JaSST’21 Tokyo

Dev&QAのOneチームによる プロダクトゴールへの第一歩

2021.3.15 – 2021.3.16

Dev&QAのOneチームによるプロダクトゴールへの第一歩

当セッションはSHIFTのアジャイル開発支援の実践事例のうち

- QA to AQの流れを汲んでいる
- SHIFT以外のQAメンバーでも実践できる

という部分を抜粋してご紹介させていただくセッションです。

QA to AQ: Patterns about transitioning from Quality Assurance to Agile Quality, by Joseph Yoder, Rebecca Wirfs-Brock, and Ademar Aquilar

自己紹介

- 名前：船橋 篤史
- 所属：ビジネストランスフォーメーション事業本部 品質・技術統轄部 技術推進部
  アジャイル推進第1グループ
- 経歴：
  - アジャイル開発体制構築、テスト自動化
  - Windowsネイティブアプリケーションのクラウドリフト
  - Webアプリケーションのクラウドシフト
  - 基幹システム構築
  - etc...

AGENDA

- 【1】Break Down Barriersとは？
- 【2】Break Down Barriersの実践

【1】Break Down Barriersとは？

Break Down Barriers（障壁の解体）とは？

障壁は必ずしも悪ではありません。

しかし、時には進捗や品質の問題を他人事として
認識させる可能性があります。

その結果、プロダクト出荷に対する障害を生みます。

そこで

- QA担当者が早い段階から参加する
- QA担当者がスプリント活動に参加する
- チーム全体で品質を評価する
- etc…

数あるアジャイルプラクティスに共通する原則「Oneチーム」というコンセプトの下で
協力しあいながら高品質な製品を生産しましょう。

QA to AQ Part 3: Shifting from Quality Assurance to Agile Quality "Tearing Down the Walls," by Joseph Yoder, Rebecca Wirfs-Brock, and Hironori Washizaki

障壁とは？

- 開発担当 vs 品質保証担当
  - 物理的な壁
    - 担当部署や担当企業の違い
  - 関係性の壁
    - 成果物を作成する側と指摘する側の違い
      - 作る側と利用する側(実際は利用しているつもり)
      - よく知らない相手から指摘がくる
    - そもそも品質保証担当がゲートキーパだったりする
      - 後だしじゃんけん感がある
  - etc...

【2】Break Down Barriers の実践

実践詳細

- Whole Team
- Product Quality Champion
- System Quality Specialist
- Automate As You Go

(a)Whole Teamの実践

アジャイル開発におけるSHIFTの入り方

ステークホルダー
製品の利用者、出資者、
管理職などの利害関係者

プロダクトオーナー
製品に対して責任をもち、
機能に優先順位をつける

スクラムマスター
スクラムプロセスが円滑
に進行するように、外部
から支援する

SHIFT

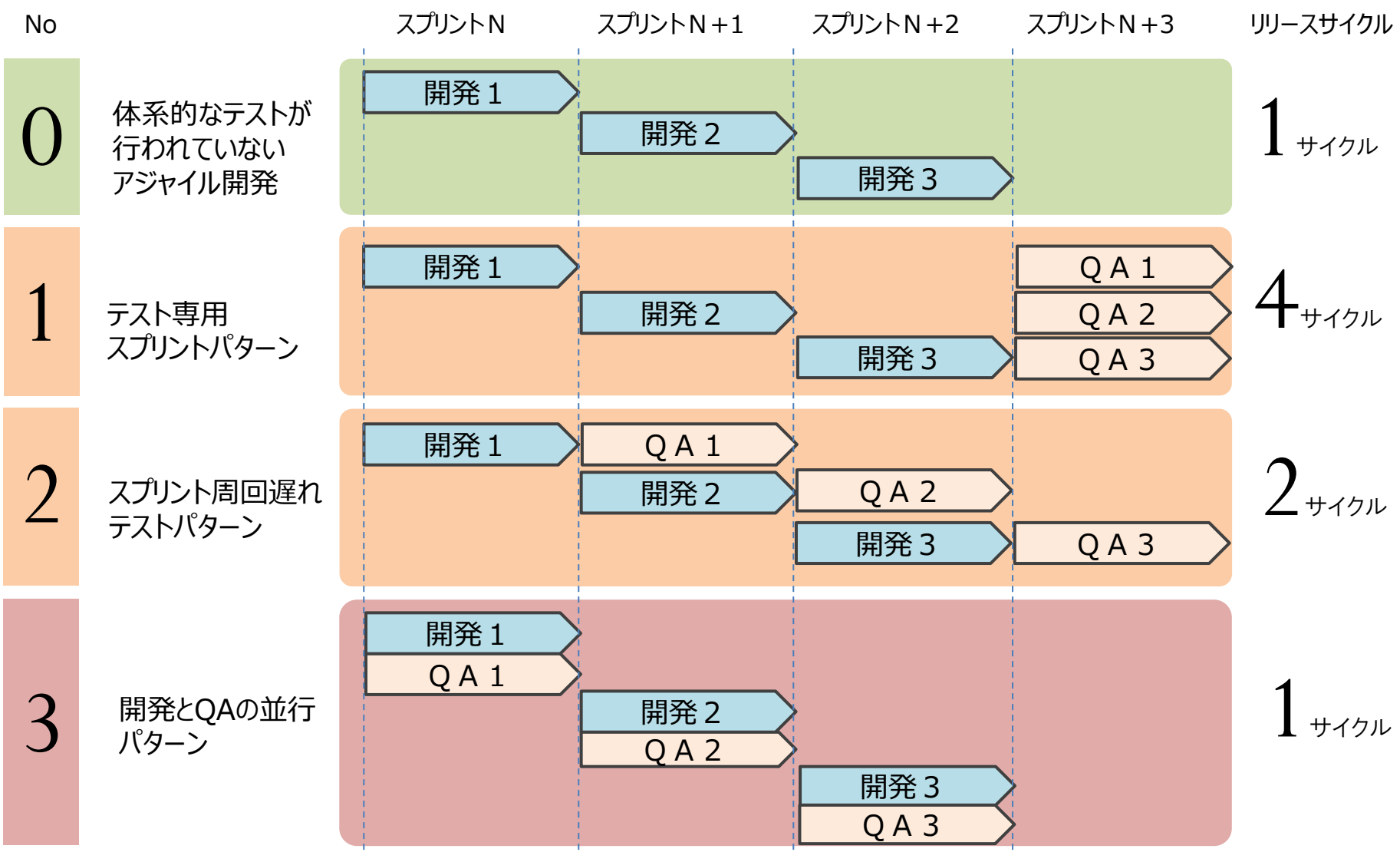
開発チーム
プロダクトの
開発を行う

開発チーム

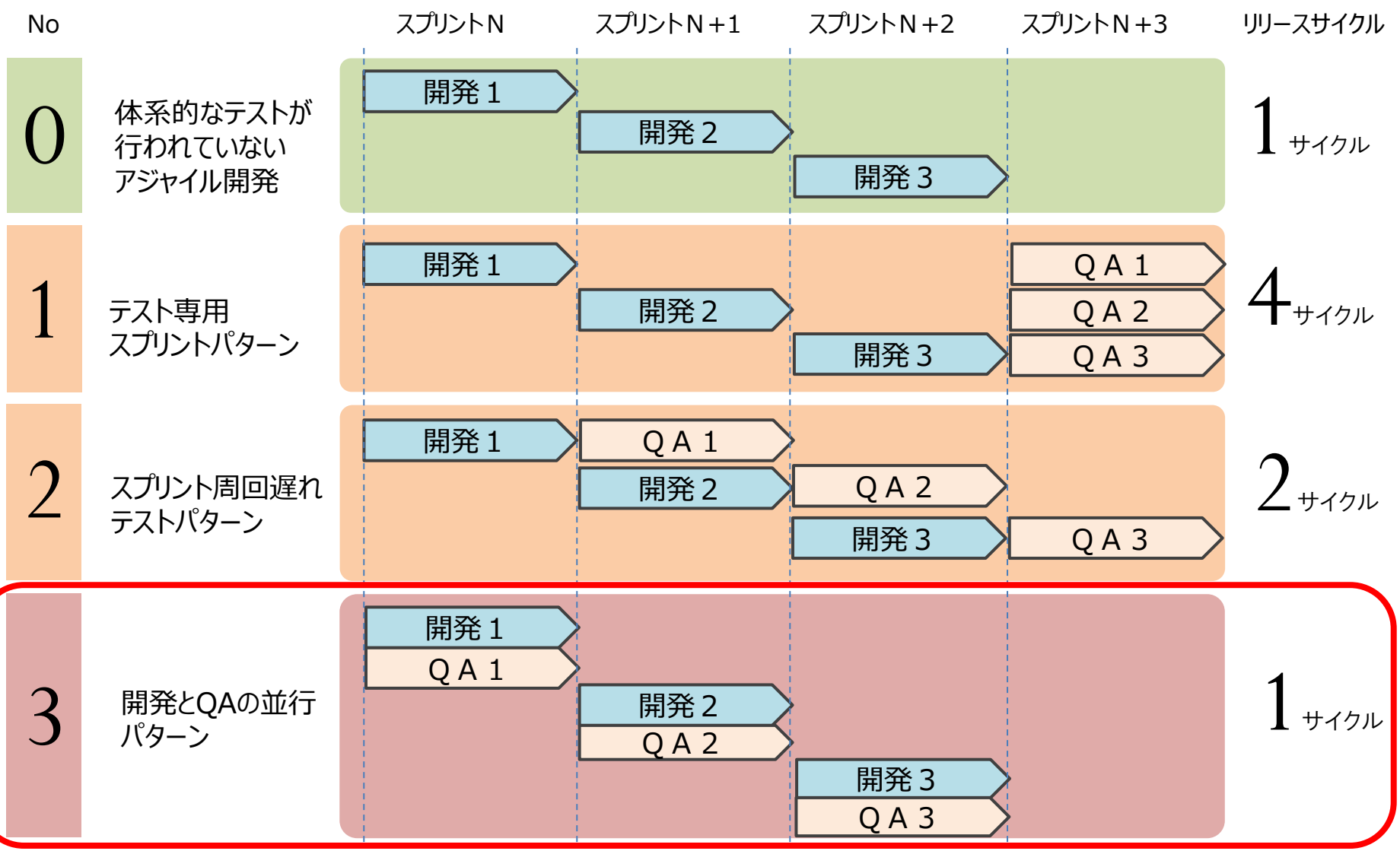
QAリード
スクラム内でQA活動が
円滑に進行するように、
リードする

QAエンジニア
開発と協調しながら
プロダクトをテスト
し、品質保証する

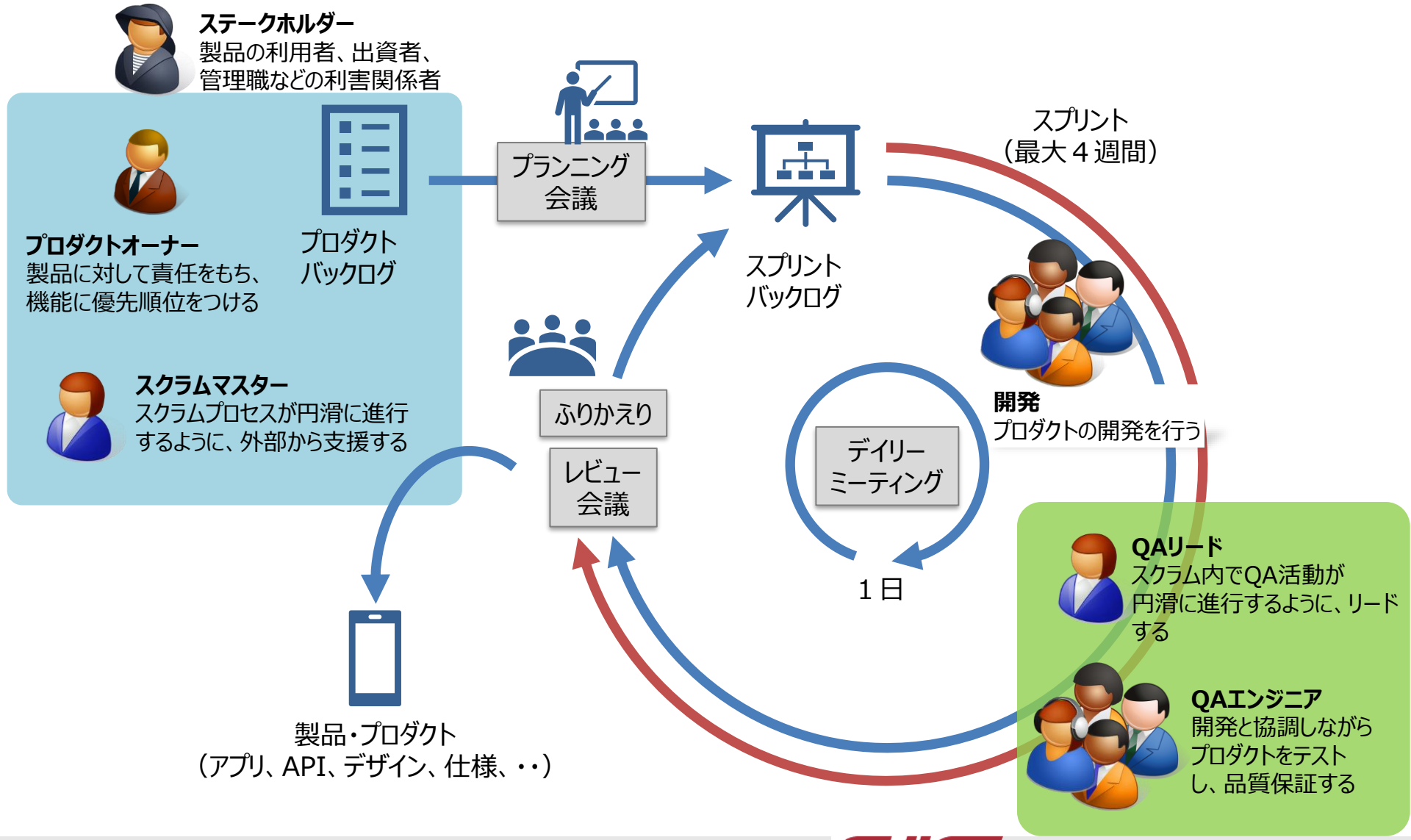
Dev+QAで考えられるスプリントのパターン



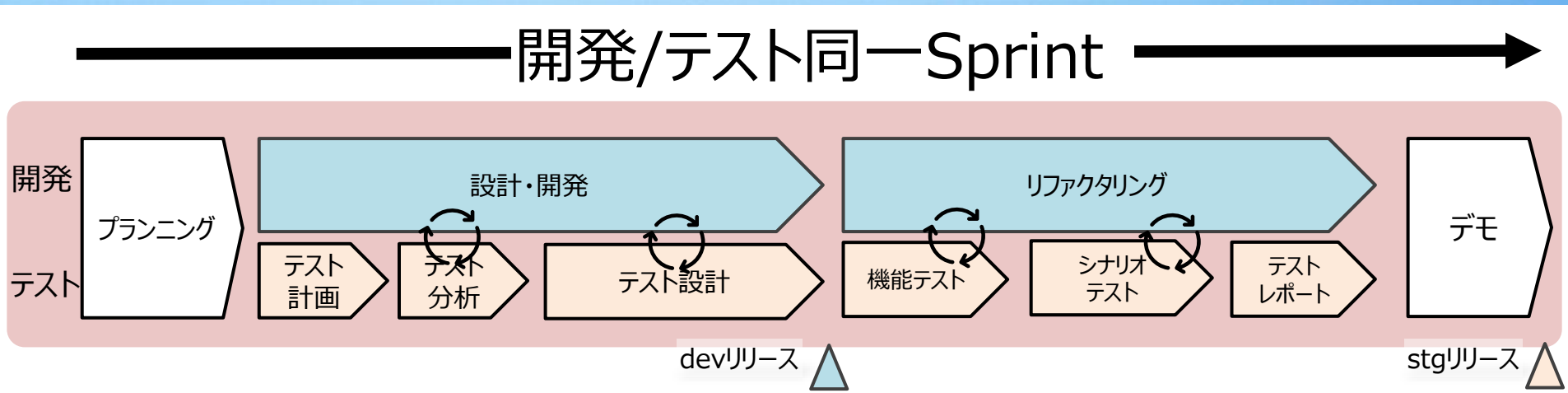
Dev+QAで考えられるスプリントのパターン



アジャイル開発におけるQAの関与



アジャイル開発におけるQAの関与



ポイント

- ✓ **開発とQAが連携できるよう、チームタスクを整理する。**
- ✓ **開発、QAが一体となったチームとなり、メンバー間のコミュニケーションが重要となる。**
- ✓ **開発タスク、QAタスクに理解のあるスクラムマスターが必要になる。**
- ✓ **スプリントへの理解があり、コーディング、リファクタリングを分けて開発ができる。**
- ✓ **各メンバーのスキルが高い必要がある。**

アジャイル開発におけるQAの関与

- 開発とテストが共に行動する
  ⇨ Whole Team実践の入口

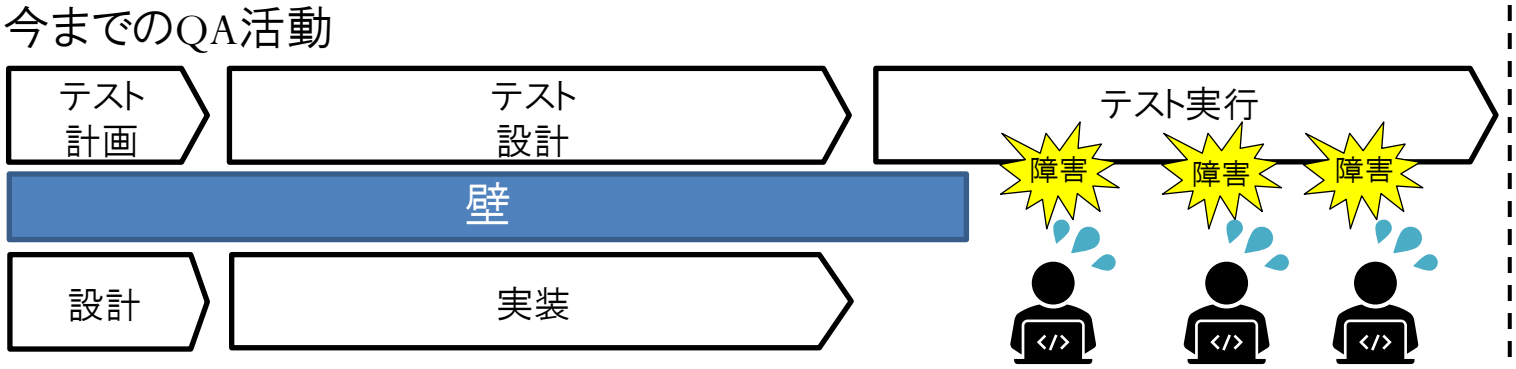
- Whole Team一部抜粋
  - QAがアジャイルチームに含まれていることで…
    - チーム全体は品質に関する知識が向上する
    - プロセス全体へ品質を組み込める
    - 品質要件にいつ対応すべきか理解や考慮することを助けられる

- ただ時間軸を同じにしただけでは？

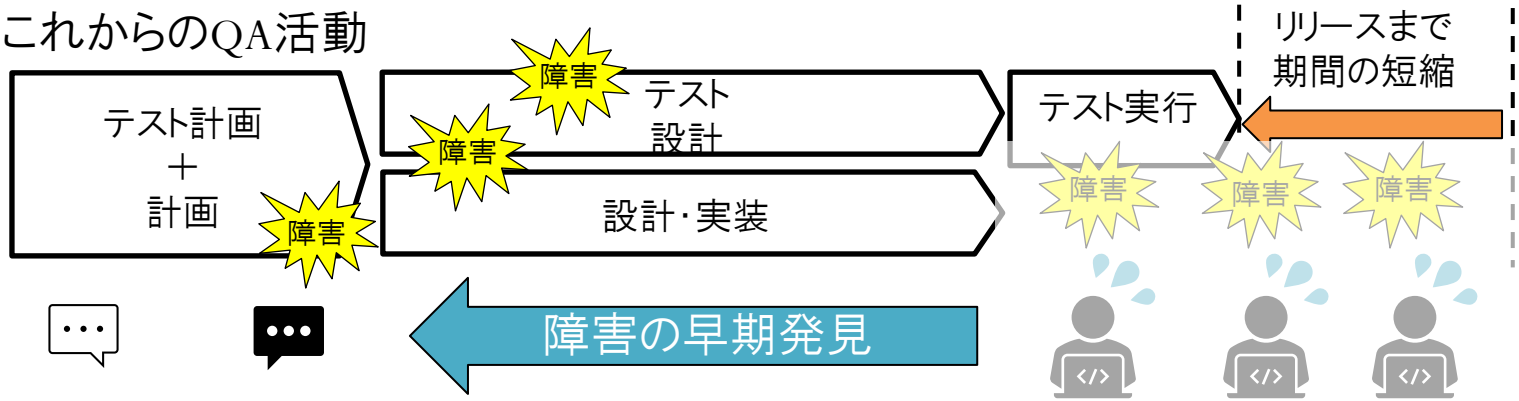
アジャイル開発におけるQAの関与

■ テスト実行で障害を発見するのではなく事前に障害を予防するQA

■今までのQA活動



■これからのQA活動



(b)Product Quality Champion System Quality Specialistの実践

QAリードの支援内容

■ テスト戦略やスクラムチームの支援

- ➢ 完了の定義に含めるテストタイプの見極め
- ➢ 各テストに対するテストアプローチの検討

			Front End			Back End(API)					DB				External Service	テストアプローチ
			業務ロジック	計算ロジック	⇔ API	業務ロジック	計算ロジック	⇔ FE	⇔ API	⇔ DB	業務ロジック	計算ロジック	CRUD	⇔ API	-	-
単体テスト	Unit Test	Front End	WB	WB	-	-	-	-	-	-	-	-	-	-	-	自動
		Back End	-	-	-	WB	WB	-	-	-	-	-	-	-	-	自動
		Database	-	-	-	-	-	-	-	-	WB	WB	WB	-	-	自動
単機能テスト	UIテスト	FE ⇔ Mock	BB	BB	-	Mock	Mock	Mock	Mock	Mock	-	-	-	-	-	自動
	APIテスト	BE ⇔ Mock	Mock	Mock	Mock	BB	BB	Mock	Mock	Mock	Mock	Mock	Mock	Mock	-	自動
	etc…	etc…	-	-	-	-	-	-	-	-	-	-	-	-	-	-
結合テスト	ユーザーストーリーテスト	FE ⇔ BE	BB	BB	BB	BB	BB	BB	BB	BB	BB	BB	BB	BB	-	手動
	APIテスト	BE ⇔ BE (API ⇔ API)	-	-	-	BB	BB	-	BB	-	-	-	-	-	-	自動
	APIテスト	BE ⇔ BE (API ⇔ External)	-	-	-	BB	BB	-	BB	-	-	-	-	-	Mock ※	自動
	APIテスト	BE ⇔ DB	-			BB	BB	-	BB	BB	BB	BB	BB	BB	-	自動
システムテスト	シナリオ	-	BB	BB	BB	BB	BB	BB	BB	BB	BB	BB	BB	BB	BB	手動
	性能	-	BB	BB	BB	BB	BB	BB	BB	BB	BB	BB	BB	BB	BB	手動
	セキュリティ	-	BB	BB	BB	BB	BB	BB	BB	BB	BB	BB	BB	BB	Mock	委託
	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
受入テスト	-	-	BB	BB	BB	BB	BB	BB	BB	BB	BB	BB	BB	BB	BB	手動

QAリードの支援内容

■ テスト戦略やスクラムチームの支援

- 完了の定義に含めるテストタイプの見極め
- 各テストに対するテストアプローチの検討
- Product Backlog Itemの受入要件に品質的要素の追加
- Un DoneなテストのProduct Backlog Item化
- etc…

⇨ 品質面でプロダクトゴールを支援

自己紹介

- 名前：言美 貴子
- 所属：ビジネストランスフォーメーション事業本部 品質・技術統轄部 技術推進部
  アジャイル推進第1グループ
- 経歴：
  - ウェブサービスのテスト設計・実行
  - 企業サイトのテスト設計リーダー・実行リーダー
  - 人材管理システムの設計リーダー・実行リーダー
  - 証券取引サービスのQAエンジニア
  - etc…

プロジェクト事例

ここからプロジェクト事例をご紹介します

ご紹介するプロジェクトの特徴は以下の通りです

- 証券系の複雑な要件のあるサービス
- アジャイルの経験はある人とない人の混成メンバー
- 人数は20人規模を2チームに分けて開発
- 期間に対して実装したい機能は多め
- 1スプリントは2週間
- FBは1スプリント毎に関係者数名から行われる
- 複雑な処理の大部分は外部APIが担っている

このような案件でどのように活動しているのか
簡単にご紹介させていただきます

QAエンジニアの主な仕事

■ テスト計画

- ➢ 全体テスト計画に基づいてどんなテストを行うのか決定する
- ➢ いつから何がテストできるか
- ➢ どのテストをいつまでに終わらせるか

■ テスト設計

- ➢ どこまでテストをするか
- ➢ 期待値（受け入れ条件）の確認

■ テスト実行

- ➢ アカウントの準備
- ➢ 不具合の検出と報告
- ➢ 不具合の修正確認

QAエンジニアの主な動き

**スプリントはじめは
開発とQAで
プランニングをみっちり
行います。**

**QAは不具合を報告し
開発が修正を行います。

ふりかえりは全員で行います。**

スプリント序盤

- **プランニングで開発・テストのスケジュールを決める**
- 細かなテストスケジュールをQAメンバーで決める
- **決定したプランを開発・QAで共有**
- **開発と認識すり合わせ（設計相談会）**

スプリント中盤

- 主にテスト設計を行う
- データ作成などテスト実行の準備をする

スプリント終盤

- テスト実行を行う
- 不具合の検出・修正確認を行う
- ふりかえり

**テストの準備をしているなかで
出てきた疑問点を中心に
設計レビュー会を
開発・QAの全員で行います。**

スプリント序盤：開発とスプリントのプランニングを行う

スプリント序盤 | スプリント中盤 | スプリント終盤

- 開発とQAが一緒にプランニングを行うことで、
  短いスプリント内でテストまで終えられるスケジュールを立てれる

ストーリーAとストーリーBはレイアウトの修正
だからすぐ終わるぞ！先に終わらせよう！

QAとしてはストーリーCの多重起動制御を
テストするのに時間かかりそうなので、先にC
から開発していただけると助かります！
Cで大きな不具合出ると怖いので…

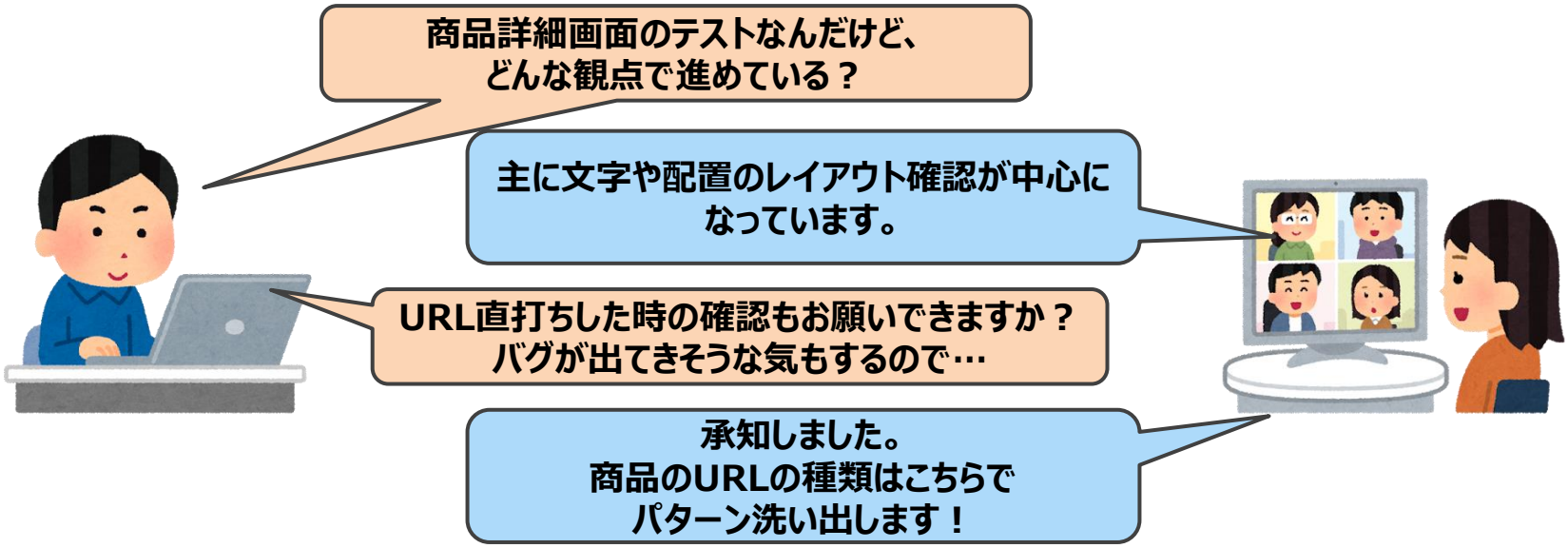
なるほど！
レイアウトは急がないね。
先にCから開発進めます！

スプリント序盤：設計相談会を行っている

スプリント序盤 | スプリント中盤 | スプリント終盤

■ 設計相談会を開発チーム（開発･QA）のみんなで行う

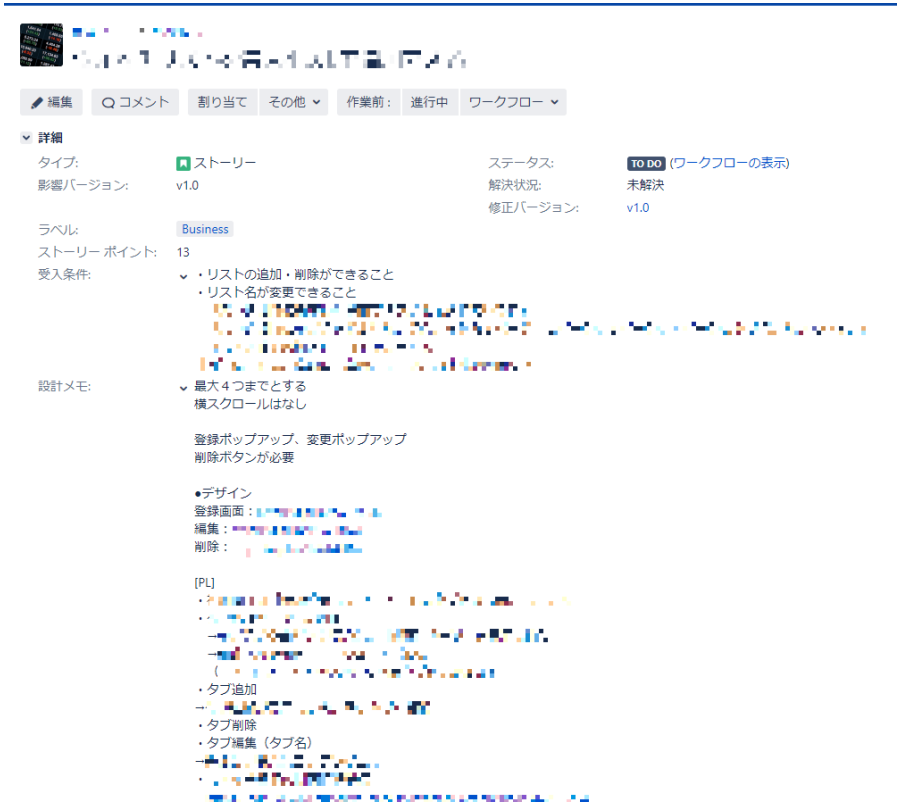
- 開発とQAの疑問点などもここで共有する
- 開発の全員と行うことで、細かな仕様の確認もできる



スプリント序盤：設計相談会を行っている

スプリント序盤 | スプリント中盤 | スプリント終盤

- 開発とQAがそれぞれ気になっている点を共有することでチームの方針を確認できる
  - **プロダクトバックログを実現させる道筋を決めて共有することでスプリントゴールがわかりやすくなる**
  - 決定事項は「設計メモ」としてJIRAのチケットに記載している



設計相談会について

スプリント序盤 スプリント中盤 スプリント終盤

【設計相談会で行っていること】

- 設計相談会はQAが主体となる会
- そのなかで開発の疑問も素直に出す
- 仕様がぶれているところがないかみんなで確認

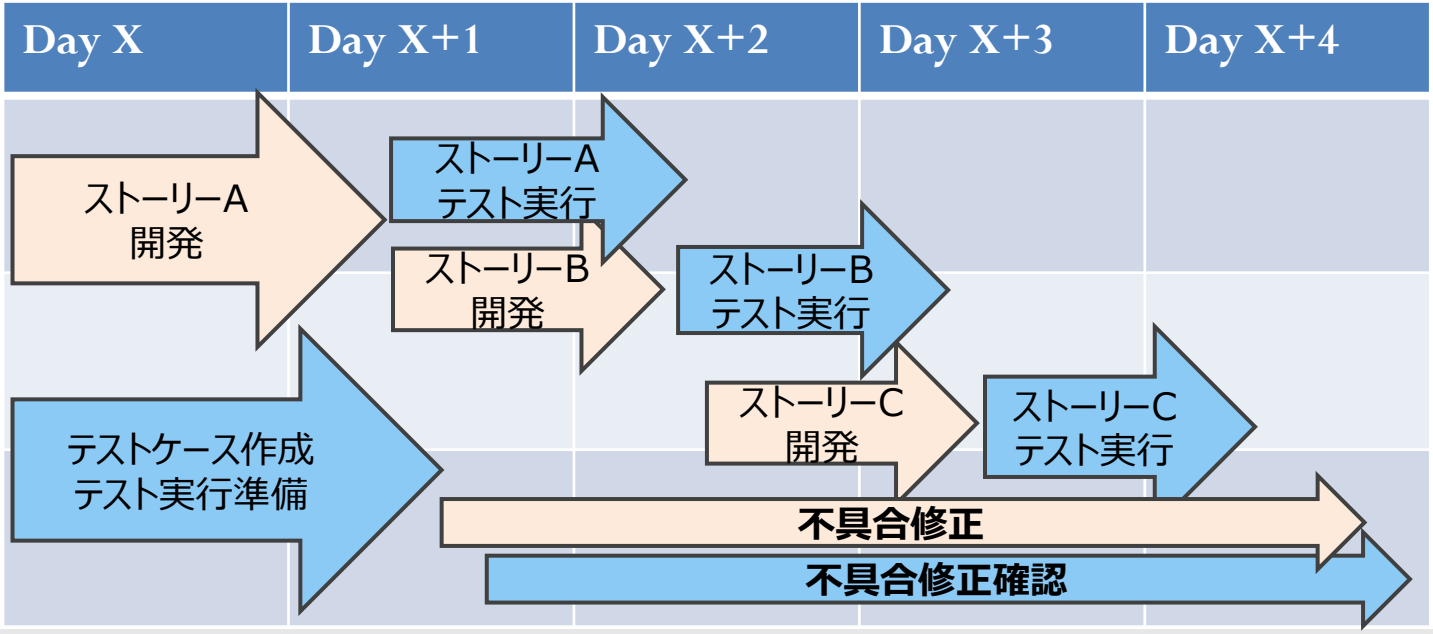
【設計相談会の効果】

- どんなテストが必要なのかを開発が理解できる
- 違う立場からのフィードバックがあるので考慮漏れが抑えられる
- 早い段階でのフィードバックがもらえる

スプリント中盤：開発が進むなか、テストの準備を行う

スプリント序盤 | スプリント中盤 | スプリント終盤

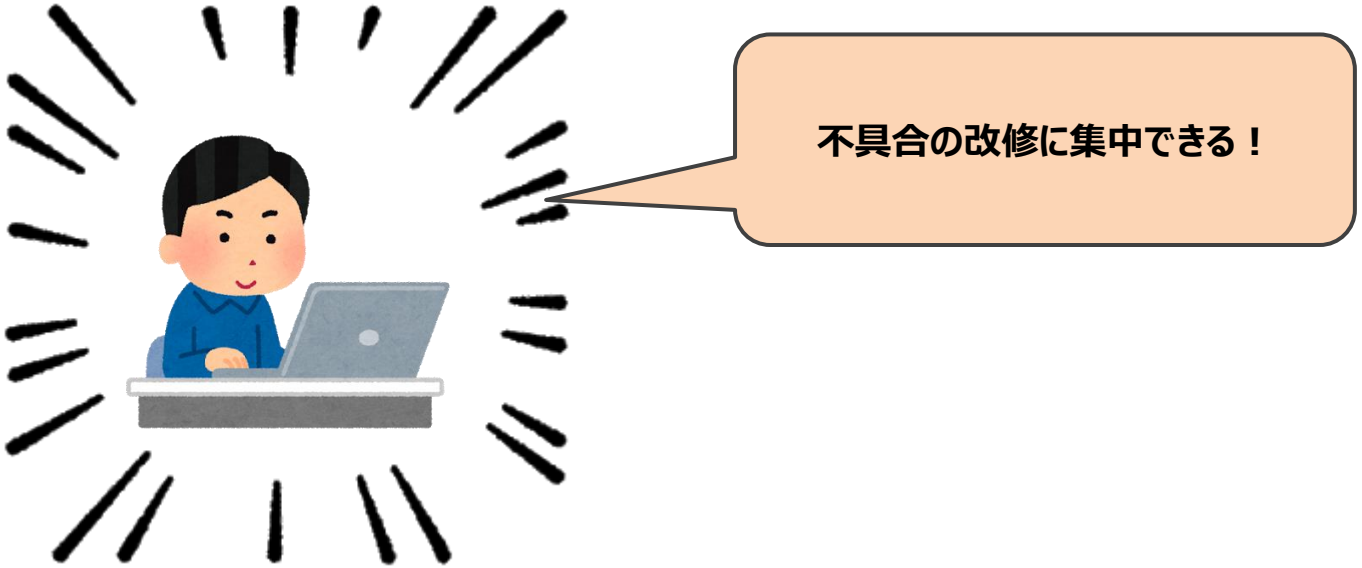
- テスト設計を行う
- 受入れ条件を確認し、テストの設計やテスト準備を行う
- 開発とQAが同時に進行する
  - **不具合の大きさによって、別ストーリーの開発よりも不具合修正を優先する判断もできる**



スプリント終盤：不具合の報告と修正

スプリント序盤 スプリント中盤 スプリント終盤

- QAが不具合を検出
  →開発が調査と修正を行う
  →QAが修正の確認を行う
- 不具合の検出と修正を同時並行で行う

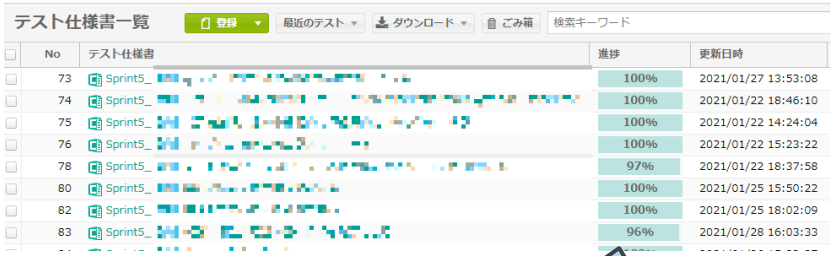


デイリースクラムでの進捗報告

CAT COMPUTER AIDED TEST

- スプリントバックログ以外に、テスト管理ツール「CAT」による進捗管理も行っている
  - テスト仕様書ごとや実行者ごとでの進捗管理も可能

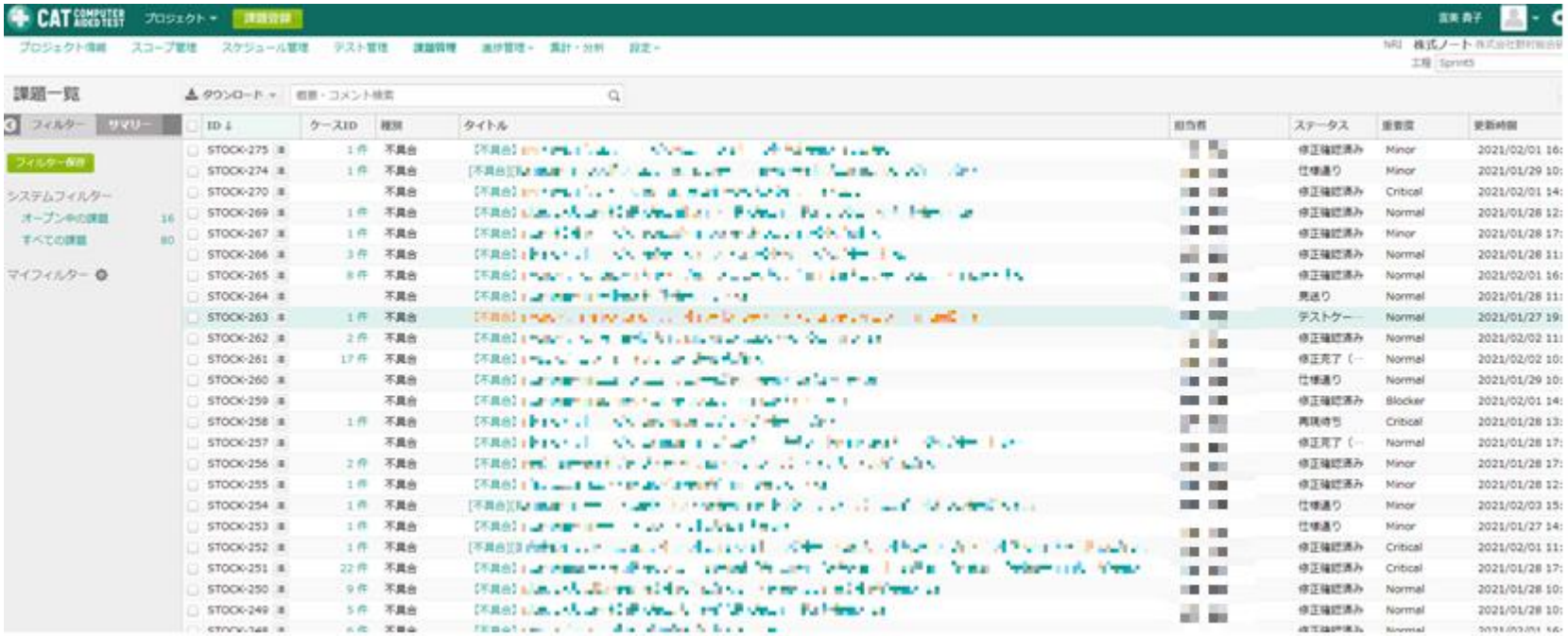
実行状況を
グラフで可視化



テスト仕様書ごとに
進捗がみれるので
どこでテストが止まっているかが
分かる

デイリースクラムでの進捗報告

- スプリントバックログ以外に、テスト管理ツール「CAT」による進捗管理も行っている
  - 不具合の共有も行っている



QAリード、QAエンジニアの存在

- Product Quality Champion
  - 品質に関するロードマップやバックログの検討
  - 品質の可視化
  - 品質重視のチームとしての支援
  - etc…
- System Quality Specialist
  - チーム全体の品質意識向上
  - 重要な品質の発見
  - 品質に関連する専門技術・知識の活用
  - etc…

⇨このような活動を通じWhole Teamを促進

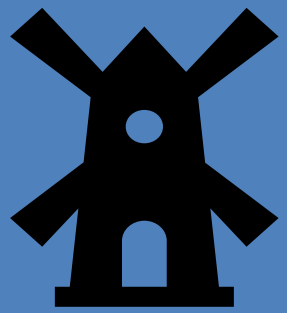
(c)Automate As You Go の実践

自己紹介

- 名前：石井 一成
- 所属：ビジネストランスフォーメーション事業本部 品質・技術統轄部 技術推進部
  高速開発推進グループ
- 経歴：
  - 運用保守フェーズにおける、GUIリグレッションテスト自動化推進
  - 運用保守フェーズにおける、RPA導入支援
  - アジャイル開発体制下における、API・GUI自動テスト導入・実装
  - アジャイル開発体制下における、CICD環境構築支援
  - etc…

自動テストはメリットがいっぱい

テスト工数の削減



作業の機械化

心理的安全性の確保

DevとQAの協調の促進

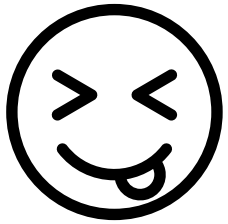
自動テストは悩みもいっぱい

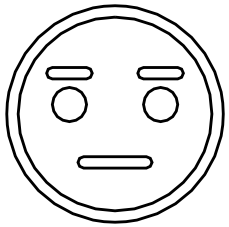
わかりやすいUIが欲しい！

すぐに導入して
すぐに結果
出したい！

クロスブラウザ、CICD、
オートヒーリング、何でも
載っててほしい！

全部は…無理…





だからプロジェクト特性に応じた
自動化戦略が重要！

自動化戦略立案 - まずはじめに ～例えばAGとWFの違い

	アジャイル	ウォーターフォール
規模	少しずつ大きくなる	最初から大きい
期間	かなり短く、 何度も繰り返す	長い
リリース後の開発	FBを受けて頻繁に行う	頻繁には行わない
リファクタリング	効率的な機能追加に必須	敬遠される
開発資料	まとまった資料はない	まとまった資料がある
テスト資料	自動化を考慮する 必要がある	自動化を考慮していない
テスター	最初から最後までいる	概ねテストフェーズのみ

※特に違いがあるところを抽出しています

自動化戦略立案 - まずはじめに ～アジャイルのテスト

	アジャイル
規模	少しずつ大きくなる
期間	かなり短く、 何度も繰り返す
リリース後の開発	FBを受けて頻繁に行う
リファクタリング	効率的な機能追加に必須
開発資料	まとまった資料はない
テスト資料	自動化を考慮する 必要がある
テスター	最初から最後までいる

アジャイルにおけるテスト
プロダクトの成長に応じて、 テストの優先・重要箇所が変化
迅速かつ高頻度な リグレッションテストが必要
DevとQAの 密なコミュニケーションが重要

**すなわち、アジャイルではWFとは異なった知見による
テスト計画、テスト自動化の戦略が必要となります**

自動化戦略立案 – プロジェクトを取り巻く環境

そして同じアジャイルであっても、テスト自動化は
以下のような要因により、いつ・どこまでするべき、できるのかが変わってきます

開発・テスト環境

- どのような環境が用意されていて、どこまで構築が完了しているか
- 開発期間中、いつ、どの程度、環境にアプローチできるか
- テストデータはどこまで柔軟に用意できるか

プロダクト特性

- サーバー側、あるいはフロント側にロジックが偏っている
- 複雑な外部サービスを多用しているなど、
  スプリント開始時に仕様を固めることが困難である

開発チームの成熟度

- アジャイルの文化に不慣れ
- 単体テストを書く習慣がない

リソース

- 自動化に理解のある人間はいるか
- 自動化のための時間・予算をどのくらい投入できるか

自動化戦略立案 – 目標設定

SHIFTではこのようなアジャイル特性、プロジェクト環境を加味して、
適切な自動化目標を立てるとともに、状況の変化に応じて
適宜目標のアップデートを行っています。

種まき

- 自動テストの文化を根付かせるために行う、
  最低限度の自動化実装
  例）環境整備～スモークテスト

理想

- 自動テストが最大限の費用対効果を発揮する
  効率的な自動化実装

過剰

- 他のテストレベルをカバーしたい、絶対に壊れてはならない
  などの理由で費用を度外視して行う、例外的な自動化実装
  例）テストスクリプト自体がロジックをもっている
  境界値や2因子間網羅を超えているなど

自動化戦略立案 – 効果最大化

**また、プロジェクトを取り巻く環境に対して、
各テストレベルのメリットを活かし、デメリットをカバーできるように、
包括的な自動化戦略を組むことで、プロジェクト推進を強力にアシストします**

テストレベル	メリット	デメリット
Unit	・高速 ・高頻度 ・高耐久	・原則、開発者自身で実装が必要 ・自動化に適する開発が必要 ・経験の少ない開発者もいる ・検証できない部分も多い
API	・そこそこ高速 ・そこそこ高頻度 ・そこそこ高耐久 ・Dev、QAどちらも着手可能 ・レポート≒IN/OUTの仕様書	・環境構築が意外と大変 ・仕様が不安定だとすぐ壊れる ・コミュニケーションコストが高め
GUI	・E2Eより高頻度 ・外的要因に左右されない ・レポート≒機能仕様書	・低速 ・低耐久 ・E2Eでも行えるテストなので、 あえてGUI専用環境を作る場合もある
E2E	・ほぼ全て検証できる ・安心感が強い ・レポート≒運用手順書	・高コスト（低速,低耐久,低頻度） ・外的要因に影響を受けやすい

プロジェクト事例 - APIテスト

当初計画

- 前PJでAPIテスト環境整備済みのため、テスト駆動な開発を目指す
- サーバー側のロジックが非常に少ないことから、スモークテストとバリデーションテストのみの高速かつ最低限の検証を行う

実際

- テスト駆動を諦め、スプリント内のすべてのAPI実装が完了したのちAPIテストを実装した

理由

- 業務上の複雑な仕様により、設計完了～コード実装までの間に少なくとも3回以上の仕様変更という状況が毎スプリント発生。そのため、改修とコミュニケーションコストの削減を目的に方針を転換した

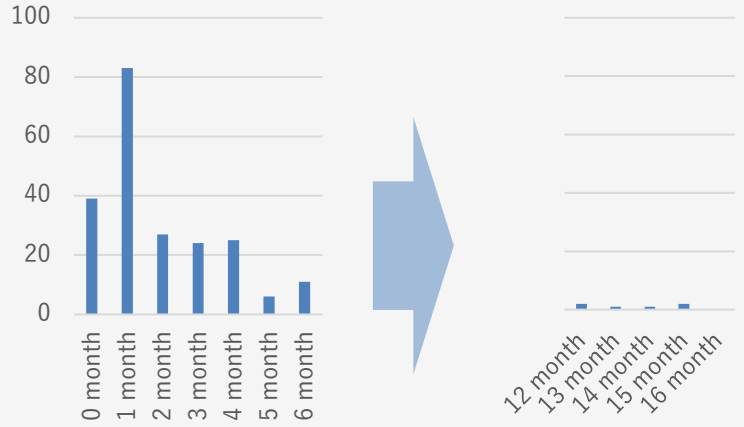
成果

- 手動GUIテストでは困難な検証の実施
- APIテスト実装工程において一部ソースのコードレビューを実施
- gitにpushするたびAPIテストが実行され、最新のIN/OUTを出力
- リグレッションテスト資産の蓄積

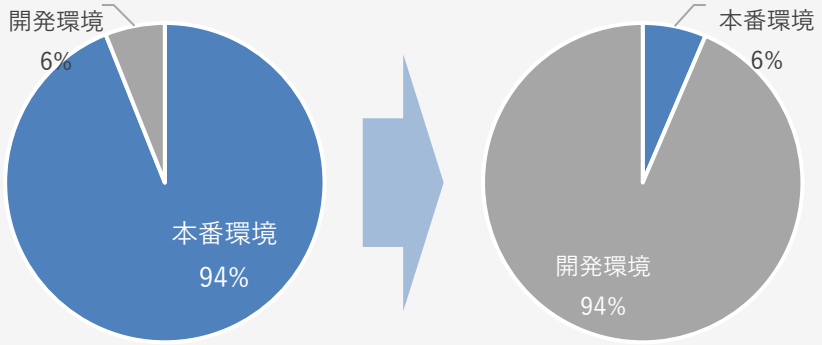
プロジェクト事例 - APIテスト（別事例）

本案件ではPJの最初期からAPIテストを導入していたため、導入前後での効果観測が困難でしたが、ユーザー公開後のシステムにAPIテストを導入した別事例では大きな効果を確認できます。この案件ではAPIドキュメントの整備、APIシナリオテストの導入、CI環境の構築と、徹底的な整頓が効果を発揮しました。

プラットフォーム利用者からの不具合報告数推移



不具合の環境別摘出比率の変化



利用者指摘の不具合数は約1年でほぼ0に

本番環境まで到達する不具合は大きく減少

プロジェクト事例 - GUIテスト

当初計画

- GUIテスト用の環境を用意し、画面キャプチャの取得と機能テストの検証をするGUIテストを実装する

実際

- 未実装、未実施

理由

- サービス時間中にE2Eの自動テストを回しておいて、あとからレポート検証するという運用で現状十分であるため
- 他に優先度の高いタスクがあり、環境整備と実装の負荷をかけてまで行うメリットがないため※機能の追加により重要度も増し、将来的には必須の想定

成果

- 未実施のため、成果なし。
- ただし実装すれば、環境制約によらず、レイアウト検証や最低限の機能検証ができるだけでなく、画面仕様書、機能仕様書としても活用が期待できる

プロジェクト事例 - E2Eテスト

当初計画

- スプリント毎に肥大していくシナリオテストを省力化する目的で実装
- 実行結果レポートの目視確認をもってシナリオテスト実施完了とする
- 最初はスモークテストのみ、Sprintが進むにつれてアサートを充実させる

実際

- 基本的には計画通りに実施できている
- ただし、手動テストより先行できていないため実施内容が重複している

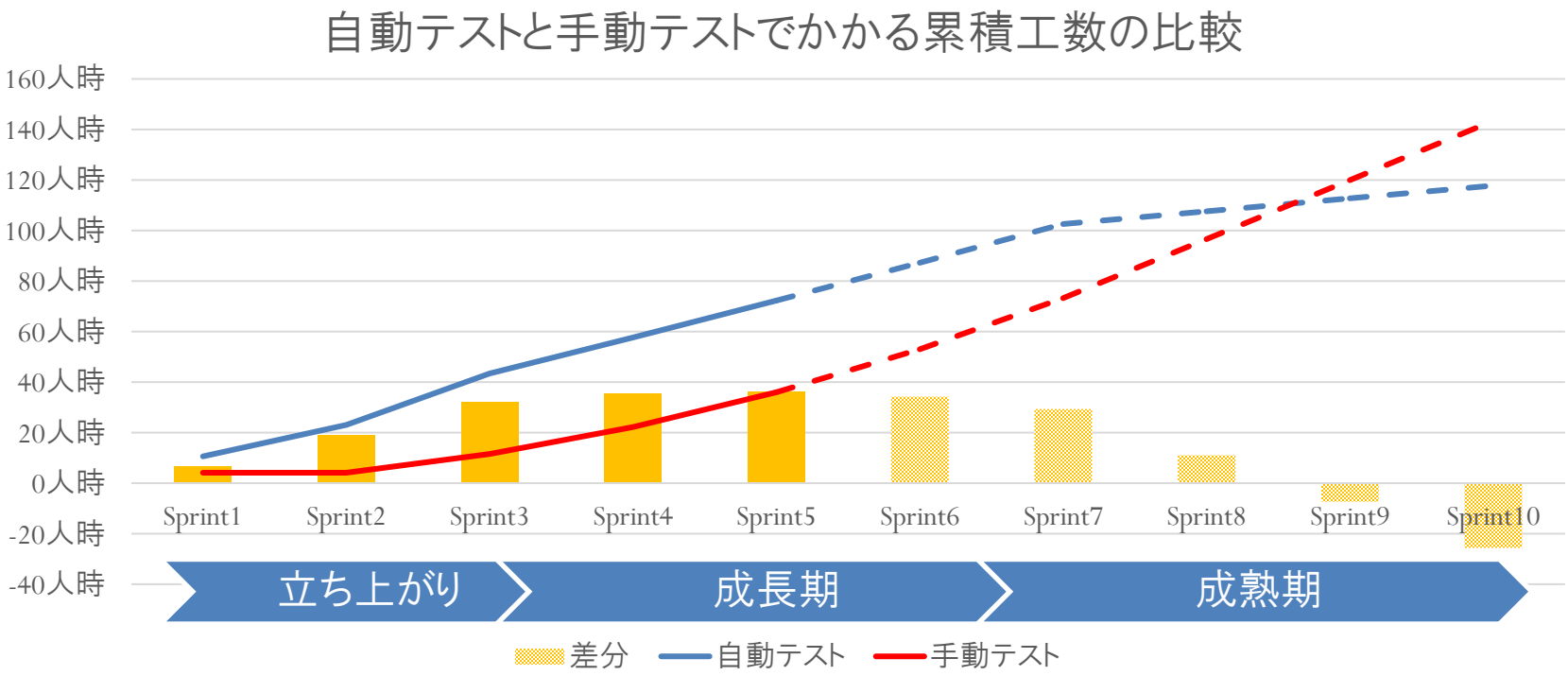
理由

- Devに求められる作業量が多く、カスタム属性の考慮など、自動テスト実装にむけたDevからの協力を受けることが難しかったため

成果

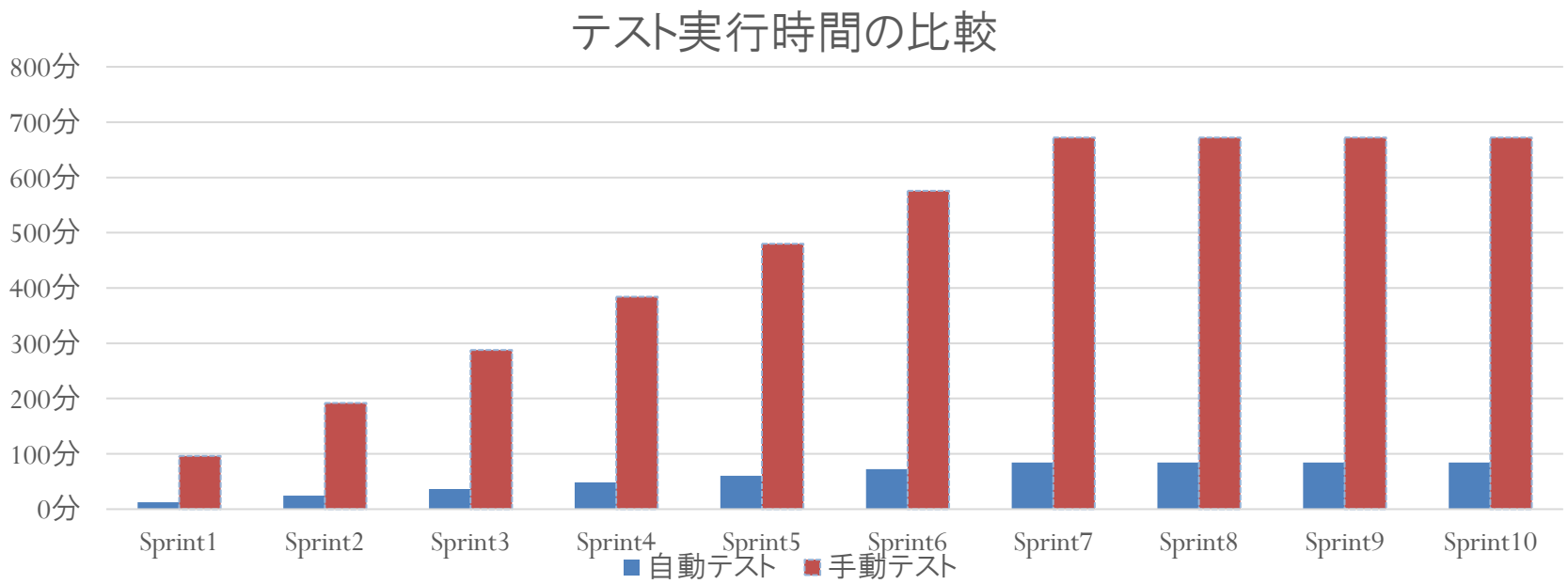
- 現段階では手動・自動テストの損益分岐点には至っていない、将来的に達成できる見込みである。
- 開発工数の確保という形でチームに寄与

E2E自動テスト導入による工数削減効果 (1/2)



開発が進むにつれ、手動テストにかかる工数は比例関数的に増えていくのに対し、自動テストにかかる工数は緩やかになっていく
Sprint9にて開発が完了すると仮定すると差分はさらに小さくなる

E2E自動テスト導入による工数削減効果(2/2)



自動化をしなかった場合、Sprint5の時点でテスト実行時間が6時間以上。
対して自動テストは1時間程度と、大幅な開きがある。
テスターの人数がいれば完了までの時間は削減できるが、常に多数のテスターを抱えるのは困難であり、テスト実行時間が長いほど、開発工数は大幅に削減されていく。
すなわち、自動化は工数上の損益分岐点より前から利益をもたらしてくれる

プロジェクト事例 - CICD

静的解析・単体テスト

- 導入済みであったため弊社からの支援なし

APIテスト

- Pushトリガーでkubernetesのpodをデプロイし環境構築するところから支援
  ※ベースとなるマニフェストファイルをCI向けに調整してデプロイしている
- Push後パイプラインが回りきると自動的にAPIテストのレポートも出力される

GUIテスト

- 未実装。E2E同様環境制約があるため、CI導入の予定なし

E2Eテスト

- 環境制約によりpushや定期実行をトリガーにし難かったので、オンラインから誰でも2～3クリックで実行できるように実装。
- ※Chrome, Firefox, Edge, Mac_Safariで実行される
- パイプラインが回りきると自動的にE2Eテストのレポートも出力される

プロジェクト事例 -Unitテスト

当初計画

- 原則として開発チームにお任せ

実際

- サーバーサイドは毎スプリント実装されている
- フロントサイドは未着手

理由

- サーバー側はナレッジもあり、もともとのロジックの少なさもあって実装できた
- フロントサイドの単体テストの有識者、ナレッジがわずかで、
  限られた期間の中で着手する時間的、心理的余裕がなかったため

成果

- サーバー側はロジックが少ないこともあり、サーバー起因の障害はわずかな件数
- フロントサイドは障害が多くデグレなども多い状態

今後

- 手動テストで集まった不具合のナレッジを用いつつ、フロントの単体テスト勉強会を開催し、スキルアップと心理的余裕不足の解消を目指しているところ

⇒サービス化も検討中！！

自動テストのまとめ

- 自動化の効果を発揮するには
  DevとQAの協力が不可欠

まとめ

- QAリード&QAエンジニア
  - 開発と協力しあいながら品質を保証する
- テスト自動化
  - 開発と協力しあいながら自動テストを構築する
  - 自動化は初期段階または早期に構築

- チーム全体で品質を評価する

「Oneチーム」が「障壁の解体」への第一歩となり、協力しあいながら高品質な製品を生産する活動へとつながります。

参考文献

- QA to AQ (http://www.wirfs-brock.com/)
  - QA to AQ: Patterns about transitioning from Quality Assurance to Agile Quality, by Joseph Yoder, Rebecca Wirfs-Brock, and Ademar Aquilar, presented at AsianPLoP 2014.
  - QA to AQ Part 2: Shifting from Quality Assurance to Agile Quality "Measuring and Monitoring Quality," by Joseph Yoder and Rebecca Wirfs-Brock, presented at PLoP 2014.
  - QA to AQ Part 3: Shifting from Quality Assurance to Agile Quality "Tearing Down the Walls," by Joseph Yoder, Rebecca Wirfs-Brock, and Hironori Washizaki, presented at SugarLoafPLoP 2014.
  - QA to AQ Part 4: Shifting from Quality Assurance to Agile Quality "Prioritizing Qualities and Making them Visible," by Joseph Yoder, Rebecca Wirfs-Brock, and Hironori Washizaki, presented at PLoP 2015.
  - QA to AQ Part 5: Being Agile At Quality "Growing Quality Awareness and Expertise," by Joseph Yoder, Rebecca Wirfs-Brock, and Hironori Washizaki, presented at AsianPLoP 2016.
  - QA to AQ Part 6: Being Agile At Quality "Enabling and Infusing Quality," by Joseph Yoder, Rebecca Wirfs-Brock, and Hironori Washizaki, presented at PLoP 2016.

その常識、変えてみせる。

SHIFT