アジャイル開発における欠陥検出のフロントローディングのための品質チェック方法の提案

The quality check methods for frontloading of defect detection in 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：

アジャイル開発において，イテレーションの中で欠陥検出のフロントローディングができれば，イテレーションの期間中に品質の高いプロダクトを開発することが可能になる．しかし，アジャイル開発における欠陥検出のフロントローディングの具体的な方法は明らかになっていない．従来のウォータフォール型の開発では要求仕様書の妥当性確認などの方法でフロントローディングを実現していたが，同じ方法をアジャイル開発にそのまま適用するのは難しい．アジャイル開発ではチームやプロダクトに合わせて開発の方法を変えるため，統一的な方法をそのまま使うことに対する難しさもある．本研究ではこれらの課題を解決するために，従来型の開発のフロントローディングの方法を応用した，アジャイル開発における欠陥検出のフロントローディングの具体的な方法を提案する．先行研究で定義されたアジャイル開発向けの品質チェック項目から，イテレーションの初期段階または開始前に実施されるバックログ作成・バックログリファインメントに関連する品質チェック項目を抽出し，従来型の開発でどのように品質を担保していたかを明らかにしたうえで，アジャイル開発に置き換え具体的な品質チェック方法を定義した．定義した品質チェック方法を用いて実際に欠陥の検出が可能かどうかを確認するために，過去のアジャイル開発で検出された欠陥に品質チェック方法を当てはめてチェックを行った．結果として，本研究で定義した品質チェック方法をバックログ作成やバックログリファインメントのタイミングで適用することで，実装前の早期段階で欠陥を検出可能であることが分かり，アジャイル開発における欠陥検出のフロントローディングが可能になることを確認できた．

キーワード：

スクラム，品質保証，バックログリファインメント，品質チェック項目

想定している聴衆

アジャイル開発において品質向上に取り組んでいる方，成果物品質に課題を感じている方

発表者の紹介（全角１００文字）：

2008 年，株式会社ベリサーブ入社．テスト設計支援ツールのプロダクトオーナとしてアジャイル開発を実践しつつ，テスト設計手法やレビュー手法の研究に従事．博士（工学）．

アジャイル開発における欠陥検出のフロントローディングのための品質チェック方法の提案

2021年9月9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

背景

- **アジャイル開発において欠陥検出のフロントローディングの方法を明らかにする必要がある**
  - イテレーションの中で欠陥検出のフロントローディングができれば、イテレーションの期間中に品質の高いプロダクトを開発でき、ユーザに早く価値を届けることにつながる

- **アジャイル開発における欠陥の早期検出の方法として、筆者らは先行研究にて「アジャイル開発向けの品質チェック項目」を提案したが、チェックの方法は具体化されていない**
  - 定義した項目だけチェックすれば良いというわけではなく、チームでプロダクトの品質向上について議論する際に活用できるようにしたい

- **従来のウォータフォール型の開発ではフロントローディングの方法が議論されており、アジャイル開発に応用できる可能性があるが、そのままでは適用できない**
  - 要求仕様書の妥当性確認、設計レビュー、見積もりの方法など、そのままでは適用が難しい
  - アジャイル開発ではチームやプロダクトに合わせて開発の方法を変えるため、統一的な方法をそのまま使うことに対する難しさもある

目的と実施内容

➢ 目的

ウォータフォール型開発の知見を応用し
アジャイル開発における欠陥検出の
フロントローディングを実践しやすくする

➢ 実施内容

先行研究で提案された品質チェック項目に対して
ウォータフォール型開発とアジャイル開発を比較しながら
具体的な品質チェックの方法を提案する

先行研究

VERISERVE

- アジャイル開発のプラクティスとステークホルダの関心事に着目して品質チェック項目を定義した
- アジャイル開発では定型のチェック項目よりもチームにとってベストな方法や顧客へのより高い価値を優先するため、画一的なチェック項目ではなく柔軟かつ具体的なチェック項目が必要である

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	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	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

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構造化，ソフトウェア品質シンポジウム2020 より一部引用

実施内容の詳細

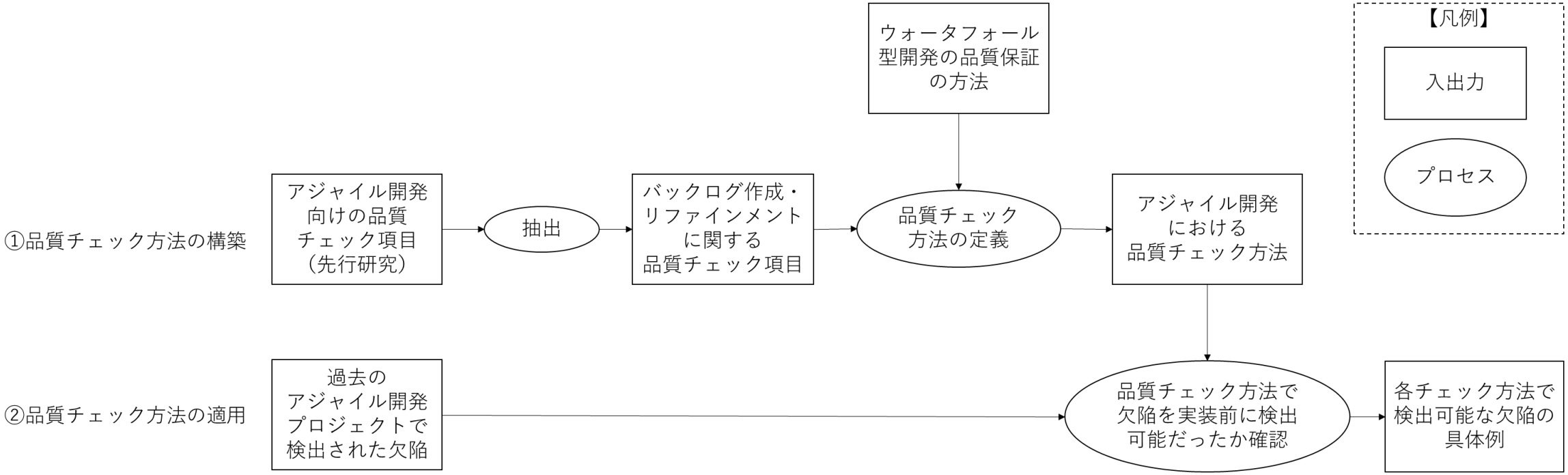
VERISERVE

➢ ①品質チェック方法の構築

- 先行研究のチェック項目38件中10件に対し、ウォータフォール型開発で品質を担保していた方法を洗い出し、その方法がアジャイル開発ではどのような方法に置き換わるかを具体的に定義した

➢ ②品質チェック方法の適用

- 過去プロジェクトの欠陥を対象に、構築した品質チェック方法で実装前に検出可能かを確認した



品質チェック方法の適用の対象とした過去プロジェクト

VERISERVE

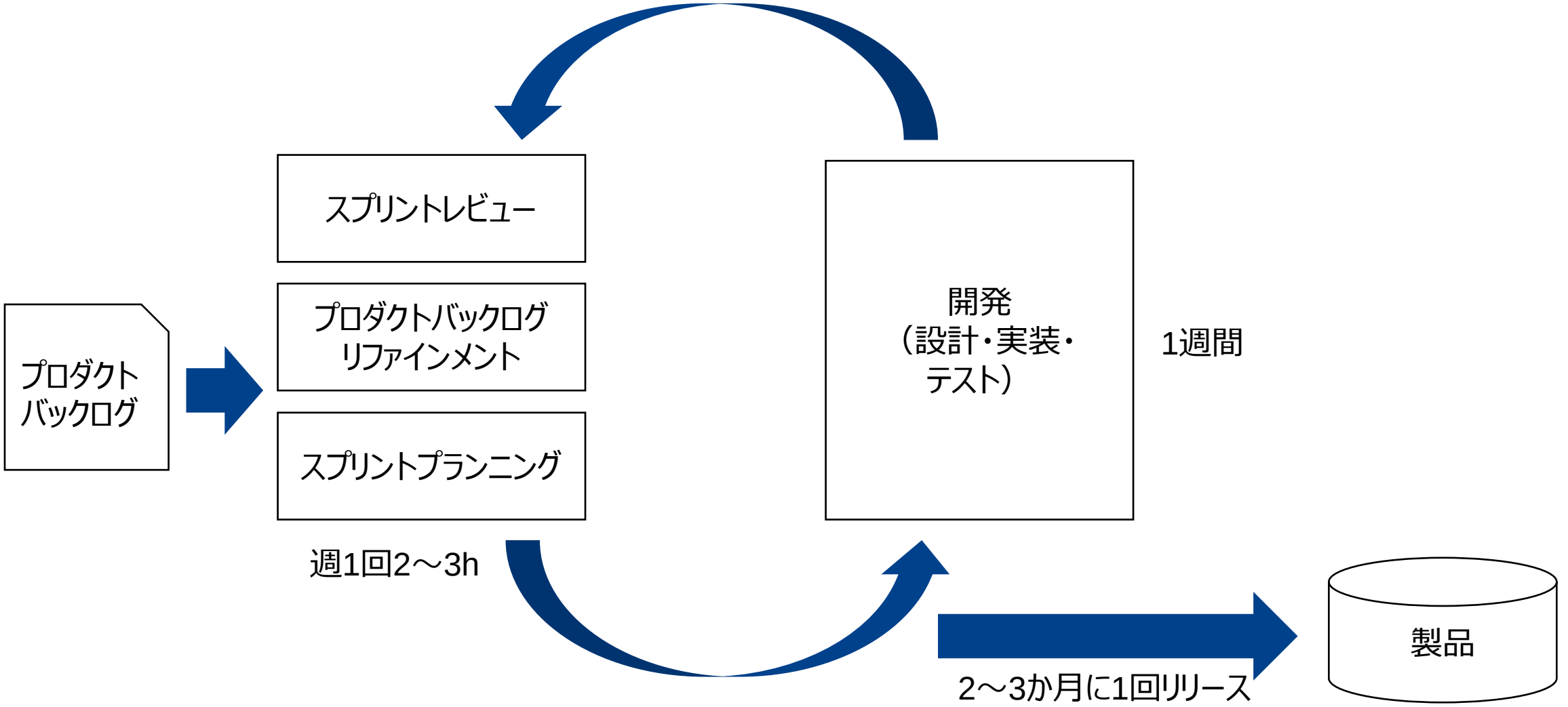
- PCアプリケーション開発プロジェクトを対象とした
- スクラムを参考にした体制・開発の進め方をしている

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
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）
確認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

- スクラムを参考にして開発を進めていた



実施結果

VERISERVE

➢ ①品質チェック方法の構築

- バックログ作成・バックログリファインメントに関連する品質チェック項目10件に対し、品質チェック方法を定義した　※下表に一部掲載。全体は最後の付録を参照

➢ ②品質チェック方法の適用

- 過去プロジェクトの欠陥のうち、定義したチェック方法で未然に検出可能な欠陥があることを確認した

※次ページ以降でチェック方法ごとに検出可能な欠陥の具体例を紹介

品質チェック項目	ウォータフォール型開発で品質を担保していた方法	アジャイル開発における品質チェック方法	検出可能な欠陥の具体例	
Q5: 仕様同士の関係性は適切か	要求仕様書のレビューで、仕様書間での整合性や矛盾がないかをチェックする	過去のイテレーションで開発した部分の仕様と、今回のバックログや将来のバックログの内容が矛盾しないか、不整合がないか確認する。今回のバックログの内容を実装した場合に、既存の機能に悪影響が出ないか、影響範囲に対するテストが膨大になり過ぎないか確認する。	ある画面で作成した項目を、別の画面で削除した場合、元の画面に表示され続ける（過去のイテレーションで作った画面への影響が考慮されていなかった）	← 後述の具体例①
Q8: 仕様が複雑すぎないか	FP法などで全体の開発工数を見積ることで、複雑すぎる部分（FPが大きすぎる部分）がないか確認する	開発する内容に対して、テストする際の条件やその組み合わせが多すぎないか確認する。バックログのストーリポイントを見積もる際に、テストの作業量も考慮して見積もり、ポイントが大きくなり過ぎないか確認する。	特定の操作を実施した場合に、操作対象以外の要素を操作できてしまう（操作手順と操作対象の組み合わせでテストすべきパターンが多く、該当のパターンを開発時に想定できなかった）	← 後述の具体例②
Q9: 受け入れ基準が適切に定義されているか	要求仕様書にあいまいさや不明な点がないかをレビューでチェックする	バックログにあいまいさや不明な点がないか確認する。バックログから開発のタスクを分解できるか、テスト項目を作成できるか確認する。ストーリポイントの見積もりを行い、チームメンバ間で作業内容に認識のずれがないか確認する。	特定の操作手順で要素を操作したときの動作がユーザの期待するものと違っていた（特定の操作手順に対しての仕様が詳細に決められていなかった）、 タブ切り替えした際の画面表示の更新が遅い（表示更新のタイミングが明確になっていなかった）	← 後述の具体例③

品質チェック方法の具体例①

VERISERVE

品質チェック項目

- 仕様同士の関係性は適切か

ウォータフォール型開発で品質を担保していた方法

- 要求仕様書のレビューで、仕様書間での整合性や矛盾がないかをチェックする

アジャイル開発における品質チェック方法

- 過去のイテレーションで開発した部分の仕様と、今回のバックログや将来のバックログの内容が矛盾しないか、不整合がないか確認する
- 今回のバックログの内容を実装した場合に、既存の機能に悪影響が出ないか、影響範囲に対するテストが膨大になり過ぎないかを確認する

品質チェック方法の具体例①

VERISERVE

- **過去のイテレーションで開発した部分の仕様と、今回のバックログや将来のバックログの内容が矛盾しないか、不整合がないかを確認する**

品質チェック方法の具体例①

VERISERVE

- 今回のバックログの内容を実装した場合に、既存の機能に悪影響が出ないか、影響範囲に対するテストが膨大になり過ぎないかを確認する

品質チェック方法の具体例①　検出可能な欠陥

VERISERVE

- **欠陥の内容：ある画面で作成した項目を、別の画面で削除した場合、元の画面に表示され続ける**
  - 過去のイテレーションで作った画面との関係性を考慮していれば、未然に検出できた

～あるチームのオンライン会議にて～

品質チェック方法の具体例②

VERISERVE

品質チェック項目

- 仕様が複雑過ぎないか

ウォータフォール型開発で品質を担保していた方法

- FP法などで全体の開発工数を見積もることで、複雑過ぎる部分 (FPが大き過ぎる部分)がないかを確認する

アジャイル開発における品質チェック方法

- 開発する内容に対して、テストする際の条件やその組み合わせが多過ぎないか確認する
- バックログのストーリポイントを見積もる際に、テストの作業量も考慮して見積もり、ポイントが大きくなり過ぎないかを確認する

品質チェック方法の具体例②

VERISERVE

- 開発する内容に対して、テストする際の条件やその組み合わせが多過ぎないかを確認する

品質チェック方法の具体例②

VERISERVE

- バックログのストーリポイントを見積もる際に、テストの作業量も考慮して見積もり、ポイントが大きくなり過ぎないかを確認する

品質チェック方法の具体例②　検出可能な欠陥

VERISERVE

- **欠陥の内容：特定の操作を実施した場合に、操作対象以外の要素を操作できてしまう**
  - 操作対象と操作方法の組み合わせでテストすべきパターンが多いことを考慮していれば、未然に検出できた

～あるチームのオンライン会議にて～

品質チェック方法の具体例③

VERISERVE

品質チェック項目

- 受け入れ基準が適切に定義されているか

ウォータフォール型開発で品質を担保していた方法

- 要求仕様書にあいまいさや不明な点がないかをレビューでチェックする

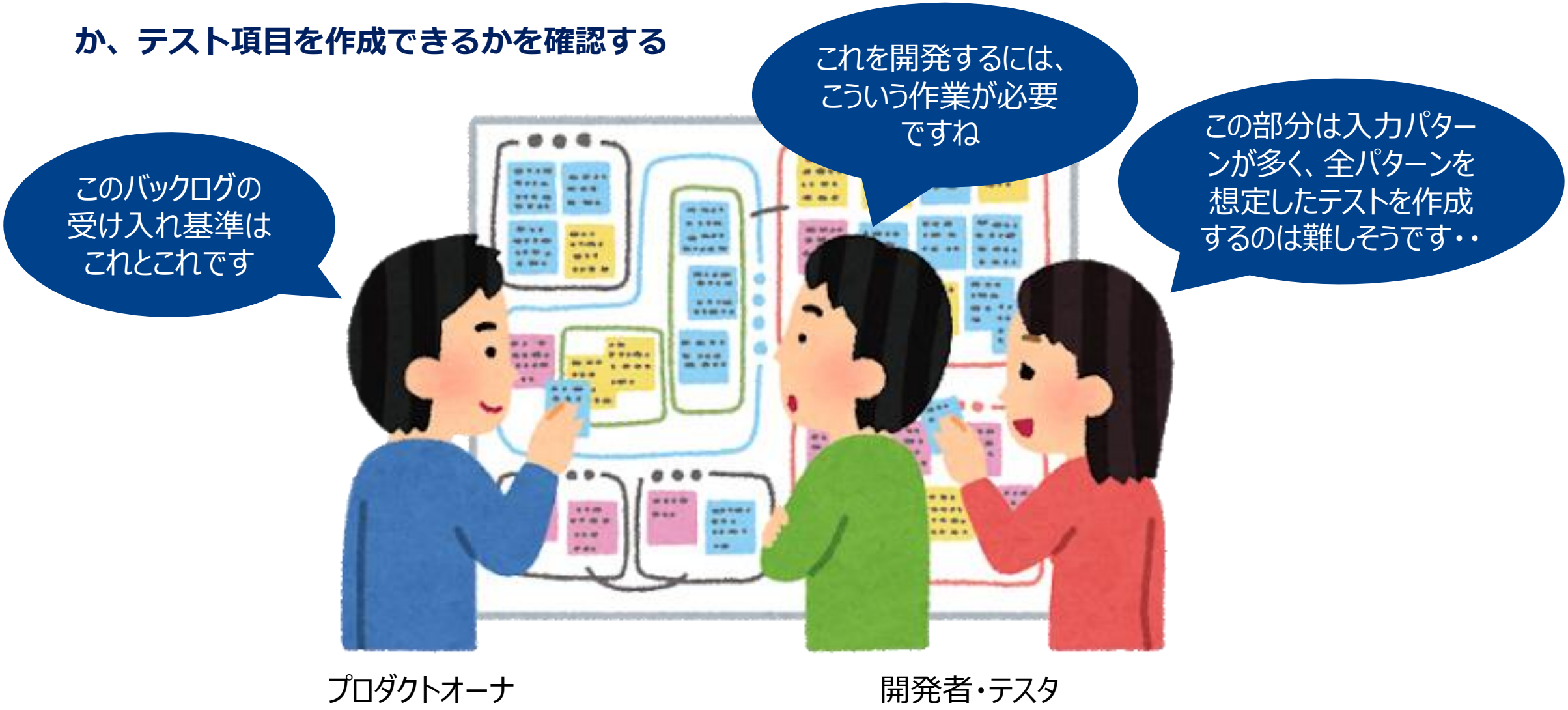
アジャイル開発における品質チェック方法

- バックログにあいまいさや不明な点がないかを確認する。バックログから開発のタスクを分解できるか、テスト項目を作成できるかを確認する
- ストーリポイントの見積もりを行い、チームメンバ間で作業内容に認識のずれがないかを確認する

品質チェック方法の具体例③

VERISERVE

- **バックログにあいまいさや不明な点がないか確認する。バックログから開発のタスクを分解できるか、テスト項目を作成できるかを確認する**



品質チェック方法の具体例③

VERISERVE

- **ストーリポイントの見積もりを行い、チームメンバ間で作業内容に認識のずれがないかを確認する**

ウサギさんが出した
ストーリポイントは
小さいですね

これとこれだけやれば
よいのかなと思ったので、
このポイントにしました

あれ・・・、
こういうこともやる
必要があるのでは？

え・・、それも必要なの？
バックログで明確になって
いなかったような・・

開発者・テスタ

品質チェック方法の具体例③　検出可能な欠陥

VERISERVE

- **欠陥の内容：画面を切り替えた際の表示の更新が遅い（画面を切り替えたタイミングで最新の情報を表示してほしかったが、ワンクリックしないと表示が更新されなかった）**
  - 受け入れ基準で表示更新のタイミングを明確にしていれば、未然に検出できた

～あるチームのオンライン会議にて～

考察

- **ウォータフォール型開発での品質チェック方法をアジャイル開発に応用することで、アジャイル開発における欠陥検出のフロントローディングが可能になる**
  - 顧客に早く価値あるプロダクトを届けることにつながる

- **ウォータフォール型開発とアジャイル開発の違いを考慮することで、従来型の品質チェック方法をアジャイル開発へ応用可能となった**
  - 違い① 全部が決まっていなくても開発が進み、進むにつれて開発するものや、やり方が変化していく
    - チーム全員で共通認識を作りながら開発を進める必要がある
    - 統一性がなくなったり、不要な部分が出てきたりするので、それに応じた品質チェックが必要
  - 違い② チェックするモノが散らばっている
    - ウォータフォールでは要求仕様書などのドキュメントに全部仕様が書いてある（はず）だが、アジャイルでは会話やチャットや付箋のメモなどを対象にチェックしていく必要がある

- **今回提案した品質チェック方法によって、過去のアジャイル開発で検出された欠陥を検出できることが分かり、有効性を確認できた**
  - ただし、このチェック方法「だけ」を実施すれば良い、というものではない
  - チームにとってベストなフロントローディングの方法を考えた上で漏れがないかを本チェック方法で確認する

まとめ・今後の取り組み

➢ まとめ

- アジャイル開発における欠陥検出のフロントローディングの具体的な方法を提案した
  - ウォータフォール型開発の知見を応用して、先行研究で提案された品質チェック項目を具体化できた
- 提案したチェック方法を過去プロジェクトに適用し、欠陥を検出可能であることを確認した
  - 実装着手前に欠陥を検出できることを確認できた

➢ 今後の取り組み

- 各チェック方法に対して、具体的にチェックを行うシーンやその際の質問内容などの具体化
- 実際のプロジェクトに適用して、欠陥検出のフロントローディングの効果の検証
- 実際のプロジェクトでのテーラリング事例の収集や、チームの規模に応じた適用方法の検討

付録：アジャイル開発における品質チェック方法（全体）

VERISERVE

品質チェック項目	ウォータフォール型開発で品質を担保していた方法	アジャイル開発における品質チェック方法
Q1: 顧客が解決したい課題は何か	要求分析で、開発の目的や背景を要求仕様書に記載する	開発の目的や背景をチーム全員が理解しているか確認する。前回のイテレーションでの製品リリース後の顧客からのフィードバックによって目的や背景に変化がないか確認する。変化している場合は、目的や背景と矛盾するバックログがないか、不要になったバックログがないか確認する。
Q2: 誰のためのものか	要求分析で、ターゲットユーザを要求仕様書に記載する、ユースケースを定義する	バックログにターゲットユーザが記載されているかチェックする。過去のイテレーションで登場しなかった対象ユーザの場合、どのような役割の人かチーム全員が理解しているか確認する。過去のイテレーションでターゲットとしたユーザに対して、今回のバックログに基づき開発する部分が悪影響を及ぼさないか確認する。
Q3: 作りたいものは何か	要求分析で、開発する対象が備えるべき機能や非機能の要求・要件を、要求仕様書に記載する	開発する対象が備えるべき機能や非機能の要求・要件がバックログや会話などで決まっているか確認する。今回開発する機能が、過去のイテレーションで開発した機能と矛盾しないか、不整合がないか確認する。
Q4: なぜ必要なのか	要求分析で、開発の目的や背景を要求仕様書に記載する	今回開発するものが、過去に開発したものや今後開発するものを含めて、ターゲットユーザの課題解決につながるものになっているか、ターゲットユーザに価値を提供できるものになっているか確認する。ターゲットユーザに提供できる価値についてチームで合意が取れているか確認する。
Q5: 仕様同士の関係性は適切か	要求仕様書のレビューで、仕様書間での整合性や矛盾がないかをチェックする	過去のイテレーションで開発した部分の仕様と、今回のバックログや将来のバックログの内容が矛盾しないか、不整合がないか確認する。今回のバックログの内容を実装した場合に、既存の機能に悪影響が出ないか、影響範囲に対するテストが膨大になりすぎないか確認する。
Q6: 実際にエンドユーザが使っていくことができるか	要求分析でユースケースを定義する	過去に開発済みの機能を含めたユースケースが想定されているか確認する。既存機能のユースケースに今回開発する機能のユースケースが追加されたときにユースケースが破綻しないか・不整合がないか確認する。
Q7: デザインや操作感に統一性があるか	画面間やユースケース間で統一性があるか、という視点で要求仕様書をレビューする	開発済みのプロダクトと今回開発する内容を比較し、ボタン配置等の画面レイアウトの統一性やUIの文言の統一性を確認する
Q8: 仕様が複雑すぎないか	FP法などで全体の開発工数を見積ることで、複雑すぎる部分（FPが大きすぎる部分）がないか確認する	開発する内容に対して、テストする際の条件やその組み合わせが多すぎないか確認する。バックログのストーリポイントを見積もる際に、テストの作業量も考慮して見積もり、ポイントが大きくなりすぎないか確認する。
Q9: 受け入れ基準が適切に定義されているか	要求仕様書にあいまいさや不明な点がないかをレビューでチェックする	バックログにあいまいさや不明な点がないか確認する。バックログから開発のタスクを分解できるか、テスト項目を作成できるか確認する。ストーリポイントの見積もりを行い、チームメンバ間で作業内容に認識のずれがないか確認する。
Q10: 技術的な制約が考慮されているか	開発するものがすべて要求仕様書に定義されたあとで、要求仕様を実現可能なアーキテクチャの設計や言語・ライブラリ等の選定を行う	将来開発する可能性が高いバックログを見据えたアーキテクチャになっているか、将来開発する可能性が高いバックログを見据えて言語やライブラリ等が選定されているか確認する。過去に開発した部分で使っているライブラリを引き続き利用する場合、問題が起きやすい部分を考慮した仕様や設計になっているか確認する。すでに使っているライブラリを利用して開発が難しい場合、代替案があるか確認する。

ご清聴ありがとうございました。

本発表に関するお問い合わせは下記までご連絡ください。

株式会社ベリサーブ　ベリサーブ　検索

研究企画開発部　谷﨑 浩一

<kouichi.tanizaki@veriserve.co.jp>

TEL：03-6629-8540

東京都千代田区神田三崎町3-1-16 神保町北東急ビル 9F