

スプリント内でテストを完了させるには？

～アジャイルスクラム開発に参加したQAエンジニアの悩みと対策～

サイボウズ株式会社

渡邊 豊幹

渡邊 豊幹 (わたなべ ゆうき)

- 2011年 サイボウズ株式会社に新卒で入社
- 品質保証部(当時)に配属となり、以降QAエンジニアとして活動
- 現在はモバイルアプリを担当

いつでも どこでも チームとつながる
モバイル版アプリ

サイボウズ Office



目次

背景

対策と効果

課題

目次

背景

開発チーム紹介

チーム構成とQAエンジニアの業務

対策と効果

取り組み前の状況

取り組んできたこと

問題

課題

目次

背景

開発チーム紹介

チーム構成とQAエンジニアの業務

対策と効果

取り組み前の状況

取り組んできたこと

問題

課題

チーム構成

	PM	Dev	QA	デザイナー	ライター
iOS	1人 *	4人	2人	1人	1人 *
Android	1人 *	4人	2人	1人	1人 *

* PMとライターはiOSとAndroidを1人で兼務

QAエンジニアの業務

	PM	Dev	QA	デザイナー	ライター
iOS	1人 *	4人	2人	1人	1人 *
Android	1人 *	4人	2人	1人	1人 *

* PMとライターはiOSとAndroidを1人で兼務

サイボウズのQAエンジニアは品質に関する組織的な取り組みを推進すると同時に、
テスト設計/実施も並行して行う

QAエンジニアの業務

	PM	Dev	QA	デザイナー	ライター
iOS	1人*	4人	2人	1人	1人*
Android	1人*	4人	2人	1人	1人*

* PMとライターはiOSとAndroidを1人で兼務

本セッションでは、テスト設計/実施作業に関してQAエンジニアが取り組んだ対策について発表

目次

背景

開発チーム紹介

チーム構成とQAエンジニアの業務

対策と効果

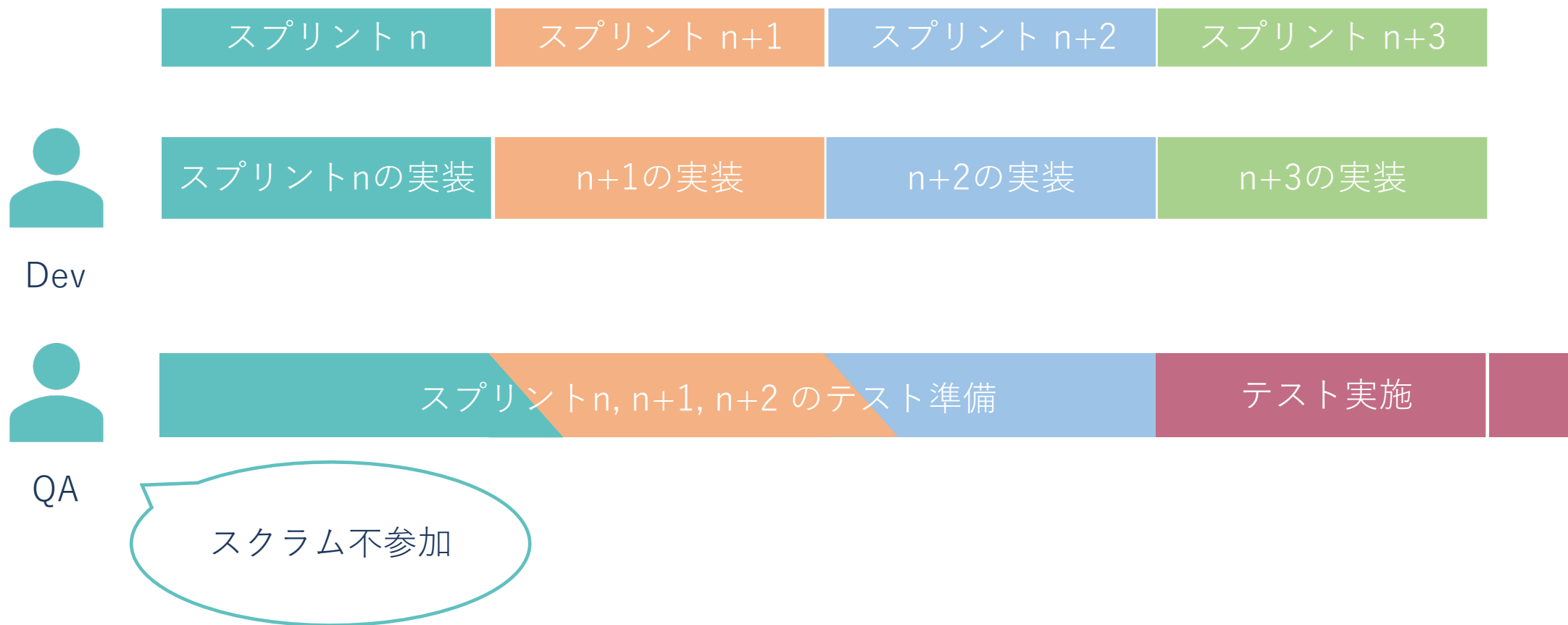
取り組み前の状況

取り組んできたこと

問題

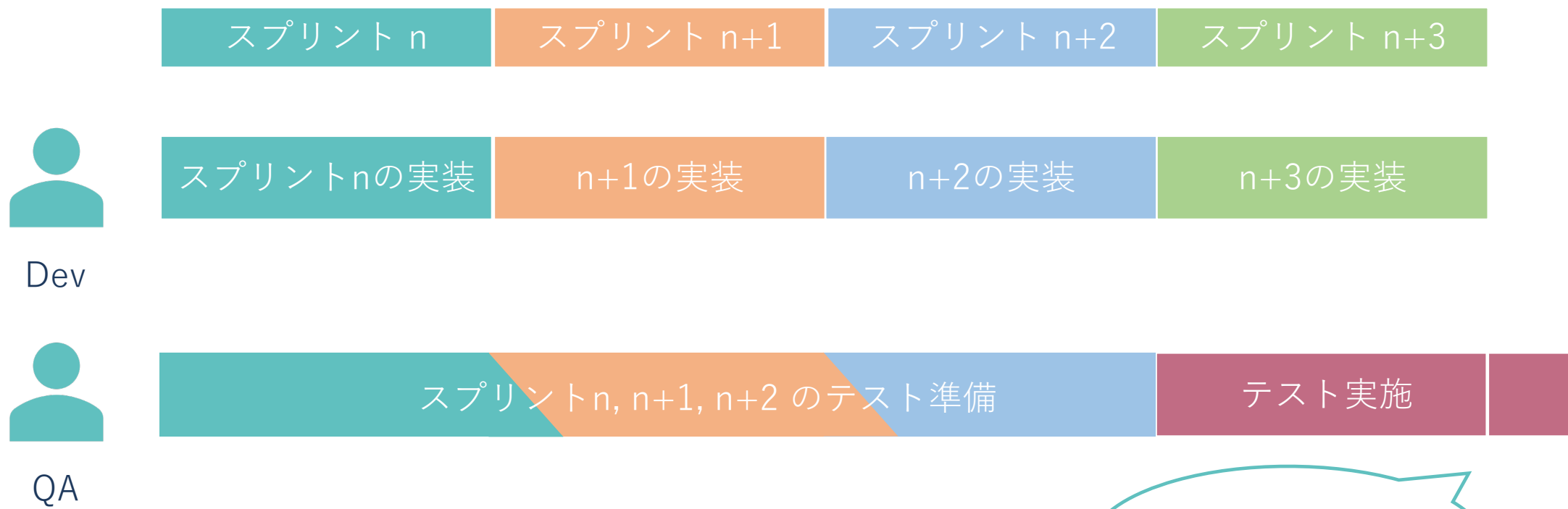
課題

取り組んできたこと



以前はDevのみがスクラム開発を行い、QAエンジニアは大きい単位で機能ごとにテストする体制

取り組んできたこと



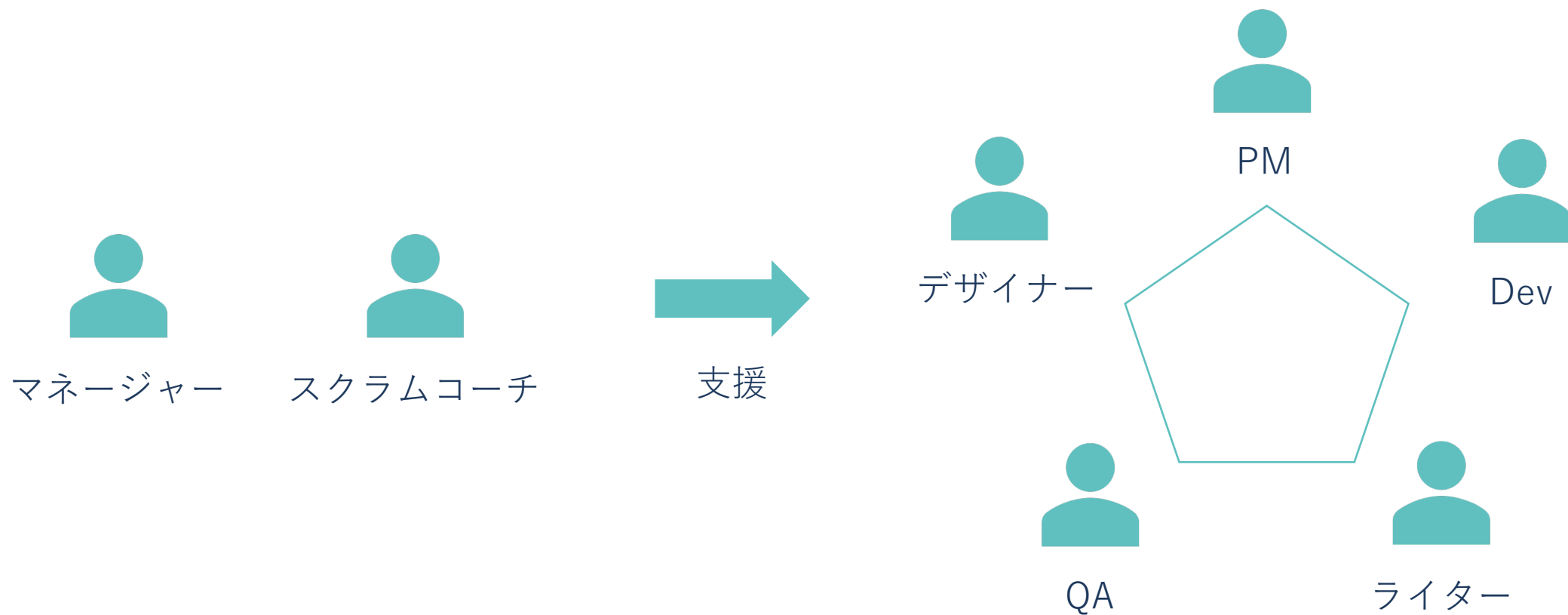
品質に対するフィードバックが数週間～数ヶ月に1度しかなく、リリース頻度も同様の間隔

取り組んできたこと



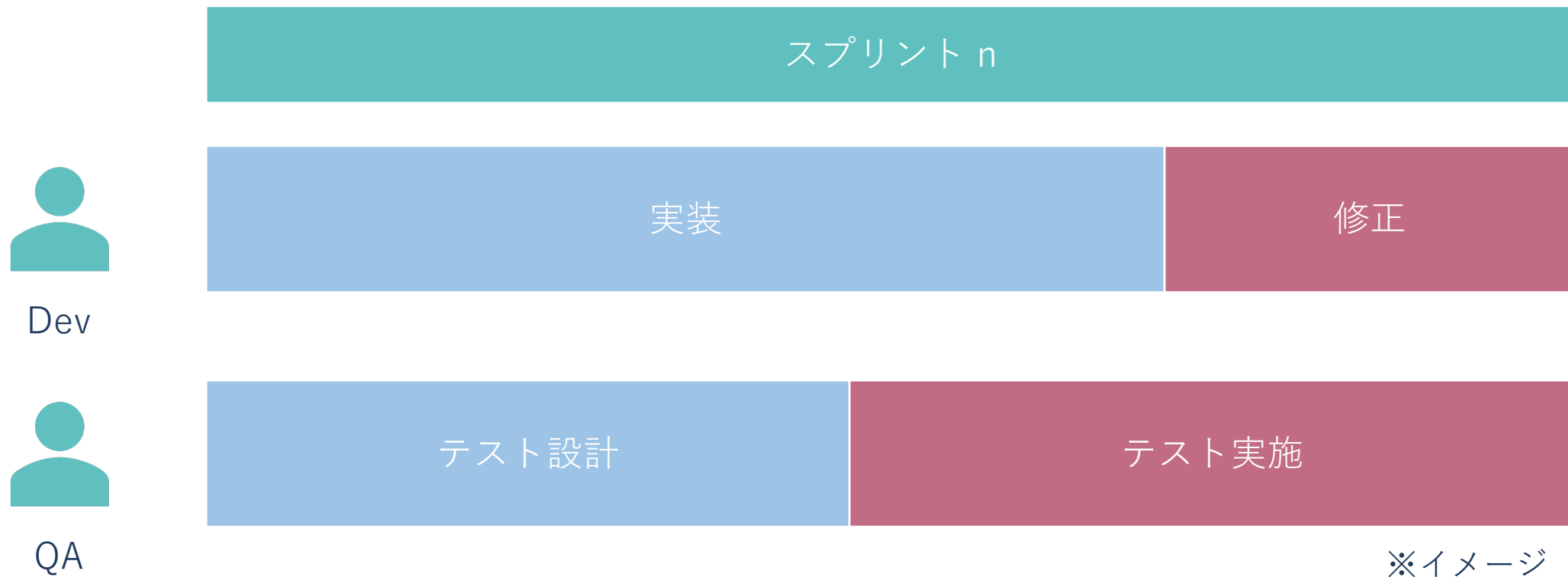
開発プロセス自体もスクラムを採用しているものの受発注のような関係性
振り返りなども職能ごとに別に開催している状態

取り組んできたこと



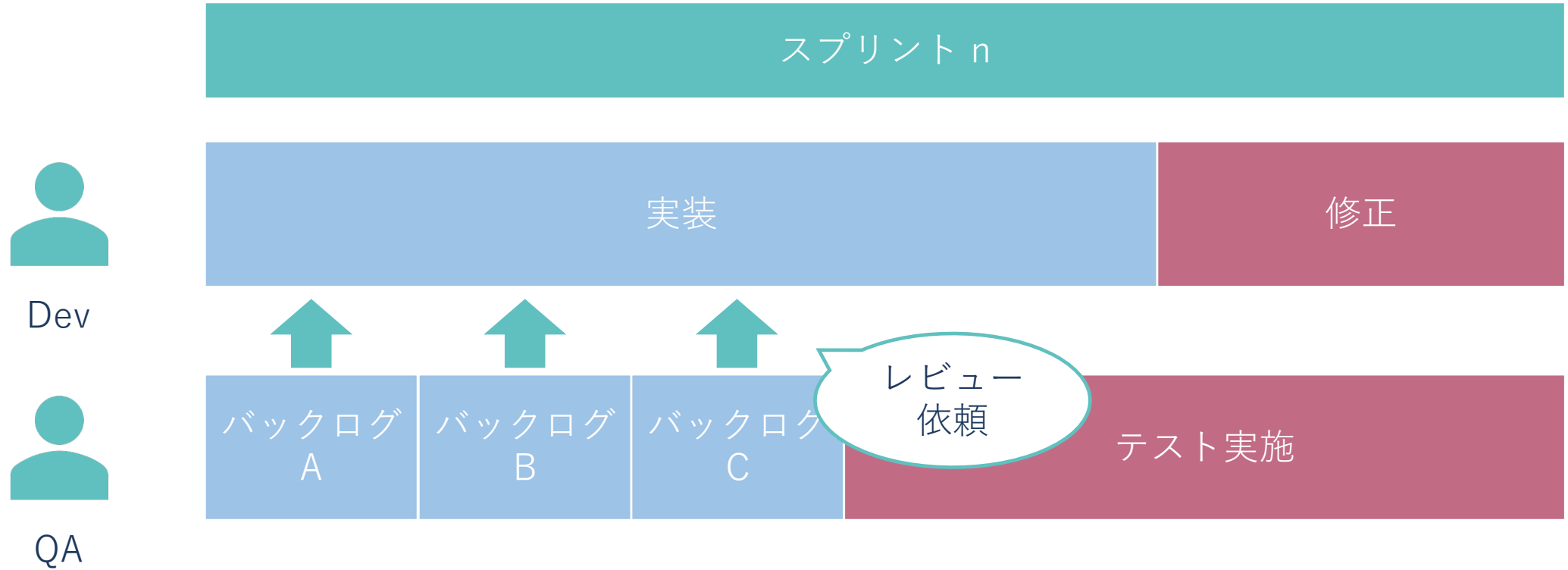
社内のスクラムコーチとマネージャーに支援してもらい、プロセスやミーティングを整理
QAエンジニアもスクラム開発チームの一員として活動を開始

取り組んできたこと



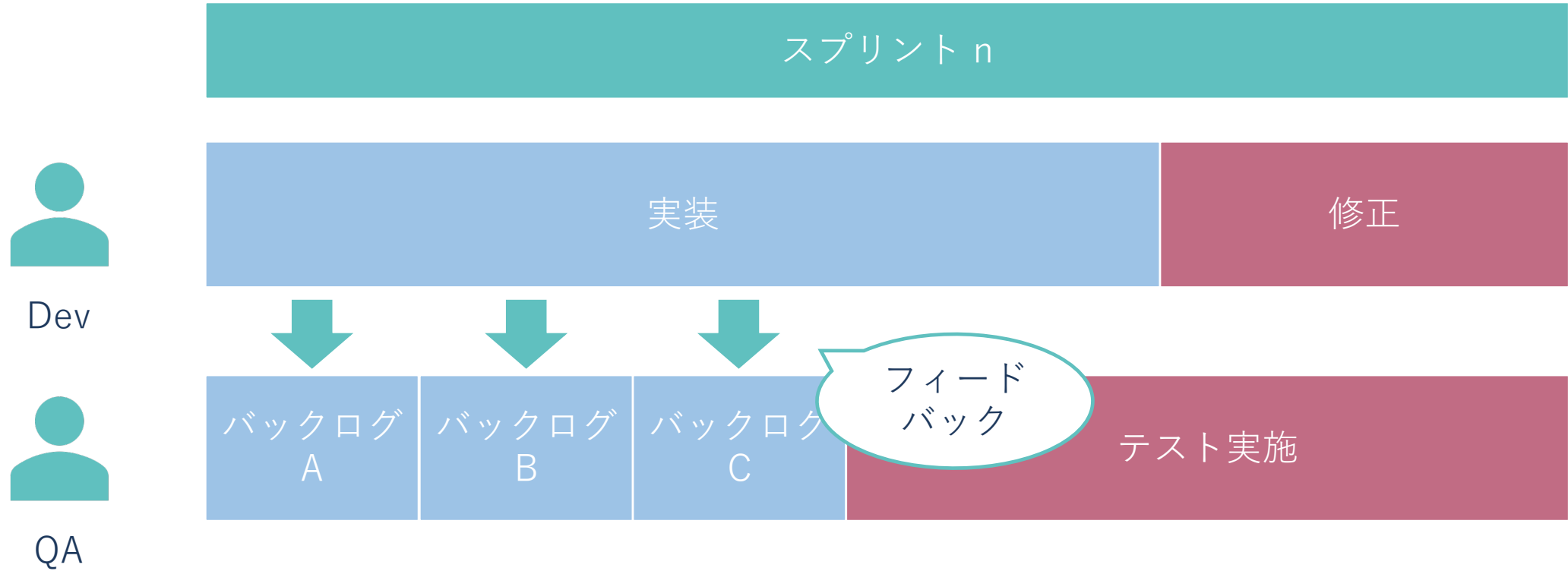
スプリント内でテスト設計とテスト実施を行うプロセスにし、
品質に対するフィードバックを早めていく取り組み

問題



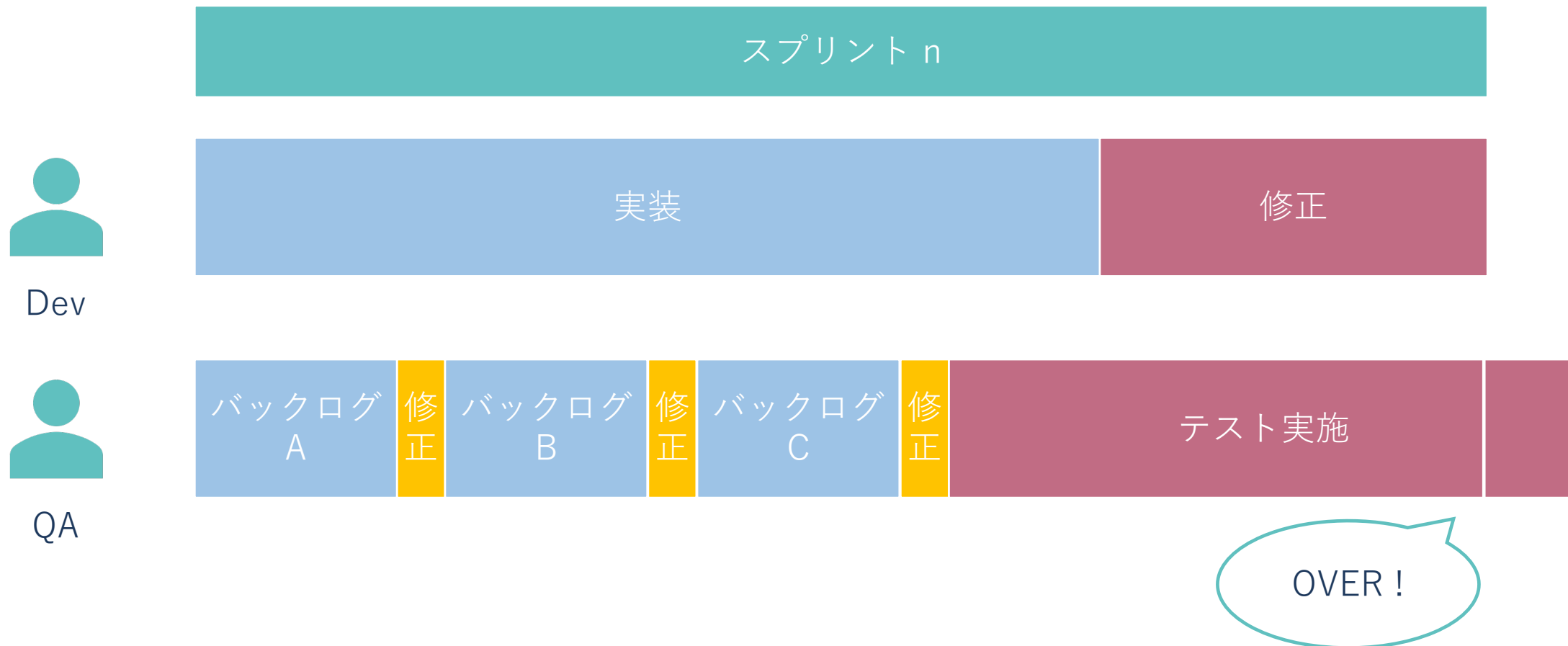
テスト設計が終わったタイミングで順次Devにレビューを依頼

問題



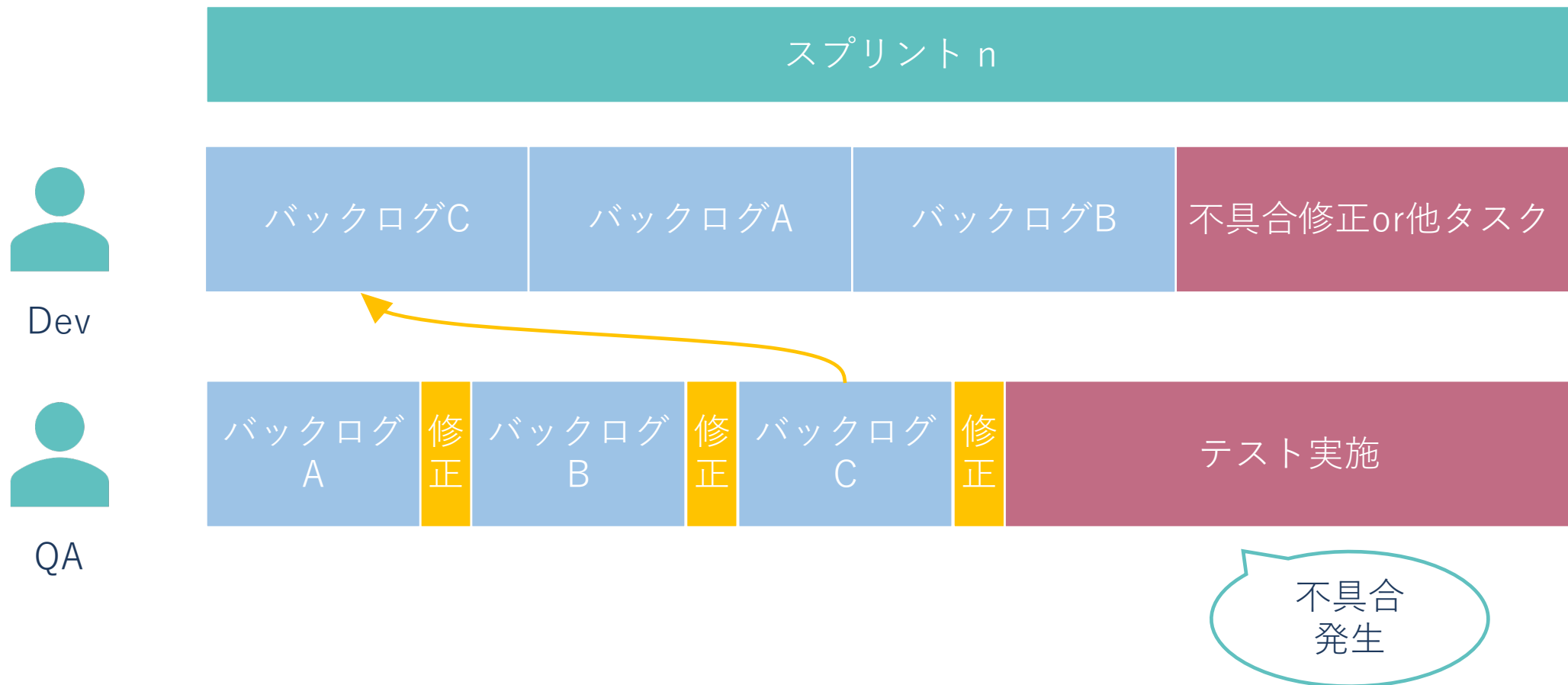
Devからフィードバックをもとに、必要があればテスト設計を修正

問題



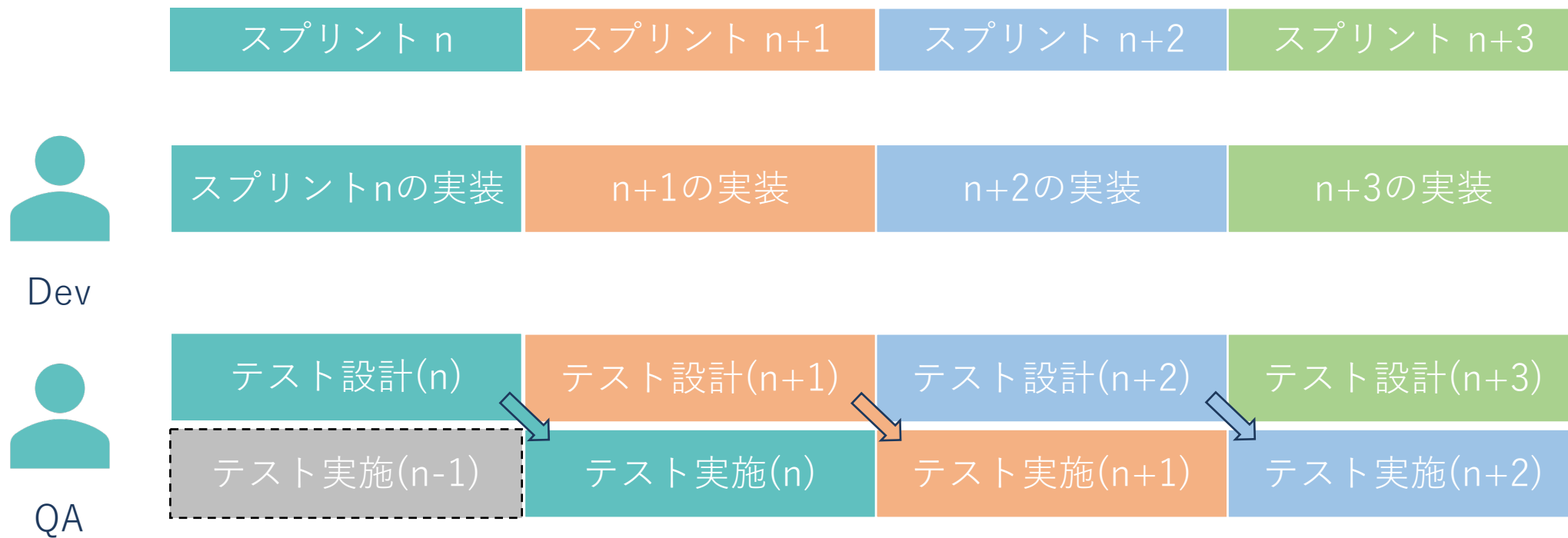
レビュー待ちや修正によりテスト実施が遅れ、スプリントレビューに間に合わない場合がある

問題



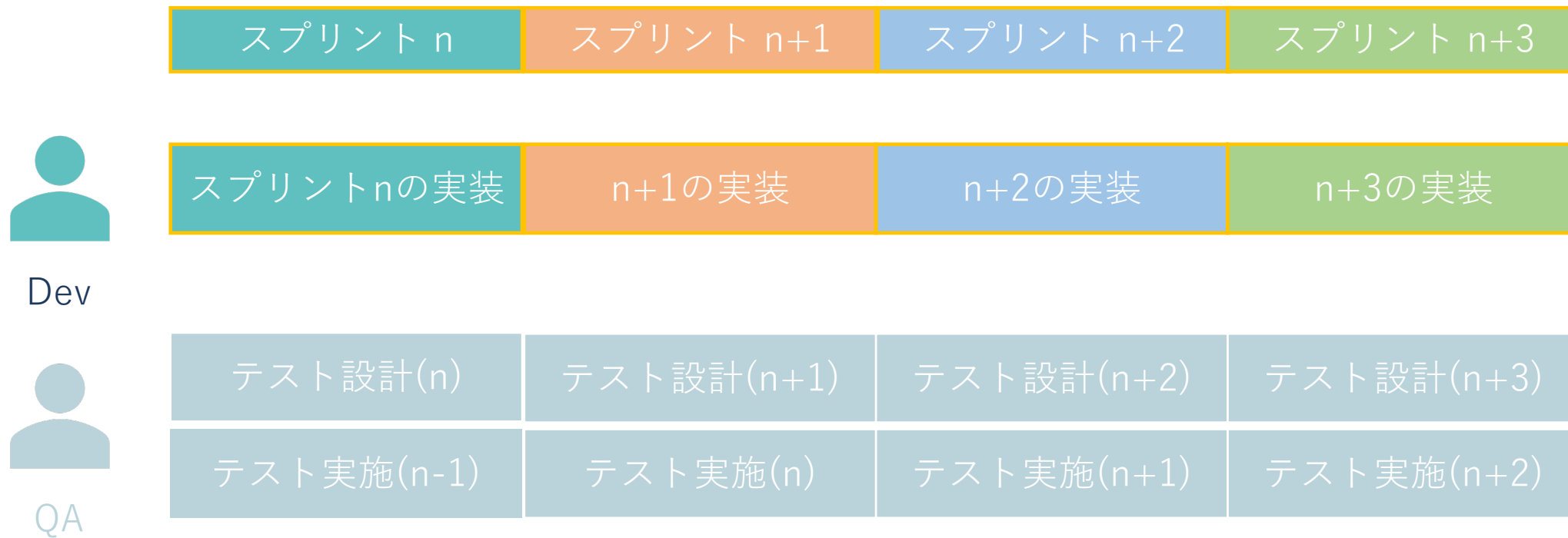
テスト設計を着手した段階ですでに実装済みのバックログがある状態で
実装前にテスト内容を共有すれば防げた不具合が発生し、本来不要な改修コストがかかる

問題



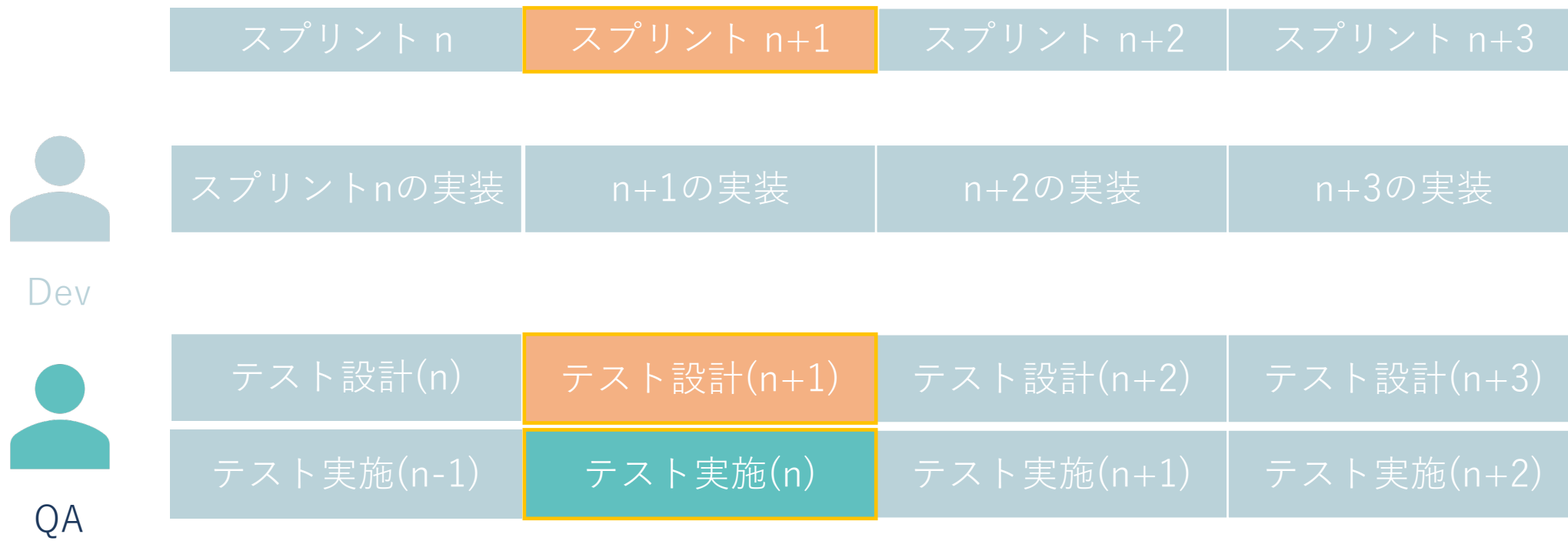
いつの間にか、QAエンジニアはDevと1週遅れでテストを実施している状態

問題



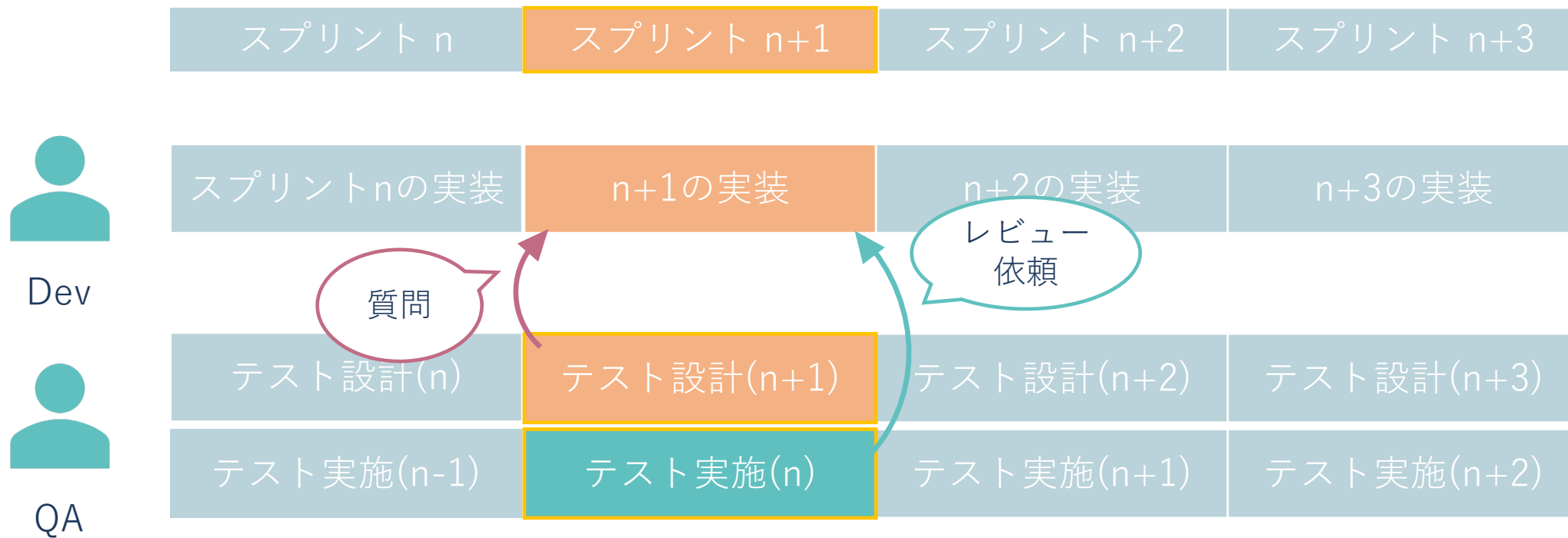
Devはスプリントに対応して実装している

問題



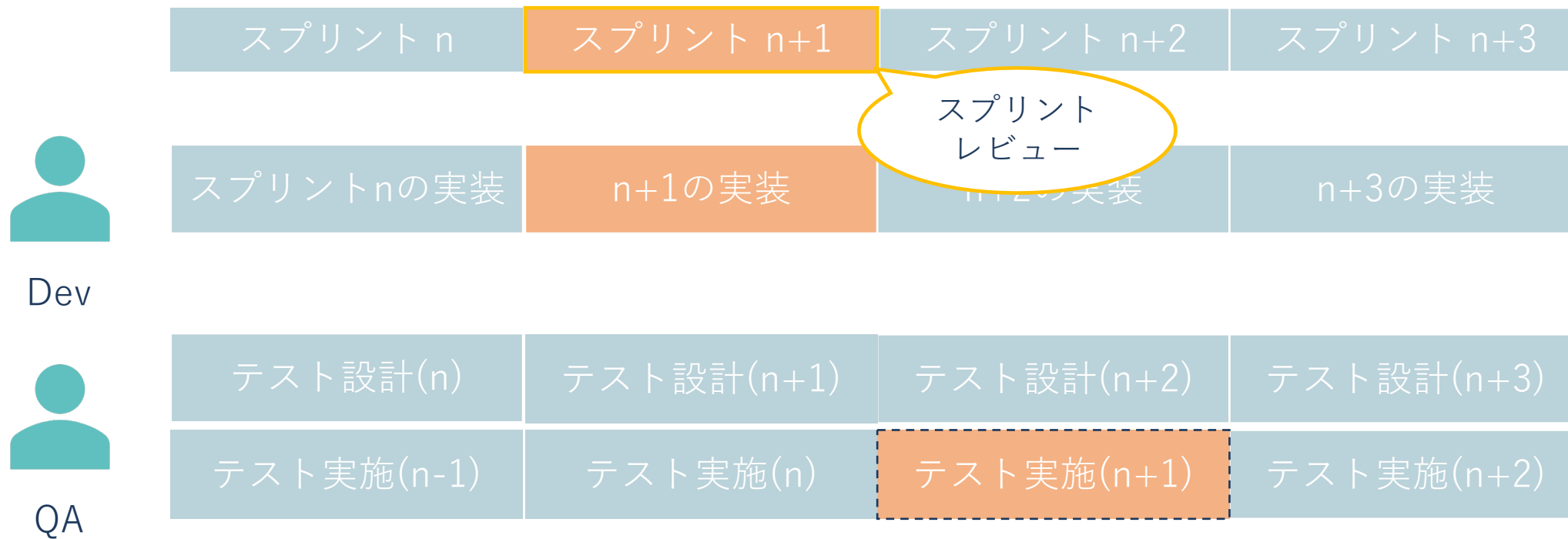
QAエンジニアは、1つ前のスプリントのテスト実施と、今のスプリントのテスト設計を行う

問題



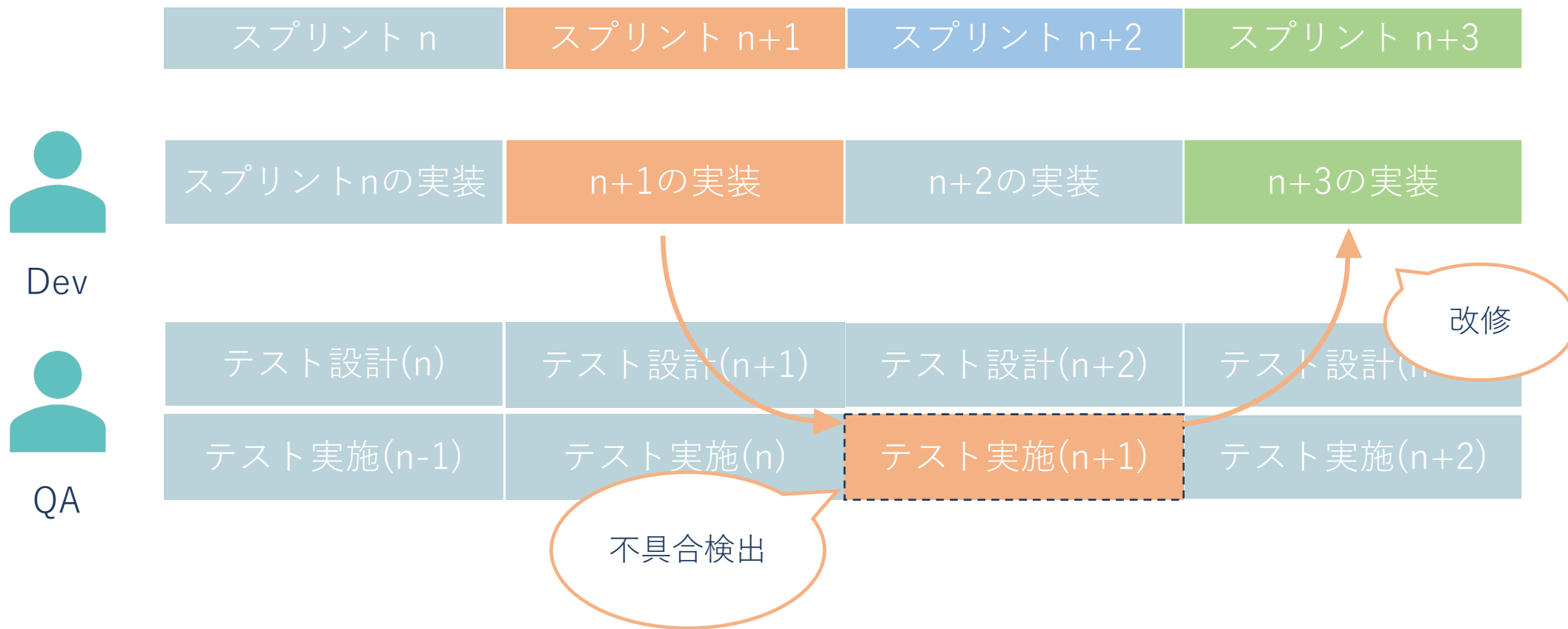
QAエンジニアからDevへ、前のスプリントの期待結果の問い合わせと、今のスプリントのテスト設計レビューをしている状態で、双方認知負荷が高い

問題



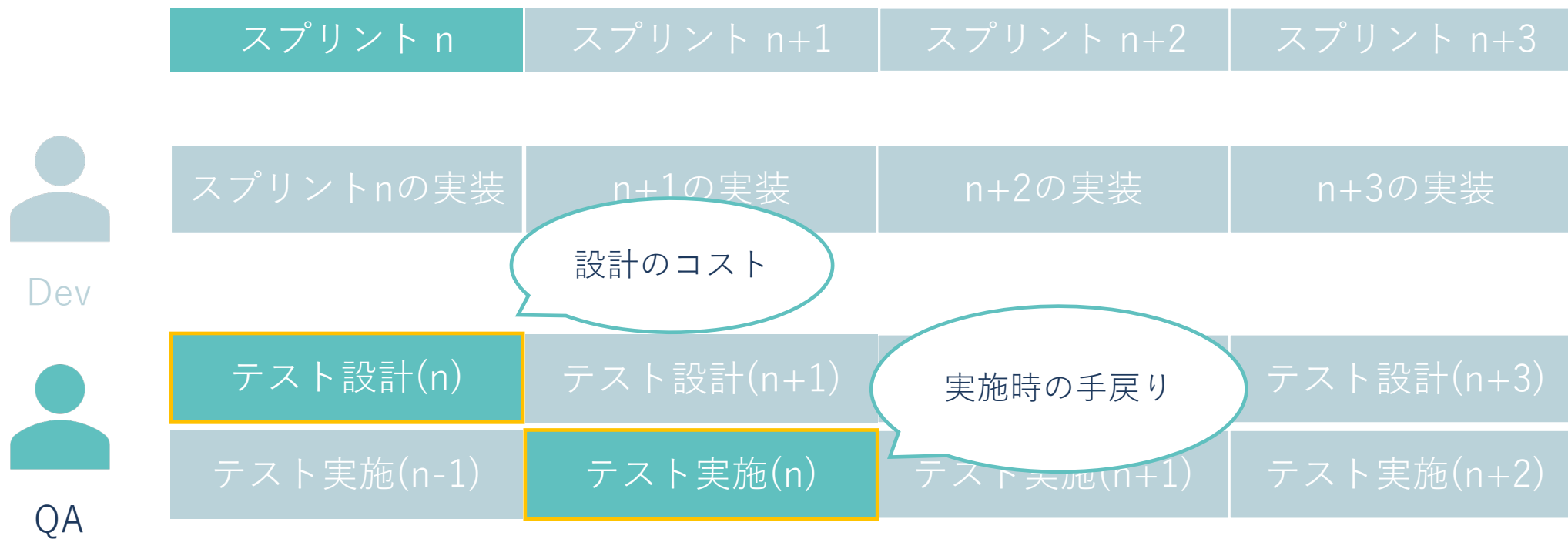
スプリントレビューの時点で、成果物に対して品質のフィードバックが無い

問題



検出した不具合を改修する場合も、実装してから2~3スプリント遅れで対応

問題



テスト設計のコストを下げられないか？テスト実施時の手戻りを防げないか？
テスト設計・実施の品質は落とさずにコストを下げられないか取り組みを始めた

目次

背景

対策と効果

課題

対策と効果	◀ テスト設計コストの軽減	マインドマップの活用
		モブの活用
課題	手戻りの軽減	スプリント開始時にテスト設計を共有
		テスト設計を前倒しで行うフローに変

問題



テスト設計のコストを下げる取り組みについて紹介

目次

背景

対策と効果

課題



マインドマップの活用



スプレットシート

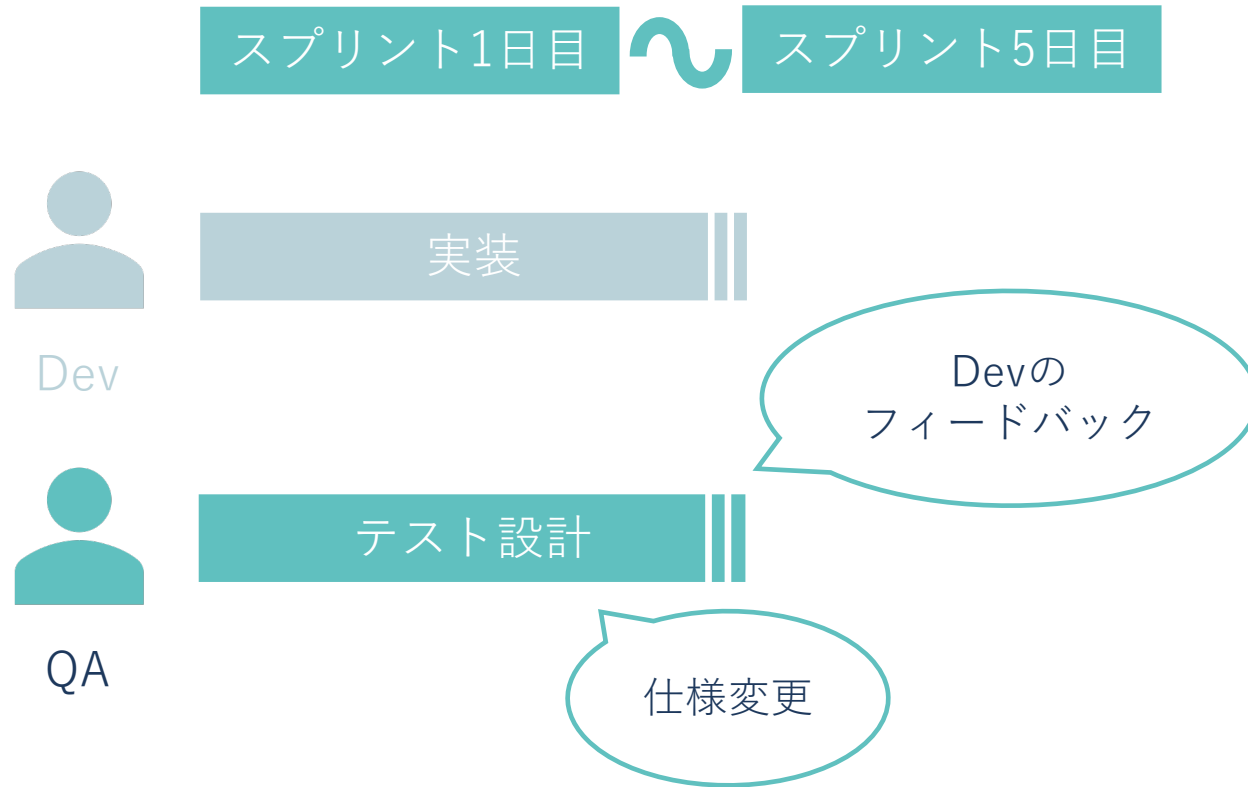
テスト設計の成果物は、もともとスプレットシート(Excel)を使用

マインドマップの活用



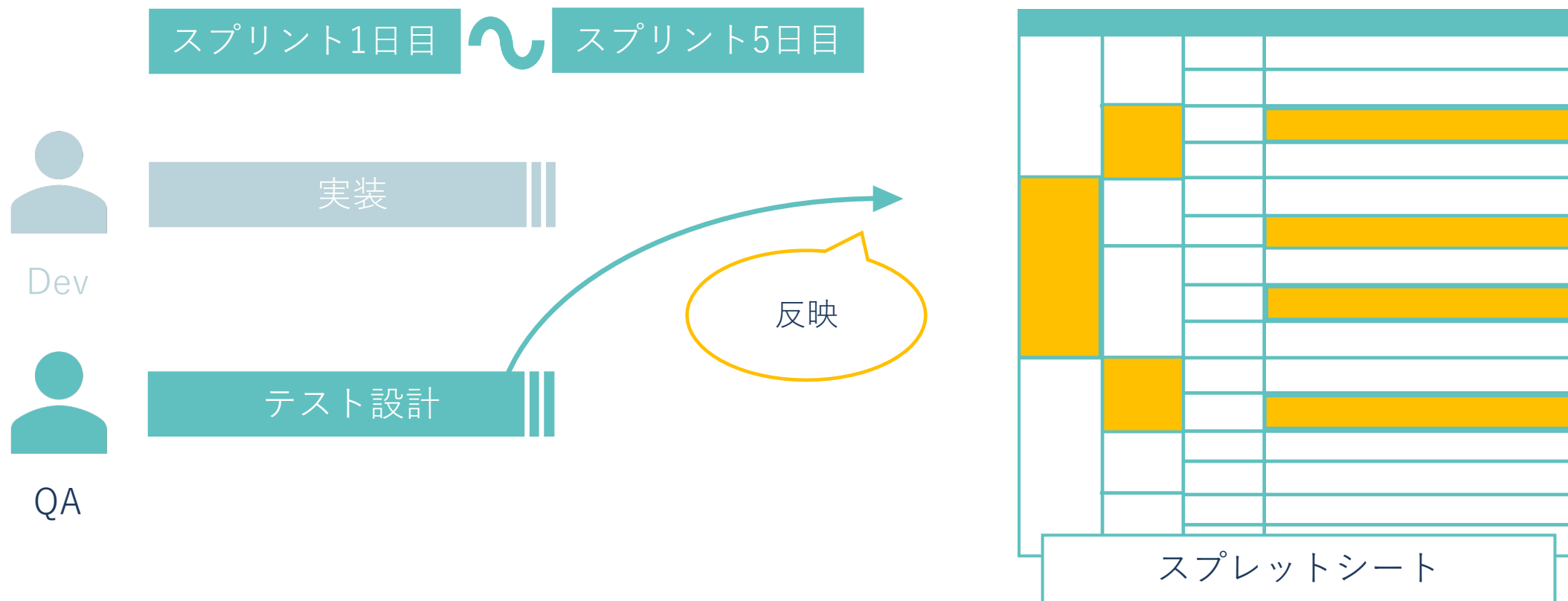
仕様はスプリント1日目時点で決まってはいるが、
実装都合で変更される場合もあり仕様書が完成するのはスプリント最終日の5日目

マインドマップの活用



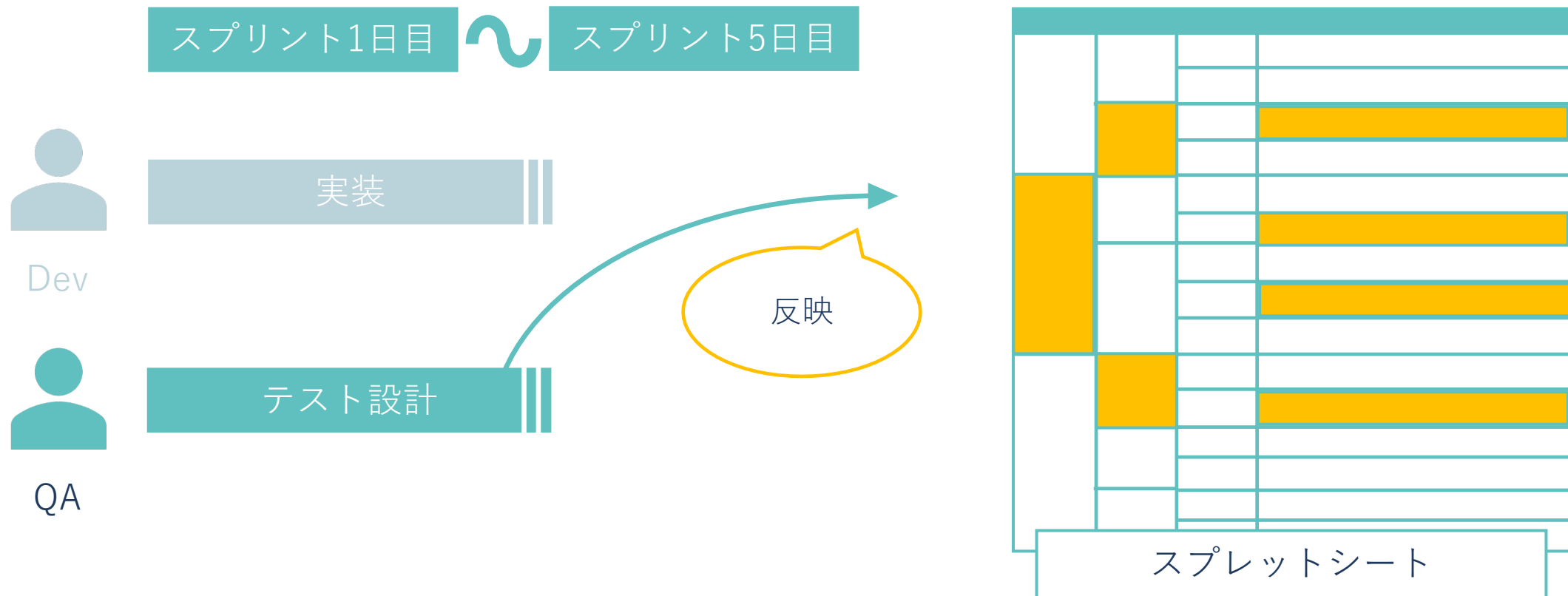
仕様変更への対応やDevに依頼したテスト設計レビューのフィードバック対応など行う

マインドマップの活用



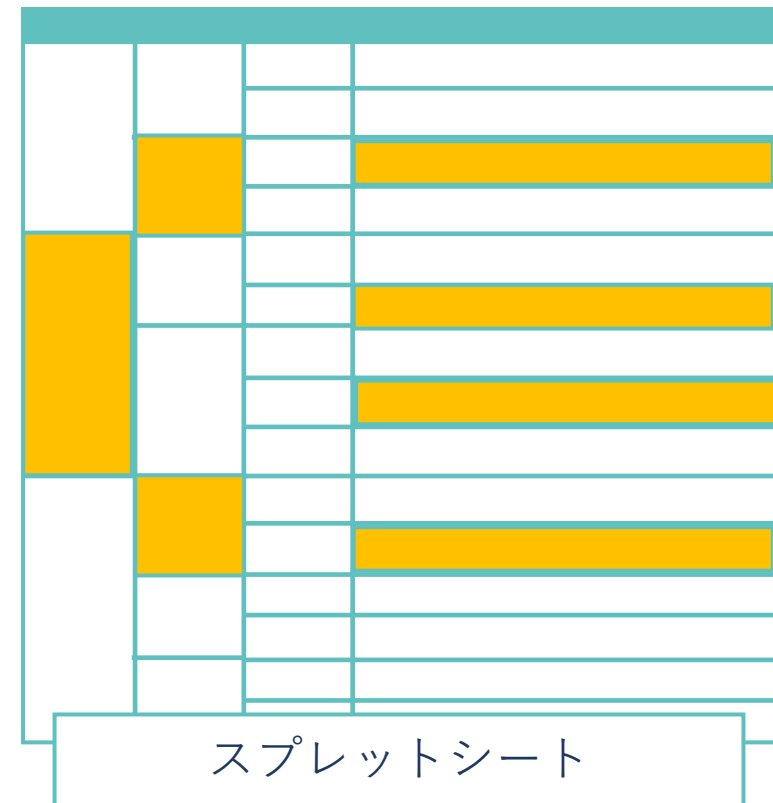
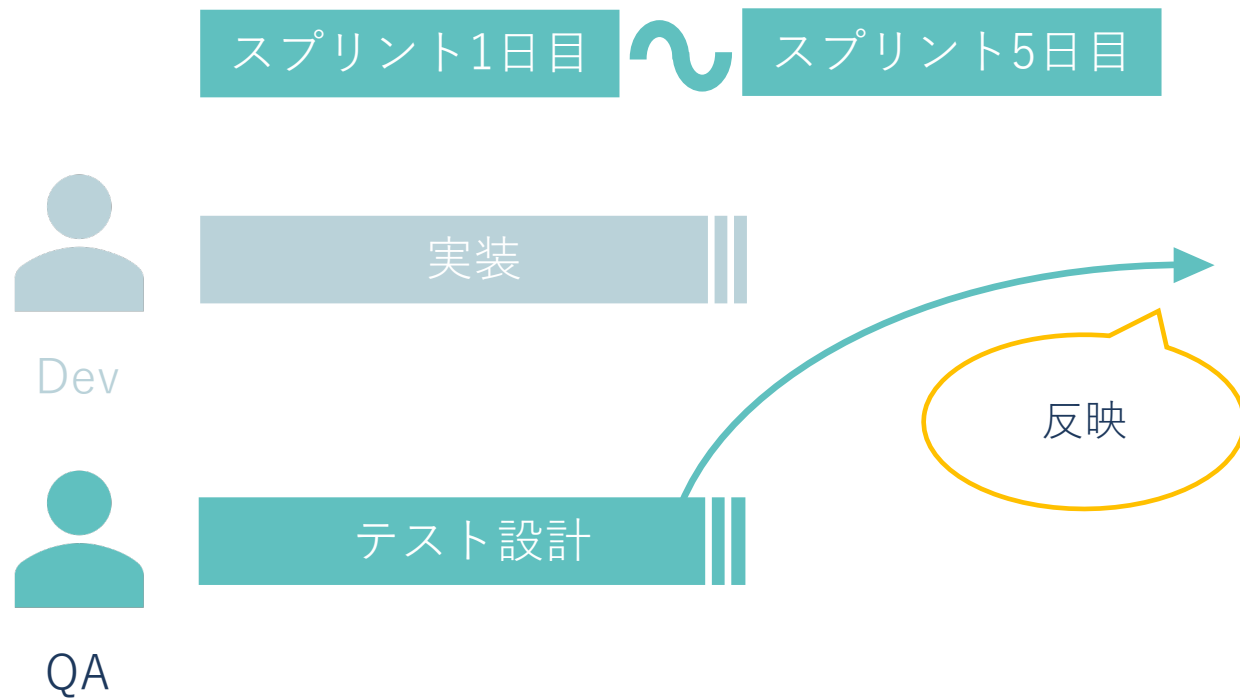
都度、修正を反映するが、テストの構成や詳細な手順に変更がある場合に修正コストがかかる

マインドマップの活用



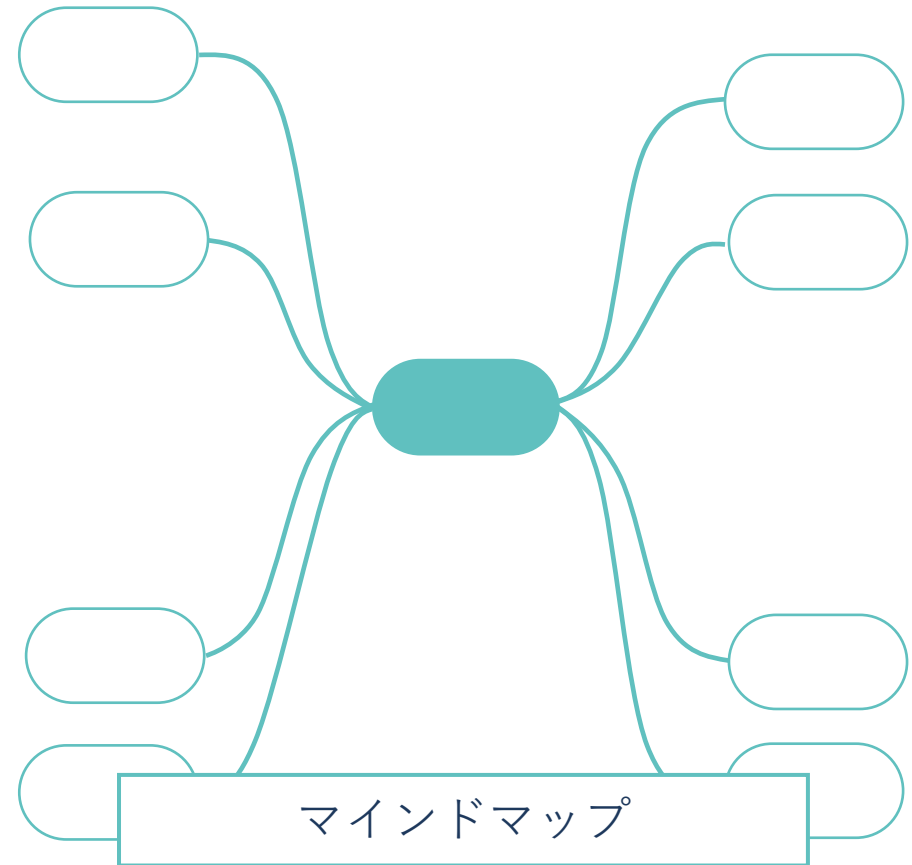
QAエンジニアが4名と比較的小規模で認識合わせがしやすいことと
製品の特徴としてテスト実施時にあまり複雑な手順を必要としない

マインドマップの活用



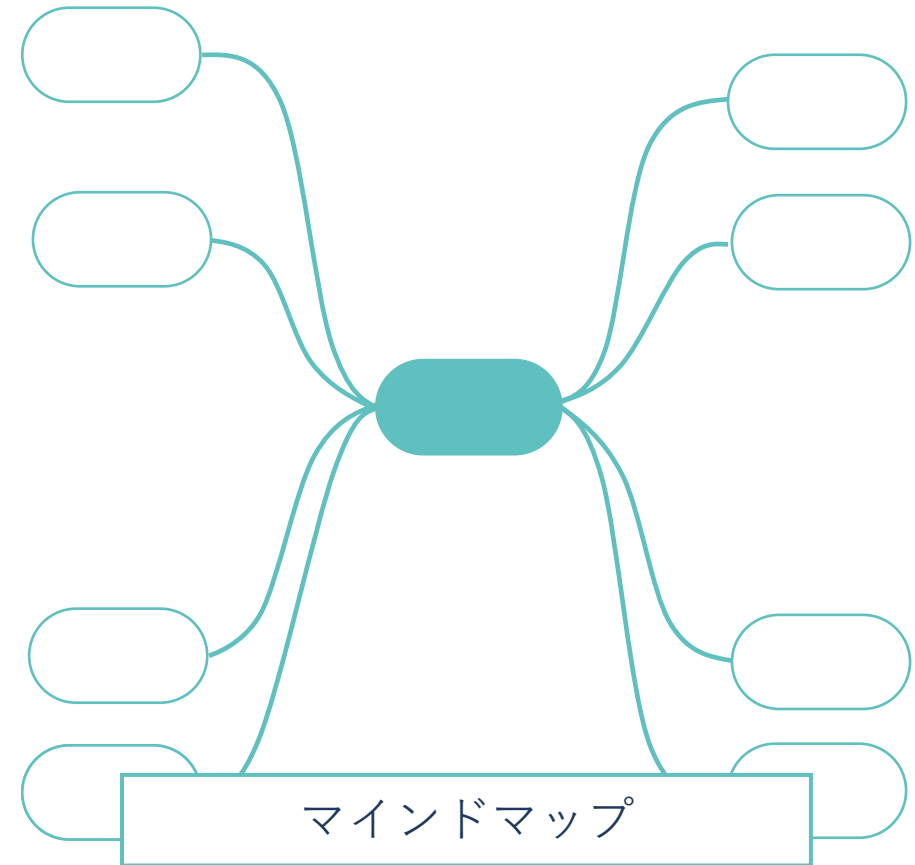
詳細な手順を書くのは作成・修正コストが大きい割に成果にあまり寄与していないのでは？

マインドマップの活用



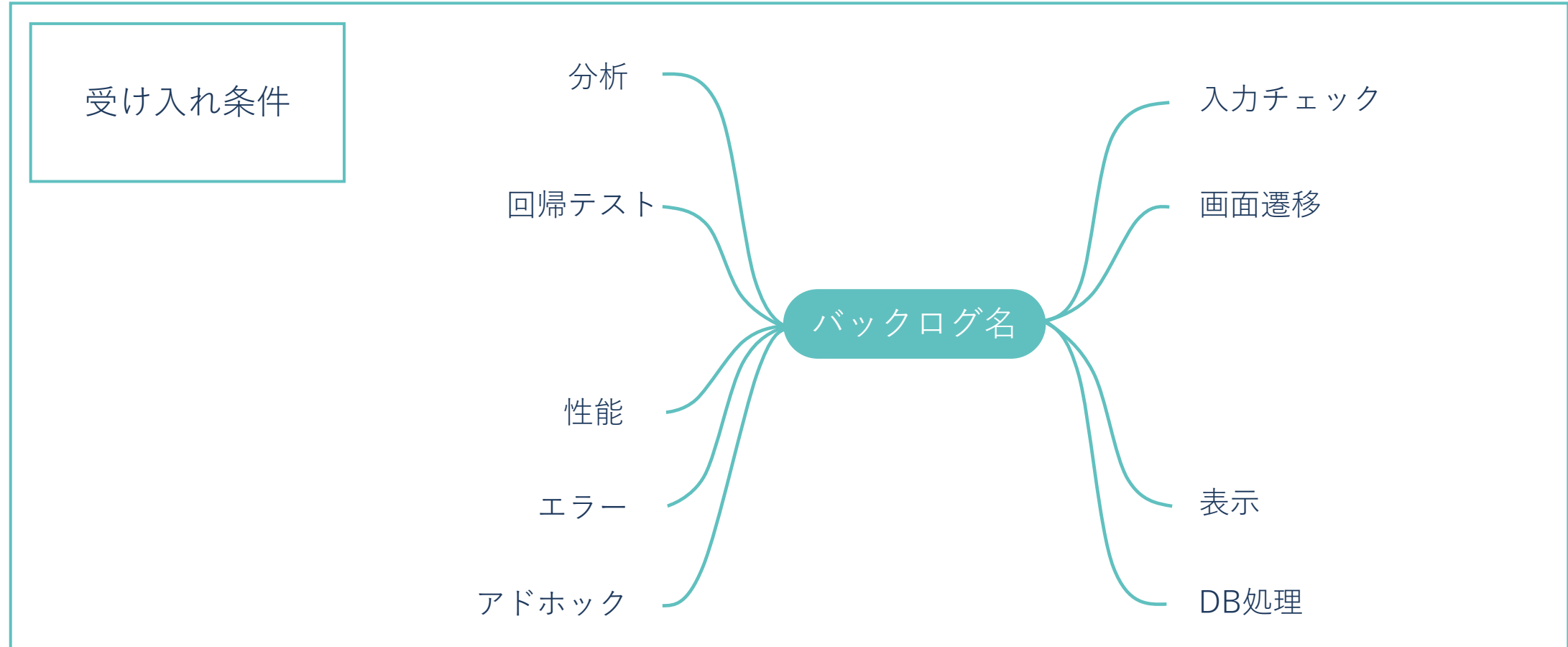
テスト設計のアウトプットを、スプレットシートからマインドマップに変更(miroを使用)

マインドマップの活用



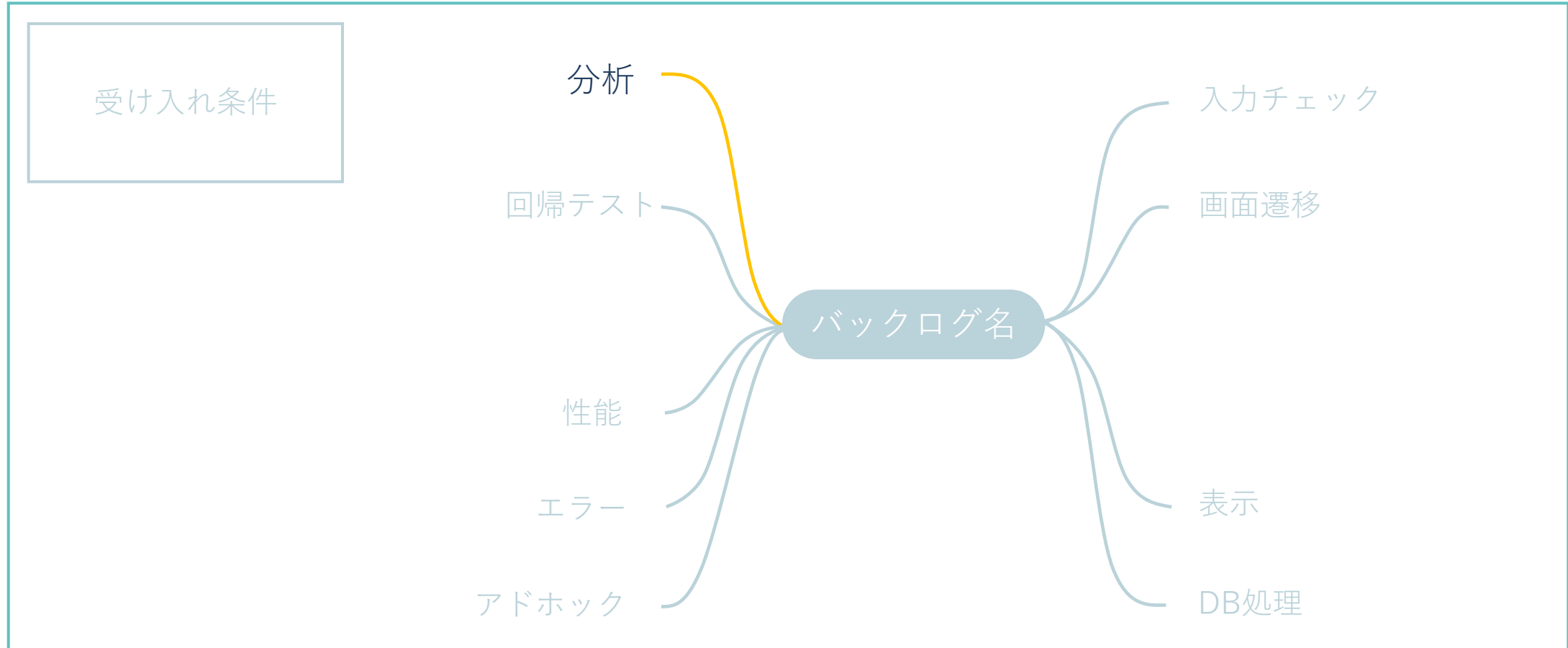
詳細なテスト手順の記載や体裁を整えることをやめ、テスト観点の洗い出しに注力

マインドマップの活用



あらかじめQAエンジニア内で作成したフォーマットに従って記載

マインドマップの活用



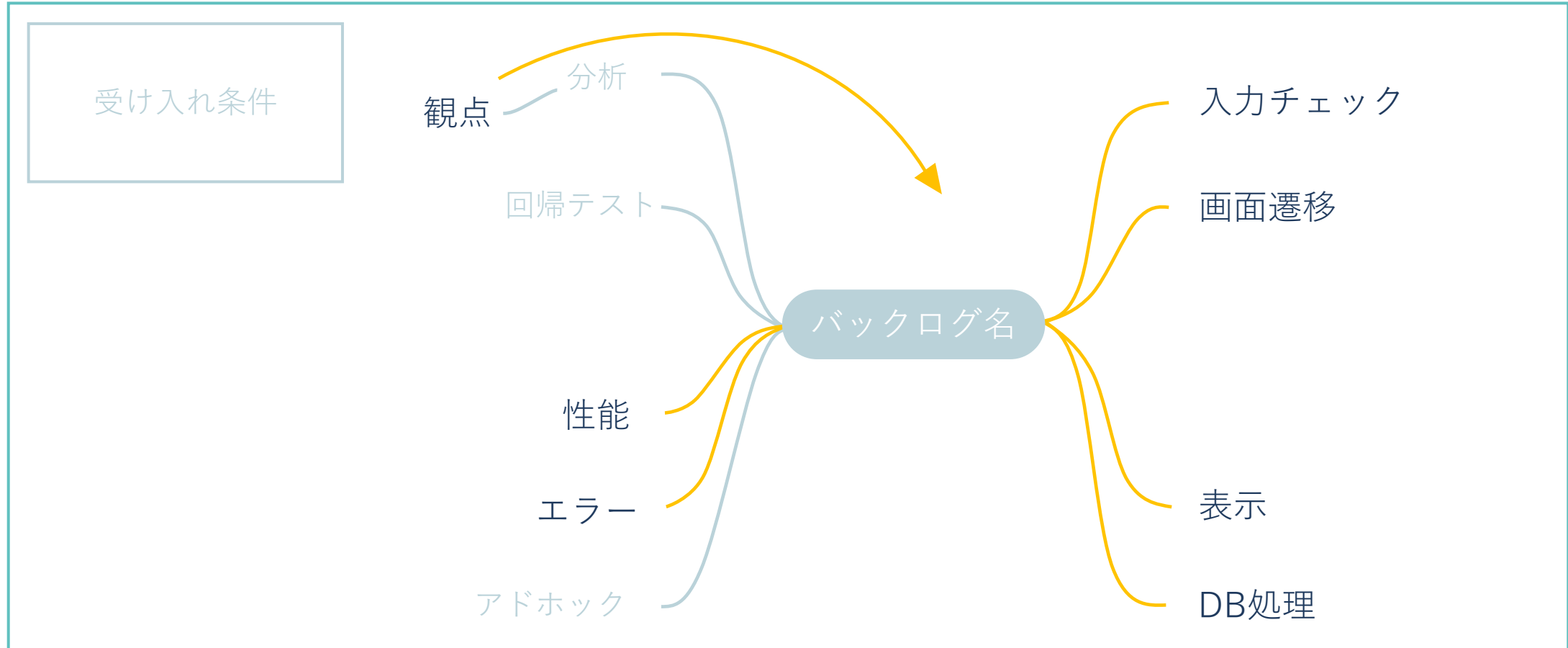
まず [分析] の枝に、テスト設計前の分析を記載

マインドマップの活用



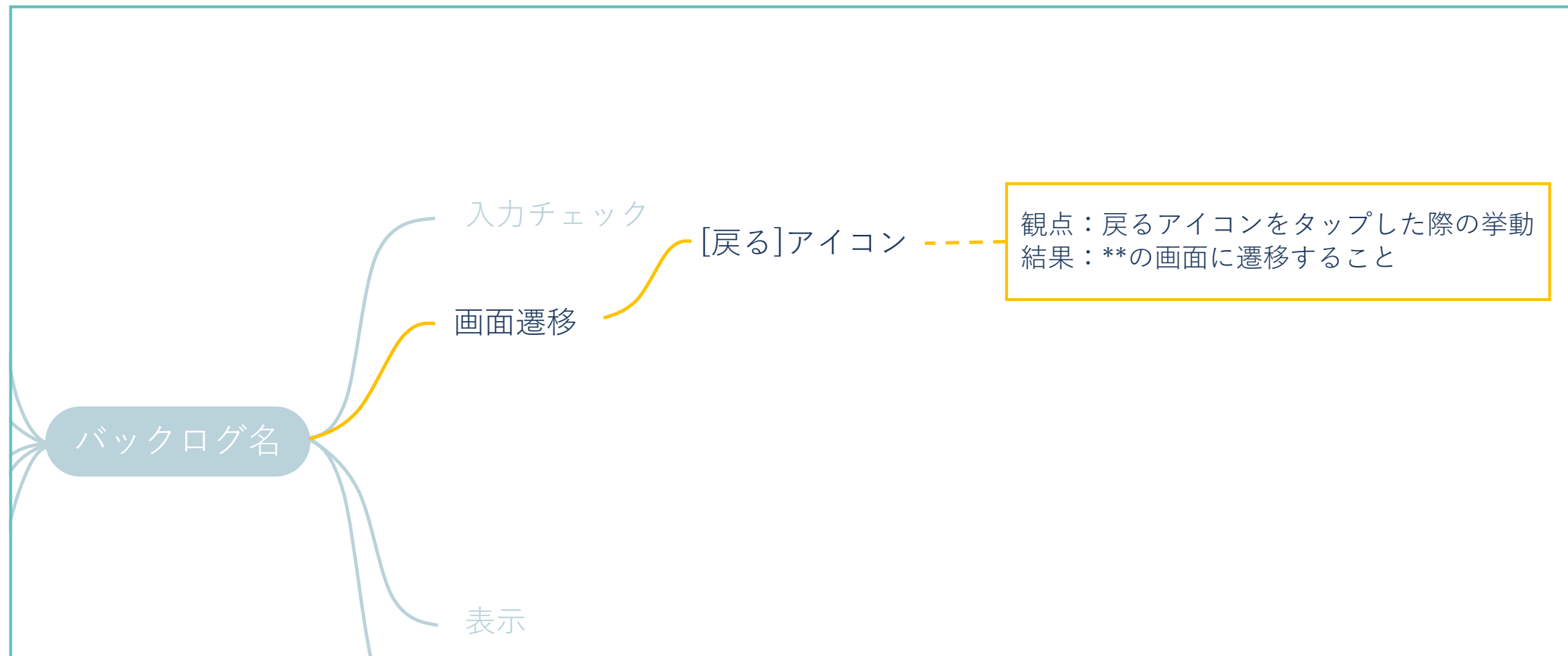
[テスト方針]に該当のバックログで何を確認するのかなど方針を、
[テスト範囲]にスコープを、[観点]にテストで確認すべき項目を記載

マインドマップの活用



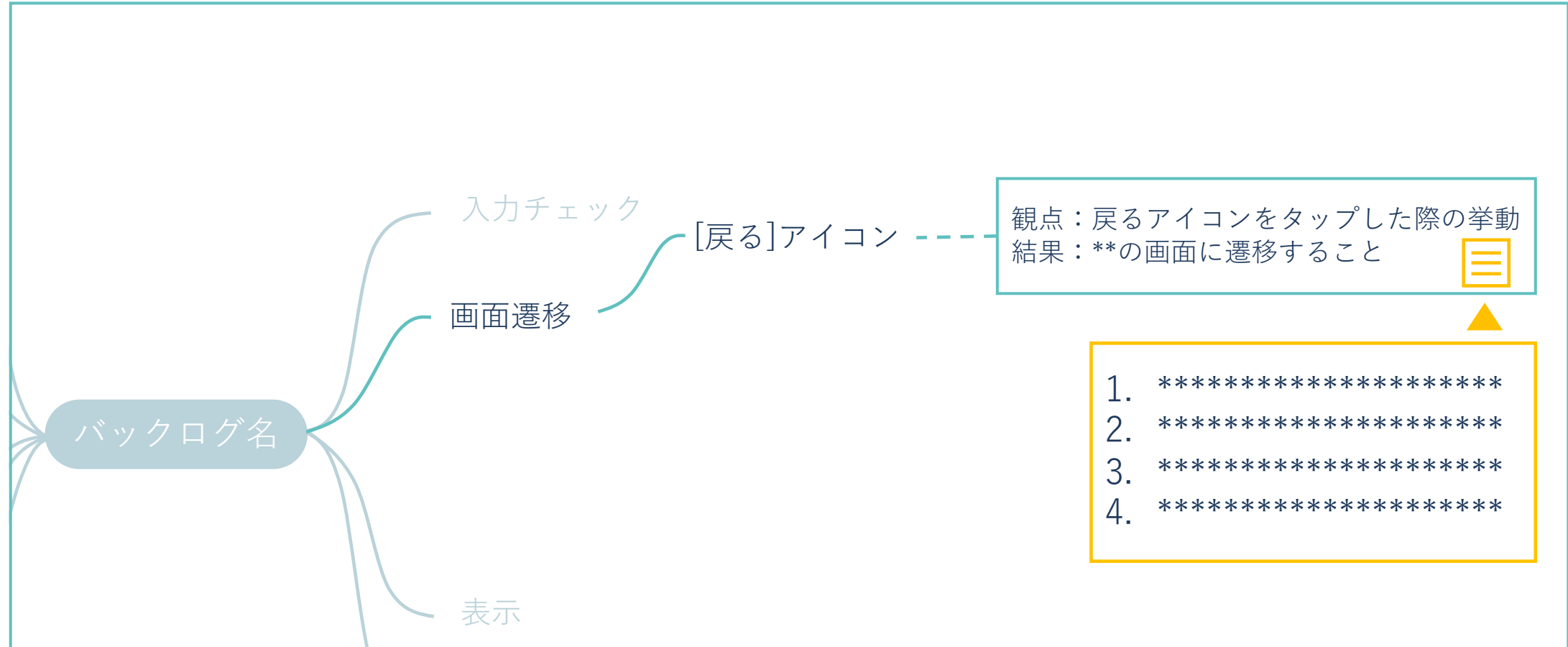
分析で洗い出した観点を元に、あらかじめ用意された分類に沿ってテストケースを記載

マインドマップの活用



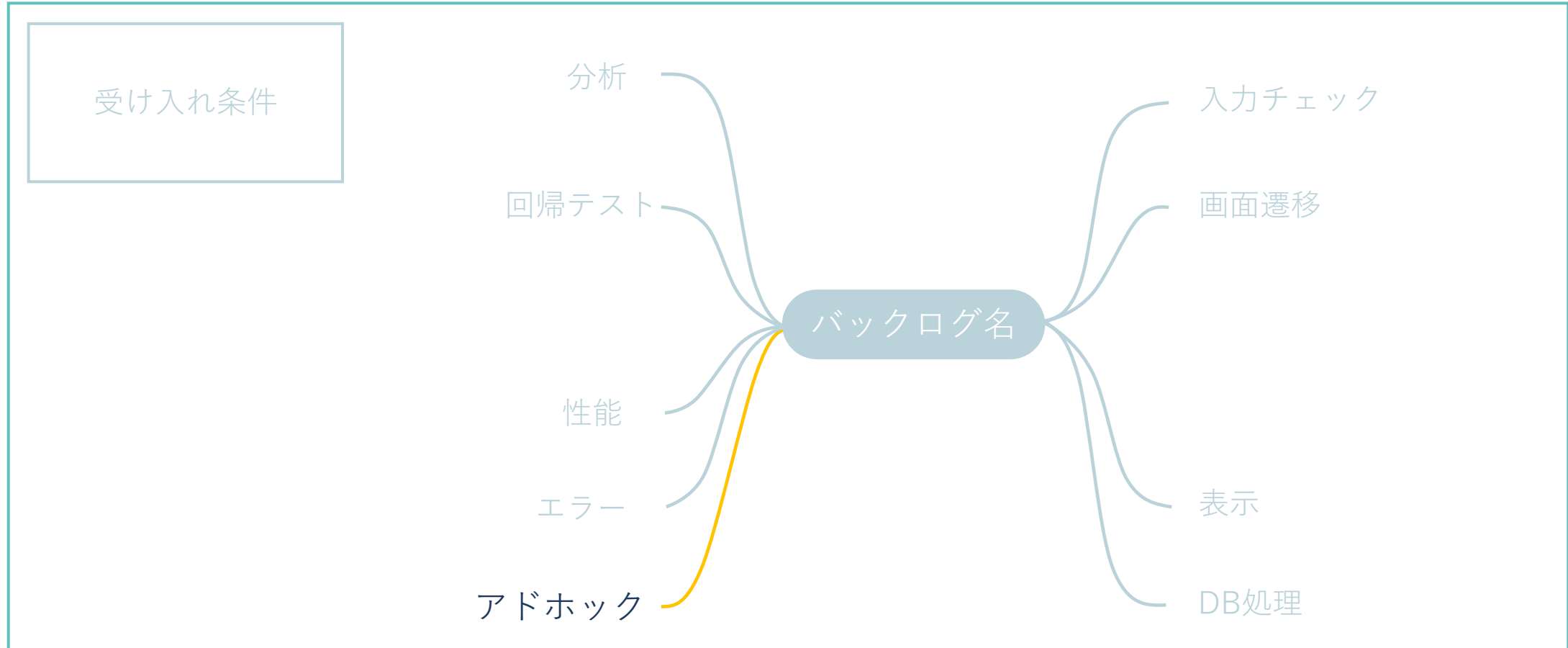
末端の枝に観点と期待結果を記載して1つのテストケースとする(miroのCardを使用)

マインドマップの活用



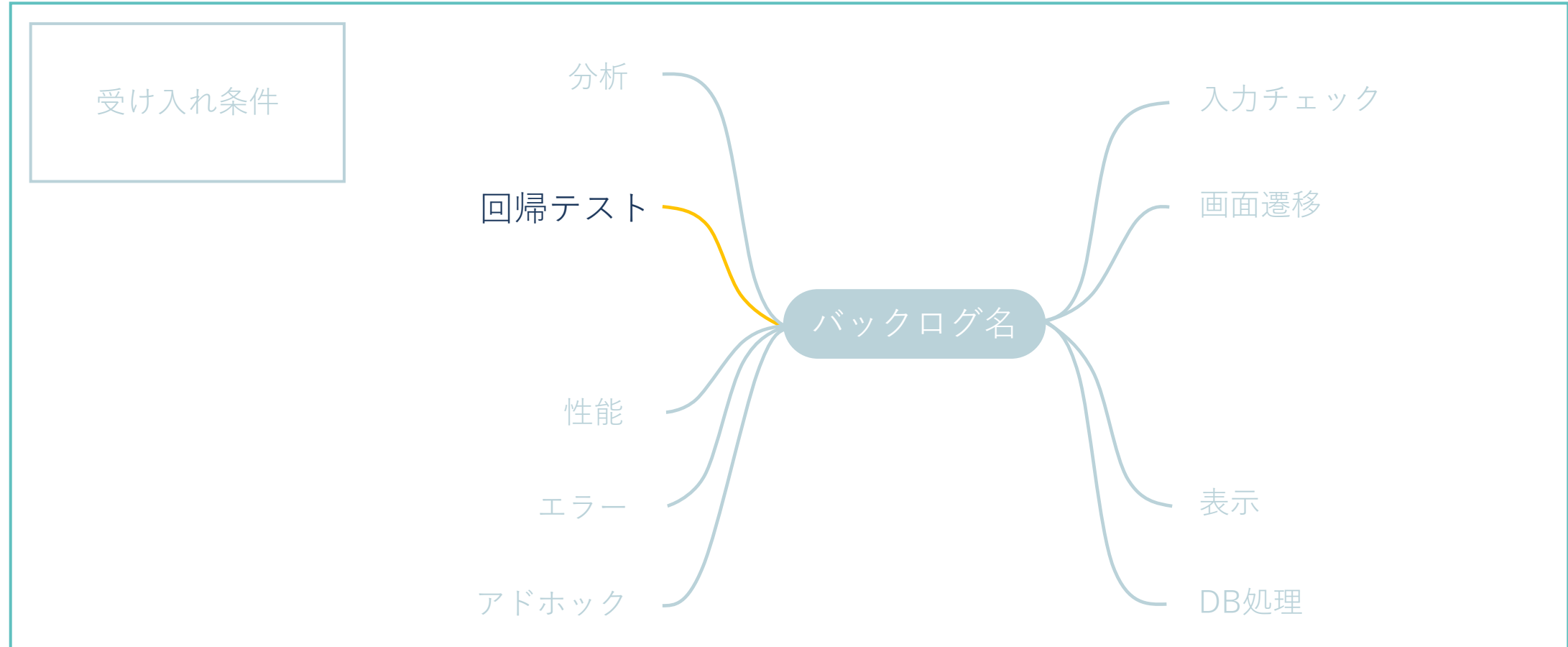
複雑な手順を持つテストケースのみ手順を記載(miroのカードパネル機能を使用)

マインドマップの活用



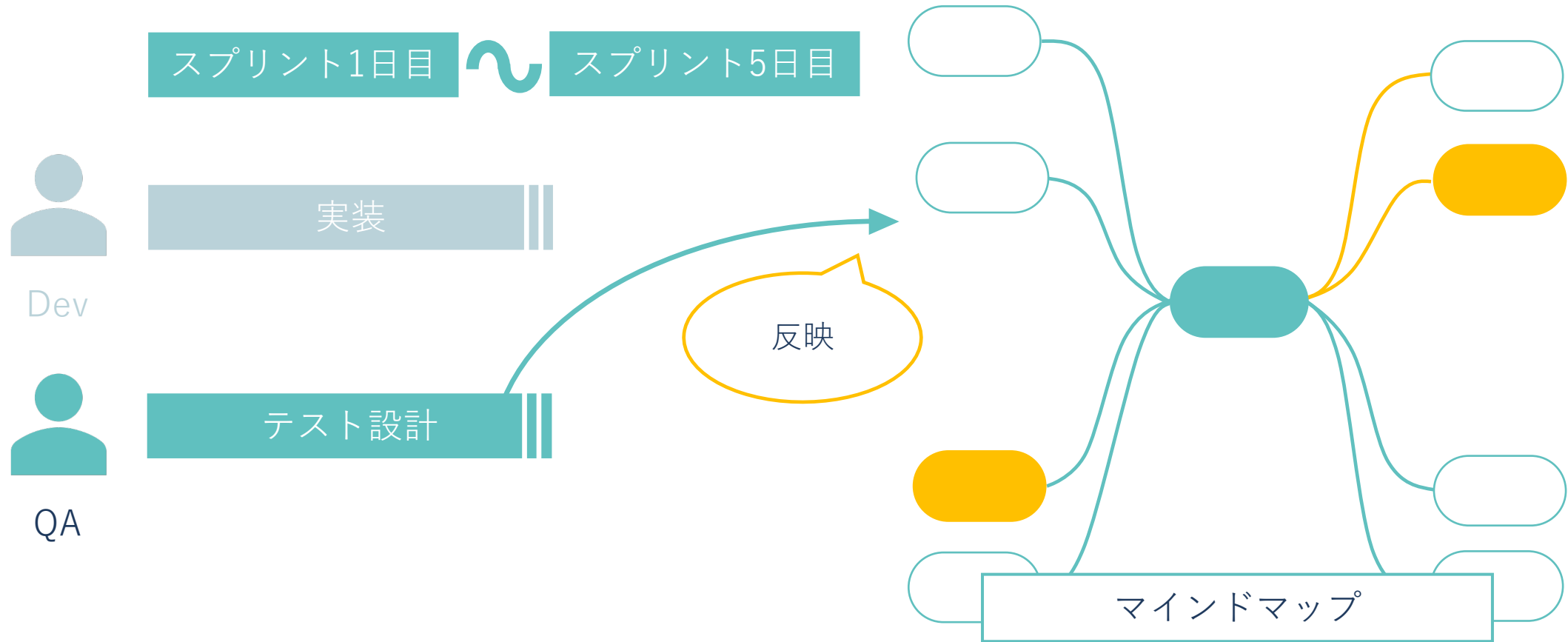
[テスト範囲]のスコープ外だが不具合があるかもしれない、という観点はアドホックの枝に記載

マインドマップの活用



ユーザーへの影響が大きい項目はリリース前に再度実施するため回帰テストの枝に記載

マインドマップの活用



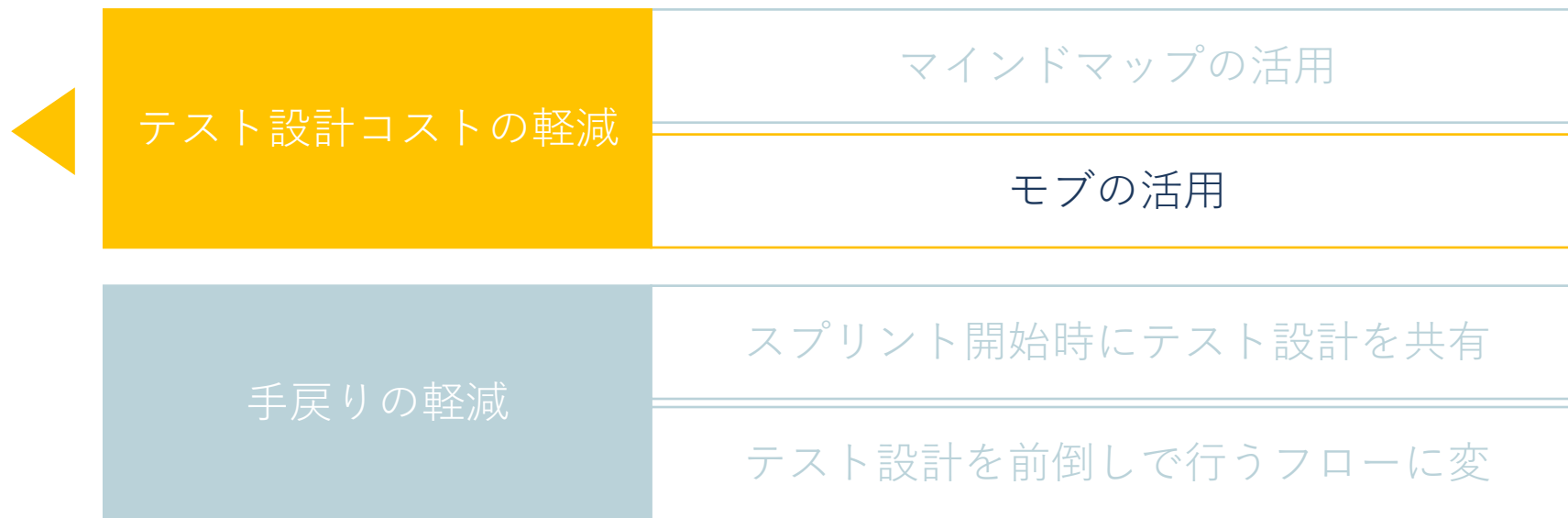
コストをかけずにテスト構成の変更や観点の修正に対応

目次

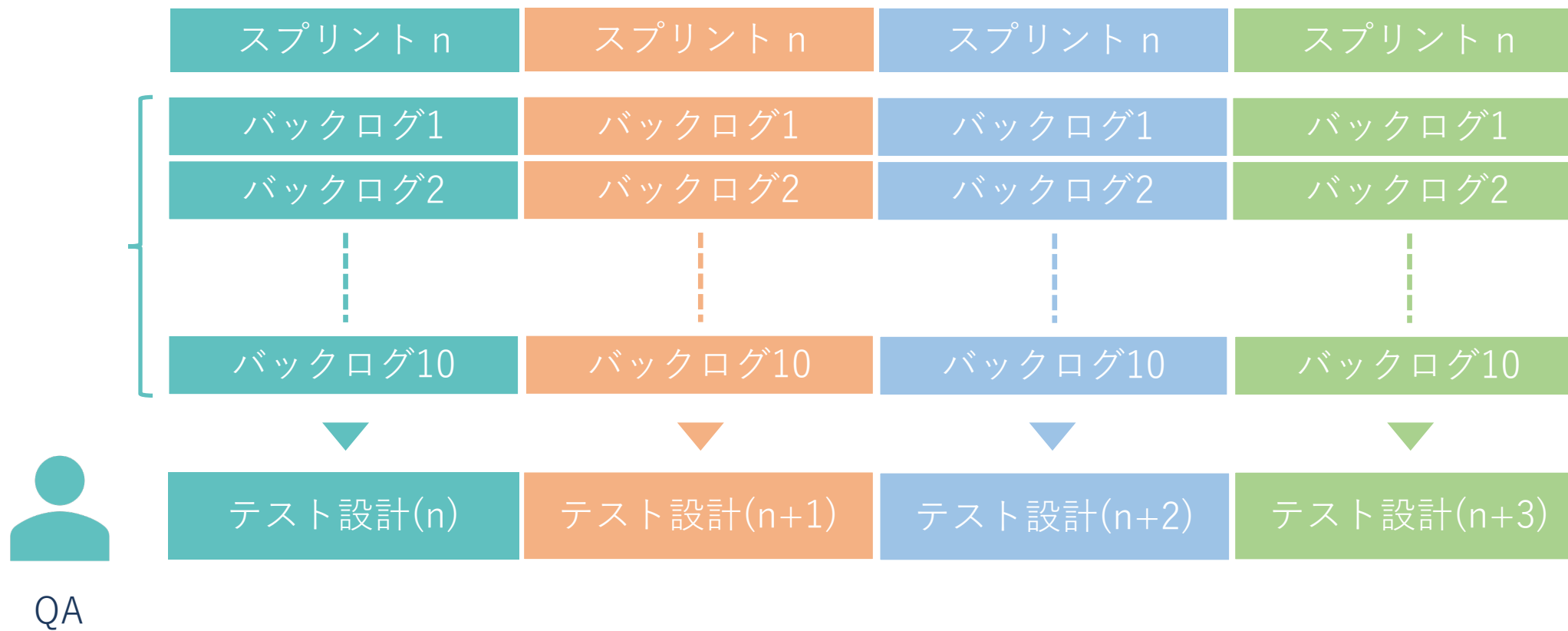
背景

対策と効果

課題

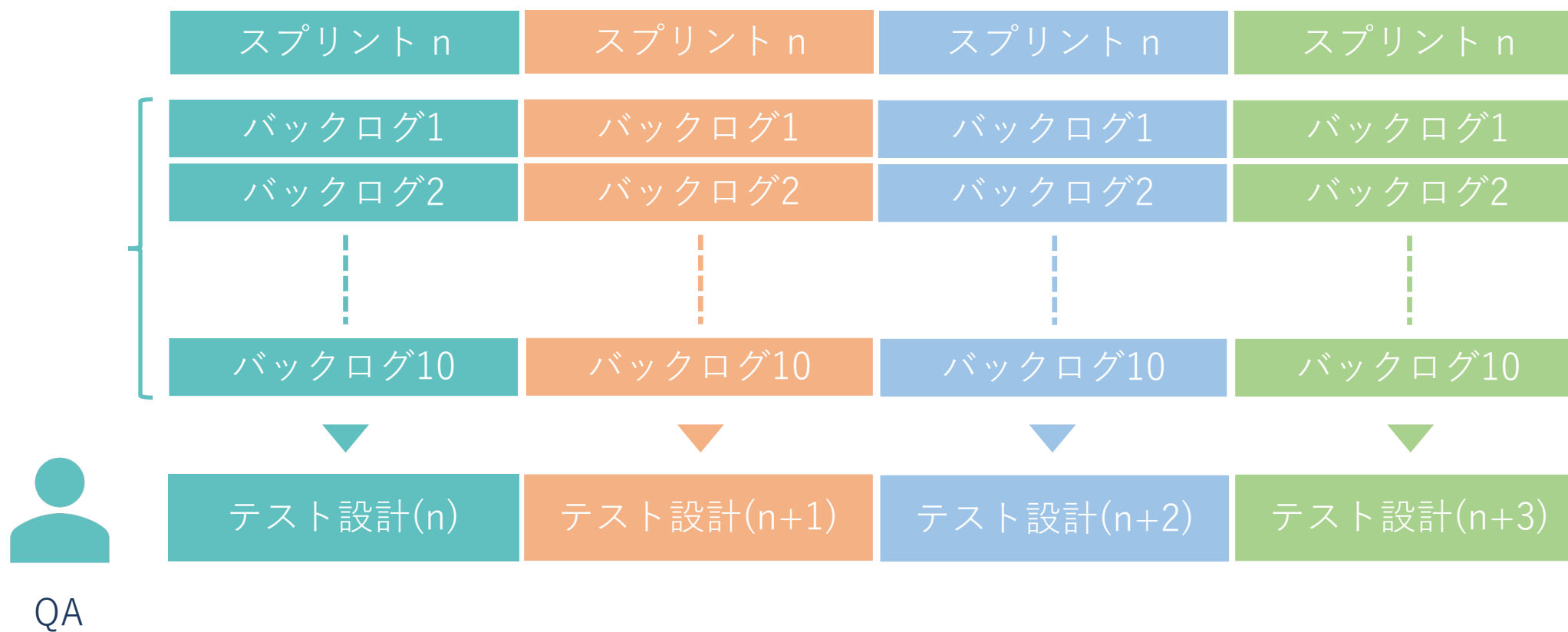


モブの活用



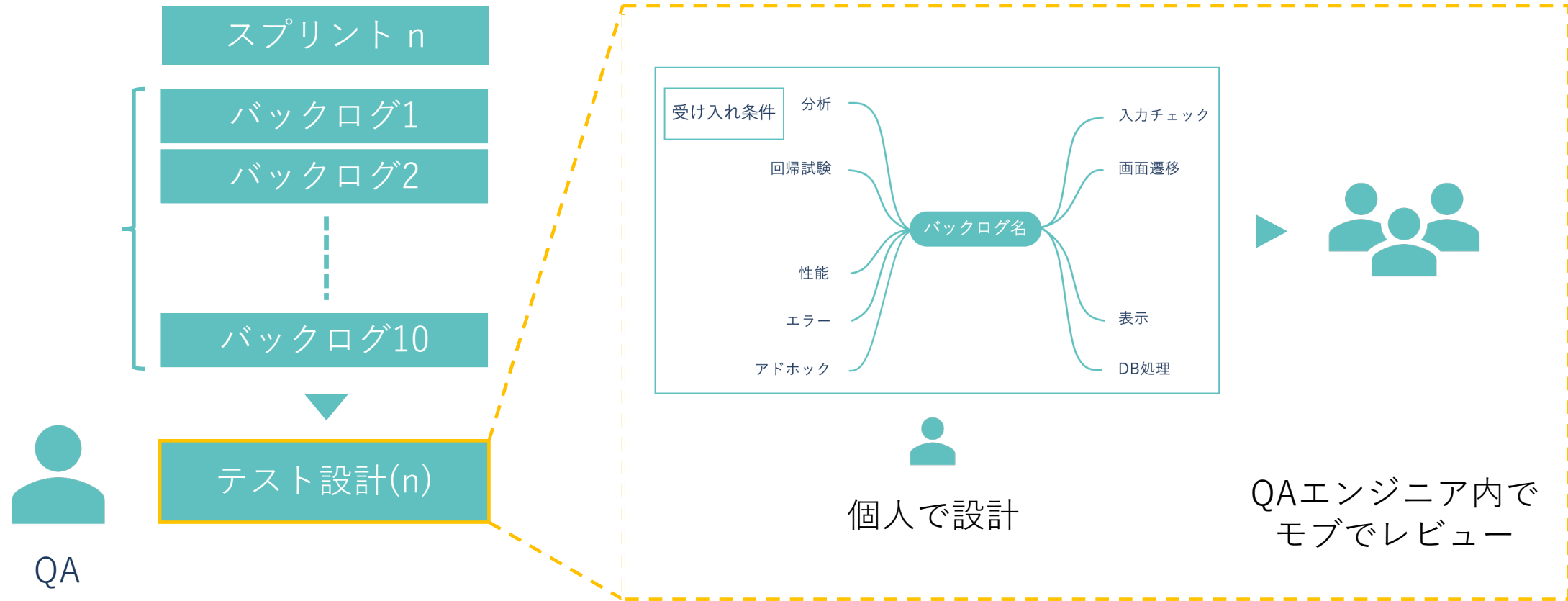
1つのスプリントで対応するバックログは10個前後ある

モブの活用



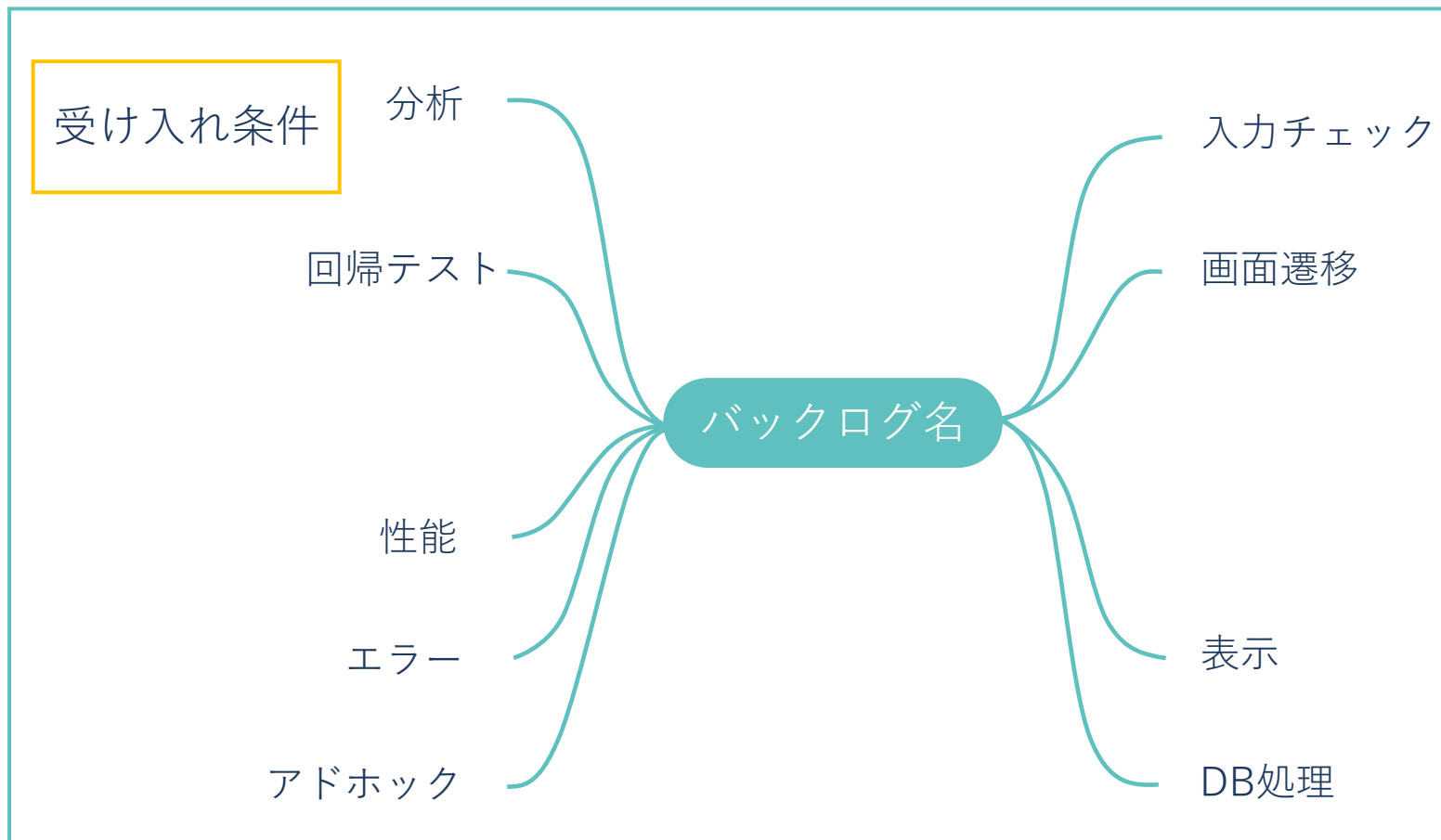
テスト設計に関わっていないメンバーが容易にキャッチアップできるようにしたい
かつ、QAエンジニア内で見落としの少ないテスト設計レビューを行いたい

モブの活用



テスト設計のレビューは決まった時間にモブで行う方式を採用

モブの活用

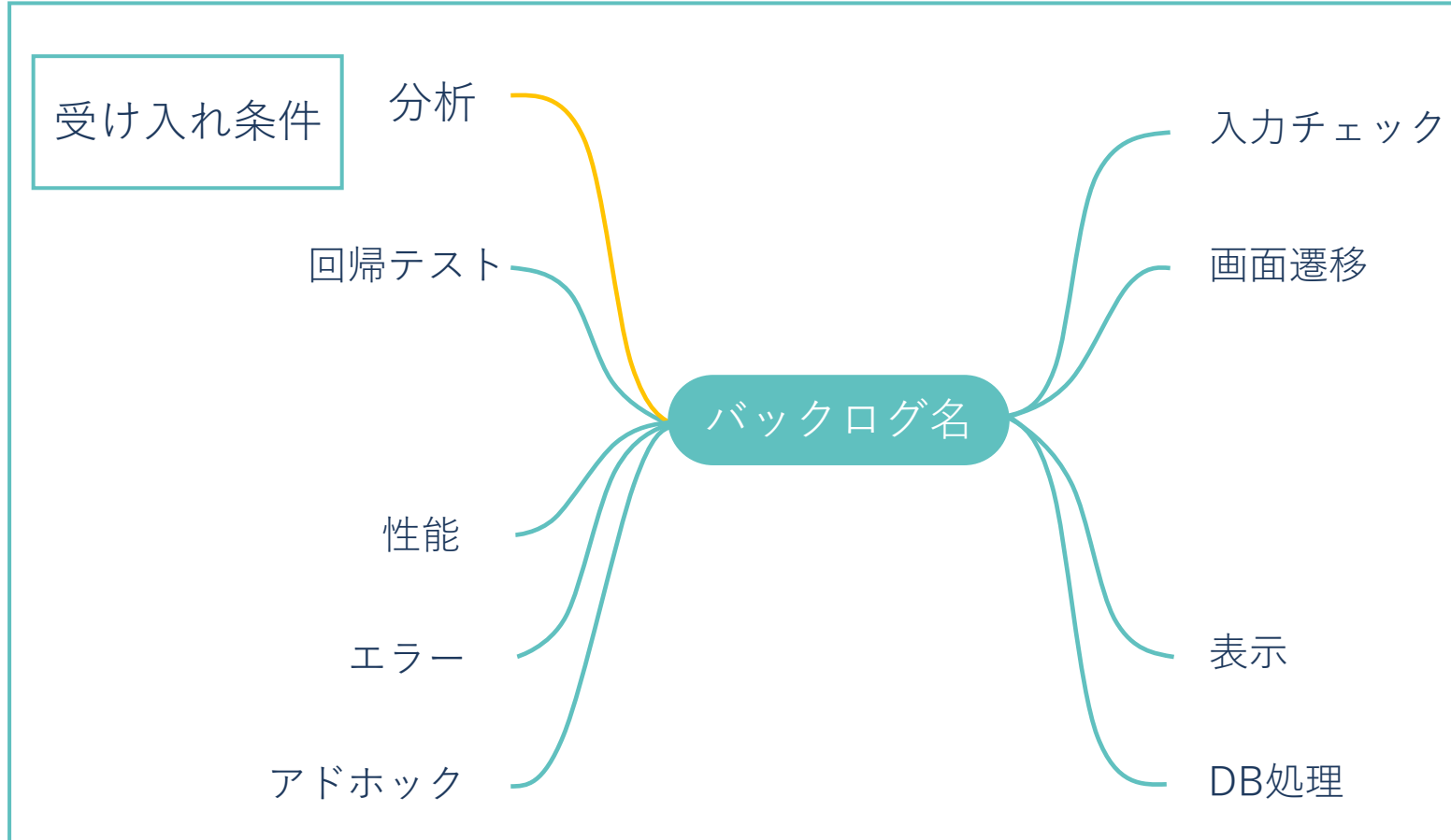


はじめに、当番制のファシリテーターがバックログの受け入れ条件やスコープなど読み上げて確認

モブの活用


ファシリテーター


レビューアー

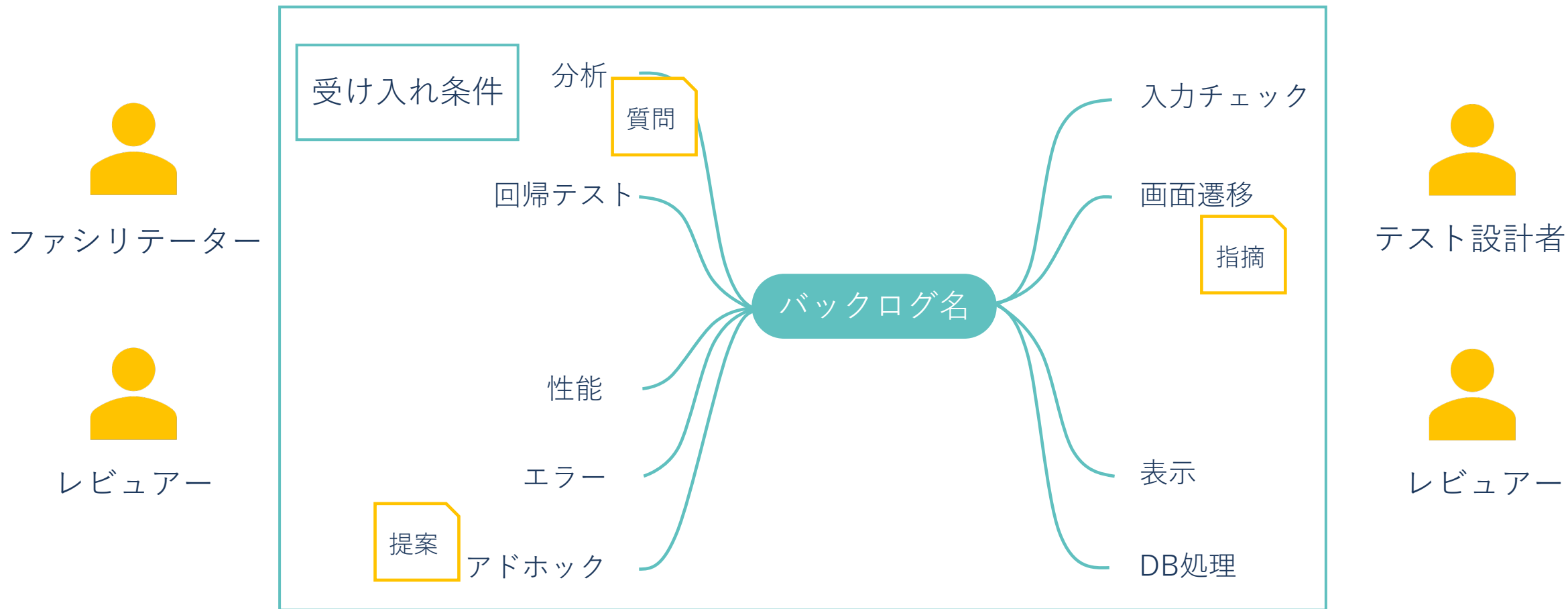



テスト設計者


レビューアー

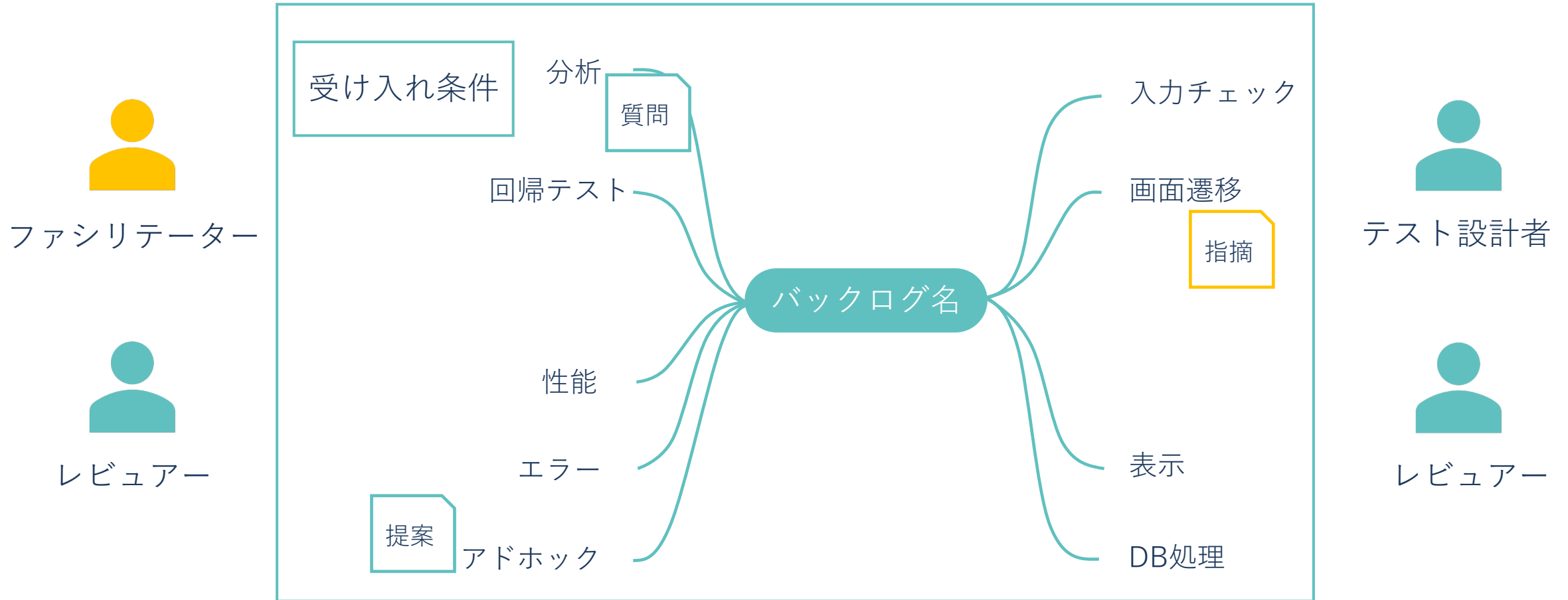
次にテスト設計者が [分析] の内容を説明

モブの活用



時間をとって、参加者全員で各項目を確認しつつ質問や指摘、提案を付箋で残す

モブの活用

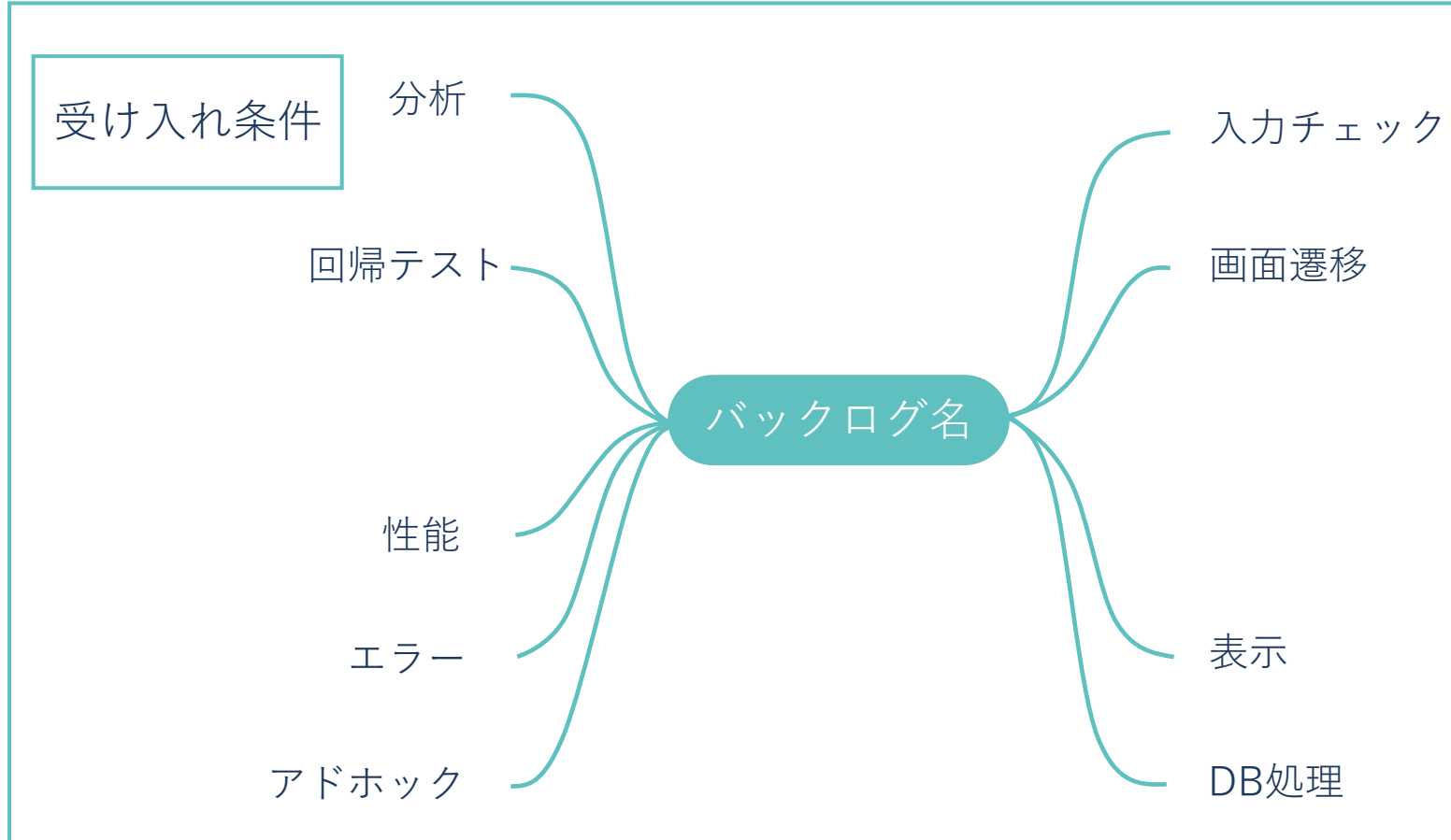


全員が確認し終わったら付箋を1つずつ確認し、議論する
必要があれば、その場でテストケースを修正

モブの活用


ファシリテーター


レビュアー

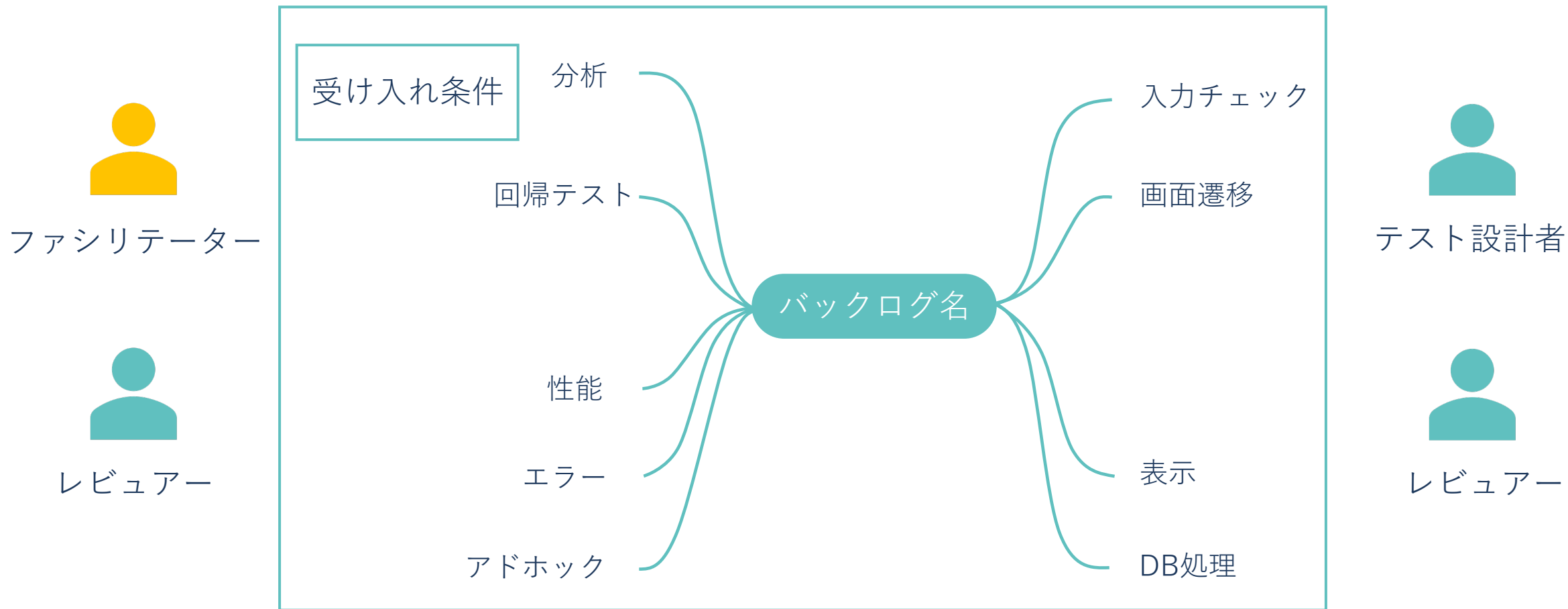



テスト設計者


レビュアー

全ての付箋を確認したらレビュー完了とし、テスト設計は完成

モブの活用



テスト設計自体の質を保つ、お互いに学びがあるというメリットがありつつ
レビューのコストや負担を軽減

モブの活用



個人で設計





個人でレビュー

メリット : 個人で空き時間に進めたり、複数のバックログで並行で作業可能
1人でじっくり考え、設計することによる成長

デメリット : レビュー時の大幅な手戻りが発生するリスク
設計者とレビュアーのみ内容を把握してる状態になりがち





初めは個人で設計し、他のQAエンジニア(1人)にレビューを依頼する体制

モブの活用

 個人で設計	▶		<p>メリット : 個人で空き時間に進めたり、複数のバックログで並行で作業可能 1人でじっくり考え、設計することによる成長</p> <p>デメリット : レビュー時の大幅な手戻りが発生するリスク 設計者とレビュアーのみ内容を把握してる状態になりがち</p>
 モブで設計	▶	レビュー省略	<p>メリット : 互いに学びがあり、かつ見落としの少ないテスト設計ができる</p> <p>デメリット : 拘束時間が長くスケジュール調整が必要 単純作業などでドライバーの作業をただ眺めている時間がある</p>




モブで設計することを試し、良い面も多かったがデメリットがあり再度検討

モブの活用

 個人で設計	▶		<p>メリット : 個人で空き時間に進めたり、複数のバックログで並行で作業可能 1人でじっくり考え、設計することによる成長</p> <p>デメリット : レビュー時の大幅な手戻りが発生するリスク 設計者とレビュアーのみ内容を把握してる状態になりがち</p>
 モブで設計	▶	レビュー省略	<p>メリット : 互いに学びがあり、かつ見落としの少ないテスト設計ができる</p> <p>デメリット : 拘束時間が長くスケジュール調整が必要 単純作業などでドライバーの作業をただ眺めている時間がある</p>
 個人で設計	▶	 モブでレビュー	<p>メリット : 個人で空き時間に進めたり、複数のバックログで並行で作業可能 1人でじっくり考え、設計することによる成長 お互いに学びがあり、見落としが少ないレビューができる</p> <p>デメリット : レビュー時の大幅な手戻りが発生するリスク</p>

3パターン試して、個人で設計しモブでレビューする体制が最もメリットが大きいと判断

モブの活用

 個人で設計	▶		<p>メリット : 個人で空き時間に進めたり、複数のバックログで並行で作業可能 1人でじっくり考え、設計することによる成長</p> <p>デメリット : レビュー時の大幅な手戻りが発生するリスク 設計者とレビュアーのみ内容を把握してる状態になりがち</p>
 個人で設計	▶	レビュー省略	<p>メリット : 互いに学びがあり、かつ見落としの少ないテスト設計ができる</p> <p>デメリット : 拘束時間が長くスケジュール調整が必要 単純作業などでドライバーの作業をただ眺めている時間がある</p>
 個人で設計	▶		<p>メリット : 個人で空き時間に進めたり、複数のバックログで並行で作業可能 1人でじっくり考え、設計することによる成長 お互いに学びがあり、見落としが少ないレビューができる</p> <p>デメリット : レビュー時の大幅な手戻りが発生するリスク</p>

複雑なテスト設計は、はじめのテスト分析をモブで行うことでリスクを軽減

テスト設計コストの軽減



マインドマップとモブの活用で、テスト設計にかかるコストを削減

目次

背景

対策と効果

課題

対策と効果	テスト設計コストの軽減	マインドマップの活用
		モブの活用
課題	手戻りの軽減	スプリント開始時にテスト設計を共有
		テスト設計を前倒しで行うフローに変更

手戻りの軽減



テスト実施時の手戻りなどのコストを下げる取り組みについて紹介

目次

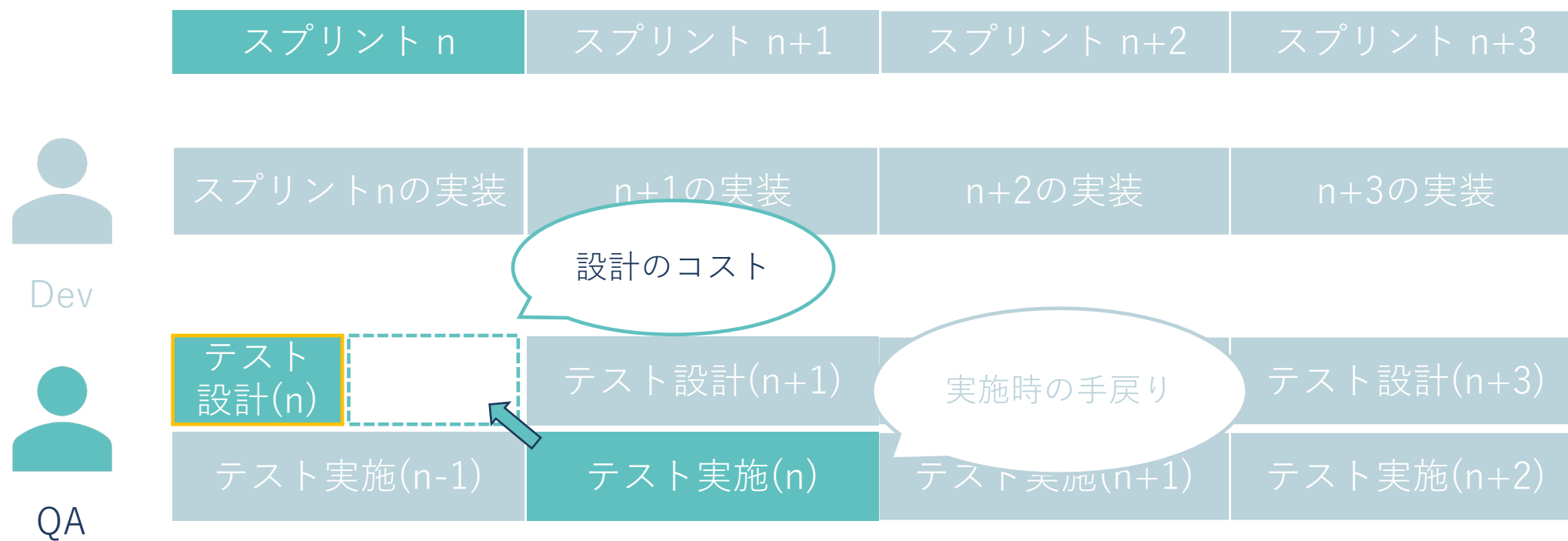
背景

対策と効果

課題

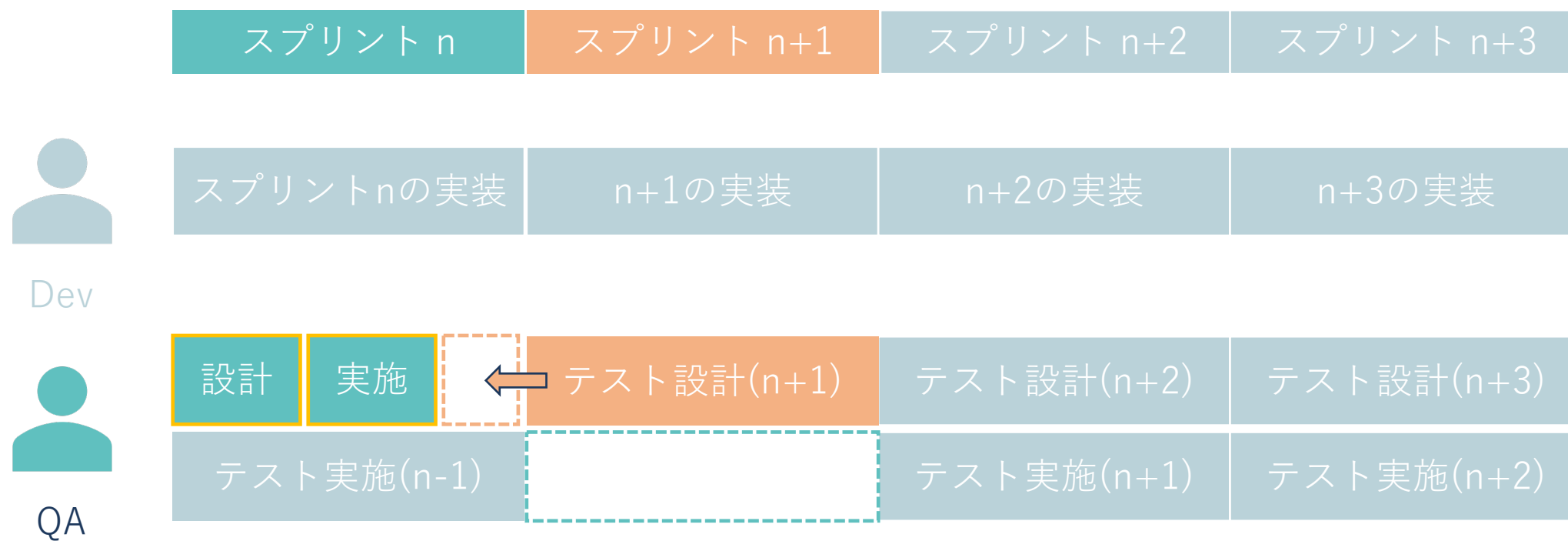
対策と効果	テスト設計コストの軽減	マインドマップの活用
		モブの活用
課題	手戻りの軽減	スプリント開始時にテスト設計を共有
		テスト設計を前倒しで行うフローに変更

スプリント開始時にテスト設計を共有



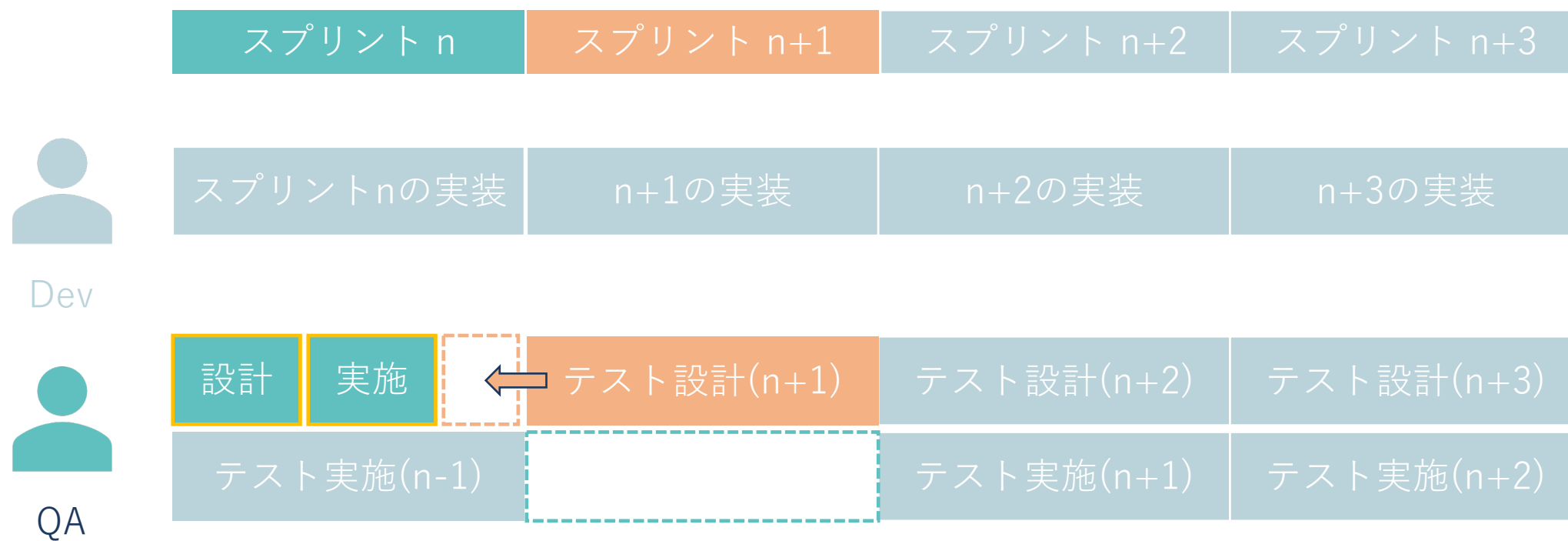
マインドマップとモブを活用してテスト設計にかかるコストを削減したことで
徐々にスプリント内でテスト実施を行えるように

スプリント開始時にテスト設計を共有



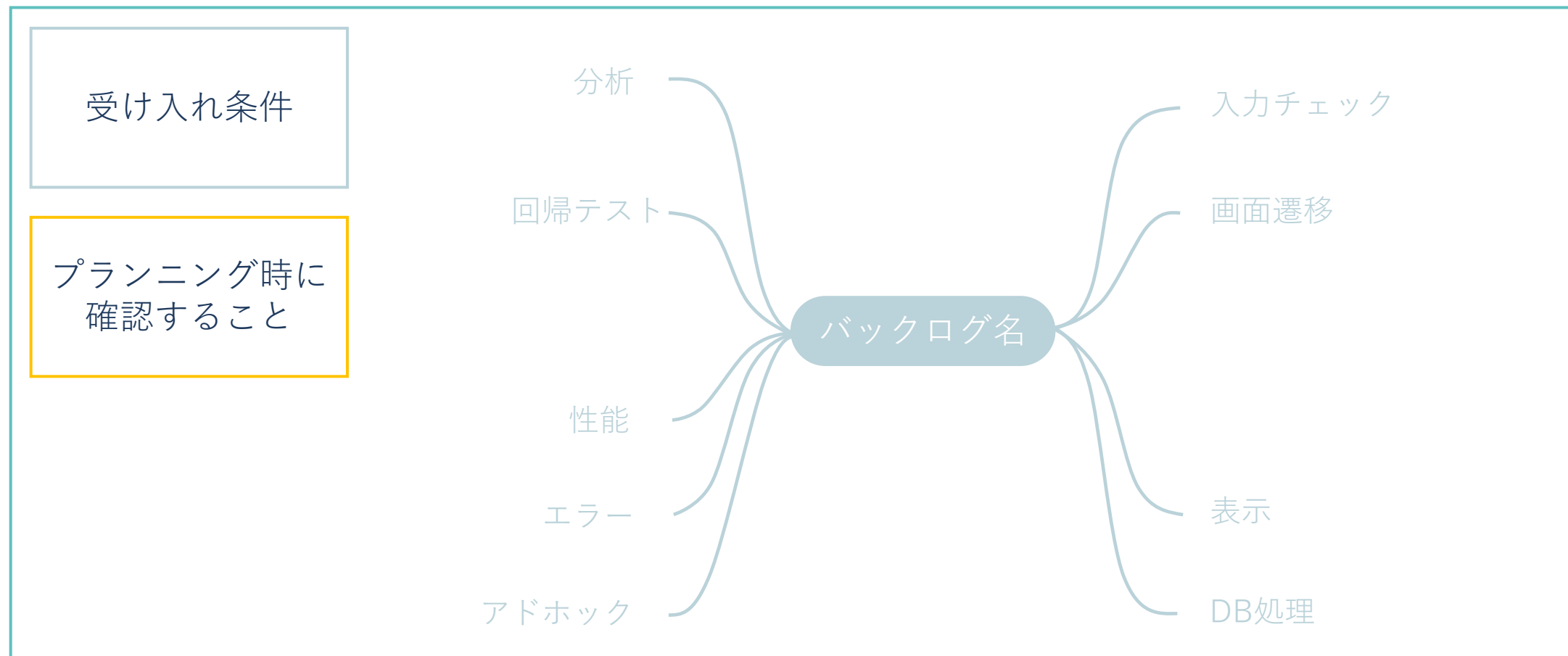
さらに、調査バックログなどテスト設計・実施が不要なバックログが多いスプリントでは次のスプリントのテスト設計を前倒して行える場合も

スプリント開始時にテスト設計を共有



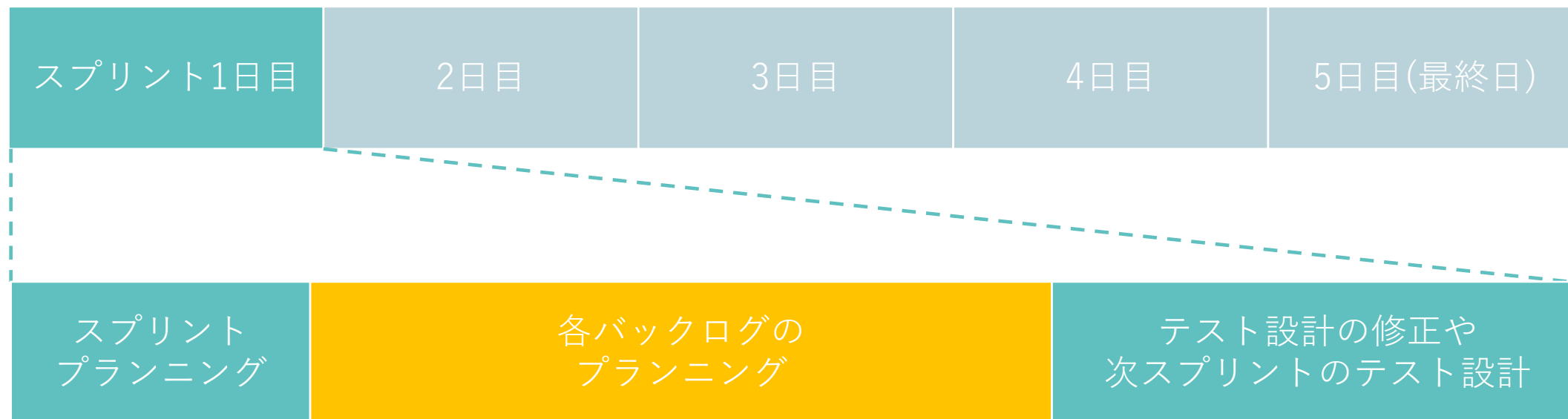
前倒しで行ったテスト設計を、スプリント開始時に開発チームに共有する取り組みを開始

スプリント開始時にテスト設計を共有



[プランニング時に確認すること]という項目を追加し、実装前に相談する内容を用意

スプリント開始時にテスト設計を共有



スプリントプランニング後に、各バックログごとに詳細なプランニングを実施している
→ ここでテスト設計を共有することに

スプリント開始時にテスト設計を共有



デザイナー



Dev



QA



ライター

PM以外の全員がスプリントプランニング直後に集合

スプリント開始時にテスト設計を共有



デザイナー



QA

JIRA

ユーザーストーリー

受け入れ条件

デザイン仕様のリンク

テスト設計のリンク

コメント



Dev



ライター

バックログ管理に使用しているJIRAの画面を共有しながら進める

スプリント開始時にテスト設計を共有



デザイナー



QA

JIRA

ユーザーストーリー

受け入れ条件

デザイン仕様のリンク

テスト設計のリンク

コメント



Dev



ライター

ファシリテーター(参加者全員の当番制)がユーザーストーリーと受け入れ条件を読んで全員で確認

スプリント開始時にテスト設計を共有



デザイナー



QA

JIRA

ユーザーストーリー

受け入れ条件

デザイン仕様のリンク

テスト設計のリンク

コメント



Dev



ライター

デザイナーからバックログごとにデザインについて説明(デザイン仕様へのリンクを記載)

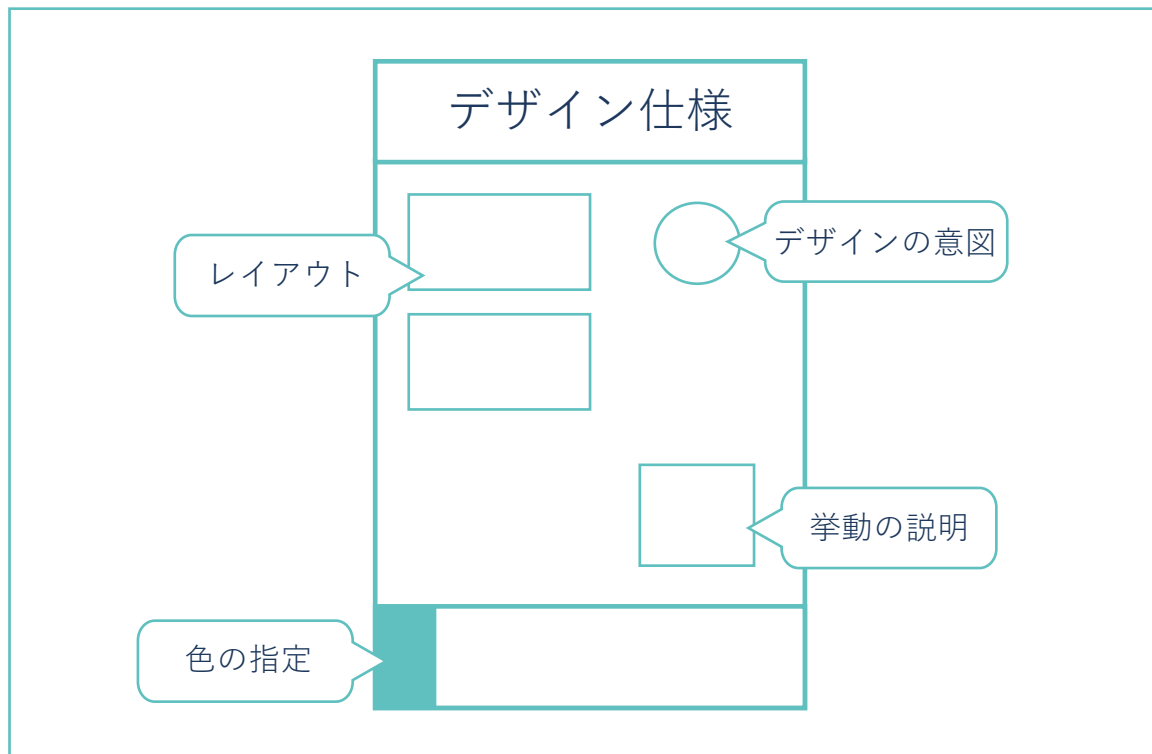
スプリント開始時にテスト設計を共有



デザイナー



QA



Dev



ライター

デザインや色の指定の意図、ユーザーにどう操作して欲しいかなど確認

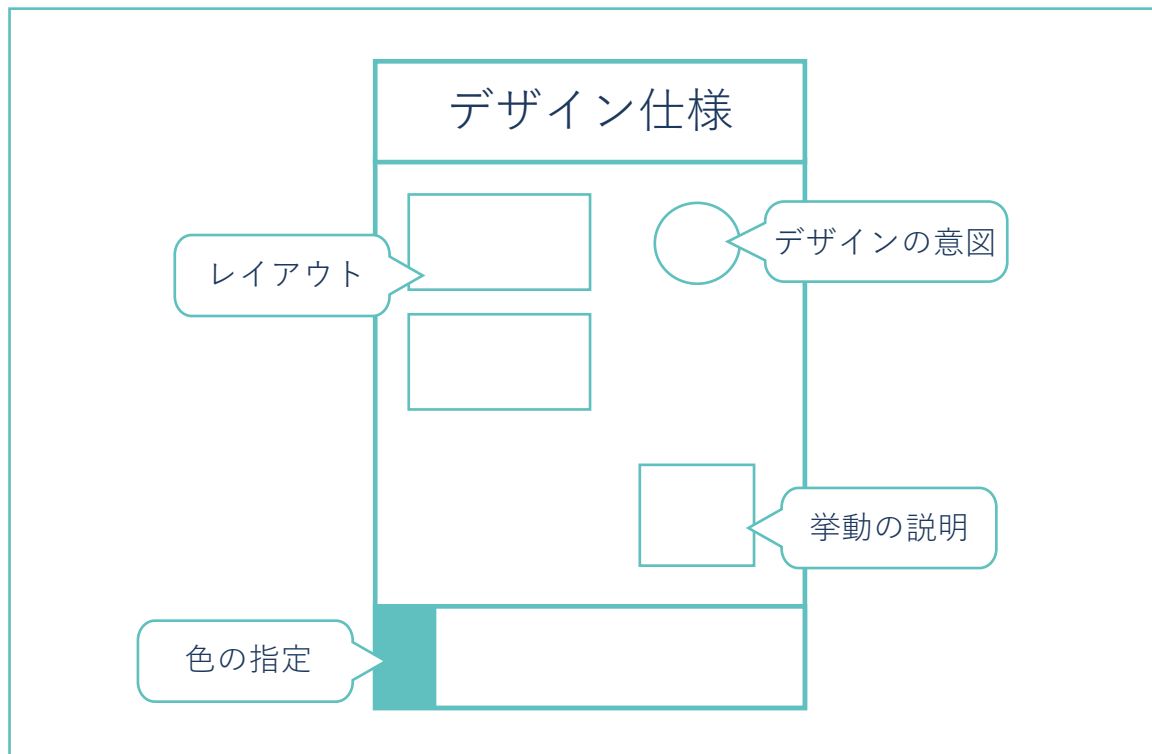
スプリント開始時にテスト設計を共有



デザイナー



QA



Dev



ライター

Dev・QA・ライターが気になるポイントを質問し、仕様不備が無いか、ユーザーストーリーを満たせるかななどを議論

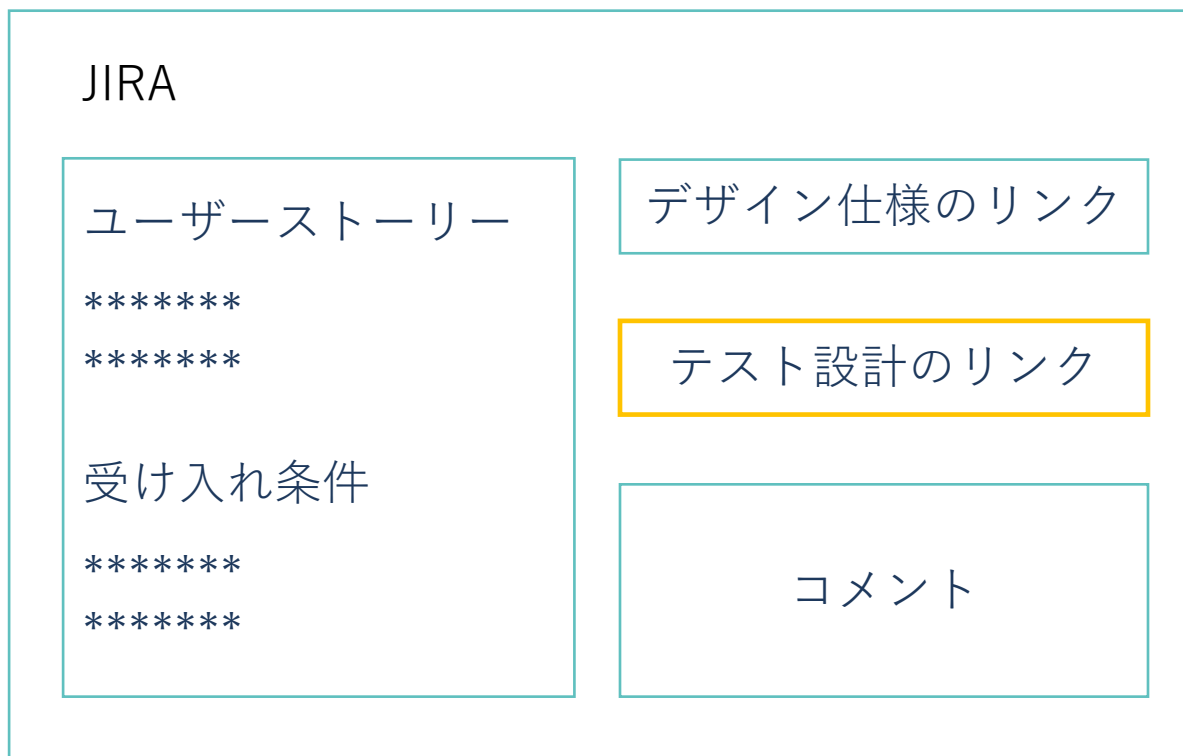
スプリント開始時にテスト設計を共有



デザイナー



QA



Dev



ライター

デザイン確認後にQAエンジニアが作成したテスト設計を確認

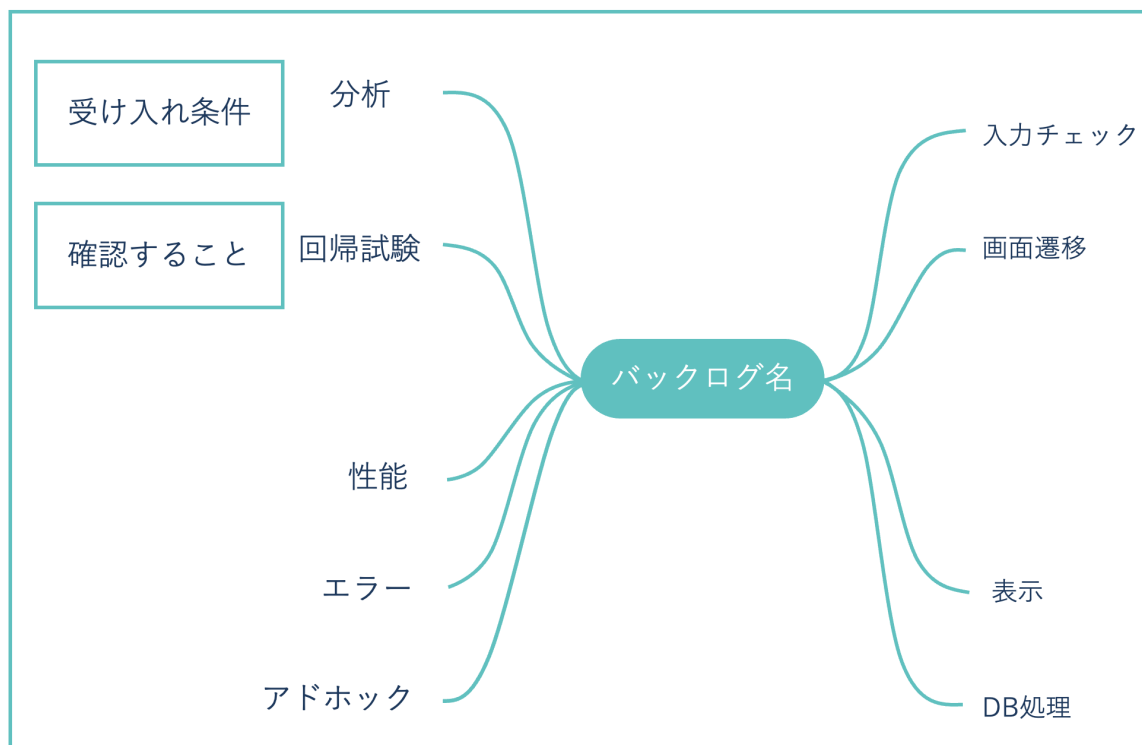
スプリント開始時にテスト設計を共有



デザイナー



QA



Dev



ライター

テスト方針や具体的なテストケースを説明、事前に用意した「確認すること」もここで解消

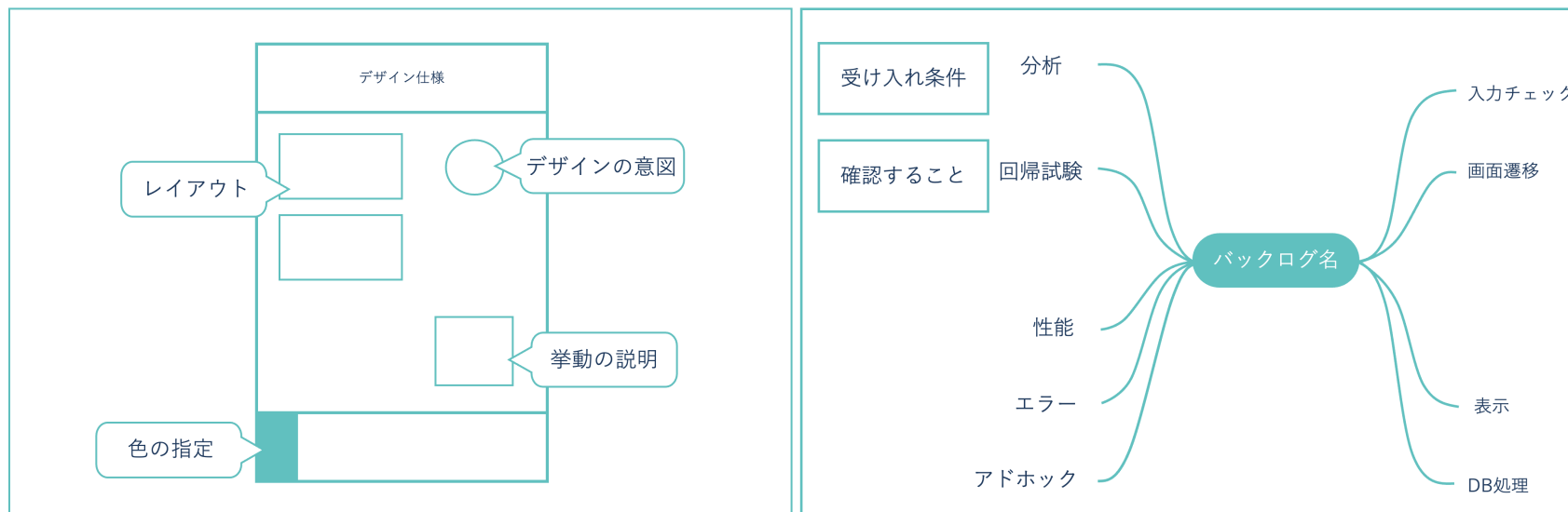
スプリント開始時にテスト設計を共有



デザイナー



QA



Dev



ライター

実装の影響範囲、エラーケースの文言/復帰方法、仕様の考慮漏れなど
各機能がデザインとテスト設計を眺めながら、バックログを確認

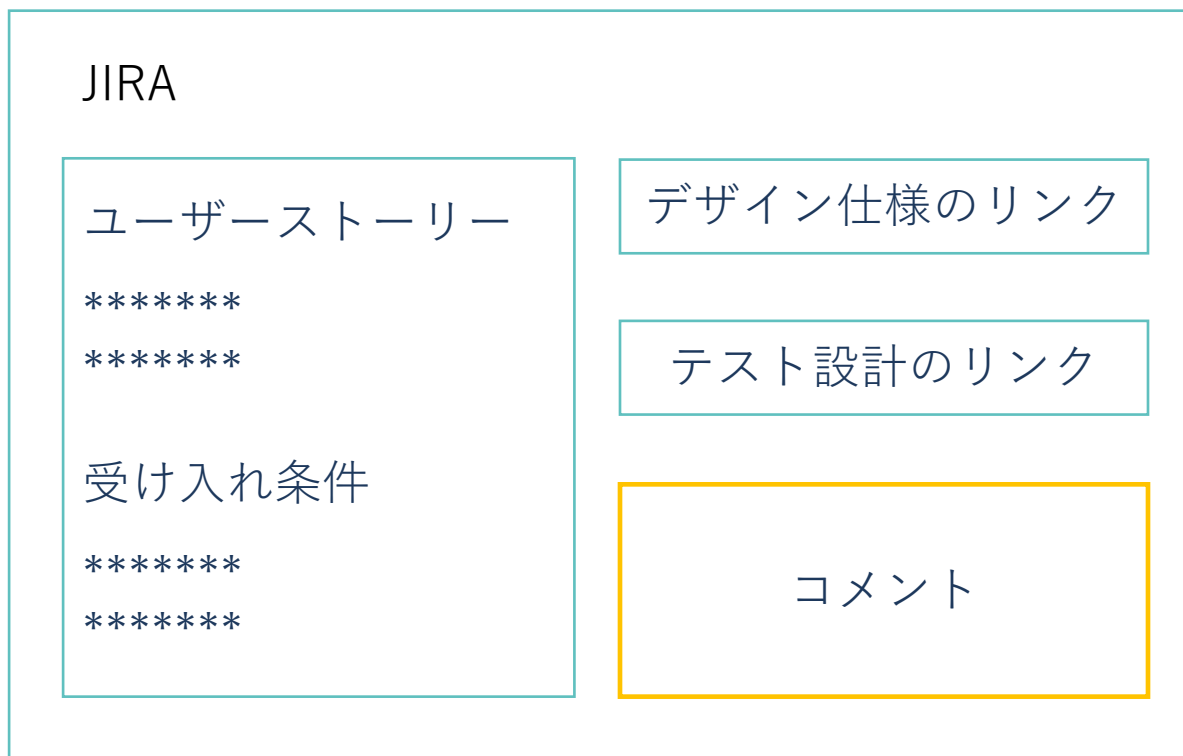
スプリント開始時にテスト設計を共有



デザイナー



QA



Dev



ライター

議論内容はファシリテーターがJIRAのコメント欄に記録

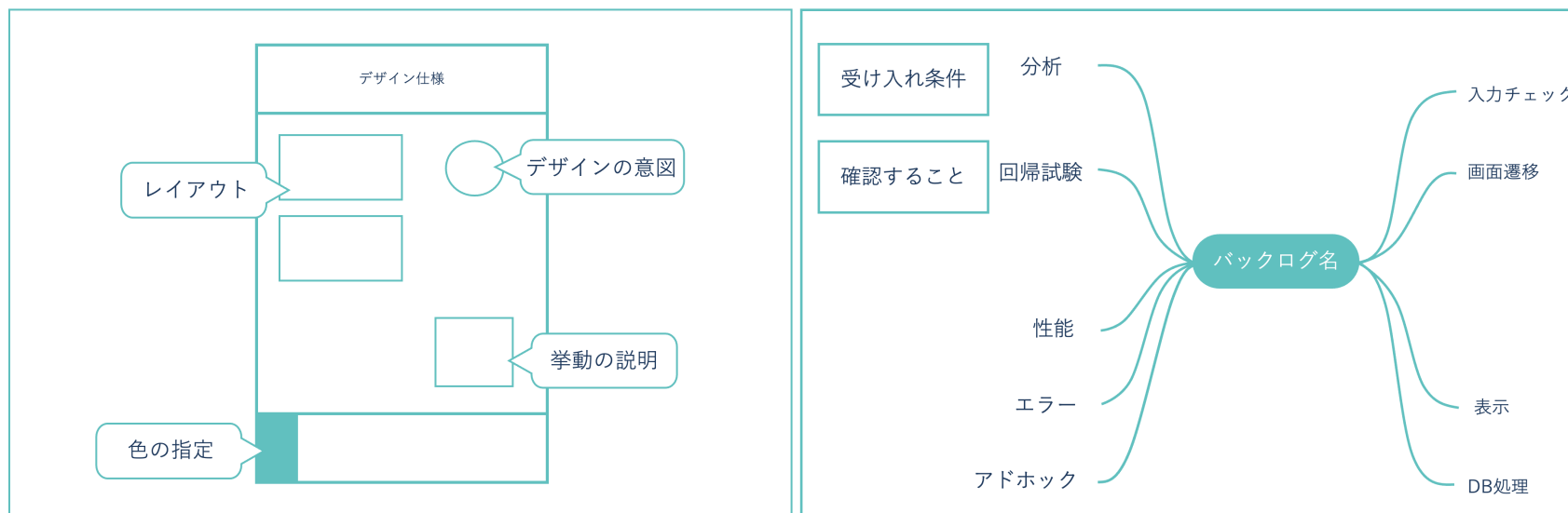
スプリント開始時にテスト設計を共有



デザイナー



QA



Dev



ライター

実装前に、デザインや文言含めて不具合を未然に防ぐ取り組みを行うことで
テスト実施から不具合登録、トリアーージなどのコストが減少した

目次

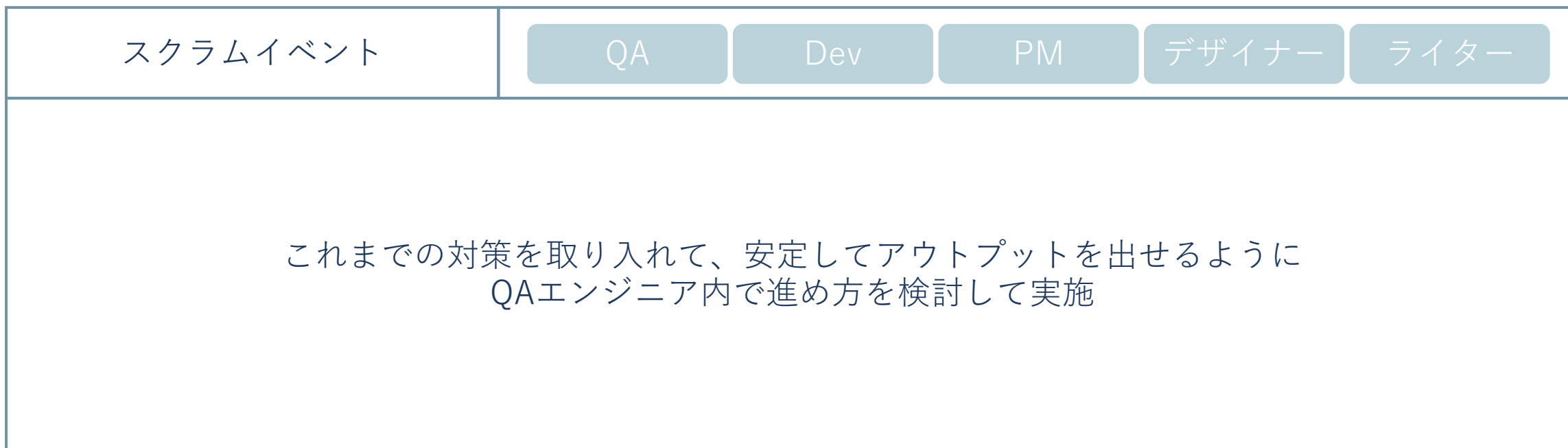
背景

対策と効果

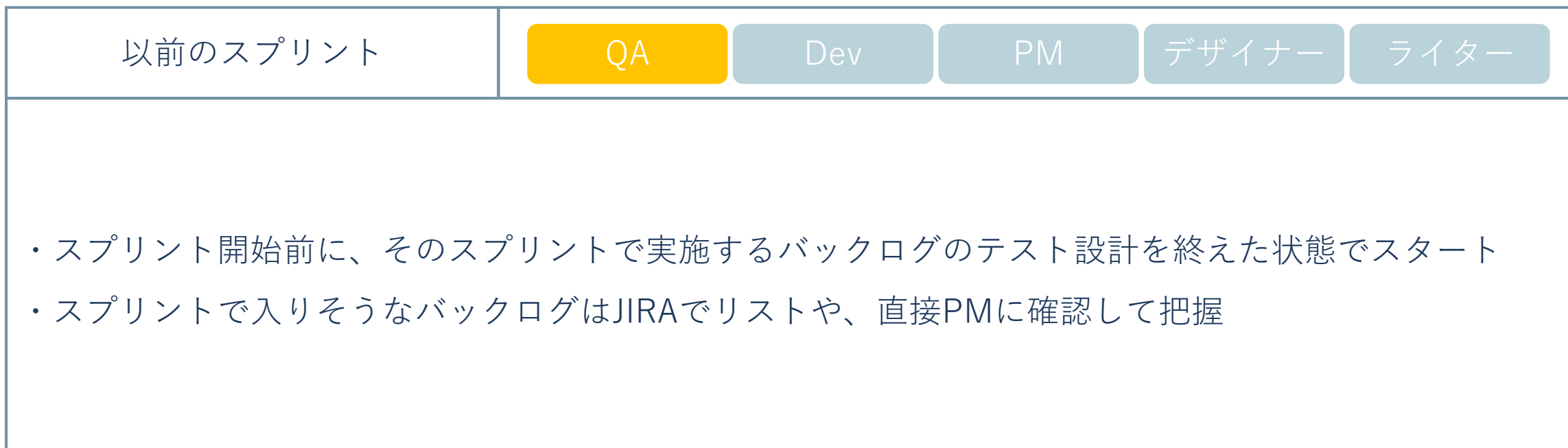
課題

対策と効果	◀ テスト設計コストの軽減	マインドマップの活用
		モブの活用
課題	手戻りの軽減	スプリント開始時にテスト設計を共有
		テスト設計を前倒しで行うフローに変更

テスト設計を前倒しで行うフローに変更



テスト設計を前倒しで行うフローに変更



テスト設計を前倒しで行うフローに変更



- ・ 基本的にはDevの実績に基づいて、スプリント内で対応できるギリギリの開発量でバックログを選択
- ・ 大体1~2日程度で開発できる量に分解したバックログを、10個程度選択
- ・ テスト実施量が多いバックログが多数入る場合はQAエンジニアから申告して調整

※事前にテスト設計を終えているため、QAエンジニアの作業量の観点からも調整を行える

テスト設計を前倒しで行うフローに変更



- ・ スプリントプランニング実施後に開発チームで実施
- ・ 事前に作成済みのテスト設計を見せながら、影響範囲の確認や考慮漏れなど検討
- ・ 大体1バックログ10分～30分程度、全体で2～4時間程度かけて実施（延長する場合も）
- ・ 不明な点があれば都度PMを呼んで確認

テスト設計を前倒しで行うフローに変更



- ・各バックログ確認後はそれぞれの作業に入る
- ・必要があれば先ほどの各バックログの確認で判明した修正点などをテスト設計に反映
- ・修正作業が終わって実装までの空き時間ができたら、次以降のスプリントのバックログをテスト設計

※メンバーによっては採用業務などを受け持っているので、その作業にあてる場合も

テスト設計を前倒しで行うフローに変更



- PM含めた開発チーム全員が集まり、スプリントゴールを達成できそうか1~5の数字を各自表明
- 1が最も悲観的、5が最も楽観的な数字で、1~3を表明したメンバーがいたら、その理由を確認
- 内容によって、今日の予定を変更して別途解決策を話し合う時間を設ける
- 昨日までに報告された不具合と問い合わせの確認も行い、必要ならスプリント内での対応を検討
- 追加作業によってスプリントゴールに影響ある場合は、スプリントゴール自体を見直し

テスト設計を前倒しで行うフローに変更



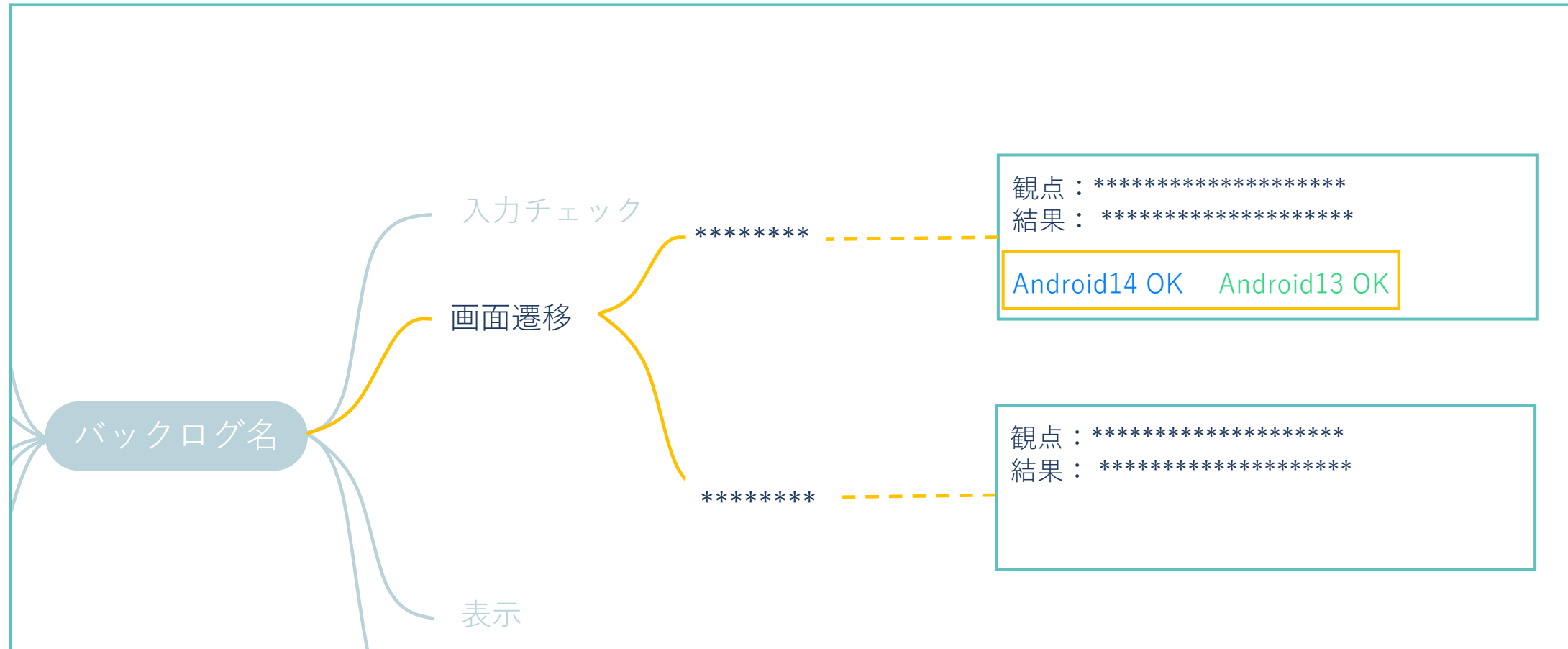
- ・スプリント初日以外は毎日実施(30分)
- ・今スプリントの追加バックログ(不具合など)や、次以降のスプリントで対応するバックログを検討
- ・デザイナーから次以降で対応する新機能のデザインの頭出しもある
- ・リファインメントが終わったバックログからテスト設計するので、疑問点など明確になるまで質問
- ・リファインメントで提供される情報が、テスト設計を行う際のテストベースになる

テスト設計を前倒しで行うフローに変更



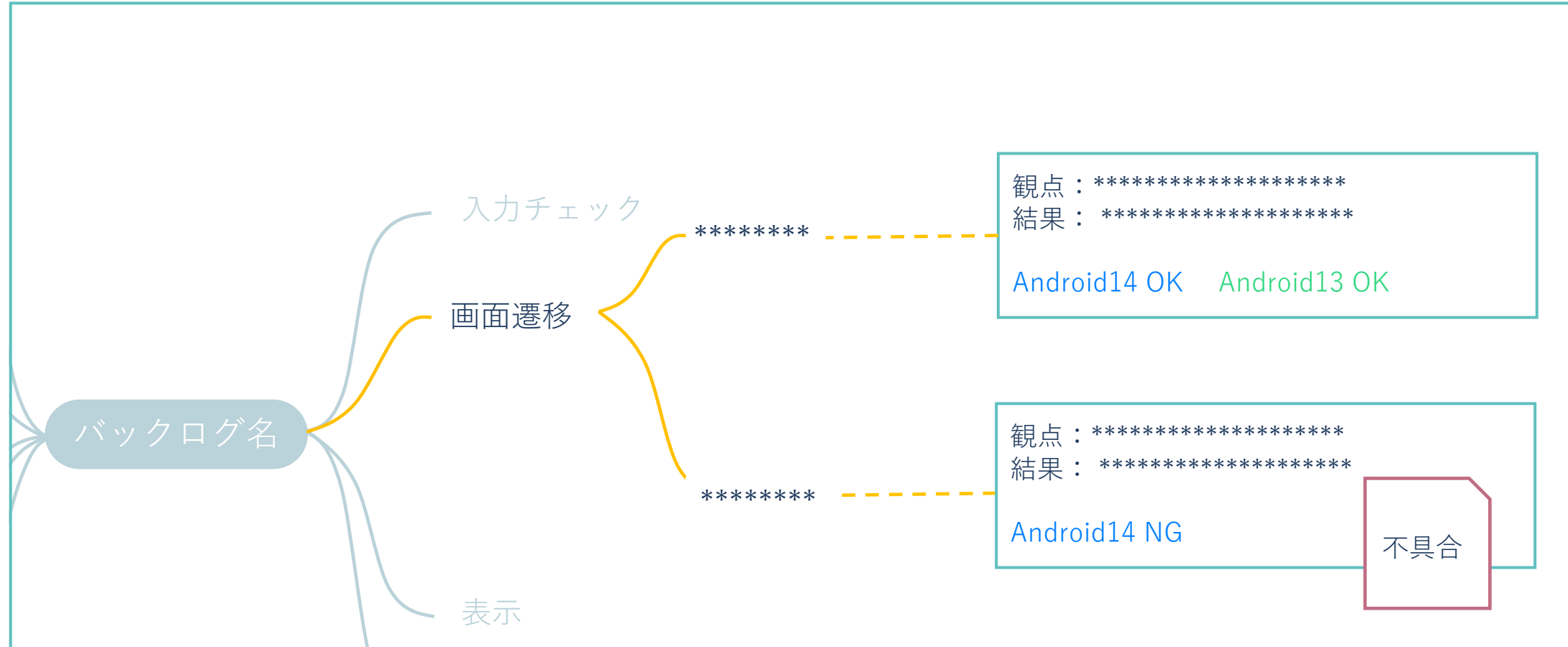
- ・ 2日目以降の個人作業はリファインメントが完了したバックログのテスト設計や、実装が完了したバックログの試験実施を行う
- ・ 不具合があった場合はBTSに登録するが、優先度が高いと判断した不具合は朝会(デイリースクラム)を待たずに開発チームに共有して対応を検討する

テスト設計を前倒しで行うフローに変更



試験実施の結果はmiroのカードにある「タグ」という機能を使って表現

テスト設計を前倒しで行うフローに変更



不具合がある場合は予め決めた色の付箋で内容を記載し、BTSにも別途登録

テスト設計を前倒しで行うフローに変更



QA MTG	QA	Dev	PM	デザイナー	ライター
<ul style="list-style-type: none">・ QAエンジニアだけのミーティングを毎日開催(基本30分~1時間で、日によっては1時間程度追加)・ 個人作業で行ったテスト設計のレビューモブはこの時間に実施・ レビュー時に回帰試験も作成し、リリース前に実施できるように準備・ リファインメントを行なってテスト設計可能になったバックログの担当の割り振りも行う・ 他に相談事項や検出した不具合の共有、採用などチーム運営に関わる相談も扱う					

テスト設計を前倒しで行うフローに変更



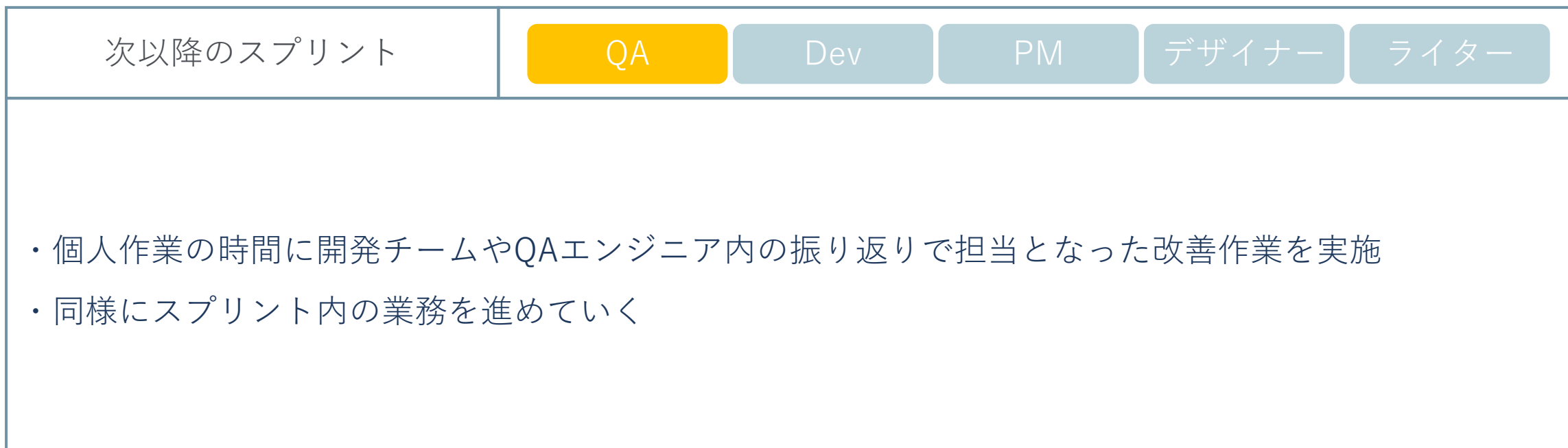
- ・以前はスプリントレビューまでテスト実施ができてなかったが、現在はほぼ完了した状態で迎える
- ・ユーザーストーリーに沿ったデモを行い、各メンバーが気になった点を口頭で伝えたり、チャットにコメントを残して、デモの後に拾う方式
- ・品質に対するフィードバックや、関係する不具合を共有する場合も
- ・実装完了が直前になる等の理由でテスト実施が完了しなかったバックログは次スプリントに持ち越し

テスト設計を前倒しで行うフローに変更

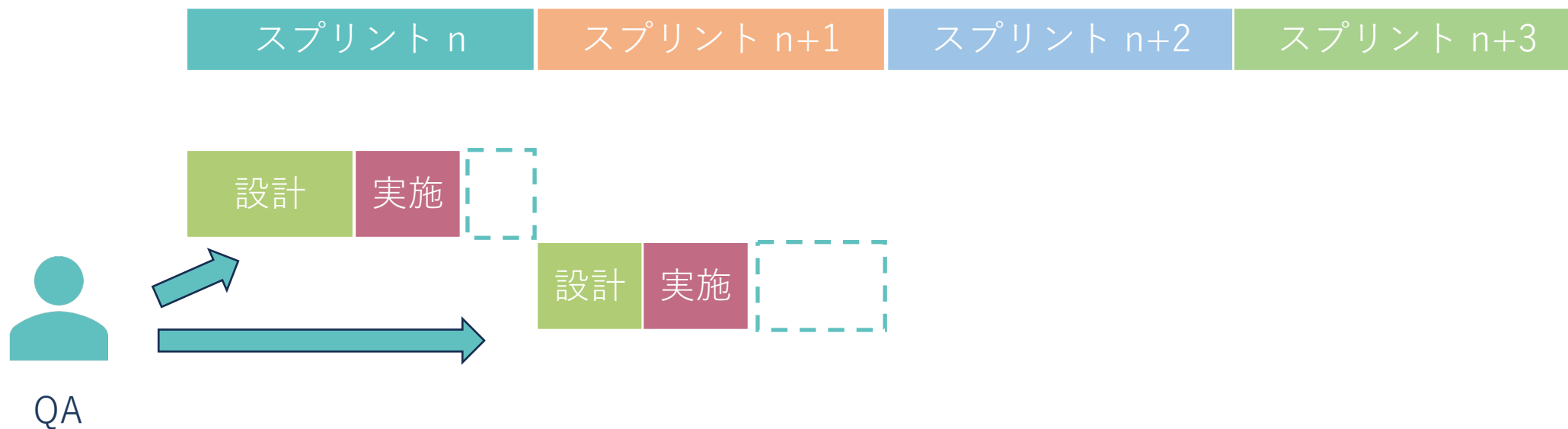


- ・開発チーム全員参加でレトロスペクティブ(振り返り)を実施
- ・スプリントゴールを達成した場合、できなかった場合で改善点や良かった点を確認
- ・開発プロセス全体の改善点や、QAエンジニアの作業の困りごとなども共有
- ・これ以外にもQAエンジニアだけの振り返りも月1回実施し、モブのやり方など改善点を探る

テスト設計を前倒しで行うフローに変更

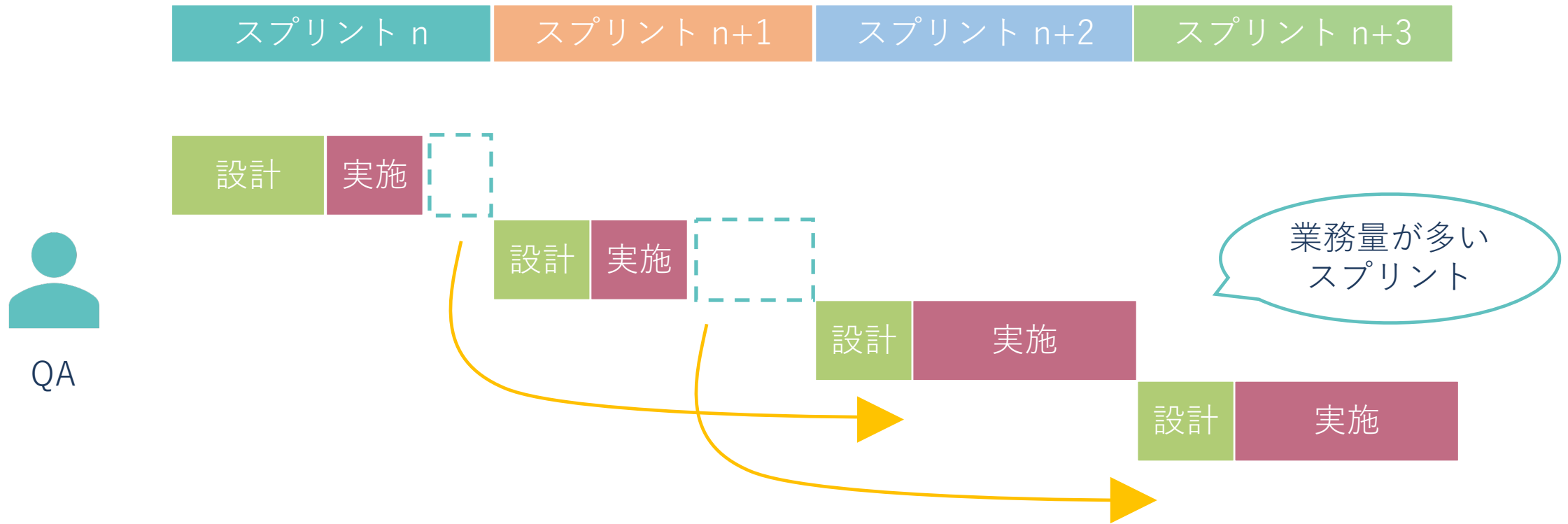


テスト設計を前倒しで行うフローに変更



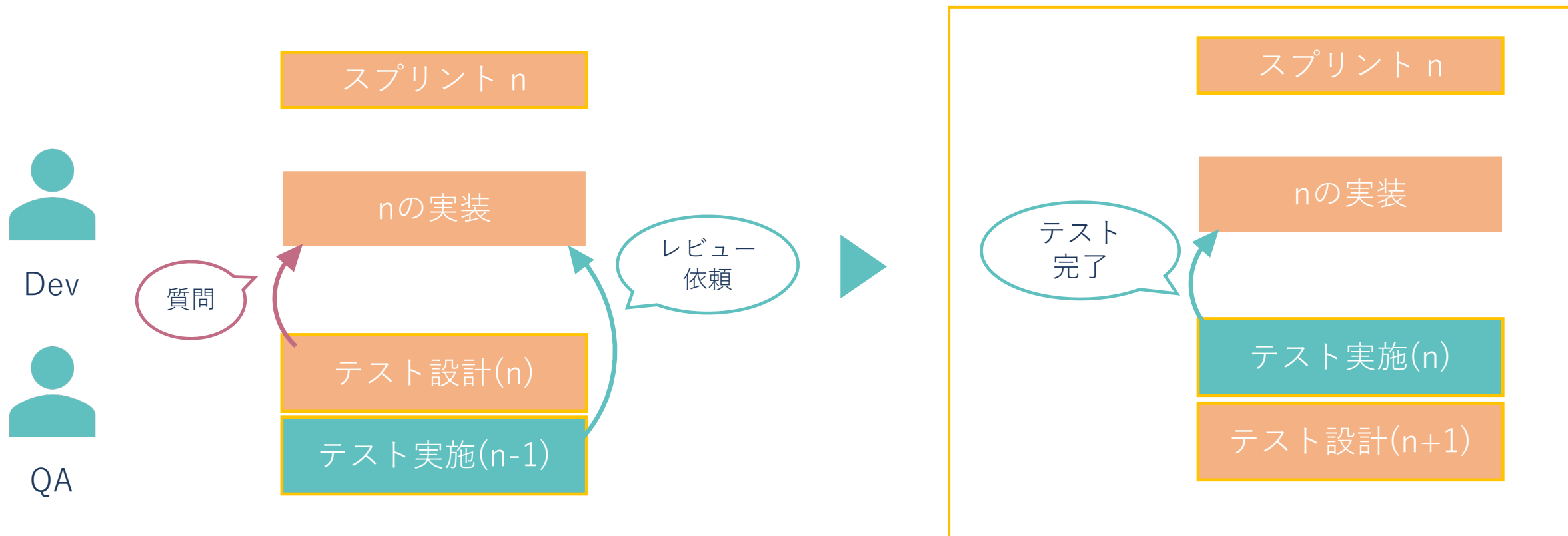
リストにあるバックログは全て事前にテスト設計するフローに変えたことで、次以降のスプリントの作業量を大体把握し、スプリント開始時に共有できるようになった

テスト設計を前倒しで行うフローに変更



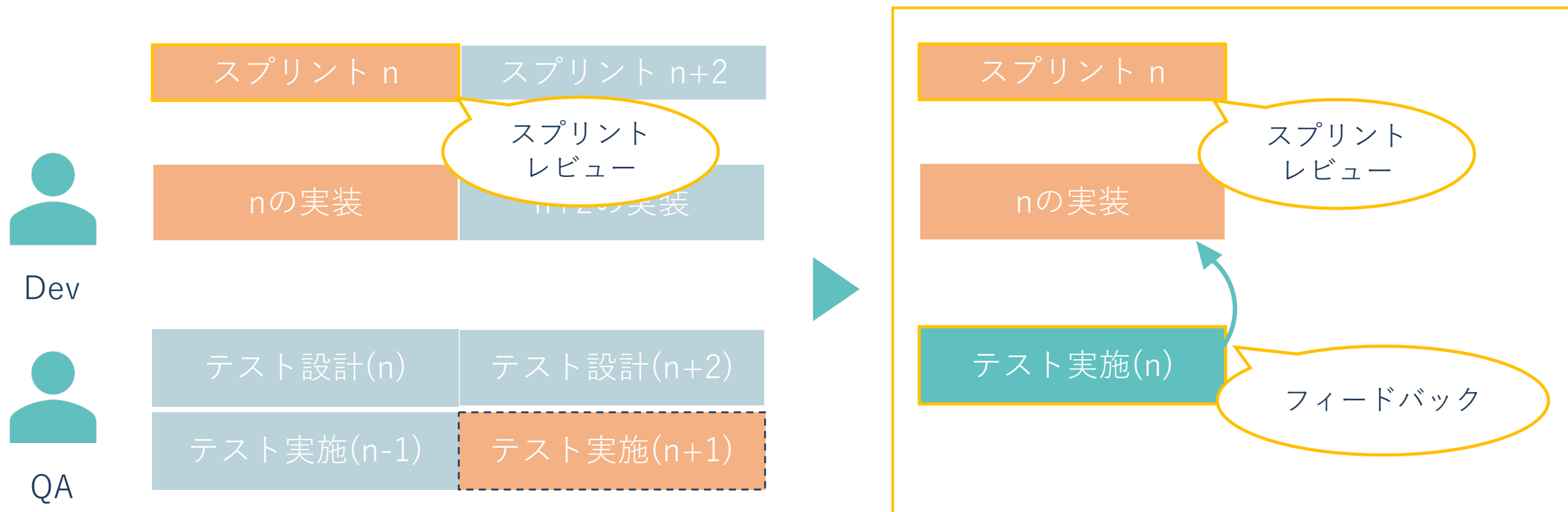
比較的業務量が少ないスプリントのときに、テスト設計を進めておくことができるので各スプリントの業務量を平準化できる効果もあった

効果まとめ



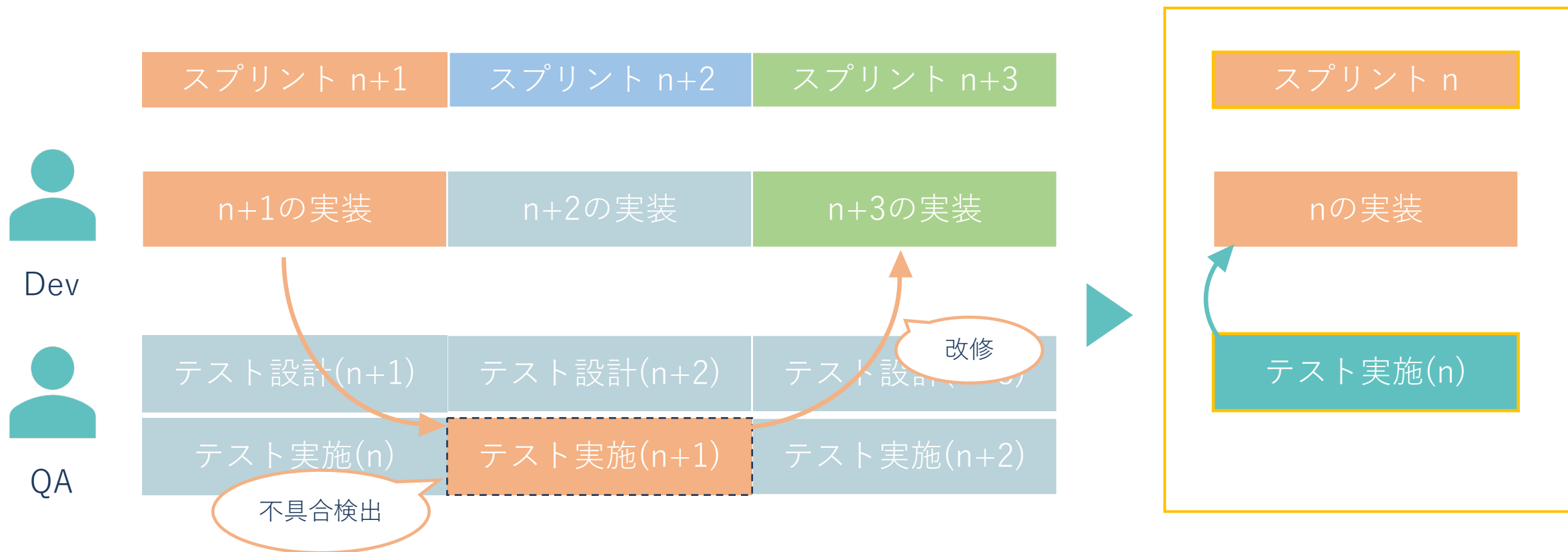
Devと同じスプリントでテスト実施完了し、フィードバックできている
また、実装待ちなどの時間で次以降のスプリントに備えてテスト設計を行う状態

効果まとめ



スプリントレビューの時点で、QAエンジニアから品質のフィードバックを終えている

効果まとめ



検出した不具合も、対応できる場合や優先度が高い場合は同じスプリントで改修

目次

背景

対策と効果

課題



目次

背景

対策と効果

課題



対策後の課題

対策後の課題

テスト実施量が多いスプリントが続いた場合でも、継続してテスト設計済みのバックログを貯めるためのより安定した仕組み作り

→

プランニング時に調整&業務量を平準化できつつあるが、単純に業務量ギリギリのスプリントが続くと徐々にQAエンジニアの工数が厳しくなるタイミングがあるため

取り組み

- ・テスト分析をモブで、テスト設計は個人で行い、テスト設計者からの希望制でレビューする
 - ・テスト分析を体系的に行えるような仕組みを入れて、分析やレビューをよりスムーズに行う
- など、次の改善策を検討中

対策後の課題

対策後の課題

より納得感のある & 効率的な回帰テストの作成

→

テスト設計レビュー時に回帰テストを併せて作成するフローを採用して、ハッピーパスや不具合が発生した場合の改修優先度を元に検討しているが、まだまだQAエンジニアの感覚に頼っている状態。

取り組み

取り組みとしては具体的な改善策は検討中という状況。

また有償ツールやXCUITestなどを使用して回帰テストを自動化しているため、そちらのより効率的 & 安定的な実装プロセスも検討中。

対策

- テスト設計をマインドマップで行い、モブでレビューを行う体制にした
- スプリント開始前にテスト設計を実施し、プランニング時点で開発チーム内で確認するプロセスにした



対策後の変化

- Devと同じスプリントでテストを実施し、フィードバックを行えるようになった
- 不具合があった場合も、同じスプリントで改修できるようになった
- 不具合リスクをなるべく事前に潰して手戻りを少なくする取り組みにつながった
- 作業の見通しが良くなり、各スプリントの業務量の平準化につながった

紹介した取り組みが
少しでも参考になれば幸いです
ありがとうございました

サイボウズではQAエンジニアを募集中しています。
キャリア採用サイト公開中です！

