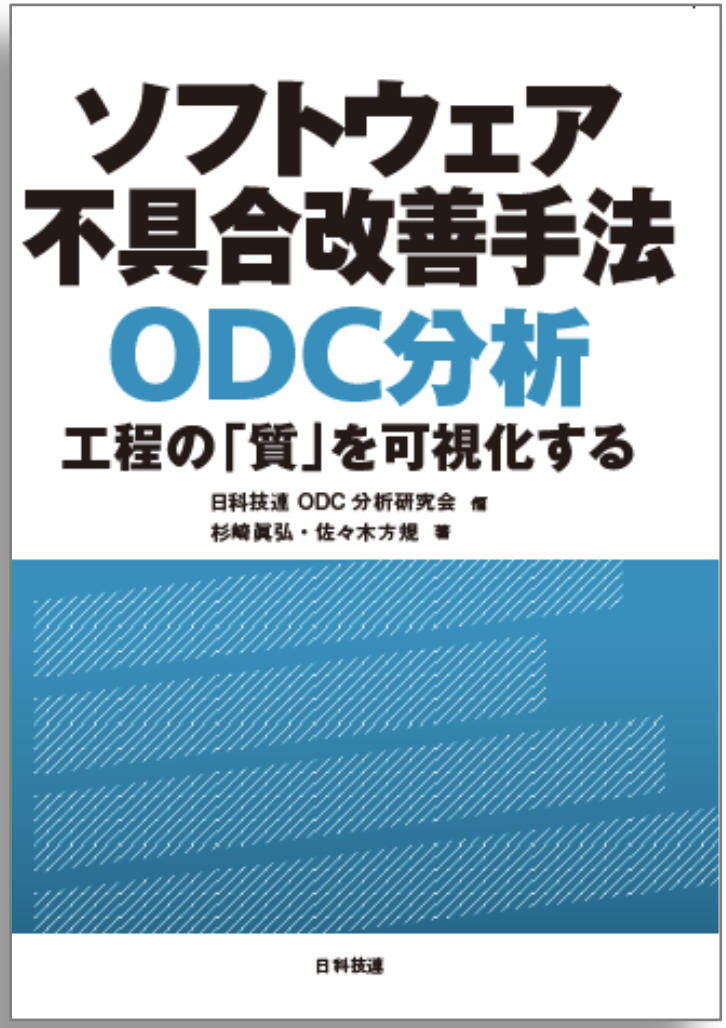
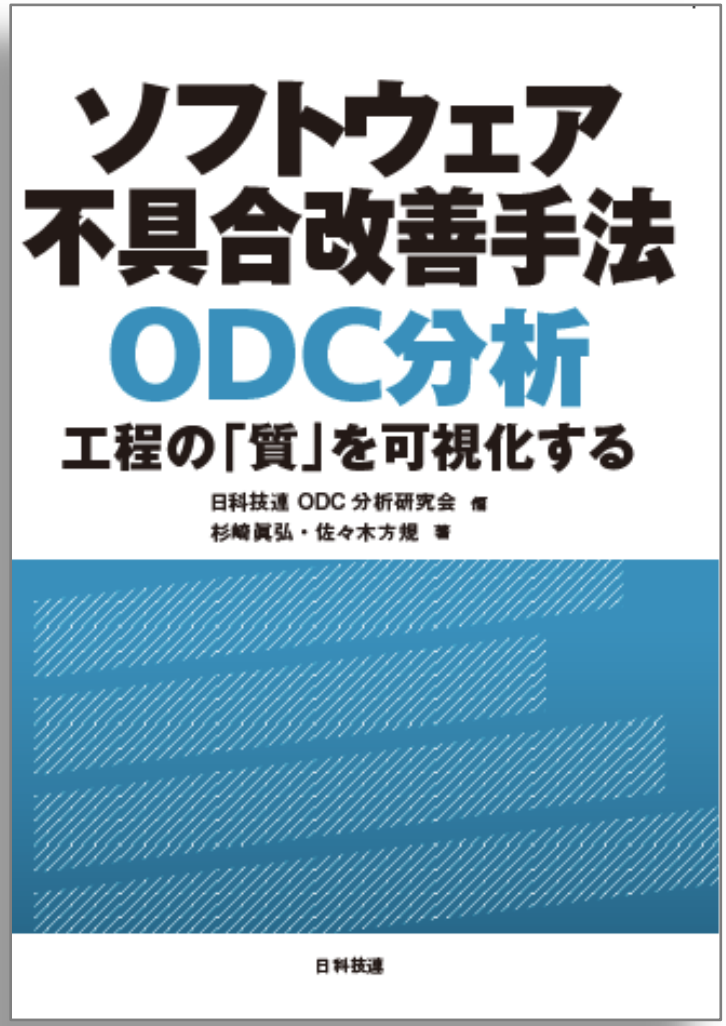


ODC分析の書籍発行について

2020年10月1日

ODC分析研究会　運営委員
杉崎 眞弘



- 書籍タイトル:

  「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 –

  (ISBN978-4-8171-9713-9)

- 日科技連　ODC分析研究会編
  著者：杉崎眞弘　佐々木方規

- 発行日： 2020年8月29日

- 出版元：株式会社 日科技連出版社

- 定価：本体3000円+税

まえがき（本書からの抜粋）

　ソフトウェア開発において、不具合を検出・修正していくことが品質向上に貢献していると一般的に信じられてきた。
確かに不具合を修正して仕様どおりに稼働することを確認するのは、品質検証作業としては必要なことではある。

　では、毎度プロジェクトの終盤で残不具合と格闘せざるを得ないのは、当然のことなのだろうか？
なぜもっと早い段階で手を打つことができなかったのか？
突き詰めて考えると、不具合を残存させてしまう「やり方」をしているのではないだろうか？　という疑問に行き着く。

　そこで考えられたのが、その「やり方」、すなわち、開発プロセスで定義された各工程作業が果たして期待どおりの
実施内容になっているのか？　という工程実施の質（妥当性、的確性、十分性、その結果の信頼性）について、その工程で
摘出された不具合の出方を分析することで、工程実施の「やり方の質」を「見える化」し、必要なアクションが示唆される手法、
それがこのODC（Orthogonal Defect Classification）分析とその手法である。

　このODC 分析は、米国IBM 社の研究チームが社外発表した研究論文をもとに手法化され、社内外で成果を上げて来た手法ではあるが、
米国IBM 社ではこのODC 分析に関する書籍を残さなかった。そのことから、興味があり試してみたいと考える多くの技術者にとって、
ネット上に公開されている断片的な情報のみで試行錯誤をするしかないのが実情ではないかと認識している。

そこで米国IBM 社にてODC 分析の手法化に携わった著者が、国内の技術者向けにオリジナルの考え方に沿って書き下ろしたのが本書で
ある。本書は、ODC 分析理論の理解に加えて、すぐに実務に適用できるよう事例を用いたODC 分析手法の手引書としても使えるよう
に編纂した、初のODC 分析の解説書である。

　著者自身、これまで多くの社内外のプロジェクトにこのODC 分析手法を適用して、その有効性と即効性を体験してきた。本書を通し
て、この優れた手法を一人でも多くの技術者の方々の理解を深め、実務に役立てていただきたいと、切に願っている。

2020年7 月
杉崎 眞弘

本書について

■本書の目的
　本書は、Orthogonal Defect Classification（以下ODC 分析）についての基礎編として、そのコンセプト、分析・評価理論と手法、実施事例について解説したもので、初学者の方々にも、一応にODC 分析についての基礎的な実践知識が得られることを目的とする。

■本書の想定する読者
　ソフトウェア開発に携わる開発技術者、品質評価者、プロジェクト管理者の方々、またプロセス改善に関わる方々とする。

■本書の原典
　巻末の参考文献［1］にある米国IBM 社においてODC 分析着想の論文をもとに手法化にいたる議論の成果としてのプロセスの考え方、ノウハウを加筆して、ODC 分析のオリジナルを理解してもらえるように努めた。

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

目　次

本書の概要

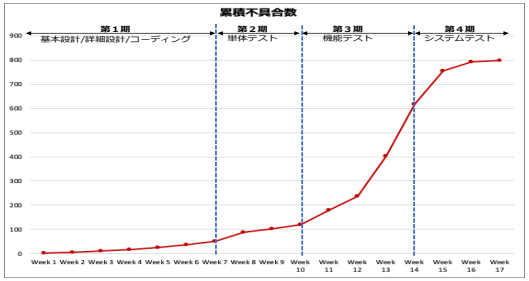


図1.1　【事例研究 1】発見された不具合の累積数

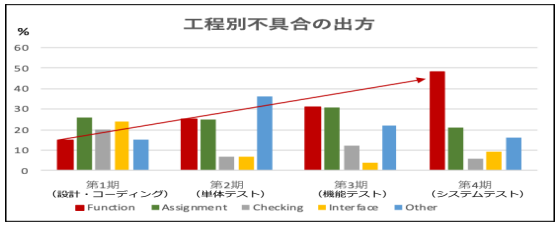


図1.2　【事例研究 1】工程別不具合の出方

1.2　ソフトウェア開発の見える化とは？

1)「何かおかしい」を気づかせる。

2) その「何か」が何であるかを示す。

3) それに対して「何を」すればよいかを示唆する。

ということを求めたい。

1.3.2 ODC分析とは

- ソフトウェア開発の見える化のための不具合分析理論と手法である。
- ODC分析は、プロセス実施の「質」を改善することをめざす。
- ODC分析は、適用する開発プロセスの特性から不具合の出方を分析する手法である。
- ODC分析では、プロセスの理論的不具合分布と実際の不具合の分布を比較することで、「何かおかしい」ことを気づかせる。
- そしてODC分析の結果が示唆する**改善策**は、個々の不具合に対してではなく、それを生み出したプロセス実施の「やり方」を改善するための改善策である。

ODC分析のコンセプト

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

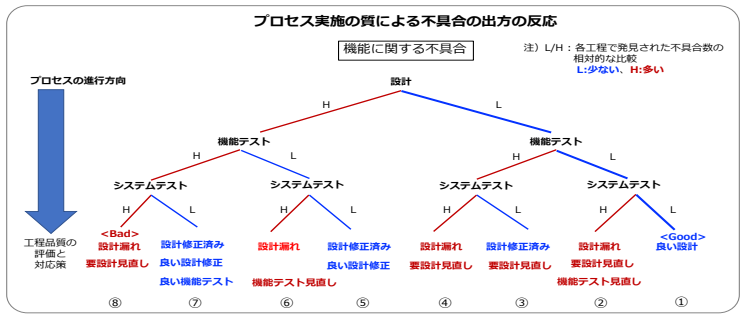
「不具合を抑制する」とはどういうことか。

「プロジェクトにおいて、現状のままの「やり方」で推移すると、
この先または次工程以降でも同じことが起こると予測・予見される不具合に対して、
今、不具合を起こしている「やり方」に適切なアクションを取ることで、
未然に防いでいく(抑制する:controlする)ことである。」

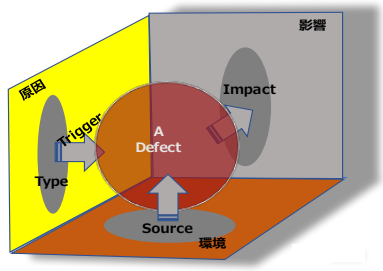
「不具合を抑制する (Defect Control)」には、
ソフトウェア品質改善への洞察と示唆を得るための計測方法と
分析方法論が必要である。
その元となるのが ODC (Orthogonal Defect Classification)での
不具合の分類法と意味論である。

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属性の観点

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

4.2.2 プロセス・シグネチャー

「プロセス・シグネチャー」とは、「開発プロセスの期待」である。

「プロセスの期待」の定量化に対する1つのアプローチとして、

ODC分析では、以下のような考えに立っている。

正しくプロセスを実施すれば、正しく「しかるべき不具合」の分布になる。

つまり、開発プロセスが期待する工程作業を正しく実施すれば、

その検証では、「しかるべき不具合」が「しかるべきとき」に出るはずである、

すなわち、期待どおりの不具合の出方をするはずである。

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		

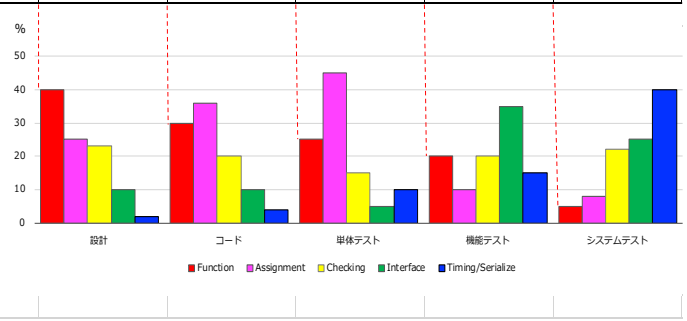
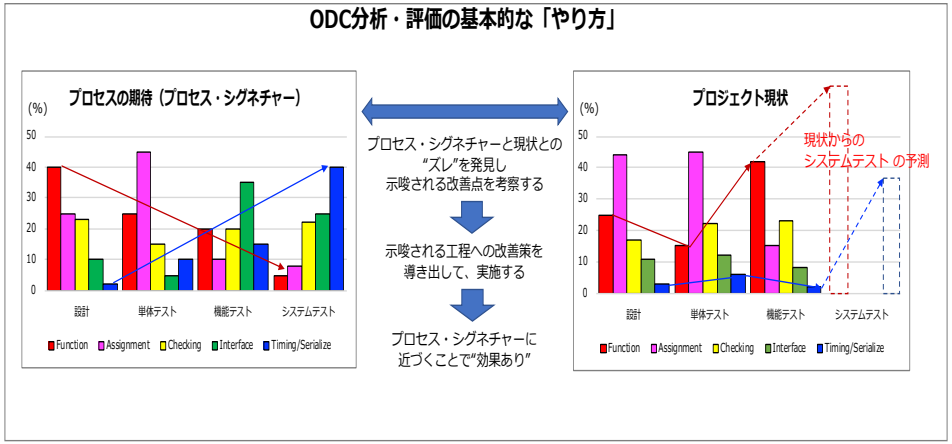


図4.1　開発プロセスと「しかるべき不具合」の傾向と分布



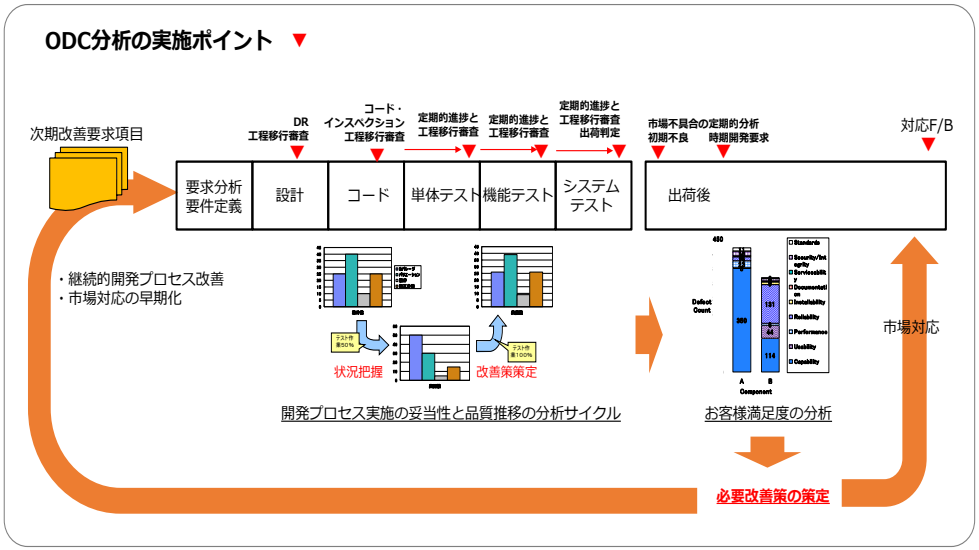


図5.2　ODC分析の実施ポイント

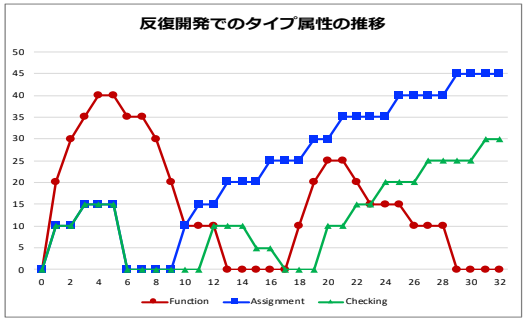


図5.8　反復開発でのタイプ属性の推移

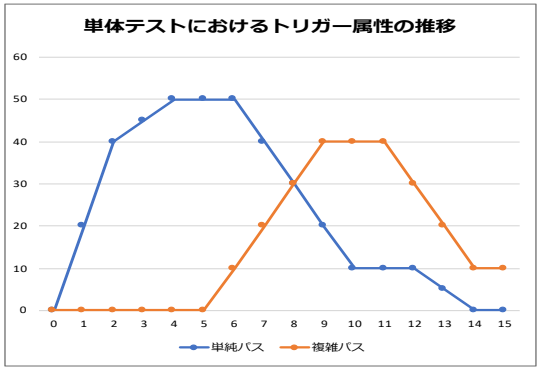


図5.14　単体テストにおけるトリガー属性の推移 (単純パスと複雑パス)

開発プロセスの心

大前提：「開発プロセスのとおりに正しく開発すれば、正しい物が作れる」

現実：しかしながら、それを実践するのは意思のない機械でなく、個々の意思を持った人間の集まりである。そのため、
実践の「やり方」の質に個人差が生じてしまう。そのことが開発プロセスの期待との乖離が生まれる根源であり、そこに不具合を生み出す原因がある。

「やることになっているので、やっている」というのではなく、
「こういう理由で、こういうことをやらないといけない」と開発プロセスの期待を論理的に理解することが重要 = 開発プロセスの心を理解する。

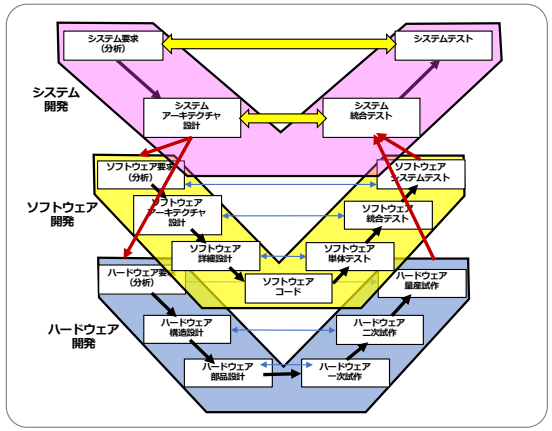


図6.8 システム開発
V字プロセスモデル(多重化)[4]

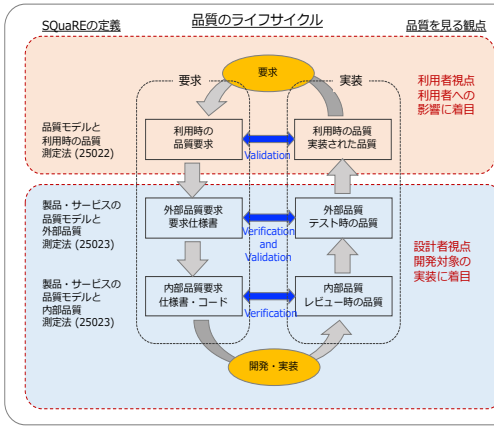


図6.22 SQuaRE
品質のライフサイクル・モデルと品質観点

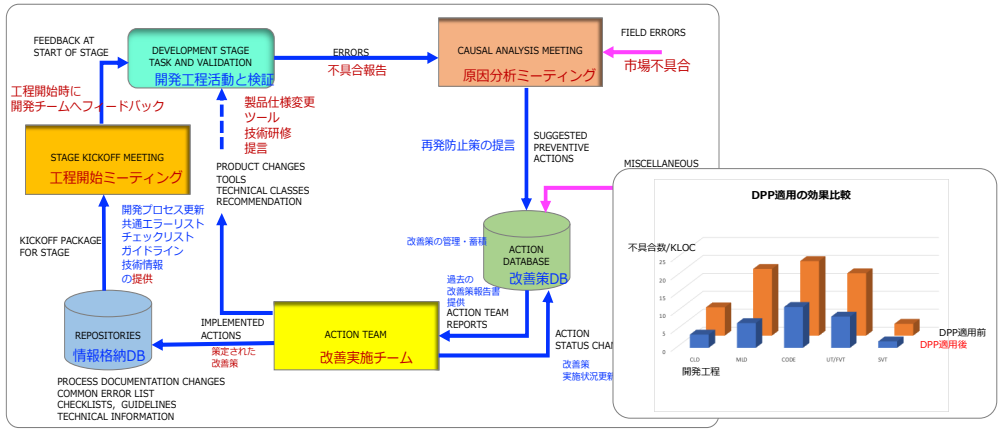


図6.14 Experiences with Defect Prevention (DPP)

あとがき（抜粋）

　ODC 分析を広くソフトウェア開発に携わる方々に知っていただき、実務に役立てていただきたいという趣旨で本書を執筆した。
本書では、ODC 分析のコンセプトと事例を通しての分析・評価の理論、その背景にある開発プロセスについての考え方を解説している。

　さて、ここまで読み進めていただいた読者の方々の多くもお気づきになられたと思うが、筆者自身も本書を書き進めながら改めて感じたのは、ソフトウェア開発における開発プロセスの役割の重要性である。これは、ソフトウェア開発業務の遂行においては、「開発プロセスをいかに自身の中で正しく理解し、咀嚼して行動に移せるようになっているか」が業務遂行の「質」として常に問われていることを意味している。
　そうした開発プロセスの重要性をより伝えたいという思いが、本書の第6 章が予想以上にボリュームが増えてしまった理由である。一重に少しでも筆者の実体験から言うところの開発プロセスの「心」を理解することが、開発プロセス実施の「やり方の質」を向上させることにつながるということを、読者の方々と共有したいという思いからであることを、ご理解いただきたい。
そうした筆者の思いは、ODC 分析の考え方の理解を深めることで、開発プロセスの「心」を相対的に理解することに通じるとも考えている。
　また、本書は、筆者が習得したオリジナルのODC 分析の分析手法、評価理論をもとに、これまでの適用経験を加味して執筆しているが、筆者自身まだまだODC 分析には研究の余地があると考えている。言葉を変えると、開発形態、開発技術の変化に対応して、成長する分析手法ともいえる。
　そこで、読者の方々には、まずは本書にあるオリジナルの考え方に沿って実施してみてその効能を見きわめることをお勧めする。そしてある程度の習熟を積んだうえで、定型的な分析にこだわらず、あらゆる角度からの見方をしてみると、また新たな発見があるかもしれないと考えている。分析手法に応用を利かせるのは、先走らず、そうした習熟を積んでからがしかるべき時期であると、これまでの経験から実感している。
まずは、実務で本書を活用されることを期待している。

2020年7 月
杉崎 眞弘