品質を創造する
VERISERVE

JaSST'21 Tokyo

テクノロジーセッション

品質創造企業のアジャイル開発におけるQAの実践

2021年3月16日

株式会社ベリサーブ　谷﨑 浩一

はじめに

- **この発表では、以下について話します**
  - テスト技法ツール「GIHOZ」の開発プロジェクトで実践しているQA活動
  - 具体的な取り組みとしての、アジャイル開発向けの品質チェック項目とテスト技法を活用した品質改善

- **アジャイル開発のプロジェクトでQA活動やテストを行っている皆さまの参考になったり、自分たちも取り入れてみよう、と思ってもらえると嬉しいです**

目次

VERISERVE

- 自己紹介
- 会社紹介
- GIHOZプロジェクトの概要
- GIHOZプロジェクトのQA活動
- 取り組み①：アジャイル開発向けの品質チェック項目の活用
- 取り組み②：テスト技法を活用したテスト設計
- まとめ

自己紹介

VERISERVE

- **名前**
  - 谷﨑　浩一（たにざき　こういち）
- **所属**
  - 株式会社ベリサーブ　研究企画開発部
- **経歴**
  - テスター歴：12年くらい
    - プリンタ、デジカメ、PCアプリ、クラウドサービス
    - 統合テスト・システムテストのテスト設計・実行・管理
  - プロダクトオーナー歴：4年くらい
    - TESTRUCTURE、GIHOZ
- **その他の活動**
  - テスト設計コンテスト’15準優勝
    - チーム名：TEVASAKIplus
  - 東京工業大学　博士後期課程　2021年3月修了見込み
    - 研究テーマ：悪条件分析に基づく仕様レビュー技法

GIHOZ

TESTRUCTURE

会社紹介

品質向上にゴールはない。
だからベリサーブはこれから先も、
「品質とは何か」を問い続ける。

進化を続ける時代に合わせて、
常に最先端の技術を取り入れ、
新しい検証の形を作り、
新時代の品質に対応していく。

それこそが、ベリサーブの目指す「品質創造」。

我々は「品質創造企業」です。

➢ GIHOZプロジェクトの概要

GIHOZとは

VERISERVE

GIHOZは、各種テスト技法を手軽に利用できる
クラウド型ツールです。

「テスト技法」は、テストケースを作成したり選択したりするための
技法です。様々なテスト技法が存在します。

コンセプト

"今すぐ使える"テスト技法ツール

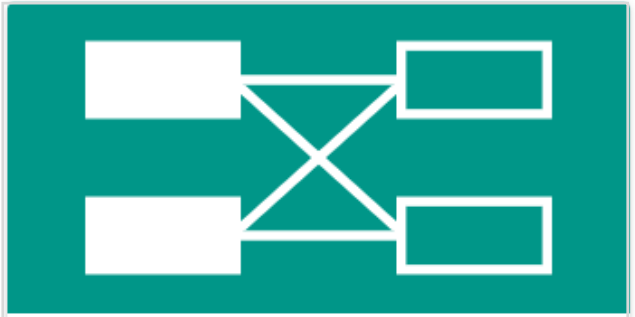
1. 手軽に正確に
テストを作成

2. 目的に応じて
テスト技法を選択

3. 高品質なソフトウェアの
開発に貢献

現場でよく使うテスト技法をサポート

VERISERVE



ペアワイズテスト

パラメータと制約条件から、パラメータのペアを網羅する組み合わせを自動生成します。組み合わせの件数を抑えたい場合に有効です。

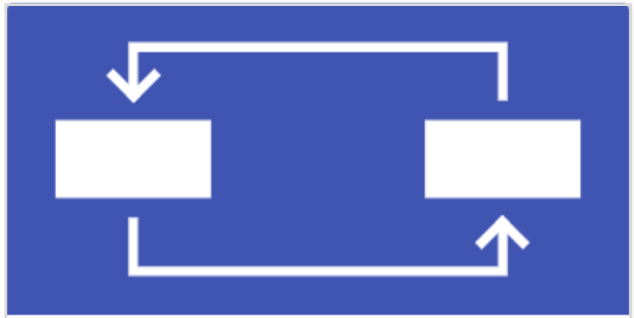
デシジョンテーブルテスト

デシジョンテーブルを使い、テストの入力条件と対応する出力結果を整理できます。複雑な論理関係を整理したい場合に有効です。

テストケースを自動生成

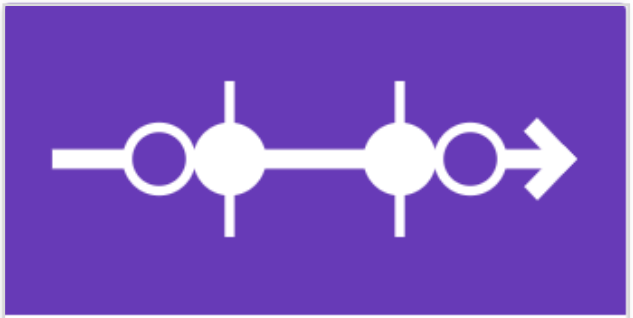
図を用いて直感的に分析

表計算ソフト感覚で入力



状態遷移テスト

状態遷移図から、状態遷移を網羅するテストケースを自動生成します。状態の変化に着目してテストしたい場合に有効です。



境界値分析

連続する値の境界を分析しテストケースを作成します。境界値にはバグが潜んでいる可能性が高いとされます。

プロジェクト体制

Product Owner

プロダクト
オーナー

Developers

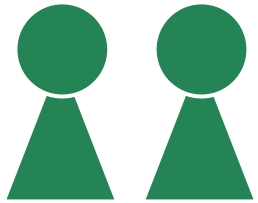
開発者

デザイナー

テスター

Scrum Master

Reviewers

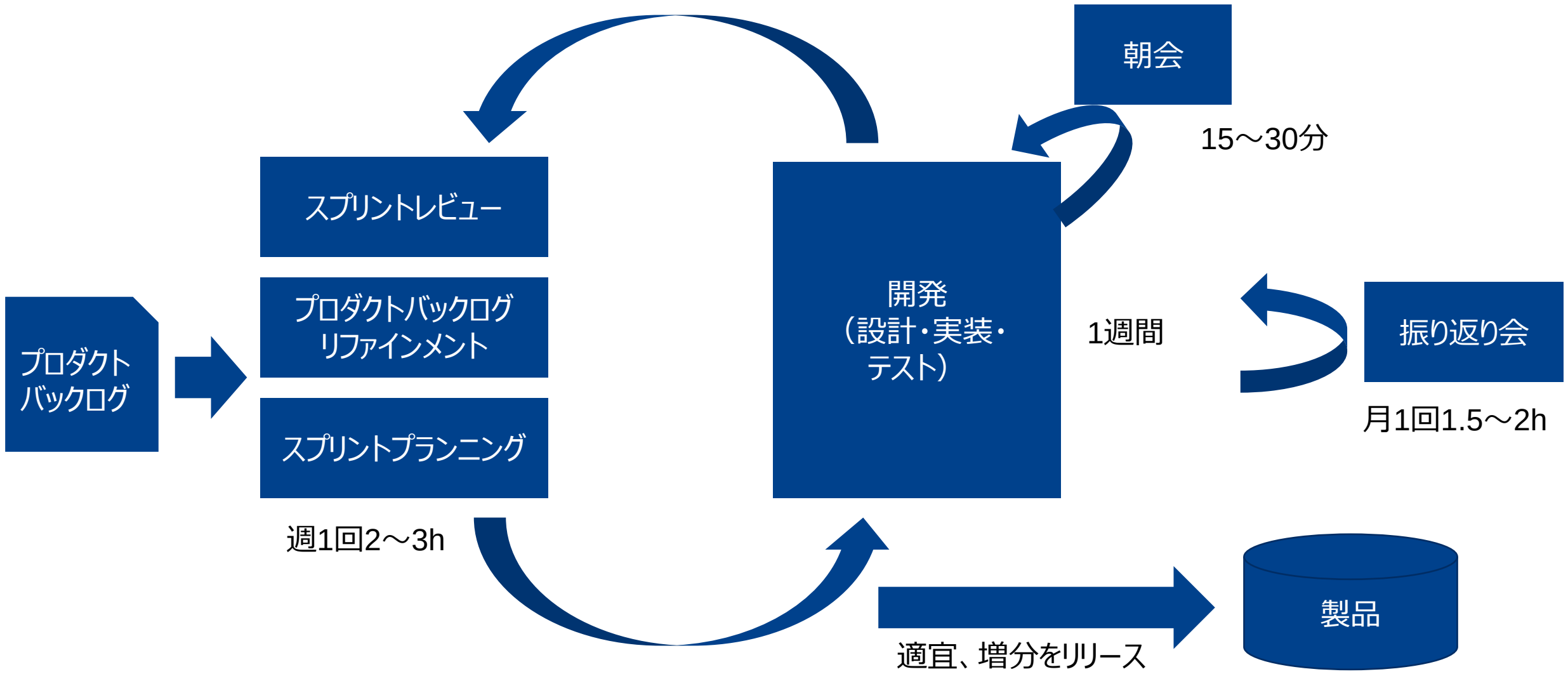


画面デザインの
レビュー、コードの
レビューなど

VERISERVE × StartupTechnology

開発の進め方

VERISERVE



使用している技術・ツール

VERISERVE

分類	概要	ツール
開発・運用	開発フレームワーク	Vue.js, Nuxt.js, Rails
	ソースコード管理	GitHub
	CI	CircleCI
	QA自動化プラットフォーム	Autify
	コンテナプラットフォーム	Docker
	デプロイ・運用	Amazon Web Services
	デザインツール	Figma
プロジェクト管理・ユーザサポート	バックログやスプリントの管理	Jira
	ドキュメント管理	Confluence
	ユーザサポート	Jira Service Management
コミュニケーション	チャット	Slack
	ビデオ会議	Zoom
	ホワイトボード	Miro

プロダクトバックログの管理

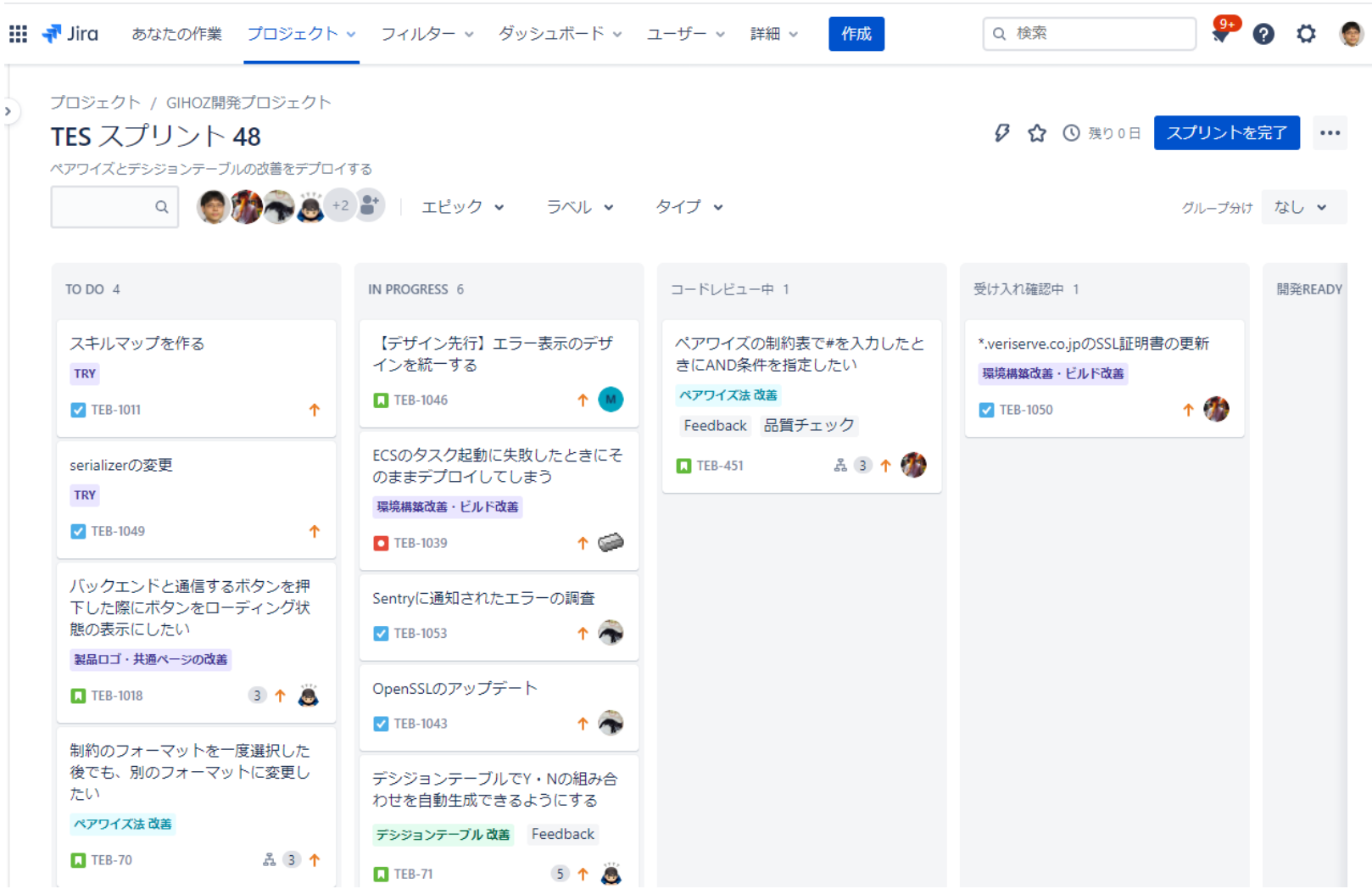
VERISERVE

- Jiraでバックログとその優先順位を一覧として管理

スプリントバックログの管理

VERISERVE

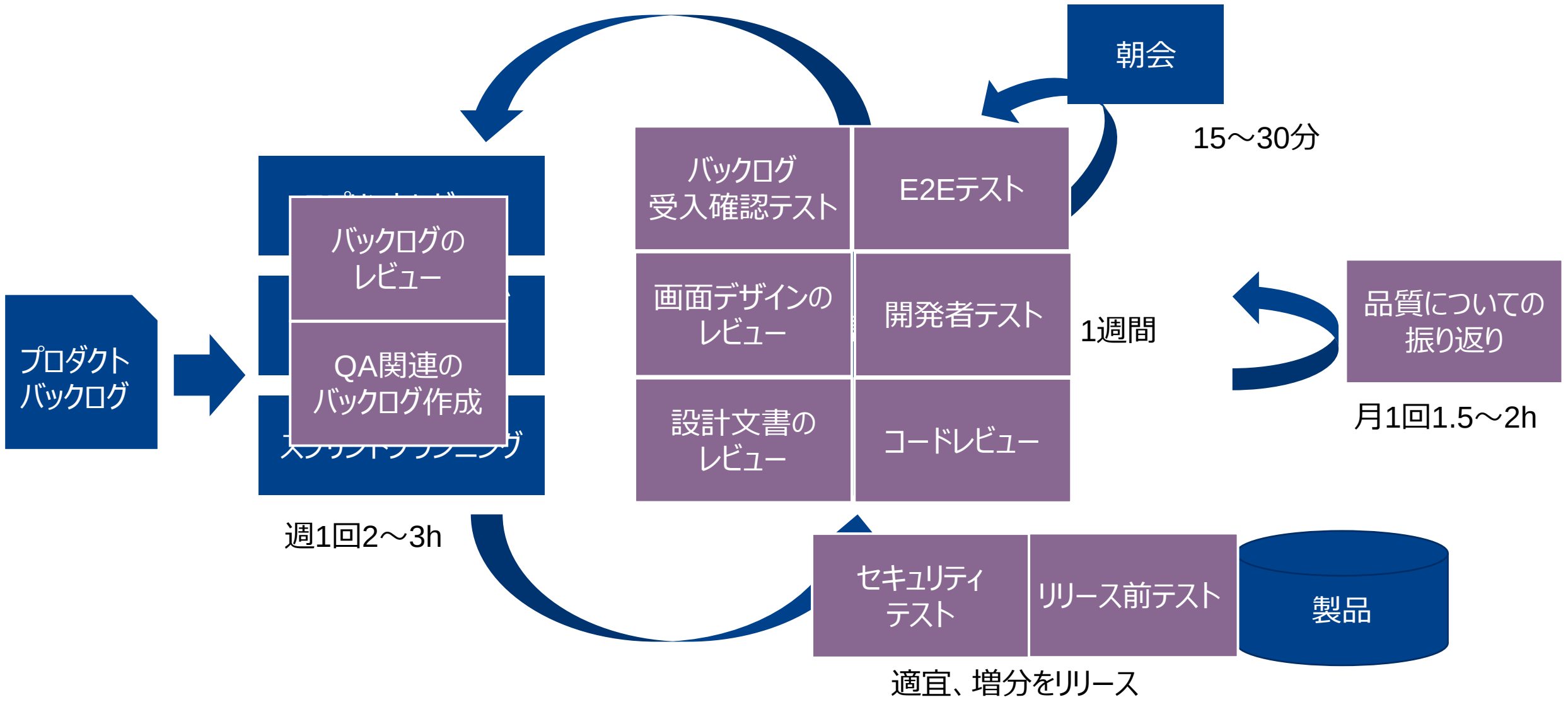
- Jiraのスプリントボードで日々の開発状況を可視化



➢ GIHOZプロジェクトのQA活動

GIHOZプロジェクトのQA活動

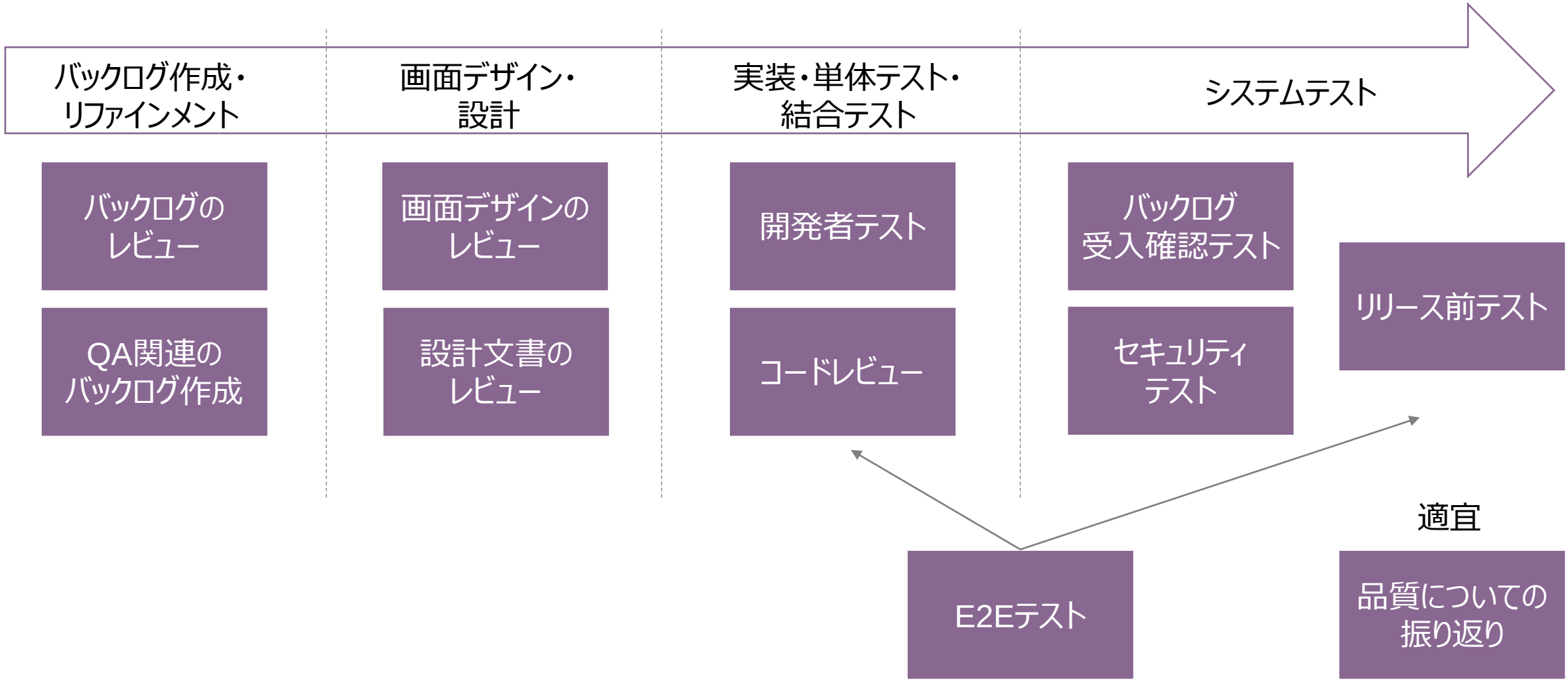
VERISERVE



QA活動のおおまかな順序

VERISERVE

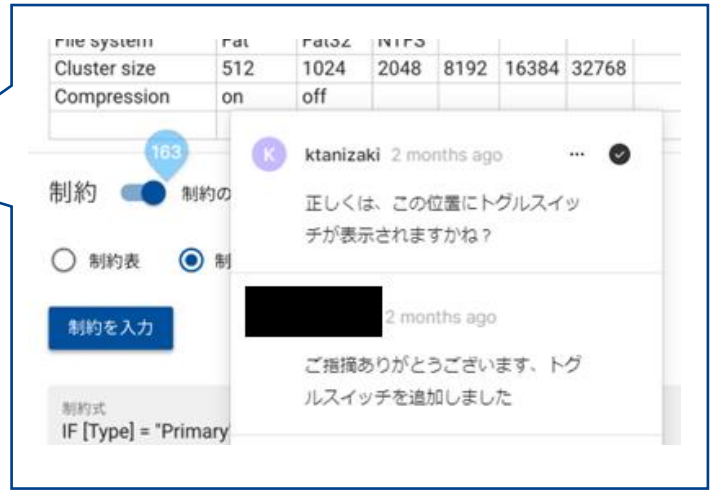
- 行ったり来たりもするが、おおまかには以下の順序で実施している



レビュー

VERISERVE

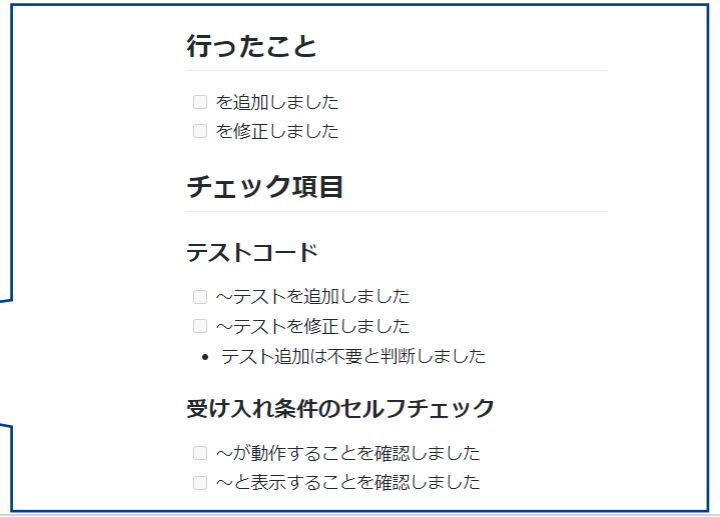
- **バックログのレビュー（PO、全員が実施）**
  - アジャイル開発向けの品質チェック項目（後述）を活用したセルフチェックやレビュー
  - チーム全員で内容を共有して議論



- **画面デザインのレビュー（主にデザイナー、POが実施）**
  - Figmaでデザイン上に直接コメントを書き込み
  - デザインのタスクの受け入れ確認をPOが実施

- **設計文書のレビュー（主に開発者、POが実施）**
  - ER図、API設計書などをチームメンバーで共有して議論
  - 文書は最終的にConfluenceで管理

- **コードレビュー（主に開発者が実施）**
  - GitHubのプルリクエストを活用



テスト

VERISERVE

- **開発者テスト（開発者が実施）**
  - 手元のPC環境での動作確認、コードの静的解析、単体テストのテストコードの実装
  - CIツールで既存のテストコードを全実行

- **バックログ受入確認テスト（主にPO、テスターが実施）**
  - 開発用サーバや手元のPC環境で、機能テスト・システムテストに該当する内容を手動で確認
  - テスト設計の際にGIHOZでテスト技法を活用して効率化

- **セキュリティテスト（チーム外の専門家が実施）**
  - バックログにテストのタスクを含めておく
  - 適宜、専門家に依頼して実施

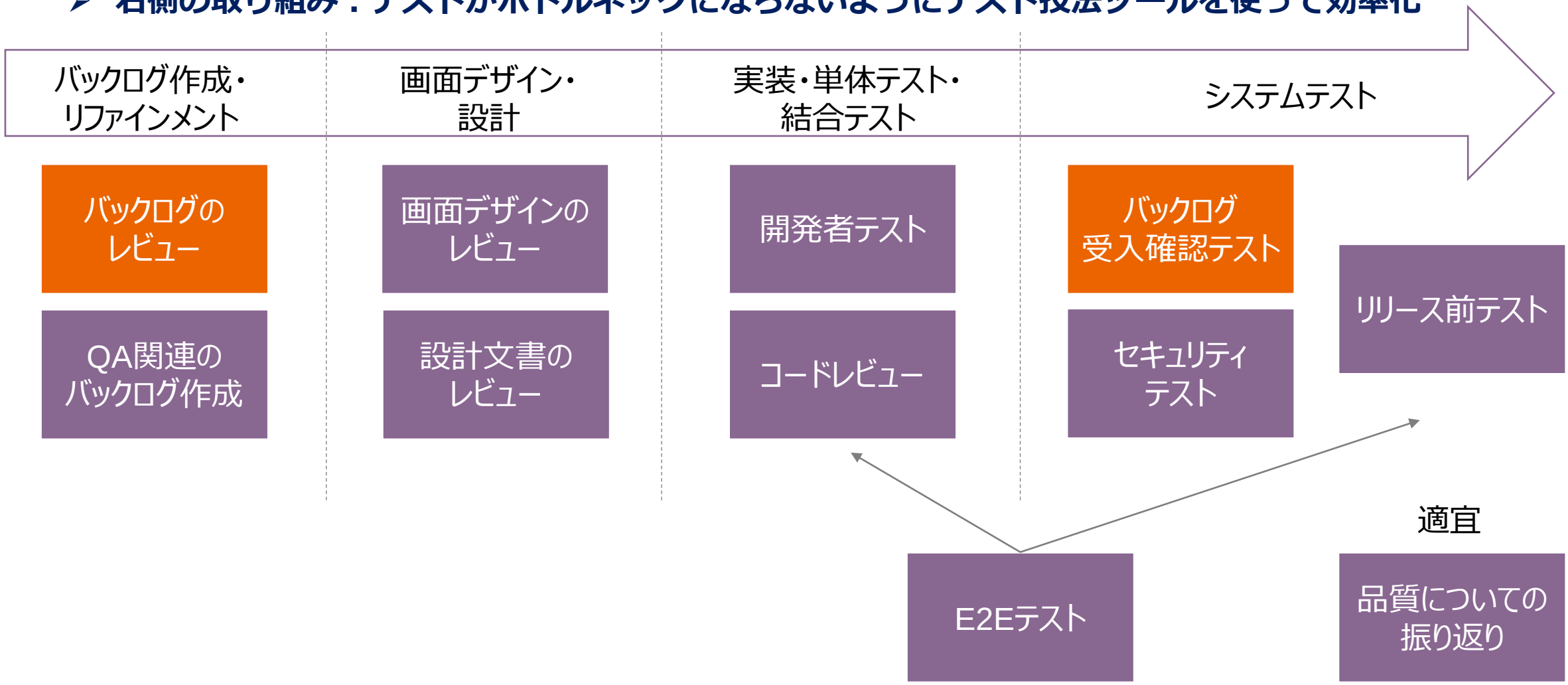
- **リリース前テスト（主にテスターが実施）**
  - ステージングサーバで、一通りの機能を実行して問題ないかを手動で確認

- **E2Eテスト（主にテスターが実施）**
  - Autifyを活用してシナリオを作成し自動で実行（開発用サーバ、ステージングサーバ、本番サーバで実行予定）

本日紹介する取り組み

VERISERVE

- 左側の取り組み：欠陥を作りこまないように品質チェック項目を使って問題を早期検出
- 右側の取り組み：テストがボトルネックにならないようにテスト技法ツールを使って効率化

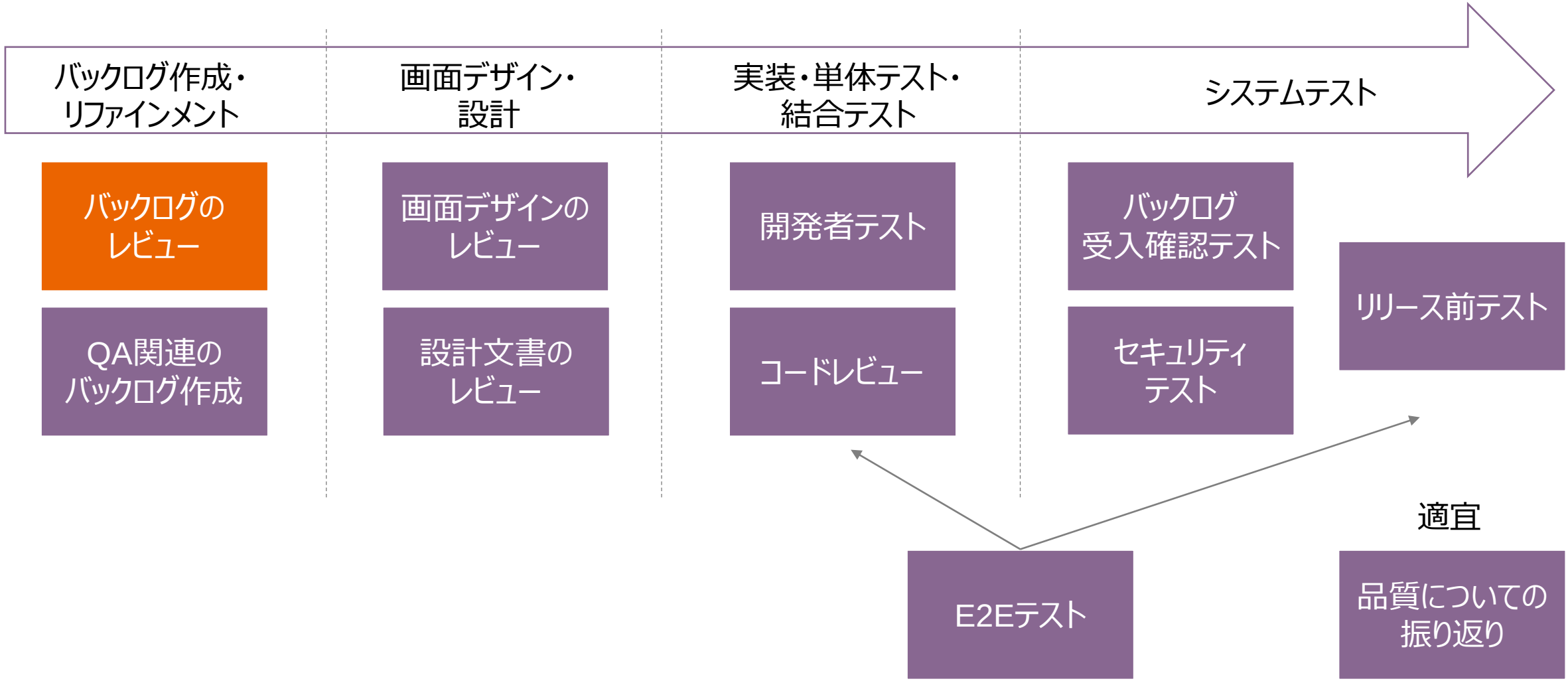


➢ 取り組み①：アジャイル開発向けの品質チェック項目の活用

アジャイル開発向けの品質チェック項目の活用

VERISERVE

- **左側の取り組み：欠陥を作りこまないように品質チェック項目を使って問題を早期検出**



アジャイル開発向けの品質チェック項目とは

VERISERVE

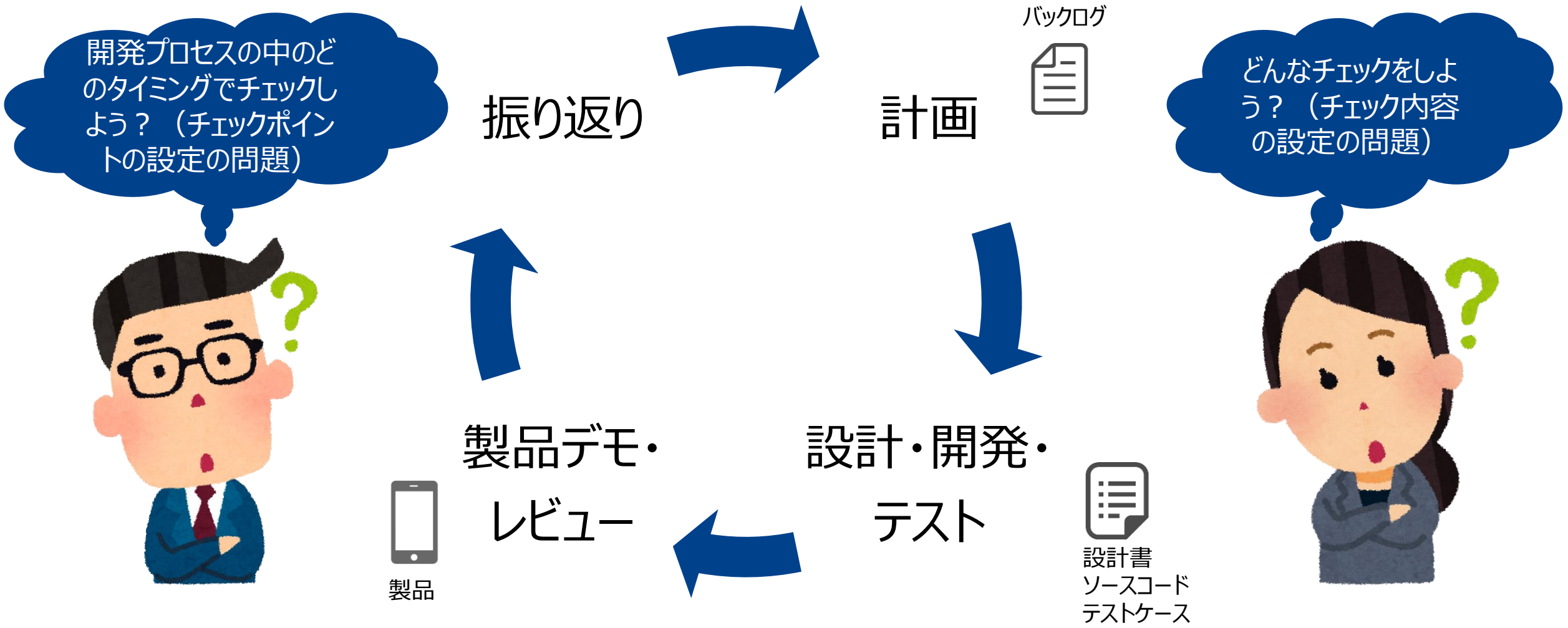
- **アジャイル開発のプラクティスとステークホルダの視点に基づいて、品質チェック項目を作成する手法が提案され、効果が示されている**

- **2つのシンポジウムで発表済み**
  - 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，88-96，2020.
  - 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報文集，B1-1 ，2020.

品質チェック項目の作成において考慮すべき点

VERISERVE

- チェックポイントの設定の問題と，チェック内容の設定の問題がある



アジャイル開発のプラクティスとステークホルダに着目

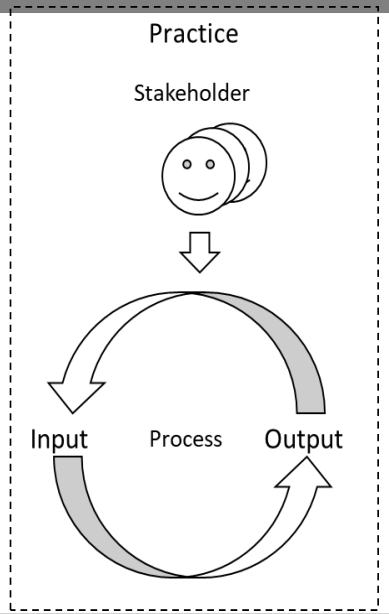
VERISERVE

チェックポイントの設定の問題

チェック内容の設定の問題

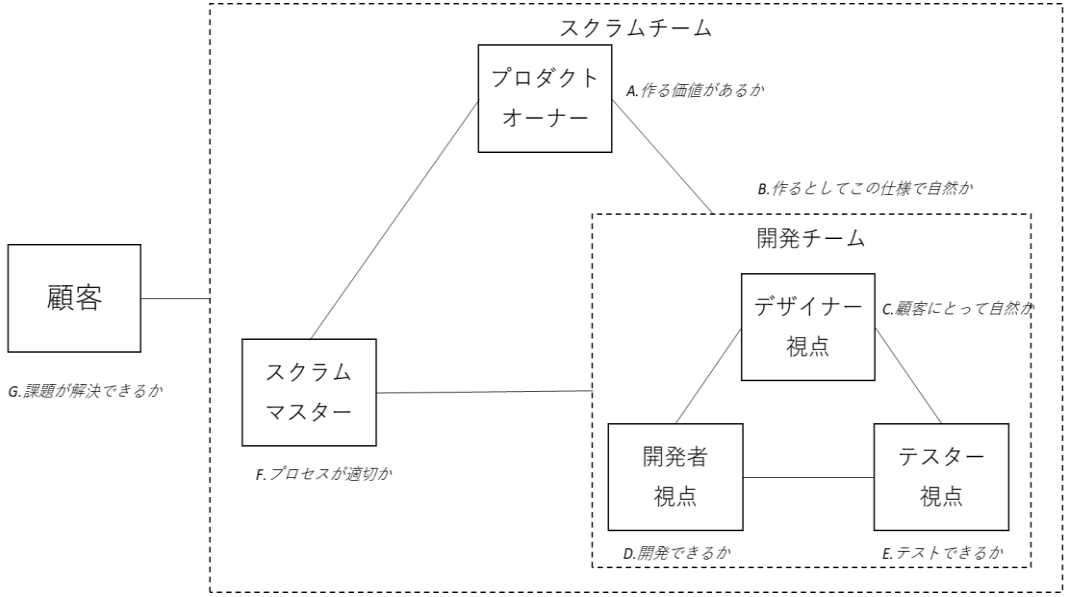
アジャイル開発の「プラクティス」に着目

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



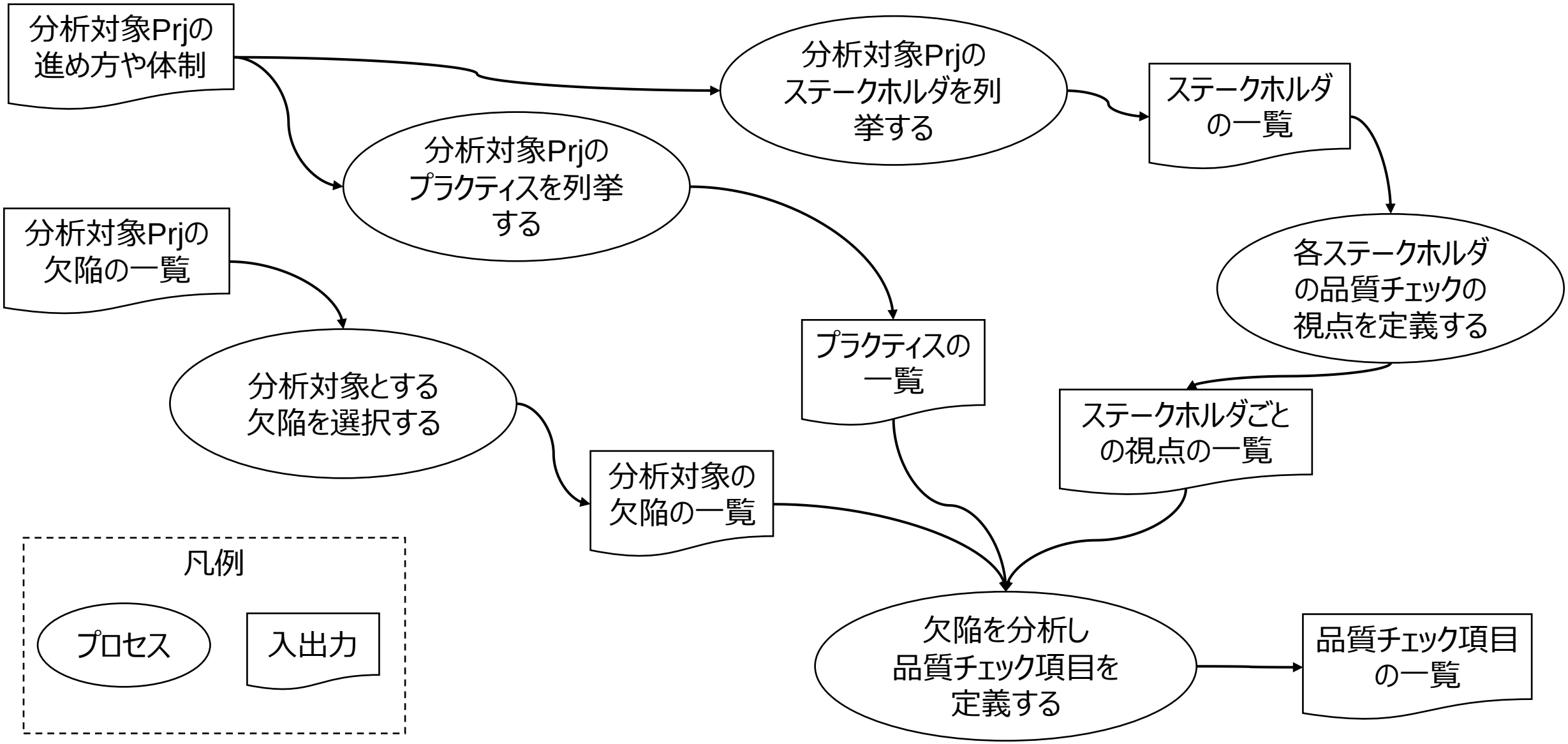
アジャイル開発の「ステークホルダ」に着目

ステークホルダごとに様々な
品質チェックの視点がある



品質チェック項目の作成手順

VERISERVE



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

アジャイル開発向けの品質チェック項目

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

GIHOZプロジェクトでの品質チェック項目の活用

VERISERVE

活用方法A

プロダクトオーナーによるバックログのセルフチェック

活用方法B

チームメンバーによるバックログのレビュー

活用方法A：プロダクトオーナーによるバックログのセルフチェック

VERISERVE

- バックログアイテムの作成時にチェック項目を意識して作成
- 一部のチェック項目はバックログのテンプレートとして設定

			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
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
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

バックログについてのチームメンバーへのアンケート結果

VERISERVE

- 記述量、 Negotiable、Valuable、Testableは適切
- 詳細度、Independent、Small、Estimableは改善の余地あり

デザインや開発に着手する際のバックログの記述量は適切だと思いますか？

詳細

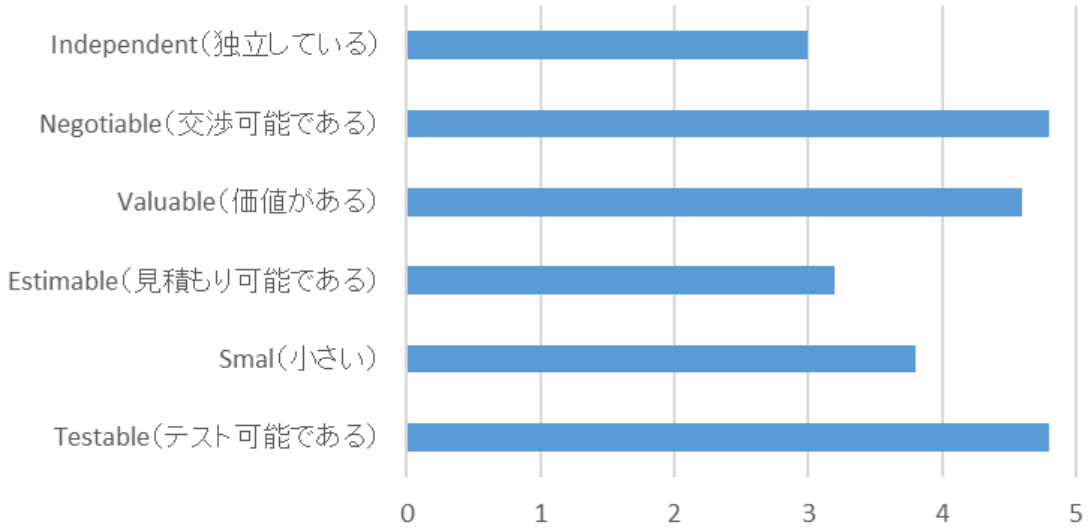
多すぎる	0
やや多い	0
適切	5
やや少ない	0
少なすぎる	0

デザインや開発に着手する際のバックログの詳細度は適切だと思いますか？

詳細

詳細すぎる	0
やや詳細	1
適切	1
やや曖昧	3
曖昧すぎる	0

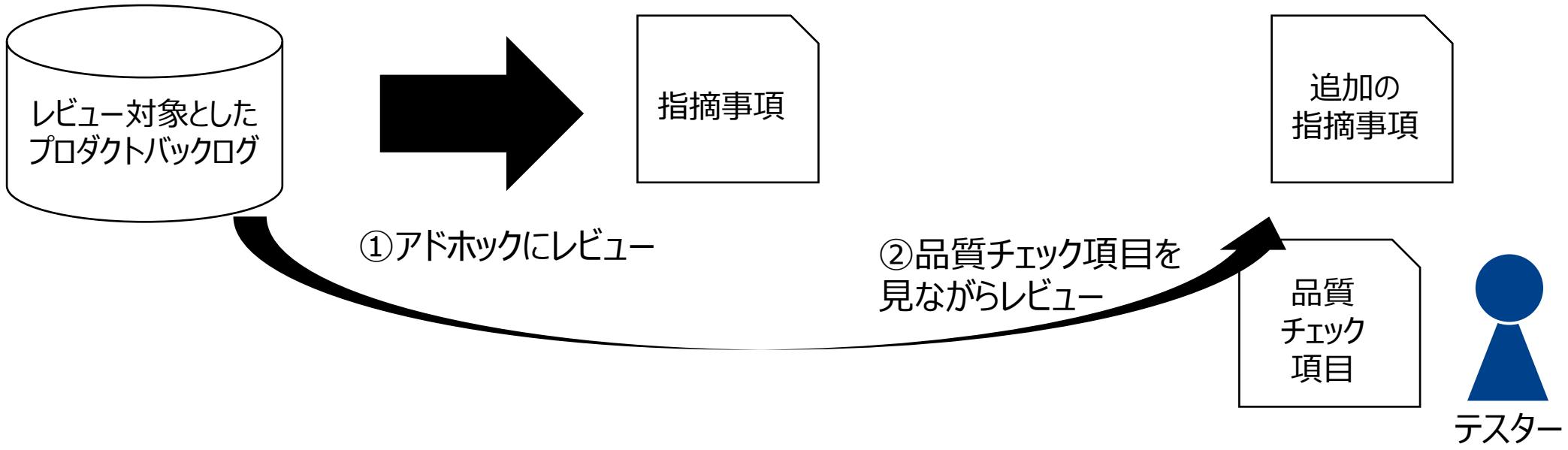
バックログがINVESTをどの程度満たしているか



活用方法B：チームメンバーによるバックログのレビュー

VERISERVE

- **実施者：テスター1名**
- **実施手順：**
  - ①アドホックにプロダクトバックログをレビューしてもらい、指摘事項（気になった点や問題点）を洗い出してもらう
  - ②品質チェック項目を渡して、同じプロダクトバックログをレビューしてもらい、追加で指摘事項を洗い出してもらう



レビュー対象としたプロダクトバックログアイテム

VERISERVE

➢ 対象としたプロダクトバックログアイテム

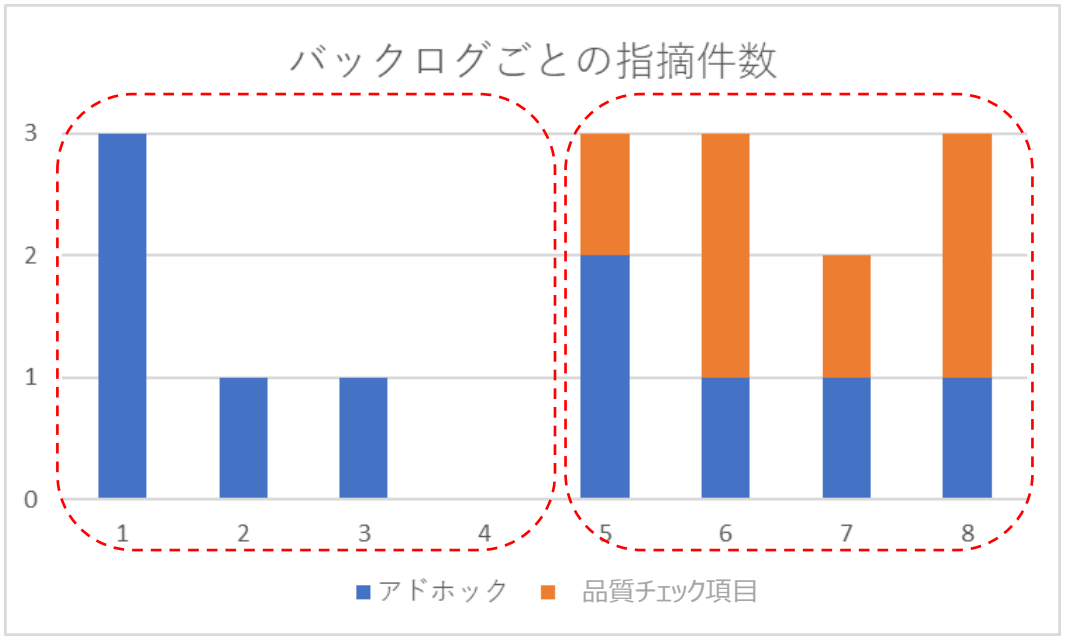
- プロダクトバックログの中で優先順が高めなもの（＝近い将来に開発予定のもの）を選定
- 上のほうのバックログは記述内容が詳細、下にいくほど記述内容が少なかったりあいまい

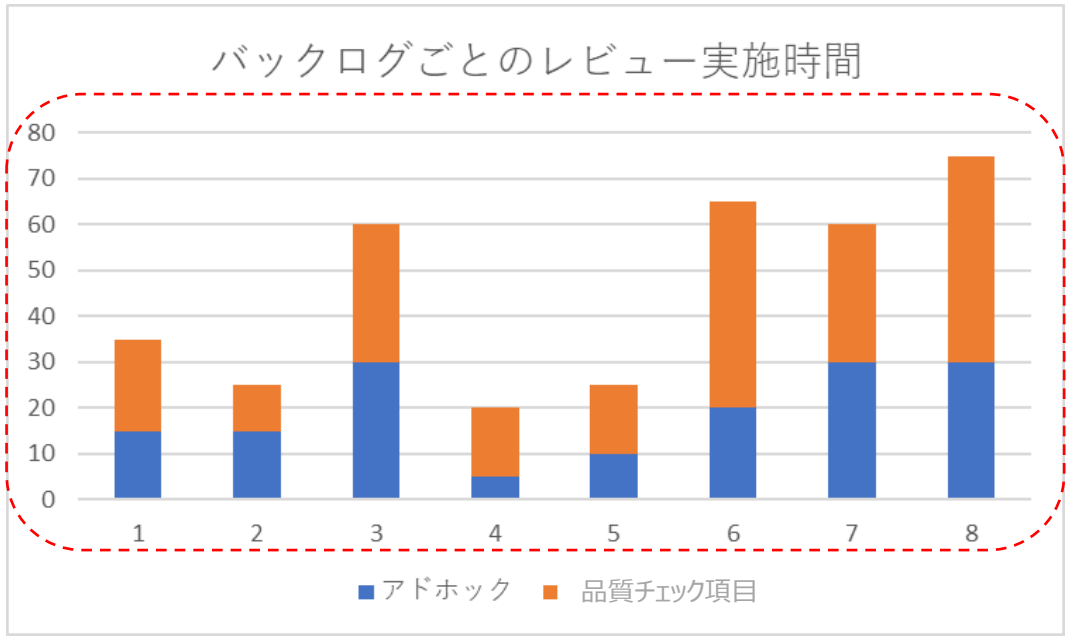
No.	プロダクトバックログアイテムの概要
1	アカウント作成ページで利用規約の確認と同意を取る
2	登録したメールアドレスを変更する
3	タイムアウトしたときに編集中のデータを維持する
4	入力データの形式を著名なツールと同じ形式で入力できる
5	（非公開）
6	（非公開）
7	（非公開）
8	（非公開）

レビュー結果

VERISERVE

- No.1～4（直近で開発予定）は、品質チェック項目を使っても追加の指摘はなかった
- No.5～8（将来開発予定）は、品質チェック項目を使うことで追加の指摘があった
- 品質チェック項目を使った場合、時間はアドホックレビューの1.5倍程度かかった

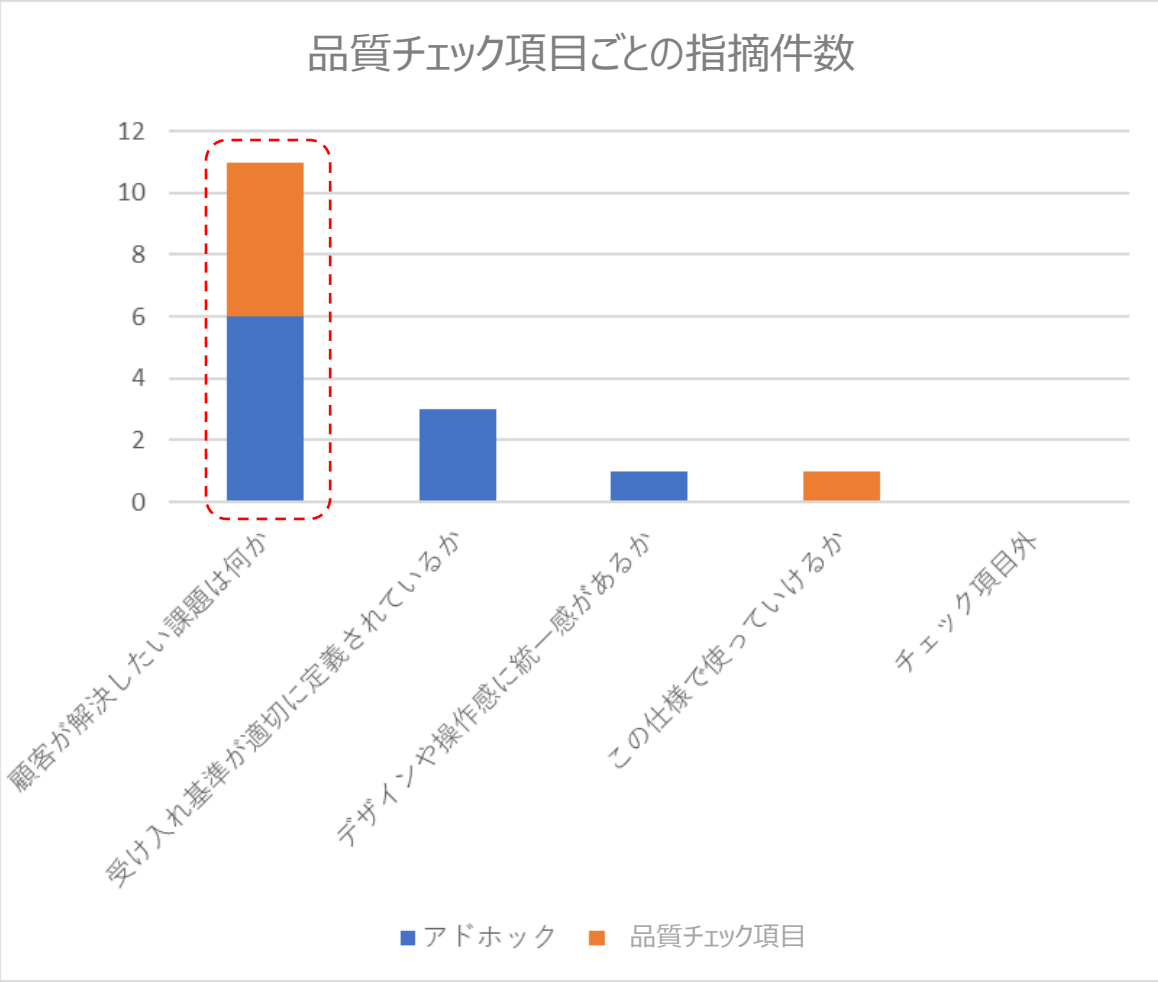




レビュー結果

VERISERVE

- 顧客視点での指摘が多く、顧客視点でのバックログの改善に繋がった
- アドホックレビューによる指摘事項も品質チェック項目でも見つけられるものだった



取り組み①まとめ

VERISERVE

- **活用方法A：POはバックログのセルフチェックを実施し、バックログの質を高めることができる**
  - テンプレート化すると、バックログ作成時に常に意識できる

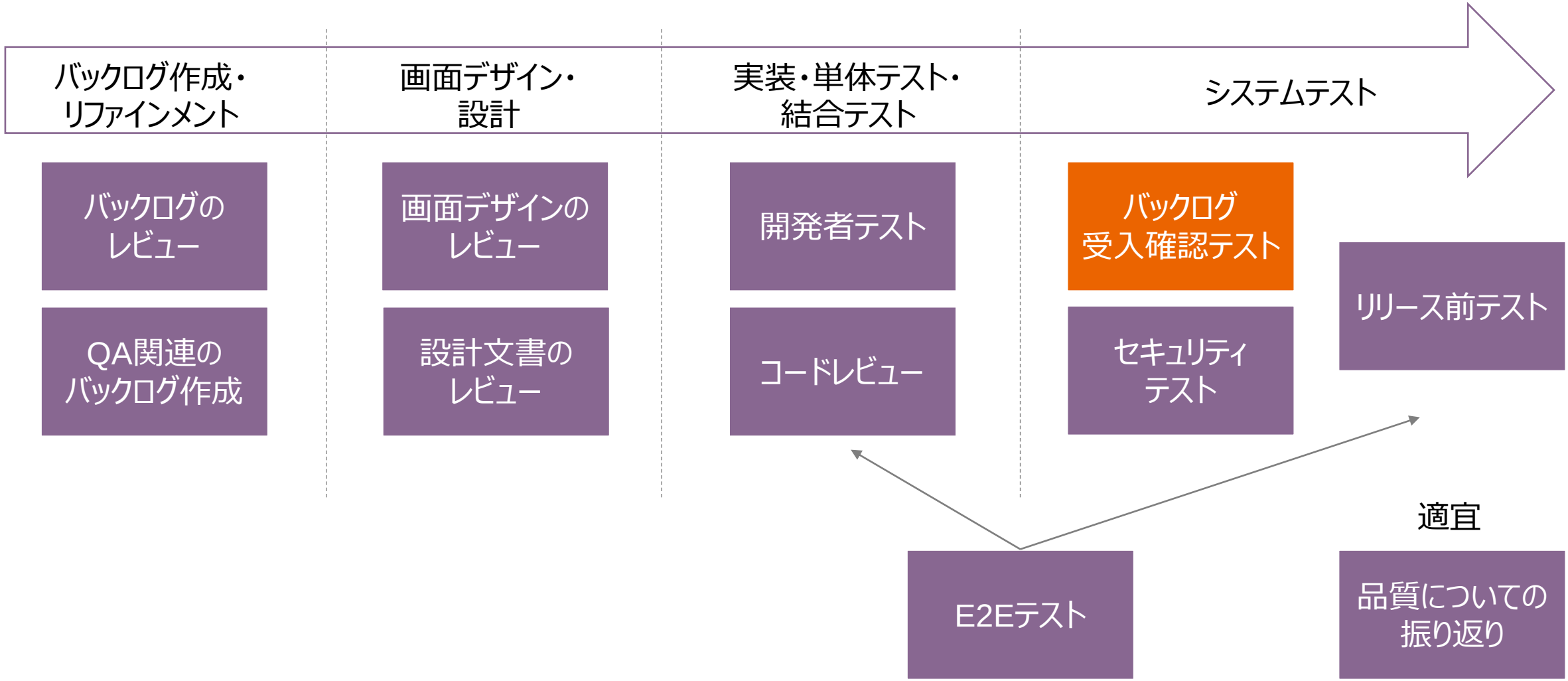
- **活用方法B：チームメンバーはバックログのレビューを実施し、POの考慮不足を早期に指摘できる**
  - 顧客視点での指摘を早期検出できることは特に有用

➢ 取り組み②：テスト技法を活用したテスト設計

テスト技法の活用

VERISERVE

- **右側の取り組み：テストがボトルネックにならないようにテスト技法ツールを使って効率化**



テスト技法（テスト設計技法）とは

VERISERVE

- **テストケースを適切に選択したり作成するための技法**

- **様々なテスト技法が存在**
  - ブラックボックステスト技法
    - 同値分割法、境界値分析、デシジョンテーブルテスト、状態遷移テスト、ユースケーステスト　など
  - ホワイトボックステスト技法
    - ステートメントテスト、デシジョンテスト　など
  - 経験ベースのテスト技法
    - エラー推測、探索的テスト、チェックリストベースドテスト　など

- **本日紹介するのはブラックボックステスト技法の一部**

テスト技法を活用したテスト設計の流れ

VERISERVE

- まずテストの内容を検討し、それから、どのテスト技法を使用するか決める

GIHOZを活用したテスト設計

VERISERVE

- **様々なテスト技法を使ってテストモデル・ケース仕様を作成。バックログのIDと紐づけて管理**

デシジョンテーブルとは

VERISERVE

- テストの入力条件と対応する出力結果を整理する技法
- 入力条件の組み合わせの網羅、重要な組み合わせの識別が可能
- GIHOZでは

例：商用施設の駐車場の料金計算に対するテスト

条件記述部に入力後、ワンクリックで条件の組み合わせを自動生成

組み合わせを生成

		有効/無効	☑	☑	☑	☑	☑	☑	☑	☑
			1	2	3	4	5	6	7	8 ⊕
条件 ⊕	3000円以上10000円未満 ⊕		N	Y	N	N	N	Y	N	N
	10000円以上30000円未満 ⊕		N	N	Y	N	N	N	Y	N
	30000円以上 ⊕		N	N	N	Y	N	N	N	Y
	シネマ利用 ⊕		N	N	N	N	Y	Y	Y	Y
動作 ⊕	30分無料		X	-	-	-	-	-	-	-
	1時間無料		-	X	-	-	-	-	-	-
	2時間30分無料		-	-	X	-	X	X	X	-
	3時間30分無料		-	-	-	X	-	-	-	X

不要な組み合わせは☑を外して除外

デシジョンテーブルの活用

VERISERVE

- 一番よく使っている。組み合わせの自動生成が便利
- テスト設計以前に、プロダクトバックログ作成時に仕様を整理する際にも使っている

名前

TEB-71-デシジョンテーブルでY/Nの組み合わせを自動生成する（メッセージ表示のルール）

組み合わせを生成

			有効/無効	☑	☑	☑	☑	☑	☑	☑	☑
				1	2	3	4	5	6	7	8 ⊕
条件 ⊕	Y/N/Xの欄に値が入力されている ⊕			Y	Y	Y	Y	N	N	N	N
	有効/無効のチェックが外れている ⊕			Y	Y	N	N	Y	Y	N	N
	組み合わせ件数 ⊕	64件以下		Y	N	Y	N	Y	N	Y	N
		64件を超える		N	Y	N	Y	N	Y	N	Y
動作 ⊕	メッセージは何も表示しない			-	-	-	-	-	-	X	-
	データが上書きされるが本当に実行するか確認する旨のメッセージを表示			X	-	X	-	X	-	-	-
	件数が多すぎて生成できない旨のメッセージを表示			-	X	-	X	-	X	-	X

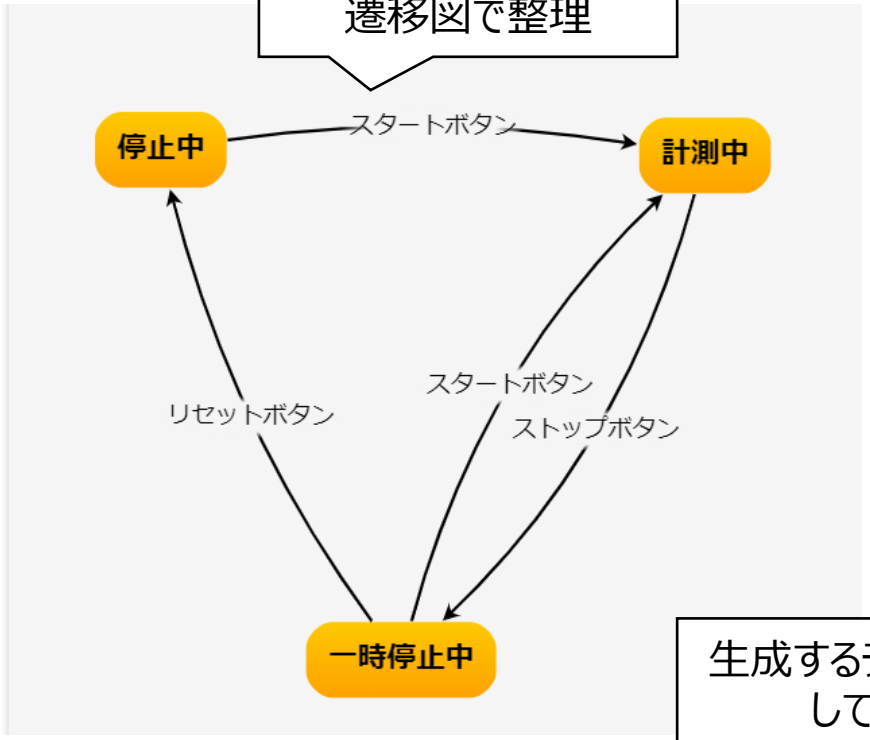
状態遷移テストとは

VERISERVE

- 状態遷移を状態遷移図や表で整理し、状態遷移を網羅するテストケースを作成する技法
- GIHOZでは

例：ストップウォッチの状態遷移テスト

状態とイベントを状態遷移図で整理



自動生成

生成するテストケースを選択して、自動生成

無効遷移を含めた全遷移テストケース

CSVダウンロード

	開始状態	入力	終了状態
1	停止中	スタートボタン	計測中
2	停止中	ストップボタン	停止中
3	停止中	リセットボタン	停止中
4	計測中	スタートボタン	計測中
5	計測中	ストップボタン	一時停止中
6	計測中	リセットボタン	計測中
7	一時停止中	スタートボタン	計測中
8	一時停止中	ストップボタン	一時停止中
9	一時停止中	リセットボタン	停止中

状態遷移テストの活用

VERISERVE

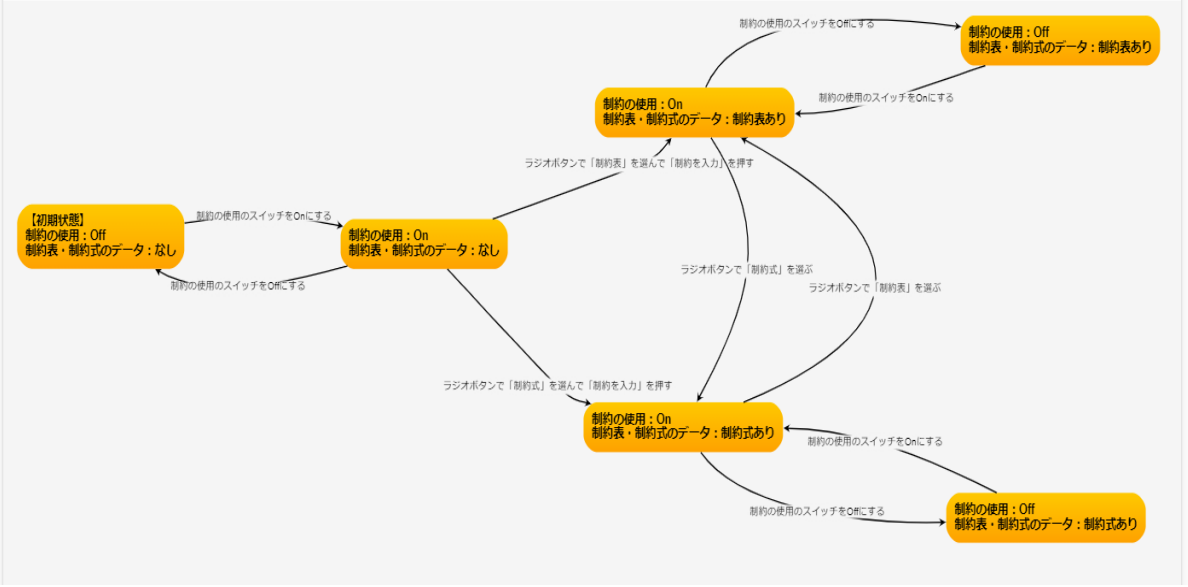
- 状態遷移や画面遷移を整理してテストする際に活用。箱と矢印を繋いだ図を作りたいときに便利
- テスト設計以前に、プロダクトバックログ作成時に仕様を整理する際にも使っている

名前

TEB-70-ペアワイズテストの制約のフォーマットを一度選択した後でも変更したい

データ入力

状態遷移図



0スイッチ テストケース

CSVダウンロード

	開始状態	入力	終了状態
1	【初期状態】 制約の使用：Off 制約表・制約式のデータ：なし	制約の使用のスイッチをOnにする	制約の使用：On 制約表・制約式のデータ：なし
2	制約の使用：On 制約表・制約式のデータ：なし	制約の使用のスイッチをOffにする	【初期状態】 制約の使用：Off 制約表・制約式のデータ：なし
3	制約の使用：On 制約表・制約式のデータ：なし	ラジオボタンで「制約表」を選んで「制約を入力」を押す	制約の使用：On 制約表・制約式のデータ：制約表あり
4	制約の使用：On 制約表・制約式のデータ：なし	ラジオボタンで「制約式」を選んで「制約を入力」を押す	制約の使用：On 制約表・制約式のデータ：制約式あり
5	制約の使用：On 制約表・制約式のデータ：制約表あり	ラジオボタンで「制約式」を選ぶ	制約の使用：On 制約表・制約式のデータ：制約式あり
6	制約の使用：On 制約表・制約式のデータ：制約式あり	ラジオボタンで「制約表」を選ぶ	制約の使用：On 制約表・制約式のデータ：制約表あり
7	制約の使用：On 制約表・制約式のデータ：制約表あり	制約の使用のスイッチをOffにする	制約の使用：Off 制約表・制約式のデータ：制約表あり
8	制約の使用：On 制約表・制約式のデータ：制約式あり	制約の使用のスイッチをOffにする	制約の使用：Off 制約表・制約式のデータ：制約式あり
9	制約の使用：Off 制約表・制約式のデータ：制約式あり	制約の使用のスイッチをOnにする	制約の使用：On 制約表・制約式のデータ：制約式あり
10	制約の使用：Off 制約表・制約式のデータ：制約表あり	制約の使用のスイッチをOnにする	制約の使用：On 制約表・制約式のデータ：制約表あり

ペアワイズテストとは

VERISERVE

- 2つのパラメータ間の組み合わせを網羅するテストケースを作成する技法
- パラメータが3つ以上になる場合でも組み合わせの件数を抑えてテストケースを作成できる
- GIHOZでは

例：OS・ブラウザなどのテスト環境の組み合わせ

パラメータと値

パラメータと値を表形式で整理

パラメータ	値	値	値	値	値
OS	Windows10	Mac OS X	Android		
ブラウザ	Chrome	FireFox	Edge	Safari	
言語	日本語	英語			
解像度	1920×1080	1280×960	800×600	3840×2400	

制約 制約の使用

制約表 制約式

制約を入力

表か式で
組み合わせの制約を
指定

テストケースを生成

組み合わせを自動生成
（PICTと同等）

自動生成

csvダウンロード

	OS	ブラウザ	言語	解像度
1	Android	IE	英語	3840×2400
2	Mac OS X	Edge	日本語	1280×960
3	Windows10	IE	日本語	800×600
4	Windows10	Safari	英語	1280×960
5	Mac OS X	FireFox	英語	800×600
6	Android	FireFox	日本語	3840×2400
7	Android	Chrome	日本語	800×600
8	Windows10	Edge	英語	800×600
9	Windows10	Chrome	英語	1920×1080
10	Mac OS X	Edge	日本語	3840×2400

ペアワイズテストの活用

VERISERVE

- バグ修正の影響範囲が不明確で、条件の組み合わせパターンが多くなるテストで活用
- 簡単にペアワイズ法での組み合わせが作成できて便利

名前
TEB-884-境界値分析で「未満」が連続している場合に削除するとnullになる

パラメータと値

パラメータ	値	値	値	値	値
0	以上	より大きい	以下	未満	
1	以上	より大きい	以下	未満	
削除する境界値	1個目	2個目	3個目	4個目	
最大値・最小値	なし・なし	あり・なし	なし・あり	あり・あり	

CSVダウンロード

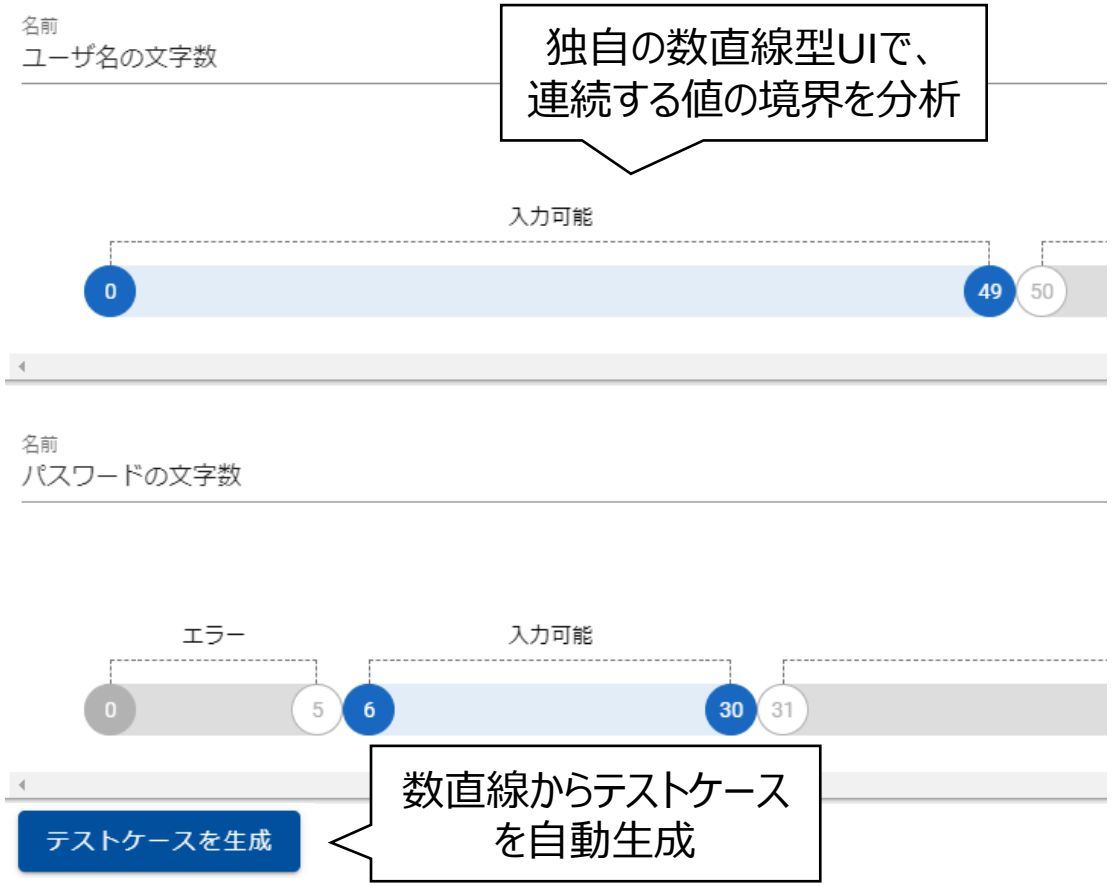
	0	1	削除する境界値	最大値・最小値
1	未満	より大きい	3個目	あり・なし
2	以上	未満	4個目	なし・なし
3	より大きい	より大きい	2個目	あり・あり
4	より大きい	未満	1個目	あり・なし
5	未満	以上	4個目	なし・あり
6	以下	以上	1個目	なし・なし
7	以上	より大きい	1個目	なし・あり
8	未満	以下	2個目	なし・なし
9	以下	以下	4個目	あり・あり
10	以上	以上	4個目	あり・なし
11	以下	未満	3個目	なし・あり
12	以上	未満	2個目	あり・あり
13	より大きい	以下	3個目	なし・なし
14	より大きい	より大きい	4個目	なし・なし
15	以下	以下	2個目	あり・なし
16	以下	より大きい	3個目	あり・あり
17	未満	以上	2個目	あり・あり
18	より大きい	以上	2個目	なし・あり
19	未満	未満	1個目	あり・あり
20	以上	以下	1個目	なし・あり
21	以上	以上	3個目	あり・あり

境界値分析とは

VERISERVE

- 連続する値の境界を分析し、境界となる値に対するテストケースを作成する技法
- 境界値を狙ってテストすることで、仕様の認識ミスや実装ミスによるバグを検出できる
- GIHOZでは

例：ユーザ名・パスワードの入力文字数に関するテスト



自動生成

ユーザ名の文字数のテストケース

csvダウンロード

	input(ユーザ名の文字数)	input(パスワードの文字数)
1	0	
2	49	
3	50	

パスワードの文字数のテストケース

csvダウンロード

	input(ユーザ名の文字数)	input(パスワードの文字数)
1		0
2		5
3		6
4		30
5		31

境界値分析の活用

VERISERVE

- 数値の境界をテストする場合に活用
- 境界上の数値を指定するだけで、境界値のテストケースを自動で作成できるのが便利

名前
TEB-71-デシジョンテーブルでY/Nの組み合わせを自動生成する（組み合わせ件数に応じたメッセージ表示の境界値）

数直線

同値分割のテストケース

組合せ件数のテストケース

csvダウンロード

	input(組合せ件数)	有効/無効	テストケースの意図
1	50	有効	警告なしに組み合わせ生成可の代表値
2	115	有効	警告は出すが組み合わせ生成可の代表値
3	139	無効	組み合わせ生成不可の代表値

境界値分析のテストケース

組合せ件数のテストケース

csvダウンロード

	input(組合せ件数)	有効/無効	テストケースの意図
1	0	有効	警告なしに組み合わせ生成可の下限値
2	100	有効	警告なしに組み合わせ生成可の上限値
3	101	有効	警告は出すが組み合わせ生成可の下限値
4	128	有効	警告は出すが組み合わせ生成可の上限値
5	129	無効	組み合わせ生成不可の下限値

取り組み②まとめ

VERISERVE

- **テスト技法を活用することで、効率よくテスト設計できる**
  - テストケースの作り方が決まっているので迷わない

- **GIHOZを活用することで、より効率的に！**
  - ボタン一つで組み合わせやテストケースを自動生成

テスト技法をある程度知っていれば簡単に使える

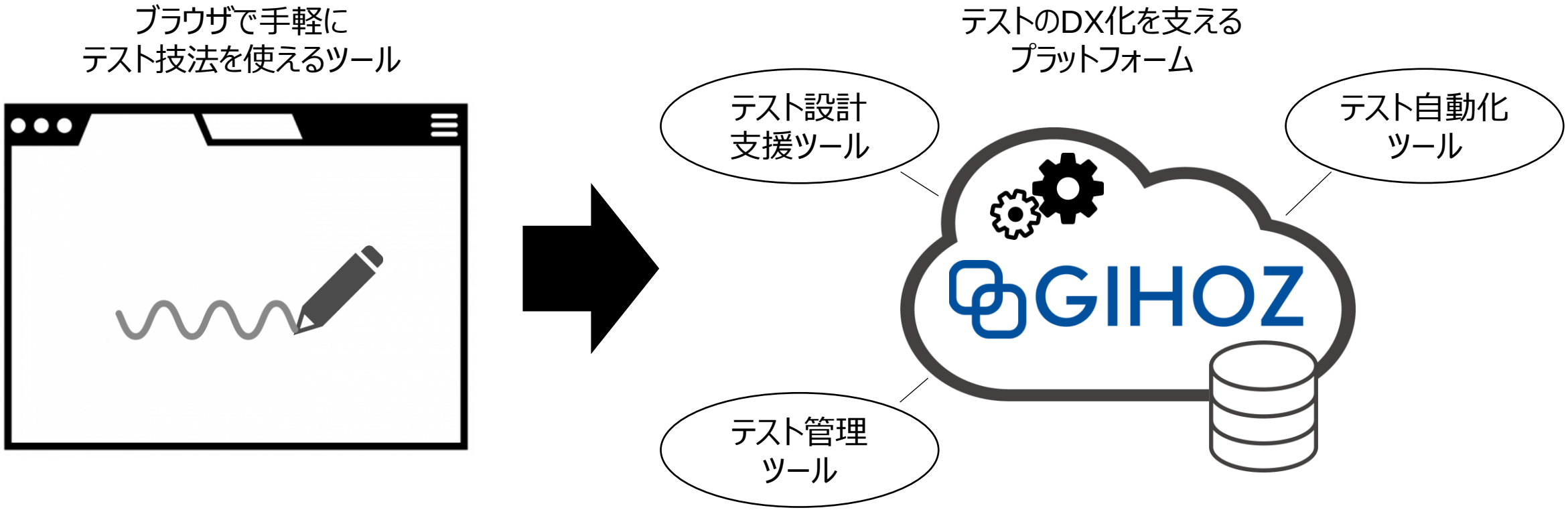
条件の組み合わせやテストケースの自動生成で効率化できる

デシジョンテーブルと状態遷移テストは活用頻度が高い

GIHOZの今後

VERISERVE

- テスト技法のデータはJSONで保存。データの読み書きのWeb APIを公開予定
- GIHOZで作成したデータで自動テストを実行するなど、テスト技法のデータ活用が可能になる



まとめ

- **アジャイル開発でも、段階的にQA活動を実施し、意識的に品質を作りこんでいく**
- **いつどんなQA活動を行うかチームで議論し続ける**
- **QA活動の武器を持つ、使ってみる**
  - アジャイル開発向けの品質チェック項目を活用して、バックログの質を高める
  - テスト技法を使って、効率よく、漏れの少ないテストケースを作成する
  - テスト技法を使うときはGIHOZ！

おわりに

アジャイル開発向けの品質チェック項目、GIHOZともにぜひご利用ください！

そして、フィードバックをいただけると嬉しいです

テスト技法を使うときはGIHOZ！（大事なことなので2回目）

https://gihoz.com

品質を創造する

VERISERVE

ご清聴ありがとうございました