JaSST’22 Tokyo

アジャイルの中と外、聞くのとやるのは違うの巻

SHIFT
DevOps推進部 佐藤 博之

目次

【お品書き】

1. アジャイル開発での取り組みの紹介
2. アジャイル開発とは？
3. アジャイル開発の中での品質活動と開発との関わり方
4. 出会った困り事の1問1答

SHIFTのアジャイル開発への取り組み

✓ SHIFTのアジャイル開発

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

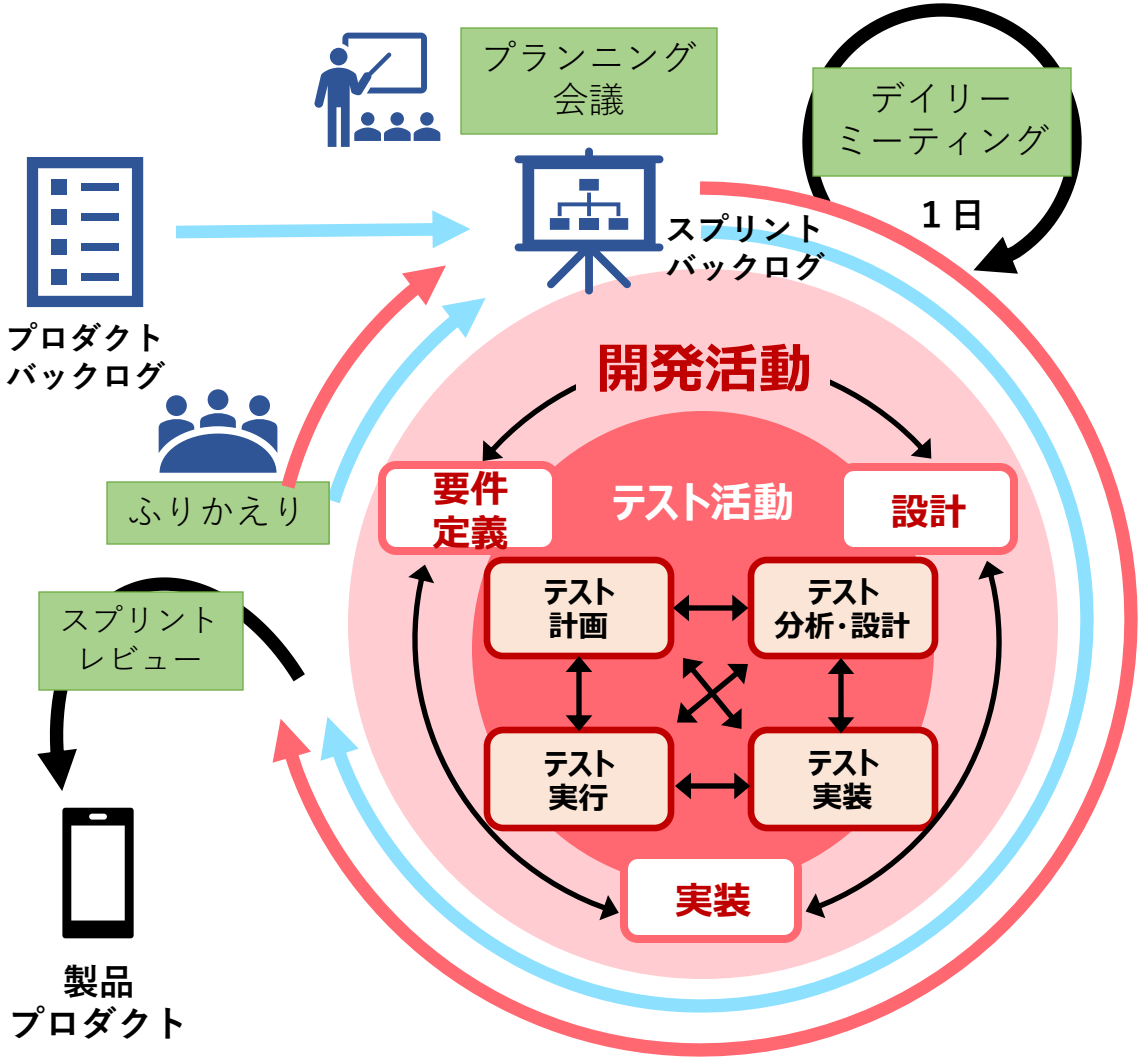
プロダクトオーナー
製品に対して責任を持ち、機能に優先順位をつける

–スプリント外支援–

プロダクトオーナー支援
(PBL作成／チーム説明／受入)

アジャイル導入支援
(出島戦略／WF差分検討／ロードマップ)

アジャイルテスト戦略支援
(自動テスト導入／テストツール選定／テスト方針)



–スプリント内支援–

アジャイルテスト支援
(計画／設計／実行／レポート)

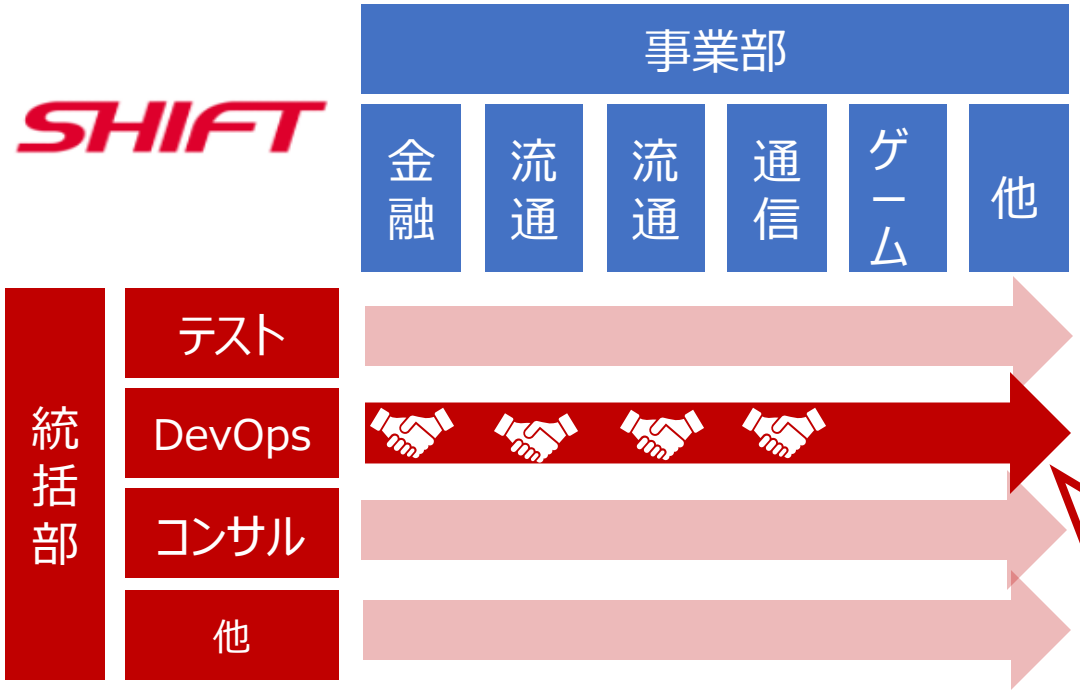
スクラム推進支援
(チームビルド／イベント定着)

スクラム開発立上げ支援
(スクラムマスター育成／チーム立ち上げ)

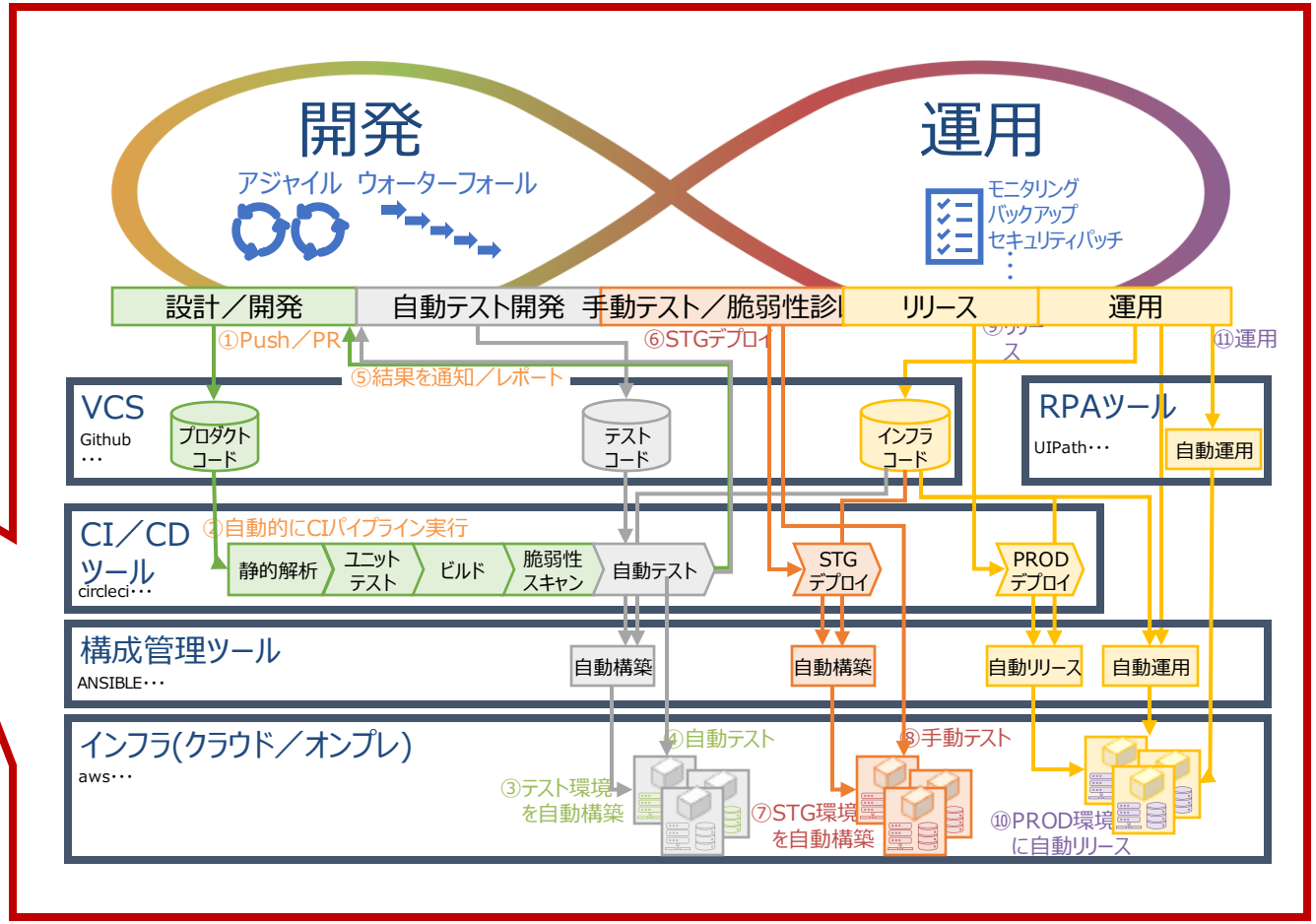
スクラム開発チーム支援
(チーム派遣／インフラ構築)

✓ 組織としてのアジャイル開発への取り組み

■インダストリ×技術のマトリクス組織



■インダストリ毎の技術展開



インダストリ、顧客ごとに特徴があるため、同じ仕組みで対応できない

✓ 事例：基幹システムへのアジャイル開発の適用事例①

- 20年間使っていた基幹システムが現在のビジネスモデルに合わない
- オンプレで構築していたサーバーの保守期限切れ

5~10年に一度のシステム変更＆その都度全国600店舗への説明による業務効率の低下が課題

1年半でリプレイスできないか見積
期間と費用の面で依頼

ベンダーへの丸投げで業務フローすら整理できていない状態でSHIFTが参画

情報を整理していく中で、立て直しが不可能な状態になる
- ベンダー人員の大量離脱
- 要件を理解している人の負担が増加し社内業務が滞る

プロジェクトを畳み、再出発

✓ 事例：基幹システムへのアジャイル開発の適用事例②

■再出発はビジネスモデルの変更に追従できるシステム開発を

きれいな現行システムを作りたいわけではない

ビジネスモデルに追従できるシステムを作る
オンプレからクラウドに移行

綿密な計画よりも小さく早く出す
各スペシャリストを揃えて、POが舵を取る

機能名	クラウド移行	問合対応	スケジュール登録	アルバイト講習会	契約処理	ダッシュボード	営業リスト	顧客管理
概要	現行システムのクラウド移行	入電ログ管理 ユーザーの名寄せ機能	スケジュール・シフトの登録	アルバイトの新規登録用モバイルサイト	口座自動割り当て 見積の電子化	PV数、サービスの予実表示	見込み客・解約予定客の見える化	ユーザーの追加情報を登録し、営業につなげる
体制	8人	10人	20人	20人	40人	40人	40人	40人
効率化ポイント	現行システムの延命とパフォーマンス改善	入力効率 50%UP	担当営業の業務効率30%UP	応募獲得数4倍に	営業効率 30%UP	担当営業の業務効率30%UP	営業効率 30%UP	情報共有の効率 30%UP

※定量的効果については推定

全員が懐疑的 ▷ 現場から入力しやすくなったとの声 | 執行役員の協力 | 新規案件の引き合い | 2チームから3チーム他部署から引き合い | データ活用へ意欲 | 情報の見える化 情報連携がしやすく | チームが積極的に

✓ 事例：基幹システムへのアジャイル開発の適用事例③

アジャイル

機能名	クラウド移行	問合対応	スケジュール登録	アルバイト講習会	契約処理	ダッシュボード	営業リスト	顧客管理
概要	現行システムのクラウド移行	入電ログ管理 ユーザーの名寄せ機能	スケジュール・シフトの登録	アルバイトの新規登録用モバイルサイト	口座自動割り当て 見積の電子化	PV数、サービスの予実表示	見込み客・解約予定客の見える化	ユーザーの追加情報を登録し、営業につなげる
体制	8人	10人	20人	20人	40人	40人	40人	40人
効率化ポイント	現行システムの延命とパフォーマンス改善	入力効率50%UP	担当営業の業務効率30%UP	応募獲得数4倍に	営業効率30%UP	担当営業の業務効率30%UP	営業効率30%UP	情報共有の効率30%UP

効果を見ながら次の一手を考える

新しい改善要望が現場からあがり、実装される

旧システムにはない改善はウォータフォールで実装されない

数年後にリリースした時点で初めて効果が分かる

ウォータフォール

工程	要件定義	基本設計	詳細設計		開発								テスト	リリース
概要／機能名	リプレイス＝「綺麗な旧システム」を作る			クラウド移行	問い合わせ対応	スケジュール登録	教師講習会	契約処理	ダッシュボード	営業リスト	顧客管理	業態の追加	大荒れ	保守＆バグ対応
体制	5人月	20人月	30人月		30人月	30人月	30人月	30人月			30人月	30人月	20人月	10人月

SHIFTのアジャイル開発への取り組み

✓ アジャイルのサイクルに追従する各種支援を実施。様々な役割が必要である

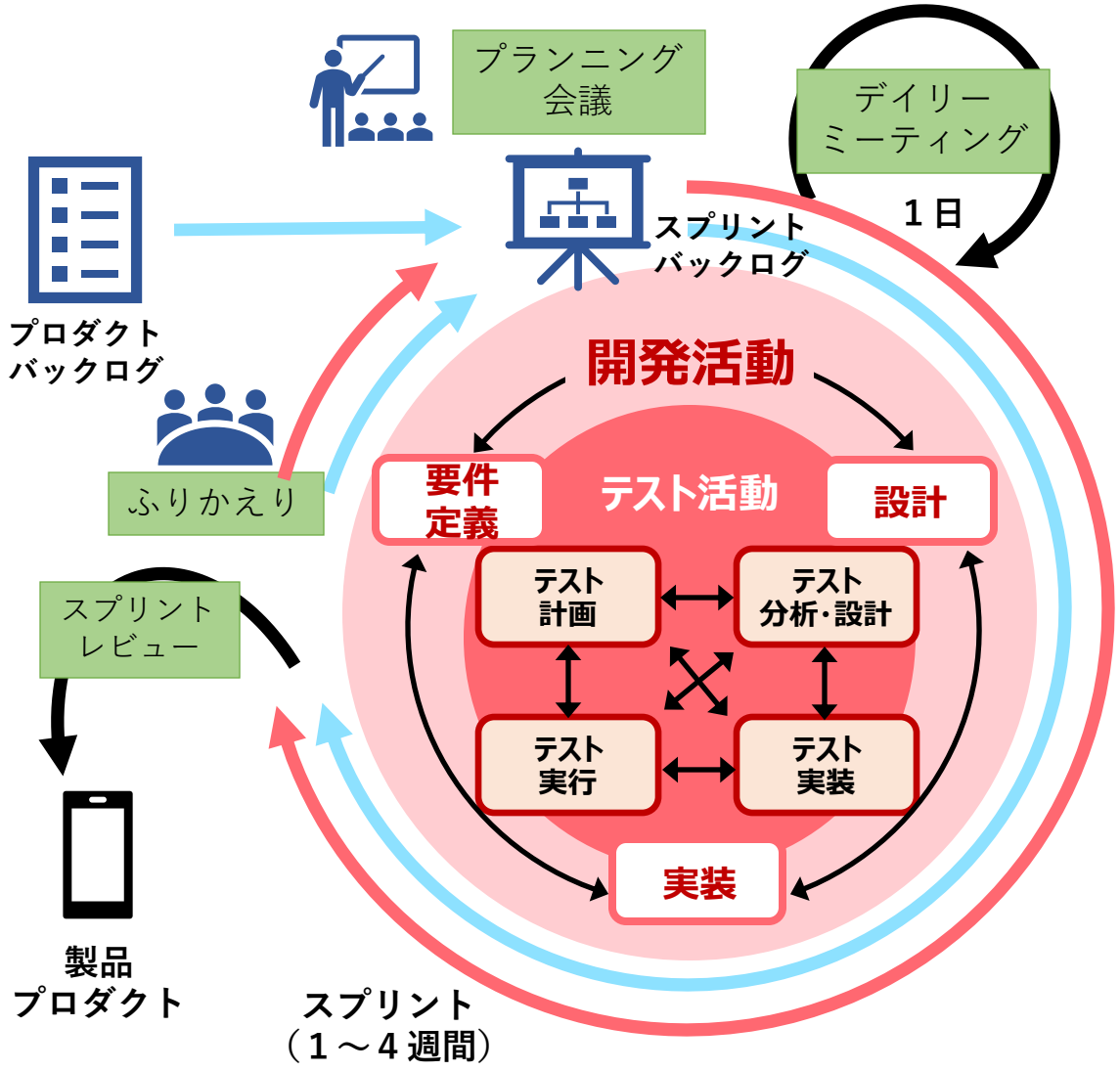
ステークホルダー
製品の利用者、出資者、管理職などの利害関係者

プロダクトオーナー
製品に対して責任を持ち、機能に優先順位をつける

–スプリント外支援–

プロダクトオーナー支援
プロダクトバックログをチームが開発できる状態に支援する

アジャイル導入支援
・導入における課題整理
・目的とロードマップの策定
・アセスメントの実施



–スプリント内支援–

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チーム
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。

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。

アジャイルコーチ
スクラムマスターを補佐し、課題の本質を見極め、解決を支援する。

開発チーム
・アジャイルを学習済の開発者がプロダクトを製作

UI／UXデザイナ
・UI／UXを中心としたデザインを製作

インフラ支援
・オンプレ、クラウドの環境を製作

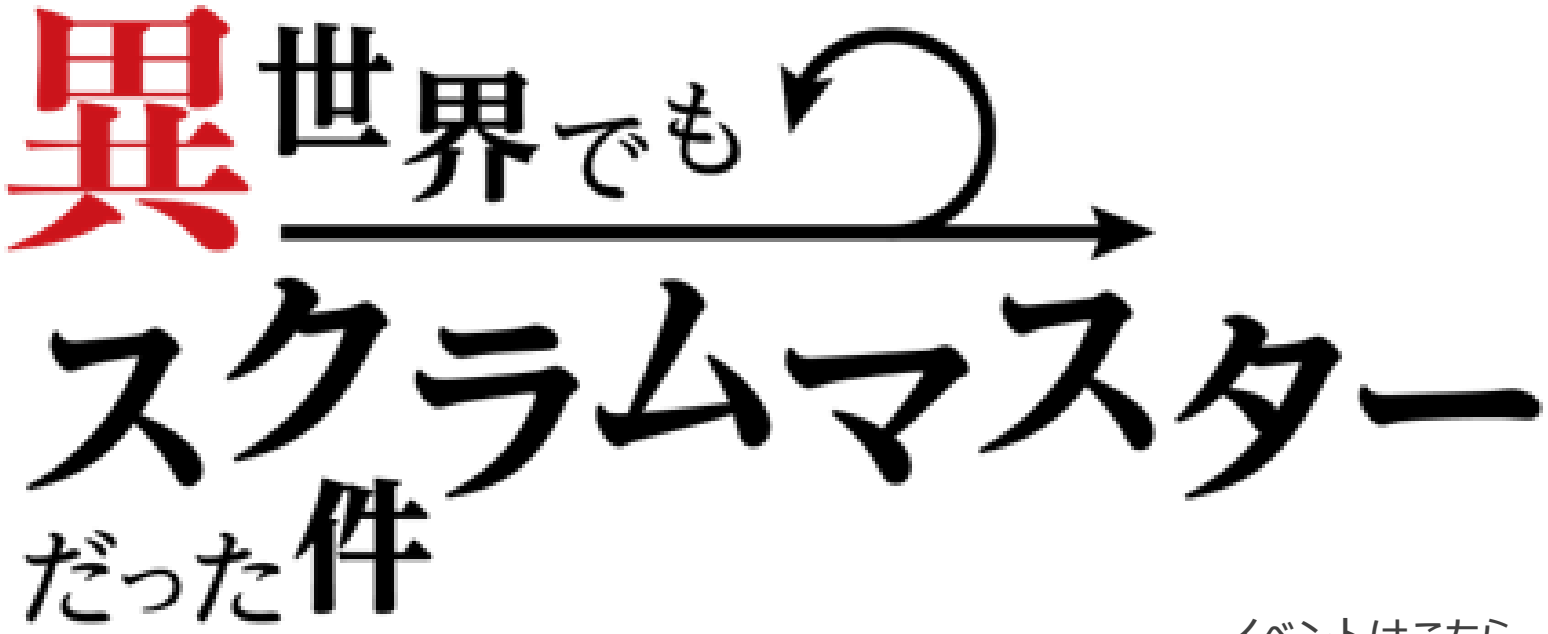
✓ 勉強会も週の中でそれぞれ開催し、学べる環境も

部内の週次勉強会

月曜	火曜	水曜	木曜	金曜
DevOps読書会	ブログもくもく会	アジャイルQAギルド	スクラムマスターサロン	テスト勉強会
	CI／CD勉強会	異世界スクラム運営	ブログ週次定例	
			アジャイル虎の穴	
			QANight（月1）	

これらの他にも、不定期で開催される勉強会も存在する

異世界アジャイル：スクラムマスターが異世界転移！？



イベントはこちら

漫画はこちら

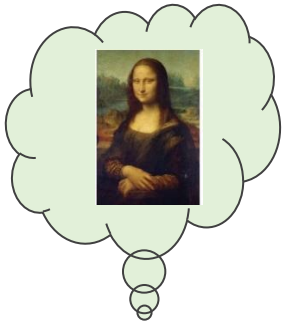
マンガ制作：株式会社シンフィールド

アジャイル開発とは？

アジャイル開発とは？

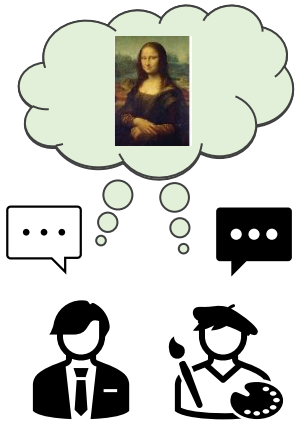
✓ いままでの開発手法と変わったこと

最初にほしいと思ったもの



ステークホルダー

ウォーターフォール開発



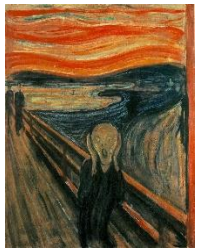
完成

すべてが出来上がるまで全体像を見ることができない

要求の変更が大きなインパクトになる

アジャイル開発

完成



完成

時間の経過やステークホルダーからのフィードバックによって、最初にほしいと思ったものが変わる。

完成

それぞれのタイミングで売れる絵を描きあげていく

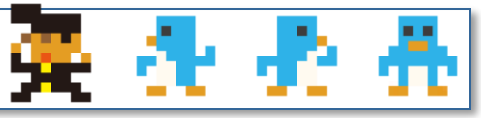
1つ1つのイテレーションごとに目に見える成果を確認できる

要求の変更ではなく次の要求になる

※パブリックドメインの絵画を使用しております

アジャイル開発とは？

✓ 絵を中心に見に行くための移動や絵を見た後の行動が存在。その活動も含めて計画する



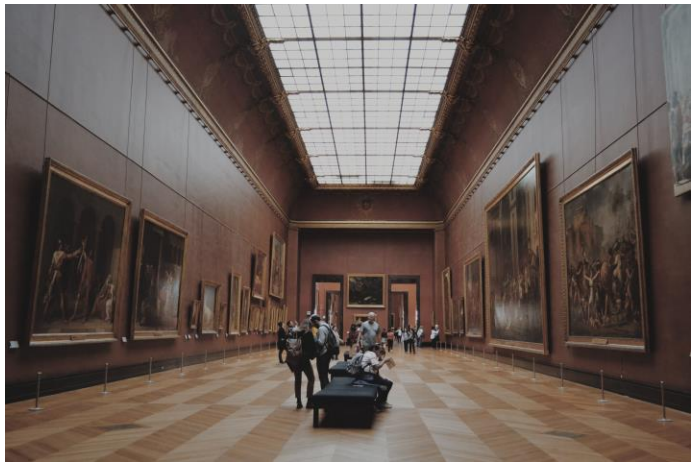
■システム開発からビジネス開発となる。これが今までの開発とは大きく異なる点である

■アジャイル開発の範囲

絵を見に行くための行動

絵を見た後の行動

■ウォーターフォール開発の範囲



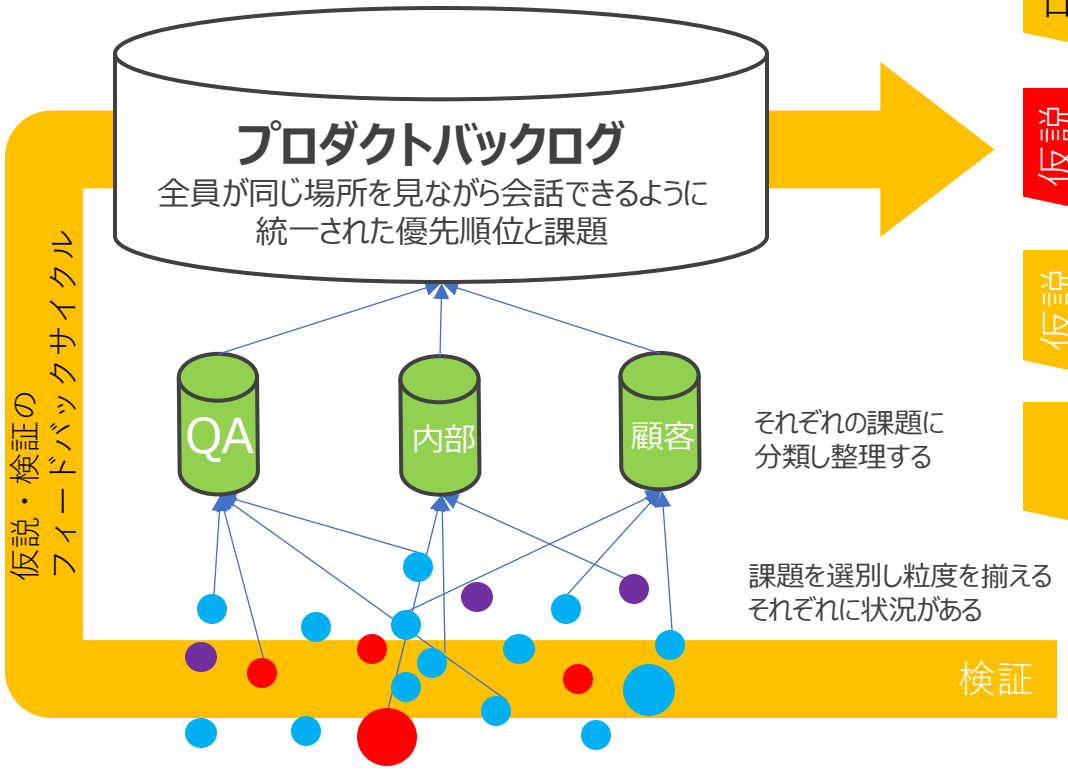
搬入できるか、額はどうするか…は検討している

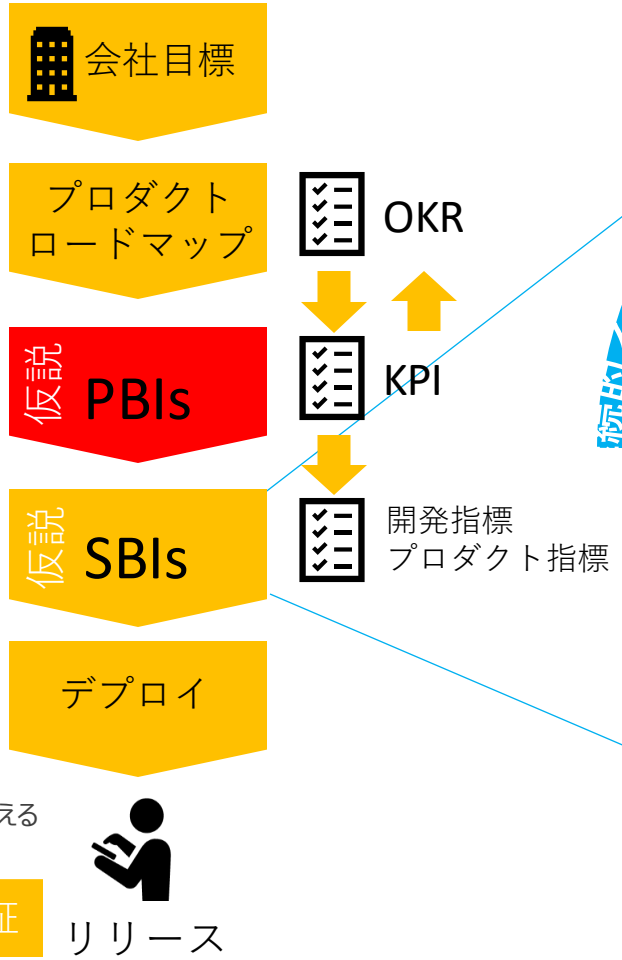
その常識、変えてみせる。 SHIFT

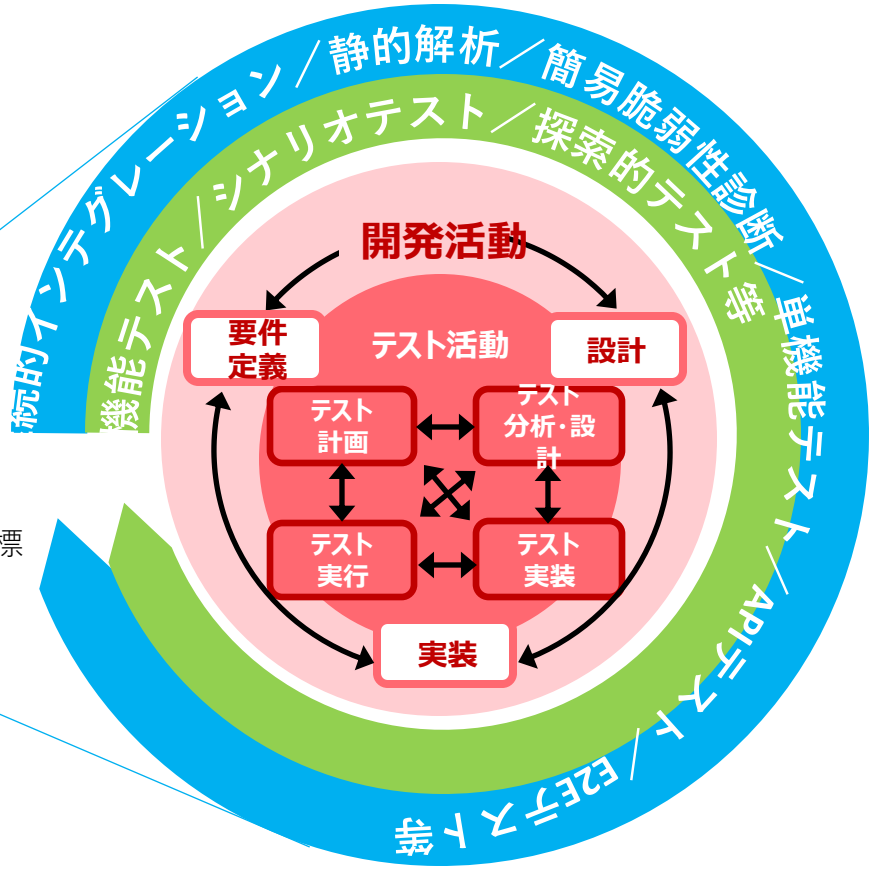
✓ プロダクトバックログを中心にコミュニケーションが必要

■ 適切な集約管理

- ✓ 書き出しましょう（取得）
- ✓ 同じ場所で管理しましょう（集約）
- ✓ 分けて並べましょう（分別）
- ✓ 優先度をつけましょう（優先）
- ✓ 誰もがいつでも簡単に見れる状態に







様々な要望、要件、課題が上がる

✓ 1週間のスケジュールと役割

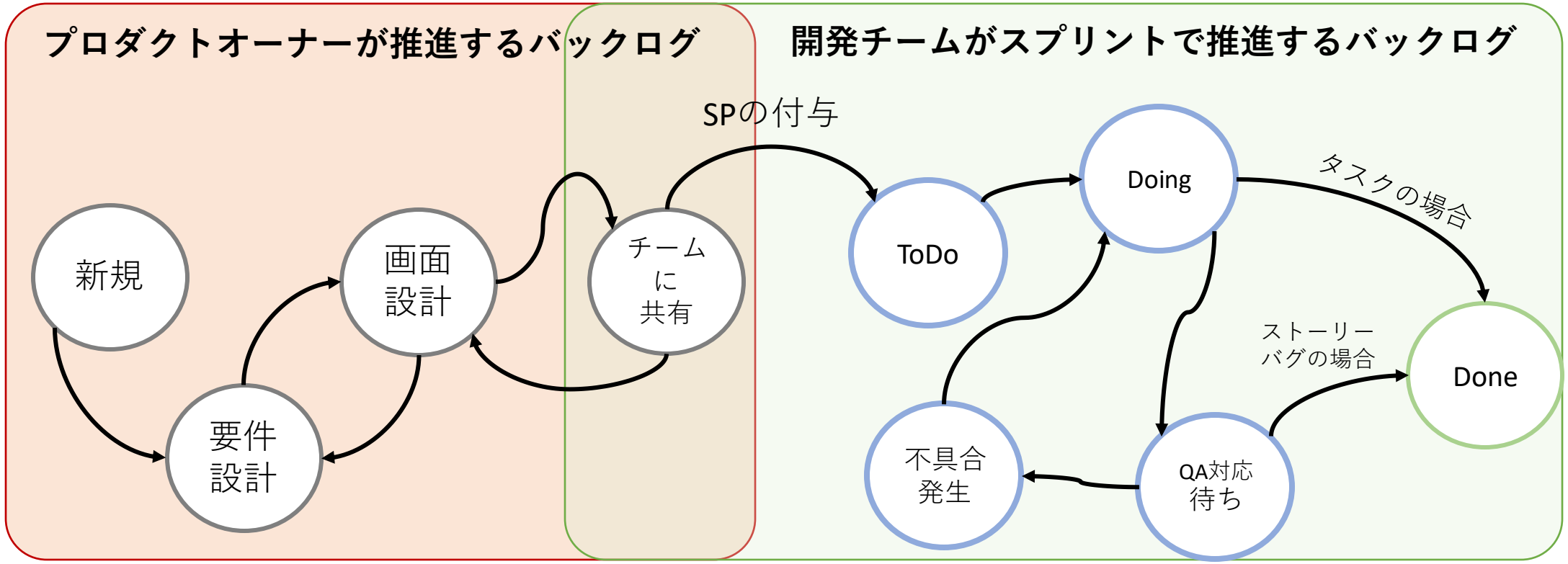
ロール	Day1	Day2	Day3	Day4	Day5
営業、CS		要件・課題検討			
PO	スプリントプランニング	プランニングポーカー リファインメント	プロダクトバックログ読み合せ	プロダクトバックログ読み合せ	リリース作業
開発エンジニア		デイリースクラム	プランニングポーカー リファインメント	プランニングポーカー リファインメント	レトロスペクティブ
QAエンジニア			デイリースクラム	デイリースクラム	スプリントレビュー
アジャイルコーチ					

スプリント内の活動　　次のスプリントのための活動

その常識、変えてみせる。 SHIFT

✓ どうやってリリースするの？ ステータス管理が重要

バックログの種類
- ストーリー
- タスク
- 不具合



新規：チケットを作成した状態
要件設計：ペルソナ、ユースケースの作成した状態
全体像を定義した状態
画面設計：画面イメージを作成した状態
チームに共有：開発チームに説明できる状態

ToDo：スプリントバックログに入れることができる状態
ストーリーポイントが付与されている状態
Doing：進行中
QA対応待ち：QA中
不具合発生：対応するBugを対応する必要がある状態
Done：スプリントレビューでPOに見せてリリースできる状態

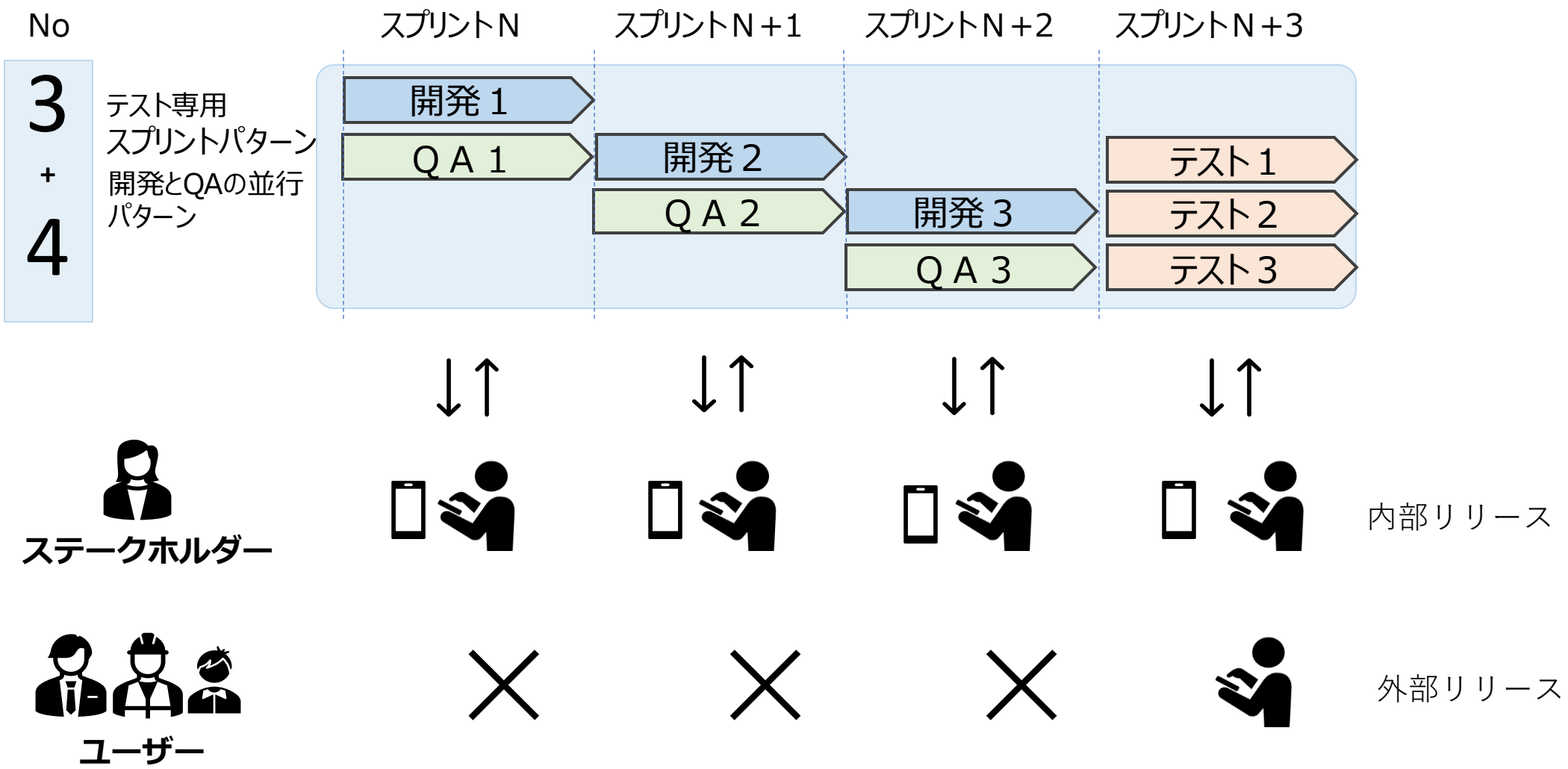
その常識、変えてみせる。 SHIFT

✓ どうやって協力するの？

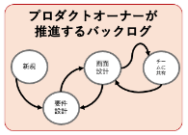
No		スプリントN	スプリントN＋1	スプリントN＋2	スプリントN＋3	平均／最小値 リリースサイクル
1	体系的なテストが実施されていないアジャイル開発	開発１	開発２	開発３		11.11%／10.93% 1サイクル
2	スプリント周回遅れテストパターン	開発１	テスト１ 開発２	テスト２ 開発３	テスト３	6.15%／2.85% 2サイクル
3	テスト専用スプリントパターン	開発１	開発２	開発３	テスト１ テスト２ テスト３	18.57%／3.46% テスト中止 4サイクル
4	開発とQAの並行パターン	開発１ ＱＡ１	開発２ ＱＡ２	開発３ ＱＡ３		2.09%／0.78% 1サイクル

その常識、変えてみせる。 SHIFT

✓ どうやって協力するの？



アジャイル開発とは？

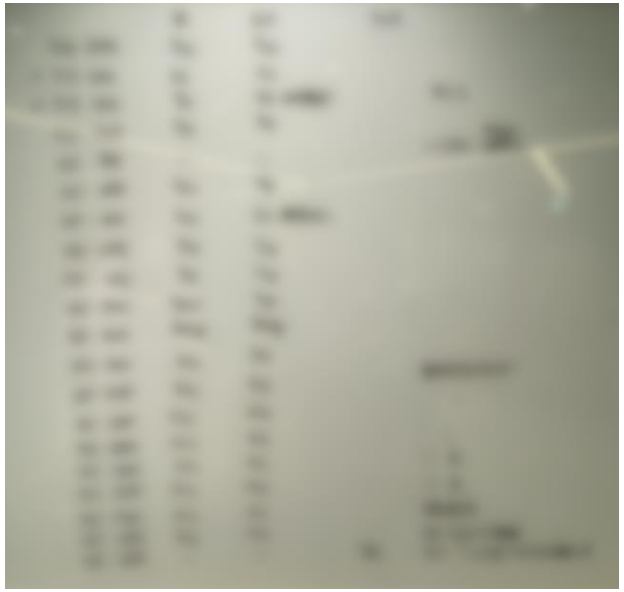


✓ 各イベントで何をしているの？

■バックログの作成／バックログリファインメント

プロダクトのバックログの新規作成、修正を通じて開発チームが作れるになっているかを確認

■バックログ読み合わせ

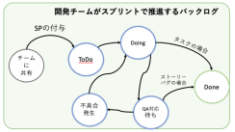


一つ一つのバックログをPOから開発、QAに説明し、対応可能か必要なものがないかを確認

■バックログ積み上げ

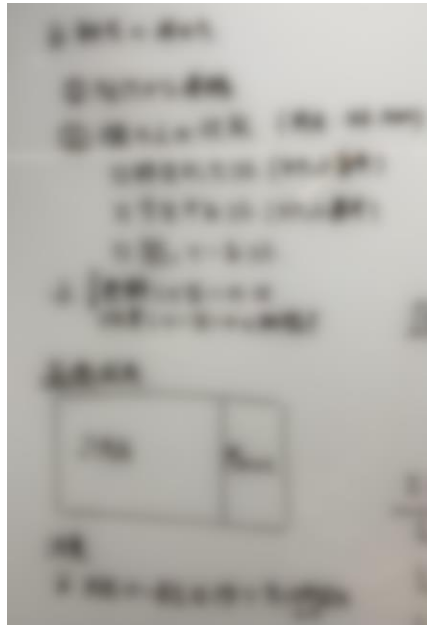
開発着手待ち

開発可能となり、ToDoとなるこの状態をReadyであると定義

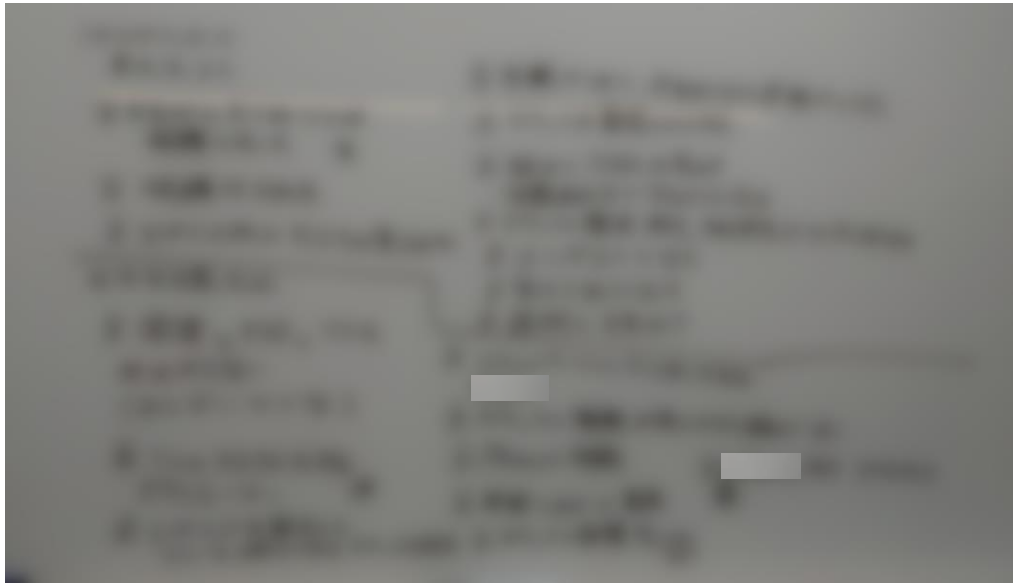


✓ 各イベントで何をしているの？

■各種イベントの進行／狙いの確認



朝会の進め方を明確にする



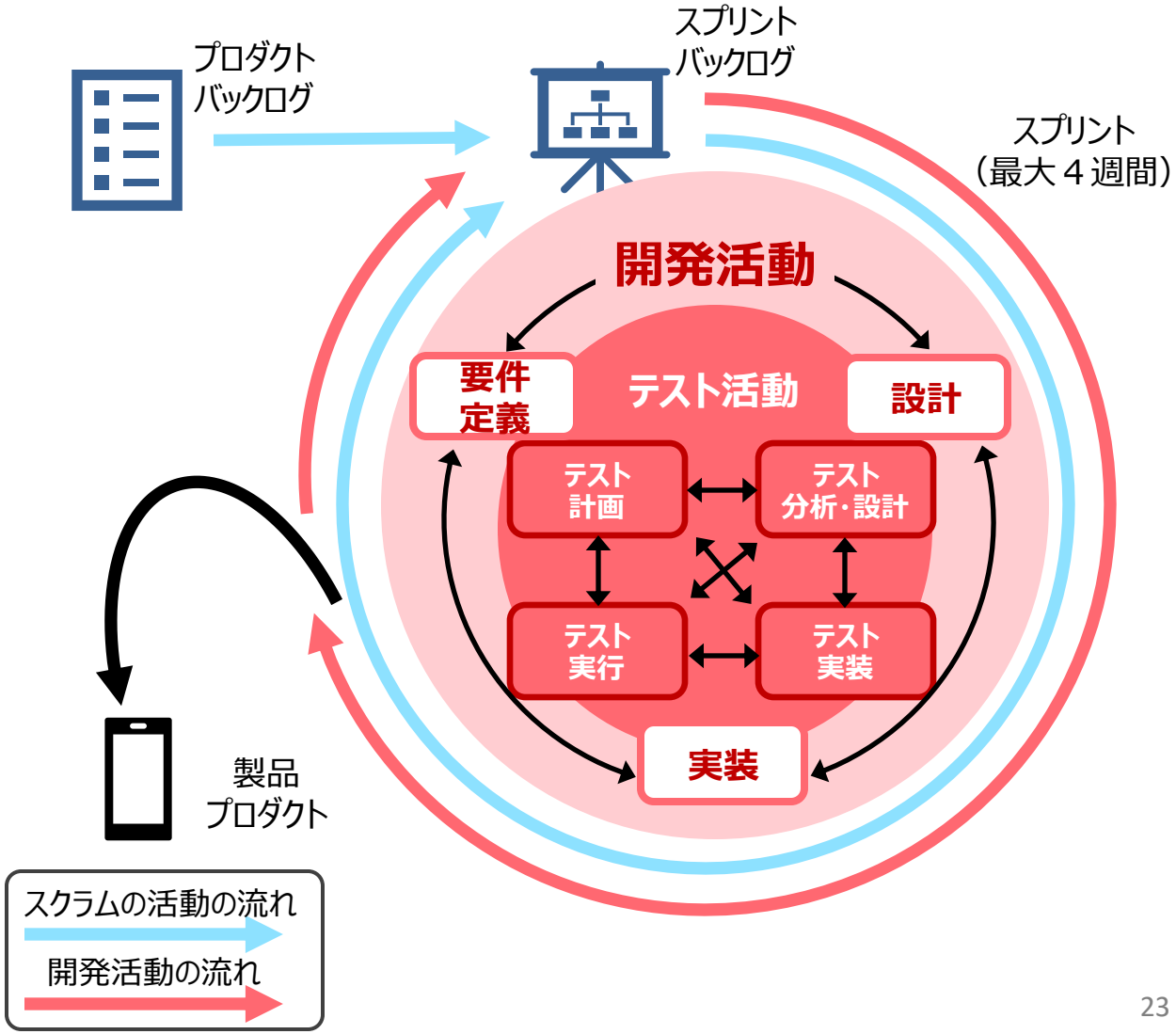
レトロスペクティブでのテーマ選定と進行、次のアクションまでのファシリテートを行う

出来上がったチームでは、今までとの開発スタイルと大きく変化するため、教科書通りの進め方では消化不良を起こすことが多い。
そのため、シンプルに実施して、必要に応じてツールやプロセスを導入していくのがよい。

品質活動 と 開発の関わり方

品質活動 と 開発との関わり方

✓ 役割分担が変わったが、テストの範囲は変わらない。



■変わったこと
- 開発活動の中の一つ。
- フェーズではなく常に行われる活動
- 作業単位の分担ではない

■変わらないこと
- テスト活動は開発活動であること
- 計画～実行は必要
- チーム全員が意識する活動

品質活動 と 開発との関わり方

SQF SHIFT Quality Framework　その常識、変えてみせる。SHIFT

✓ あらゆる手段で品質を確保 3STEPをぐるぐるとイテレーションを回す

スプリント外での活動

STEP1
テスト戦略

何をするか？
どうやるか？

スプリント内での活動

STEP2
テスト設計～実行

実際にやってみる

STEP3
テストレポート

どうだったか
評価する

※社内ではもっと細かく定義しています

SQF SHIFT Quality Framework

その常識、変えてみせる。SHIFT

✓ あらゆる手段で品質を確保 3STEPをぐるぐるとイテレーションを回す

■検討すること
・テストタイプ
・チームの誰が担当するか
・テストの手段
・それぞれのバックログに適用してみる

※社内ではもっと細かく定義しています

品質活動 と 開発との関わり方

SQF SHIFT Quality Framework　その常識、変えてみせる。SHIFT

✓ あらゆる手段で品質を確保 3STEPをぐるぐるとイテレーションを回す

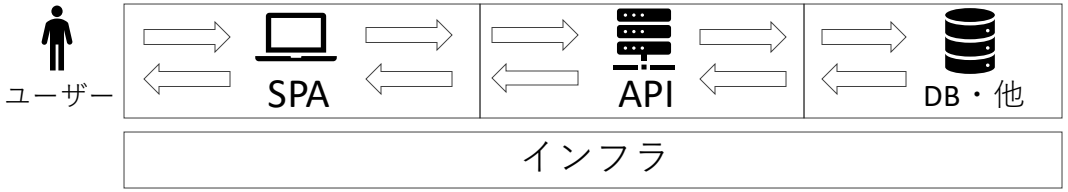
スプリント外での活動

STEP1
テスト戦略

何をするか？
どうやるか？

アーキテクトのモデルに則ったテストレベルを設けることで、障害を次のテストレベルに持ち越さない網羅的にテストを実施する。

■アーキテクチャーの概略図



No	テストレベル	マイクロサービス(連携を含む)	主な自動化ツール	SPA	API	DB	DB 外部API 等	テストアプローチ 手動 スクリプト	手動 探索的	自動 スクリプト	自動 Mock
1	単体テスト	SPA	Karma	WB	-	-	-	×	–	◎	なし
2		API	JUnit	-	WB	-	-	×	–	◎	なし
3		DB	JUnit	-	-	WB	-	×	–	◎	なし
4	単機能テスト	SPA↔MOCK	Selenium	BB	MOCK	-	-	○	○	○	必要
5		API↔MOCK	Karate	-	BB	MOCK	-	×	–	◎	必要
6		DB↔MOCK	Karate	-	-	BB	MOCK	×	–	◎	必要
8	結合テスト	API↔DB↔MOCK	Karate	-	BB		MOCK	×	–	◎	必要
9		SPA↔API↔DB↔MOCK	Selenium	BB			MOCK	◎	○	△	必要
10	システムテスト		Selenium	BB				◎	○	△	必要
11	受入テスト		Selenium	BB				-	◎	×	なし
12	性能テスト		Jmeteer	BB				×	△	○	なし
13	セキュリティテスト										

WB:ホワイトボックステスト BB:ブラックボックステスト

■検討すること
- ・テストタイプ
- ・チームの誰が担当するか
- ・テストの手段
- ・それぞれのバックログに適用してみる

※社内ではもっと細かく定義しています

品質活動 と 開発との関わり方

SQF SHIFT Quality Framework　その常識、変えてみせる。SHIFT

✓ あらゆる手段で品質を確保 3STEPをぐるぐるとイテレーションを回す

スプリント外での活動

STEP1
テスト戦略

何をするか？
どうやるか？

スプリント内での活動

STEP2
テスト設計～実行

実際にやってみる

STEP3
テストレポート

どうだったか
評価する

■ 検討すること
- ・テストタイプ
- ・チームの誰が担当するか
- ・テストの手段
- ・それぞれのバックログに適用してみる

■ 実施すること
- ・スプリント内での活動を定義
- ・テストの方針決定
- ・テスト設計／実行
- ・不具合のレポート／修正確認

※社内ではもっと細かく定義しています

品質活動 と 開発との関わり方

SQF SHIFT Quality Framework　その常識、変えてみせる。SHIFT

✓ あらゆる手段で品質を確保 3STEPをぐるぐるとイテレーションを回す

スプリント内での活動

STEP2
テスト設計～実行

実際にやってみる

■実施すること
・スプリント内での活動を定義
・テストの方針決定
・テスト設計／実行
・不具合のレポート／修正確認

■手動テスト

仕様書ベース

メリット

- 曖昧な仕様でも補完してテスト実施ができる
- デザインや挙動に対しても確認ができる
- ユーザーの心理を想像しながら実施できる
- ブラックボックス、グレーボックスのテストが中心

デメリット

- 単体テストや結合テストには向かない
- 実施には時間がかかる
- 人によって能力に差がある

■探索的テスト

システムベース

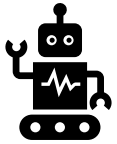
メリット

- テスト観点リストとタイムボックスを用いて実施するため、時間が明確に実施できる
- 学習＆ふりかえりを短いサイクルで実施するため、業務知見がなくても実施可能

デメリット

- タイムボックスが存在するのでシステムの品質に左右されやすく十分なテストを行うことができないことがある
- 学習＆ふりかえりのため、再現性がない

■自動テスト



仕様書ベース

メリット

- 決められた手順に沿ったテスト
- 繰り返し実施するテスト
- 単体テスト～総合テストまで各種テストレベルに適用できる
- いつでも実施ができ、短時間でレポートを出力できる

デメリット

- 開発の進め方によっては壊れやすい
- プログラムの作りによっては壊れやすい
- 失敗時に切り分けが必要

※社内ではもっと細かく定義しています

品質活動 と 開発との関わり方

SQF SHIFT Quality Framework　その常識、変えてみせる。SHIFT

✓ あらゆる手段で品質を確保 3STEPをぐるぐるとイテレーションを回す

スプリント外での活動

STEP1
テスト戦略

何をするか？
どうやるか？

スプリント内での活動

STEP2
テスト設計～実行

実際にやってみる

STEP3
テストレポート

どうだったか
評価する

■ 検討すること
- ・テストタイプ
- ・チームの誰が担当するか
- ・テストの手段
- ・それぞれのバックログに適用してみる

■ 実施すること
- ・スプリント内での活動を定義
- ・テストの方針決定
- ・テスト設計／実行
- ・不具合のレポート／修正確認

■ 評価すること
- ・不具合の状況
- ・リスク／積み残しの課題
- ・テストの状況
- ・QAとしてのリリース可否

※社内ではもっと細かく定義しています

SQF SHIFT Quality Framework

その常識、変えてみせる。SHIFT

✓ あらゆる手段で品質を確保 3STEPをぐるぐるとイテレーションを回す

スプリント内での活動

STEP3
テストレポート

どうだったか
評価する

■評価すること
- 不具合の状況
- リスク／積み残しの課題
- テストの状況
- QAとしてのリリース可否

◇リリースレポートと合わせたテストレポートの作成

リリースレポート

- ◻ リリース日
- ◻ ブランチ　タグ名称
- ◻ テスト内容
  - ◻ テスト計画書 URL
  - ◻ テスト実施結果　テストレポートURL
- ◻ QA担当者
- ◻ リリース判断者(PO)

1 ・・・1

テストレポート(簡易)

- ◻ テスト対象
- ◻ テスト環境
- ◻ 対象のテストケース
  - ◻ 実施したテストケース
- ◻ テスト結果 サマリ
  テスト実行者によるリリース可否判定
- ◻ テスト結果 詳細
  - ◻ ステータス別のケース数
    (OK、NG、修正済み等)
  - ◻ スプリントで発生した障害
    - ◻ 障害一覧
      (発生数、発生率、対応状況など)
  - ◻ 過去スプリントとの比較
    (重要度別×障害率の推移など)
  - ◻ 障害に対する見解
    (発生状況、偏り、デグレード確認など)
- ◻ 今後に向けた活動
  アクションアイテム、
  次のスプリントに向けた改善提案など

1 ・・・N

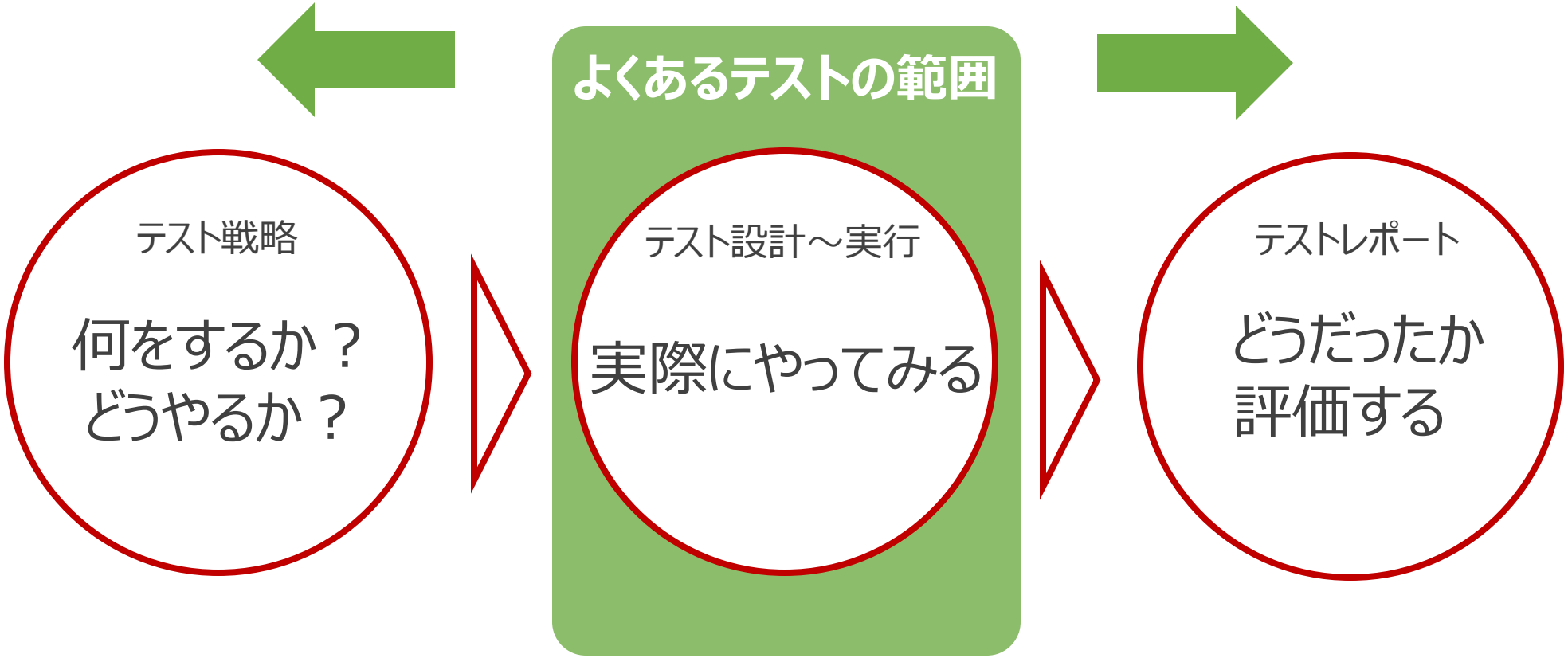
リリースレポート(個別)

- ◻ リリース日
- ◻ リリース内容
  変更点に関する概要/プロダクトバックログURL
- ◻ テスト実施の必要性
- ◻ QA担当者
- ◻ リリースに影響する障害の有無
  - ◻ 障害の発生条件&再現手順
  - ◻ 解決方法 、回避方法
- ◻ エンドユーザーへの影響が周知されているかどうか
- ◻ サポートへの影響が周知されているかどうか
- ◻ 記載者
- ◻ 記載日

※社内ではもっと細かく定義しています

✓ アジャイル開発で品質活動が変わった？

どうやって視点を広げるかがテーマになる



1問1答

アジャイル開発でイテレーション回して
リリースはまとめて出したいです

アジャイル開発でイテレーション回して
リリースはまとめて出したいです

しっかりと要件決めて動く
ウォーターフォール開発がうまくいきますよ

アジャイル開発やプロセスに興味はないですが早くリリースしたいです

アジャイル開発やプロセスに興味はないですが
早くリリースしたいです

しっかりと要件決めて動く
ウォーターフォール開発が早く出せますよ

ストーリーポイントの基準を教えてください

その常識、変えてみせる。SHIFT

ストーリーポイントの基準を教えてください

ポイント＝数値≠時間です
バックログのカテゴライズです

ストーリーポイントってなんですか？

ストーリーポイントってなんですか？

人類にはまだ早かったツール

その常識、変えてみせる。SHIFT

ストーリーポイントを振ったとおりにスプリントが進みませんでした

ストーリーポイントを振ったとおりにスプリントが進みませんでした

ポイントはチームで考えましたか？
一部のメンバーで振ったポイントでは
うまくいかないです

スプリント内で決めたバックログが消化されずに残ります

その常識、変えてみせる。SHIFT

スプリント内で決めたバックログが
消化されずに残ります

バックログは事前に共有済みですか？
Readyの状態でしたか？

Readyな状態ってどういう状態ですか？

Readyな状態ってどういう状態ですか？

チーム全員が着手可能な状態です
チームに共有して懸念点を洗い出した状態

え？でもいつでもバックログの優先順位を変えていいんですよね？

え？でもいつでもバックログの優先順位を変えていいですよね？

スプリント内のバックログの変更はNGです
スプリント外のバックログの変更はOKです

ただし

え？でもいつでもバックログの優先順位を変えていいですよね？

バックログにはステータスがあってチームがReadyまではPOの仕事です

品質が安定しないです

品質が安定しないです

テスト戦略はありますか？
短いサイクルなので、品質を上げるいくつもの
施策が必要になります

ほかにも

品質が安定しないです

テストだけではなく
進め方やプラクティス 各種方法で
品質を高めています

そんなの急に言われても……

そんなの急に言われても……

SHIFTでは開発の
どんな相談でも乗ります!!
1度ご連絡を♪

スプリントレビューで成果を見せる相手は来ている全員ですよね？

スプリントレビューで成果を見せる相手は
来ている全員ですよね？

見せるのは全員でよいですが
リリースの判断はPOです

おわり

画像

https://www.irasutoya.com/

https://unsplash.com/

https://dotown.maeda-design-room.net/