新規プロダクト開発における 品質向上のための プロセス改善に関する取り組み

ソフトウェア品質シンポジウム 2023（SQiP2023）

エクスジェンネットワークス株式会社

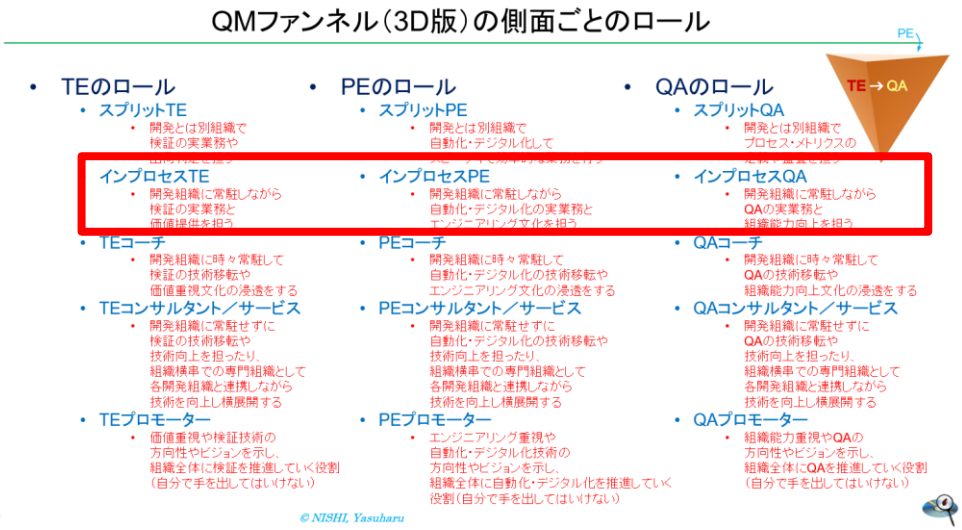
LDAP Manager 部

井関 武史 : iseki@exgen.co.jp

- はじめに
  - 自己紹介
  - 開発製品特徴
- 目的
- 背景
- 現状分析・整理
- 施策 (方針)
- 対策
- まとめ

はじめに (自己紹介)

- エクスジェンネットワークス株式会社
  - LDAP Manager 部
  - BtoB 向け ID 管理パッケージ製品
    - テスト･品質保証(管理) 担当



- 社外活動
  - SigSQA
  - テストの街「葛飾」

SigSQA

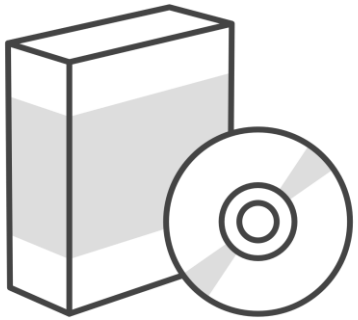
はじめに (開発製品特徴)

パッケージ製品

- 機能の抽象化の必要性
- ビックバンリリース (部分的なリリースができない)
- 一度リリースした機能の変更 (削除) が困難
- 厳密な (設定・動作の) 例外処理の考慮
  - システムに組み込むのは Sier 様 (社内の技術者が構築しない)
- トラブル対応のための情報収集機能の考慮 (保守の考慮)
- 継続的な製品の拡張、新しい環境への対応性

BtoB 製品

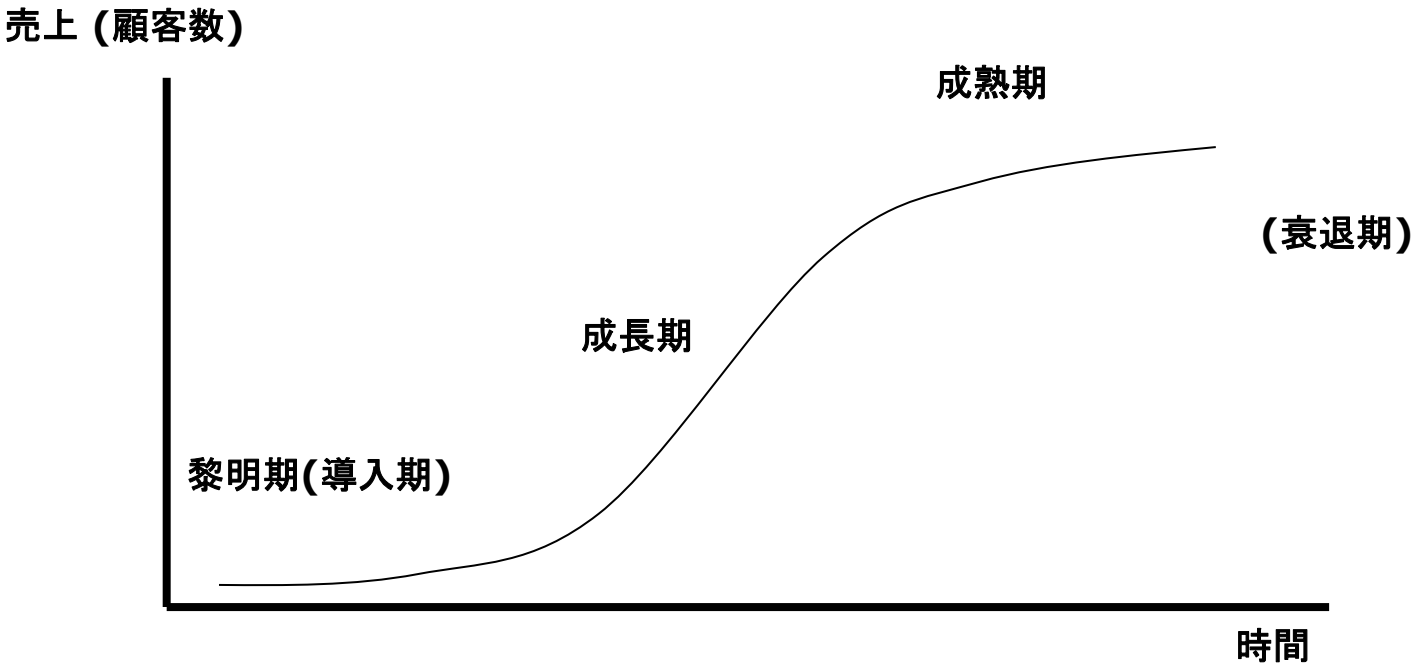
- 保守 (サポート) 期間が比較的長い (5～10年)
- 大規模システム
- 厳密な品質説明の必要性
- 業務プロセスに組み込むための (ある程度の) サポート
  - 周辺システム・ツールなどとの組み合わせ



背景

- プロダクトの成長 (機能の増加) に伴い、顧客サポートにおいて、プロダクトを利用した構築・運用の問い合わせが多様化、複雑化

- 問い合わせ内容の調査や解析、プロダクトのバグ修正・仕様変更、そしてリリース作業も含めたプロダクト保守業務において、非常に複雑化、煩雑化、困難化し、工数が増大しリードタイムも長くなる傾向



背景

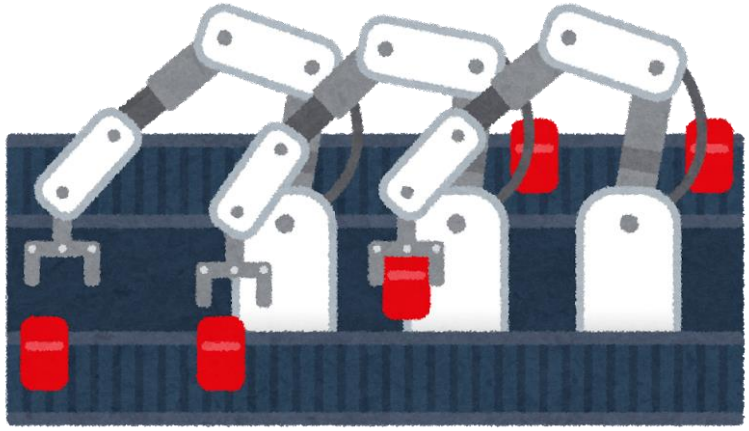
- プロダクト黎明期、成長期(初期) で開発プロセスが定義されておらず、設計書 (仕様書) もほぼ存在せず、製品 (機能) を追加・修正、そしてプロダクト保守を行っていた。
  - 仕様の共有認識や品質概念の欠如
  - リリース基準の認識相違
  - プロダクトの成長計画の考慮不足

- プロダクト成長期(後期)、成熟期に入ると以下のような状態が発生
  - 問い合わせの調査で仕様、またはバグであるかを判断することが難航
  - リリースするプロダクトの品質が安定しない (開発・テスト担当者ごとの部分的な仕様)
  - 局所的なバグ修正・仕様変更に伴いデグレ発生
    - 影響を考慮しようとすると過度に工数の増大
  - 機能ごとに仕様が異なりプロダクトの動作や目的の正しさがあいまい

- 修正・リリースを繰り返すことで、顧客サポートにおいて非常に多くの工数が必要

目的

- 背景にある問題を低減するために、プロダクトを開発を行い、安全・安心にリリースを行うプロセスを用意
  - 担当者の技能 (スキル) だけに頼らずプロダクト品質を一定以上にすること
  - プロダクトのリスクを把握して可視化すること
  - プロダクト保守にも寄与するべくプロダクトの品質向上すること



現状分析・整理

- バグ修正・仕様変更しリリースを行う工数
  - 年間平均で 1020.575 時間 (約 5.5 人月強)のバグ修正・仕様変更
  - バグ修正・仕様変更の対応工数
    - ※)負債対応率 = (バグ対応工数 + 仕様変更工数) / メンバー全工数

年度	バグ修正 工数（H）	仕様変更 工数(H)	対応メンバー (総工数: H)	負債対応率
2019	798.5	332.00	—	(計測不能)
2020	620.75	283.75	—	(計測不能)
2021	663.75	389.00	4 (7,680)	13.71%
2022	706.5	288.50	3 (6,480)	15.36%
(平均)	697.375	323.20	—	

現状分析・整理

- サポート問い合わせの調査・分析・調整工数
  - 100チケット/1か月程度の問い合わせ
  - 100～200時間/1か月の工数

年度	サポート対応工数(h) [チケット数]
2019	1843.00 [1057]
2020	2482.75 [1167]
2021	2132.25 [1324]
2022	1411.75 [964]
(平均)	1967.25 [1128]

- 設計・マニュアル修正なども含めたリリースにかかわる工数

年度	設計・マニュアルなど修正工数 (h)	リリースにかかわる工数 (h)
2019	406.50	124.50
2020	370.50	113.25
2021	390.00	168.50
2022	135.50	110.00
(平均)	325.5	129.06

現状分析・整理

- 保守業務としての総工数
  - 保守業務として年間 21人月程度。(1人月160h として計算)

年度	保守工数 (h)
2019	3504.00
2020	3871.00
2021	3743.50
2022	2382.25
(平均)	3375.06

- プロジェクトメンバー9名(開発6、テスト・QA2、PO)で、1か月に2名程度必要
  - プロダクトの調査、実証確認は開発・テスト系の担当者である必要性

- 保守専属の開発・テストのチームを編成 (開発3、テスト・QA1)
  - 新規開発チーム (開発3、テスト・QA1、PO)と分離。

現状分析・整理

- 保守で修正を行うバグ・仕様変更がどのようなことで起因するかを分析
  - 上流工程の問題であることは体感で理解できるが実情を可視化
  - 2019年9月から計測開始

不具合要因	個数	割合
設計検討不足	144	**62.0%**
設計間違い	15	6.4%
実装ミス	50	**21.5%**
テスト不足	21	9.1%
外部要因	2	1.0%
合計	232	

- 明らかに設計 (ドキュメント作成) を行っていないことが起因

現状分析・整理

- 設計検討不足
  - ユーザーストーリー(要求分析・要件定義)など検討不足による実装機能不足
  - 一定条件のみでしか動作しない設計
  - 考慮されない例外
  - 出力されないログ (状況の解析ができない)
- 設計間違い
  - 背景・理由の表現不足による意図が異なった実装
- 実装ミス
  - 実装できることに注力して目的を見失った実装・テスト (ビルドトラップ)
  - 設計がないための情報欠如・誤認や思い込みによる実装間違い
  - 考慮されていない内部設計・品質 (モノリシス設計) による実装漏れ・実装間違い
- テスト不足
  - 設計書がないため担当者の誤認、誤謬
  - テスト要求分析ができていないための認識不足

現状分析・整理

- 積み重ねられた負債の対応による保守工数増大
  - 調査・実証確認の複雑化・肥大化
  - プロダクトの新規開発の体力低下
- 内部品質向上業務が膠着
  - リファクタリング・リグレッションテストなどの内部品質向上の業務が膠着 (モノリシス)
  - 潜在的バグ修正が困難 (逆に埋め込むデグレ)
- 顧客への価値提供機会の低下
  - 新技術 (環境) の対応遅れ
  - 新機能の実装見送り

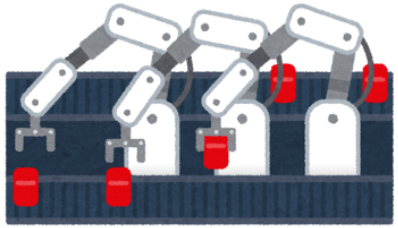
施策 (方針)

- 目的を達成するため、以下の3つの定義を行い、開発チームの規律・基準・倫理を定義をして時間をかけて文化に醸成する

- プロダクトの価値(品質) ・クラフトマンシップの定義
- プロダクト製造プロセスと成果物の定義
- テストプロセスの改革・品質保証方法の定義

目的

- 背景にある問題を低減するためにプロダクトを開発を行い、安全・安心にリリースを行うプロセスを用意
  - 担当者の技能 (スキル) だけに頼らずプロダクト品質を一定以上にすること
  - プロダクトのリスクを把握して可視化すること
  - プロダクト保守にも寄与するべくプロダクトの品質向上すること



対策

- 開発チーム (プロダクトを新規に作るチーム) でのミッション
  - 顧客に価値をすばやく届けることは重要
  - 保守業務などにも寄与できるプロダクト開発であること
  - 安心 (納得) してプロダクトをリリースできる状態にすること

- 実践
  - プロダクトの価値(品質) ・クラフトマンシップの定義
  - 品質概念・テスト・バグなどの意識 (マインド) 改革
  - 開発プロセスと各プロセスにおける成果物の定義
  - テストプロセスの改革
  - (プロセスの)継続的改善・ふりかえり

対策

- プロダクトの価値(品質)・クラフトマンシップの定義
  - [基準] プロダクトの品質をどのような状態にしたいのか・どのようなプロダクトにしたいのか
  - [倫理] どのようなプロダクト (価値) を世の中に届けたいのか
  - [規律] そのためのものづくりをするためには何が (どのようなプロセスが) 必要か

自分たちのマインド

思想(思考)、哲学、意思というものは、目標を達成するにあたり非常に重要なものである。

偉い人も言っているけど、そもそもの思考が他人事だったり、人任せだったり、人(他の事)のせいだったりすると、その言動に関連してくという話。

思考に気をつけなさい、それはいつか言葉になるから。
言葉に気をつけなさい、それはいつか行動になるから。
行動に気をつけなさい、それはいつか習慣になるから。
習慣に気をつけなさい、それはいつか性格になるから。
性格に気をつけなさい、それはいつか運命になるから。

そこで、(開発チームに限らないけど) 自分たちがモノづくりをして、サポートしていくためのマインドセットを以下に書いて共有している。

開発チーム よりよい品質のプロダクトを作るためのマインドセット

さらに、開発担当者としての心得は以下として共有している。

開発チーム 業務概要 開発者の心得

目的

新規開発でリリースをする機能やモジュールの品質を高めて後戻り工数などを削減する。

不具合修正に伴う保守工数を削減しサポートを円滑にする。

スローガン的なものは以下。

- 自分たちで価値 (売れるもの) を考えて機能として実装する (どのような品質が必要であるのか)
- ニーズに合った機能（製品）を安心して納期にリリース（そんな機能使えない、リリースおせぇー、とか言わせない）
- リスクを識別し (プロセス内で) 軽減させ重大な不具合はなくして安全に利用してもらう（バグ多いとか言わせない）
- 不具合があっても、リスクを回避できるようにし、アタフタしなくてもいい（その不具合、知ってるしー。すぐ再現できるし回避策も書いているしー）

対策

- 品質概念・テスト・バグなどの意識 (マインド) 改革
  - 「品質 (価値) は製品にかかわるメンバー全員のマインドと、そこから生まれたプロセス・システムに宿る。」というマインドセットの定義と共有

開発チーム よりよい品質のプロダクトを作るためのマインドセット

所有者: Takefumi iseki •••
最終更新日：2023/05/01 • 12 人が閲覧 • Page numbering

目次

はじめに

「品質 (価値) は製品に携わるメンバー全員のマインドと、プロセス・システムに宿る。」

対策

❑ 品質概念・テスト・バグなどの意識 (マインド) 改革

■ ISO で定義されている品質特性を利用して自分たちで必要な品質を再定義

品質特性	品質副特性								
機能性	正確性	合目的性	相互運用性	セキュリティ	適合性				
信頼性	成熟性	障害許容性	回復性		適合性				
使用性	理解性	習得性	運用性	魅力性	適合性	ユーザエラー防止性	満足性	補助性	
効率性	時間効率性	資源効率性			適合性				
保守性	解析性	変更性	安定性	試験性	適合性	作業効率性	拡張性	継続性	メンテナンス性
移植性	環境適応性	設置性	共存性	置換性	適合性				
プロジェクト遂行性									

ISO/ISE 9126 で定義されている品質特性

経済産業省が定義している品質特性

EXGEN 独自の品質特性

EXGEN 独自の品質特性

品質特性		ISO/ISE 9126・JSTQBの定義	EXGENでの定義
機能性		暗黙のユーザー要求または定められたユーザー要求を満足させる機能の必要かつ十分な実現に関する属性 ソフトウェアがユーザからの機能的な要求(明示的機能と暗示的機能)を満たす度合い。 ユーザ目的に合った正確な動作を満たすか。外部IFや規格、セキュリティ基準を満たすか	
	正確性	合目的性で実現されていることが確認された機能に対して評価 以下のようなこと • 伝言ゲームのように情報が変化しない ◦ データが欠けない ◦ 意図しないデータ(登録する場所が異なる)に変化しない ◦ 誤差が生じない • 設計と同じデータ形式 (値) を受信しているか 示的または暗黙的な要件にアプリケーションが準拠していることをテストする。また、計算の正確性もテストする。 仕様または旧システムの実行 (テスト) 結果をテストオラクルとして使用する。	仕様書・設計書、および、テストアーキテクチャ設計であげられたテスト条件 (テストパターン) に記載した機能が意図した動作になっていること また、何度動作させても同じ結果を得ることができること 基本は、この正確性は開発担当者がユニットテスト、結合テストのテストレベルでテストを行うものとする
	合目的性	使用目的に対して要求される機能が、ソフトウェアに必要十分に実現されていることに対してを評価 JSTQB TAシラバスより (http://jstqb.jp/dl/JSTQB-Syllabus.Advanced_TA_Version2012.J01.pdf)	ユーザーのニーズを満たすためにシステムが実行しなければならない機能が実現されているかを確認すること。 基本は、テスト担当者がシステムテストのテストレベルで確認をする。

対策

❑ 開発プロセスと各プロセスにおける成果物の定義

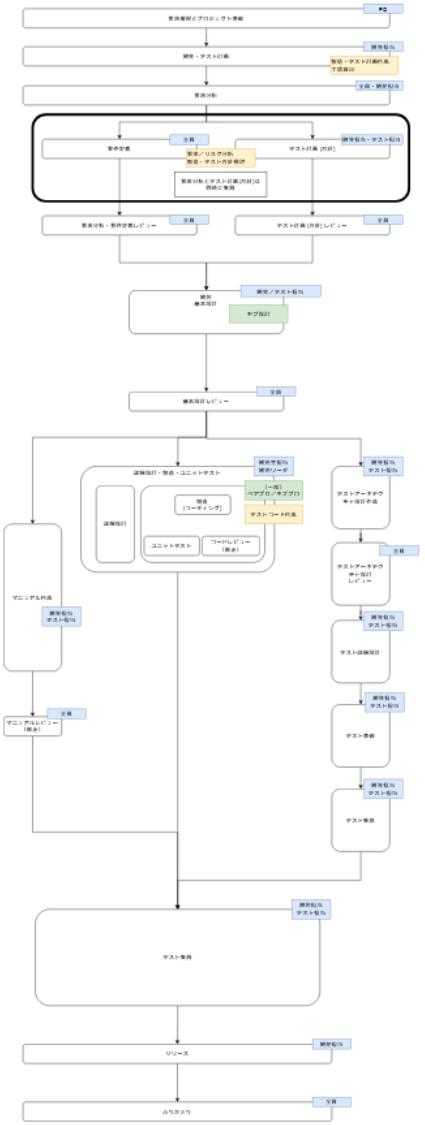
- ■ キックオフからリリースまでの各プロセスの定義と役割(責任)の定義
  - □ キックオフ
  - □ 要求分析・要件定義
  - □ 開発設計 (基本設計・詳細設計)
  - □ テスト設計(方針・要求分析・アーキテクチャ設計・詳細設計)
  - □ テスト実施
  - □ リリース
- ■ 各プロセスの成果物の定義

例) 基本設計の成果物

基本設計項目

以下のことを考慮して設計を行う。

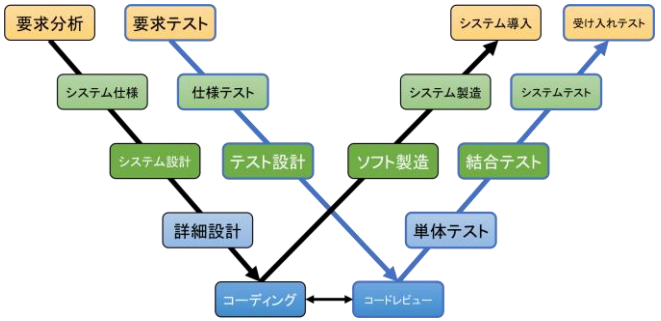
- システムの目的と品質要件
  - 背景や意図
- システムの動作要件 (環境)
- システムの全体像 (構成)
- システムの構成ごとに、どのような機能が必要か
- それぞれの機能の入出力(入出力先、入出力形式)
- 画面一覧と画面表示項目、画面遷移
- プログラム停止の方針
  - どのような機能で、どのようなエラーが発生した場合に即時停止
- セキュリティ
  - 作成・改修する箇所にどのような脆弱性の可能性があり、対策をどのようにするか

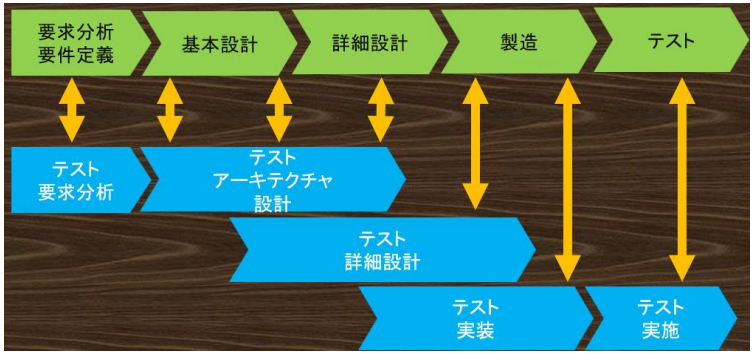


対策

テストプロセスの改革・品質保証方法の定義

- テストプロセス (役割) を定義
  - 要求分析、アーキテクチャ設計、詳細設計、実装、実施
- 開発プロセスと同時にテスト設計 (シフトレフト)
  - 設計と同時にテスト設計開始
- テストの構造化
  - VSTeP によるテスト観点の構造化
  - NGT 表記による機能性テストのMECE化
- 探索的テスト
  - CPM 法テストからの脱却
    (○×のチェックするテストのみからの脱却)
  - 受け入れテストの考慮
- コンテキストに合わせた保証する品質の合意
  - プロダクト成長度合い、特徴、特性による担保する品質、リスクの受容(シフトライト)
    - AQUA フレームワーク





対策

- (プロセスの)継続的改善・ふりかえり
  - 各プロセスでの成果物の妥当性の確認
  - インシデント発生と原因究明
    - バグ・要望のバックログ登録
  - (技術・設計・品質) 負債の共有
    - リスクの把握
  - 長期的、または短期的(タスク化)する改善点の共有と対策案などの検討

結果

- 開発プロセスの定義・構築を行い図やドキュメントで可視化することで、プロジェクト関係者は従来の業務の標準化され改善点が明確となった
  - リスクを設計・開発へフィードバック
    - 設計・実装中にフィードバックすることで後戻りの低減
    - 将来へのリスク把握 (バックログ登録)
  - 成果物のレビュー精度向上
    - どのプロセスで何をどのようにレビューをしないといけないか判断しやすい
    - 記載内容・記載ドキュメントが確定しているため漏れない
      - 存在しないものは理由付けされる

- 管理・保証すべき品質を把握し、逆に保証できないこと (リスク) を明確化することで、安心・安全にリリース
  - プロダクトの品質 (価値) の共有
    - どのようなことを解決したいのか、そのためにどのような機能が必要なのか
      メンバーで共通認識
  - 各プロセスの実施目標の共有
    - どのようなドキュメントを用意しないといけないのか・どのようなことを
      検討すべきなのかメンバーで共通認識

結果

- 設計不足によるバグ・仕様変更が減少しはじめた
  - 設計不足の理由の質も変化 (「設計・実装・テストしない」など要件定義・設計に理由付けされて記載しているため判断できる状態)
    ※) 2019年から開始されているので母数が少ない

2021年度

	4	5	6	7	8	9	10	11	12	1	2	3	
設計検討不足	4	3	1	4	1	2	2	4			1	3	25
設計間違い		1							1 (1)		1	1	4
実装ミス	2	2		3 (3)	1			1					9
テスト不足					4			1					5
外部要因													
合計	6	6	1	7	6	2	2	6	1		2	4	43

※) 開発プロセス再定義しているときのバグ。 (n) と記載しているものが新しいプロセスで開発をしたときのバグ

2022年度

	4	5	6	7	8	9	10	11	12	1	2	3	
設計検討不足	4(1)	2	2	2	11	1	12(3)	7	1	12	20(2)		74
設計間違い	1	1								1			3
実装ミス	1	1			1			2			3		8
テスト不足													
外部要因										1	1		2
合計	6	4	2	2	12	1	12	9	1	14	24		87

※) 開発プロセス再定義しているときのバグ。 (n) と記載しているものが新しいプロセスで開発をしたときのバグ

課題 (これから)

- シフトレフトを目指してプロセスを再定義した結果、シンプルなプロセスの構造には至らず実施すべき内容が上流工程で多く、複雑な状態 (トップヘビー)

- 自動化なども含めて開発担当者のみで対応できるプロセスにシンプルに改善していく必要あり

参考文献

- VSTeP・NGT
  - 電気通信大学 西康晴氏 ソフトウェアテスト開発方法論・テスト設計技法
  - https://www.slideshare.net/YasuharuNishi/vs-te-p130510
- AQUA フレームワーク
  - 電気通信大学 西康晴氏
  - https://www.slideshare.net/YasuharuNishi/line-developer-meetup-in-tokyo-39-presentation
- QM ファンネル
  - 新しい品質保証の形を目指して　Sig-SQA
  - https://www.jasst.jp/symposium/jasst22tokyo/details.html