DXITフォーラム　品質WG

DX時代に求められる品質

DXITフォーラム　DX時代の品質課題を考えるワーキンググループ(品質WG)

https://www.ipa.go.jp/digital/dx/dxit-forum.html

2024年4月

内容

はじめに

DXITフォーラム 品質WGでは「DX時代のソフトウェア品質がどうあるべきか」をテーマに、ソフトウェア品質の調査やその結果を検討するために2023年5月から活動しており、この活動の中で紹介・議論したものをここにまとめた。

この活動で議論されるキーワードには、アジャイル/アジリティ、価値、ユーザなどがあり、これらのキーワードとDXやソフトウェア品質との関連を探っている。

本書はソフトウェア品質の入門編から事例までを含む読本になっており、読者の役に立つことを望んでいる。

DX時代の品質課題

1. ソフトウェア品質
（アジャイル）アジャイル開発での品質
（パッケージ）ノーコードなどのパッケージ利用での品質
（ユーザ内製化）ユーザ主導の開発での品質
（非機能品質）SQuaREの品質特性の観点での品質
（ウォーターフォール）ウォーターフォール開発の品質

2. 価値に関する品質
（UX）ユーザ体験の観点での品質
（サービス）サービス観点での品質
（顧客価値）顧客価値の観点での品質
（利用品質）SQuaREの利用時の品質

3. 品質計測と制御
（定義）品質の定義や範囲
（計測）品質計測/分析/管理、メトリクス(計測基準)
（制御）品質施策/制御
（コスト）品質に掛かるコストと効果

4. 品質に関わる組織・人
（ベンダ）SIベンダ、開発ベンダの品質面での役割
（ユーザ）ユーザの品質面での役割、２次ユーザの行動

5. 対象（過程と結果）
（プロセス）開発プロセスや運用プロセス
（プロダクト）人を含むエコシステム全体

品質とは

- 辞書的意味「**品物の性質**」
  性質とは先天的に持っている特徴

- 一般的定義「**その品物の性質が要求を満たす程度**」
  ISO9000,PMBOK,TQC

- ソフトウェア品質の変遷

1970年代	1980年代	1990年代	2000年代	2010年代
欠陥がない Capers Jones	要求を達成 Watts S. Humphrey	品質とは価値 Gerald M. Weinberg	ニーズを満たす能力 JIS X 0129-1 6個の品質特性	X 25010 改訂

SQuaREとは

SQuaRE Systems and software quality requirements and evaluation
JIS 25010 : 2013 (ISO/IEC 25010:2011)

製品品質　　：ソフトウェア開発時における一般的な品質
利用時の品質　：ユーザ視点での品質、顧客満足度の一視点
データ品質　　：製品で使用するデータに関する品質

参考文献
つながる世界のソフトウェア品質ガイド
https://www.ipa.go.jp/publish/qv6pgp0000000wkj-att/000055008.pdf

pp.129- 利用時の品質
pp.149- 製品品質
pp.197- データ品質

・品質特性の「名称」をそのまま使う
・品質特性を（代替手段も含めて）
　定量化する方法を紹介

SQuaRE　製品品質の特性

主特性	副特性					
機能適合性	機能完全性	機能正確性	機能適切性			
性能効率性	時間的効率性	資源効率性	容量満足性			
互換性	共存性	相互運用性				
使用性	適切度認識性	習得性	運用操作性	ユーザエラー防止性	UI快美性	アクセシビリティ
信頼性	成熟性	可用性	障害許容性	回復性		
セキュリティ	機密性	完全性	否認防止性	責任追跡性	真正性	
保守性	モジュール性	再利用性	解析性	修正性	試験性	
移植性	適応性	設置性	置換性			

（参考）セキュリティのCIA
機密性(C)、完全性（I）、可用性（A)

SQuaRE 利用時の品質特性

主特性	副特性			
有効性	有効性			
効率性	効率性			
満足性	実用性	信用性	快感性	快適性
リスク回避性	経済リスク緩和性	健康・安全リスク緩和性	環境リスク緩和性	
利用状況網羅性	利用状況完全性	柔軟性		

SQuaRE データ品質の特性

(主)特性

正確性	アクセシビリティ	追跡可能性
完全性	標準適合性	理解性
一貫性	機密性	可用性
信憑性	効率性	移植性
最新性	精度	回復性

SQuBOK ソフトウェア品質知識体系

ソフトウェア品質知識体系ガイド https://www.juse.or.jp/sqip/squbok/
（概要） https://www.slideshare.net/hironoriwashizaki/squbokv3-iotaidx

ソフトウェア品質知識体系ガイド （第3版） －SQuBOK Guide V3－
・ソフトウェア品質の基本概念
・ソフトウェア品質マネジメント
・ソフトウェア品質技術
（メトリクス、モデル化、形式手法、要求分析、設計、実装、レビュー、テストなど）
・専門的なソフトウェア品質の概念と技術
（ユーザビリティ、セーフティ、セキュリティ、プライバシー）
・ソフトウェアの応用領域
（AI、IoT、アジャイルとDevOps、クラウド、OSS）

AIやアジャイルにも踏み込んでいる！
第５章　ソフトウェア品質の応用範囲(AI、IoT、アジャイル/DevOps、クラウド、OSS)
AI：訓練データ/モデル/システム、疑似オラクル/メタモルフィック/頑健性/ニューロンカバレッジ/説明生成
アジャイル：アジャイルメトリクス、QA2AQ(アジャイル品質保証パターン)
セーフティ：STAMP/STPA

日科技連(JUSE)/SQiP(Software Quality Profession)

PMBOK 品質マネジメント

PMBOK プロジェクトマネジメント知識体系

品質マネジメント：プロジェクトマネジメントの一つ

・品質マネジメントの計画（品質目標の立案など）
・品質マネジメントは第三者(PMOや品質保証部)
・品質目標を満たす（逆に低品質の原因の特定も）
・品質マネジメントのアウトプット
　品質報告書、テスト・評価文書、変更要求、PM計画書更新、
　プロジェクト文書更新、教訓登録簿、リスク登録簿

PMBOKは品質管理の基礎として有用、教訓/リスク管理簿

(参考) https://ssaits.jp/promapedia/concepts/manage-quality.html

品質機能展開 QFD（Quality Function Deployment）

顧客の要求品質を製品企画や開発仕様に変換する手法

- JIS Q9025：2003
- 要求品質と品質特性をマトリックスにした品質表で点数化
- 品質表：縦軸に要求品質、横軸に品質特性
  これから右段に企画品質、下段に開発仕様に変換
- 技術展開、コスト展開、信頼性展開、業務機能展開

品質表の例 https://qctoranomaki.com/qms/qa/qfd/

品質表（ワイヤレス型マウス）　5段階評価を入力　◎、○、△を選択　自動計算

要求品質		品質特性														企画品質								
		操作性			携帯性			デザイン性			メンテナンス性		耐久性			要求品質重要度	自社	A社	B社	企画品質	レベルアップ率	セールスポイント	絶対ウェイト	要求品質ウェイト
一次	二次	センサー方式	分解能	本体裏面材質	質量	本体寸法	使用可能距離	色	本体形状	本体表面材質	使用可能時間	耐摩耗性	耐熱	耐湿	耐衝撃									
持ち運びやすい	質量が軽い				◎	○										5	4	4	2	5	1.25	◎	9.4	13.6
	サイズがコンパクト					◎										5	5	4	2	5	1.00	◎	7.5	10.8
手入れの手間が少ない	傷が付きにくい			○						○		◎			△	3	4	3	4	4	1.00		3.0	4.3
	電池交換の頻度が少ない										◎					3	3	3	4	3	1.00		3.0	4.3
見た目がよい	色がおしゃれ							◎								3	4	5	2	3	0.75		2.3	3.3
	デザインがスタイリッシュ								◎	○						3	3	5	2	3	1.00		3.0	4.3
使いやすい	マウスの滑りが良い	◎	◎	○		△										5	4	3	4	3	0.75	○	4.5	6.5
	ボタンの操作が軽い					△			◎							4	3	3	4	4	1.33		5.3	7.7
	PCから離れた場所でも使える	○	○				◎									4	4	2	3	3	0.75		3.0	4.3
心地よく使える	手の形状にフィットしやすい					◎			○							4	3	2	5	4	1.33		5.3	7.7
	クリック感が良い								○	◎						3	3	3	4	3	1.00		3.0	4.3
	クリック音が静か								○	◎						5	4	2	3	5	1.25	○	7.5	10.8
壊れにくい	熱に強い	○		○						○			◎			4	3	2	4	4	1.33		5.3	7.7
	湿気に強い	○		○						○				◎		3	3	2	4	3	1.00		3.0	4.3
	落下に強い	○		○	◎					○					◎	4	4	3	3	4	1.00		4.0	5.8
品質特性重要度		70	37	57	45	69	20	15	71	91	15	15	20	15	23								69.1	100
品質特性ウェイト		99.1	45.6	86.1	96.7	147.7	21.7	16.3	129.0	155.5	21.7	21.7	38.6	21.7	33.3									
開発仕様（設計品質）		Blue LED	●●dpi以上	●●材または▲▲材	●●g以下	●●×▲▲mm以下	◆◆m以上	●●または▲▲	非対称形状	●●材または▲▲材	■■hr以上	▲▲規格相当	動作環境●●～▲▲℃	動作環境●●～▲▲%	●●規格相当									

相関度		セールスポイント	
◎：関連が強い	(5)	◎：大	(1.5)
○：関連あり	(3)	○：中	(1.2)
△：可能性あり	(1)	－：小	(1)

Society5.0と品質

Society5.0

サイバー空間（仮想空間）とフィジカル空間（現実空間）を高度に融合させたシステムにより、経済発展と社会的課題の解決を両立する、人間中心の社会（Society）
狩猟社会、農耕社会、工業社会、情報社会の次の社会
（内閣府「Society 5.0」https://www8.cao.go.jp/cstp/society5_0/）

キーワード

(1) CPS/ビッグデータ/AI（仕組み）
(2) IoT：人とモノがつながる
(3) AI：人の知的作業を支援
(4) ロボット、自動走行：人の可能性がひろがる
(5) イノベーション：様々なニーズに対応
(6) 社会的課題（カーボンニュートラル、食料ロス、社会コスト、SDGs、地域格差）
(7) DX

品質

(1) AI品質
(2) イノベーション品質
(3) 社会的課題の解決に関する品質

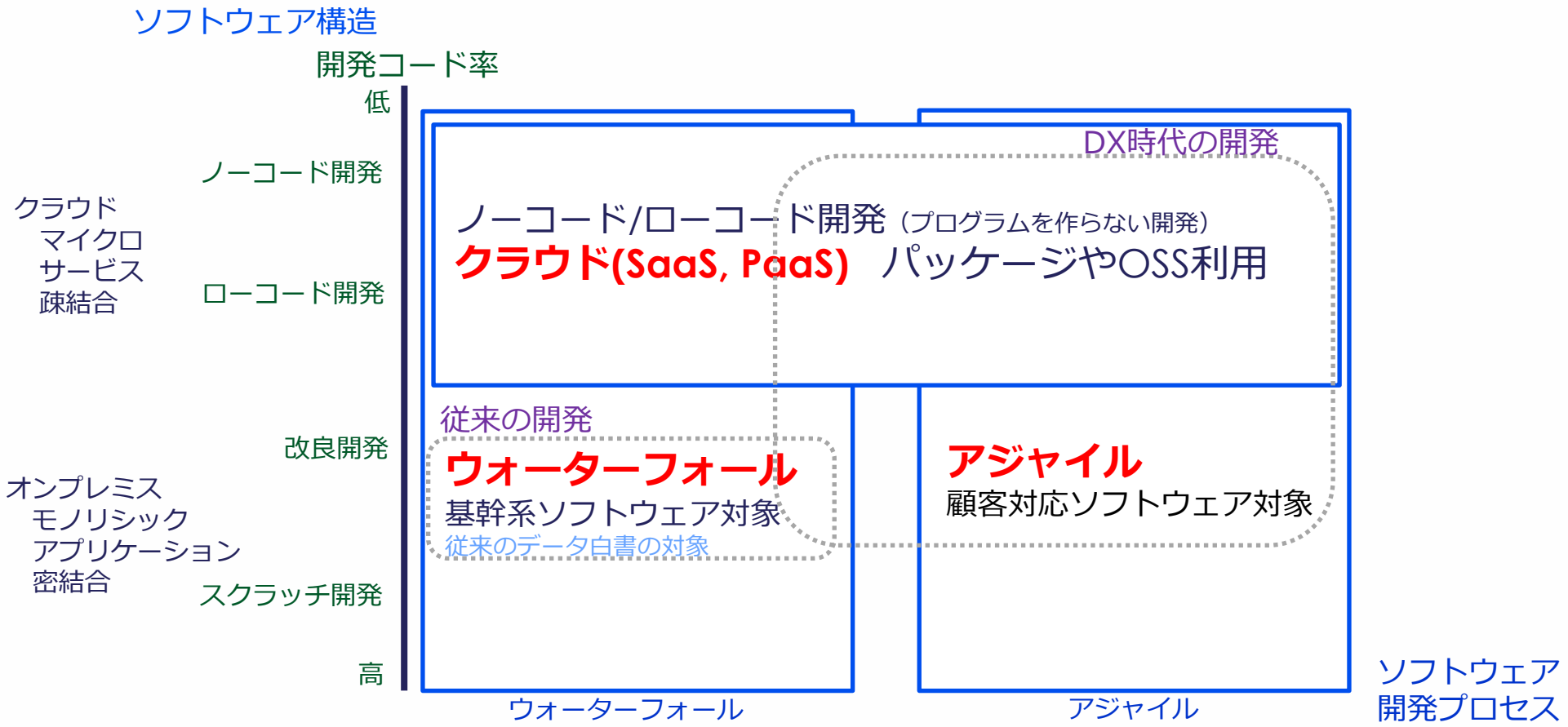
Society5.0と品質（WGでの議論）

Society5.0
・ドイツのIndustry4.0に対抗
・実態が不明

リファレンスアーキテクチャ
・Society5.0ジャパンのリファレンスアーキテクチャが必要
Society5.0ジャパンのあるべき姿を想定
・定義があいまいであれば「自分たちで」定義（するチャンス）
調査項目にも自分たちの定義で項目を入れる
・SDQ (SDGs、DX、QC）でモデル化
・Society/DX/QCの組み合わせ

Society5.0の実行主体
・DXは企業経営者が社員と進める
・Society5.0の実行主体者は？
行政？ 1企業？ 抽象的な社会全体？

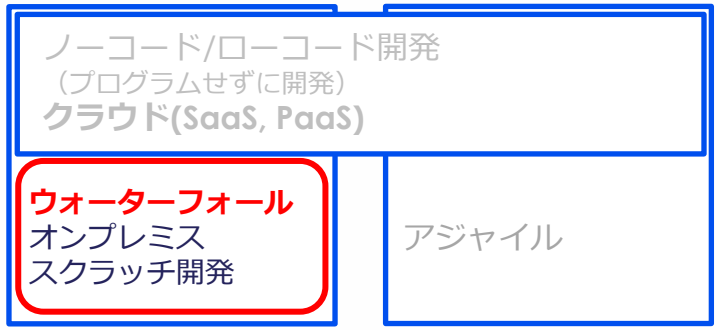
ソフトウェア構造/開発プロセスの分類



開発コード率はプロジェクトで手動開発するコードが全体のプロダクトに占める割合
ノーコード開発は SaaS のようにプログラムを作成しない開発
ローコード開発は PaaS のようにプログラムを少ししか作成しない開発
スクラッチ開発はゼロからプログラムを手動で作成する開発
改良開発は既存のコードを流用し、それを改良する開発

ソフトウェア構造/開発の変遷

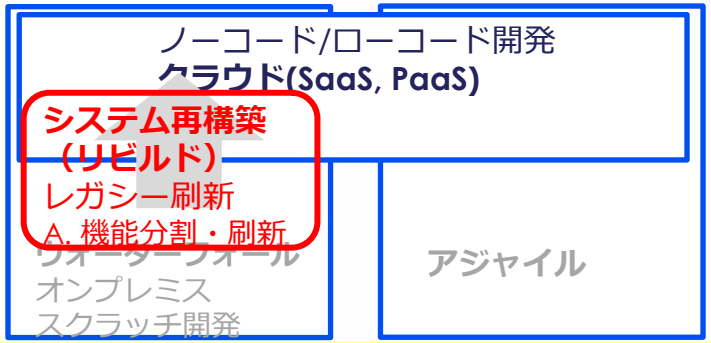
1. 従来のソフトウェア開発



QCDを重視
- ウォーターフォール型開発プロセス
- オンプレミス・モノリシック
- スクラッチ開発・改良開発

1→2
システム再構築へ

2. システム再構築によるレガシー刷新

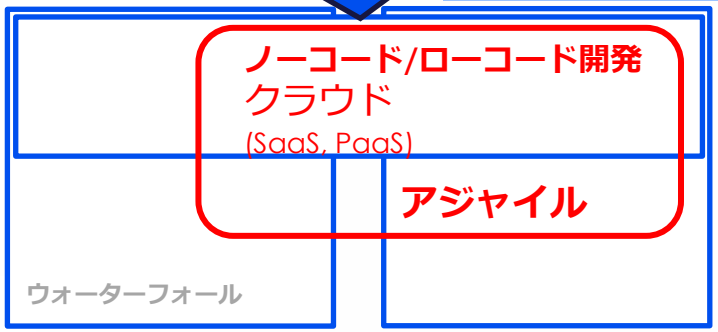


技術負債を解消し来るべき DX時代に備える
- レガシーシステムの再構築（リビルド）
- クラウド活用
- 機能分割（マイクロサービス化）や機能刷新
- DX時代のシステムにリビルトする

⇒　レガシー刷新

1→3
アジャイル/モーコード開発へ

2→3
アジャイル/モーコード開発へ



3. アジャイル開発/ノーコード開発

ビジネス環境の頻繁な変化に対応
- ユーザ主導のアジャイル型開発
- クラウド、SaaS、PaaS
- マイクロサービス

ウォーターフォール型開発も残る

アジャイル開発の特徴

アジャイル開発とDevOpsの永続的価値探索
・価値の探索は永続的

大規模アジャイル
・品質が問題
・アーキテクチャが必須　→　疎結合のモジュール化が可能
・インタフェースの品質が重要
・ジョブ型（ブロック型）が重要
　×メンバーシップ型/石垣型では、ビジネスがモノリス（一枚岩）型になる

アジャイル開発の品質（WGでの議論）

アジャイル開発
・アジャイルは本来は価値
　しかし開発プロセス（サイクル）がアジャイルになった
・アジャイルはビジネス単位（価値）と開発単位が１対１

アジャイル(SCRUM)開発の「ユーザストーリー」に注目
・ユーザストーリーは「誰が、なぜ、この機能を使うか」
・この「なぜ(要求)」が「価値(ビジネス価値)」
・「なぜ」を品質機能展開
・ユーザストーリーの最小機能単位=ファンクション
・「なぜ」がないユーザストーリーは間違い（警句）

アジャイル(価値)の計測
・ユーザストーリーはカスタマーとファクション、バリュー
・カスタマーポイントを入れて顧客満足度も価値に入れる
・ユーザーストーリーポイント=
　カスタマーポイント+ファンクションポイント+バリューポイント

アジャイル開発の品質での重要事項

・ユーザストーリーとデザイン思考

デザイン思考でユーザストーリーを考える
ユーザストーリーの作り方とその必要性を認識する
「なぜ（必要なのか）」の深さが足りない（デザイン思考で7枚くらいほしい）
＃ これを含めて人間系のものは「バスタブ曲線」になり適当な範囲が有効となる

・アジャイル開発の品質として「収益」

収益は状況や運にも大きく影響されるが大事な視点

・メンバがメンバーシップ型かジョブ型は大きな影響要因
・行動評価（プロセス評価）がやはり有効

サービス/データ/製品の品質（WGでの議論）

サービス品質（利用時の品質）
- サービス成熟度指標（DX推進指標のようなもの）
- サービスのリスクマネジメントやサービスのオペレーションも考慮
- サービスのメタモデル（サービス構造のモデル化）
  モデルの要素（人間、デバイス（モノ）、情報、システム）モデルの入出力（サービスのINPUT/OUTPUT)
- サービス構造の品質評価は少ない
- 利用品質で問題になる「顧客価値」：要求工学
- （アナログの）他分野のサービス品質を参考 ウェイターや半田付けのサービスコンテストなどの基準を参考

データ品質
- データの価値（利用性、用途性）を考慮すべき:無駄なデータや無意味に収集していないか価値評価が必要
- SQuaREのデータ品質特性に「適切性（価値）」を追加
  SQuaREの製品品質特性の機能適切性と同様（データ適切性）
- 適切性よりも強い「必要性」「価値性」のような名称がいい
- データの「無駄」を定義する（分析にまったく使われないデータなど）

製品品質
- 機能適切性はあるが、その他の主特性には個別の適切性はない
- 機能適合性以外の各主特性の適切性も考慮
- 利用時の品質には「満足性」がある

ユーザ主導の開発での品質 (WGでの議論)

□ 対象にしている典型的なシステム
・例. タクシーの自動配車のようなサービス中心のシステム
・ノーコード/ローコード開発のような開発中心の案件でないものも含む
・対象を想定して検討

□ ユーザ主導の開発での品質
・地方の中小企業ではユーザ主導の開発が増加
  ・ベンダーの役割
    共創、伴走、下請け（役割低下）
  ・ベンダーの品質活動
    ユーザビリティやセキュリティなどの付加価値の面
・プログラム作成の自動化の限界
  ・ベンダー：ものづくりのプロとしての支援

アジャイル開発の品質測定

□ アジャイル品質
・規模に相当するもの
  ・ユーザストリーポイント：相対評価
  ・コスト：QCDは基本、ただし他の要因が大きい
  ・カバレッジ：要求をどの程度カバーしたか
・ユーザストリーポイント(品質測定の母数になるもの)
  ・標準化するものではない
  ・ただしチーム内では標準化するのもあり
・価値：社会科学的アプローチ（アンケートの主観評価を定量分析など）
・チーム内評価：相対的
・非技術的要因：チームのエンゲージメント/健全性など
・プロセス監査：アジャイルプロセスをチェックリストやヒアリングなどで監査
  ・ただし定量評価とは言いずらい

□ （参考）ウォーターフォール型開発のメトリクス
・ウォーターフォール型開発も多く残存するのでこのメトリクスも重要

アジャイル開発の品質測定

1. 目的変数：品質の結果（価値）がわかる変数

(1) 最終リリース後のバグ密度（機能適合性） - 最終リリースがあるプロダクトの場合
(2) ユーザアンケート（個々の品質特性、満足度、価値）
(3) 使用回数や使用者数（品質の代替）
(4) 問い合わせ数（品質の代替）

2. 説明変数：品質を説明できる変数

(1) 工数、工期（、要員数）
(2) 繰り返し数、ユーザストーリーポイント、ベロシティ
(3) ウォーターフォールのメトリクス（検出バグ密度、レビュー指摘密度など）

3. ウォーターフォールとの差異

(1) ウォーターフォールはプロジェクト全体、アジャイルは現イタレーションの対象のみ
(2) アジャイルは作り直し部分も対象

アジャイル開発の品質測定 (WGでの議論)

- 品質で重要視　→　「保守性」
  - コード直接的なメトリクスでなくソフトウェア構造的なメトリクス
    - サイクロマティック数のようなコード直接的なメトリクスでなく
    - 「凝集度」や「結合度」のようなソフトウェア構造的なメトリクスがいい
      - 凝集度を計測するツール：Lattix や Understandなど
      - スプリント期間での変化を測定
      - 構造とバグ率の関係のMITの論文がある

- 価値の計測
  - カスタマーサポートに来るユーザからのメッセージをChatGPTで分類
    - 不具合、質問、苦情、要望
    - 苦情や要望には（対応者の）主観が入る
      - そこでChatGPTによる分類
        - ChatGPTでは不具合、苦情、質問などは別のものとして定義
    - 一律でクレームとしていたものを分類できた
    - この分類した件数で顧客満足度を計測する一つにする
  - 多様性が本質

DevOps/アジャイルの品質 by SQuBOK

□ DevOps
・バス 2016
・反復可能性、処理性能、信頼性、回復可能性、相互運用性、テスト可能性、変更可能性
・SQuBOK V3 pp.304-305

□ アジャイル
・QA to AQ (従来型の品質保証(QA)からアジャイル品質(AQ)) by Yoder
・主要パターン：体制、プロセス
・体制：QAを含むOneチーム、プロダクト品質チャンピオン、
　　品質エキスパートをシャドーイング、QAペアプログラミング
・プロセス：重要な品質の発見、品質ロードマップ、測定可能なシステム品質、
　　着陸ゾーンの合意、品質ストーリー、品質バックログ、
　　システム品質ダッシュボード
・SQuBOK V3 pp.306-307

・誉田直美「品質重視のアジャイル開発」 https://www.juse-p.co.jp/products/view/827

アジャイルメトリクスの例

(1) John Kammelar: Agile Metrics, Nesma, 28/04/**2015**.
http://nesma.org/2015/04/agile-metrics/
規模: フィーチャー、ストーリー
工数: ストーリーポイント
信頼性: 欠陥/イテレーション
生産性: ストーリーポイント/イテレーション(ベロシティ)
監視・制御：バーンダウン/イテレーション、チームベロシティ/イテレーション、バーンダウン/リリース、バーンアップ/リリース、テストケース数/イテレーション、開発サイクル時間、不具合解決時間

(2) アジャイルと品質会計，誉田直美(NEC)，情報処理学会デジタルプラクティス，Vol7,No3(July **2016**)
工数（人月）、ストーリー数、ストリーポイント/スプリント、SLOC、テストケース数、欠陥数、コードメトリクス（カバレッジ、ネスト数、複雑度）

(3) JUASユーザー企業ソフトウエアメトリックス調査**2016**
リリース回数、スプリント期間、
リリース当たりの各種データ（費用、画面数、帳票数、ファイル数、FP、データ要素数、他）

(4) Prabhat Ram, Pilar Rodriguez, and Markku Oivo: ”Software Process Measurement and Related Challenges in Agile Software Development: A Multiple Case Study”, 19th PROFES **2018**, pp.272-287.
132個のメトリクスで4社の比較
カテゴリはTesting Performance、Issues‘ Velocity、Code Quality、 Testing Status、 Development Speed、 Blocking Code、 Quality Issues’ Specification、Delivery Performance、 External Quality、 Issues' Estimation Accuracy

(5) (参考：アジャイルの定性的調査) VersionOne, The 17th State of Agile Report (**2023**)
https://digital.ai/resource-center/analyst-reports/state-of-agile-report/
経験、導入理由、分散開発状況、成熟度、利点、技法、手法、外注先適用度、失敗原因、拡大障壁、測定方法、日々測定、大規模適用技術、利用ツール、管理ツール、他

アジャイルヒアリング（対象：複数のアジャイル有識者）

・アジャイル開発のメトリクスや定量化の特徴や課題
(1) 基本的に定量管理は不要 （それよりもプロセスやルール、ツールが重要）
(2) ベンチマークのような定量管理は困難
(3) アジャイル開発を定量データによってガバナンスできない
(4) ウォーターフォールと目的は違うがアジャイルでもリスクやプロセス改善の可視化などのメトリクスは重要
・アジャイル開発の定量データの現状
(1) アジャイルガバナンスのデータとウォータフォール開発でのデータを収集
(2) メトリクスが未確定なので多めにデータ収集（しかし使えていないデータが多い）
(3) アジャイルメトリクス（収集項目も含む）は規定していない
(4) 従来のウォーターフォールのときの SLOCを収集しているが実態に合わない
・アジャイル開発の定量化の施策
(1) リスク検出や意思決定、改善の可視化では定量化は有用
(2) 大規模や高信頼化のアジャイル開発では定量管理が必要
(3) 計測の自動化が必要
(4) 絶対評価は困難なので相対評価の変化量を計測
(5) 開発プロセス内でリスクやパフォーマンスを可視化(改善のための見える化)
(6) メトリクスの目的を明確にする（リスクやプロセス改善の可視化、意思決定など）
・アジャイルメトリクスの例
(1) 仕様変更数や質問数、リリース頻度、ユーザ数などもメトリクスの対象
(2) 価値計測・メトリクスも必要 (例. ペルソナのフィット率(要求に対する質問数など))
(3) その他ストーリーポイントなどの一般的なアジャイルメトリクスのデータ項目
・(参考) アジャイル開発全般の特徴や施策
(1) アジャイル向きでないものを無理やりアジャイルで開発しない
(2) 仕様が確定すればウォーターフォールに戻す
(3) 大手ベンダのアジャイル開発は遅れていて参考にならない、一方Webサービス企業は参考になる

DXソフトウェア品質宣言（DX品質WGの提言）

(1) 機能品質から価値品質へ変革

従来の機能性を重視した製品品質から、ユーザ利用時品質やデータ品質を含む顧客・ビジネス・社会価値をベースにした品質に変革する

(2) 2極化するソフトウェア品質手法に対応

S級エンジニアによる専門的・高度な品質確保の手法

ノーコード・ローコード開発や生成AIなどに取って変わられる品質確保の手法

このように2極化する品質確保の手法に対応する

(3) 少数S級エンジニアの投資・育成

ソフトウェアの上流工程を担当する突出したS級エンジニアへの投資とその育成

ソフトウェアを作成したエンジニアに敬意を与える文化

(4) 割り切った迅速な判断で国際競争力向上

日本の強みは調整型の開発と言われているが、弱みである割り切り欠如の判断による開発の遅延

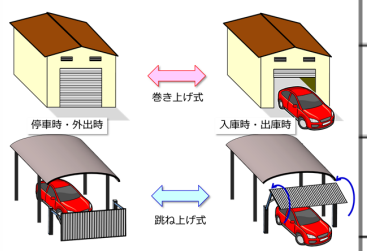
物事を割り切って迅速な判断を行い、アジリティを実現

品質公開問題「スマートガレージ」
WGでの検討問題として採用

ＪＥＩＴＡ スマートガレージ (2019年)

https://home.jeita.or.jp/cgi-bin/page/detail.cgi?n=1155&ca=1

現状の自動開閉ガレージシステムを各種の価値を付加したスマートガレージシステムに
JEITAスマートガレージでは以下のビジネスモデルを提案 (詳細は上記サイトを参照)



モデル名	想定するビジネス形態
ベーシックモデル	ガレージメーカーが、個人所有、個人利用するガレージオーナーに、ガレージを販売する売り切り型のビジネス。
保守機能搭載モデル	ガレージメーカーが、個人のガレージオーナーに保守機能を搭載したガレージを販売する売り切り型のビジネス。 ガレージオーナーへの付加価値として、保守機能を提供する。
共同利用モデル	ガレージメーカーが、第三者に個人所有のガレージを時間借りするサービスを提供するリカーリング型のビジネス。 ガレージオーナーにとっては、所有するガレージの賃貸によって得られる収入が付加価値となる。
サービス連携モデル	旅行会社が、共同利用モデルのガレージをマイカープランの駐車場確保に利用する協業型のビジネス。

公開問題

これらのモデルを開発・運用するときの
DX時代の品質観点を検討

・モデルに対応した品質観点
・モデル毎/共通の顧客価値に対応した観点
・（オプション）新規モデルの検討

検討例

・顧客(ガレージオーナー)価値観点
　基本：基本機能の適合性(コスト・時間)
　保守：上記＋保守機能の適合性
　共同：上記＋エンドユーザマッチング成果率
　連携：上記＋企業マッチング成果率
・観点の選択理由、計測手段も検討

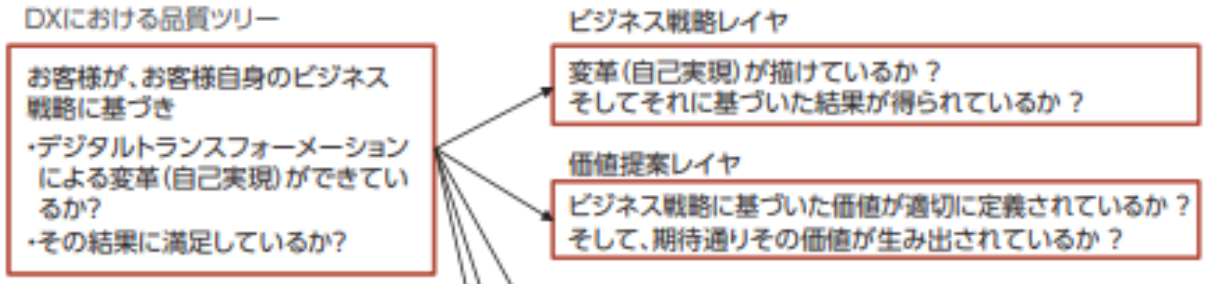
品質公開問題「スマートガレージ」コメント

ビジネス観点では収益性に収斂
サービス品質観点では社会性（社会貢献）も対象
安全・安心
GX/SDGs
エンドユーザには体験も価値
集まるデータに価値がある
民泊や日帰り旅行、移動の多様性：
ユースケース、それの品質
自動開閉：カメラで監視でも十分、コスパ
ビジネス・ユースケースの範囲
コンシューマ主体ビジネスのサービス品質
品質はトレードオフしたバランス
担保できない品質を外部委託・保険（尻ぬぐい）

まとめ：ソフトウェアエンジニアリングとメトリクス

1. 計測の重要性
   - ソフトウェアの開発プロセスやプロダクトに関する計測はいつでも重要
2. 従来の開発プロセスとメトリクス
   - これまではウォーターフォール開発でのSLOCやFPをベースにしたメトリクスを採用
   - これらのメトリクスは多大な蓄積があり見積りやプロジェクト管理で多用
3. 従来の課題と新しい開発プロセスとメトリクス
   - しかしアジャイル開発やノーコード開発などの流行で不整合が発生
   - アジャイル開発やノーコード開発では従来のメトリクスとは不整合
4. 新しいメトリクスの課題
   - しかし上記に合うベンチマーキングできる新しいメトリクスはなく、相対的なメトリクスで計測
5. ソフトウェアエンジニアリングの変革の必要性
   - そこでソフトウェアメトリクス、エンジニアリングの変革が必要
   - ウォーターフォール開発も含めたハイブリッド開発に関する上記のもの
   - ソフトウェアエンジニアリングの変革（手法やツール、プラットフォーム、人材、組織、国際競争/共創など）

（参考）NTTデータ 「DXの品質モデル」



p.40 図2 DX品質ツリー

・顧客
顧客の変革（自己実現）

・ビジネス戦略
ビジョンと結果

・価値提案
価値の定義と結果

・サービス
サービスのデザイン

・PoC
実現可能性

・技術
技術の選択と結果

p.43 ビジネス視点の品質評価項目例

p.43
・ヒアリングよるDXの価値を表出
質的分析手法（KHCoder、SCATなど）

ビジネスコミュニケーション　2021 Vol.58 No.1 pp.40-44　酒瀬川泰孝、坪井豊、寺尾健司「デジタルトランスフォーメーションの品質保証モデル構築」

(参考) SWEBOK v4 DRAFT Software Measures

CHAPTER 5 SOFTWARE TESTING　4. Test-Related Measures
coverage、SUT(*)の評価/テストそのものの評価
(*) System Under Test、テスト対象システム
CHAPTER 7 SOFTWARE MAINTENANCE　5. Measuring Requirements
Functional size measurement (FSM)
CHAPTER 9 SOFTWARE ENGINEERING MANAGEMENT　6. Software Engineering Measurement
対象：組織、プロジェクト、プロセス、製品

[9] Practical Software and Systems Measurement Continuous Iterative Development Measurement Framework Parts 1-3: Concepts, Definitions, Principles, and Measures, Version 2.1, April 15, 2021.

SWEBOK V3 2013
Chapter 5 Software Maintenance　2.4. Software Maintenance Measurement
Chapter 7 Software Engineering Management　6. Software Engineering Measurement
Chapter 8 Software Engineering Process　4. Software Measurement
Appendix B
ISO/IEC 19761:2011 Software Engineering—COSMIC: A Functional Size Measurement Method
ISO/IEC 20926:2009 Software and Systems Engineering—Software Measurement—IFPUG Functional Size Measurement Method

(参考)
新谷勝利　SEC Journal
SWEBOK V3.0の紹介　https://www.ipa.go.jp/archive/files/000066471.pdf
SWEBOK V3.0 日本語訳版の連続紹介
https://www.ipa.go.jp/archive/files/000063880.pdf
https://www.ipa.go.jp/archive/files/000063748.pdf
https://www.ipa.go.jp/archive/files/000063723.pdf

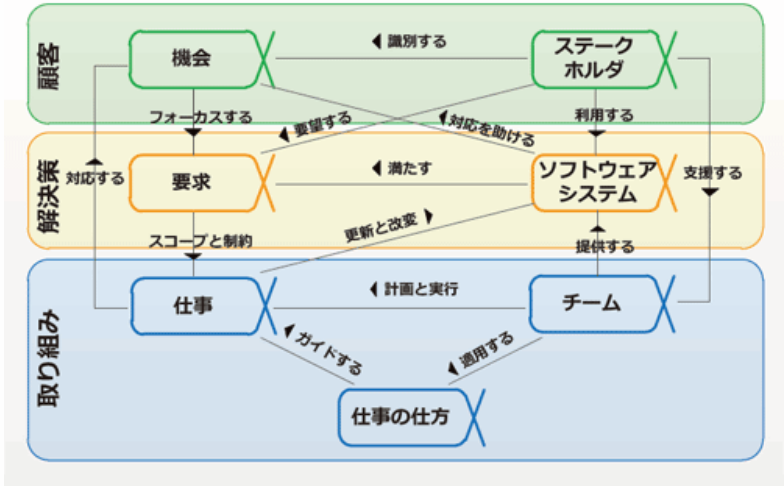
（参考）Essence/SEMAT

□ Essence

・OMG (Object Management Group) 標準、SEMATベースのソフトウェア開発方式
・https://www.omg.org/hot-topics/essence.htm
・開発の健全性と進捗を示す「**アルファ**」
alpha (AspirationLed Progress and Health Attribute)：願望に導かれた進捗と健全性（言葉優先？）
→ **機会、ステークホルダ、要求、ソフトウェアシステム、作業、作業方法、チーム**
・ソフトウェアエンジニアリングはソリューションにより価値を提供
・直接：要求とソフトウェアシステム　間接：その他のアルファ
・価値の源泉：機会、そのベースとなるステークホルダ
・アルファの考慮状況（アンケート結果）：(多)要求、システム ⇔ チーム、機会(小)
→　価値をモデリングするときの対象としてのアルファ
・SES2020 鷲崎先生（SE4BS）https://ses.sigse.jp/2020/wp-content/uploads/2020/09/WS4-3.pdf

□ SEMAT (Software Engineering Method and Theory)

・ソフトウェア開発方式
・Ivar Jacobson、Bertrand Meyer、Richard Soleyが提唱
・理論に裏打ちされた実践的なソフトウェア開発方式
・https://xtech.nikkei.com/it/article/COLUMN/20130526/479662/
・アルファ（上記参照）：活動目標
・(1) 開発の健全性と進捗を (2)肯定的な成果で
(3) 個別ではなく全体を示す
・原則： 活動可能性、拡張性、実務性
・https://www.bcm.co.jp/site/youkyu/youkyu103.html (山本先生の解説)



（参考）海外のソフトウェア工学機関

機関名	組織名	内容
ドイツ 情報分野特化型 フラウンホーファー	実験ソフトウェアエンジニアリング研究所 (IESE)	・ソフトウェアエンジニアリング ・Insustry4.0のアーキテクチャ (モデルベース分析・シミュレーション) ・ビッグデータ(データ品質、可視化) ・スマートエコシステム (モデリング、アーキテクチャ、ソフトウェア開発) (JEITAソフトウェア事業基盤専門委員会と共催ワークショップ実施 2007年)
アメリカ カーネギーメロン大学(CMU)	ソフトウェア・エンジニアリング・インスティチュート(SEI)	・国防総省(ＤｏＤ)が設立、予算を拠出 ・ソフトウェアエンジニアリング (アーキテクチャ、コスト見積り) ・モデルベース開発･テスト ・調達ガイド策定 ・組込み・制御システム ・セキュリティ・バイ・デザイン ・サイバーセキュリティ対策(コーディング規約等) （IPAメトリクスチームと定期ディスカッション 2016～2018年）
アメリカ 国立標準技術研究所 (NIST)	NIST-EL エンジニアリング部門 NIST-ITL/SSD ソフトウェア・システム部門	・CPS ・製造情報システム統合エンジニアリング ・システムズエンジニアリング (スマート製造システムのモデリング) ・超大規模システム ・ソフトウェア・アシュアランス (メトリクス、アルゴリズム) ・ヘルスケア・IT連携
アメリカ The International Council on Systems Engineering (INCOSE)	日本事務局: 慶應SDM	・MBSE　モデルベースシステムズエンジニアリング ・システムズエンジニアリング
韓国 National IT Industry Promotion Agency (NIPA)	ソフトウェアエンジニアリングセンター(SEC)	・ソフトウェアプロセス品質認証制度 ・QMO支援（品質管理の支援） ・ソフトウェア開発のスキル標準の展開・適用 ・海外の先進技術の導入支援。 ・ソフトウェアバンク（研究開発の結果を蓄積し、関連事業に活用する仕組み）

（参考）学会のソフトウェア工学の分類

学会	参照	大分類
情報処理学会 (IPSJ)	https://www.ipsj.or.jp/journal/submit/manual/j_keyword.html	大項目の「ソフトウェア工学」の中には以下の中項目とさらに小項目がある 要求工学 分析・設計技法 形式的手法 テスト技法・保守技術 システム評価・管理技術 プログラミング言語の実装技術 開発支援環境・自動化技術 ソフトウェアプロセス システム運用技術 小項目は省略 なお「モデル化」は小項目に分散
Association for Computing Machinery (ACM)	https://cran.r-project.org/web/classifications/ACM.html	大項目「Software」の中項目「Software Engineering」には以下の小項目がある D.2.0: General D.2.1: Requirements/Specifications D.2.2: Design Tools and Techniques D.2.3: Coding Tools and Techniques D.2.4: Software/Program Verification D.2.5: Testing and Debugging D.2.6: Programming Environments D.2.7: Distribution, Maintenance, and Enhancement D.2.8: Metrics D.2.9: Management D.2.10: Design D.2.11: Software Architectures D.2.12: Interoperability D.2.13: Reusable Software D.2.m: Miscellaneous

(注) 電子情報通信学会は「ソフトウェア工学」が最小項目
https://www.ieice.org/jpn_r/shiori/faq_transactions.html

（参考）SWEBOK v4 DRAFT contents

contents

（参考）MBSE Model-Based Systems Engineering

MBSE
・Model-Based: モデル(IPAの文献ではSysML＆２元V字モデル推し)をベース
・Systems Engineering: 複数の分野を統合開発
　・異種複数システム、大規模開発、高品質開発向き：ロケットなど

2元V字モデル Dual Vee Model
・アーキテクチャーV字(システム、サブシステム、コンポーネントのV字)と
　エンティティV字（システム、各サブシステム、各コンポーネント単位のV字）の２元V字モデル

SysML Systems Modeling Language
・要求図、ユースケース図、ブロック定義図（クラス図）、シーケンス図、状態マシン図、内部ブロック図（配置図）、パラメトリック図、アクティビティ図などで記載
赤字はSysMLで追加、青字はUMLから変更（括弧内はUMLでの名称）

団体・組織
・INCOSE The International Council on Systems Engineering
・慶応大学 SDM (白坂先生)

参考文献
システムズエンジニアリングの推進(2018) https://www.ipa.go.jp/archive/digital/iot-en-ci/se.html
モデルベースシステムズエンジニアリング 導入の手引き(2013) https://www.ipa.go.jp/archive/files/000033609.pdf

（参考）テストケース粒度の標準化

ITシステム可視化協議会 (旧)日本ファンクションポイントユーザーズグループ

テストケース粒度の標準化について

https://www.mcis-jp.org/_files/ugd/d8abb8_b34bc686c2ae46e3be0d74cb1092b4b0.pdf

内容

・テストにおける課題（妥当性、テスト遅延、潜在不具合、再テスト）
・問題提議（テスト粒度の不統一）
・各工程におけるテストの目的と内容（単体/結合/総合テストで粒度を統一）
・トランザクションフローテストにおける粒度定義
　フロー中の分岐・結合点での網羅テスト
　データアクセスを伴う機能の組合せをテストケースの最小粒度と定義
　FPで補正したCRUD表からテスト粒度を統一

ITシステム可視化協議会 (MCIS)　(旧)日本ファンクションポイントユーザーズグループ (JFPUG)
https://www.mcis-jp.org/

（参考）IPA ソフトウェア開発文献集

■ソフトウェア開発調査

https://www.ipa.go.jp/digital/chousa/software-engineering/result_software-engineering2023.html
新時代のソフトエンジニアリングの動向をアンケート調査、結果をオープンデータとして公開

■ソフトウェア開発分析データ集 https://www.ipa.go.jp/digital/chousa/metrics/

ソフトウェア開発プロジェクトの定量データの分析結果を公開

■組込み/IoT産業動向調査 https://www.ipa.go.jp/digital/chousa/kumikomi/

組込み/IoT産業のソフトウェア開発を中心にDXなどの取り組みをアンケート調査、結果を公開

（参考）ITインフラや性能の品質を考える情報

■非機能要求グレード

https://www.ipa.go.jp/archive/digital/iot-en-ci/jyouryuu/hikinou/ent03-b.html

情報システムの安定的なサービスが求められるようになっており、複雑なシステムを構成する多様なコンポーネントがきちんと連携してそのようなサービスを提供する「システム基盤」の実現が重要になっている。「非機能要求グレード」は、「非機能要求」についてのユーザと開発者との認識の行き違いや、互いの意図とは異なる理解を防止することを目的とし、非機能要求項目を網羅的にリストアップして分類するとともに、それぞれの要求レベルを段階的に示したものになっている。

■性能エンジニアリング入門

https://atmarkit.itmedia.co.jp/ait/series/2418/

ピーク時になると応答時間が急激に悪化したので、とりあえずCPUとメモリを倍増しておけば大丈夫ではないかと勘に頼って対応し、ドツボにはまった経験はないだろうか。これは性能問題について最低限理解しておきたい基礎を解説したものになっている。

執筆

DX時代に求められる品質
2024年4月10日　発行

DXITフォーラム DX時代の品質課題を考えるワーキンググループ(品質WG)（順不同）

主査	五味 弘	独立行政法人情報処理推進機構
	小西 広朗	株式会社電算システム
	齋藤 幸久	株式会社電算システム
	金子 博	株式会社東芝
	三部 良太	株式会社日立製作所
	野尻 周平	株式会社日立製作所
	片岡 祥啓	株式会社日立製作所
	岡田 侑里	株式会社日立製作所
	河野 太基	富士通株式会社
	佐々木 崇晃	みずほリサーチ&テクノロジーズ株式会社
	田中 秀人	独立行政法人情報処理推進機構
	溝口 則行	独立行政法人情報処理推進機構
	滝田 典子	独立行政法人情報処理推進機構
	山崎 昭司	独立行政法人情報処理推進機構
	西本 靖	独立行政法人情報処理推進機構
アドバイザ	山本 修一郎	名古屋国際工科専門職大学