SPI Japan 2025

TDDベースの高品質アジャイルにおける品質管理プロセス構築の取り組み

2025/10/23

株式会社NTTデータ
掛川悠

目次

略称	
プロジェクト	➡ PJ
ウォーターフォール	➡ WF
テスト駆動開発	➡ TDD

01

はじめに

アジャイル開発を取り巻く状況

ビジネス環境の変化に伴い、従来WF型の開発が主流であった大規模ミッションクリティカルシステムにおいても
アジャイル開発へのシフトが進んでいる。

こうした**ミッションクリティカルシステムにおけるアジャイル開発**では、柔軟性、アジリティといったアジャイルのメリットを
享受しつつも、WFと同等の品質を求められる、いわゆる**品質重視のアジャイル開発**になることが多い。

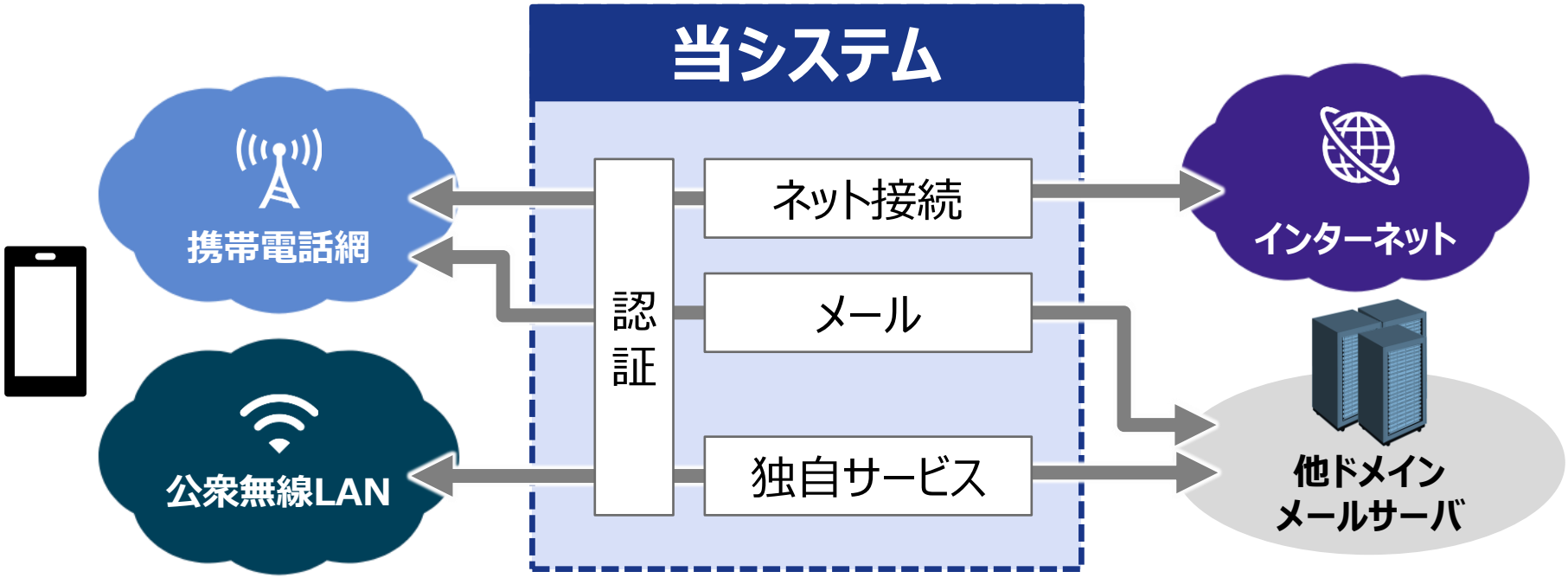
ミッションクリティカルシステム＝ **高品質**

02

PJ概要

システム概要

某通信キャリア向けに携帯電話（スマートフォン・タブレット・フィーチャーフォン）から、
インターネットへの「アクセス」とメールなどの「サービス」を提供する、2000年代前半から稼動を続けるゲートウェイシステム。



数千万ユーザ

ミッションクリティカル

システムの特徴

膨大な業務処理量

- お客様数：約 **数千万** 人
- Webアクセス：約 **数百億** PV/日
- メール送受信：約 **数十億** 通/日
- 通信データ量：約 **数** TBps

膨大な機器数と開発規模

- サーバ台数：**数千** 台
- NW機器台数：**数千** 台
- ストレージ台数：**数百** 台
- 母体規模：**数十** Mstep
- コンポーネント（機能群）：**数百**

膨大な作業量

- アラームメッセージ件数：約 **数千** 件/日
- 商用作業件数：約 **数千** 件/年
- HW故障発生件数：約 **数百** 件/月

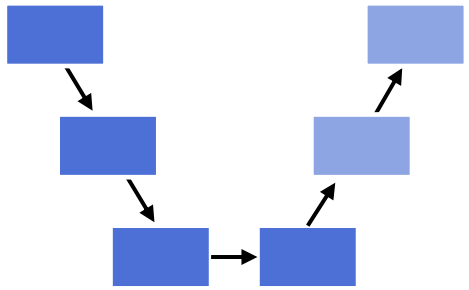
PJ特性

対象システム

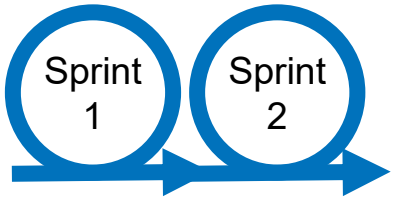
通信基盤システム

開発手法

Waterfall

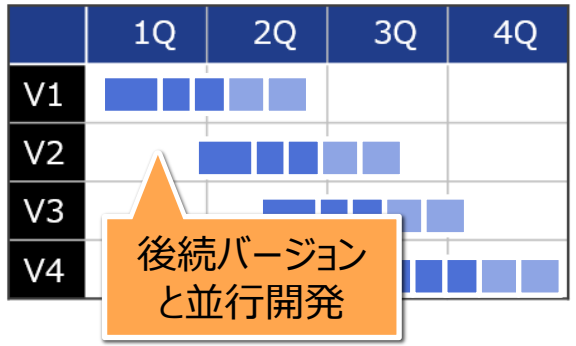


Agile



WF開発周期

年四回以上の開発



WF開発の特徴

- ✓ 1バージョンで複数のサービスを開発
- ✓ 1サービスは複数コンポーネント（機能群）で構成
- ✓ 1コンポーネントは同時に複数バージョン開発

組織構造

マトリックス型（強）

要員数

常時1000人以上

大規模かつ繰返し開発が継続するPJ

03

TDD高品質アジャイルの品質課題

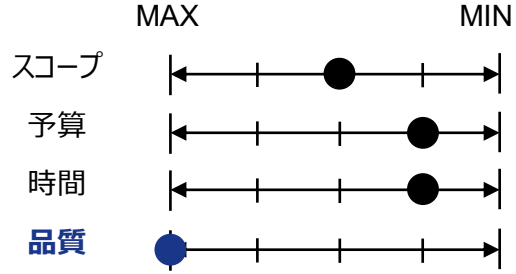
案件概要

開発機能

七千万ユーザを抱える通信基盤システムのアカウント認証基盤

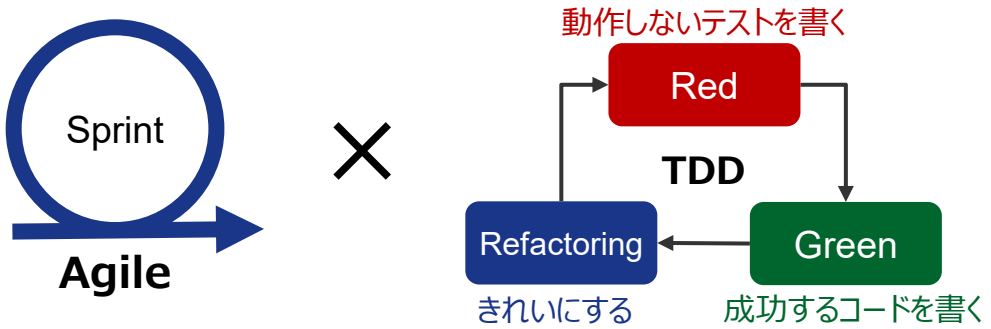
トレードオフスライダー

スコープ、予算、時間よりも品質が最優先



開発プロセスの特徴（詳細は後述）

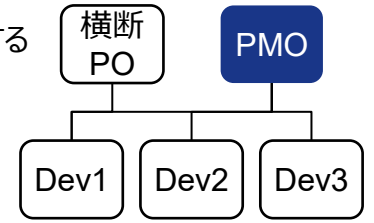
全自動かつバリエーション全網羅のテスト駆動(TDD)アジャイル



開発体制

開発チーム横断のPMOが共通的な管理プロセスを策定

PJ全体のQCDに影響する
プロダクトバックログ管理



- 品質管理プロセス策定
  分析観点/帳票/運用etc
- 品質分析

- 品質データ記録
- 品質対策要否判断および実施

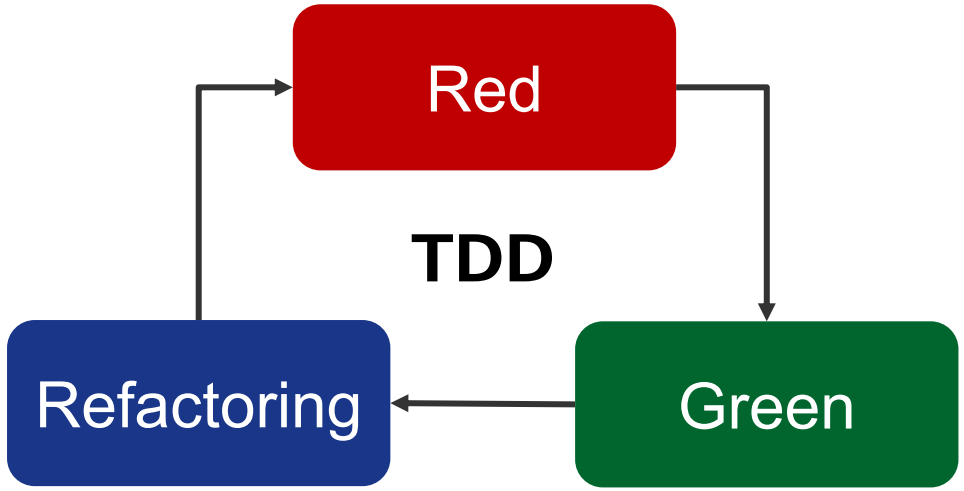
ミッションクリティカルシステムに求められる**高品質**を維持しつつ、**柔軟性、アジリティ、**そして**生産性を高次元で両立**させる開発

テスト駆動開発（TDD）をベースとした品質重視のアジャイル開発（以降、TDD高品質アジャイル）

参考）TDD（テスト駆動開発）について

1.動作しないテストを書く

まず、実装したい機能のテストを書く。
この段階では対象機能が
未実装のためテストは失敗する。



2.成功するコードを書く

次に、テストをパスするために、
必要最小限のコードを実装する。

3.きれいにする

最後に、コードをリファクタリングして、
より効率的で保守しやすい形に改善する。

メリット

- **バグの早期発見と修正**
- （リグレッション）テスト自動化による**品質および生産性向上**
- リファクタリング促進による**保守性向上**

デメリット

- テストコード作成**コスト**
- TDDの**学習コスト**
- テストコードの**メンテナンスコスト**

参考）アジャイル開発へのテスト自動化、TDD適用に関する論文例

例 1

国内外のアジャイル開発事例/論文/インターネットを網羅的に調査し、アジャイル開発に適用できそうなテスト技法・考え方を6件抽出

デグレードを防止し効率的に品質担保するには、
テスト自動化およびTDDが有効。
特別な理由がない限りアジャイル開発で実践すべき

分類	テスト技法･考え方
RECOMMEND ※特別な理由がない限り適用すべき	**テスト自動化、TDD、** 探索的テスト、テスト専任化
OPTIONAL ※可能であれば適用し方が良い	複座づ度測定、非機能バックログ

出典）山本ら(2023). アジャイル開発の品質確保に向けたマネジメントポイント. プロジェクトマネジメント学会春季研究発表大会予稿集, 219-225.

例 2

マトリクス型のテスト設計手法を用いた
E2Eテスト自動化によりアジャイル開発の品質及び
生産性向上を両立した事例

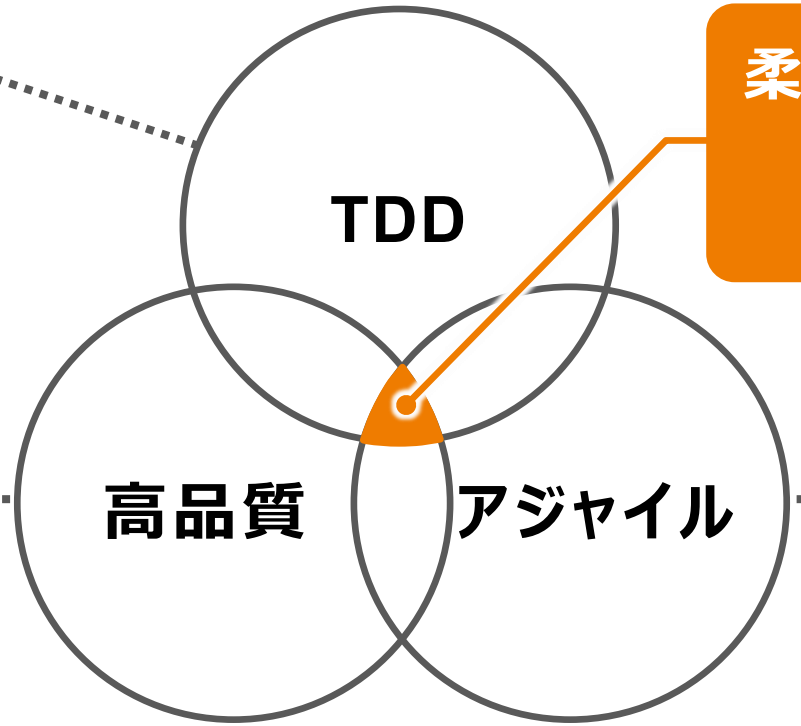
出典）清水歩, 加藤大受 (2022). アジャイル開発における E2E テスト自動化の効率的なテスト設計手法の研究. ソフトウェアエンジニアリングシンポジウム2022論文集, 29-35.

TDD高品質アジャイルの品質課題

アジャイルの品質管理手法は未だ確立されていない中、**ミッションクリティカルシステムの高品質**、**アジャイルの柔軟性/アジリティ**、**TDDの高生産性**を両立するために、**それぞれに求められる品質管理要件や先行事例を矛盾なく統合して、TDD高品質アジャイルに最適化した実践的な品質管理プロセスを構築する必要がある。**

自動試験の高生産性を阻害せず、テスト駆動に最適な品質管理

- 自動試験前提のバグ分析、試験品質評価
- 設計書＜試験項目前提の設計品質評価



柔軟性・アジリティ・高生産性
×
高品質

ミッションクリティカルシステムならではの高品質を実現可能な品質管理

- PJ特性に基づく多角的な定量定性分析
- 過去実績ベースの統計的分析
- ゲート型/モニタリング型の併用
- 鳥の眼/虫の眼/魚の眼

アジャイルの柔軟性、アジリティを阻害しない品質管理

- 小規模・短期の反復開発を前提とした品質評価
- 他PJとの比較の難しさ
- ゲート型/モニタリング型の併用

04

TDD高品質アジャイルの品質管理プロセス

品質管理プロセス構築にあたっての検討方針

基本方針

高度化（高品質）と効率化の両立を目指す

具体的には

① TDD高品質アジャイルの**開発プロセスの特徴をおさえる**

② TDD高品質アジャイルの**柔軟性、アジリティ、高生産性を阻害せず**、同時に高品質を実現するための**品質管理プロセス**を**ゼロベース**で検討

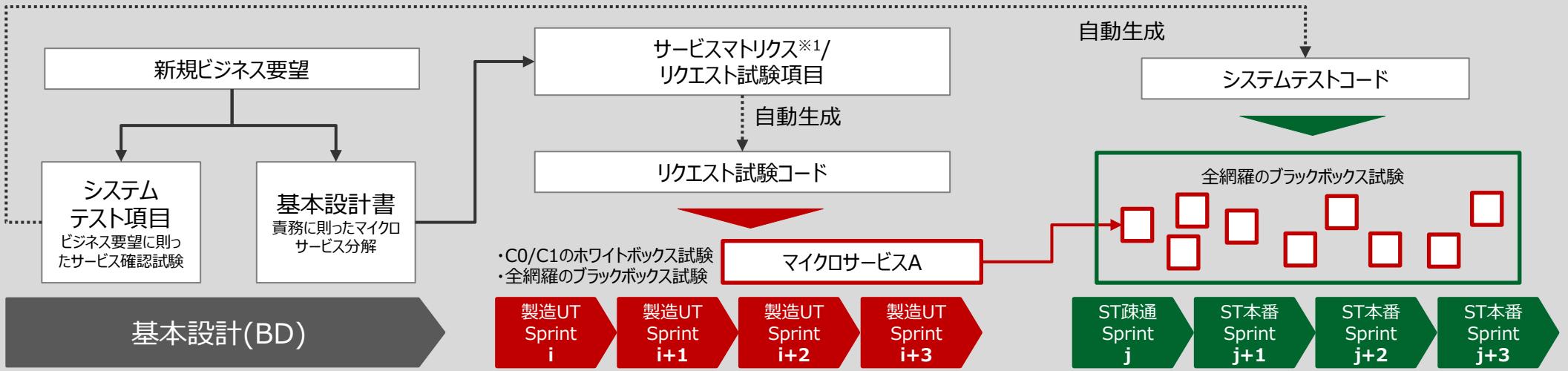
③ **大規模ミッションクリティカルのWFで培った品質管理ノウハウやアジャイルの品質管理に関する先行事例を極力活用**した上で、**合わない部分はテーラリング**する

なお、品質管理プロセスの**外部仕様の構築**（監視対象プロセスの選定・分析観点・分析タイミング・分析帳票・バグ票発行基準etc）を優先し、ツール対応等一定量の対応が必要なものは後回しとする。

TDD高品質アジャイルの開発プロセスの特徴

- 基本設計（以降、BD）でサービスの外部設計、マイクロサービスのアーキテクチャを設計、製造UT Sprintでマイクロサービス設計～単体テストを実施し、最後にST Sprintでマイクロサービス間の結合試験を行う。
- **製造UT Sprint内の試験及びシステムテスト**（以降、ST）はマイクロサービス設計及び外部設計を元に**各IFを全網羅した自動テスト**。
  また、**テストコード**は対応する試験項目/マトリクスから**自動生成**する。

開発プロセス



※1 試験シナリオごとに試験の前提条件および入出力パラメータ値（表示画面/リクエストの要求・応答項目値etc）を整理したマトリクス

特徴

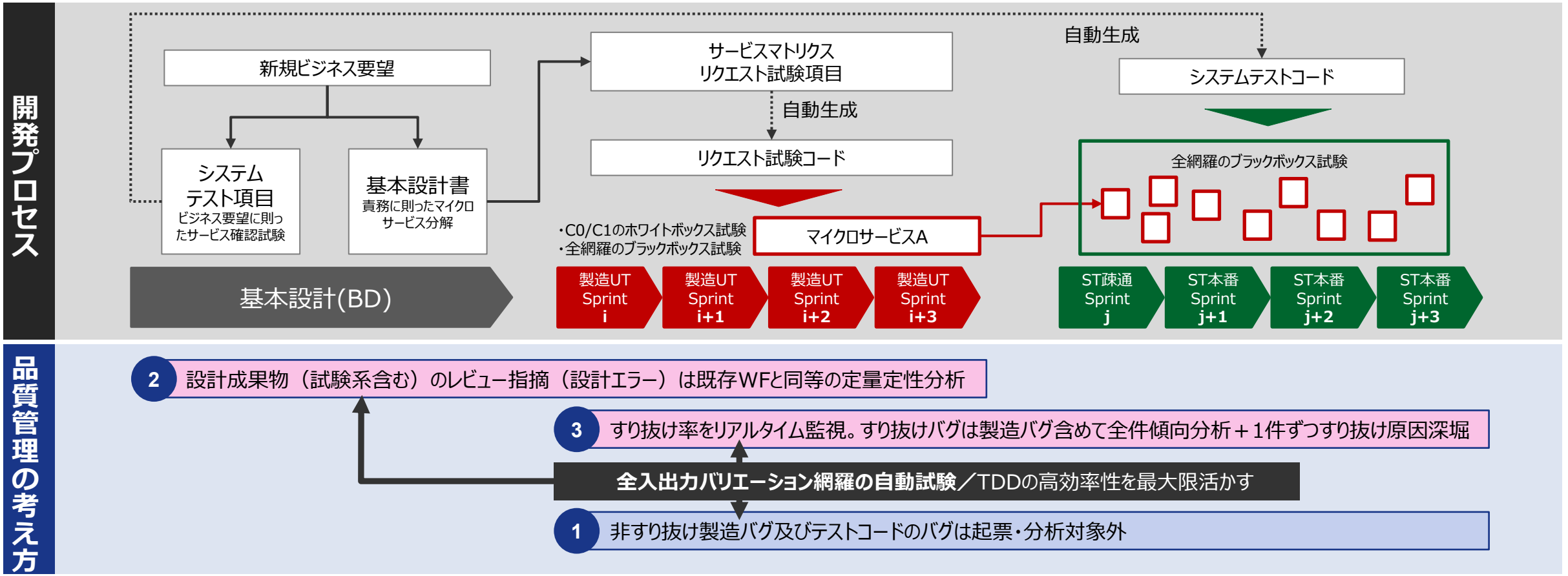
- サービス全体の外部設計、マイクロサービスのアーキテクチャ、**ST項目**作成。
- 開発プロセスは既存WFと同様。

- マイクロサービス単位の設計・製造・試験。
- サービスマトリクス/リクエスト試験項目からテストコード自動生成
- C0/C1のホワイトボックスとマイクロサービスの全入出力バリエーション網羅のブラックボックスの自動試験
- ソースレビューはソースの保守性観点でのみ実施。

- マイクロサービス間の結合試験。
- ST項目からテストコード自動生成
- 基本的にはサービスの全入出力バリエーション網羅の自動試験。オンプレとの接続等環境制約等で自動化が難しい一部の試験のみ手動。

高度化（高品質）と効率化の両立を狙う品質管理プロセスの考え方

1. TDDの高効率性を活かすには自動試験でバグ票分析を行うのは非現実的で、品質観点でも**自動試験で全入出力バリエーションの網羅性が担保される前提**なら製造バグの品質分析の意味合いは薄い※ため**非すり抜け製造バグは品質分析対象外**。また、自動生成される**テストコードも品質分析対象外**とする。
2. この前提を保証するには試験系成果物含む設計品質が既存WF以上に重要。**BDおよび製造UT Sprintの設計成果物（試験系成果物含む）を品質評価対象とし、既存WFの設計工程と同等の定量定性分析**を行う。
3. **すり抜け率をリアルタイムで監視するとともに、製造バグ含めてすり抜けバグは傾向分析＋1件ずつすり抜け原因を深堀**することで、前提の有効性を監視する。



※全入出力バリエーション網羅の自動試験なら、デグレリスクはほぼゼロ。また、テストを全てクリアすれば製造品質は担保されるためバグ分析の必要性は薄い

品質監視対象プロセスと品質評価観点

- BD書やST項目、サービスマトリクス、リクエスト試験項目の**設計エラーは既存WFと同様の定量定性分析**で評価を行う。
- 自動試験の網羅性を前提に、各マイクロサービスの**製造バグについては、すり抜けバグのみを分析対象とし、非すり抜けバグは分析不要**とする。
- 各マイクロサービスを結合した上で実施する**STバグは全件分析対象**とする。特に**すり抜けバグの分析を重視し多角的な観点で分析**を行う。

開発プロセス

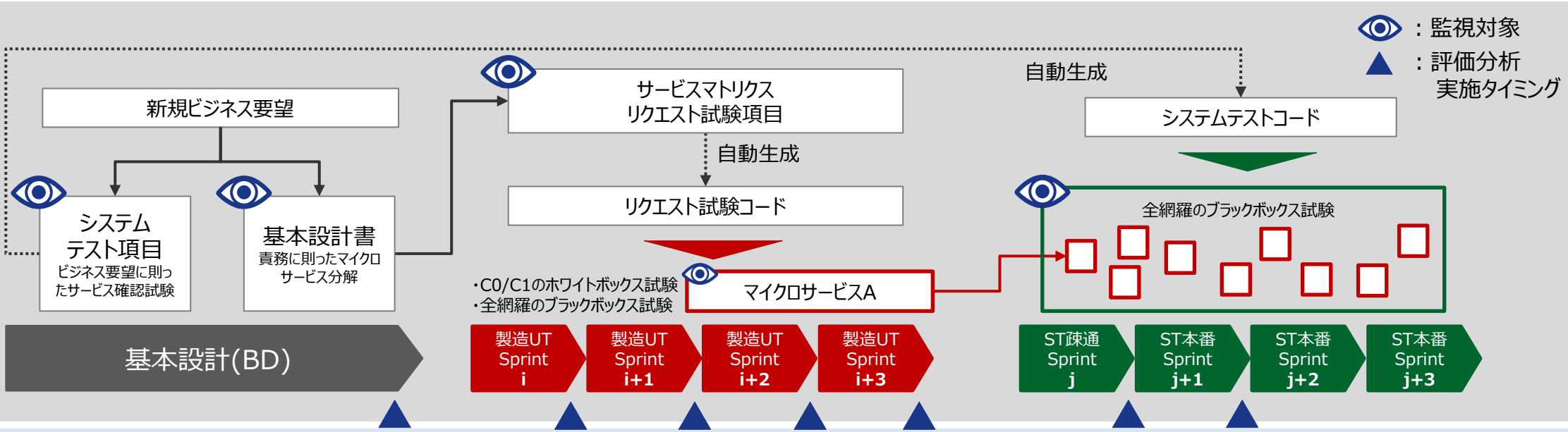
品質評価観点

評価対象と観点	**BD書、ST項目のレビュー指摘（設計エラー）**に対する**頁数ベース**の機能・人・原因別**定量定性分析**	● **サービスマトリクス/リクエスト試験項目**の**レビュー指摘**に対する**SPベース**の機能・人・原因別**定量定性分析** ● 前Sprintからの**すり抜け製造バグ**のすり抜け分析	**すり抜けバグ**に対する混入Sprint/混入機能/すり抜け試験種別/担当者etcの**偏り分析＋1件ずつすり抜け原因深堀**
想定不具合	BD書及びST項目の設計エラー	詳細設計書及び試験系成果物の設計エラー	試験観点・項目・データバリエーション漏れ等、試験設計に関するバグ
対象外	-	自動生成のテストコード、非すり抜けの製造バグ、ソースレビューバグ （保守性観点の改善指摘のみとなるため）	自動生成のテストコード

品質評価タイミングと分析観点

- ゲート型　　：**基本設計、製造UT Sprint、ST Sprint終了タイミングで偏り分析やすり抜け原因分析等の詳細な定量定性分析**を行う
- モニタリング型：**チーム/製造UT Sprint単位の累積エラー・バグ密度及びすり抜け率**を中心に**効率優先の俯瞰的な定量分析**を行う

開発プロセス



品質評価タイミングと分析観点

ゲート

BD書執筆終了時に**設計エラー**の**機能・人・原因別の定量定性分析**を行う

運用：
- レビュー指摘の記入取り纏めは横断PO
- 評価分析報告はPMO
- フィードバックは次回の基本設計執筆時

製造UT Sprint終了時に（or 複数Sprintまとめて）設計エラーの機能・人・原因別の定量定性分析を行う

運用：
- レビュー指摘/バグ票の記入取り纏めは各チームのPO※1
- 評価分析報告はPMO
- フィードバックは次回の製造UT Sprint時

※1 SMは分析フォーマットの作成やPOの支援も実施する

ST Sprint終了時※2**にすり抜けバグの混入Sprint/機能/すり抜け試験種別/担当者etcの偏り分析**及び**すり抜け原因分析**を行う

運用：
- バグ票の記入取り纏めはシステムテストチームのSM
- 評価分析報告はPMOと横断PO
- フィードバックは次回のBD執筆時や製造UT Sprint時

※2 Sprint内で複数実行した場合は初回で検出したエラーのみを対象とする

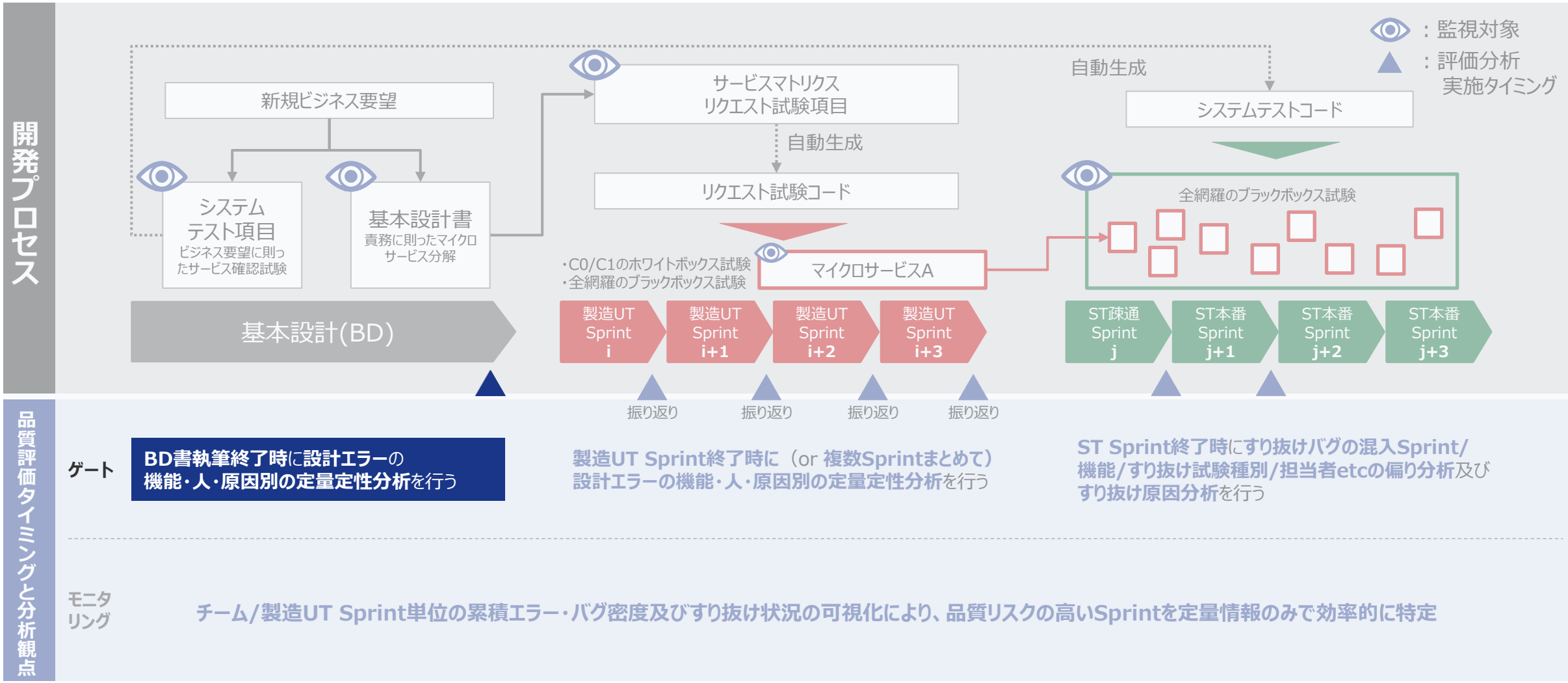
モニタリング

チーム/製造UT Sprint単位の累積エラー・バグ密度及びすり抜け状況の可視化により、品質リスクの高いSprintを定量情報のみで効率的に特定

05

品質分析帳票例と適用結果

ゲート型：基本設計の分析例

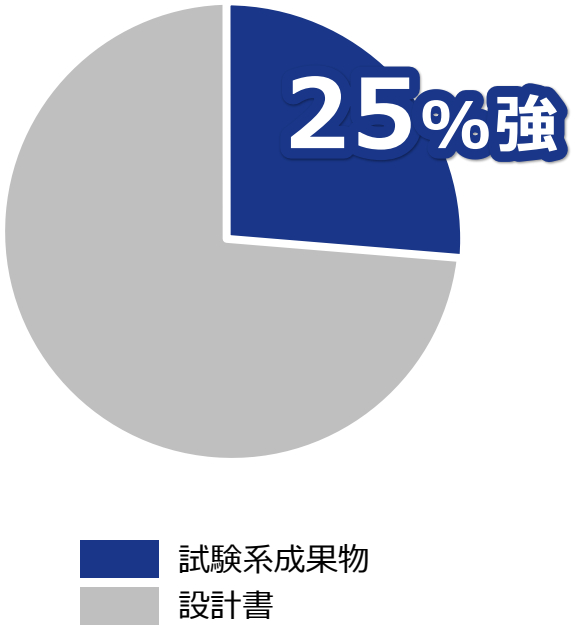


ゲート型：基本設計の分析例

概要　**BD書（ST項目含む）のレビュー結果**に対して、**頁数ベースのエラー密度、レビュー密度で定量定性分析（担当者・機能・原因別等）**を実施

結果　設計書に加えて、**ST項目等の試験系成果物のエラーも摘出（全エラー中25%強）**しており想定通りの結果となった。

設計エラー



定量定性分析

原因

No.	エラー原因
1	要件確認不足
2	既存業務習熟不足
3	既存実装習熟不足
4	設計条件の確認不足
5	技術不足
6	実現方式の検討不足
7	周知連絡の不徹底
8	凡ミス
9	プロセス不備
10	母体設計書不備
	全体

・問題点：要件確認不足（設計も
・原因：指摘を受けるまで、設計も
・対策（横展開及び再発防止）：
・横展開実施結果：類似データ、類
・事例①：要件確認不足(設計も
・事例②：要件確認不足(設計誤

機能

No.	設計担当者
1	機能A
2	機能B
3	機能C
4	機能D
5	機能E
6	機能F
7	機能G
8	機能H
9	機能I
	全体

・問題点：XXに関する基本知識が不足してい
・原因：XXに関する基本知識が不足しており、
・対策（横展開及び再発防止）：XXに関する
・横展開実施結果：類似データ、類似処理を追
・事例①：設計もれ
返却データ漏れ：レスポンス詳細情報に「XX
フロー漏れ：XXの認証を追加、XX送信回数
・事例②：設計誤り

担当者

No.	設計担当者	作成（頁）	エラー摘出率（件/100頁）	エラー摘出率評価	エラー現象									エラー原因									
						エラー																	
						設計もれ	設計誤り	再利用誤り	説明内容不明確	標準違反	設計の改善	重複	非エラー（新規件追加）	非エラー（その他）		要件確認不足	既存業務習熟不足	既存実装習熟不足	設計条件の確認不足	技術不足	実現方式の検討不足	周知連絡の不徹底	凡ミス
1	Y川	38	0.0	▼	0						8			6	0								
2	A山	98	5.1	▼	5	2		2	1		4		2		4	2		2					
3	Y本	98	1.0	▼	1	1					4		1	24	1				1				
4	I股	21	23.8	○	5	1	4				3			8	5	3	2						
5	M田	53	9.4	▼	5		2		3		4	5		8	2	1	1						
6	S村	18	44.4	▲	8		4		4		5			3	4				2		2		
7	K林	3	0.0	▼	0							1		1	1				1				
8	Y根	8	25.0	▲	2	1			1		7			1	0								
9	T井	19	31.6	▲	6	2	4				12	3		3	6	3	3						
10	I崎	1	100.0	▲	1		1				1			1	1			1					
11	F田	16	43.8	▲	7	2	4		1		5	1		2	6	3	1		2				
12	B久	10	50.0	▲	5	2	3				6		5	4	5	2					3		
13	K村	6	66.7	▲	4		4				4		4	3	4	2					1		
14	E口	5	40.0	▲	2		2				3		1		2						2		
	全体	394	-	-	51	11	28	2	10	0	66	10	13	64	41	16	7	3	6	0	8	0	
						22%	55%	4%	20%	0%	-	-		-		39%	17%	7%	15%	0%	20%	0%	

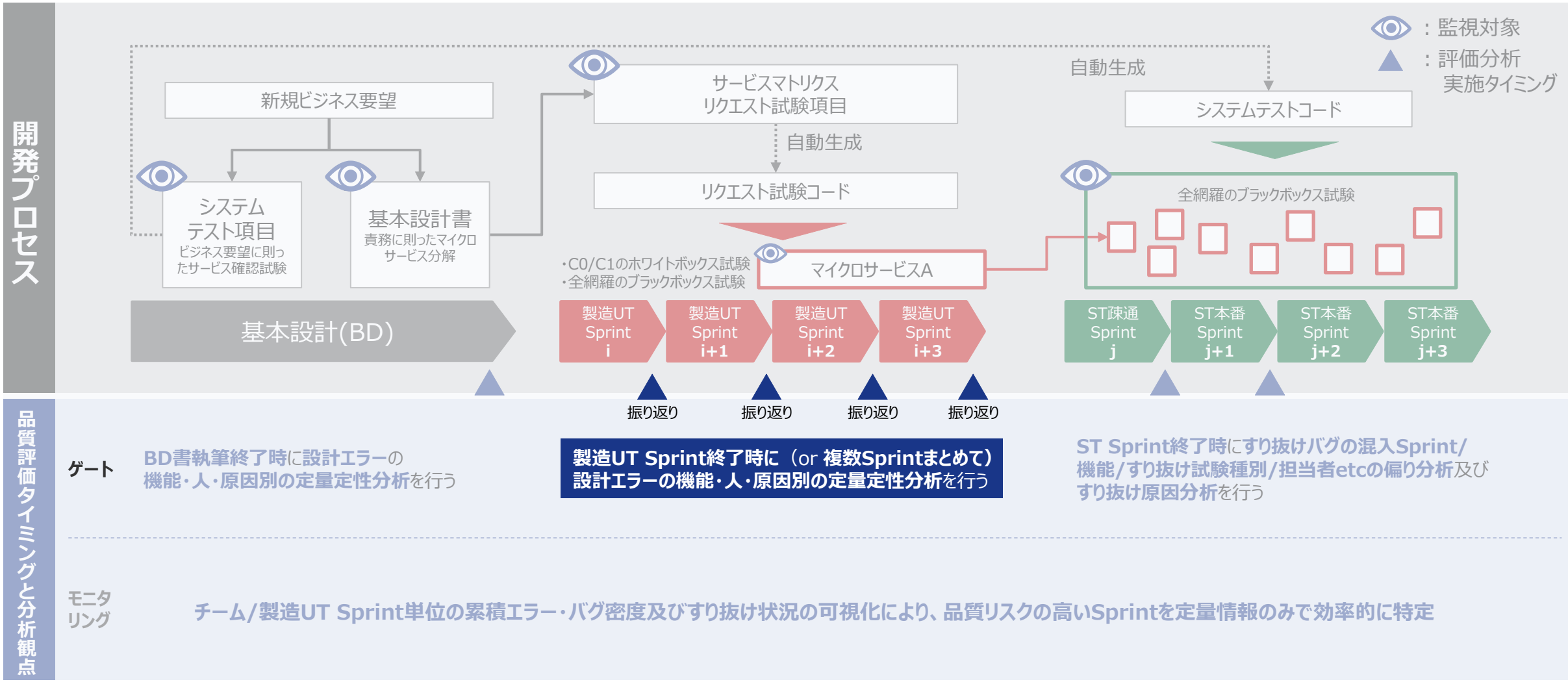
※原因は複数選択可のため、原因合計件数とバグ

・問題点：XXに関する基本知識が不足している担当(S村、Y根、T井、I崎、F田、B、K村、E口)のエラー摘出率が高い。
・原因：XXに関する基本知識が不足しており、指摘を受けるまで、設計もれ(返却データ漏れ、フロー漏れ)、設計誤り（不要な返却データ、誤った返却データ、誤った処理分岐）に気付くことが出来なかった。
・対策（横展開及び再発防止）：XXに関する知識が不足している担当者を含め、横展開は記録を取り漏れなく実施した。
・横展開実施結果：類似データ、類似処理を追加/修正した。
・事例①：設計もれ
返却データ漏れ：レスポンス詳細情報に「XX識別子」を追加
フロー漏れ：XXの認証を追加、XX送信回数管理登録およびXX送信回数管理更新の処理を追加
・事例②：設計誤り
不要な返却データ：送信メールアドレスが返却できるのはSMS/メールでセキュリティコードを送信した場合のみのため、アプリに通知送った場合に「送信メールアドレス(伏字変換)」の返却は不要

…

※説明上の都合で、数値は実績値を一部修正

ゲート型：製造UT Sprintの分析例



全量把握

- システム（機能）の全量
- **チームごとにインフレ・デフレしない**
- WFでよく使われる

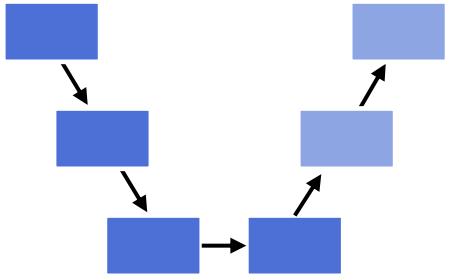
- **PJ（機能＋非機能）の全量**
- チームごとにインフレ・デフレする
- **アジャイルでよく使われる**

SPの基準をチーム間で統一した上で、全量把握のメトリクスとして**SPを採用**

※初期のSprintはプログラムコード行数も併用

参考）中本傑ら(2022). エンタープライズ向けアジャイル開発におけるスコープマネジメントの考察. プロジェクトマネジメント学会秋季研究発表大会予稿集, 340-350.

品質指標値の考え方

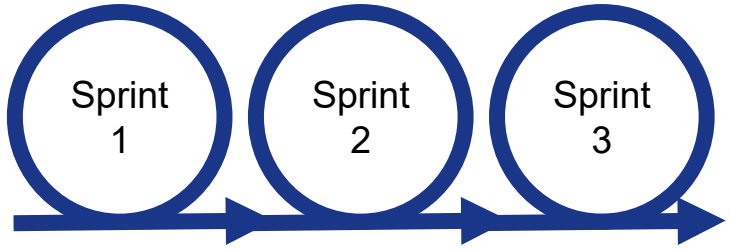


WF

- **限定された技術**
- **標準化されたプロセス**
- 反復なし

豊富な類似PJ

類似PJの
過去実績ベースの指標値



Agile

- 多様な技術
- 多様なプロセス
- **反復あり** ➡ **同一PJ内Sprint間では体制/プロセス/利用技術に強い類似性**

同一PJ内Sprintの
実績推移ベースの指標値

※初期のSprintはWFの指標値や相対評価を併用

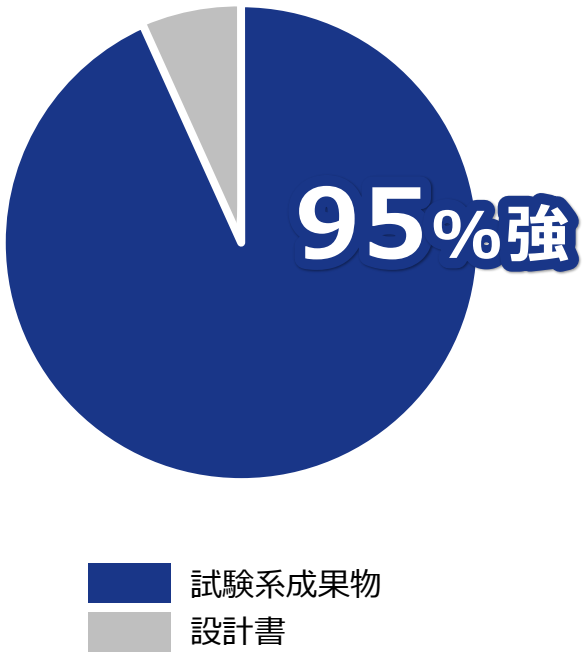
参考）射場千尋(2023). Sprint ごとの不具合状況推移に着目した品質管理手法の適用事例. プロジェクトマネジメント学会春季研究発表大会予稿集, 177-190.

ゲート型：製造UT Sprintの分析例 1/2 Sprint単位

概要 **サービスマトリクス/リクエスト試験項目のレビュー指摘**に対して**SPベースのエラー密度で担当者・機能・原因別の定量定性分析**を行った。

結果 **大半を占める試験系成果物のエラーに対しても、定量定性分析は実態に即した偏りを示しており、分析の有効性を確認できた。**

設計エラー



定量定性分析

機能

設計種別	実規模（Ks）	ストーリーポイント（Sp）
	1.62	64
試験データ	1.62	82
試験項目	1.62	78

サービスマトリクスは、試験項目（マトリクス表の列）は観点(列)の不足/誤り、試験データは

データで14件（設計もれ：2件

れは、異常応答のケースが不足

が自分の作業を終わらせること

■横展開：横展開(サービスシーケンスとサービス

■対策：P.Oから指摘内容集計結果をチーム内

※事例

・設計もれ：「異常応答(XX情報取得APIエラー

・設計誤り①：「XX認証API」に渡す「XX識別子

・設計誤り②：タイムアウトのエラー理由が"0002

試験系成果物

担当者 SPベース

担当者	実規模（Ks）	ストーリーポイント（Sp）	エラー数	エラー密度（件／Ks）	コピー密度（件／Sp）	エラー密度評価	エラー現象								エラー原因									
							設計もれ	設計誤り	再利用誤り	説明内容不明確	標準違反	設計の改善	重複	非エラー（新規要件追加）	非エラー（その他）	要件確認不足	既存業務確認不足	既存実装確認不足	設計条件の確認不足	技術不足	実現方式の検討不足	凡ミス	周知連絡の不徹底	プロセス不備	母体設計書不備	セルフチェック漏れ
A	0.	17	10	33.33	0.	▲	1	9	0	0	0	6	0	0	0	0	0	0	1	0	0	9	0	0	0	0
B	0.	9	1	1.67	0.	○	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
C	0.3	20	4	11.43	0.	▲	3	1	0	0	0	3	0	0	1	0	0	0	3	0	0	1	0	0	0	0
D	0.3	18	3	8.11	0.	▲	1	2	0	0	0	2	0	0	0	0	0	0	1	0	0	0	0	0	0	0

■問題点：Aがエラーを10件摘出した。

■原因：設計誤りは凡ミスとコピー元の修正ミスとなっているが、Sprit1での遅延を取り戻すために、チーム全員が時間に追われることになり、各担当が自分の業務経験の浅いメンバへのサポートが出来なかった。

■横展開：横展開(サービスシーケンスとサービスマトリクスの対比チェック)を実施し、問題ないことを確認した。

■対策：P.Oから指摘内容集計結果をチーム内で共有した。Sprint3の対策は、改善各担当の強化ポイントを明確にして担当内クロスチェックの確認ポ

①BD書を読み解けている担当：Bが、読み解けてない担当：Aのクロスチェックを行う

②各担当がSprint2の指摘内容を再確認し、Sprint3で同じ指摘を受けない様にセルフチェック(BDとDDの対比チェック)を行う

※担当者別傾向（P.Oヒアリング結果）

・A、C、DがSprint2で初めてサービスマトリクスを完成した。

・Aは業務経験が浅く、今回の指摘を受けたBD書（サービスシーケンス）の読み解き方への指示をSprint3で実践することで改善を図る。

・C、Dは、Sprit1で完成しなかったUT2試験実施などのSprint1→Sprint2へ持ち越したタスク作業に予測以上に時間をとったことで、時間に追われることになり、BD書確認が不十分となった。

・BはBD書設計から関わりがある経験者のため、エラー数は少ない。

■まとめ：BD工程に関わりがある担当のエラー数は少なく、DD工程から参画した担当はBD書(サービスシーケンス/IF仕様書)の読み解きに課題がある。

PMOとP.O間で協議し、Sprint2～4までに課題抽出/施策実行を繰り返し、Sprint5以降に成果を出す方針とした。

担当者特性に即したエラー偏り

…

※説明上の都合で、数値は実績値を一部修正

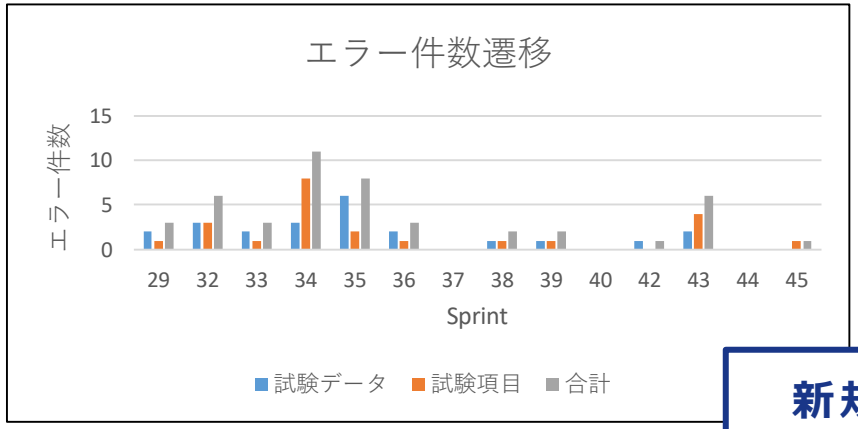
ゲート型：製造UT Sprintの分析例 2/2 Sprint横断

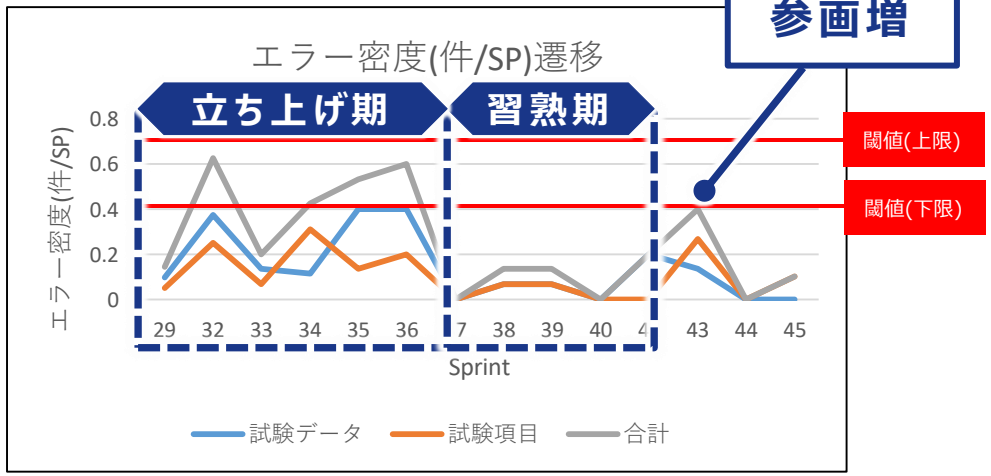
概要

アジャイル＝小規模開発の反復→**SPの基準をチーム間で統一**した上で、**SPベースのエラー密度でSprint横断の定量定性分析**を行った。

結果

SPベースのエラー密度でも定量定性分析（特に時系列推移）は実態に即した傾向を示しており、分析の有効性を確認できた。





定量定性分析

担当者

担当者名	ストーリーポイント（Sp）	エラー数	エラー密度（件／Sp）	エラー密度評価
M永	3	2	0.6667	○
T井	5	0	0	▼
K村	13	1	0.0769	▼
T井	8	5	0.625	○
T井	5	1	0.2	▼
F永	5	1	0.2	▼
K村	5	1	0.2	▼
N川	5	2	0.4	▼
A木	8	4	0.5	○
F永	13	5	0.3846	▼
A木	5	3	0.6	○
T井	5	2	0.4	▼
K村	5	3	0.6	○
F永	5	3	0.6	○
I股	5	0	0	▼
G田	10	0	0	▼

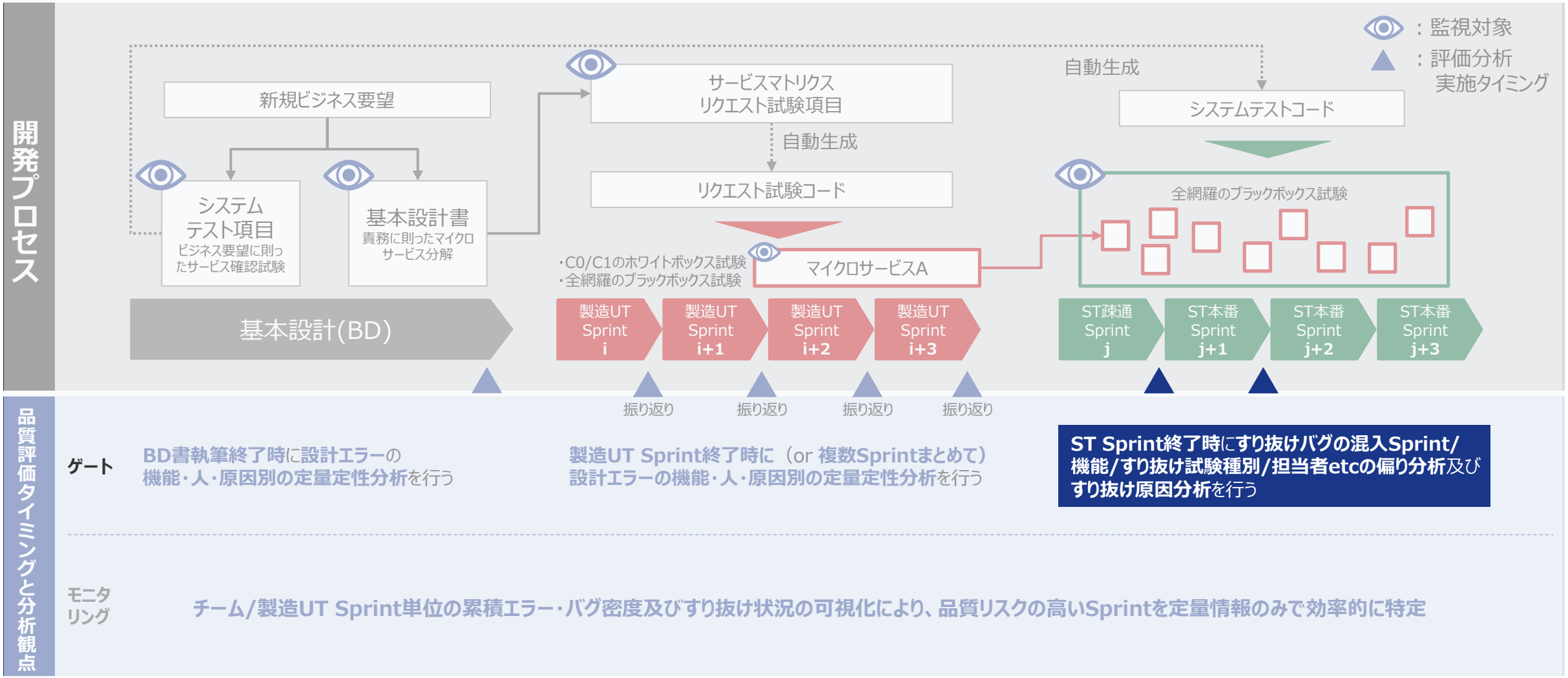
Sprint

設計種別 試験データ：サービスマトリクスの行 試験項目：サービスマトリクスの列	ストーリーポイント（Sp）	エラー数	エラー密度（件／Sp）	エラー密度評価	エラー現象					
					設計もれ	設計誤り	再利用誤り	説明内容不明確	標準違反	設計の改善
試験データ(Sprint29)	21	2	0.0952	▼	0	2	0	0	0	0
試験項目(Sprint29)	21	1	0.0476	▼	1	0	0	0	0	
試験データ(Sprint32)	8	3	0.375	▼	0	2	0	0	1	0
試験項目(Sprint32)	8	2	0.25	▼	0	2	0	0	0	0
試験データ(Sprint33)	15	2	0.1333	▼	0	2	0	0	0	
試験項目(Sprint33)	15	1	0.0667	▼	0	0	0	1	0	0
試験データ(Sprint34)	26	3	0.1154	▼	1	2	0	0	0	0
試験項目(Sprint34)	26	8	0.3077	▼	4	2	0	0	2	0
試験データ(Sprint35)	15	6	0.4	▼	0	5	0	1	0	1
試験項目(Sprint35)	15	2	0.1333	▼	0	1	0	0	1	
試験データ(Sprint36)	5	2	0.4	▼	0	2	0	0	0	1
試験項目(Sprint36)	5	1	0.2	▼	1	0	0	0	0	0
試験データ(Sprint37)	5	0	0	▼	0	0	0	0	0	
試験項目(Sprint37)	5	0	0	▼	0	0	0	0	0	0
試験データ(Sprint38)	15	1	0.0667	▼	0	1	0	0	0	0
試験項目(Sprint38)	15	1	0.0667	▼	0	1	0	0	0	0
試験データ(Sprint39)	15	1	0.0667	▼	0	1	0	0	0	0
試験項目(Sprint39)	15	1	0.0667	▼	0	1	0	0	0	
試験データ(Sprint40)	5	0	0	▼	0	0	0	0	0	

…

※説明上の都合で、数値は実績値を一部修正

ゲート型：STの分析例



ゲート型：STの分析例 1/2

概要　**ST検出のすり抜けバグ**に対して、混入Sprint/混入機能/すり抜け試験種別/担当者etcの**多角的な偏り分析**を行った。

結果　**すり抜けバグは試験設計ミス（試験観点・項目・データバリエーション漏れ等）に偏り**があり**想定通りの結果**となった。

チーム/Sprint別すり抜けバグ

		案件A			案件B									
チーム	Sprint	10	11	12	13	14	15	16	17	18	19	20	21	合計
A			1						1	1				3
B					2	1	3				3	2	3	14
C		1								2				3

Bチームのすり抜けバグ傾向

		案件A			案件B									
	Sprint	10	11	12	13	14	15	16	17	18	19	20	21	合計
①混入Sprint	すり抜け(混入Sprint)				2	1	3				3	2	3	14
②試験種別	S3(E2E)				2	1	2				3	2	2	12
	S3(手動)						1						1	2
③機能別	機能A				1						1		1	3
	機能B				1	1								2
	機能C						1						1	2
	機能D						2							2
	機能E											1	1	2
	機能F										1			1
	機能G										1			1
④すり抜け試験別	試験観点漏れ（UT2）						1						1	**2**
	結果確認ミス（UT1）						1							1
	結果確認ミス（UT2）				1	1					2		1	**5**
	テスト項目抽出漏れ（UT2）						1				1	1		**3**
	データバリエーション不足（UT2）				1							1	1	**3**
⑤担当別	H野										1		1	2
	O高						1					1		2
	K村				1						1			2
	A村					1								1

※説明上の都合で、数値は実績値を一部修正

すり抜けバグは製造UT Sprint内の

- **試験設計ミス**
- 結果確認ミス※

※製造UT Sprint内試験では画面遷移等、環境制約で目視確認になる試験が一部あり、その確認ミス。
STでは画面遷移で必ず検出できる。

ゲート型：STの分析例 2/2

概要 特に製造UT Sprintからのすり抜けバグについては**全件すり抜け原因分析**を行った。

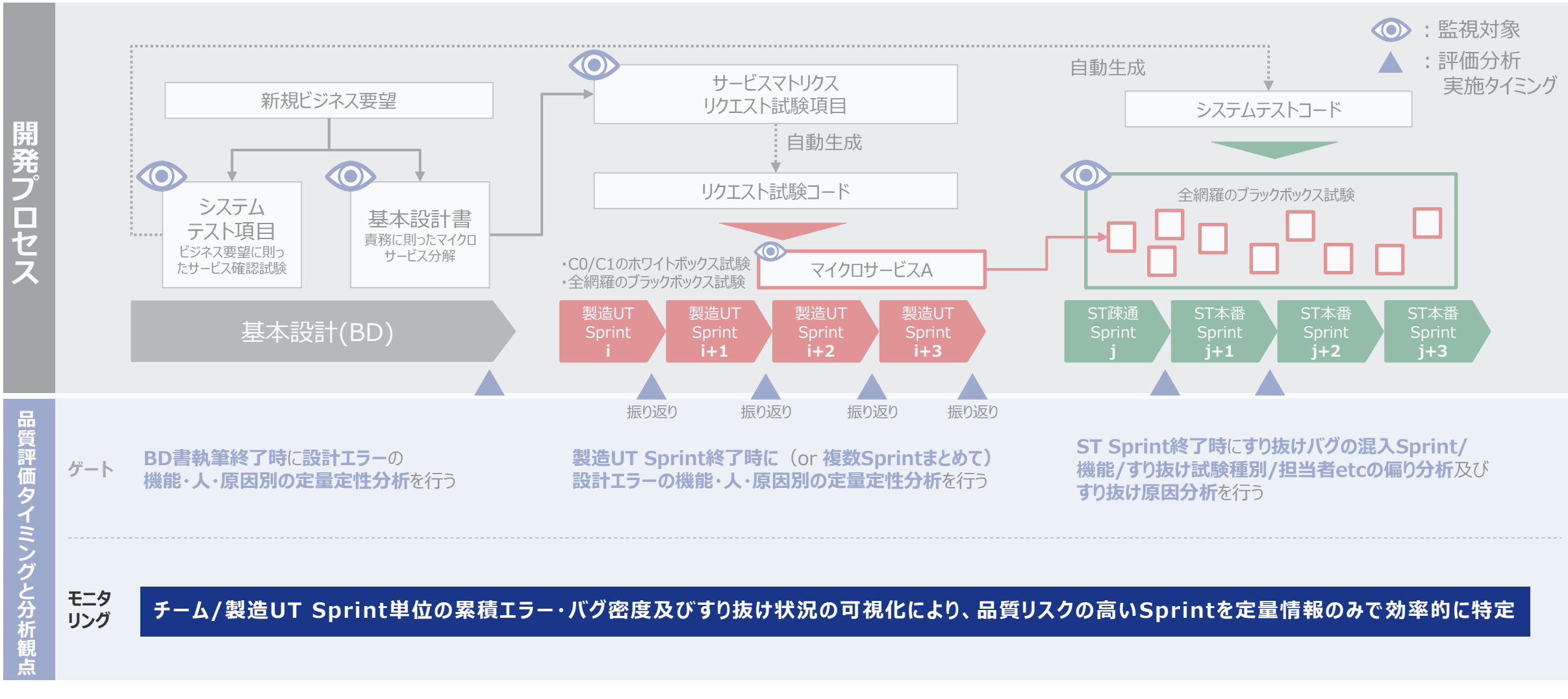
結果 **成果物ベースでどんな問題があったのかを適切に把握し、具体的な品質アクションに繋げる**ことができた。

すり抜けバグ分析例

傾向分析用のメタ情報　　**すり抜け原因を1件ずつ深堀**

故障事象			混入分析			すり抜け分析（概要）				すり抜け分析（詳細）			
概要	直接原因	故障機能	混入工程	混入Sprint	混入原因	要摘Sprint	要摘テスト	レビューすり抜け原因	試験すり抜け原因	大分類	要因分析	すり抜けバグに対する品質強化施策	今後の施策（恒久対処）
XX APIからXXリスト参照時に400応答	XXX	機能A	製造UT Sprint (製造)	19	要件確認不足	19	製造UT Sprint UT2	レビュー時の根拠ドキュメント漏れ	テスト項目抽出漏れ	サービスマトリクスで誤った定義をした（空文字）	空文字の解釈が曖昧であったため、担当者/レビューア間で意識齟齬が生じた	サービスマトリクスで空文字のデータが入るルートについて、ログアサーションを追加し、値チェックを行う	• デフォルト値をundefined、デフォルト型はstring/undefinedにする • 共通でSET/GETを作る

モニタリング型：Sprint横通しの俯瞰分析例

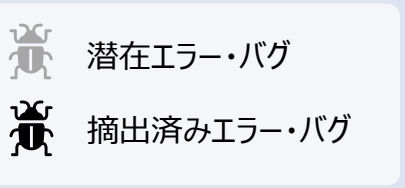


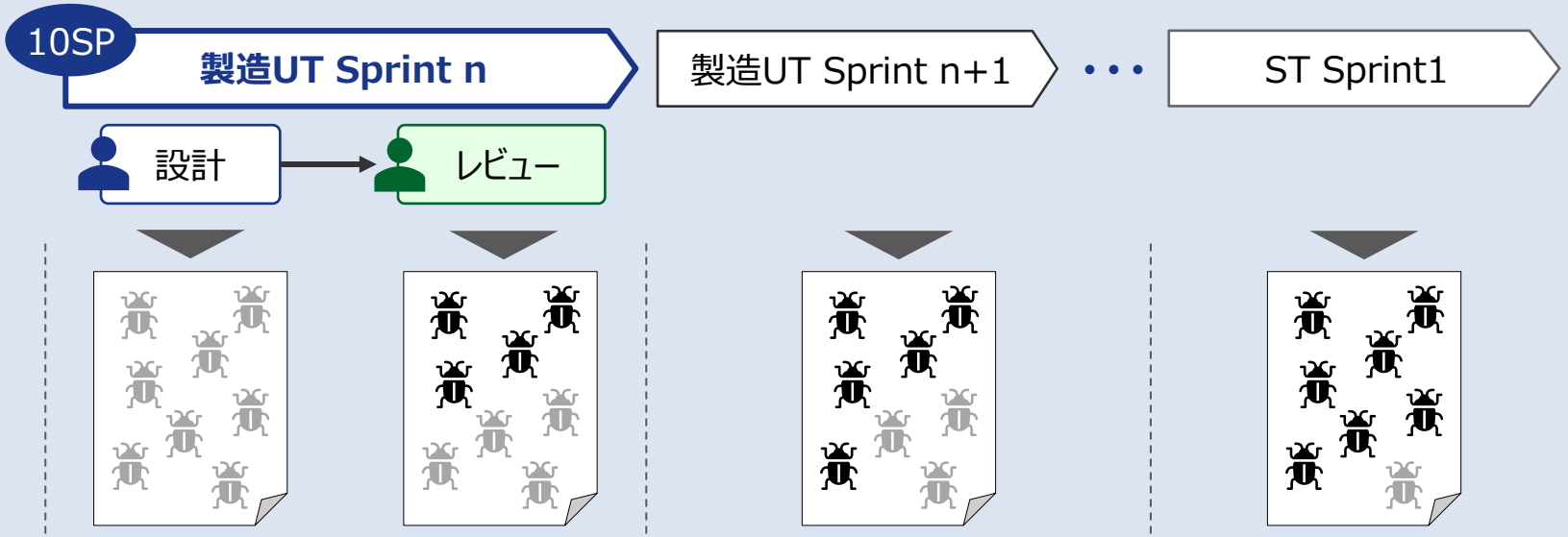
モニタリング指標

各製造UT Sprint単位の設計（作り込み）品質、レビュー品質をそれぞれ累積エラー・バグ密度、すり抜け率で可視化

製造UT Sprint nの設計エラー摘出イメージ

※TDD高品質アジャイルでは各製造UT Sprintの
製造バグはすり抜けを除いて分析対象外としている。
従って、「製造UT Sprint n作り込みかつ
製造UT Sprint n摘出のエラー・バグ≒製造UT
Sprint nの設計レビュー指摘」となる





			すり抜けバグ	
	エラー・バグ数	4	1	2
	累積エラー・バグ数	4	5	7
設計品質	**累積エラー・バグ密度** [件/SP]	0.4	0.5	0.7
レビュー品質	**すり抜け率** すり抜けバグ数÷累積エラー・バグ数	—	20%	43%

モニタリング型：Sprint横通しの俯瞰分析例

概要　開発チームごとに、**製造UT Sprint単位のエラー・バグ推移、累積エラー・バグ密度、すり抜け率**を可視化

結果　**作り込み or レビュー品質が悪いSprintを定量情報のみで効率的に特定**できた。一方、**各Sprintの開発機能が独立している場合、製造UT Sprint内ですり抜けバグは出にくいため、タイムリーな品質問題の検知という点では課題は残る**。

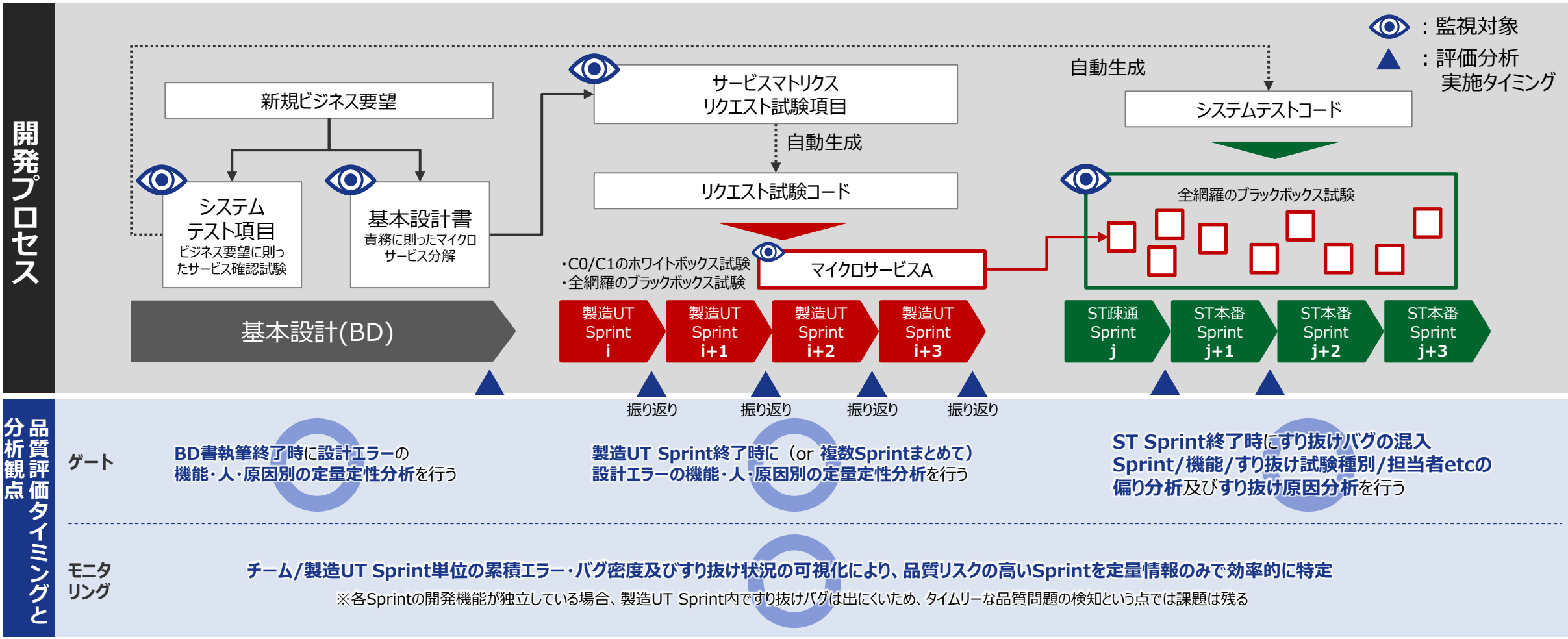
	設計（作り込み）品質			レビュー品質			
	累積エラーバグ数	累積エラーバグ密度	定量評価	累積エラーバグ数内訳		すり抜け率	定量評価
				作り込みSprint	すり抜け		
BD	53	0.70	▼	51	2	3.8%	○
Sprint1	25	5.97	▲	18	7	28.0%	▲
Sprint2	13	2.01	○	11	2	15.4%	○
Sprint3	67	15.69	▲	66	1	1.5%	○
Sprint4	5	1.71	○	5	0	0.0%	○
Sprint5	9	2.74	○	6	3	33.3%	▲
Sprint6	3	2.38	○	3	0	0.0%	○

Sprint1：設計品質、レビュー品質ともに×　→　Sprint終了時品質×だったので当時の対策の妥当性検証
Sprint3：設計品質×だがレビュー品質○　→　収束傾向あり
Sprint5：設計品質○だがすり抜けが多い　→　試験系成果物のレビュー品質に問題ありの可能性
Sprint7：設計品質、レビュー品質ともに×　→　Sprint終了時品質○だったので作り込み、レビューともに要再点検

※説明上の都合で、数値は実績値を一部修正

TDD高品質アジャイルの品質管理プロセスの結果まとめ

- 非すり抜け製造バグを分析対象外としたことで**TDDの効率性を損なわない品質管理**になった。**歯止めとしてすり抜け製造バグは分析対象としていたが、結果的に該当バグは検出されておらず、品質観点でも問題ない結果**となった。
- 想定通り**一定量の試験設計エラー摘出**があった。また、**SPベースのエラー定量定性分析は実態を可視化**できており、有効性を確認できた。
- **ST検出のすり抜けバグ/商用バグはほぼ試験設計ミスで目論見通り。**また、**適切な品質対策のためのすり抜け原因分析の重要性**も確認できた。



06

結果に対する評価

商用品質

商用リリース後6カ月で
商用AP品質は既存ミッションクリティカルシステムのWF同等かそれ以上で安定

商用APバグ密度

0.01件/ks

※ミッションクリティカルシステムWFの目標値：0.03件/ks

商用APバグ内容

試験設計ミス数件で製造バグはゼロ件

- サービスマトリクスの試験パターン抽出漏れ
- 試験データ誤り

試験設計を重点監視対象とした品質管理プロセスの妥当性を示す結果

結果に対する評価

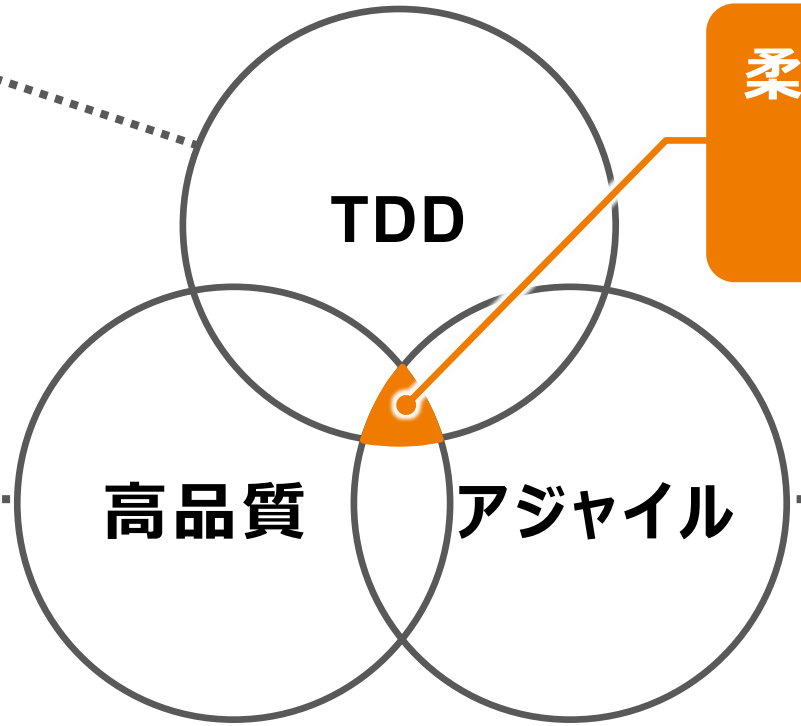
アジャイルの柔軟性/アジリティ、TDDの高生産性を損なわずに、試験設計に関する故障を早期に摘出し**既存ミッションクリティカルシステムのWF開発以上の商用AP品質を実現**した。

自動試験の高生産性を阻害せず、テスト駆動に最適な品質管理

- **試験設計およびすり抜けバグを重点監視**する一方**製造バグは管理対象外**とする**メリハリ**のある品質管理方針

ミッションクリティカルシステムならではの高品質を実現可能な品質管理

- **商用AP品質は既存ミッションクリティカルシステムのWF開発以上**
- 商用APバグは試験設計ミス数件で製造バグはゼロ件



柔軟性・アジリティ・高生産性
×
高品質

アジャイルの柔軟性、アジリティを阻害しない品質管理

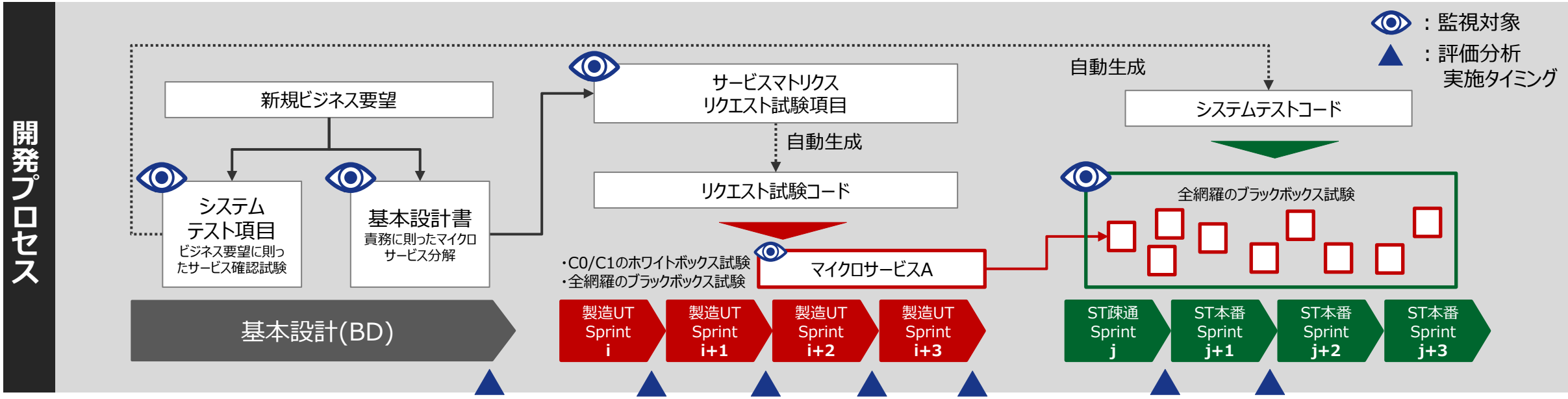
- **ゲート型、モニタリング型を併用**した**多角的な定量分析**手法を具備した品質管理プロセス
- **すり抜けバグの重点監視**

07

おわりに

TDD高品質アジャイルの品質管理プロセスサマリ

- TDDの高効率性を活かすには自動試験でバグ票分析を行うのは非現実的で、品質観点でも**自動試験で全入出力バリエーションの網羅性が担保される前提**なら製造バグの品質分析の意味合いは薄いため**非すり抜け製造バグは品質分析対象外**。また、自動生成される**テストコードも品質分析対象外**とする。
- この前提を保証するには試験系成果物含む設計品質が既存WF以上に重要。**BDおよび製造UT Sprintの設計成果物（試験系成果物含む）を品質評価対象とし、既存WFの設計工程と同等の定量定性分析**を行う。
- **すり抜け率をリアルタイムで監視するとともに、製造バグ含めてすり抜けバグは傾向分析＋1件ずつすり抜け原因を深堀**することで、前提の有効性を監視する。



品質評価タイミングと分析観点		
ゲート	**BD書執筆終了時に設計エラーの機能・人・原因別の定量定性分析**を行う	**製造UT Sprint終了時に（or 複数Sprintまとめて）設計エラーの機能・人・原因別の定量定性分析**を行う	**ST Sprint終了時にすり抜けバグの混入Sprint/機能/すり抜け試験種別/担当者etcの偏り分析**及び**すり抜け原因分析**を行う
モニタリング	**チーム/製造UT Sprint単位の累積エラー・バグ密度及びすり抜け状況の可視化により、品質リスクの高いSprintを定量情報のみで効率的に特定**		

考察

品質管理プロセスを構築する上で最も基本的かつ重要な考えについて一段抽象度を上げて考察した

1. 品質リスクの見極め

開発初期段階でリスク評価を行い、潜在的な問題点を洗い出す。
本活動での例）TDD高品質アジャイルの開発プロセスの特徴を踏まえて試験設計の品質が最重要かつ高リスクであると判断した。
一方、自動生成されるテストコードや自動試験でバグ検出可能な製造品質のリスクは低いと判断した。

2. リスクに応じたメリハリ

品質管理のリソースは限られているため、リスクが高いところは重点監視、低いところはほどほどに監視するメリハリをつけるのが重要。
本活動での例）試験設計やすり抜けバグを重点監視する一方、非すり抜け製造バグや自動生成のテストコードは品質分析対象外とすることで、効率的な品質管理を実現した。

3. 効率的かつ高度な品質評価の仕組み

アジリティ/高生産性/高品質を高次元で両立するためには効率的かつ高度な品質評価の仕組みが求められる。
本活動での例）小規模かつイテレーティブな開発スタイルを踏まえたゲート型、モニタリング型評価の併用やグラフやマトリクスを多用した定量評価主体の分析、SPベースのSprint横断の時系列評価などにより、効率的に品質問題の早期検知と対策が可能となった。

WFでもアジャイルでも変わらない品質管理プロセス構築の勘所

「高度化と効率化の両立」を徹底追及

高品質（高度化）

低品質

低コスト

高コスト

ターゲット

必要なことを効率的に実施

- 品質リスクに応じたメリハリ
- 開発効率を阻害しない最低限の基礎情報
- 品質状況を効率的に可視化
  - PJ特性を踏まえた多角的な分析観点
  - 数字、マトリクス ＞ 文章
  - 機械化、標準化の徹底
- 再発防止管理による自律的な品質改善
- 品管G要員数少ない＆PJ規模によらず一定

必要なことはやれてるけど非効率

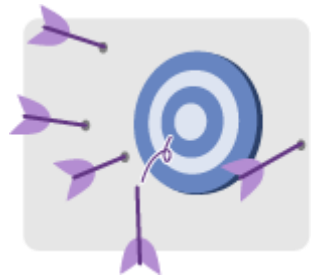
- 品質リスクによらず全力投球
- 冗長な基礎情報/内容チェックで稼働逼迫
- 稼働をかけて品質状況を可視化
  - PJ特性を踏まえた多角的な分析観点
  - 数字、マトリクス ＜ 文章
  - 手作業が多い、属人性が強い
- 再発防止管理による自律的な品質改善
- 品管G要員数多い＆PJ規模に比例して増える

何もやってない

- 品質保証ストーリーなし
- 分析の基礎情報不足
- 品質状況不明（分析観点少なすぎ）
- 同じトラブルが頻発
- 品管G要員数少ない＆PJ規模によらず一定

やってることが的外れ

- リスク観点が的外れ
- 冗長な基礎情報/内容チェックで稼働逼迫
- 稼働かけてるのに品質状況不明
  - 分析観点が的外れ
  - 数字、マトリクス ＜ 文章
  - 手作業が多い、属人性が強い
- 同じトラブルが頻発
- 品管G要員数多い＆PJ規模に比例して増える

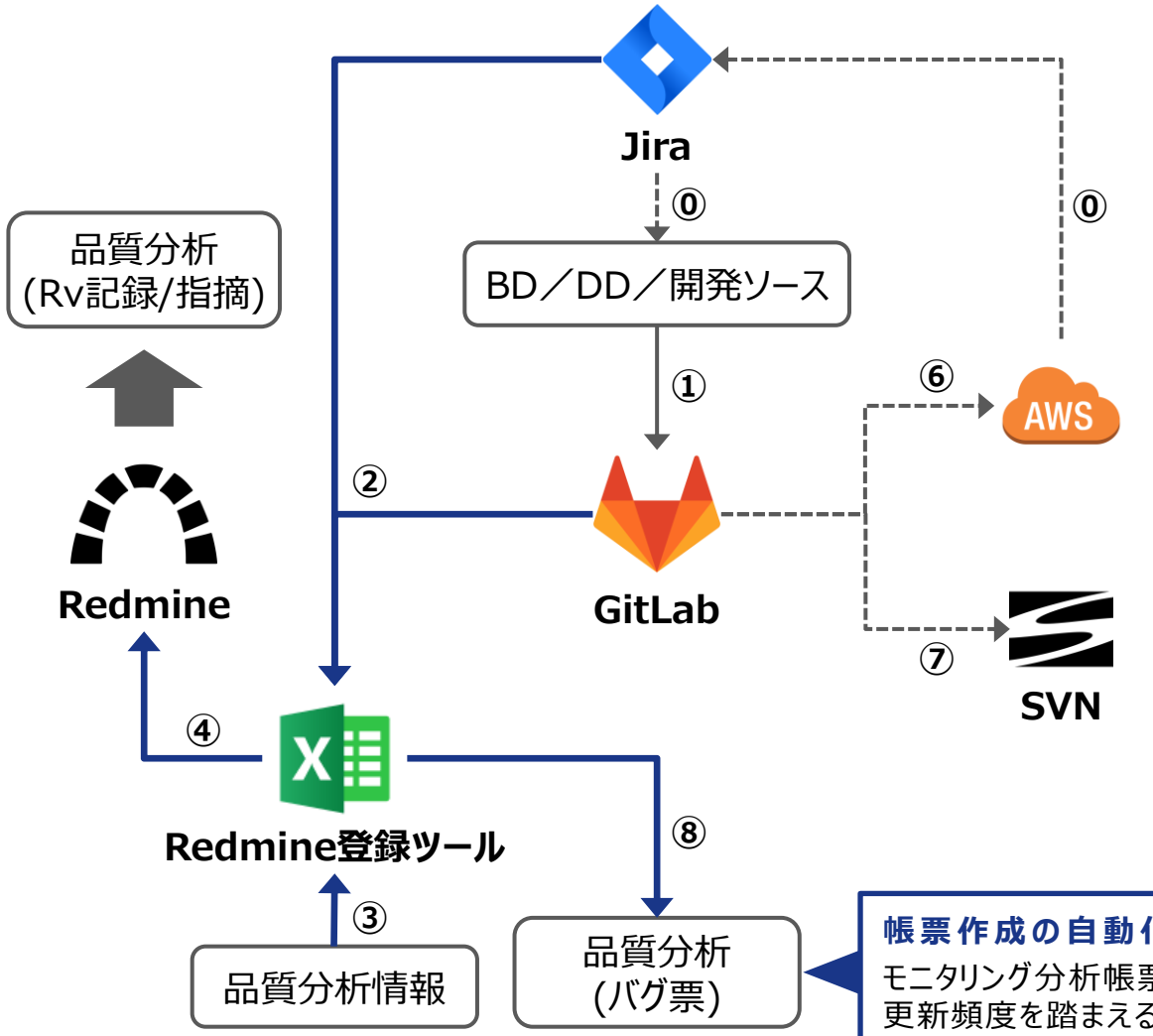


残課題

#	概要	詳細
1	バグ票のタイムリーな 更新と帳票作成の 自動化	①バグ票のタイムリーな更新に課題があり、後半モニタリング分析用帳票が タイムリーに更新できなかった ● 開発者の評価はベロシティなので、バグ票記入へのインセンティブがない ● 分散したツール間の転記が手間 ②帳票作成の自動化 モニタリング分析帳票作成の自動化が不十分。 更新頻度を踏まえると更なる自動化が必要
2	データドリブンな 仕組みへのモダナイズと 先進技術の活用	今回は品質管理プロセスの外部仕様の構築が主たる目的であったため、帳票ベースでの品質管理プロセスになった。今後、TDD高品質アジャイルの柔軟性・アジリティ・高生産性をより活かすには、データドリブンな仕組みを構築した上で、生成AIをはじめとした先進技術の活用も見据える必要がある。
3	分析観点、 メトリクスの拡充	品質問題の早期検知のために、分析観点としてはリソース品質やプロセス品質の拡充、メトリクスとしては成果物のコミットログ等も活用したデータドリブンな指標の導入なども検討していきたい
4	検証対象の拡大	今回は1件のPJ事例であったが、異なるPJでの有効性を判断する必要がある

参考）バグ票のタイムリーな更新と帳票作成の自動化

品質分析のワークフロー



▼開発トリガ

⓪Sprint開発情報をJiraでPBIとして管理する
- 横断POがJiraに開発要望のPBIを起票⇒BD
- POがJiraで開発用のPBIを起票⇒DD／開発ソース
- POがJiraでバグ修正用のPBIを起票⇒BD／DD／開発ソース

※バグ票はPBIと同時に起票＆適宜更新

▼品質分析ワークフロー

①登録時に指摘／修正内容を記入する
- 開発者がBD／DD／開発ソースをGitLabへ登録
- 開発者が横断PO／POへ登録申請
- 承認者の横断PO／POが指摘＆差し戻し
- 開発者が指摘反映+修正後に再申請
- 横断PO／POの承認(2名分)で登録完了

②GitLabとJiraから指摘／バグ情報／修正内容を転記する
- **GitLabから横断PO／POが各自の指摘／修正内容を転記**
- **JiraからPOがバグ情報／修正内容を転記**

③②以外の品質分析情報を記入する
- **横断PO／POが各自でRv記録／指摘／バグ票を記入**

④Redmineに登録する
- **横断PO／POが各自でRv記録／指摘をRedmineに登録**

※バグ票は登録しない

▼リリース作業

⑥GitLabとAWSを連携して各環境にリリース作業を自動化
- ST環境で定刻ST実施、開発者が任意でST実施
- 商用環境は顧客承認者トリガでリリース開始

▼Fix版BD管理

⑦顧客と共用のSVNで管理

▼STのすり抜け分析

⑧**バグ票はRedmineに登録せず**、バグ票からすり抜け分析を実施

バグ票のタイムリーな更新

（青字）に課題があり、
モニタリング分析も後半息切れ
- 開発者の評価はベロシティで、バグ票記入へのインセンティブがない
- 分散したツール間の転記が手間

帳票作成の自動化

モニタリング分析帳票作成の自動化が不十分。
更新頻度を踏まえると更なる自動化が必要

参考文献

1. アジャイル開発での品質保証. October, 22, 2020. Retrieved July, 8, 2023 from 株式会社NTTデータ: https://www.nttdata.com/jp/ja/data-insight/2020/102201/
2. アジャイル開発の品質管理技法. February, 6, 2020. Retrieved July, 8, 2023 from 富士通株式会社: https://www.fujitsu.com/jp/services/agile/featurestories/about-agile-02.html
3. アジャイル開発実践ガイドブック. March, 30, 2021. Retrieved July, 8, 2023 from 内閣官房情報通信技術（IT）総合戦略室: https://cio.go.jp/sites/default/files/uploads/documents/Agile-kaihatsu-jissen-guide_20210330.pdf
4. スクラムガイド. November, 1, 2020. Retrieved July, 8, 2023 from Ken Schwaber and Jeff Sutherland: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Japanese.pdf
5. スクラム入門 version 1.2. (2009). Retrieved July, 8, 2023 from Pete Deemer, Gabrielle Benefield, Craig Larman, Bas Vodde: https://www.goodagile.com/scrumprimer/scrumprimer_jp.pdf
6. 先進的な設計・検証技術の適用事例報告書 ２０１５年度版「アジャイルプロセスにおける実践的な品質向上施策の適用事例」. November, 18, 2015. Retrieved July, 8, 2023 from 独立行政法人情報処理推進機構: https://www.ipa.go.jp/archive/files/000049403.pdf
7. 町田欣史(2021). リソース品質に着目したアジャイル開発における品質管理. プロジェクトマネジメント学会秋季研究発表大会予稿集, 301-304.
8. 中本傑, 宮島雄一, 岡村龍也, 鈴森康弘(2022). エンタープライズ向けアジャイル開発におけるスコープマネジメントの考察. プロジェクトマネジメント学会秋季研究発表大会予稿集, 340-350.
9. 清水歩, 加藤大受 (2022). アジャイル開発における E2E テスト自動化の効率的なテスト設計手法の研究. ソフトウェアエンジニアリングシンポジウム2022論文集, 29-35.
10. 射場千尋(2023). Sprint ごとの不具合状況推移に着目した品質管理手法の適用事例. プロジェクトマネジメント学会春季研究発表大会予稿集, 177-190.
11. 石川研吾(2023). アジャイル開発における品質管理. プロジェクトマネジメント学会春季研究発表大会予稿集, 191-195.
12. 山本椋平, 石津大輔, 桑田直樹, 後藤卓司, 浅田隼人, 山村喜恒(2023). アジャイル開発の品質確保に向けたマネジメントポイント. プロジェクトマネジメント学会春季研究発表大会予稿集, 219-225.
13. 別府薫,佐伯明音, 遠藤圭太, 仁尾圭祐(2023). ウォーターフォール開発に照らしたアジャイル開発の品質管理事例と考察. プロジェクトマネジメント学会秋季研究発表大会予稿集, 158-165.
14. 綾野未来, 滝澤健人, 杉原直樹, 宮本充, 高槁僚史, 内山航輔, 渕上恭平(2024). ストーリーポイントを用いた品質評価手法の適用事例. プロジェクトマネジメント学会春季研究発表大会予稿集, 443-450.
15. 掛川悠, 大貫正也, 米原大輔(2024). TDD ベースの高品質アジャイルにおける品質管理プロセス構築事例. プロジェクトマネジメント学会秋季研究発表大会予稿集, 591-605.


NTT DATA