ソフトウエア品質シンポジウム 2020

プロダクトマネージャーが考えるQAと歩むアジャイルなチーム作りとQAの越境体験

株式会社リクルートライフスタイル
データソリューションユニット　マネージャー
◯坂東塁

株式会社ProVision
羽鳥温子

RECRUIT
リクルートライフスタイル

Agenda

- 背景・課題
- 目指すべき方向
- 取り組み・プロセス
- 実施結果
- 結論と今後に向けて

Agenda

- **背景・課題**
- 目指すべき方向
- 取り組み・プロセス
- 実施結果
- 結論と今後に向けて

背景：JaSSTに参加して”QA”の状況を知る

PMとして改めて効果的なプロダクト開発のために各役割を整理
各役割を整理する中、**”QA”**知見の把握のため**JaSST'19**への参加
”QA”を取り巻く状況と課題、そして課題解決に向けたノウハウを知る

JaSSTで上がっていた
“QA”における負

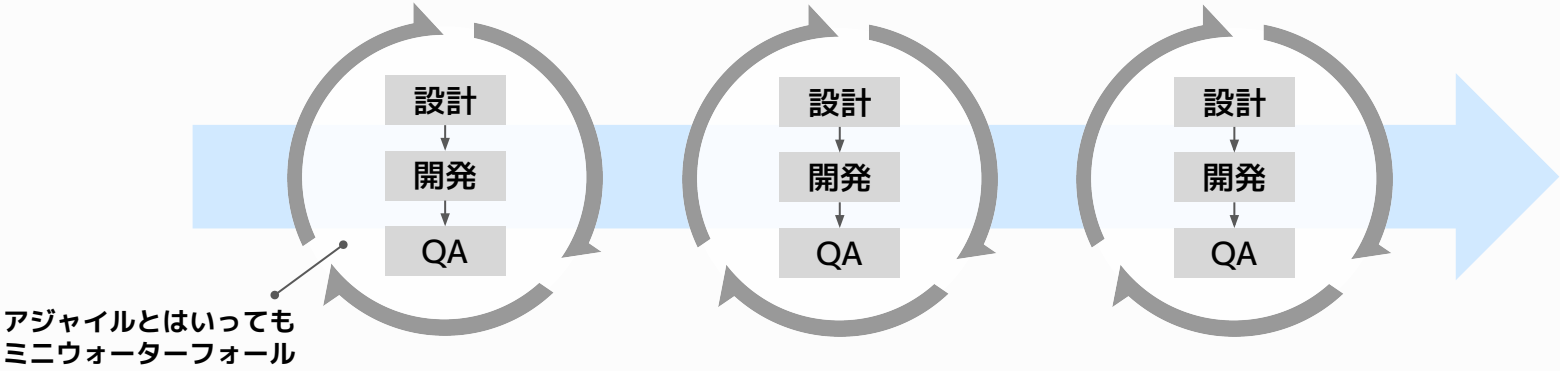
- **仕様認識**
- **新しいテクノロジーへの対応**
- **QA内での問題完結**
- **モチベーション**

自分たちの組織の”QA”の状況は？

アジャイルチャレンジにおける”QA”課題

”じゃらんnet”の開発の一部をウォータフォール開発からアジャイル開発に変更し、生産性の向上に向けてチャレンジを続けてきた。しかし、生産性向上に向けてアジャイルを意識しただけでは、ミニウォーターフォールという傾向が見られ、開発メンバーやQAメンバーの役割が不変であるこも多く、その結果として期待したパフォーマンスに繋がらない可能性もあった。

■役割分担によるスタイル例

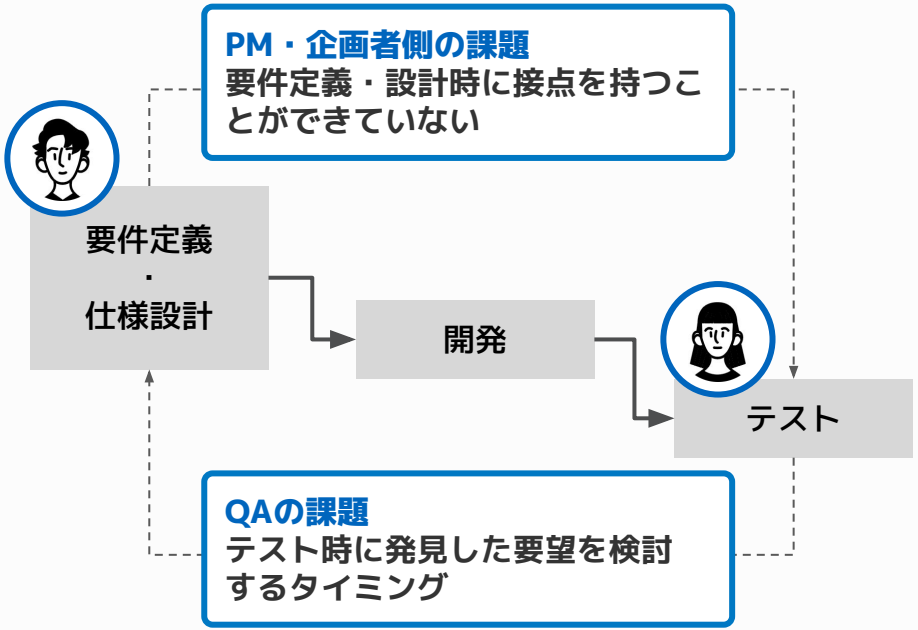


新たなの変化として”QA”の課題に目を向ける

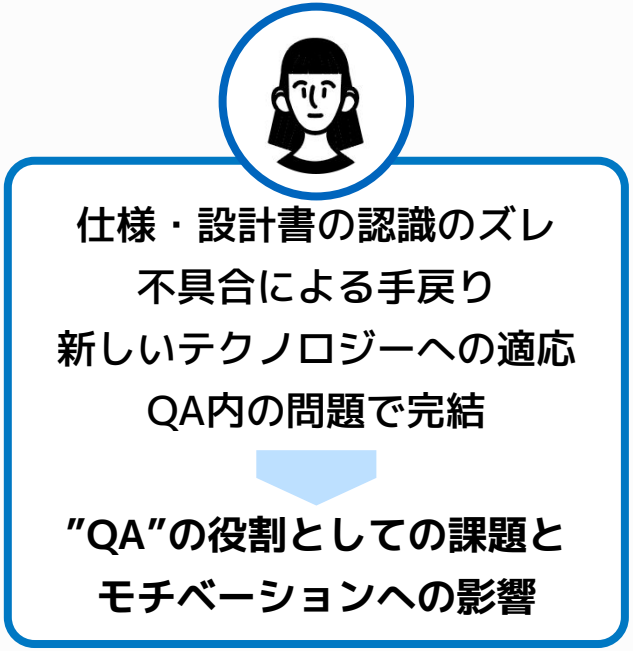
※企画者の認識の変化、リリースタイミングや開発の規模の変化のみ

自組織における”QA”課題について

タイムボックス的な開発における接点の壁



外部で聞いた内容と同様の負が存在



”QA”課題をチームとして考える

”QA”課題例	チームにおける課題として考える
仕様認識のズレの発生による負	❑ 企画者が意図した仕様が担保されないことで、手戻りなどが発生 ❑ QA項目書が仕様書を正とした設計になることで、仕様・設計外の内容に関しては、QAテストによる品質が担保されずに、品質の低下リスクに繋がる ❑ 正しい情報確認にための無駄なやりとりの発生に繋がり、全体の工数を引き上げることに繋がってしまう
不具合と手戻り	❑ 生産性を期待・意識するあまり、QAテストへ回されるプロダクト品質の低下によって不具合の増加（バグが多いことで、手戻り・バグ再確認の発生増加など）に繋がりQAの負担の増加 ※アジャイル開発に変化したことで、プロダクトの開発フローにおける品質への意識に変化（ウォーターフォール開発＝”厳しい”・アジャイル開発＝”緩い”という意識など）によって生まれた可能性 ❑ メンバー間（開発⇔QA間）での「品質担保」の意識のズレによるQAの手戻り工数の増加
テクノロジーへの適応の遅れ	❑ 役割を固定化されていることで、仕様面にだけ目を向けてしまい、テクノロジー面が疎かになる ❑ 新しいテクノロジーに適応できないことで、QAテストの正確性が担保されていないことに繋がる可能性 ❑ 新たなテクノロジーの採用をした際に、各役割において認識の齟齬が生まれる可能性がある
QA内での問題で完結	❑ QAチーム内で”QA”課題が完結される（QA内での努力で終わる）ことで、チームとして解決すべき課題が解決されずに、結果としてチーム全体のパフォーマンスが改善されない（パフォーマンスの低下） ❑ QAチーム内だけに閉じることで、役割の固定化が進み、情報共有が少なくなる結果として、プロダクト開発のノウハウの越境がなくなる（パフォーマンスの低下）

各”QA”課題は、チームにおける課題であるべきではないか？

QA内での改善努力だけで終わらせて解決できるものは少なく、”QA”課題はチームにおける課題でもある

”QA”を取り巻く状況と課題

アジャイルへの取り組み

アジャイルという意識だけでは”QA”の課題は生まれ続ける

- アジャイルと言えでも小さいウォーターフォールのような体系を突破できないことが多く、その中で両者を隔てる壁は残る

”QA”課題の顕在化

役割の固定化による弊害と立ち位置の課題

- ウォーターフォールでもアジャイルを意識した取り組みにおいてもQAの立ち位置（役割）には変化がなかった
- 明確に役割分担された中で、QAが介在するタイミングや役割はいつも同じ（ウォーターフォール/アジャイルでも変わらなかった）

大くのマイナスが発生

課題から生まれるプロジェクトへの影響

- 仕様漏れの増加や仕様把握のスピード低下・不具合の増加
- 各役割での認識齟齬によるプロジェクトパフォーマンスの低下
- モチベーションにも影響し、開発効率の悪化

プロジェクトのパフォーマンス最大化のため解決すべき課題

Agenda

- 背景・課題
- 目指すべき方向
- 取り組み・プロセス
- 実施結果
- 結論と今後に向けて

”QA”課題に対して目指すもの

QAの課題ではなく、チームの課題として考える

チームとしてのパフォーマンスを最大化するために
QAの立ち位置（役割）を再定義することに取り組む

”QA”の立ち位置（役割）を整理した上で再定義へ

- **目的はチームのパフォーマンス（開発効率）の最大化**
  - QAメンバーの課題を無くすことが目的ではなく、チームのパフォーマンスを最大化する過程でQAの課題解消を目指す
- **“QA”の役割を改めて見直し、各フェーズでのQA立ち位置を再考する**
  - これまで”シフトレフト”によるQAメンバーの上流工程への染み出しという意識だけで、”QA”の役割の見直しが置き去りにされていた
- **QAメンバーが”テスト”だけをやる体制は正しいのかの見直し**
  - 「品質担保」はテストフェーズでの”テスト”によって、QAメンバーが担保するという意識が強かった
  - 本来は最もプロダクトに触れる時間の長い”QAメンバー”の意見は重要であることを理解
  - 品質担保のための”テスト”はQAメンバーがテストフェーズで実施するという考えを見直す必要があった

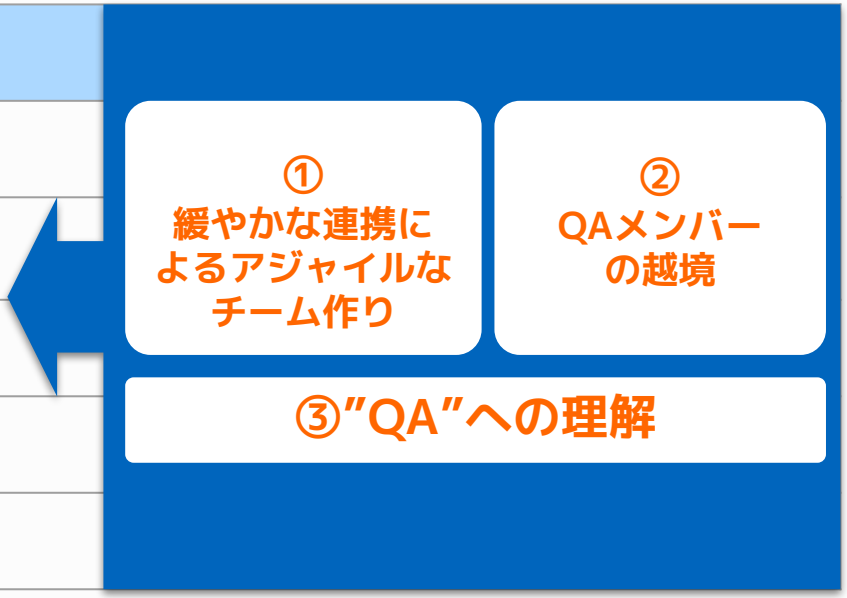
Agenda

- 背景・課題
- 目指すべき方向
- 取り組み・プロセス
- 実施結果
- 結論と今後に向けて

”QA”課題へのアプローチを考える

”QA”における課題へのアプローチ

”QA”における課題
仕様認識
不具合・手戻りの負
新しいテクノロジーへの適応
QA内での問題で完結
モチベーションの低下



アプローチ【１】～緩やかな連携とアジャイルなチーム作り～

実現のための課題整理

ゴールが曖昧なプロダクトにおける開発スタイルの整理

これまでのリーンアプローチは、すでに存在するプロダクトの改善に主に用いられていたため、課題はあるもののミニウォーターフォールのような固定化された役割とタイムラインでも稼働していた。
ただし、ゴールが曖昧なプロダクトにおいては、より柔軟な役割とタイムラインが求められる

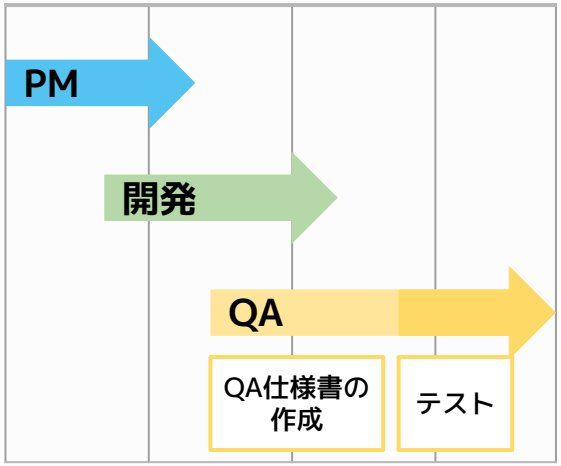
必要となるポイント整理

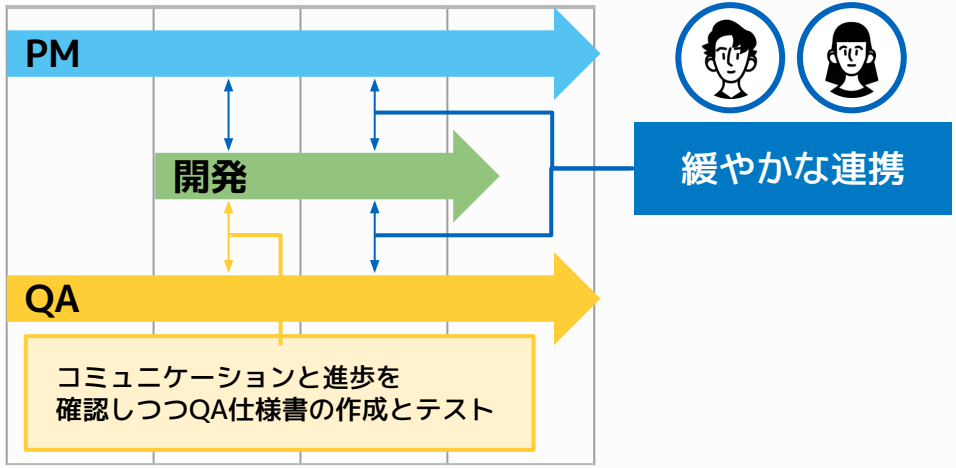
これまでのリーンアプローチの見直しすることで アジャイル開発に求められるポイントを再整理

アプローチ【１】～各役割間の緩やかな連携～

企画者・開発者との緩やかな連携による改善

固定化されたタイムボックスの中で動くのではなく、PM（企画者）や開発者と緩やかな連携を図れるようにする。

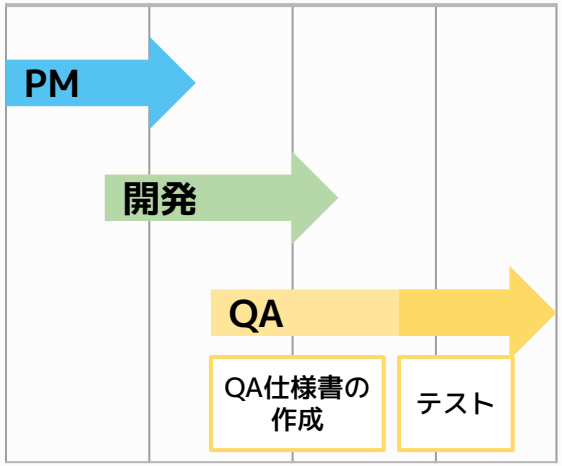




- アジャイルな意識を持っても、ミニウォーターフォールであることも多いため、各役割のタイムラインは変化せずに、QAが関わるタイミングもウォーターフォールの時と同様

- 明確なテスト期間を設けずに開発の進捗に応じたテスト設計・テストを実施。
- 常に各役割のメンバーの状況を把握することで、柔軟にタスクに取り組めるようにする。

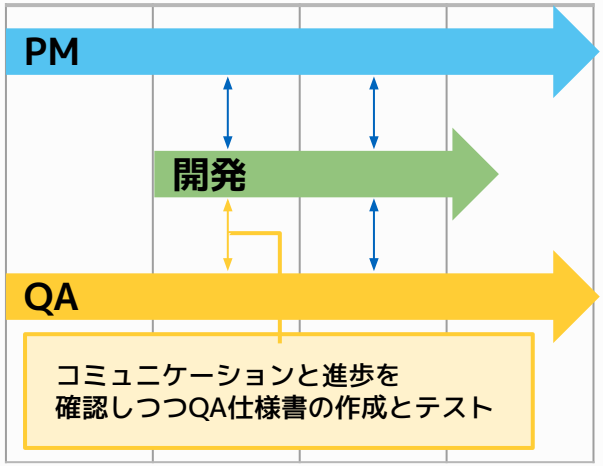
アプローチ【１】～各役割間の緩やかな連携～



- ❏ QAが関わるタイミングは開発終盤以降
- ❏ タイムボックスが固定化されているため、柔軟に動くことができない

Before	
起案タイミング	❏ 携わるタイミングなし
開発前	❏ 携わるタイミングなし
インプット	❏ PMから一方通行
開発中	❏ QA業務（項目書作成など）のみ実施
開発完了後	❏ 全ての開発が完了した時点で、テスト開始
リリース後	❏ QA内部で振り返り実施

アプローチ【１】～各役割間の緩やかな連携～

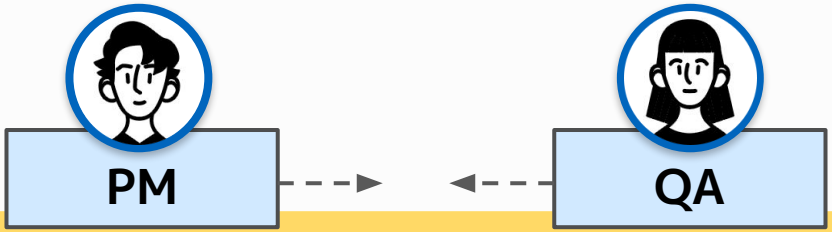


- 開発はタイムボックスで開発を実施し、QAはタイムボックスに沿わない
  （タイムボックスには沿わないものの進捗確認などは朝会でカンバンなどを用いて確認）

After	
起案タイミング	❏ 企画会議に参加
開発前	❏ 仕様書レビューへの参加・意見交換の実施
インプット	❏ PMと協業による仕様理解・検討
開発中	❏ QA業務以外にも、開発側との意見交換（新テクノロジーの共有） ❏ 実装が完了した部分から、開発側と連携し、適宜テスト開始
開発完了後	❏ 開発中に発生する利用品質をPM側へフィードバック
リリース後	❏ チーム全体でのKPT実施

アプローチ【２】～”QA”メンバーの越境～

緩やかな連携によって促される”QA”メンバーの越境



チームマネジメントにおいてQAメンバーの役割の越境に取り組む

“QA”というタスクは残すものの、障壁となっていた”QA”という明確な役割を取っ払う
各役割間の緩やかな連携によって、PM（企画者）とQAの距離が近くなる

- ❏ **タイムラインの立ち位置に固執せずに、プロダクトの仕様に早い段階から精通し、企画者とともにカスタマーにとって最適なプロダクトの品質**（製品品質だけでなく利用時品質に関しても）**を意識**

アプローチ【２】～既存のモデルとの棲み分け～

”QA”メンバーの越境と”W字モデル”との棲み分け

QAメンバーの越境

”テスト”というキーワードには固執せずに、”QA”という観点だけではなくユーザー目線でPM（企画者）などと協業し、要件定義や仕様作成・整理などを含めた取り組みに参加することで、パフォーマンスの最大化を目指す。QAメンバーが”テスト”という役割・観点から抜け出すことを期待するもの

W字モデルとは

”テスト”というキーワードをもとに、上流工程や早い段階から並走してタスクを進めるといった取り組みであり、企画者や開発者とは異なり、QAメンバーが”テスト”観点で並行稼働することで、短納期や手戻りの防止などを期待するもの

QA・テストという観点から少し離れた上流工程（仕様検討など）の役割も担うことで、プロダクトの仕様に早い段階から精通し、プロダクトの品質（**製品品質だけでなく利用時品質も**）を意識するようになる

“QA”という動きではなく、チーム全体のことを考えて取り組む

アプローチ【２】～QAの役割変化について～

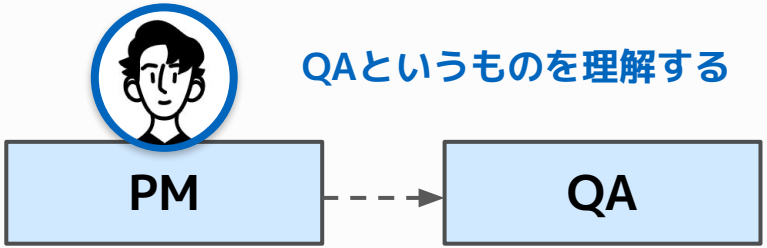
	Before	After
仕様・要件	❑ 仕様・要件はインプットがメイン	✓ 仕様に対して、ユーザ目線・機能目線で指摘
試験設計	❑ 決まった仕様の範囲内で設計	✓ 企画会議などで把握した内容を考慮し、設計
バグ・不具合	❑ 機能不具合のみ	✓ 機能不具合に加えて改善提案も
テクノロジー	❑ 項目作成時には考慮しない	✓ 上流工程への染み出し・開発メンバーとの連携によるテクノロジー理解の上での項目書作成

QAという明確な役割がプロダクト開発に取り組む中で障壁になる事も多く、QAというタスクだけは残し、チームマネジメントにおいてQAメンバーの役割の越境に取り組む

QAメンバーの役割が”テスト”に偏っていたため
プロジェクトへ関わりきれていない想いもあったが
”QA”の役割変化により、案件により携われるようになったことで想い入れも変化

アプローチ【３】～”QA”に関する理解を深める～

PM（チーム責任者）の”QA”理解



- チームの責任者自身がソフトウェア品質に対する興味をもつ
- ”QA”メンバーと積極的に意見交換を実施する
- ”QA”メンバーの立ち位置を理解する
  - 本来は一番プロダクトに触れている時間の長い”QAメンバー”の意見は重要
  - カスタマーが触れるプロダクトを誰よりも長い時間触れている

現状の“QA”を知ることで、”QA”メンバーの立ち位置を意識する

Agenda

- 背景・課題
- 目指すべき方向
- 取り組み・プロセス
- **実施結果**
- 結論と今後に向けて

結果：タイムボックスに依存しないことによる変化

Before

タイムボックスの固定による待ちの発生

- QAが開発状況を把握できない
- スケジュールの固定化による弊害

固定タイミングでのインプット

- 案件を把握するのが直前(PMインプット)
- スケジュール立てるのが直前

下流工程での役割の全う

- QAへの仕様インプットは企画後
- QAが携わるタイミングは開発完了後
- スケジュールの固定

After

✓ 企画会議へ参加
  -インプット前に案件把握
  -案件の温度感などを把握
  -先のスケジュール整理が可能
✓ 開発とPMの会話に参加
  -開発状況の把握
  -適宜スケジュールの更新が可能に
✓ 項目書の作成
✓ 項目書レビュー実施
✓ PM（企画者）と協業による仕様理解

QA待ち時間減少
マネジメント効率のUP

柔軟なタイミングでのタスク実施

緩やかな連携の結果による手戻りの減少
【平均手戻り指摘/案件】
Before:8.4 -> After:3.0

固定されたタイミングで動くのではなく、QAメンバーのプロダクト開発への関与の柔軟性を最大化
結果として開発効率(リリーススピードの最大化)へと繋がった

結果：QAメンバーの越境による変化

Before	After	
仕様認識のズレの発生による負 ❑ 仕様書からすべて項目書を設計する ❑ 仕様書の記載漏れなどで、手戻りが発生 ❑ 確認等に都度時間がかかる	✓ PMと同じ目線で議論し意見を共有 ✓ 結果として、認識齟齬がなくなり、効率よく改善が行われる ✓ 仕様検討漏れを検討段階から指摘 ✓ 試験項目書設計に必要な情報を事前に知れるため、設計がスムーズになる	**指摘（欠陥）項目の認識のズレの減少** 【平均仕様検討指摘/案件】 **Before:0 -> After:2.0**
不具合・手戻りの負 ❑ 役割の固定によってプロダクトに携わる部分が限定的 ❑ アジャイル化によるプロダクトへの「品質担保」の意識の変化	✓ PM・QA間の連携だけではなく、QA・開発間の連携を始めたことで、各役割の中でプロダクト品質に対する共通認識を持つ ✓ 各役割の中で認識合わせが行いやすくなったことにより、認識齟齬がなくなりミスが減った	**QAによる「品質担保」からチームとしての「品質担保へ」**
テクノロジーへの適応の遅れ ❑ QAに必要な最低限な部分のみ理解	✓ 開発メンバーとの連携により、プロジェクトにおける新しいアーキテクチャを把握 ✓ クラウド技術などの新しいテクノロジーに触れることでのテスト観点の向上	**技術力・知識の向上 テスト観点のレベルUP**

QAという役割を固定し下流でのテスト実施だけではなく、全体を通してプロジェクトに携わることで負の解消へ

プロダクト品質の向上、チームパフォーマンスの最大化に繋がった

2018年度の不具合検出率「平均 6.0%」/ 2019年度の不具合検出率「平均 2.3%」

Agenda

- 背景・課題
- 目指すべき方向
- 取り組み・プロセス
- 実施結果
- **結論と今後に向けて**

結論：QA課題解決によるパフォーマンス最大化

”QA”のという意識ではなくワンチームとしての意識へ

**明確な立ち位置（役割）がなくなることでQA内で問題解決に固執せずに
チームで取り組めるようになり、パフォーマンスが最大化**

”QA”の立ち位置（役割）の再定義と”QA”の越境による変化

- **プロダクト開発効率の最大化**
  - QAメンバーのプロダクトへの関わりの柔軟性が最大化されたことで、各タイミング・役割で相互のコミュニケーションの増加に繋がり、仕様認識のズレが最小に抑えらた。その結果として、手戻りの減少や不要なものを作り出さないことに繋がった。
- **負の改善、チームパフォーマンスの最大化**
  - QAメンバーの越境によってプロダクト全体が見えるようになり、プロダクト仕様や各プロセスに対するQAメンバーの理解が深まることで、”QA”の意味が明確になった。結果、高速にPDCAを回す中でも”QA”の正確性が向上し、プロダクトの品質向上に貢献するとともにチームパフォーマンスが向上した。

”QA”=”テスト”という役割だけでは、プロジェクトへ関われきれていない想いもあったが
”QA”の越境により職域も増え、案件により携われるようになったことで想い入れも変化
結果として、QAメンバーのモチベーションも改善

我々のアプローチへの期待と今後について

”QA”の越境による変化で見えた効果と残課題へのチャレンジ

今回の一例であるタイムボックスという概念の再整理は、チームのパフォーマンス改善に至ったが、QAメンバー越境と役割の変化により、求められるスキルやタスクの多様化

- **QAメンバーの変化の必要性と負担増加の可能性**
  - 新たな取り組みに関しては、これまで固定化されたタスクだったものに対して、”QAメンバーの意識の変化が必要となること”や”役割増加による負荷の増加”などデメリットやリスクの増加に繋がる可能性がある
  - QAメンバーの越境だけが注目されることで、QAメンバーへの依存度が高まってくるが、チームにおける課題はQAメンバーの改善努力だけで解決できるものは少なく、引き続きチームにおける課題と捉え続ける必要性がある

”QA”メンバーの越境だけに頼らないチームに向けて

PM⇔QAメンバー間の役割の越境だけではなくチーム全体の相互越境へ

“QAメンバーだけの越境だけではなく、開発やPMが相互の越境によって、適切なタイミングで適切なメンバーが役割を担い、QAだけの負担の増加に偏らない環境を作ることで、継続してパフォーマンスを最大化できると感じている