QAからDevOpsへの挑戦
～QCDはトレードオフじゃない～

２０１８年３月8日
株式会社ＳＨＩＦＴ
山下 裕晃／脇坂 雅幸

DevOpsのイメージは？

- 開発と運用が連携を深める？
- 色んな技術を導入して自動化する？
- ただのバズワード？

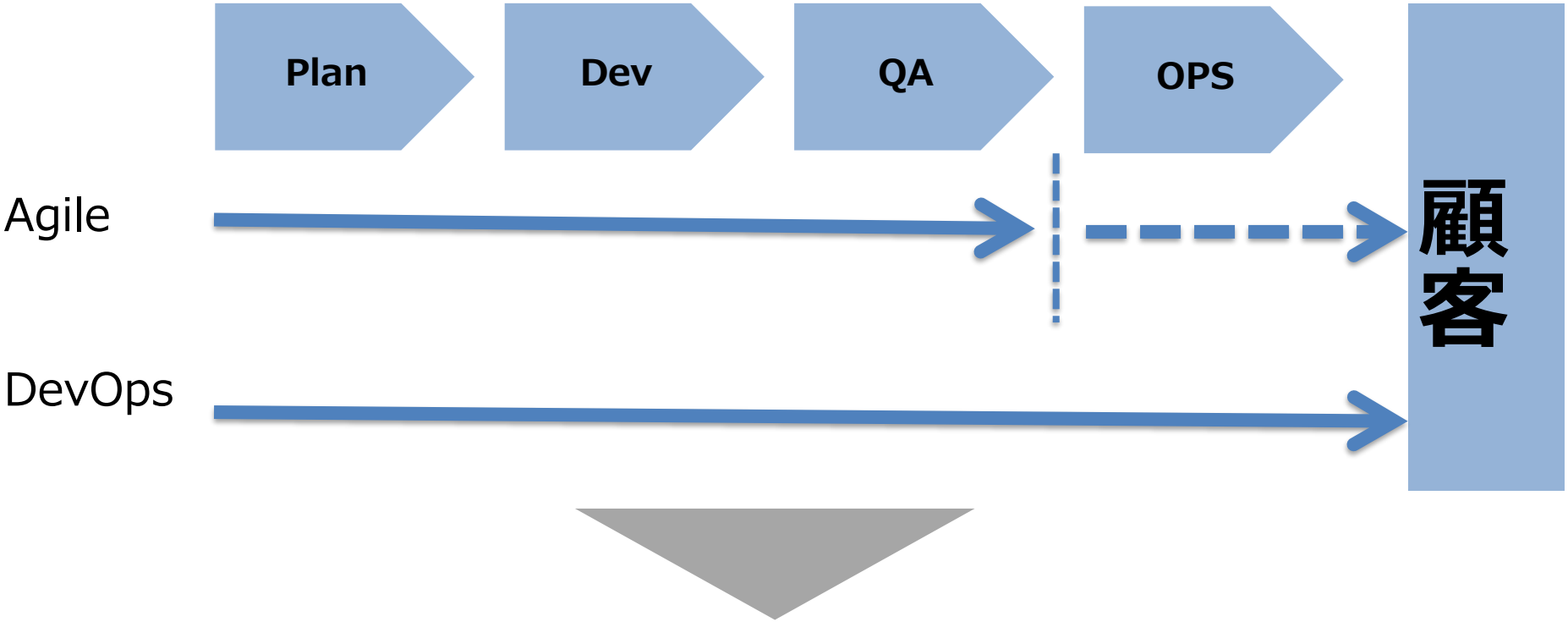
顧客に届く価値を最大化させるための

全組織的な改善活動及びその文化

企画したイメージ通りのものを
早く顧客に届ける

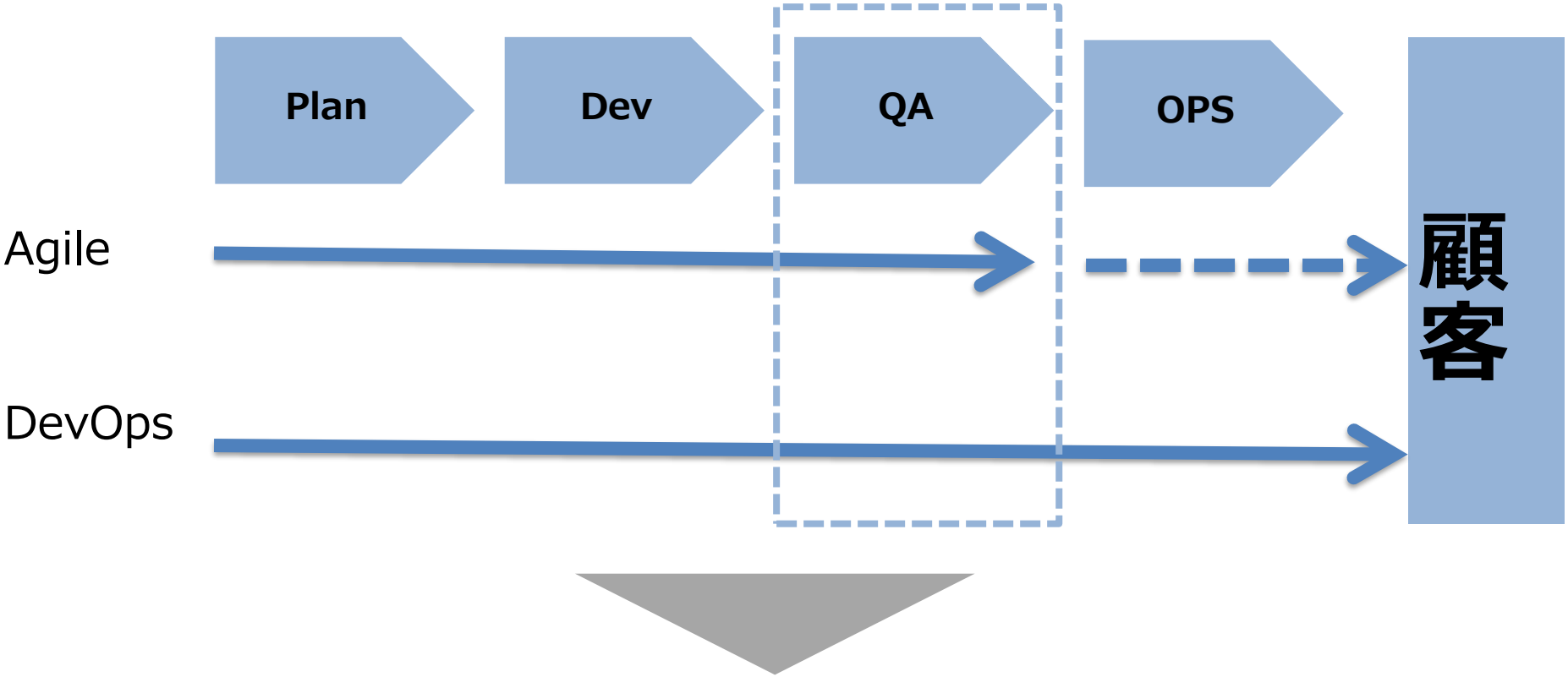
DevOpsとAgileの違いは？

DevOpsはAgileよりも実価値に踏み込んでいる



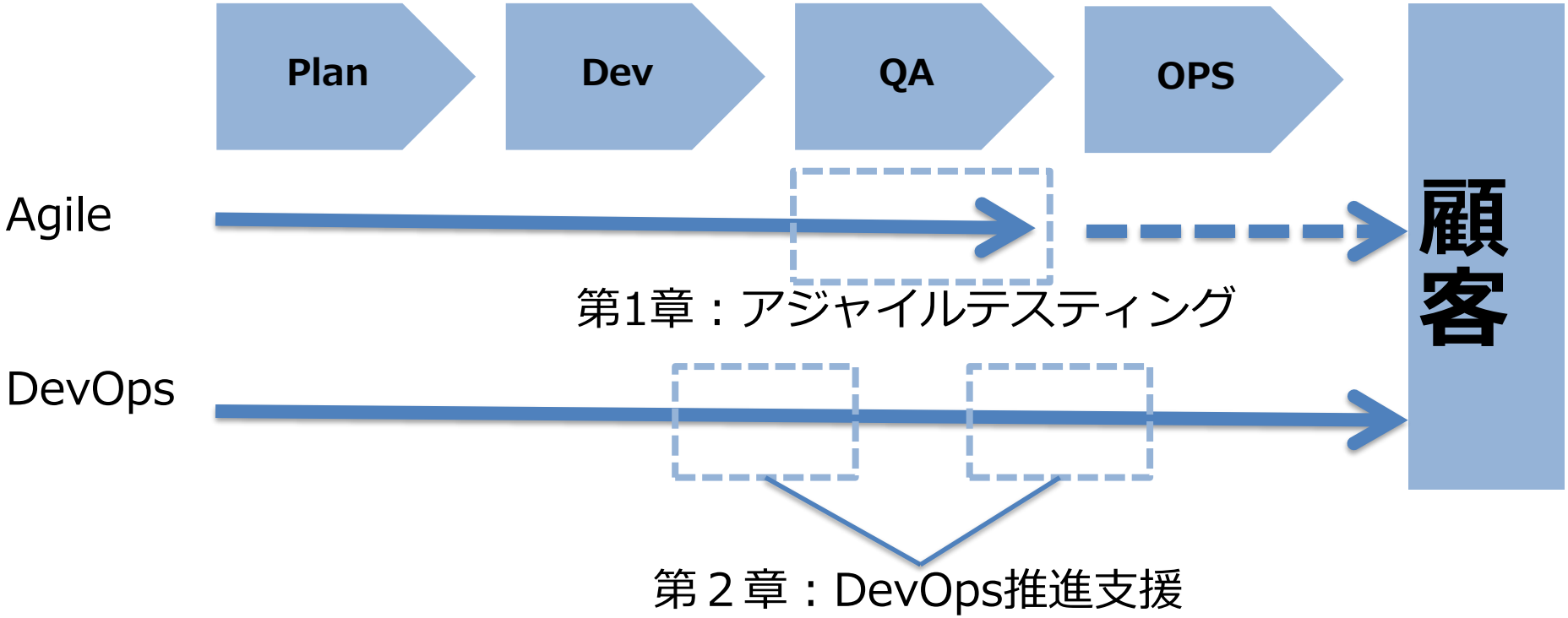
顧客に届いて初めて価値が生まれる

DevOpsの要は実はQA



QAの良し悪しが前後の工程に大きな影響を与える

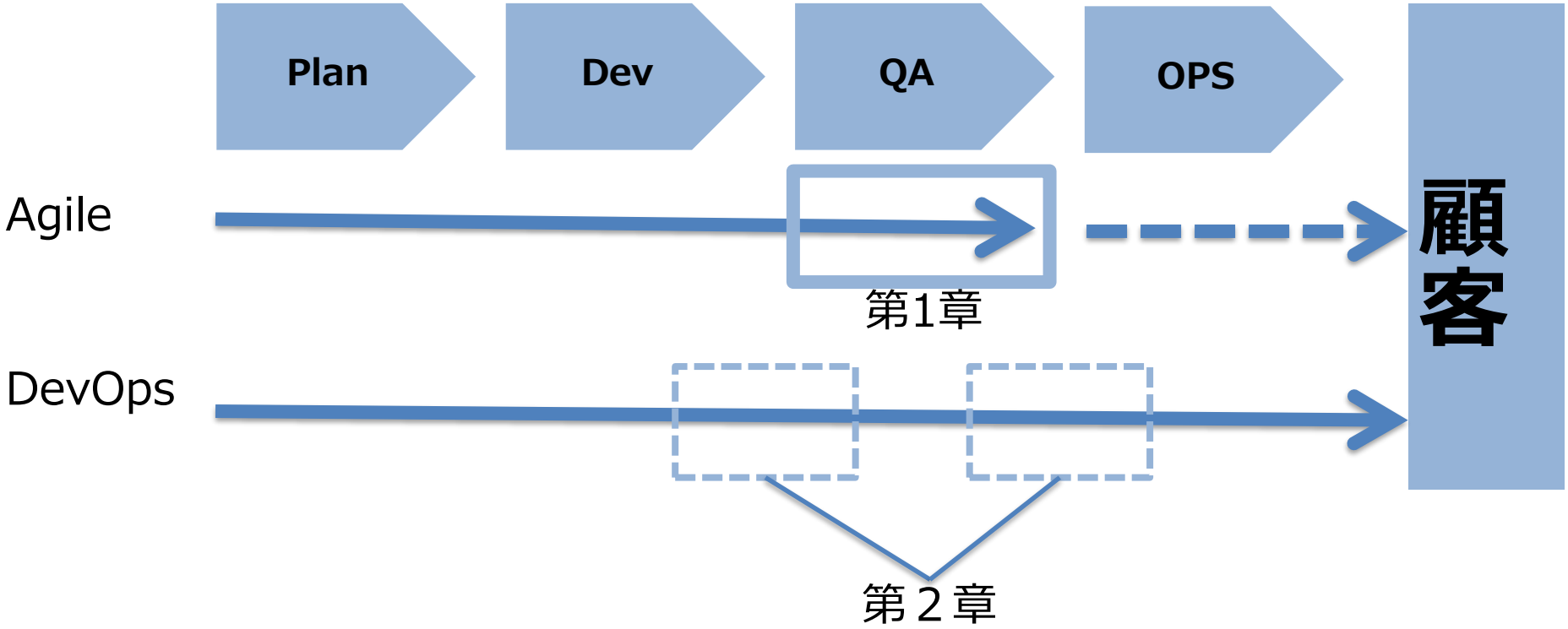
QAを軸にした改善活動を２章に分けて説明



第1章
アジャイルテスティング
~品質とスピードを両立するための自動テスト~

株式会社ＳＨＩＦＴ
脇坂 雅幸

QAとして、DevとOpsを橋渡し



目次

1. 自己紹介
2. はじめに
3. 自動テストの課題
4. 自動テストの課題解決
5. まとめ

1. 自己紹介

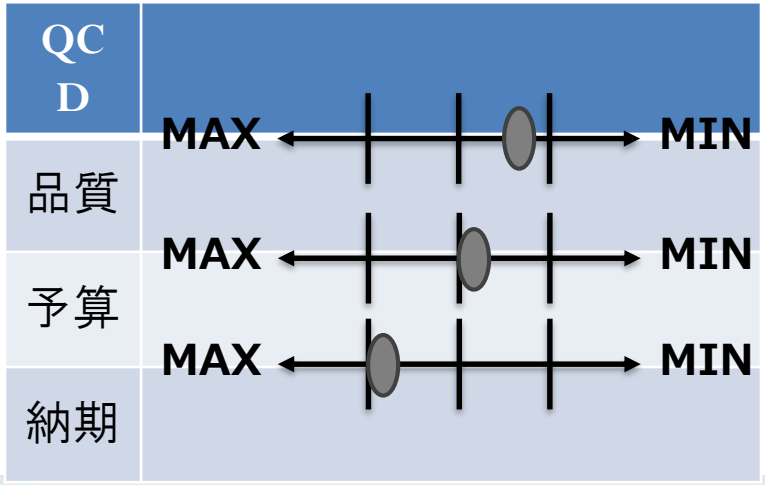
テスト自動化エンジニアからアジャイルテスターへ

- 名前：脇坂 雅幸
- 所属：株式会社SHIFT
  サービスプロモーション部
  技術推進室
- 経歴：
  - 開発会社でiPhoneアプリの開発に従事後、SHIFTに入社
  - 入社後はテスト自動化エンジニアとして、SIerの案件に参画
  - GUIテスト自動化により、テスト工程のリードタイム短縮に貢献
  - 現在は、スクラムの現場にQAメンバーとして参画している

2. はじめに

BtoCのチャットボットを開発するチーム

- プロセス: スプリント1週間のスクラム
- スクラムを採用した理由
  - 状況: β版から商用への過渡期
  - 方針: 続々と上がる課題を対応し、素早くリリースしたい
- よって、スピード重視の開発



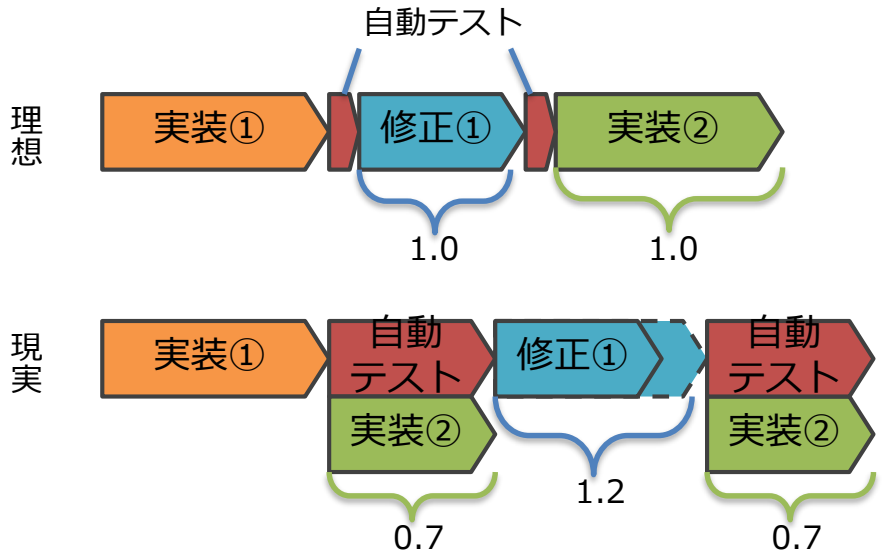
リリースが度々延期になる

- アジャイルに不可欠な多くのプラクティスを採用
  - テスト自動化、CI/CDパイプライン、IaCなど
  - ペアプロ、モブプロによるコーディング
  - マイクロサービスアーキテクチャで構成されたプロダクト
- にも関わらず、平均して1回/月のリリース
  - W/Fでも実現可能な頻度

3. 自動テストの課題

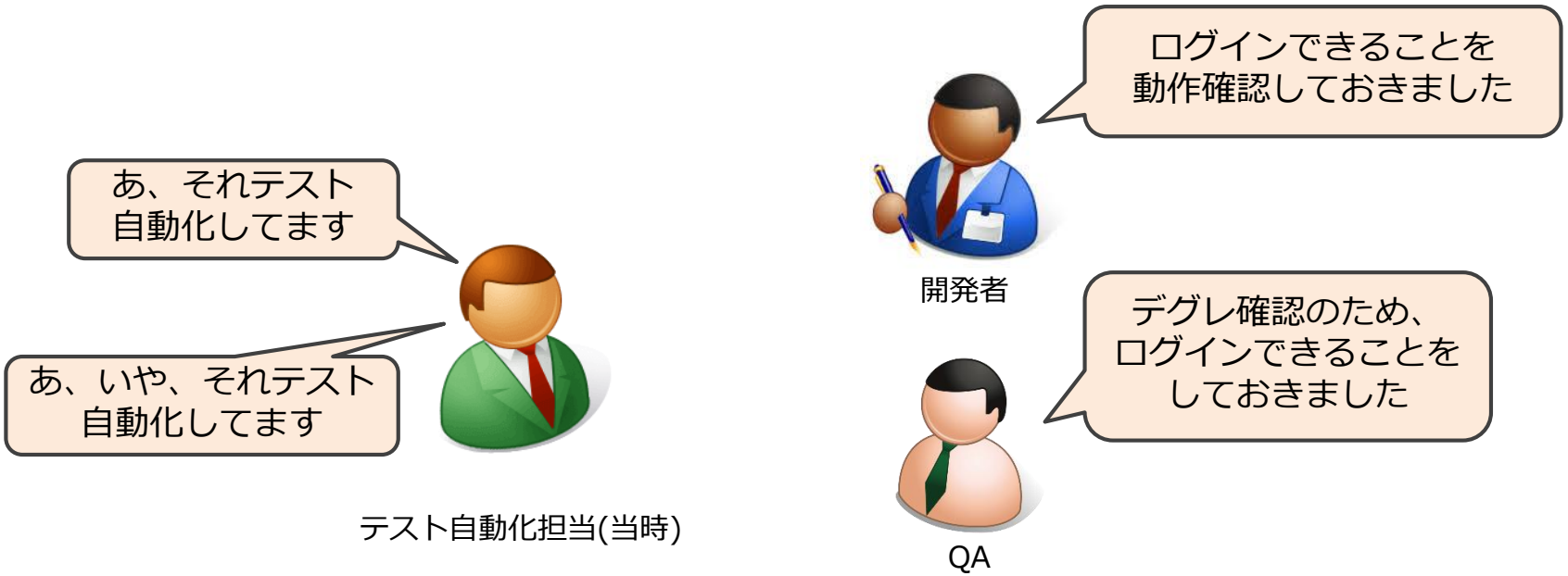
スローテスト問題

- 自動テスト(統合レベル)実行に1時間かかっていた
- 開発者も待ちきれず、別のタスクを始めてしまう
  - スイッチングコストが発生し、リードタイムに影響が出てしまう



自動テストのブラックボックス化

- 自動テストの内容が担当者以外把握できていない
  1. 自動テストで実施しているケースを
     開発者も同じように動作確認で実施していた
  2. 別のテスト担当者も同じ内容のテストを実施していた



実は品質も気にしていた

- 品質への懸念から、リリースに慎重になっていた
- スピードも大切だが、それ以上に品質も必要
- この心の枷を取っ払う施策が必要不可欠だった

実は最も高かった

QCD	当初の目標	現場の感覚
品質	MAX ← → MIN	MAX ← → MIN
予算	MAX ← → MIN	MAX ← → MIN
納期	MAX ← → MIN	MAX ← → MIN

結局、スピードと品質は
トレードオフ？

いや、トレードオフではない

4. 自動テストの課題解決

まずは目標時間の設定

60分 → **10分**

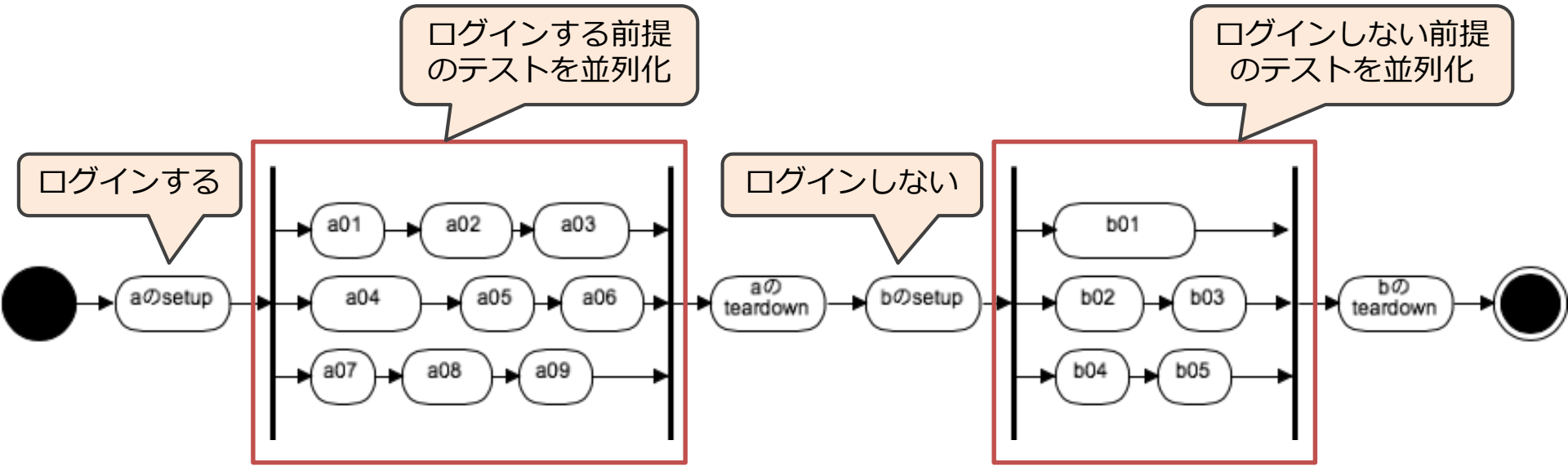
自動テストの並列化

- 並列化のアプローチ
  - テスティングフレームワークの機能を利用

- 並列化の際の課題
  1. システムの制約
     - 同時ログインが許されないシステムだった
  2. 異なる事前条件を持つテストケースの混在
     - ログインしなければならないテストと
       ログインしてはならないテストが混在していた

自動テストの並列化の実現

- 並列テスト用のスイートを作成
  - ログインセッションをテストケース間で共有
  - テストの事前条件ごとにテスト並列化



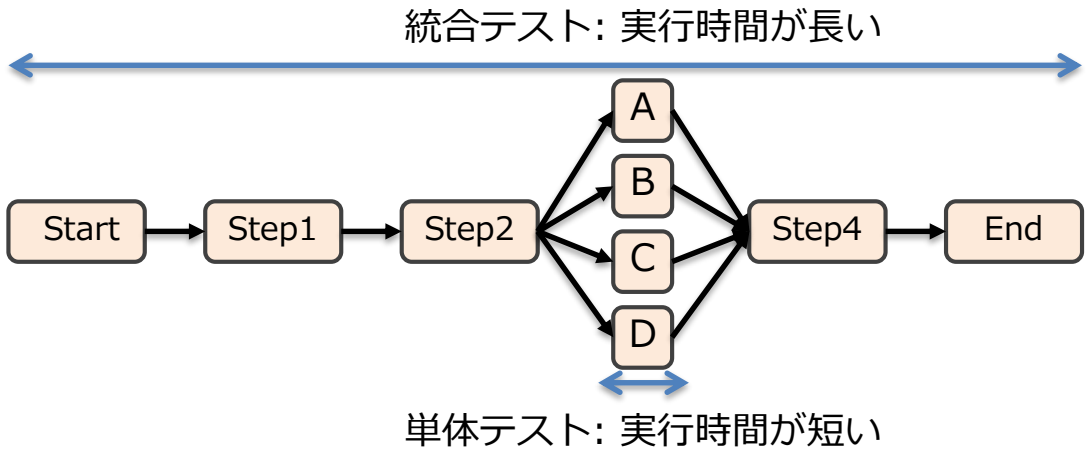
自動テストの削減

- そもそも何故自動テストの数が膨らんでしまうか
  1. テストパターンの網羅を追求してしまう
  2. 問題の局所化を意識してしまう

- 削減方針
  1. テストパターンの削減
  2. 重複テストコードの削除

テストパターンの削減

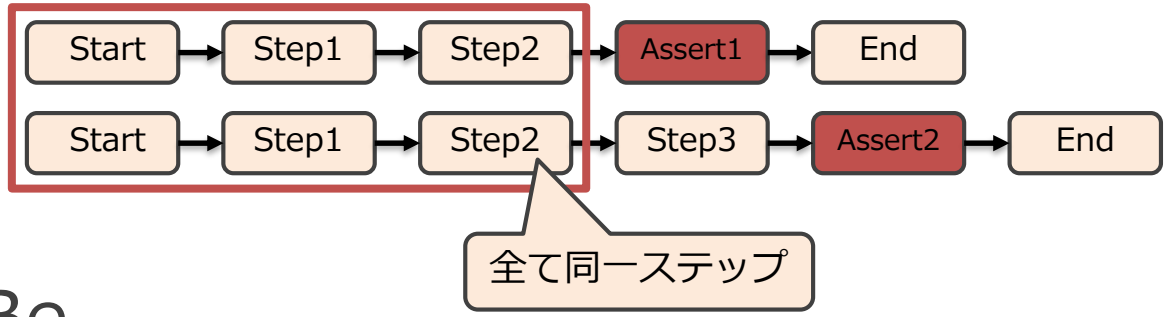
- AsIs
  - 統合テストでテストパターンの網羅を追求してしまう



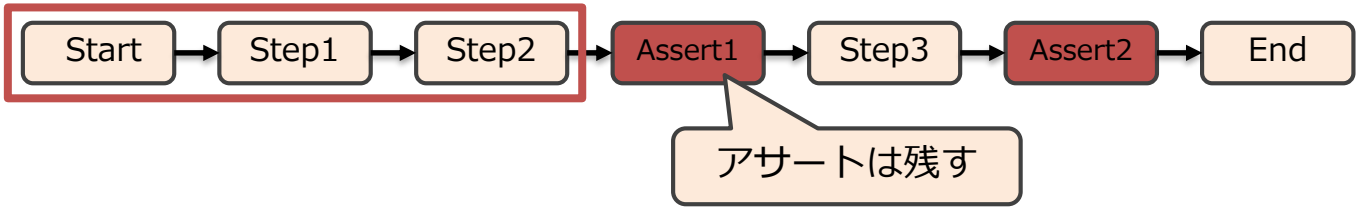
- ToBe
  - チームと合意の上、統合テストはパターン数を抑える
  - パターン網羅が必要なテストはなるべく単体テストに落とし込む

重複テストコードの削除

- AsIs
  - テストケースの意図の明確化を追求してしまう



- ToBe
  - 共通のステップを持つ自動テストを1本化する



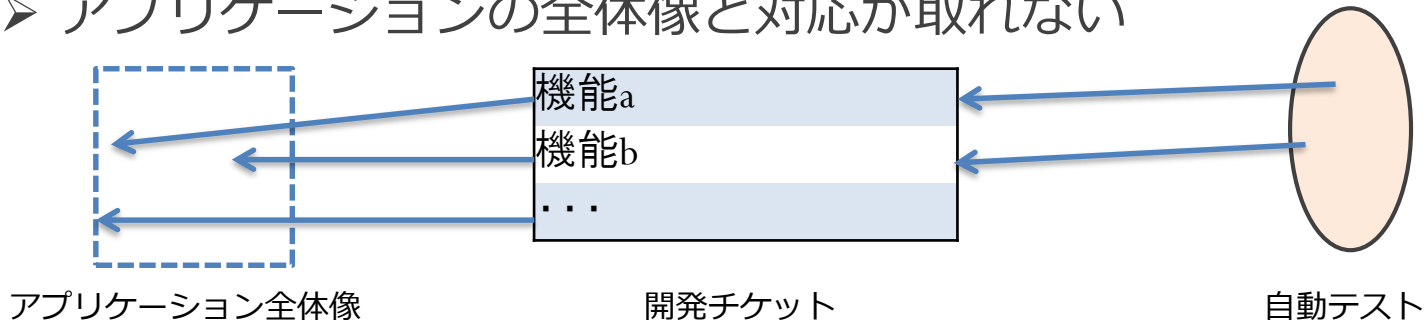
効果

- 達成
  - 開発者はタスクをきっちり終わらせてから、
    別のタスクへ移るようになり、マルチタスクをしなくなった

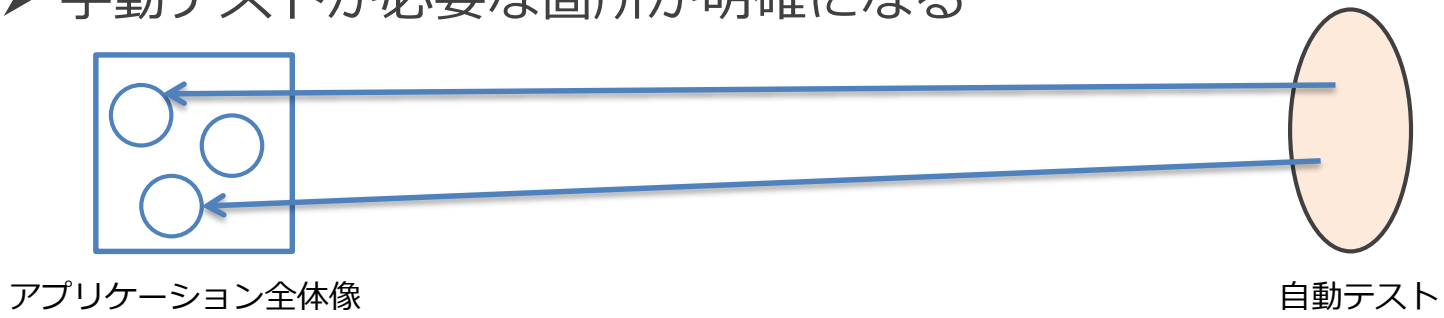
60分 → **8分**

ブラックボックス化の解消

- AsIs: 開発チケットと紐付ける
  - アプリケーションの全体像と対応が取れない

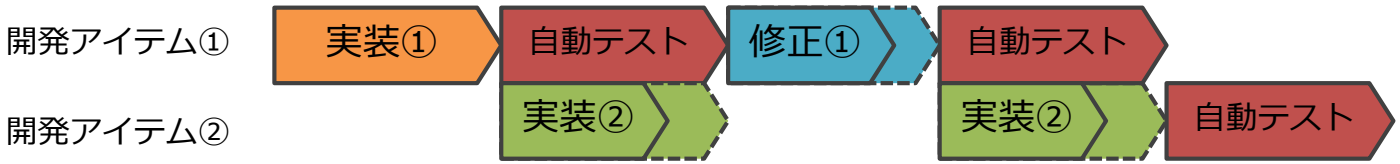


- ToBe: 製品仕様と紐付ける
  - 手動テストが必要な箇所が明確になる



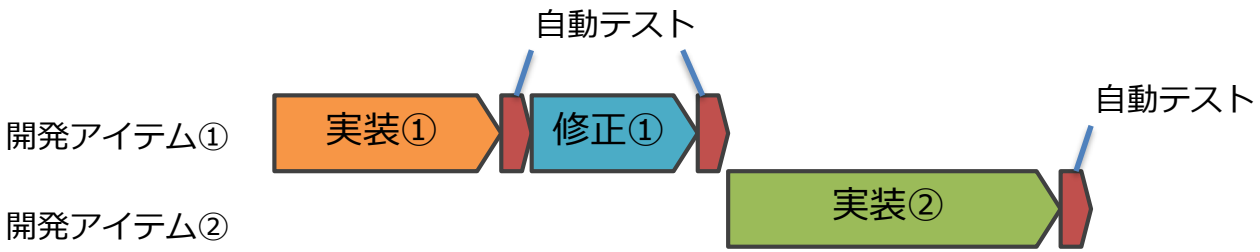
1リリース/2週を達成

Before



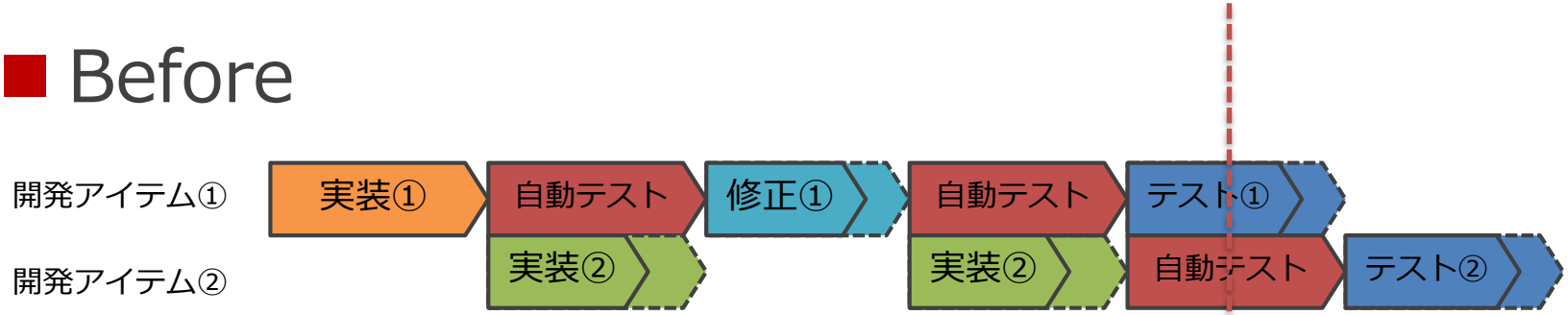
After

1.スイッチングコストの削減



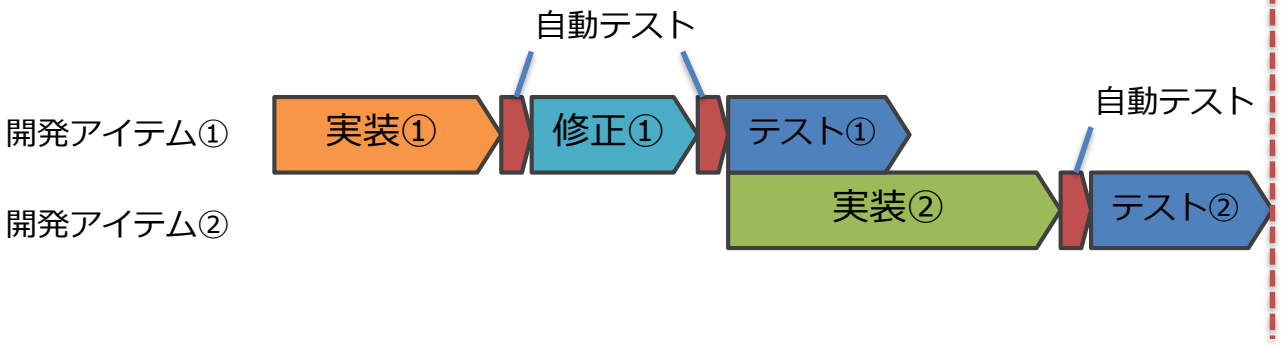
1リリース/2週を達成

Before



After

1.スイッチングコストの削減
2.回帰テスト実行工数の削減
3.動作確認工数の削減



5.（第1章）まとめ

QAとして、QとDの両立を実現

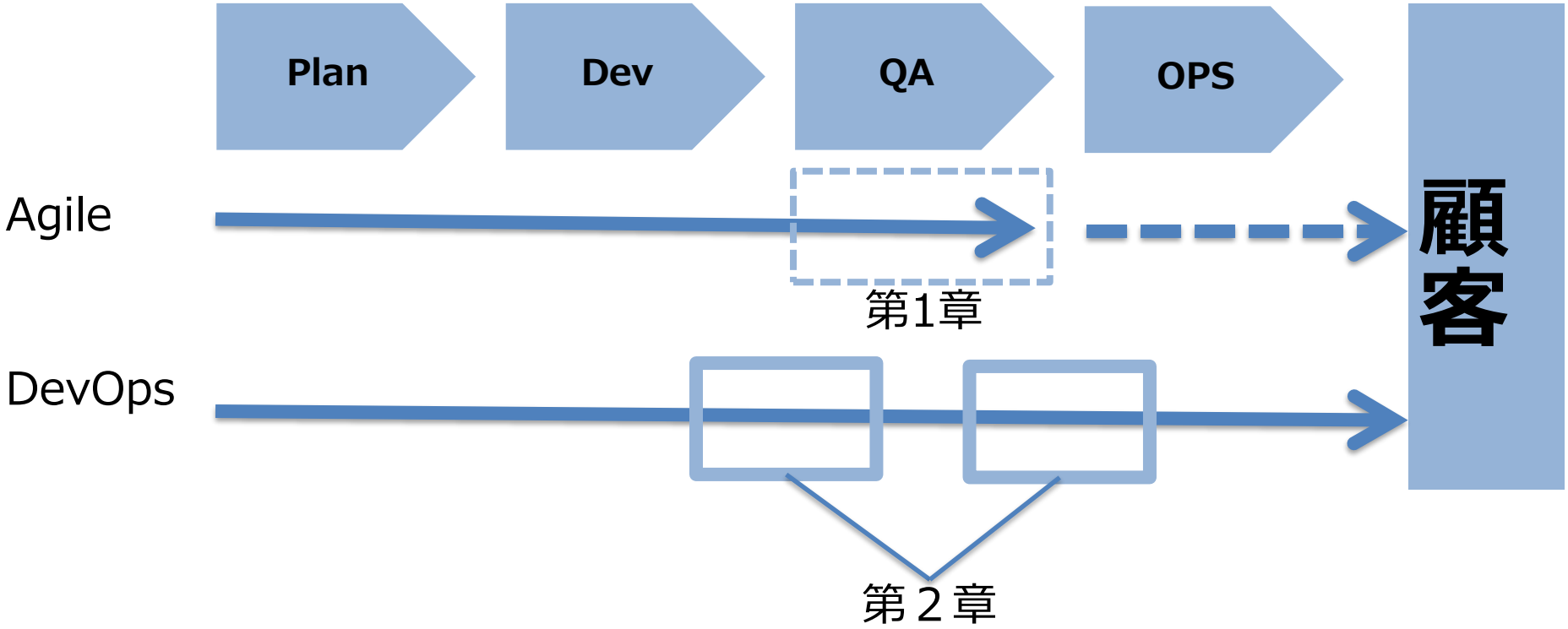
- DevとOpsの橋渡しに
  不可欠なテスト自動化の改善に着手

- 自動テスト実行時間を60分 → **8分** に短縮
- **1リリース/2週**を実現
- 回帰テスト実行工数を約**28%**削減

第2章
DevOps推進支援
~QAからの越境~

株式会社ＳＨＩＦＴ
山下 裕晃

DevとQA,QAとOpsとの連携を促進



目次

1. 自己紹介

自動テストからDevOpsの世界に

- 名前：山下 裕晃
- 所属：株式会社SHIFT
  サービスプロモーション部 技術推進室
- 経歴：
  - 自動テストエンジニアからキャリアスタート
  - テストの実行管理・設計業務も挟みながら
  - 現在は、DevOps推進支援業務に従事

2. はじめに

こんな状態になっていませんか？

- 改善したいことは山積み
- どこから着手すればいいか分からない
- 周りに理解がある人もいない

同じ苦労をする人を減らしたい

- 芋蔓式に出てくる課題
- 効果が上がらない施策
- 中々広がらない改善の輪

3. 改善事例

お客様は着実な成長を続ける旅行会社

- 二年で売り上げは1.5倍以上、開発人員も約２倍に
  - オフショアが5分の3を占める
  - オフショアとは対等な関係で、現地メンバーがリード
- 環境や開発手法はレガシー
  - 言語もFWもOSも２世代前（全てサポート切れ）
  - 少人数で開発していた時のやり方のまま成長
- 開発でも歪みが発生し対応が必要
  - 多くの手動オペレーションによって負荷が増大
  - 品質悪化や手戻りの発生など

最初の依頼は自動テストで本番障害の防止

デグレで本番障害

＋

リグレッションテスト未実施

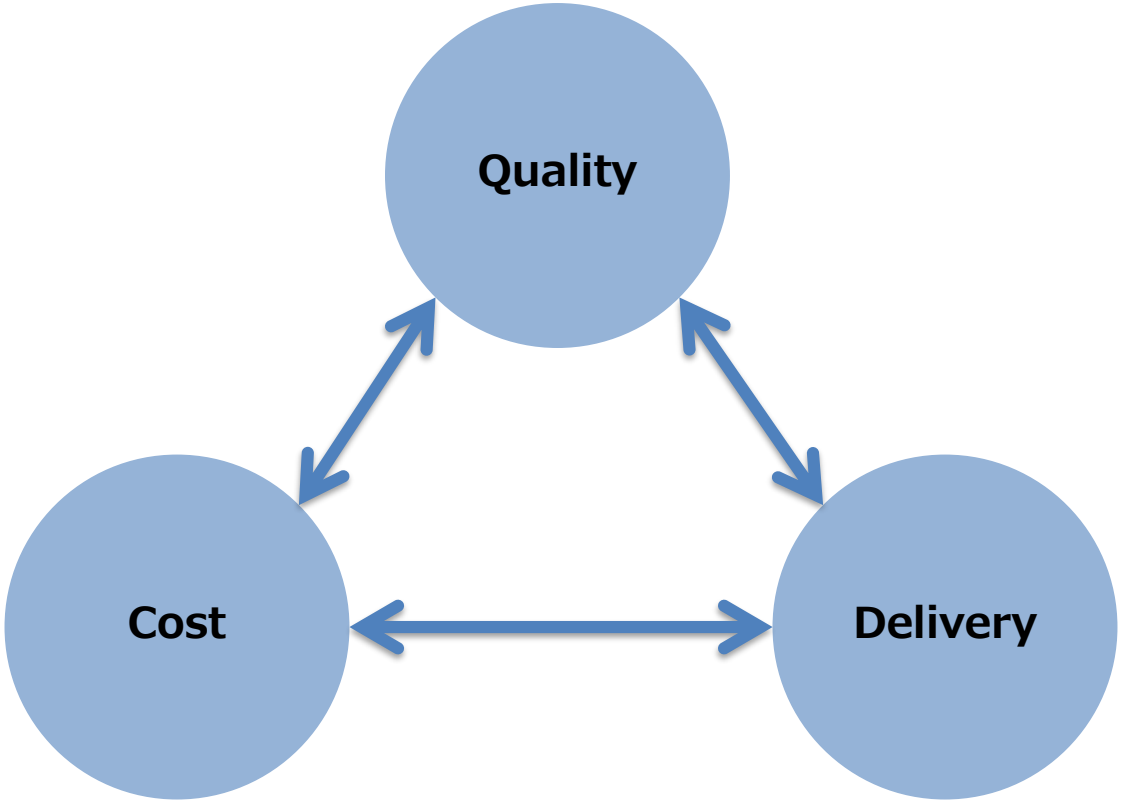
＝

重要機能のテストを自動化

障害は幾つか検知したものの

引き続き問題が発生

QCDの問題が絡み合い悪循環に



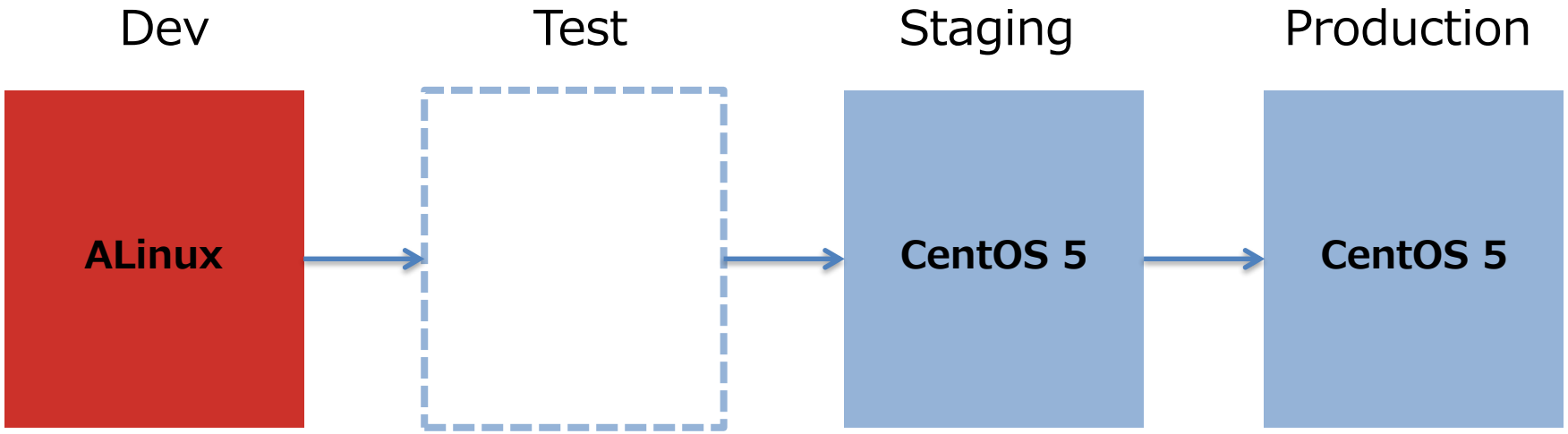
Qualityの課題

1) 環境差異による障害

2) 変化し続けるテスト対象

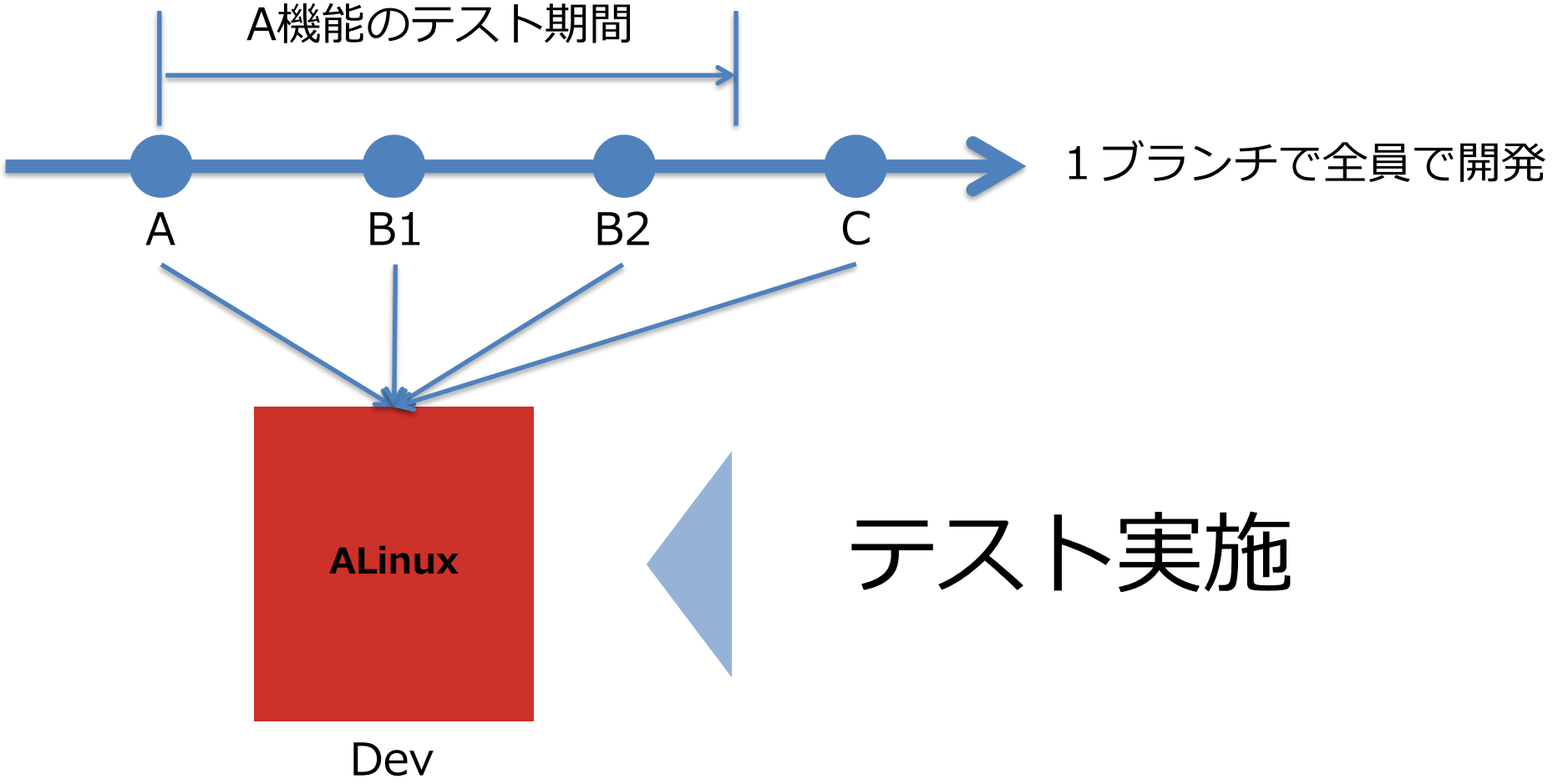
3) リリース作業時の不具合混入

環境差異により障害が発生

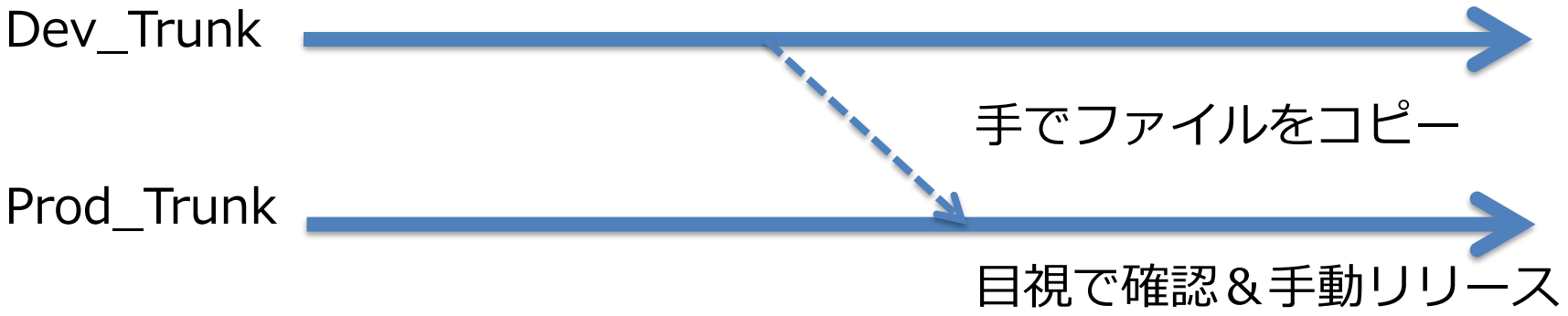


- テスト環境がなく、開発環境でテストを実施
- 開発環境は**本番と違う構成**

変化し続けるテスト対象



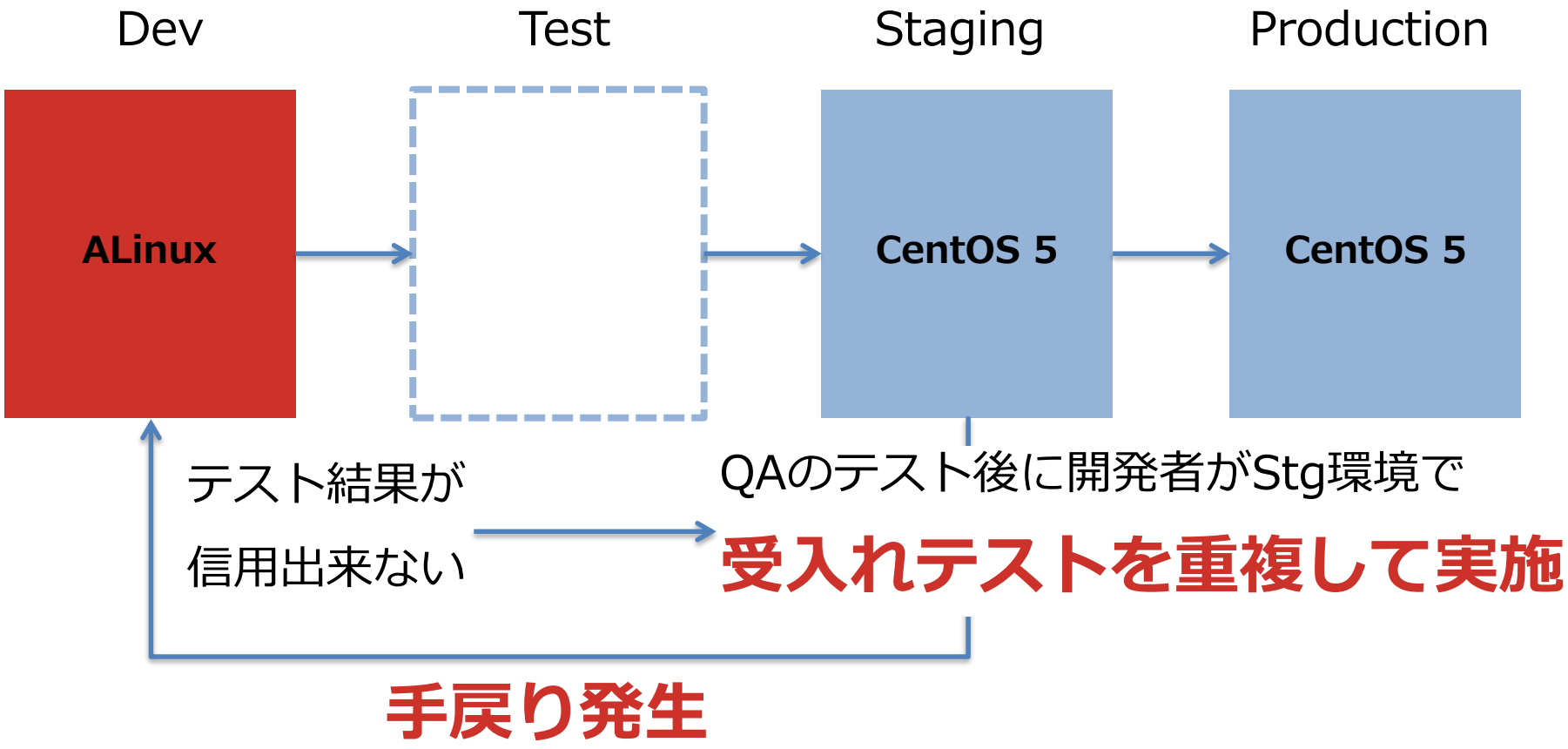
手動作業時に不具合が混入



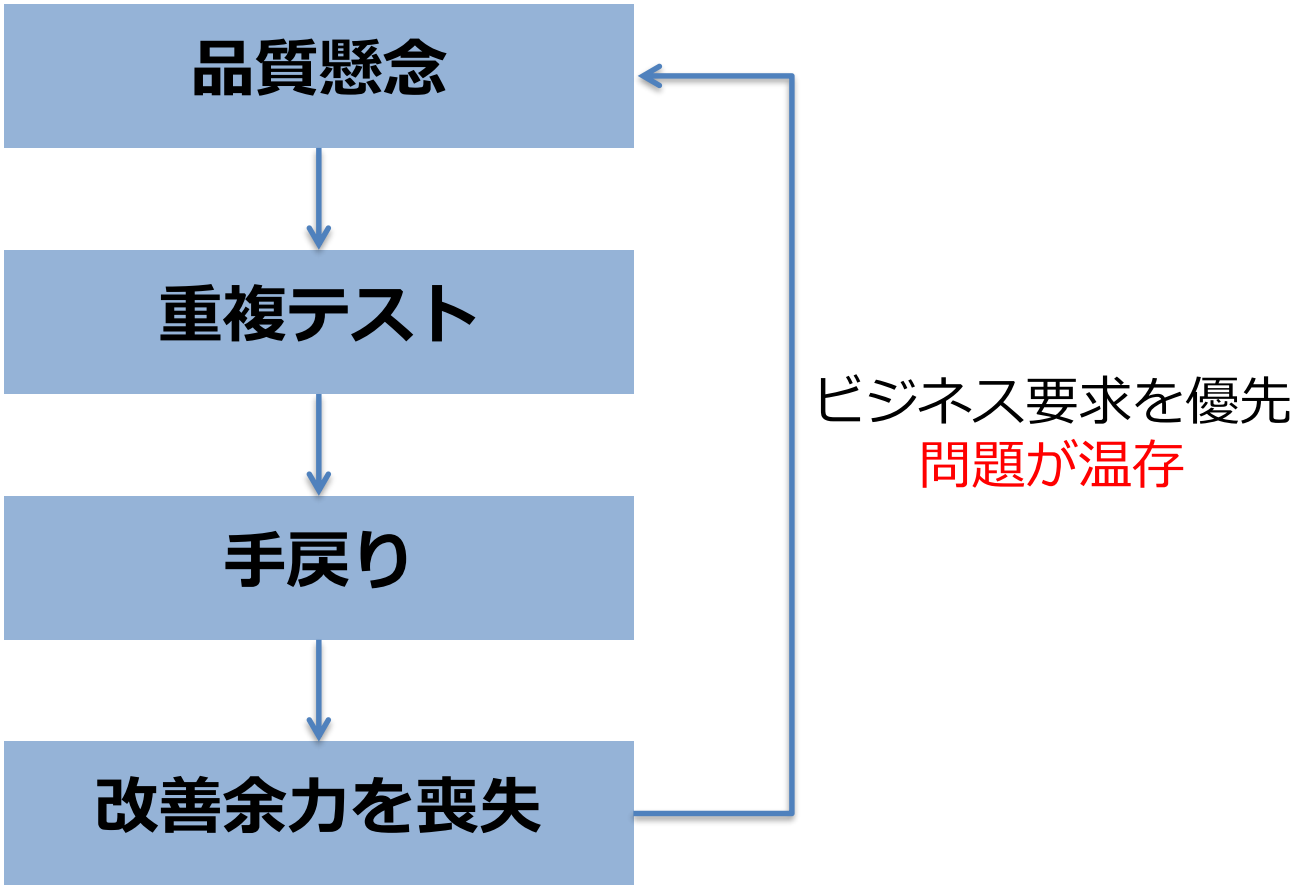
1. 手動でファイルコピーによるマージ作業
2. マージ後にミスがないか目視で確認
3. 最後に手作業でリリース

CostとDeliveryの問題が波及

コスト(C)と時間(D)を不要に消費



悪循環になり非効率な状態が継続



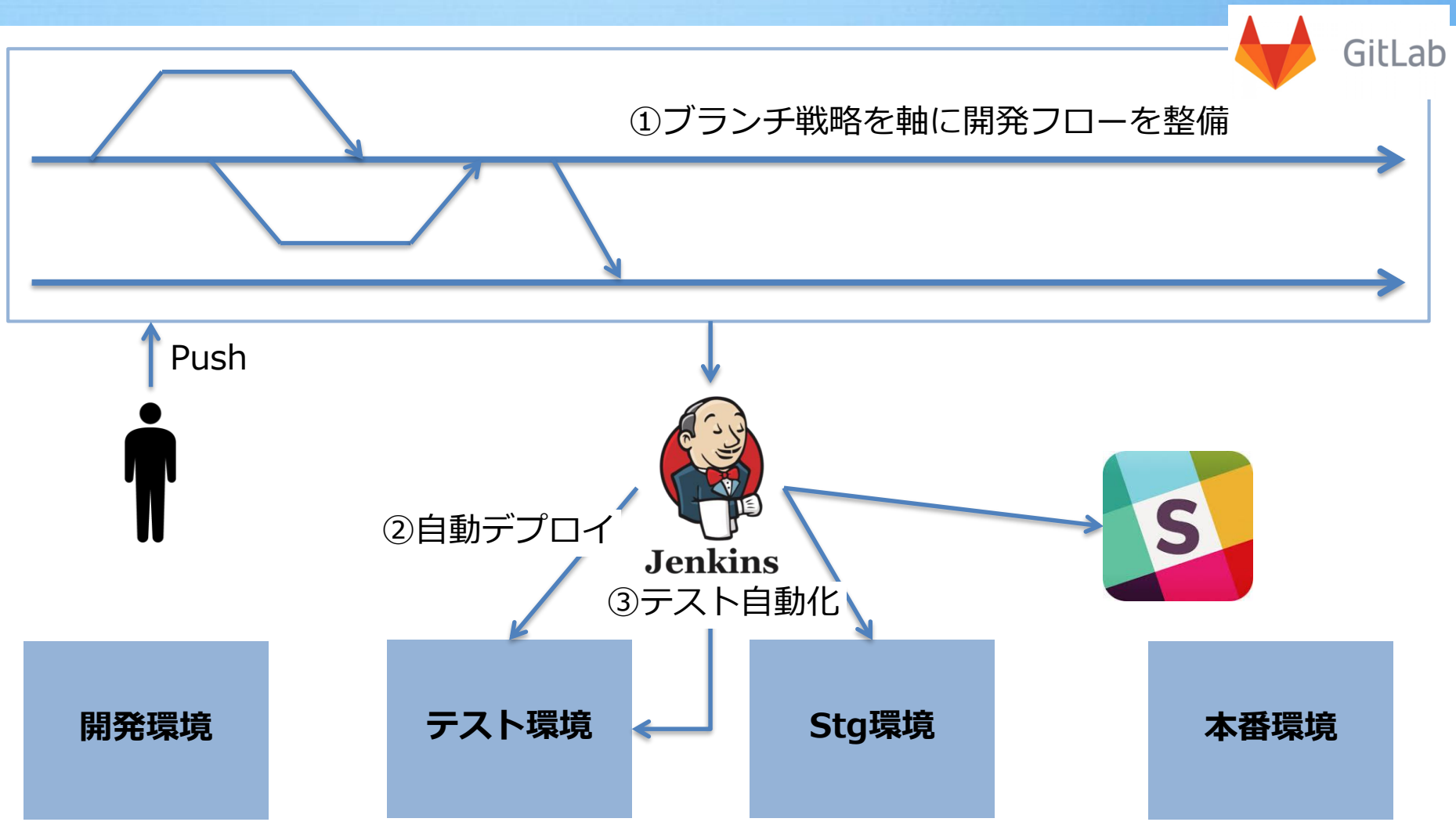
解決策

1) 開発フローを整える

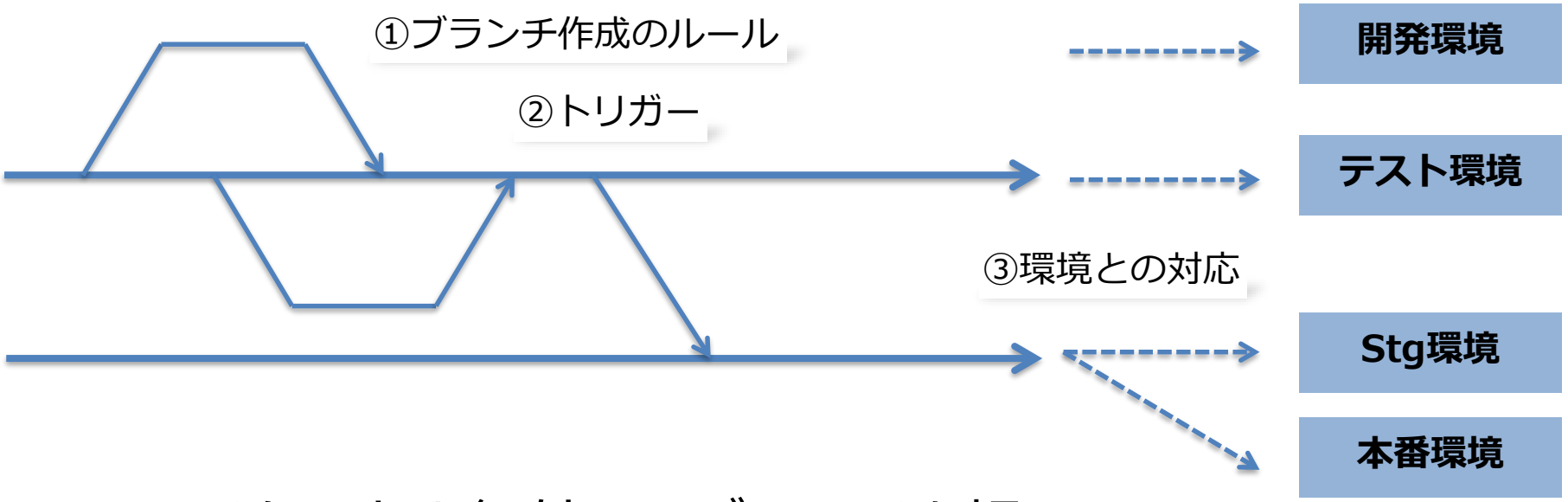
2) 環境を整える

3) 自動テストの導入

実施内容の全体像

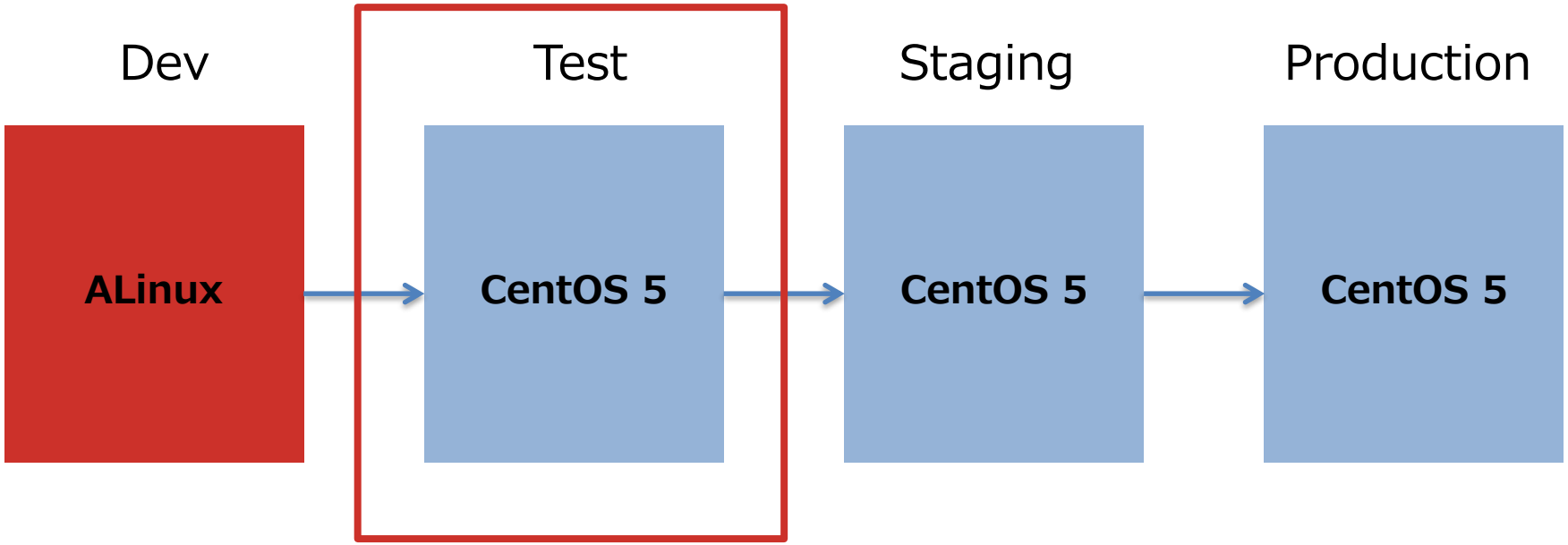

GitLab
①ブランチ戦略を軸に開発フローを整備
Push
②自動デプロイ
Jenkins
③テスト自動化
開発環境
テスト環境
Stg環境
本番環境

ブランチ戦略を軸にフローを策定



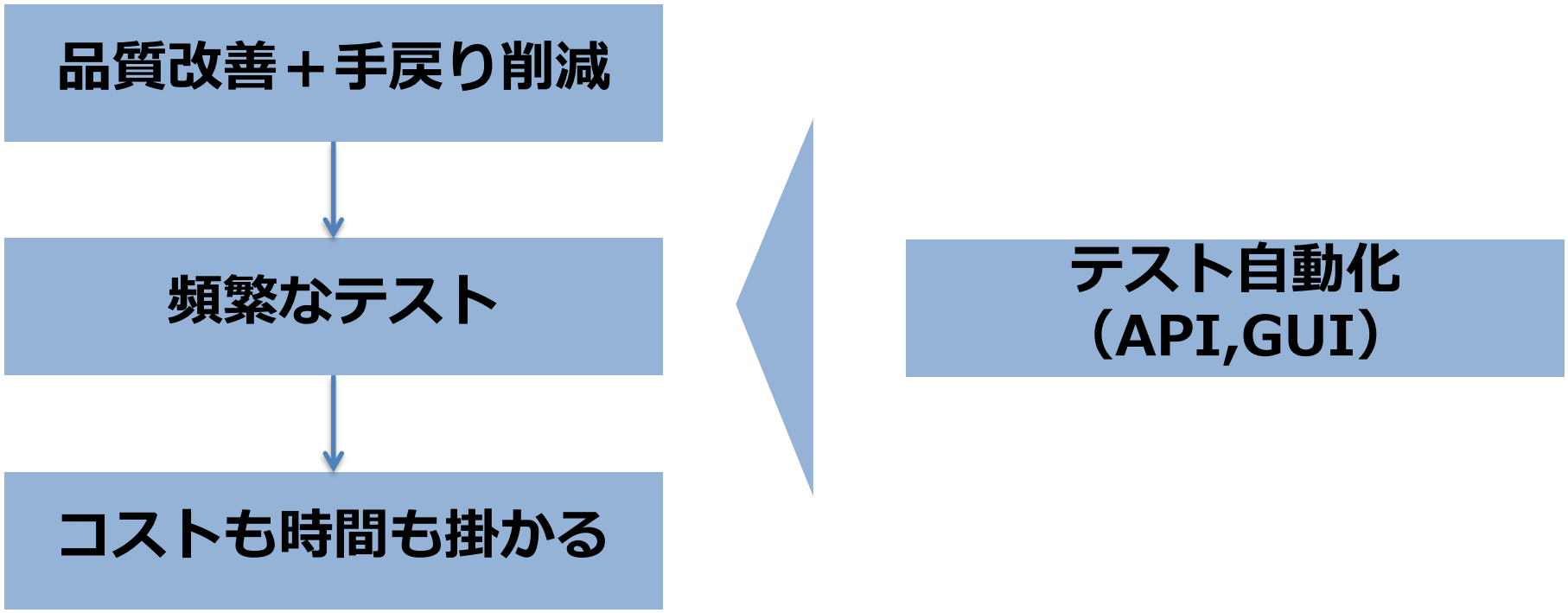
1. チケットと紐付けてブランチを切る
2. 自動デプロイ＋自動テストのトリガー決め
3. 環境との対応を決定

最新かつ差異がない環境を用意



1. 本番相当のテスト環境を作成
2. トリガーと連動した自動デプロイを実装し
   常にテストに相応しい環境を準備

自動テストでQCDのバランスを取る



技術的課題は多数あった

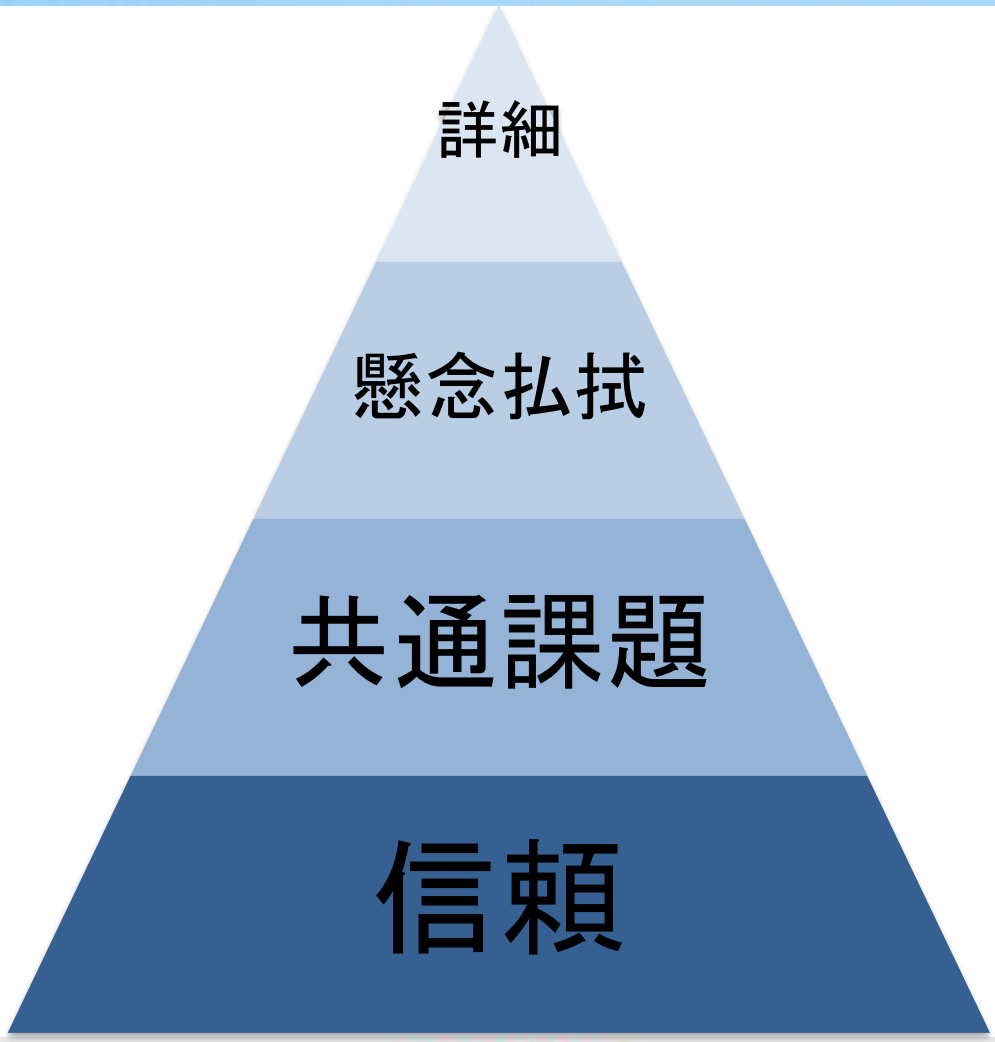
- どのブランチ戦略をベースに？
- それは本当にチームに合うの？
- どんなツールを使って、どう連携？
- 安定して運用するためには？

最大の壁は浸透

実際にお客様も苦戦

- 一気に改善方針を作って共有 => 猛反発
- そもそも課題感が違う => 話が平行線に
  - マージ作業大変だよね？品質のリスクあるよね？
  - 別に困っていない。そう思わない
- 途中で妥協する => 問題が拗れる
  - 暫くすると課題が出る
  - その時になって直すと言うと今更なんだ！！となる

信頼構築と課題感の共有が肝



- ×：詳細から詰める
- ○：土台から築く

- ×：懸念事項を否定
- ○：不安の源泉を聞く

- ×：私にとって必要
- ○：我々にとって必要

- ×：自分の正当性を主張
- ○：敬意と尊重

効果

Q：バグの早期発見

C：重複テストの削減 + 必要工数の抑制

D：手戻りの削減

3ヶ月で１サービスのみで

47件19種類の障害を事前検知

手戻り削減で

開発工数の **1 割**を削減

年間

120人日の重複テスト工数を削減

API：月間**412.8人日**抑制

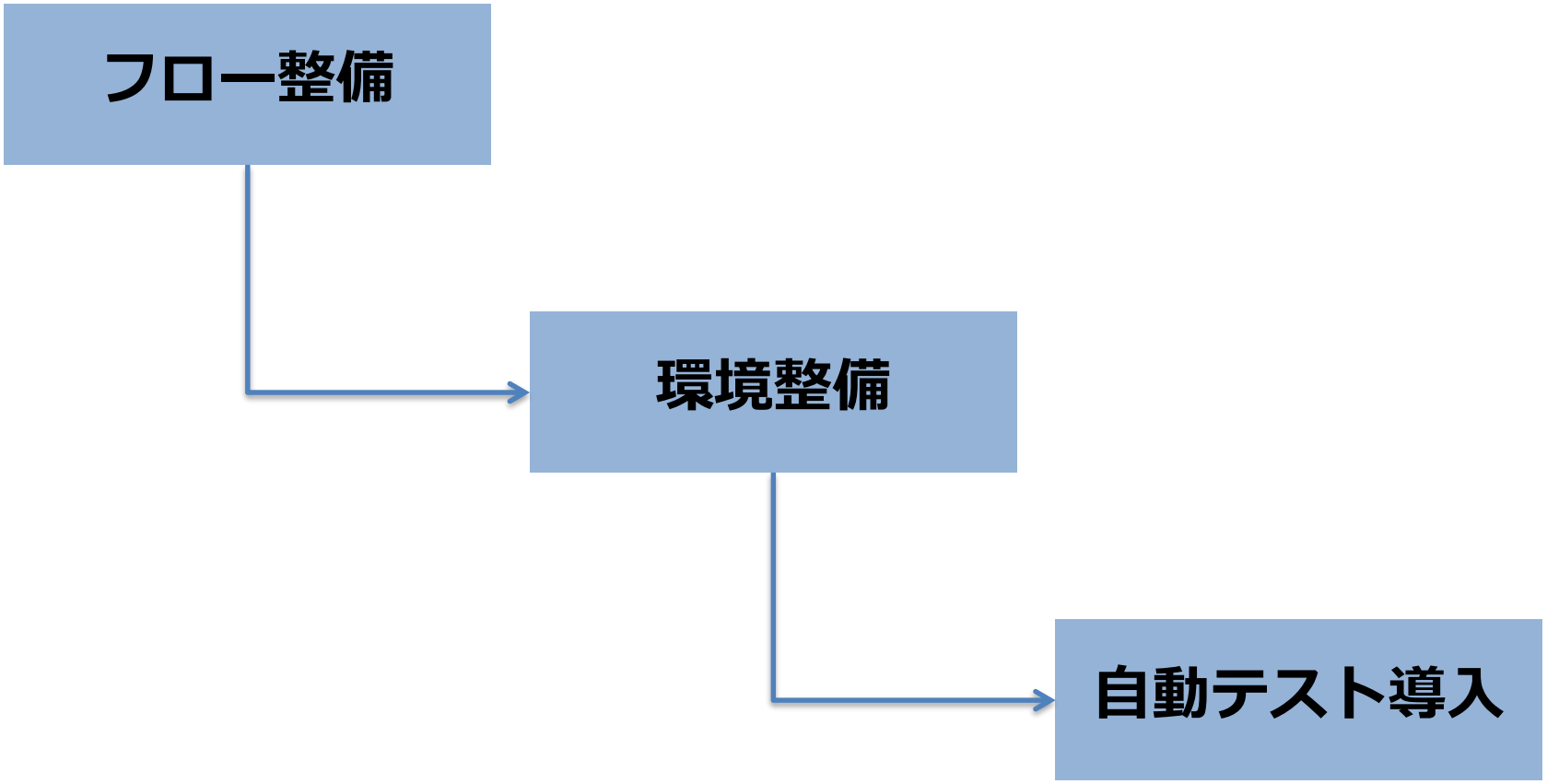
GUI：月間**176人日**抑制

月に**約600人日**の工数を抑制

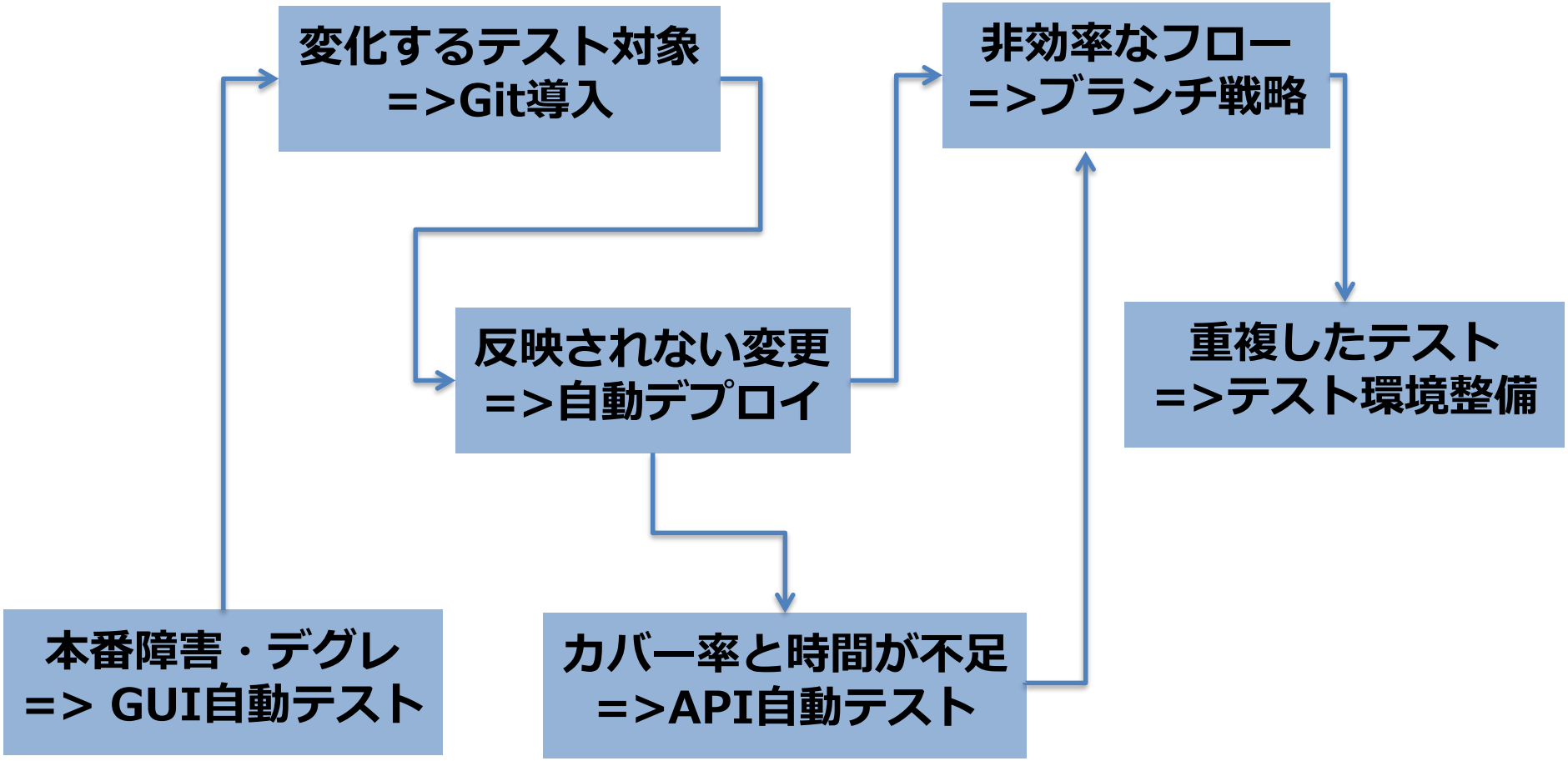
そんなに上手くいくの？

4. 改善時の障害と乗り越えた方法

理想的なフロー



実際は課題が課題を呼ぶ状況

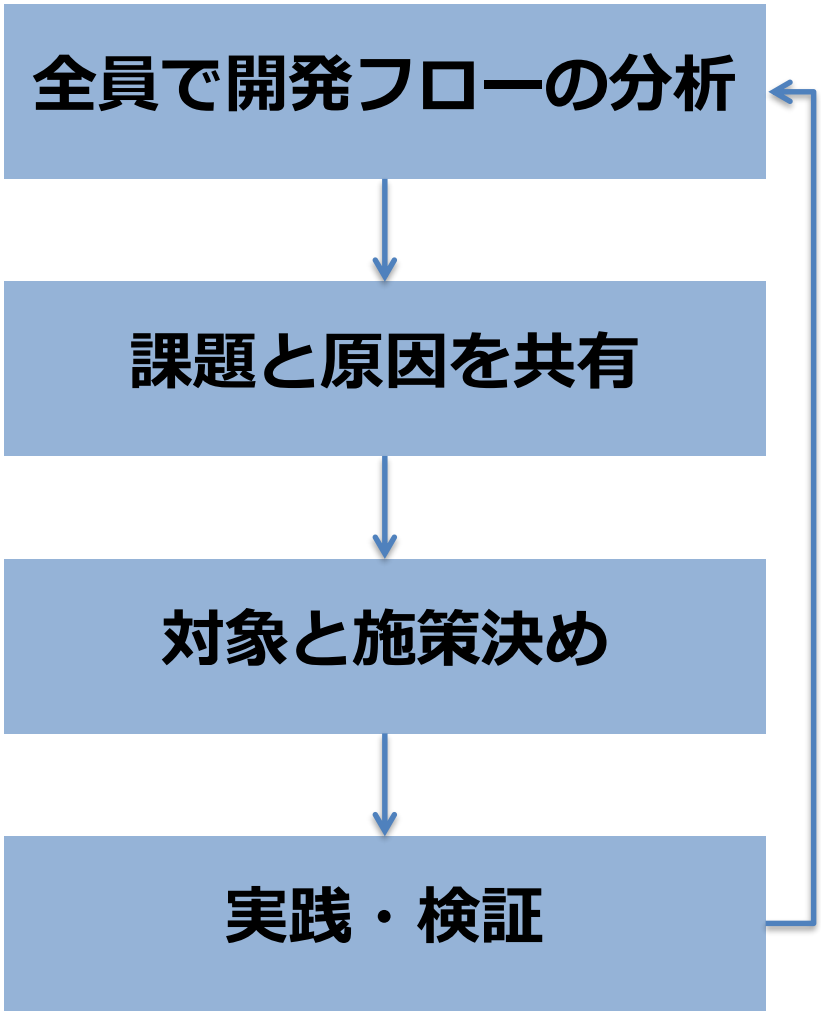


ぶつかった壁

- 施策が限定的にしか機能しない
- 何が本質的な課題なのか分からない
- 散発的な改善になり

  効果が出るまでに時間が掛かる

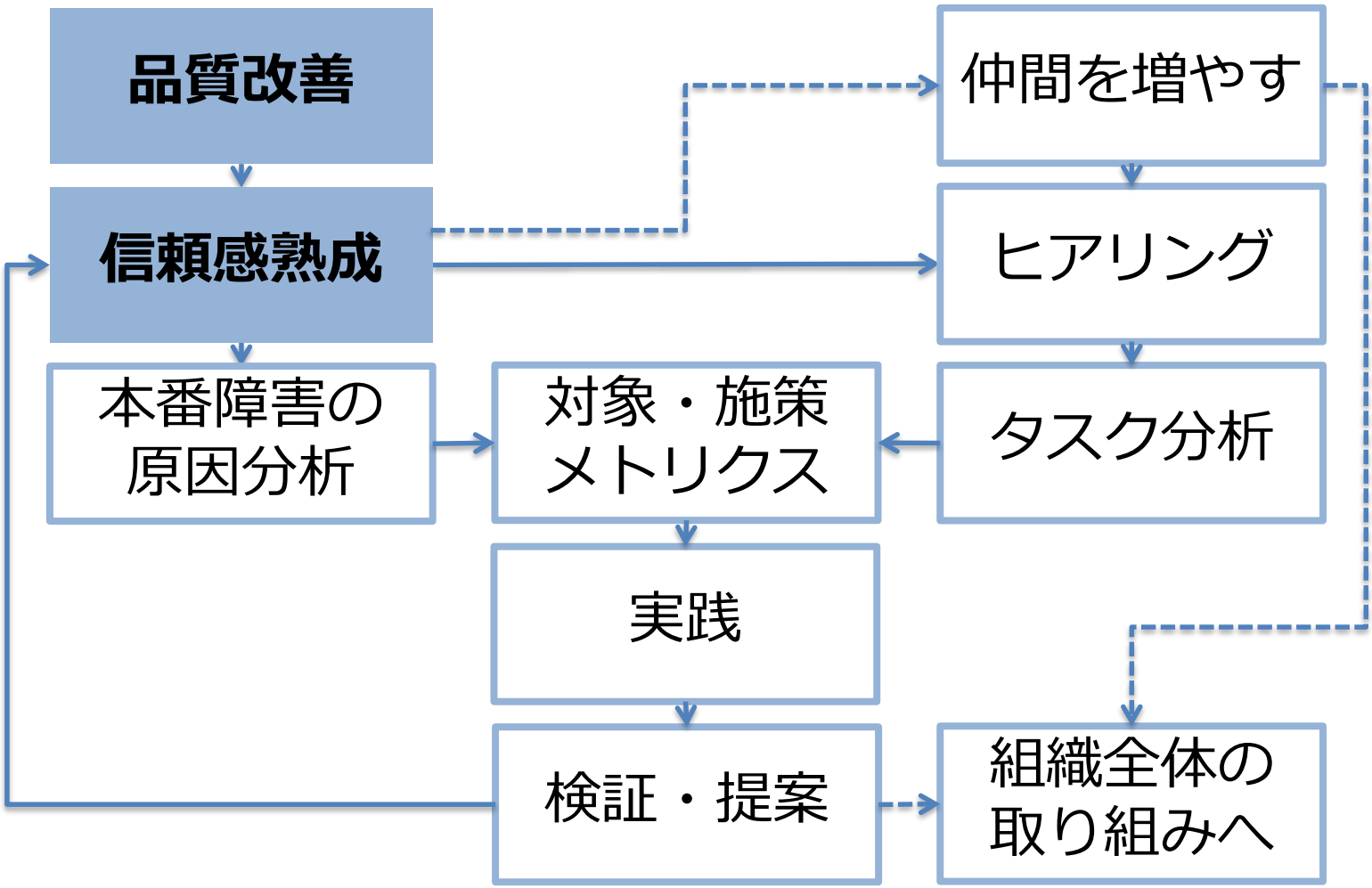
理想は課題の全体像から



組織を巻き込むことが一番の難関

まずは1人から、改善の流れを作る

普段の業務が土台になる



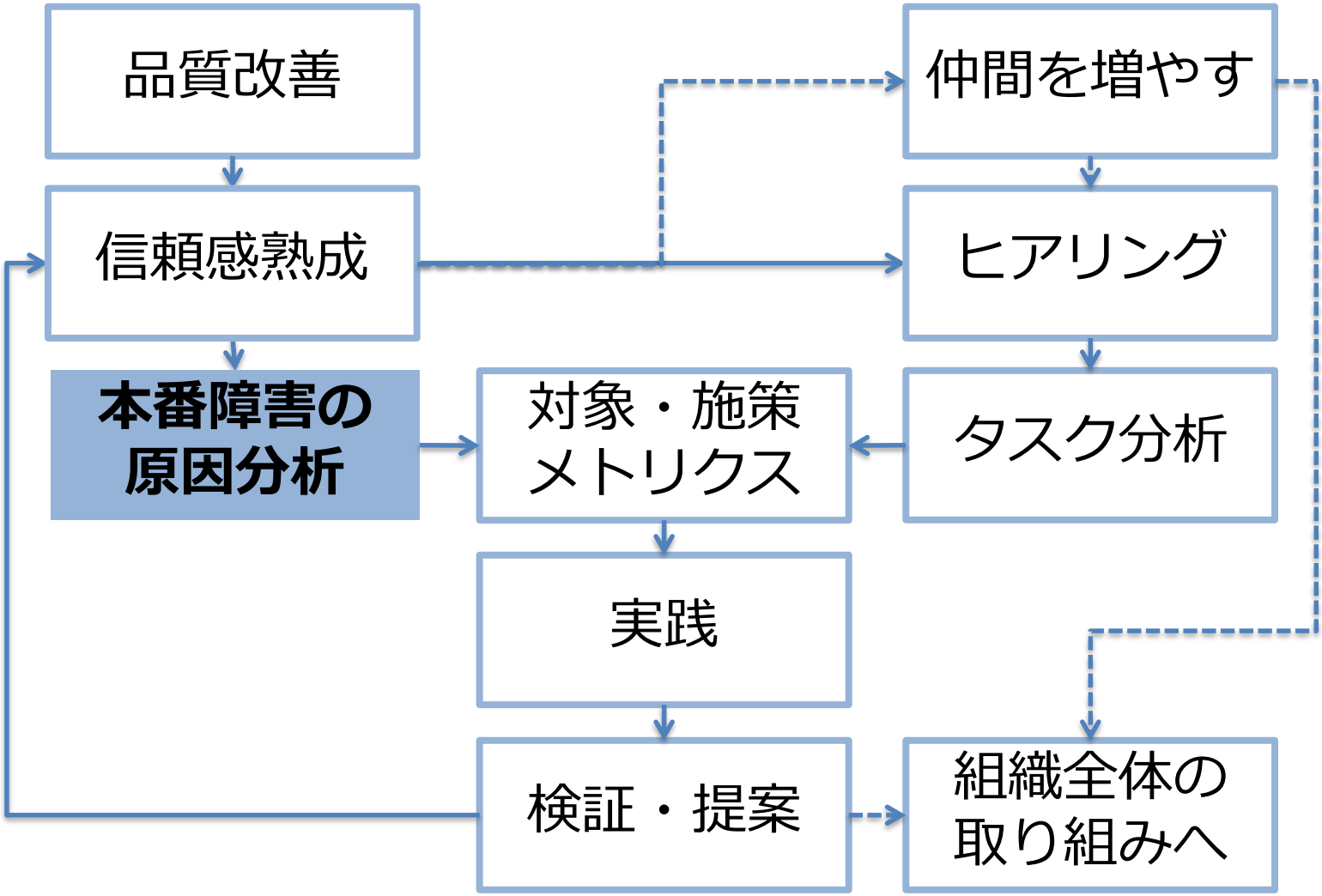
如何にしてQA業務から DevOpsに繋げるか

QAからDevOpsへの道

1) 本番障害分析

2) タスク分析

本番障害は宝の山


品質改善
信頼感熟成
本番障害の
原因分析
対象・施策
メトリクス
実践
検証・提案
仲間を増やす
ヒアリング
タスク分析
組織全体の
取り組みへ

本番障害から課題を洗い出す

- What:　不要な変更を取り込み障害発生
- Who: リリース担当者が
- Why: リリースをするために
- When:　マージ作業時に
- Where: 二つのブランチ間で
- How: 手動コピーを行なった

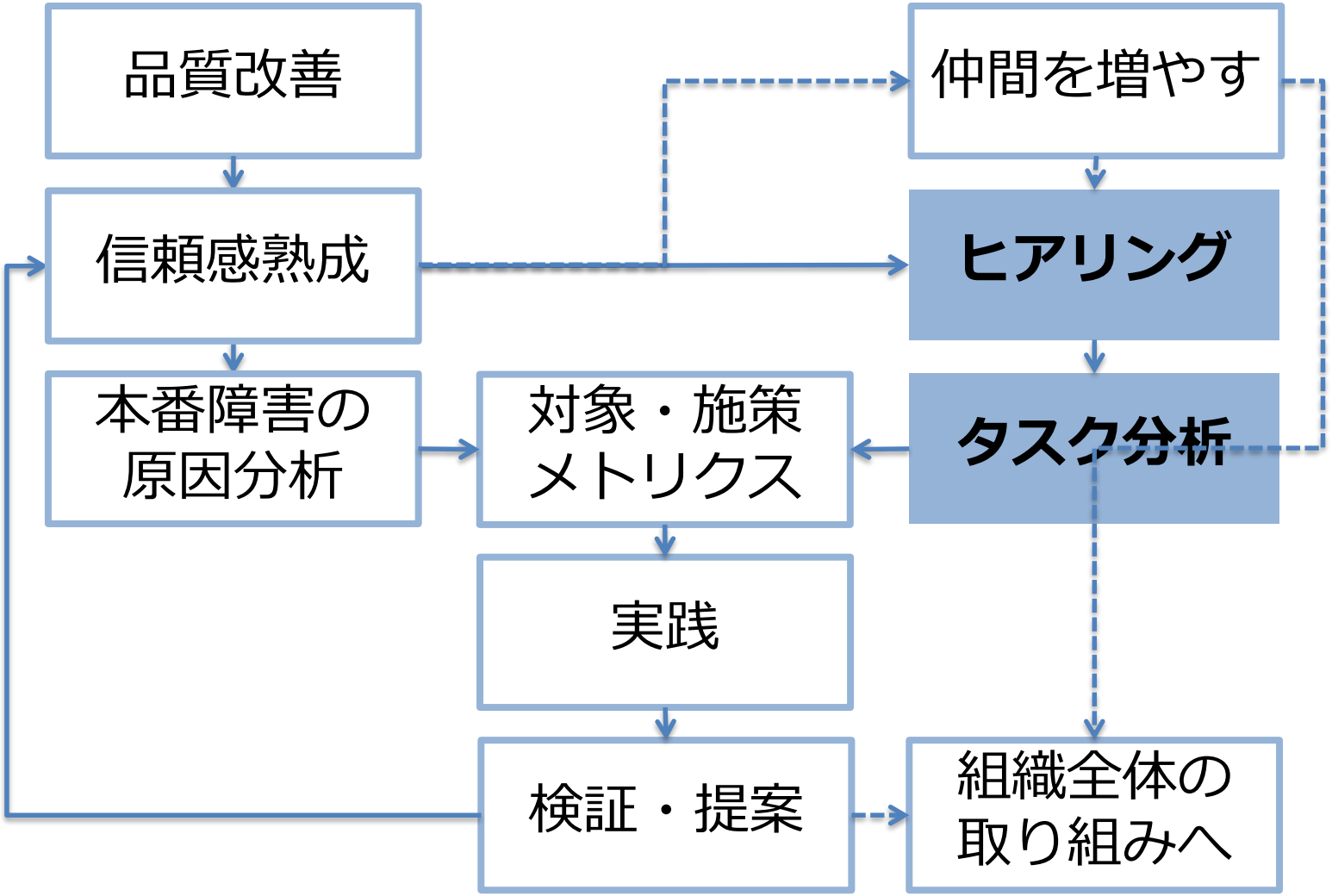
QAの枠を飛び出す

△　リリース前にテストをする、設計書の観点を見直す

◎　開発工程の中に不具合を作り込んでいる工程はないか

組織として解決すべき課題

1人からでもタスク分析



丁寧にタスクを
教えてくれる人は少ない

愚痴を聞くことから始める

- 何に苦労しているのか
- どんな作業をしているのか
- なぜそれが起きているか
- 本当はどうなって欲しいのか

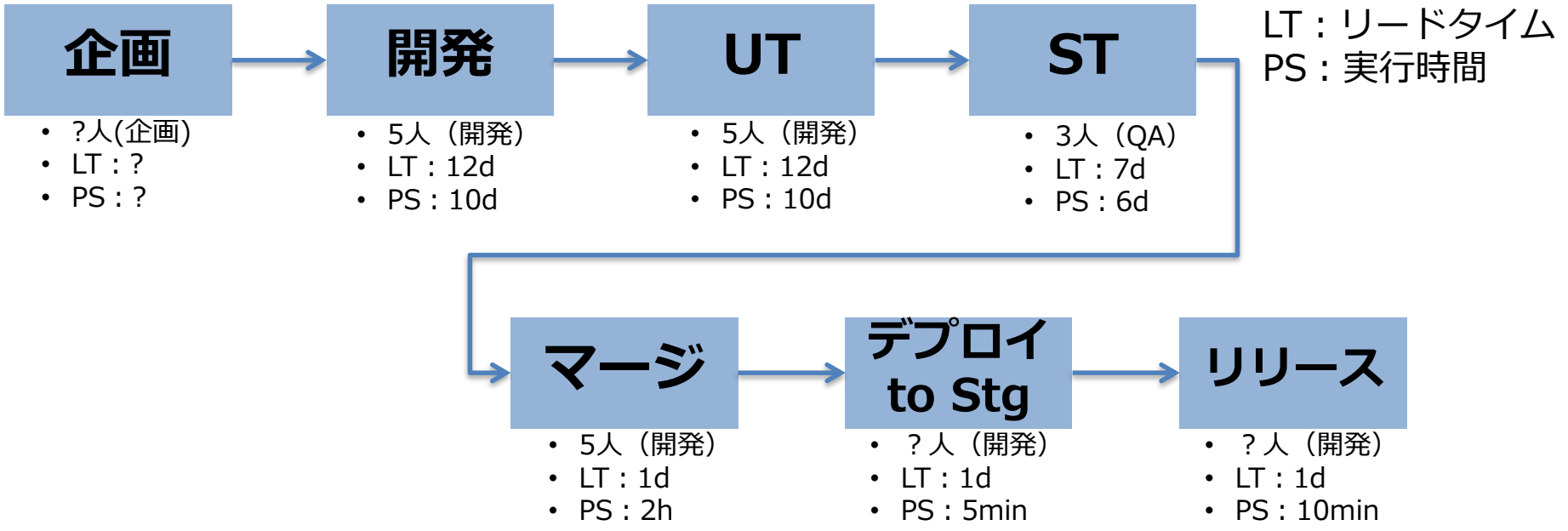
例えば

- 何に苦労しているのか
  - リリース作業がつらい、担当やりたくない
- どんな作業をしているのか
  - 手作業でコピーしてるんだよ！！
- なぜそれが起きているか
  - １つのブランチ、２つのリポジトリー
- 本当はどうなって欲しいか
  - 目視の確認とかやめて楽にリリースしたい

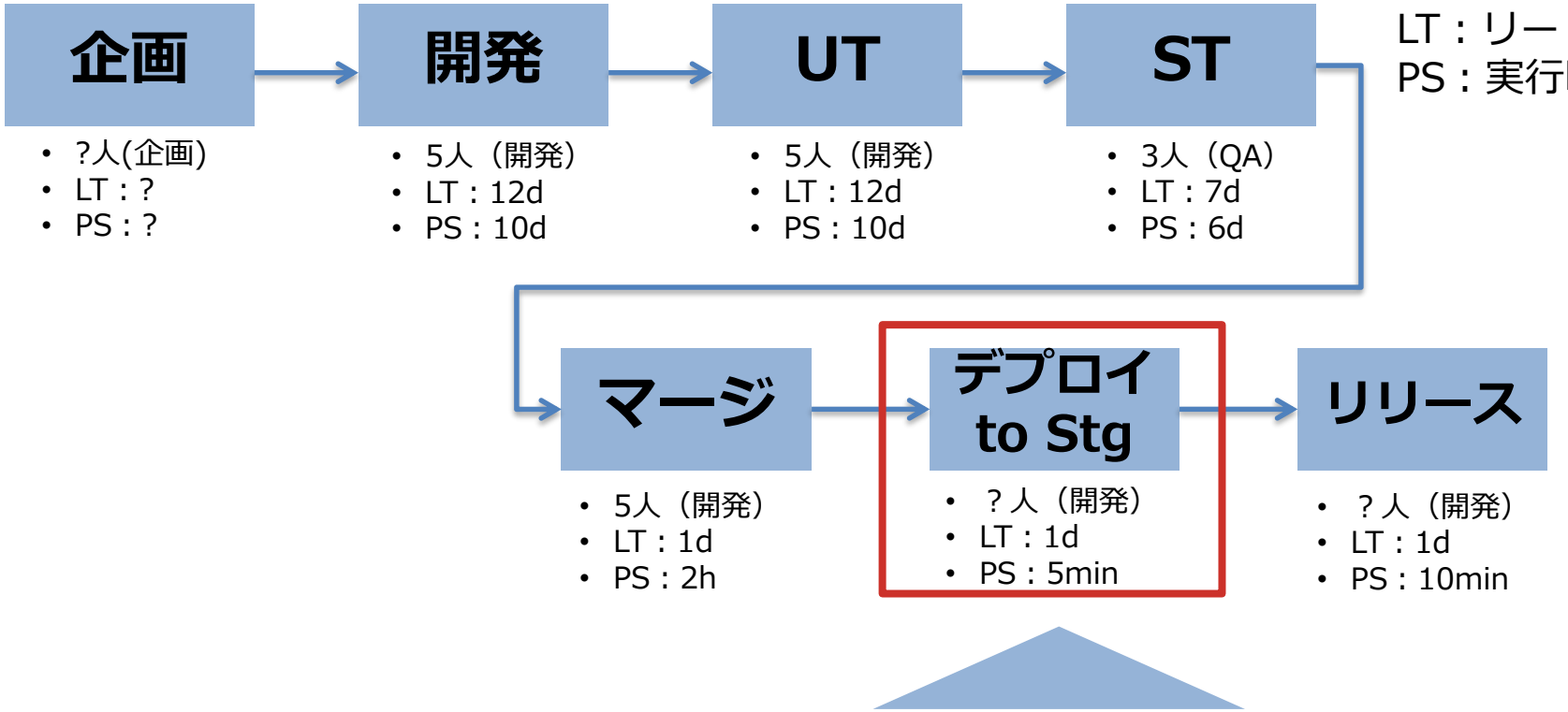
分かる情報からフローを書く

下記のような情報からタスクを洗い出し並び替える

1. 既知の情報
2. ヒアリング情報
3. チャットツールやメール、BTSなどの共有情報

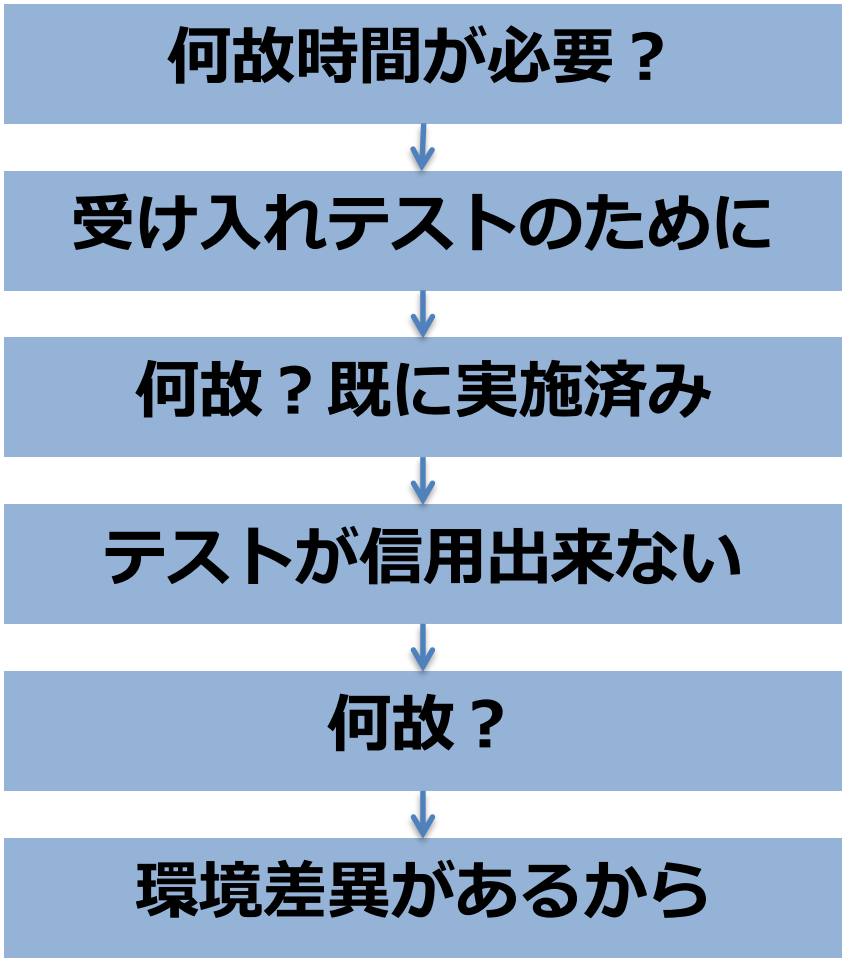


違和感を感じる部分を深堀する

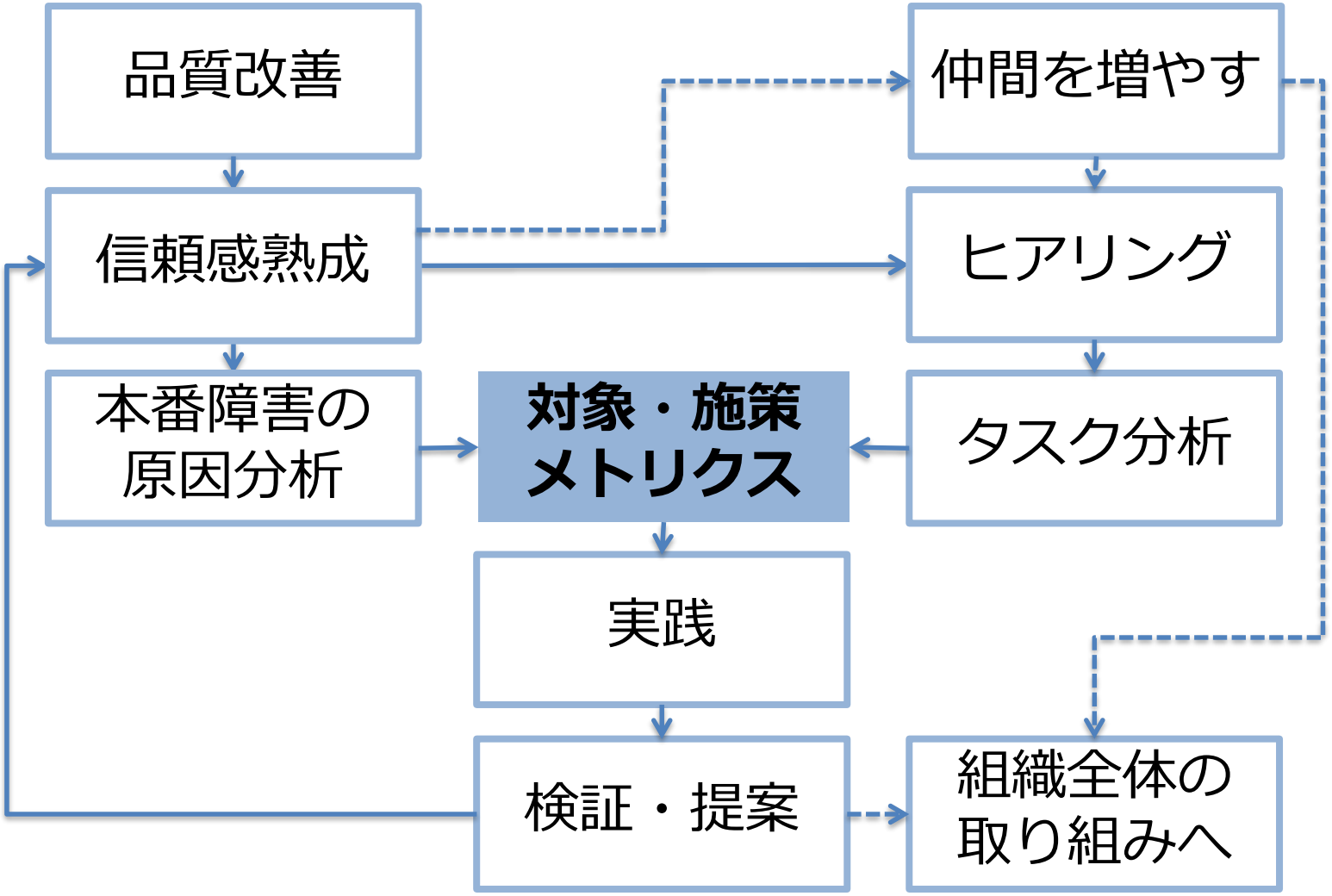


PSは短いのにLTが何故長い？

深堀りによって課題と原因を発見



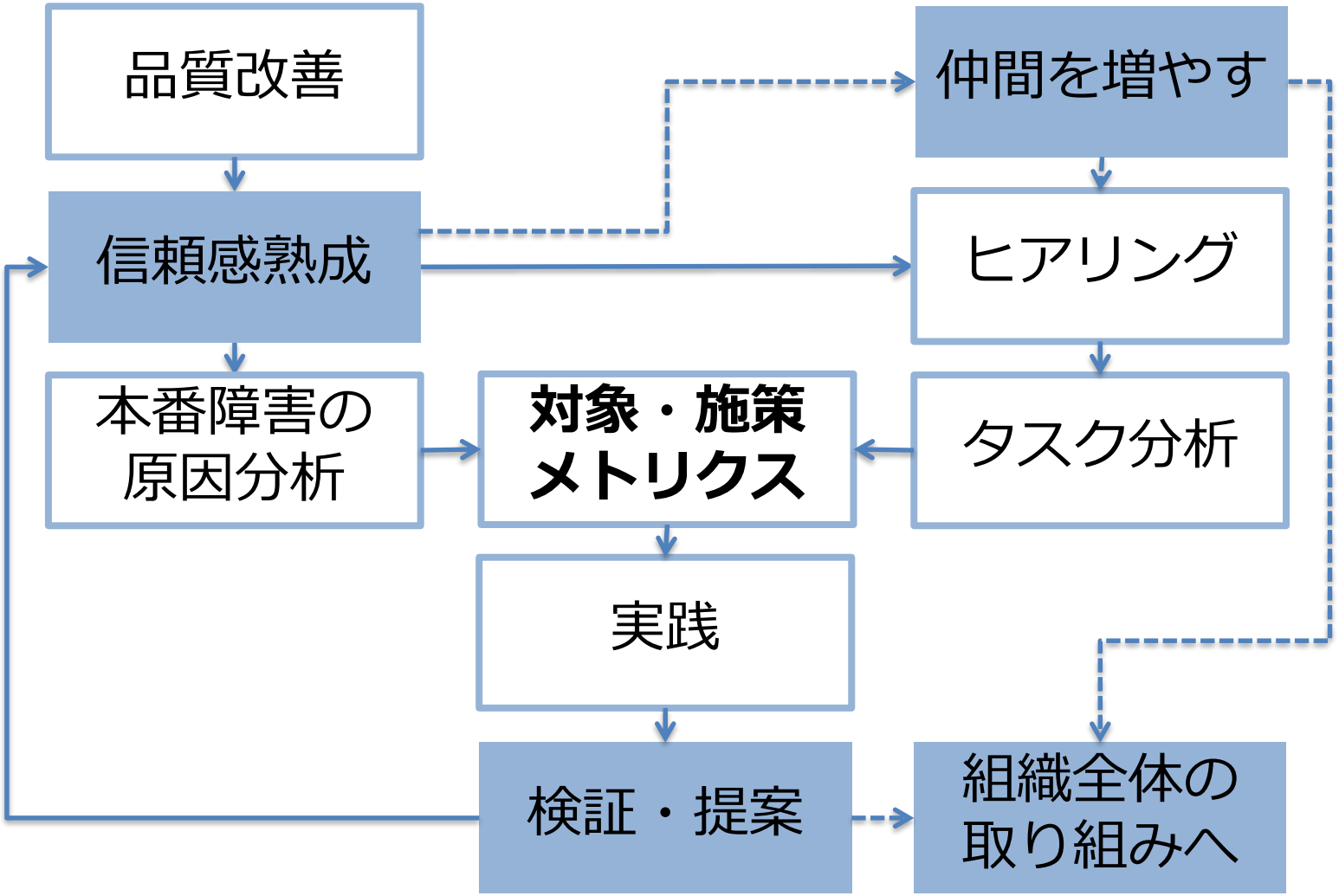
二つの分析から改善対象を決定



実施可能かつ効果最大のものを選択

1. 手動マージ作業中に不具合混入
   - 施策：ブランチ戦略とリリース戦略を策定
   - メトリクス：マージミスによる不具合を~%減
2. デプロイ時に作業漏れによる障害発生
   - 施策：作業項目の整理＋自動デプロイを導入
   - メトリクス：LTを~%削減、作業漏れを0に
3. 環境差異により障害発生
   - 施策：環境を統一
   - メトリクス：環境起因の不具合を~%減
     新メンバーの参加後の立ち上がり時間~%減

如何にしてチームを巻き込むか



立場にあった効果を示して提案

決裁者

- ビジネスに貢献出来るか？
- 他部署に説明がつくか？

上司

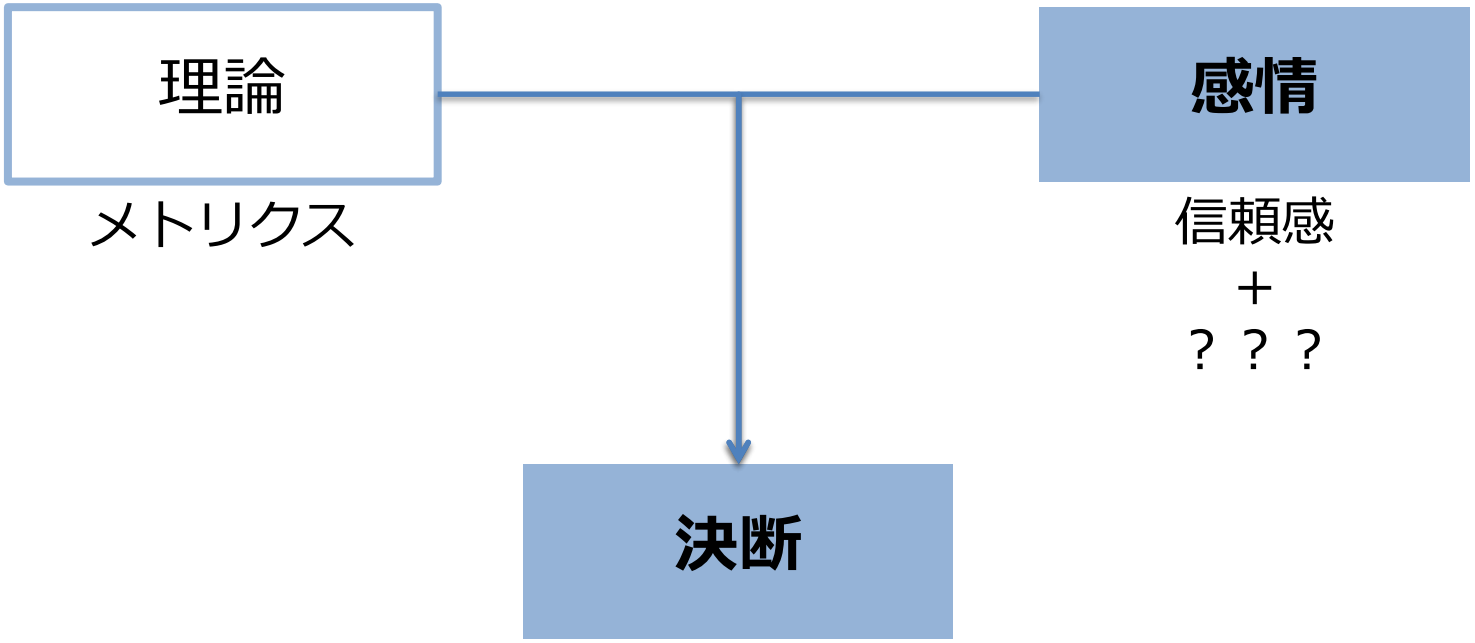
- 報告出来る成果か？
- 実施内容をイメージ出来るか？

同僚

- 楽になる？
- 楽しくなる？
- スキルアップ出来る？

必ず成功する改善プランは存在しない

理論と感情の両面から



共感

改善への強い思いとビジョン

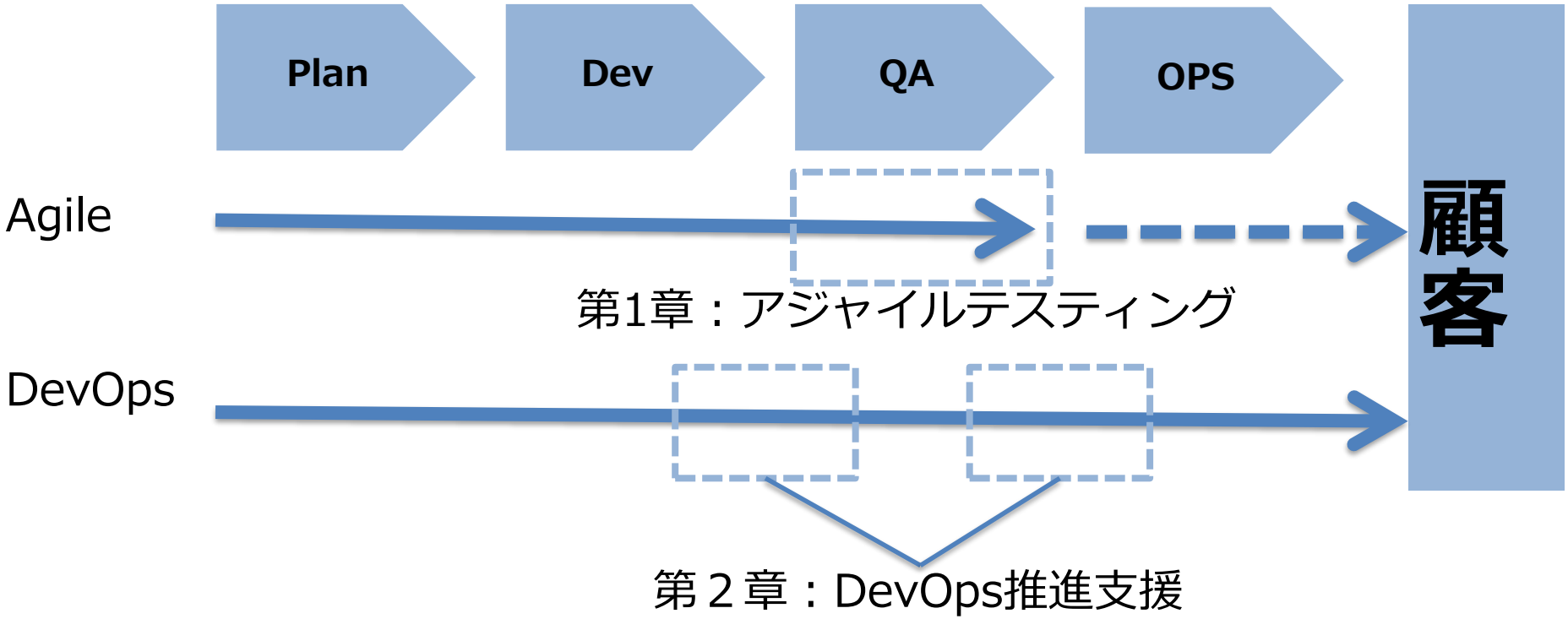
5.（第2章）まとめ

まとめ

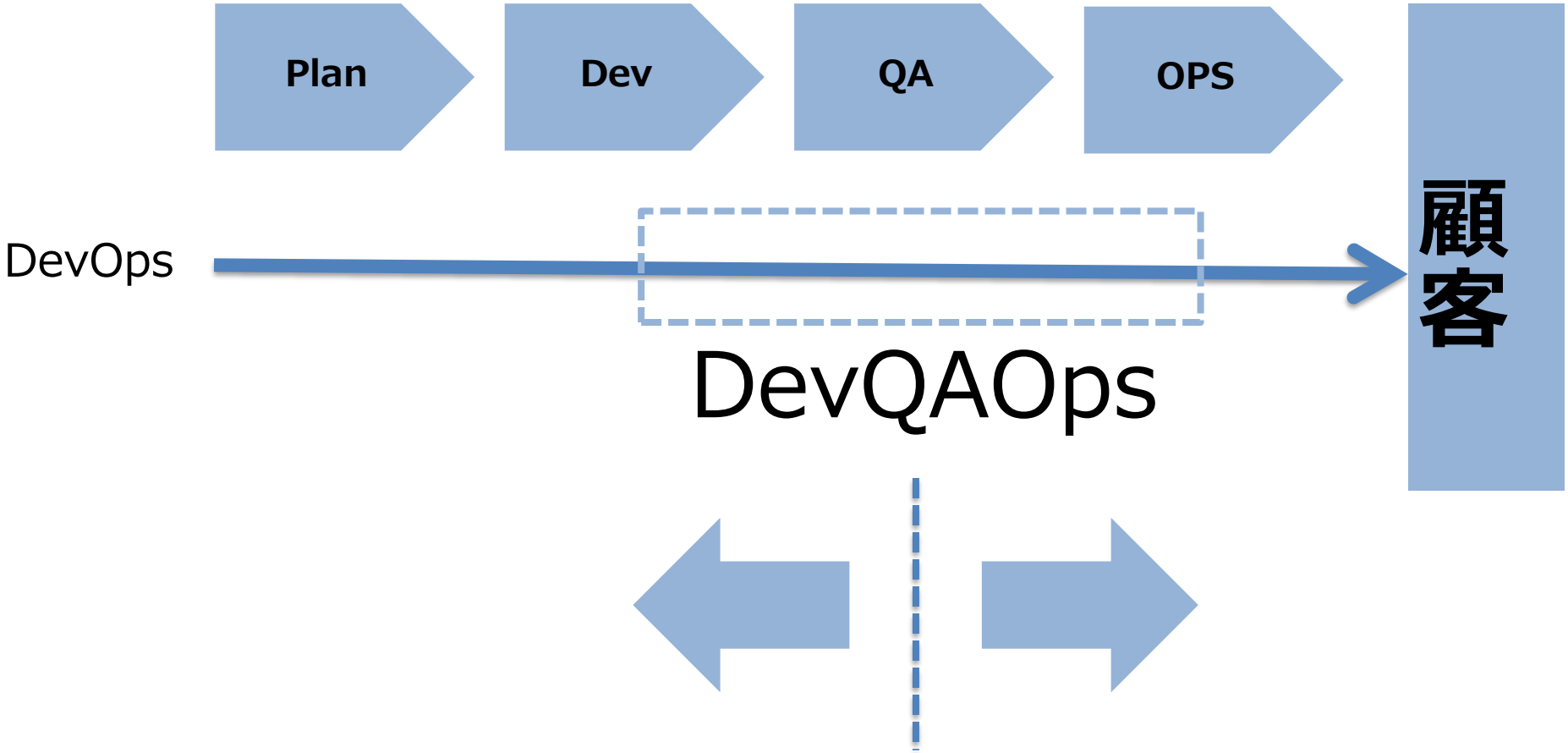
- QAの枠を飛び出す
- 本番障害とフローが解決のヒント
- やりきった後は共感によって動かす

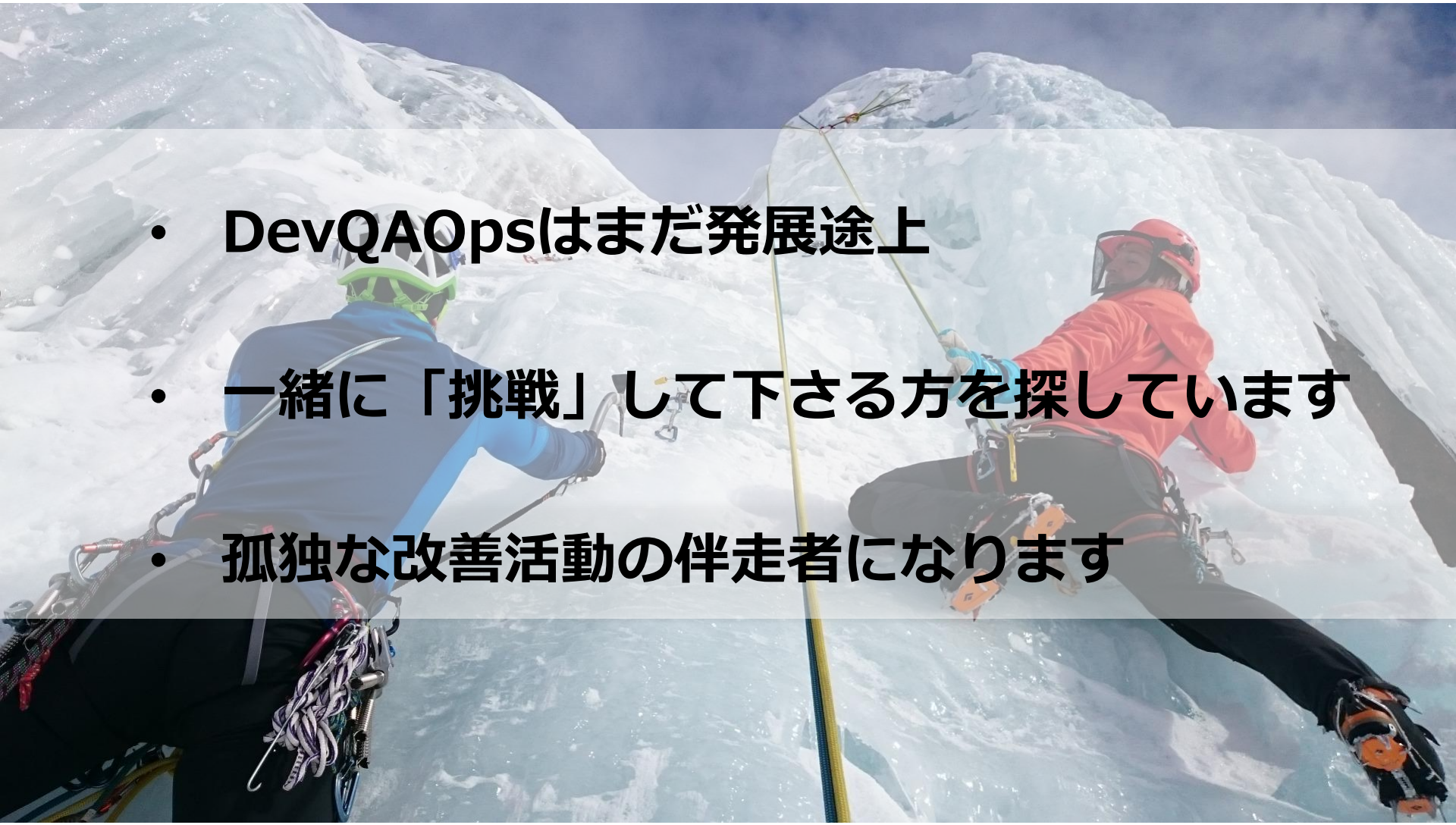
（全体）まとめ

QAからでも様々な改善活動が出来る



QAを軸にしたDevOps支援サービスを開発





- DevQAOpsはまだ発展途上
- 一緒に「挑戦」して下さる方を探しています
- 孤独な改善活動の伴走者になります

ご清聴ありがとうございました