、前工程の不十分なところを並行して実施する策は、本来の次工程進捗の支障をきたすことが多いので、経験的にうまくいくことは少ない。

ご清聴　ありがとうございました。