アジャイル開発に適した品質チェック項目の構造化

Structuring the quality check items suitable for agile software development

谷﨑 浩一，田上 諭，森 龍二，蛭田 恭章，森崎 修司

{kouichi.tanizaki, satoshi.tanoue, ryuuji.mori, yasuaki.hiruta}@veriserve.co.jp,

morisaki@i.nagoya-u.ac.jp

株式会社ベリサーブ，名古屋大学大学院情報学研究科

発表要旨：

従来型のウォーターフォール開発では，工程ごとに次工程に移行するための品質チェックの基準が設けられ，後戻りのないように上流工程から品質を作り込んでいく．一方，アジャイル開発などのイテレーティブな開発手法では，ウォーターフォール開発において工程ごとに実施されていた品質チェックが暗黙的となりやすく，欠陥の発見が遅れることがある．我々は，従来型の工程ごとの品質チェックとは異なる観点で，アジャイル開発に適した構造化された品質チェック項目を作成する方法を検討してきた．本発表では，その方法を実際のアジャイル開発プロジェクトに適用し，効果を確かめた結果を報告する．アジャイル開発のプラクティスのインプット・アウトプット・ステークホルダーを整理し，マトリクス形式で品質チェック項目が構造化できることを確かめた．その上で，品質チェック項目の効果を検証するため，過去に検出された欠陥のうち，品質チェック項目により早期検出可能な欠陥がどの程度あるか分析した．分析の結果，実際に検出された時点よりも早期に検出できた可能性がある欠陥があることが示され，当初の狙いを達成できていることが分かった．さらに，早期に検出できた可能性のある欠陥の多くが従来の要件定義のレビューやテストケースのレビューに該当するチェック項目で検出できた可能性があることが分かり，対象としたアジャイル開発プロジェクトの改善点を見出すことができた．

キーワード：

品質管理，アジャイル，スクラム，欠陥分析，フロントローディング，レビュー

想定している聴衆

アジャイル開発における品質チェック・品質向上に取り組んでいる方，アジャイル開発の成果物の品質に課題を感じている方

発表者の紹介（全角１００文字）：

株式会社ベリサーブにてソフトウェアテスト設計の支援ツールの開発，およびテスト設計手法やレビュー手法の研究に従事．テスト設計コンテスト’15　準優勝．

アジャイル開発に適した品質チェック項目の構造化

2020年9月10日

○谷﨑 浩一[1]，田上 諭[1]，森 龍二[1]，蛭田 恭章[1]，森崎 修司[2]
[1] 株式会社ベリサーブ，[2] 名古屋大学大学院情報学研究科

目次

- 本研究の背景
- 本研究の目的と実施内容
- 品質チェック項目の作成手法、本研究で作成した品質チェック項目
- 品質チェック項目による過去プロジェクトの欠陥分析
- 考察
- まとめ
- 今後の取り組み

本研究の背景

- **アジャイル開発においても、要所要所で品質チェックを行うなど、従来型（ウォーターフォール型）のような品質を担保する取り組みが必要**
  - 開発のサイクルの早い段階（スプリントの中の早い日）で、品質チェックを行い欠陥を早期検出できれば、品質担保の取り組みとして価値がある
    - 例：バックログアイテムの内容の不備に起因する欠陥は、実装着手前にバックログアイテムの品質チェックを行うことで検出できる可能性がある

- **そのための品質チェック項目の作成方法を我々は研究してきた**
  - アジャイル開発のステークホルダーとプラクティスに着目した品質チェック項目の作成手法
    - ソフトウェアエンジニアリングシンポジウム2020にて発表予定

本研究の目的と実施内容

VERISERVE

➢ 目的

品質チェック項目を実際のプロジェクトのデータに適用することで
欠陥の早期検出に役立つか確認する

➢ 実施内容

品質チェック項目を作成し、品質チェック項目による
過去プロジェクトの欠陥分析を実施

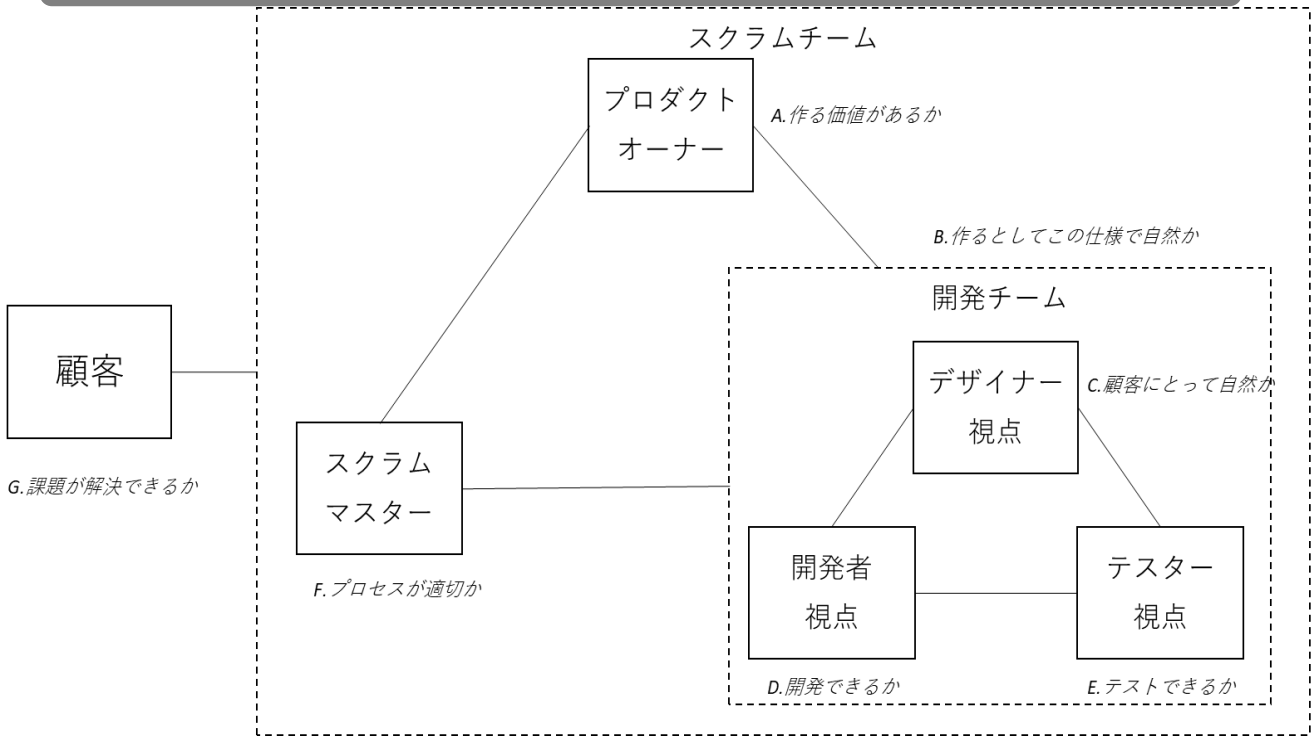
- 既知の欠陥のうち品質チェック項目により早期検出可能な欠陥がどれぐらいあったか確認する
- どのチェック項目でどのような欠陥を検出可能か確認する

品質チェック項目の作成手法

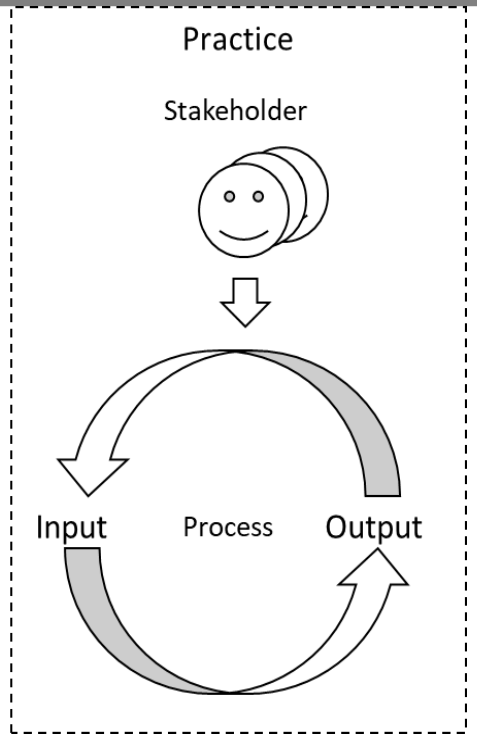
VERISERVE

- **①ステークホルダーの関心事をベースにチェック項目を検討する**
  - さまざまな視点からチェック内容を検討できる
- **②アジャイル開発のプラクティスにチェック項目を対応付ける**
  - どのタイミングで何をチェック対象とするかが明確になる

ステークホルダごとに様々な品質チェックの関心事がある



プラクティスには入出力がありステークホルダが関与する



谷﨑浩一，田上諭，森龍二，蛭田恭章，森崎修司：アジャイル開発に適した品質チェック項目の作成手法，ソフトウェアエンジニアリングシンポジウム2020論文集（掲載予定）

アジャイル開発のプラクティス

VERISERVE

- アジャイル開発では課題を解決するための様々な「習慣」を「プラクティス」と呼ぶ
- 様々なプラクティスが知られ、実践されている

カテゴリ	品質チェック項目	検出可能な欠陥の概要
プロセス	イテレーション計画ミーティング	イテレーション（スプリント）ごとのリリース計画やアクティビティなどを計画するミーティング
	イテレーション	ゴールや結果にアプローチするプロセスを繰り返すこと
	スプリントレビュー	完了した仕事を表明するスプリントレビューミーティング
プロダクト	ユーザーストーリー	要求についての会話を行うときの開発チームとプロダクトオーナーの間の合意事項
	スプリントバックログ	プロダクトオーナーとチーム間でのスプリントバックログへの相互コミットメント
	プロダクトバックログ（優先順位付け）	プロダクトオーナーによる優先順位（プロダクトバックログ）の管理

山下博之：アジャイル開発の現状と「プラクティス・リファレンスガイド」 ～アジャイル開発を適切に導入するために～，独立行政法人情報処理推進機構，https://www.ipa.go.jp/files/000035261.pdf

本研究で作成した品質チェック項目

VERISERVE

➢ 「プラクティス」×「ステークホルダの関心事」のマトリクスで整理

プラクティス

			ステークホルダの関心事				
			プロダクトオーナー	開発チーム			
					デザイナー視点	開発者視点	テスター視点
Practice	Process	Input/Output	A.作る価値があるか	B. 作るとしてこの仕様で自然か	C. この仕様で顧客にとって自然か	D. この仕様で開発できるか	E. この仕様でテストできるか
1. バックログアイテム作成	プロダクトバックログに新しいバックログアイテムを追加する	Input：顧客の要望 Output：プロダクトバックログ	A1-1. 誰のためのものか A1-2. 作りたいものは何か A1-3. なぜ必要なのか A1-4. 他のバックログとの関係性は適切か	B1-1. 仕様同士の関係性は適切か（無矛盾） B1-2. 実現手段を交渉可能か B1-3. 他のバックログと独立しているか	-	-	-
2. バックログリファインメント	・バックログアイテムの不明確な点を詳細化する ・バックログアイテムを適切な大きさに分割する ・バックログアイテムの優先順を見直す	Input：プロダクトバックログ Output：プロダクトバックログ	同上	同上	C2-1. 顧客の価値を最大化できるか C2-2. 実際にエンドユーザが使っていくことができるか	D2-1. 仕様が複雑すぎないか D2-2. 受け入れ基準が適切に定義されているか D2-3. 技術的な制約が考慮されているか D2-4. 規模が大きすぎないか D2-5. 見積もり可能か	E2-1. 実行すべきテストケース・テストシナリオを特定できるか E2-2. 自動化の実装が難しい部分がないか
7.スプリント-テスト設計	テストケースを作成する	Output：テストケース	-	-	-	-	E7-1. 受け入れ基準を満たすことを確認できるテストケースになっているか E7-2. 自動化の実装が難しい部分がないか
8.スプリント-テスト実装	テストコードを実装するする	Output：テストコード	-	-	-	-	E8-1.テストケースを網羅しているか E8-2.テストコードが正しく実装されているか

※紙幅の都合で一部のみ掲載
全体は最後の付録を参照

品質チェック項目が欠陥の早期検出に役立つか確認する

VERISERVE

既知の欠陥を早期検出できたか、さかのぼって欠陥を分析してみよう

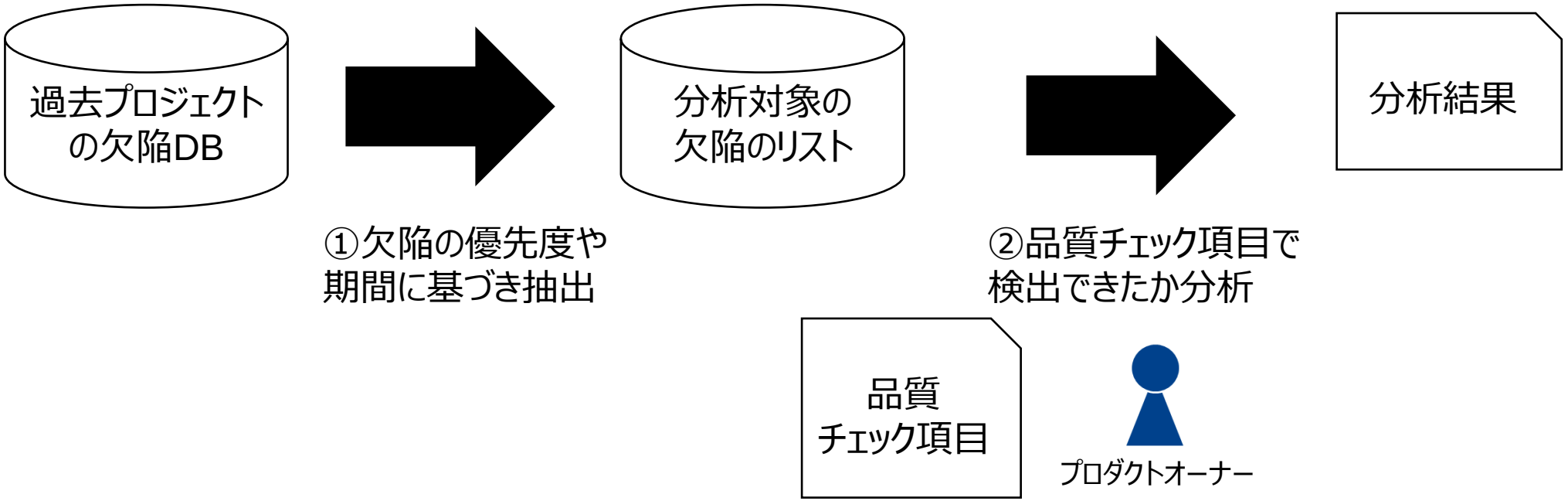
			Stakeholderの関心事				
			プロダクトオーナー	開発チーム			
					デザイナー視点	開発者視点	テスター視点
Practice	Process	Input/Output	A.作る価値があるか	B. 作るとしてこの仕様で自然か	C. この仕様で顧客にとって自然か	D. この仕様で開発できるか	E. この仕様でテストできるか
1. バックログアイテム作成	プロダクトバックログに新しいバックログアイテムを追加する	Input：顧客の要望 Output：プロダクトバックログ	A1-1. 誰のためのものか A1-2. 作りたいものは何か A1-3. なぜ必要なのか A1-4. 他のバックログとの関係性は適切か	B1-1. 仕様同士の関係性は適切か（無矛盾） B1-2. 実現手段を交渉可能か B1-3. 他のバックログと独立しているか	-	-	-
2. バックログリファインメント	・バックログアイテムの不明確な点を詳細化する ・バックログアイテムを適切な大きさに分割する ・バックログアイテムの優先順を見直す	Input：プロダクトバックログ Output：プロダクトバックログ	同上	同上	C2-1. 顧客の価値を最大化できるか C2-2. 実際にエンドユーザが使っていくことができるか	D2-1. 仕様が複雑すぎないか D2-2. 受け入れ基準が適切に定義されているか D2-3. 技術的な制約が考慮されているか D2-4. 規模が大きすぎないか D2-5. 見積もり可能か	E2-1. 実行すべきテストケース・テストシナリオを特定できるか E2-2. 自動化の実装が難しい部分がないか
7.スプリント-テスト設計	テストケースを作成する	Output：テストケース	-	-	-	-	E7-1. 受け入れ基準を満たすことを確認できるテストケースになっているか E7-2. 自動化の実装が難しい部分がないか
8.スプリント-テスト実装	テストコードを実装するする	Output：テストコード	-	-	-	-	E8-1.テストケースを網羅しているか E8-2.テストコードが正しく実装されているか

欠陥分析の結果から、プロジェクトの改善点も見えるかも？

品質チェック項目による過去プロジェクトの欠陥分析

VERISERVE

- **実施者：プロダクトオーナー1名**
- **実施内容：品質チェック項目によって、既知の欠陥を検出時より早い段階で見つけられたかを確認**
- **実施手順：**
  - ①過去プロジェクトの欠陥DBから優先度の低い欠陥などを除外
  - ②品質チェック項目と欠陥情報を照らし合わせ、品質チェック項目で検出できたかどうか分析



分析の対象としたプロジェクト

VERISERVE

- PCアプリケーション開発プロジェクトを対象とした
- スクラムを参考にした体制・開発の進め方をしている

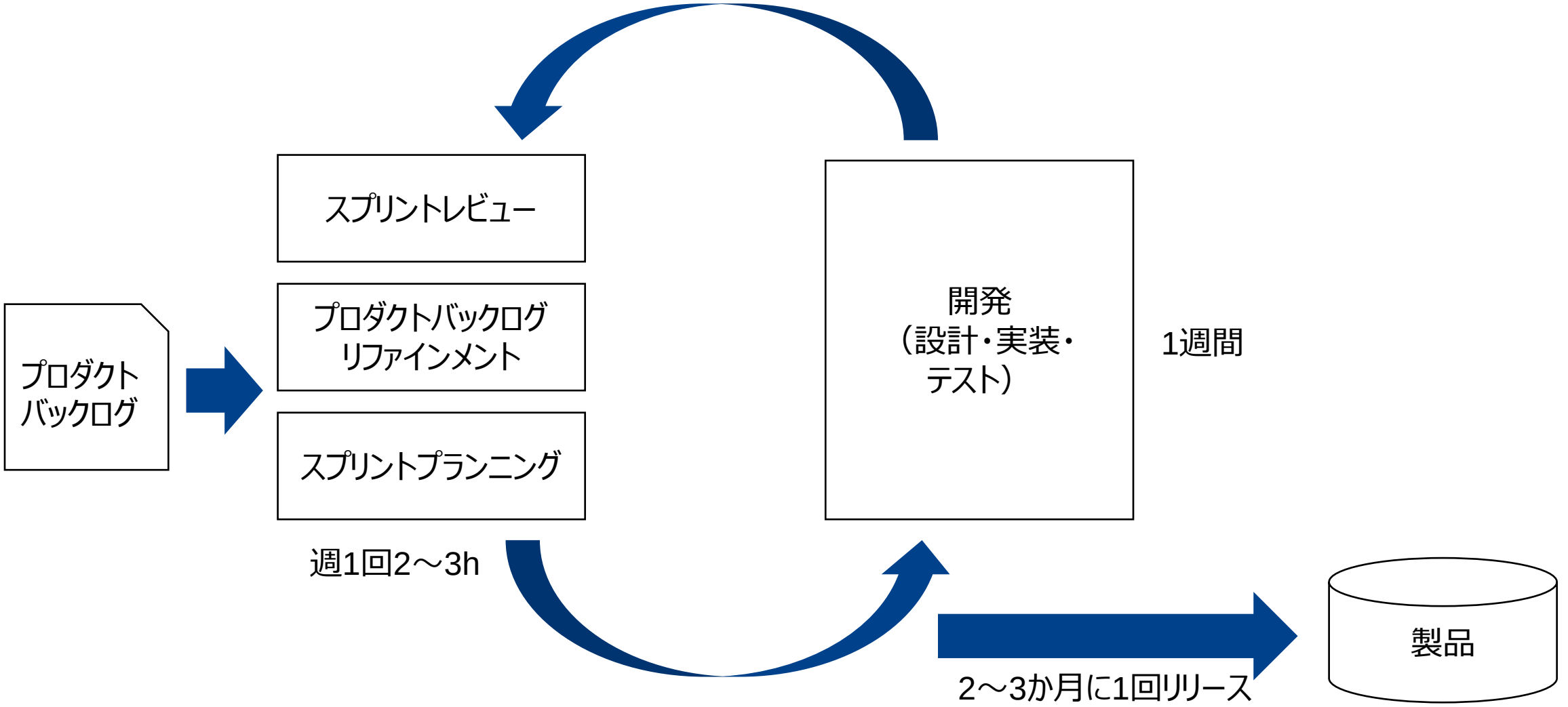
対象プロジェクトの概要

概要	PCアプリケーションの開発プロジェクト
分析の対象とした開発期間	約1年半
開発手法	アジャイル開発（スクラムを参考にしている）
人数	4名程度（増減あり）
分析に利用した欠陥の収集フェーズ	受け入れテスト以降（ブラックボックステストで見つかった欠陥やユーザから報告のあった欠陥を対象とした）

対象プロジェクトの開発の進め方

VERISERVE

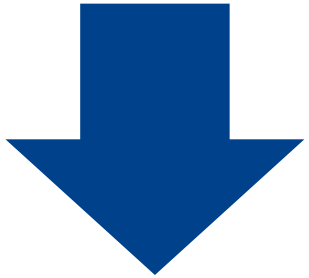
- アジャイル開発の一手法であるスクラムを参考にして開発を進めていた



分析対象の欠陥の抽出ルール

VERISERVE

- **ステータスが「却下」の欠陥は除外した**
- **優先度が「低め」の欠陥は除外した**
  - 除外理由：対応不要だった欠陥や優先度が低い欠陥は、品質チェックで頑張って早期検出しなくてもよいものが多いと考えた



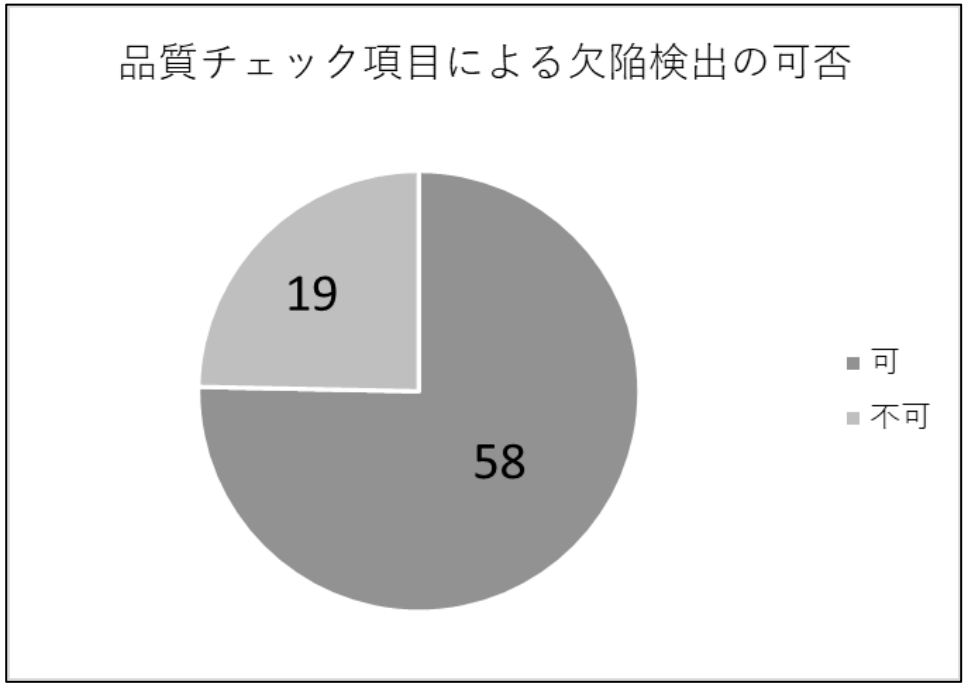
77件の欠陥が対象となった

優先度	説明
急いで（ランクS）	復旧手段のないデータ消失/破壊、パフォーマンスの著しい低下など、ユーザにとって致命的な欠陥
高め（ランクA）	データ編集中のシステムクラッシュや作業を継続できないなど、ユーザへの影響が大きい欠陥
通常（ランクB）	機能の動作が仕様と異なる問題や頻繁に利用されるケースでのユーザ期待との相違など、ユーザへの影響が中程度の欠陥
低め（ランクC）	軽微な表示崩れやレアケースで発生する問題など、ユーザへの影響が少ない欠陥

欠陥分析の結果　～品質チェック項目で欠陥を検出できたか～

VERISERVE

- **品質チェック項目によるチェックを行っていれば、約75%の欠陥を、実際に検出されたタイミングよりも早期に検出できた可能性がある**

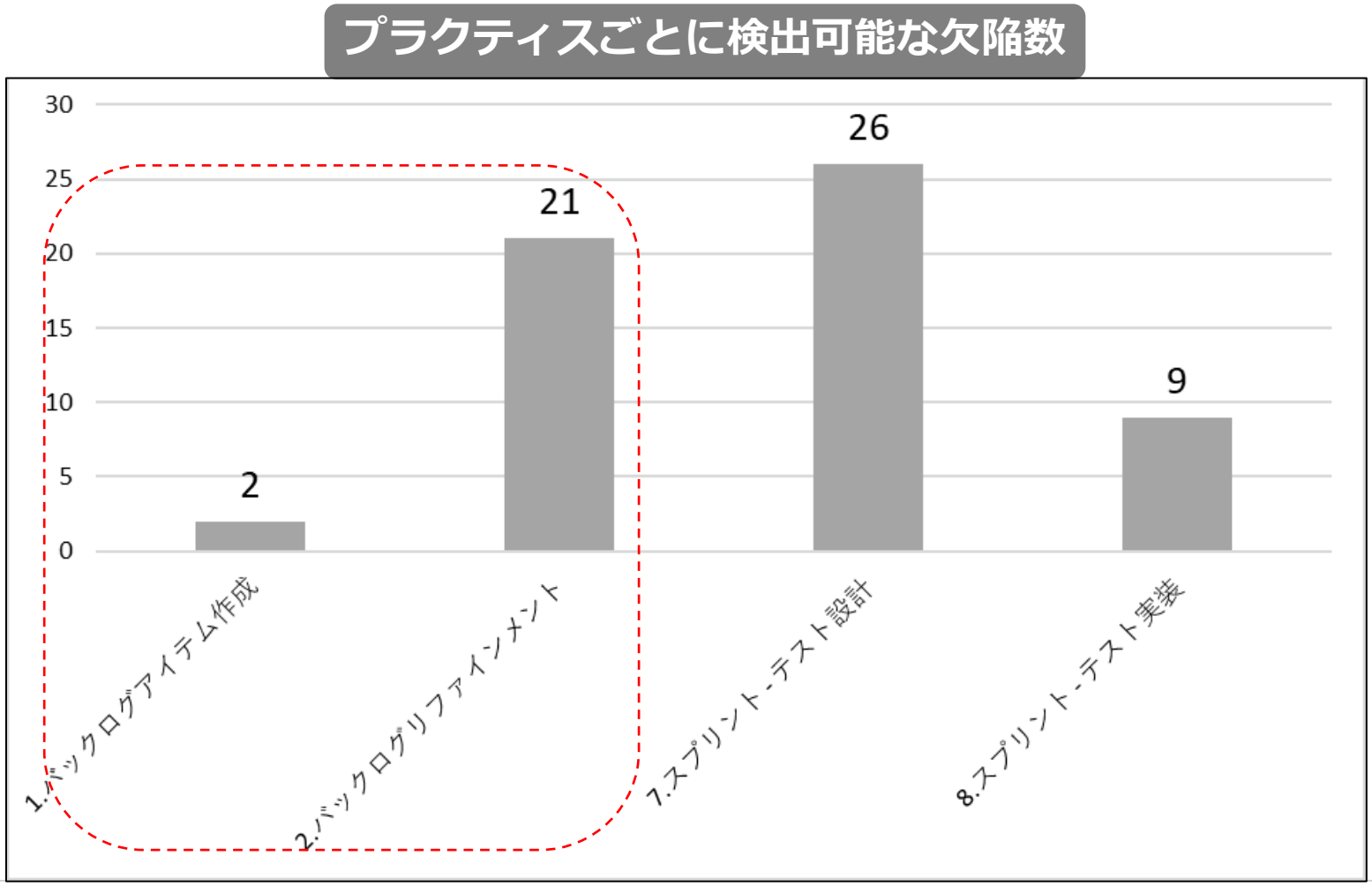


- **検出可能だった不具合を次頁以降でさらに分析する**
  - プラクティスごとに検出可能な欠陥数を分析　→　実装着手前の早期に欠陥検出可能かを確認する
  - 品質チェック項目ごとに検出可能な欠陥数を分析　→　対象プロジェクトで見逃しが多かった点を確認する

欠陥分析の結果　～どのプラクティスで欠陥を検出できたか～

VERISERVE

- 半数近くの欠陥はバックログアイテム作成やバックログリファインメントの段階（＝対象プロジェクトにおいては実装着手前の段階）で検出できた可能性があることが分かった



実装着手前に検出できた可能性がある欠陥

- バックログアイテム作成やバックログリファインメントのタイミングで、バックログアイテムをチェックすることで、以下の欠陥を検出できた可能性がある

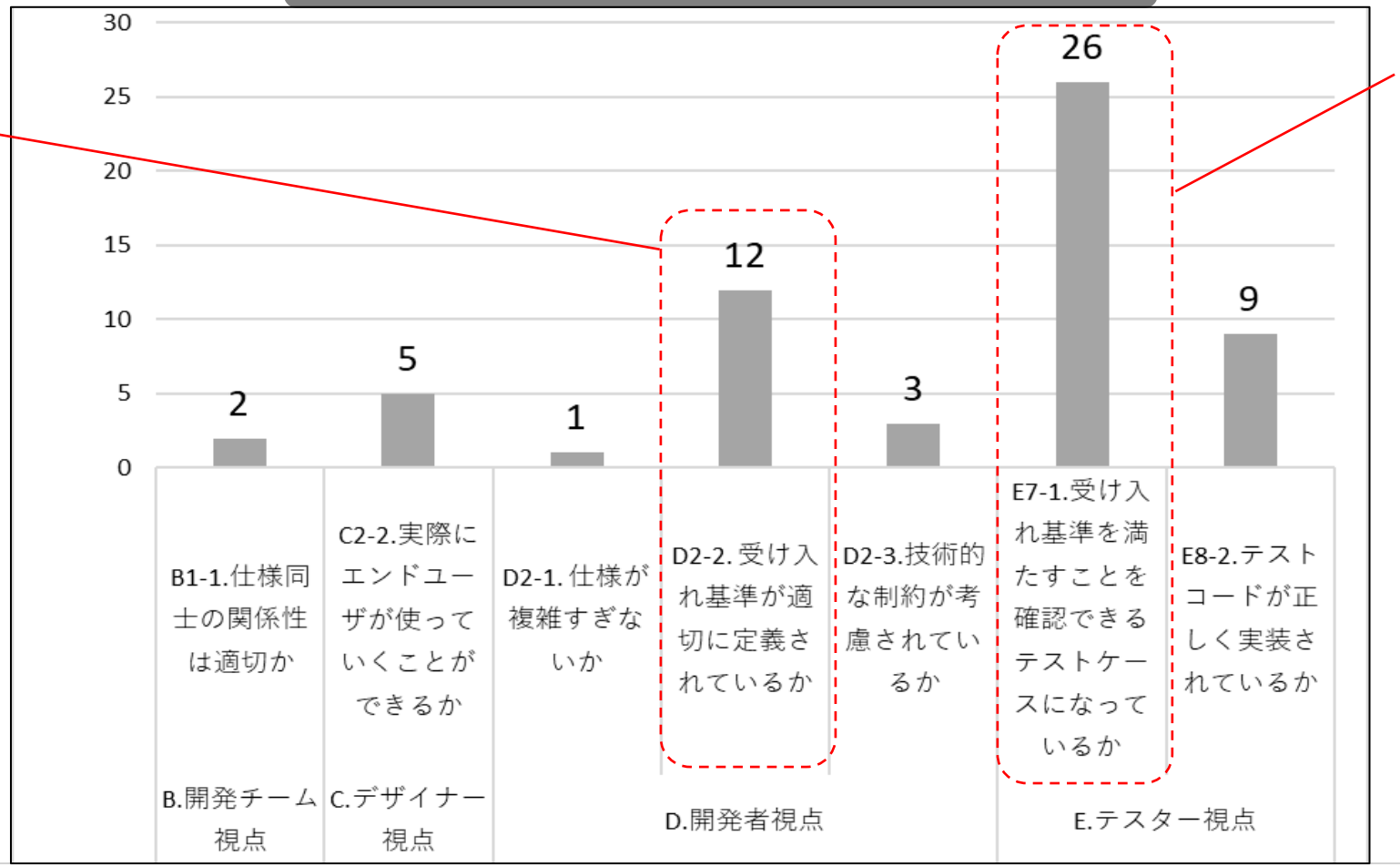
品質チェック項目	検出可能な欠陥の概要
実際にエンドユーザが使っていくことができるか	エンドユーザにとって不自然な操作感やメッセージ内容の不備といった欠陥
技術的な制約が考慮されているか	使用しているライブラリに起因する欠陥
仕様同士の関係性は適切か（無矛盾）	他の機能で作成したデータとの矛盾が発生する欠陥
受け入れ基準が適切に定義されているか	データ量の上限や非機能要件の定義漏れ

欠陥分析の結果　～どのチェック項目で欠陥を検出できたか～

VERISERVE

- 従来型の「仕様書のレビュー」や「テストケースのレビュー」に該当する品質チェック項目で検出できた可能性がある欠陥が多かった

品質チェック項目ごとに検出可能な欠陥数



仕様書のレビューに該当する

テストケースのレビューに該当する

「受け入れ基準が適切に定義されているか」で検出できた可能性のある欠陥

VERISERVE

- **サイズの大きいファイルをインポートした場合に異常終了する**
  - ファイルサイズの上限がバックログアイテムで規定されていなかった
- **特定の画面の操作時のレスポンスが悪い**
  - レスポンスの目標がバックログアイテムで明確に定義されていなかった
- **画面を切り替えた際の表示の更新が全体的に遅い**
  - 表示更新の性能の目標がバックログアイテムで定義されていなかった

対象プロジェクトの改善点：バックログアイテムのレビューを行い、
データ量の上限や性能面の目標などを明確に定義すると良い

「受け入れ基準を満たすことを確認できるテストケースになっているか」で検出できた可能性のある欠陥

VERISERVE

- **特定の操作を行うと、本来作成できないデータを作成できる**
  - 操作のパターンを考慮したテストが不足していた
- **空のファイルをインポートすると例外エラーが発生する**
  - 空のファイルに対するテストが不足していた
- **特別な記号（<,>,&など）が含まれる文字列を入力すると文字化けする**
  - 特別な記号に対するテストが不足していた

対象プロジェクトの改善点：テストケースのレビューを行い、
パターンの不足を補うと良い

考察

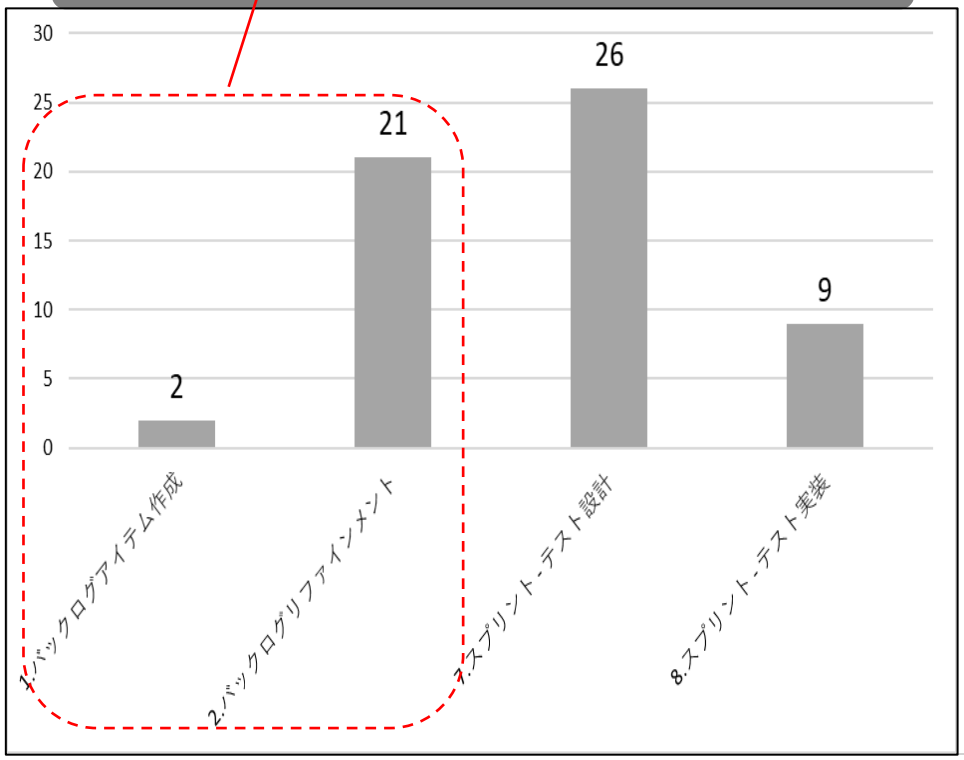
VERISERVE

- 品質チェック項目により実装着手前の早期に欠陥を検出できる
- 欠陥分析により対象プロジェクトの改善点を見出すことができる

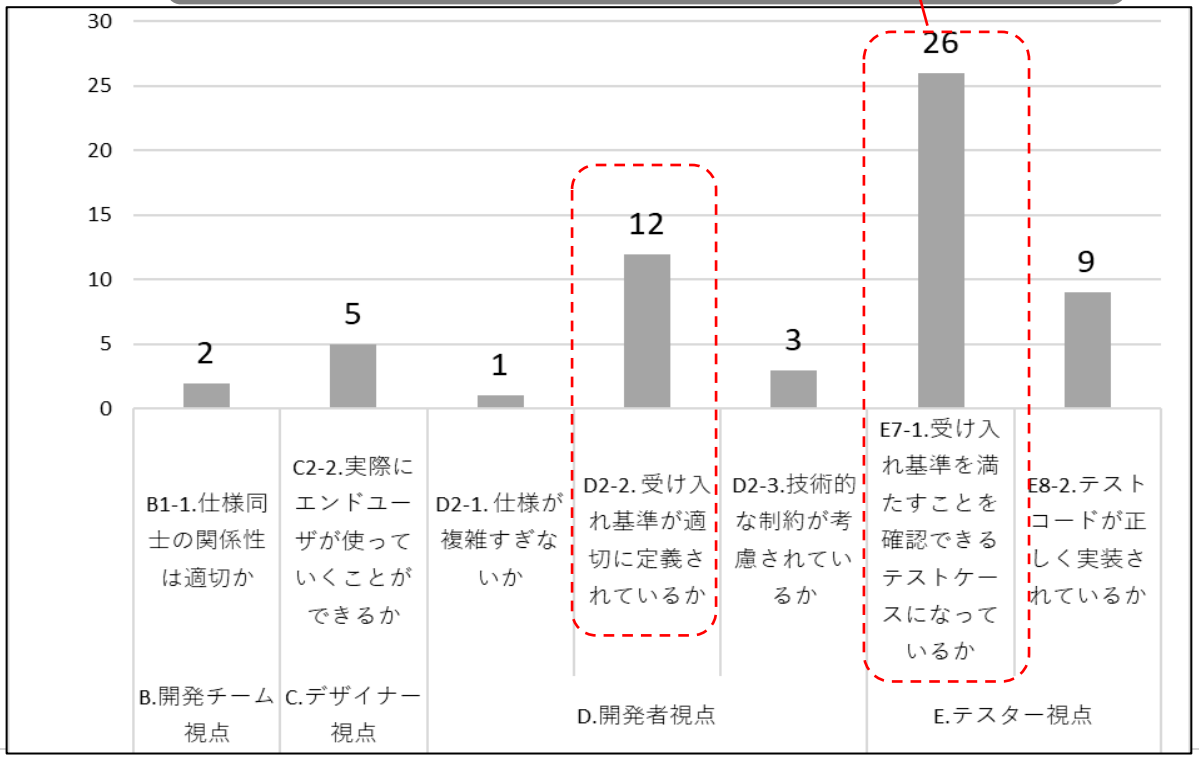
実装着手前の早期に欠陥を検出できると、フロントローディングの実現に繋がる

従来型の仕様書やテストケースのレビューに該当する活動が対象プロジェクトの改善点と言える。これらの活動はアジャイル開発では明確に定義されず、これらの活動で見つかっていた欠陥が見逃されやすい可能性がある

プラクティスごとに検出可能な欠陥数



品質チェック項目ごとに検出可能な欠陥数



まとめ

VERISERVE

- **「アジャイル開発に適した品質チェック項目の作成手法」に従って品質チェック項目を作成した**
  - 「プラクティス」×「ステークホルダの関心事」のマトリクスで整理

- **品質チェック項目で過去プロジェクトの欠陥を分析した**
  - 早期に検出可能なものが約75%、そのうち半数近くは実装着手前に検出できる可能性あり
  - 対象プロジェクトの改善点を見出すことにもつながった

品質チェック項目をプロジェクトで利用することで欠陥の早期検出に役立つ

今後の取り組み

- **品質チェック項目をあらかじめ適用したプロジェクトでの効果検証**
- **スクラム以外のアジャイル開発手法での品質チェック項目の作成**
  - 前者は試行中
    - エンドユーザが本当に解決したい課題が不明確なバックログアイテムや、このまま開発すると使い勝手が悪くなると思われるバックログアイテムに対して、実装着手前に欠陥を指摘できた
    - 指摘事項を考える際の「とっかかり」として使える、との意見を利用者から得た

あまり自分が意識していない観点でもチェックできた

チェック項目を使うと時間はかかったが、そこまで増えすぎという感覚はなかった

アドホックレビューで検出できる欠陥も含め、チェック項目で検出できる

付録：品質チェック項目（全体）

VERISERVE

			Stakeholderと関心事						
			顧客	プロダクトオーナー	開発チーム				スクラムマスターの視点
						デザイナー視点	開発者視点	テスター視点	
Practice	Process（Practiceで実施していること）	Input/Output（レビュー対象）	課題が解決できるか	作る価値があるか	作るとしてこの仕様で自然か	この仕様で顧客にとって自然か	この仕様で開発できるか	この仕様でテストできるか	プロセスが適切か
バックログアイテム作成	プロダクトバックログに新しいバックログアイテムを追加する	Input：顧客の要望 Output：プロダクトバックログ	・顧客が解決したい課題は何か	・誰のためのものか ・作りたいものは何か ・なぜ必要なのか ・他のバックログとの関係性は適切か（無矛盾・無重複・副作用・優先順位）	・仕様同士の関係性は適切か（無矛盾） ・実現手段を交渉可能か ・他のバックログと独立しているか	-	-	-	・追加されたバックログの内容をスクラムチームが共有する場があるか
バックログリファインメント	・バックログアイテムの不明確な点を詳細化する（開発に着手できるようにする） ・バックログアイテムを適切な大きさに分割する ・バックログアイテムの優先順を見直す	Input：プロダクトバックログ Output：プロダクトバックログ	-	同上	同上	・顧客の価値を最大化できるか ・実際にエンドユーザが使っていくことができるか ・デザインや操作感に統一性があるか	・仕様が複雑すぎないか ・受け入れ基準が適切に定義されているか ・技術的な制約が考慮されているか ・規模が大きすぎないか ・見積もり可能か	・実行すべきテストケース・テストシナリオを特定できるか ・自動化の実装が難しい部分がないか	
スプリントプランニング	・ストーリーポイントを見積もる ・プロダクトバックログからスプリントバックログへ移動する・合意する	Input：プロダクトバックログ Output：スプリントバックログ	-	同上	同上	同上	同上	同上	・スクラムチームのベロシティに納まる分量をスプリントバックログに入れているか ・スクラムチーム全員が結果に合意しているか
ディリースクラム	スプリントのゴールを達成するためにやることを議論する	Input：スプリントバックログ Output：アサインされたスプリントバックログ	-	-	-	-	-	-	・適切な時間内で実施しているか ・タスクと担当者が明確になっているか ・未アサインのスプリントバックログが無いか
スプリント	設計する	Output：設計に関する絵や文書や会話	-	-	-	-	・アーキテクチャが適切か	-	・開発チームの状況が可視化されているか ・開発チーム内で適切にコミュニケーションを取れているか
	コーディングする	Output：ソースコード	-	-	-	-	・コードが適切か	-	
	テスト設計する	Output：テストケース	-	-	-	-	-	・受け入れ基準を満たすことを確認できるテストケースになっているか ・自動化の実装が難しい部分がないか	
	テスト実装する	Output：テストコード	-	-	-	-	-	・テストケースを網羅しているか ・テストコードが正しく実装されているか	
スプリントレビュー	インクリメントのデモを見てフィードバックを出す	Input：インクリメント Output：フィードバック	・この仕様で使っていけるか ・現状の課題が解決できるか	-	-	-	-	-	・適切な顧客を呼んでいるか ・フィードバックがプロダクトバックログに加えられているか
レトロスペクティブ	KPTを洗い出して共有する Tryを決めて合意する	Output：Try	-	-	-	-	-	-	・スクラムチームの全員が参加しているか ・Tryが次回のスプリントバックログに加えられているか

ご清聴ありがとうございました