

Rebuild Team

急成長プロダクトのDev&Opsで生じる悪循環とその解決策

2017-10-13 (Fri)
SPI Japan 2017

presented by @yuzutas0

<https://www.pexels.com/photo/colleagues-cooperation-fist-bump-fists-398532/>

90枚 / 20min

どんどんいきます

Sho Yokoyama (@yuzutas0)



- リクルートテクノロジーズ
- 認定スクラムマスター
- エンジニアチームの立ち上げ・立て直し

本日の内容

“急成長に伴う痛み”を1つ1つ解消して
チームを安定化した事例を共有します

想定する対象

急成長フェーズのプロダクト開発・運用に関心のある

- ソフトウェアエンジニア
- チームリーダー
- 組織マネージャー

アジェンダ

1. 背景・課題

2. 改善活動

①サービスレベル ②セレモニー ③バリューストリーム ④ドキュメント

3. 振り返り

成長痛に苦しむプロダクト

会社の次を担う事業としての期待

売上3倍/年

グロース製品

メンバー2倍/Q

スケール期

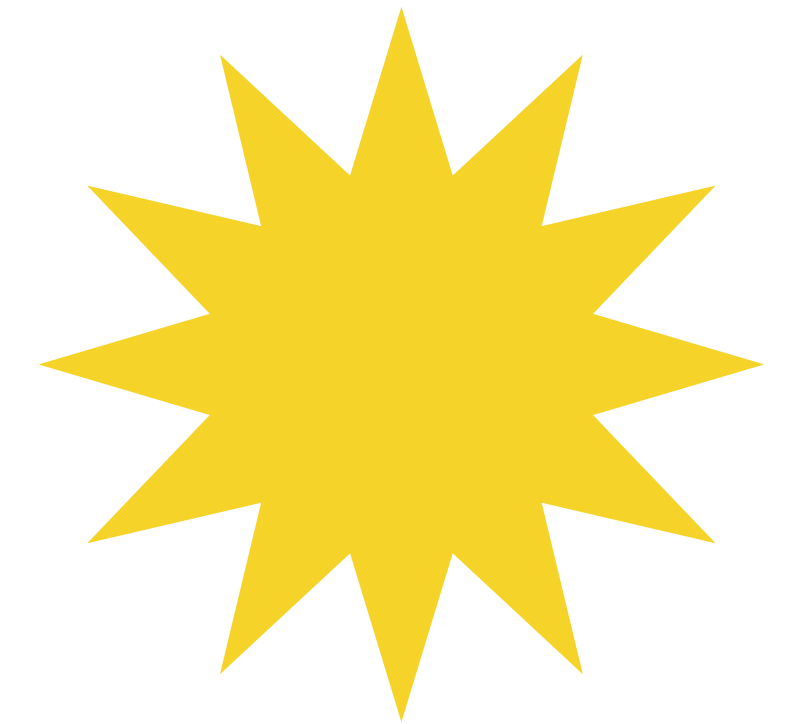
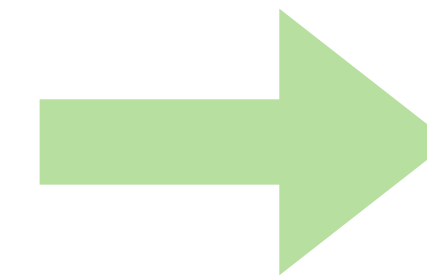
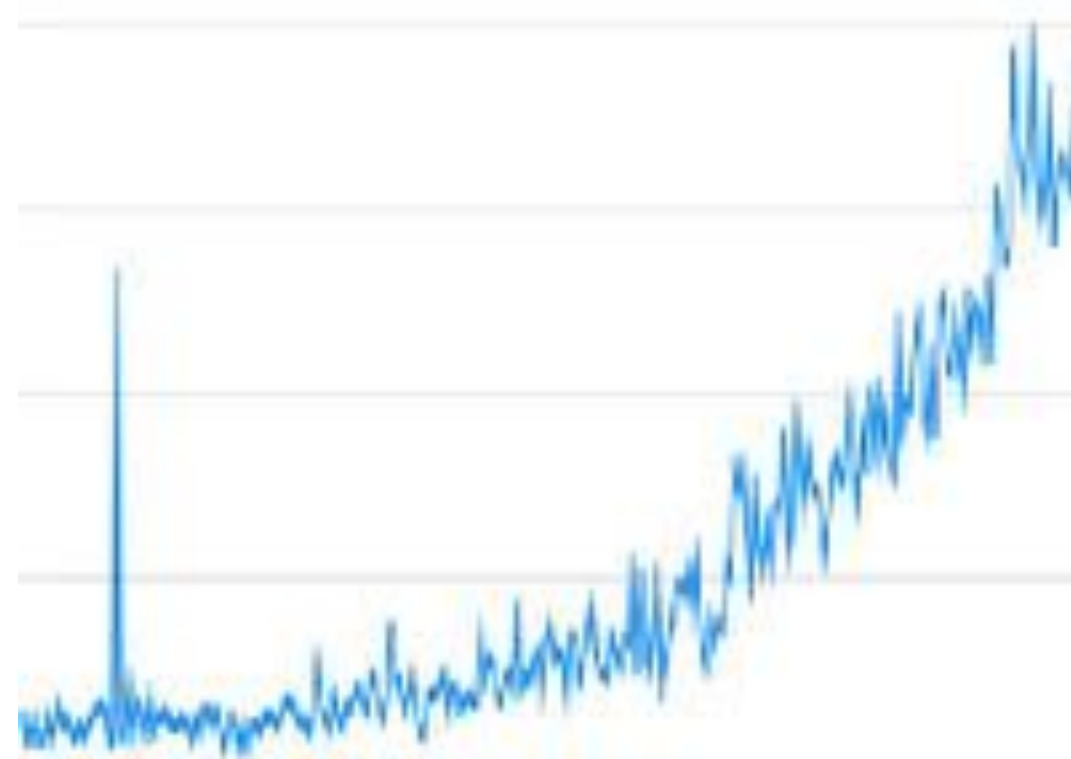
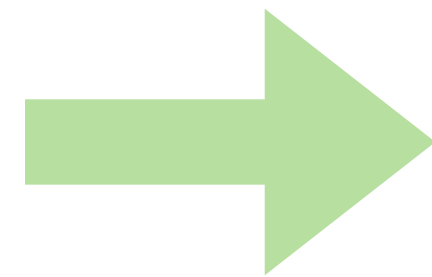
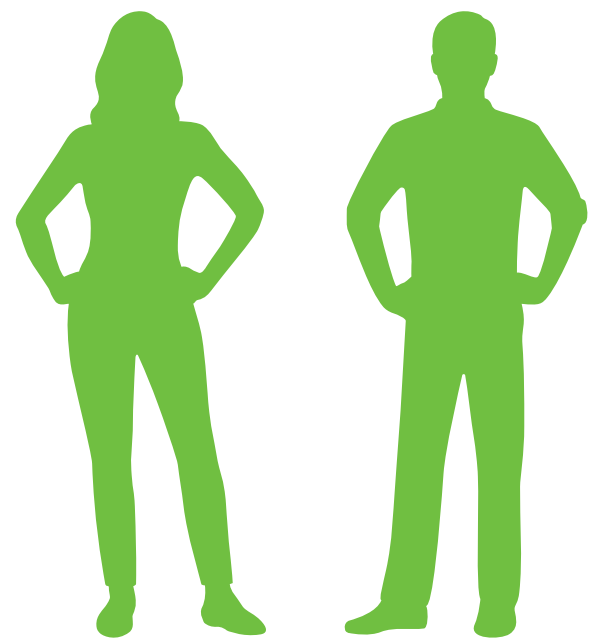
障害2件/営業日

危機的状況

チーム立て直しの必要性

“0→1”が生んだ負債

ヒットするか不確定でコストを割けなかった



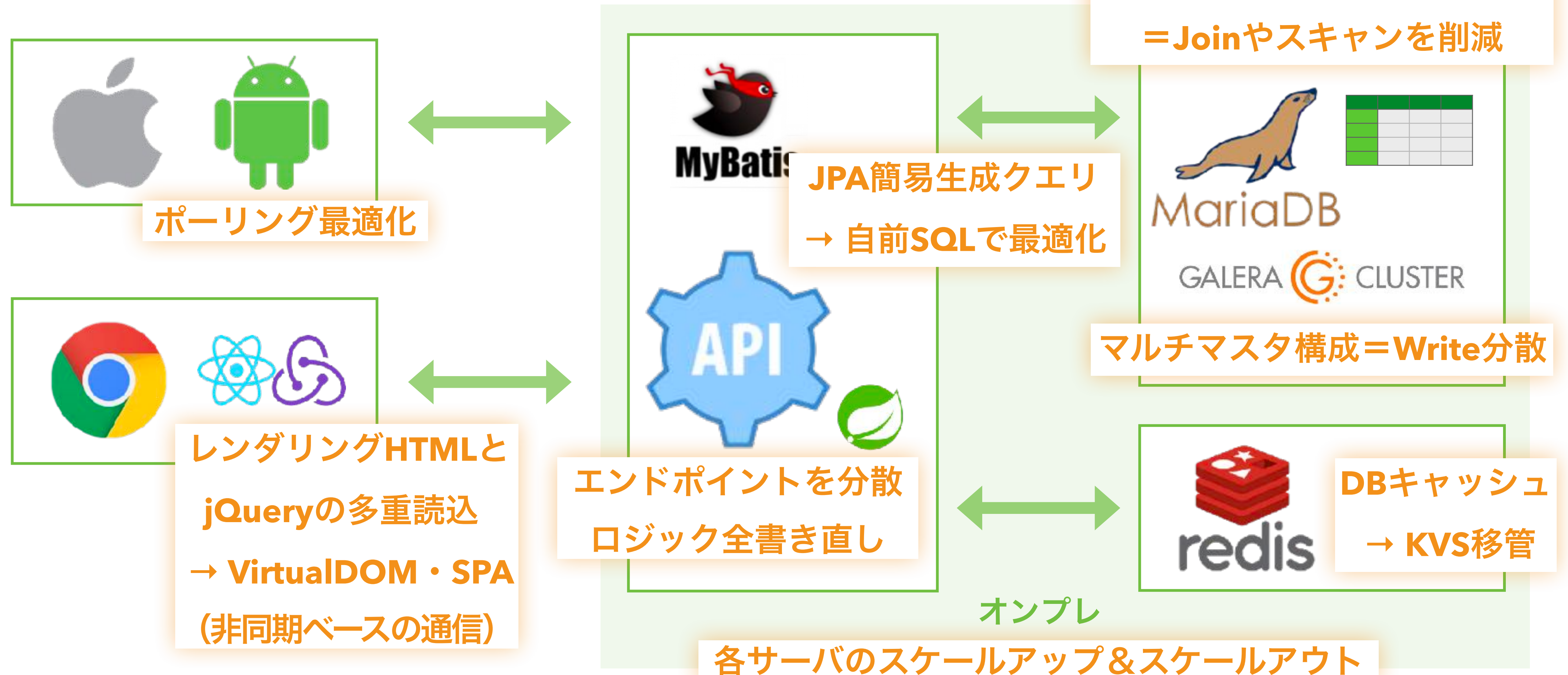
外注に丸投げ
システムは低品質

市場拡大
ユーザー増加

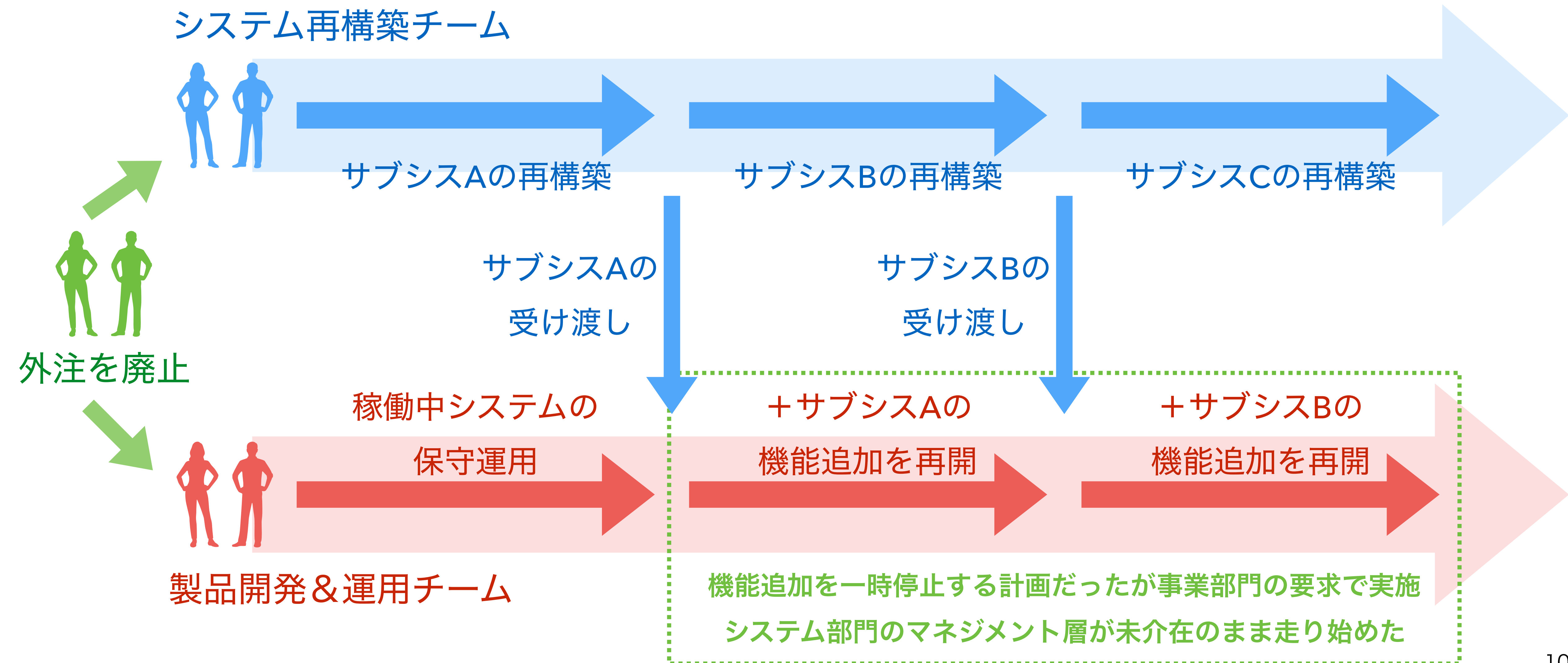
システムの限界
ピーク時間帯の応答が20sec

“1→10” に備えた再構築

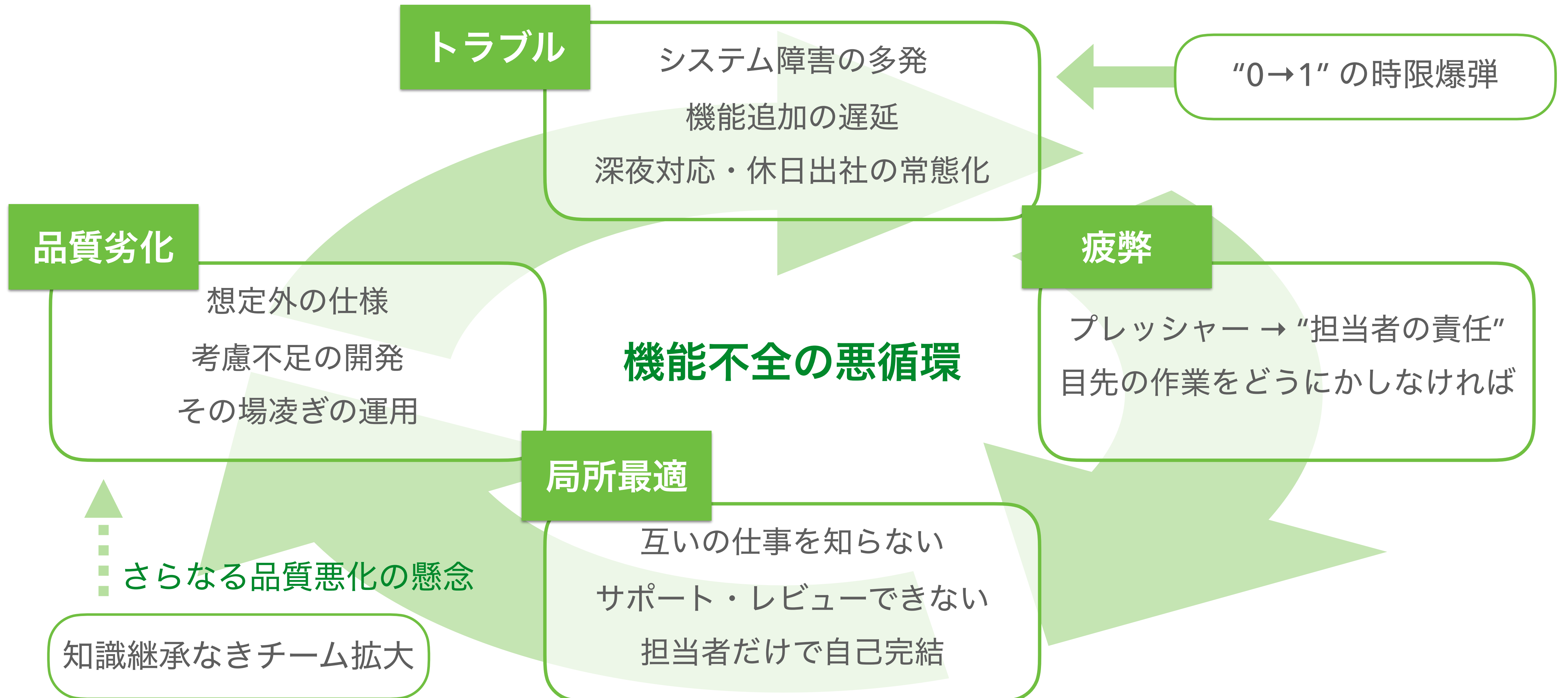
ピーク時間帯のレスポンスタイムを0.05secに400倍の改善



"1→10" に備えた内製化



Dev&Opsで生じた痛み



ミッション

- 開発・運用チームを立て直すこと
- さらなる成長の土台を構築すること

アジェンダ

1. 背景・課題

2. 改善活動

①サービスレベル ②セレモニー ③バリューストリーム ④ドキュメント

3. 振り返り

なにはともあれ火を消す

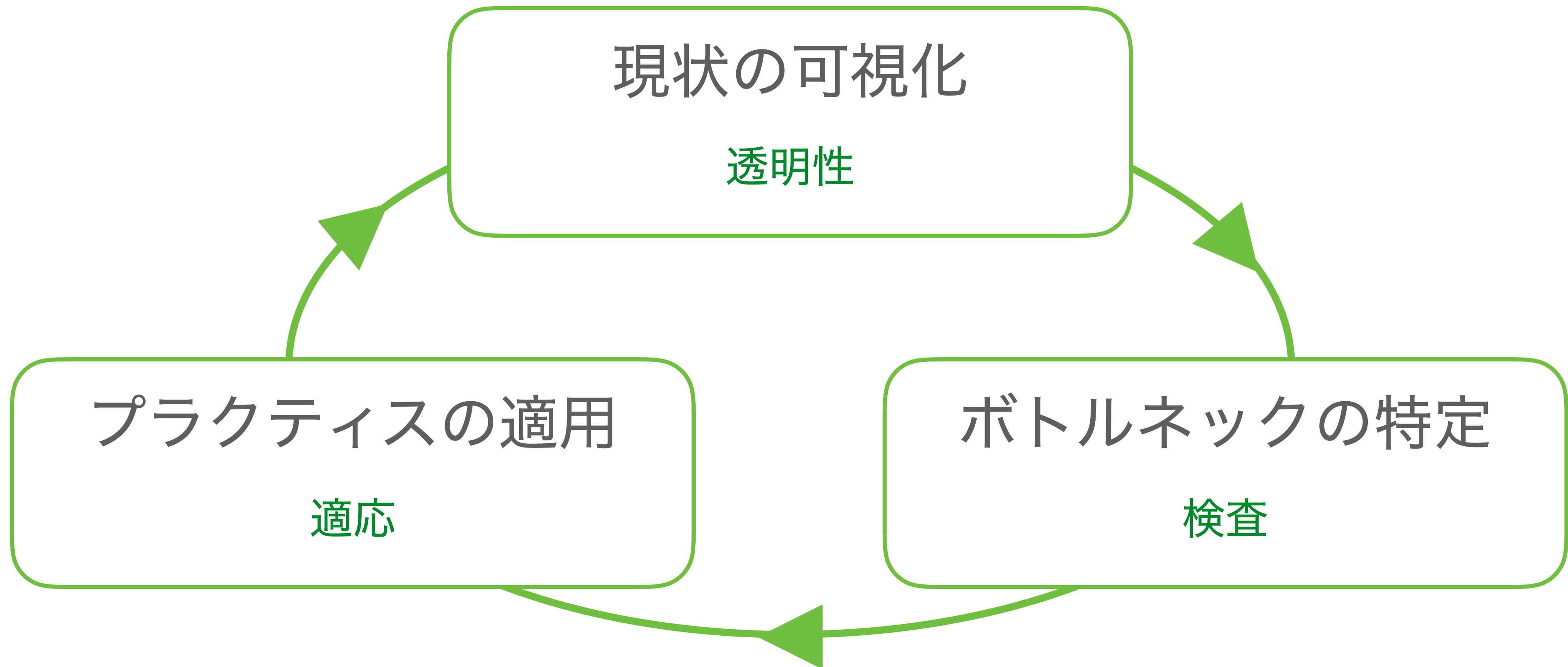
システム障害が営業日あたり2件発生する状況が1ヶ月続いた

→ 全チームのあらゆる開発を止めて沈黙化

徐々に収束したが

- 恒久対応は申し送り
- 週に何回か時限爆弾が爆発
- リリースの度に新たな障害が発生

悪循環を打開する改善活動

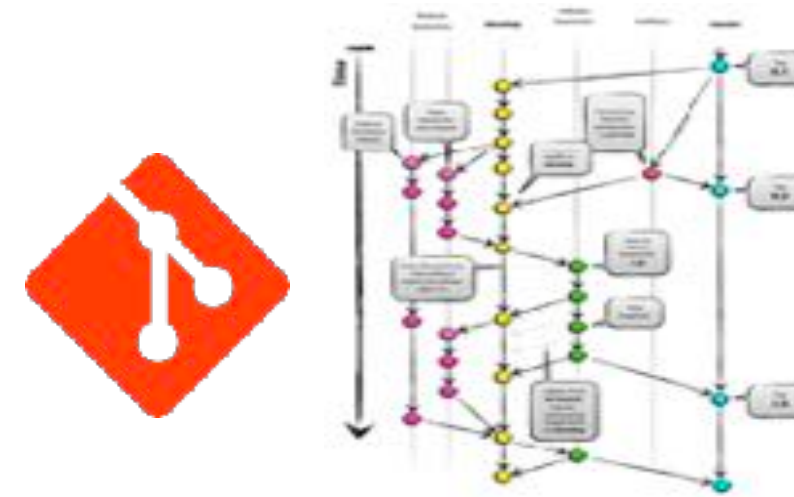


改善サイクルをひたすら回し続けた

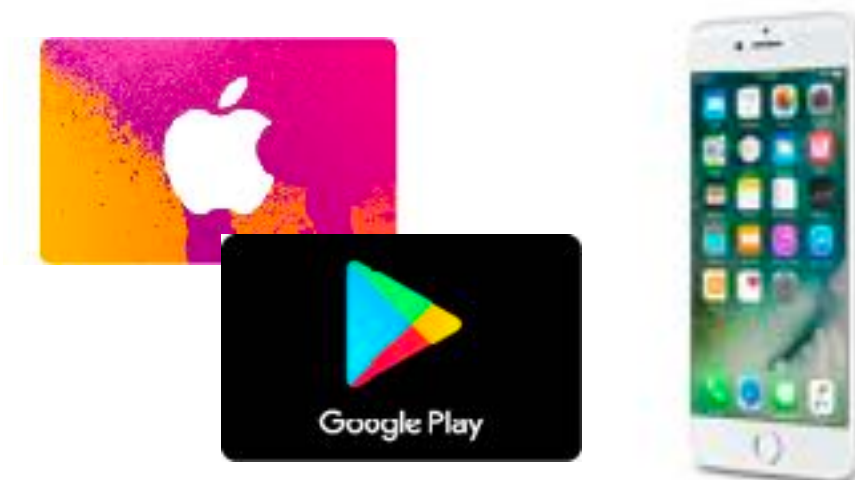
※技術観点もコツコツ改善しながら



コード品質の担保



ブランチ戦略の導入



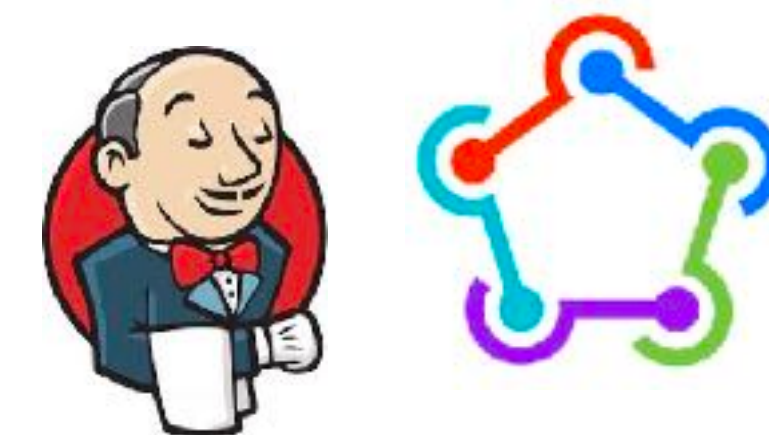
テスト向け物資管理



QA支援・テスト強化



要件フォーマット整備



リリース作業の自動化

※外部リソースで不足を補いながら

物資：ホワイトボードやモニター

知識：特定分野の専門家サポート



結果的に「現場だけで無理に解決しない」「相談すれば解決できることもある」のメッセージにもなった

アジェンダ

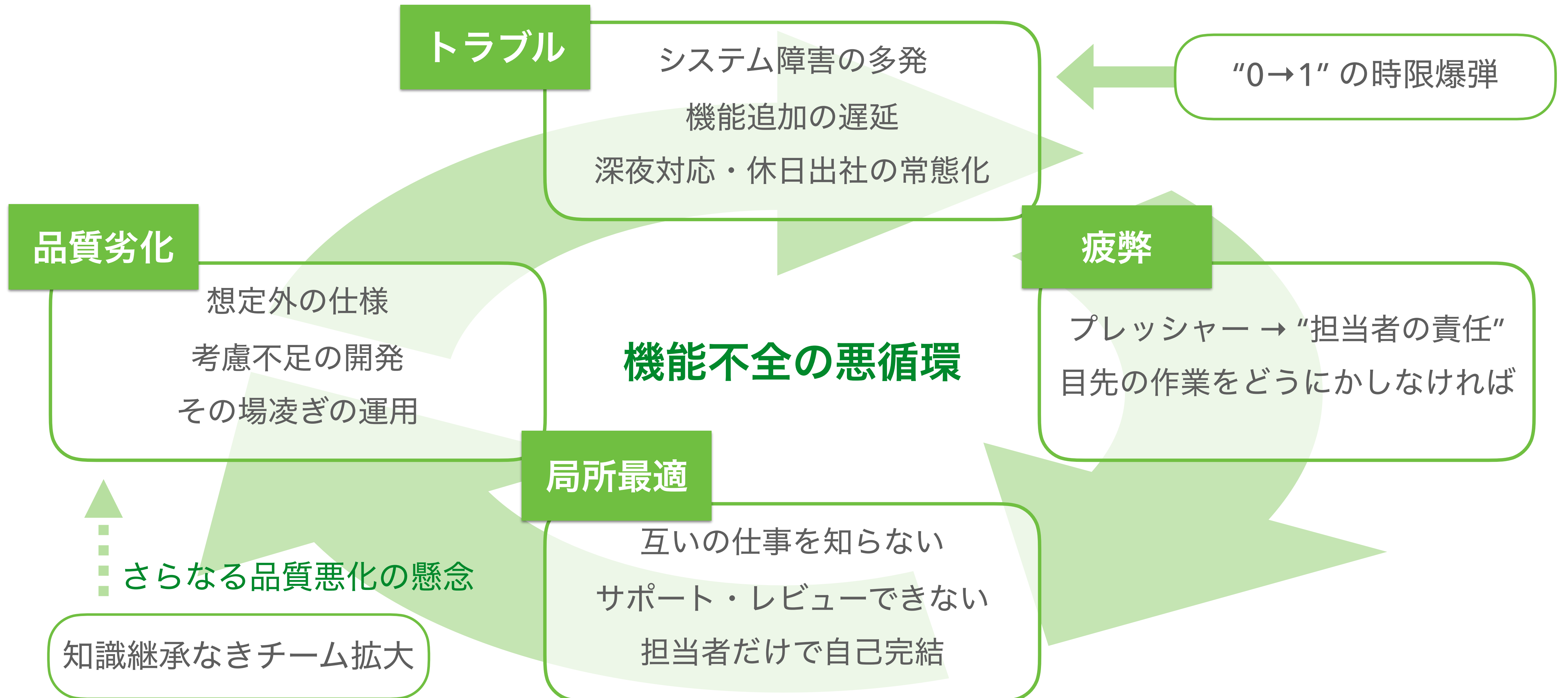
1. 背景・課題

2. 改善活動

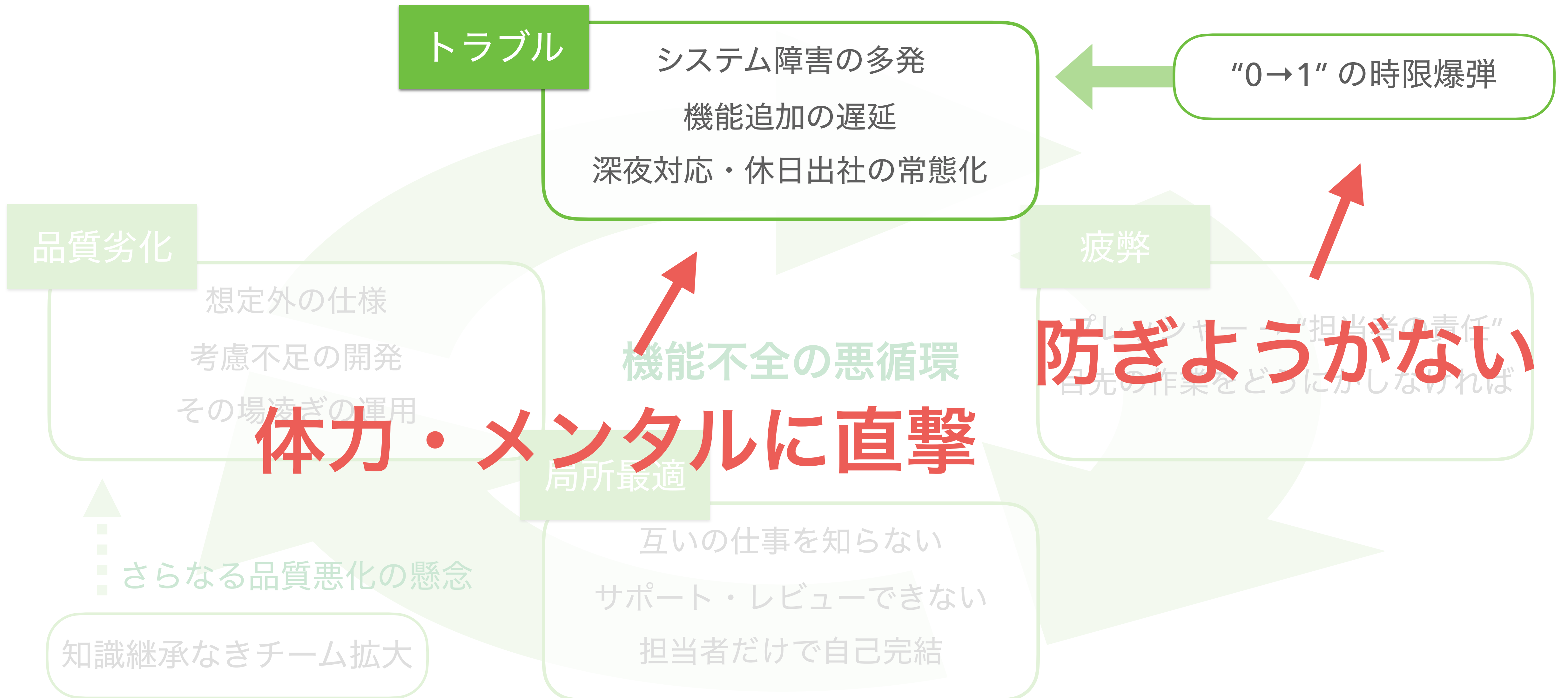
①サービスレベル ②セレモニー ③バリューストリーム ④ドキュメント

3. 振り返り

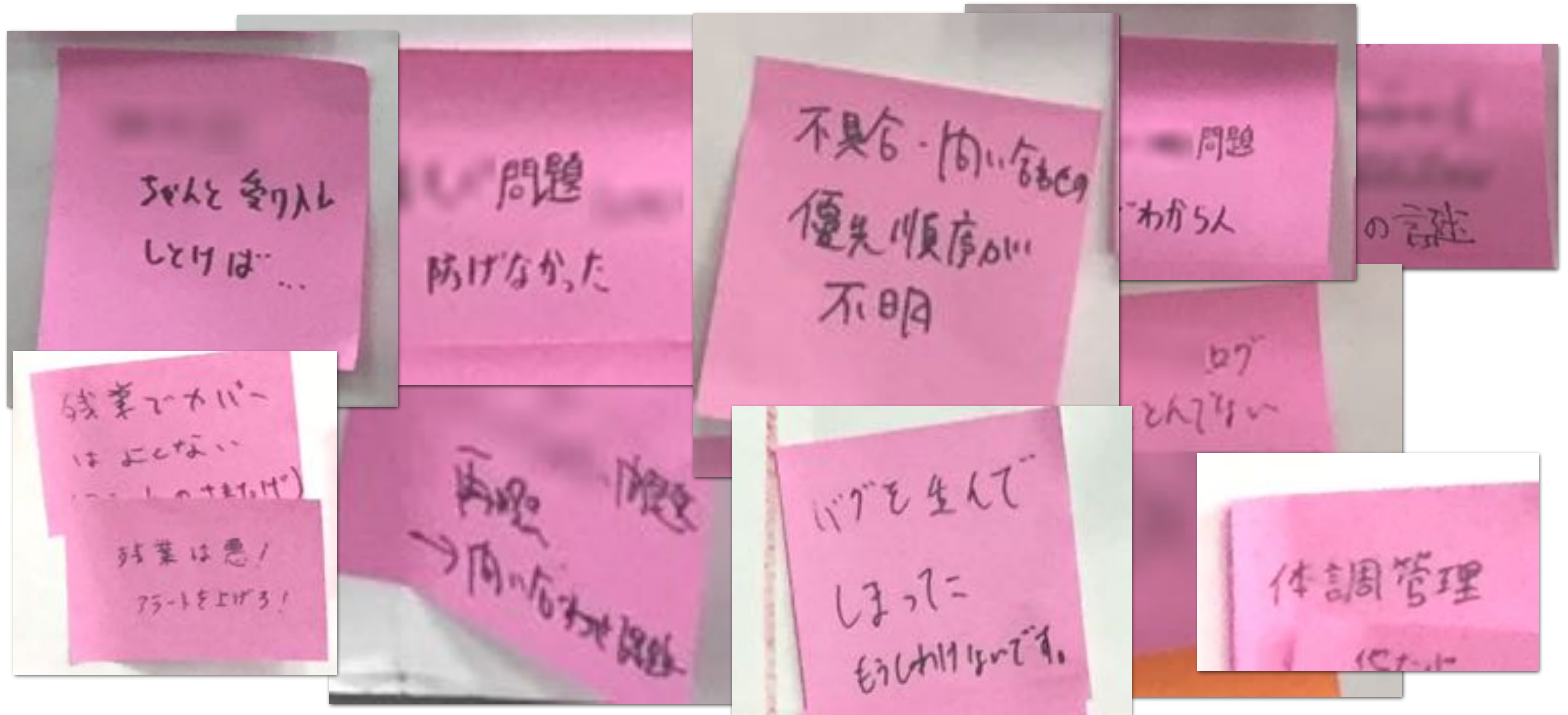
Dev&Opsで生じた痛み



ボトルネック



混乱したまま障害に対峙する



障害の傾向分析から着手

- 軽微な不具合でのオンコールや残業対応が発生していた
- 不適切な対応による二次災害が少なからず発生していた

サービスレベルを定義

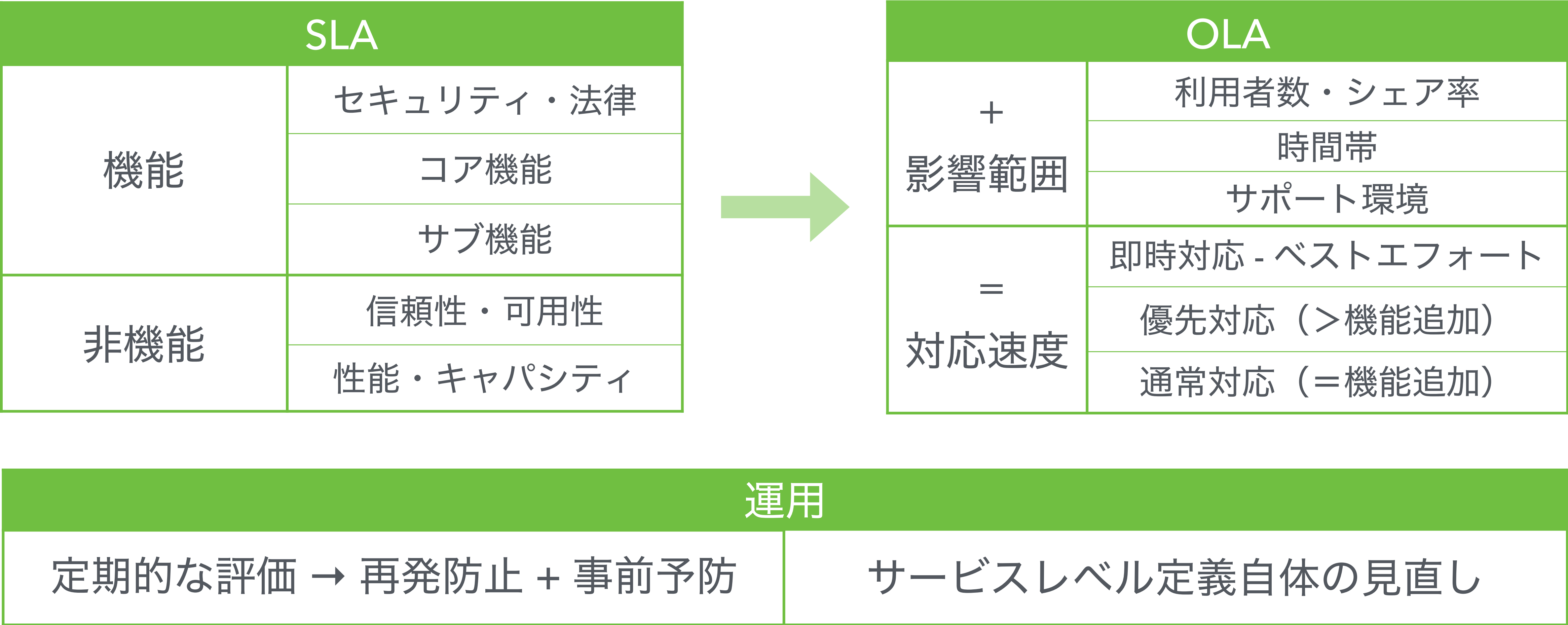
項目	内容	備考
1. 概要	本計画は、〇〇地区の発展を目的として、地域の特性を活かした産業の育成と、住民の生活の向上を図ることを目的とする。	
2. 目的	〇〇地区の産業の育成と、住民の生活の向上を図ることを目的とする。	
3. 計画の範囲	〇〇地区の範囲内とする。	
4. 計画の期間	〇〇地区の範囲内とする。	
5. 計画の予算	〇〇地区の範囲内とする。	
6. 計画の実施	〇〇地区の範囲内とする。	
7. 計画の評価	〇〇地区の範囲内とする。	
8. 計画の修正	〇〇地区の範囲内とする。	
9. 計画の廃止	〇〇地区の範囲内とする。	
10. 計画のその他の事項	〇〇地区の範囲内とする。	

[illegible]

产品名称	产品描述	产品规格	产品价格
产品名称1	产品描述1	规格1	价格1
产品名称2	产品描述2	规格2	价格2
产品名称3	产品描述3	规格3	价格3
产品名称4	产品描述4	规格4	价格4
产品名称5	产品描述5	规格5	价格5
产品名称6	产品描述6	规格6	价格6
产品名称7	产品描述7	规格7	价格7
产品名称8	产品描述8	规格8	价格8
产品名称9	产品描述9	规格9	价格9
产品名称10	产品描述10	规格10	价格10

- 品質・障害対応における関係者の目線合わせ
- 定義そのもの < “定期的な会話” を通じた共通認識の醸成

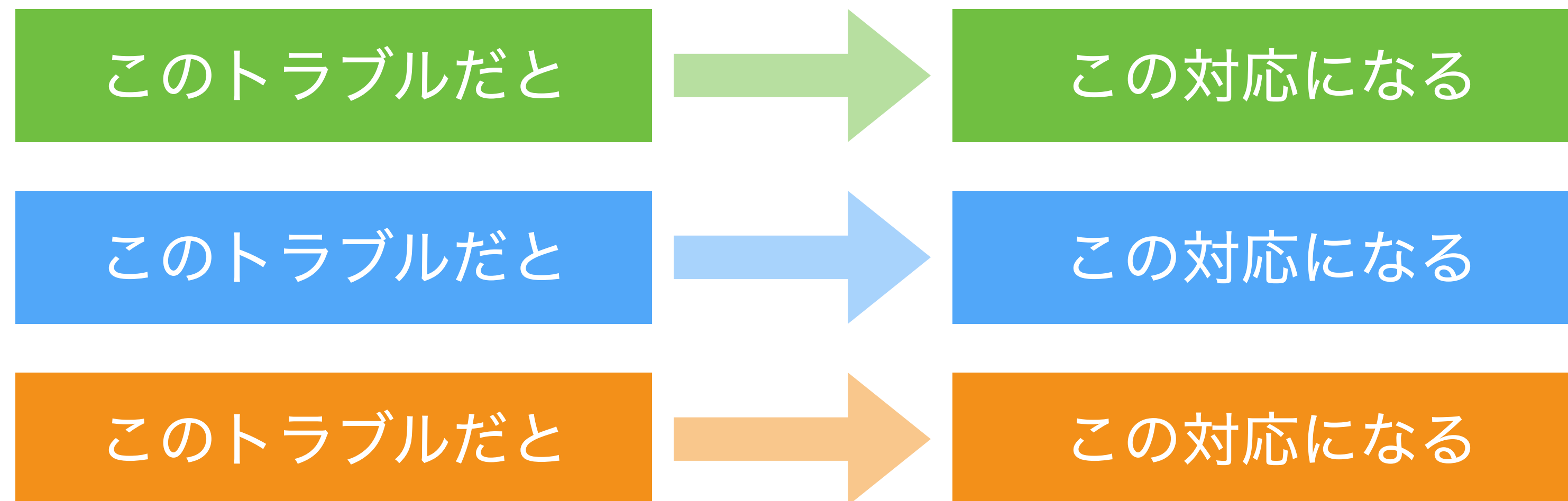
どのくらい“やばい”レベルか



参照 『サービスレベル：設計と運用のプラクティス』 <http://yuzutas0.hatenablog.com/entry/2017/05/23/073000>

「これを決めて何の意味があるの？」

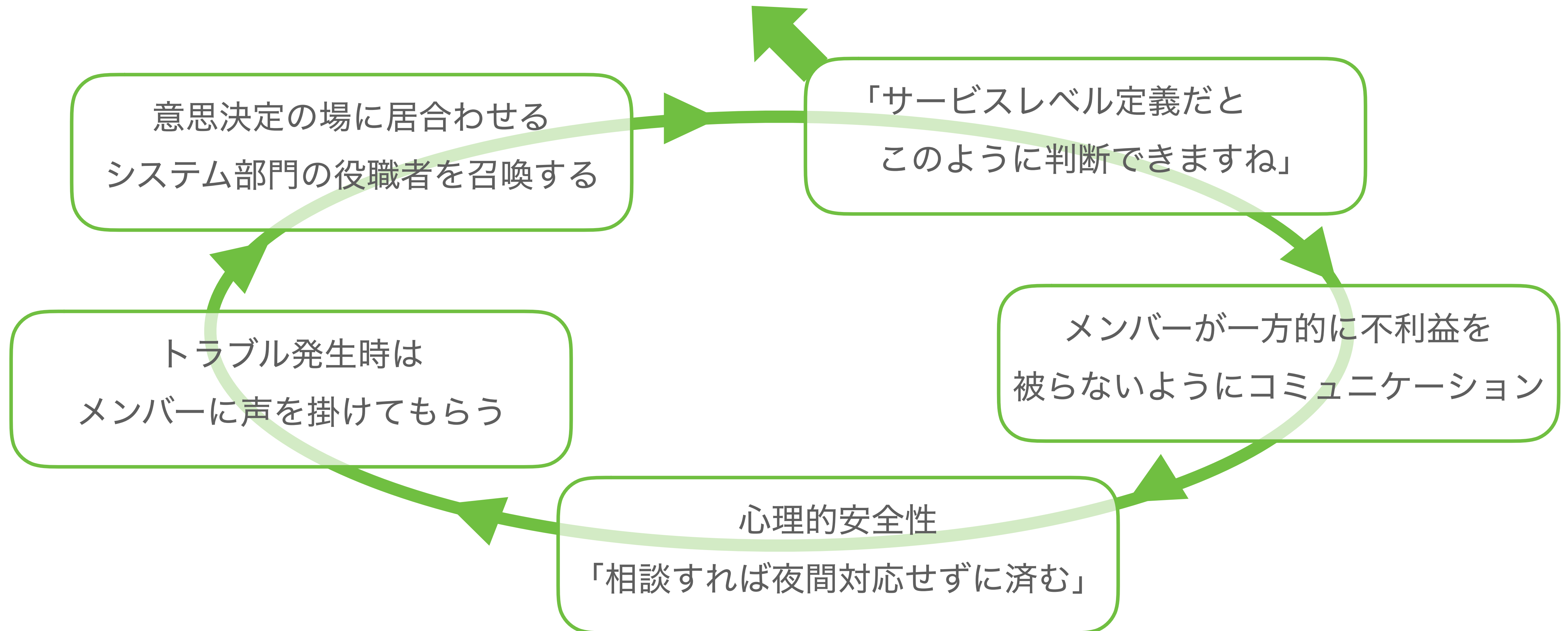
導入時には過去事例を踏まえたシミュレーションで説明



冷静に話すと「このレベルの不具合で」「夜通しの対応をして」
「二次災害が起きたこと」の残念さが浮き彫りになる（判断基準の必要性）

感情や焦りで意思決定がなされがち

サービスレベルを判断の寄る辺として少しずつ定着させる



当初は考慮外だった新観点が出る

形骸化・硬直化を防ぐための運用設計

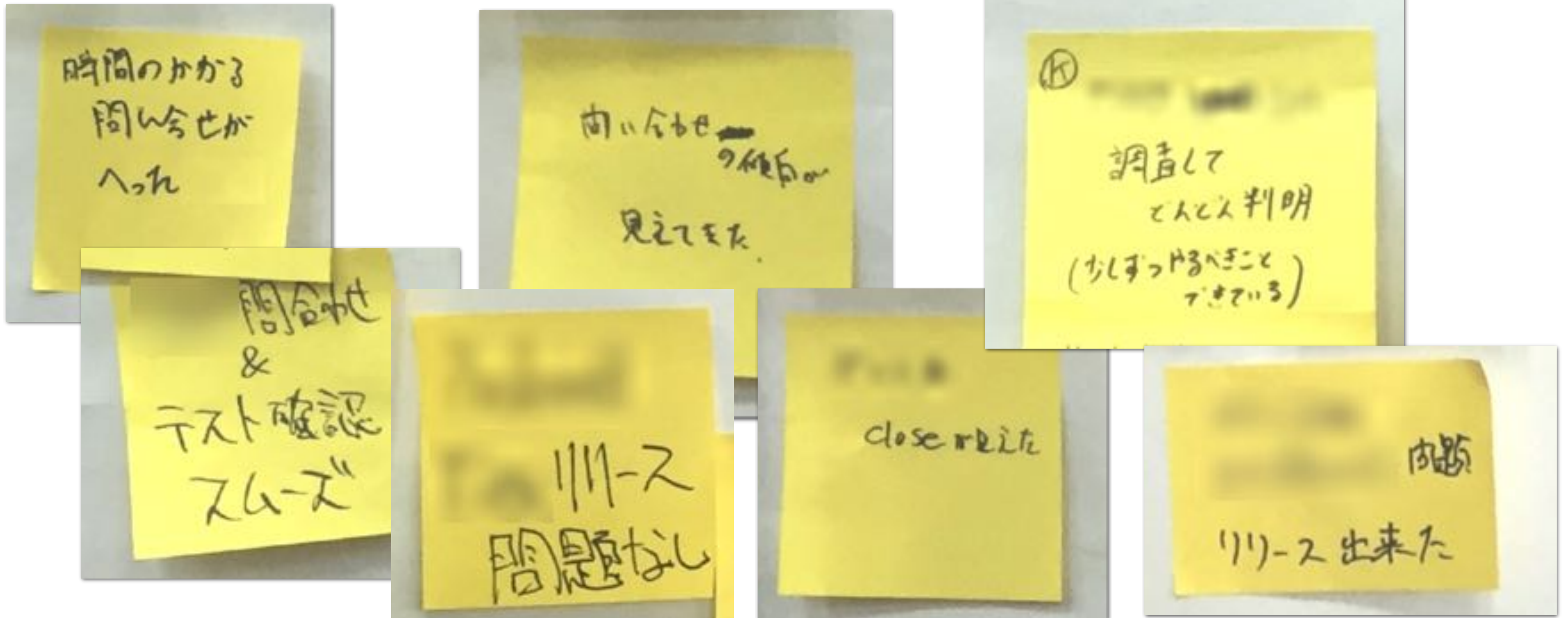
注意点

SLA・OLAは1度決めたら終わりではなく、定期的に振り返り・改訂を行う

- 初期バージョンは、運用実績を踏まえて観点を抽出したものである。
- 最終的には事業MPの判断に基づいて行動する
- MP判断をSLA・OLAに反映し、関係者が同じ基準で会話できるようにする

- 月次の振り返りで定義自体を加筆・修正
- 過大・過少なオペレーションレベルで対応 → 次から適切なレベルとする会話
- アップデートしやすいようにコラボレーションツール（Confluence）に記載

適切なレベルで品質に対峙する



アジェンダ

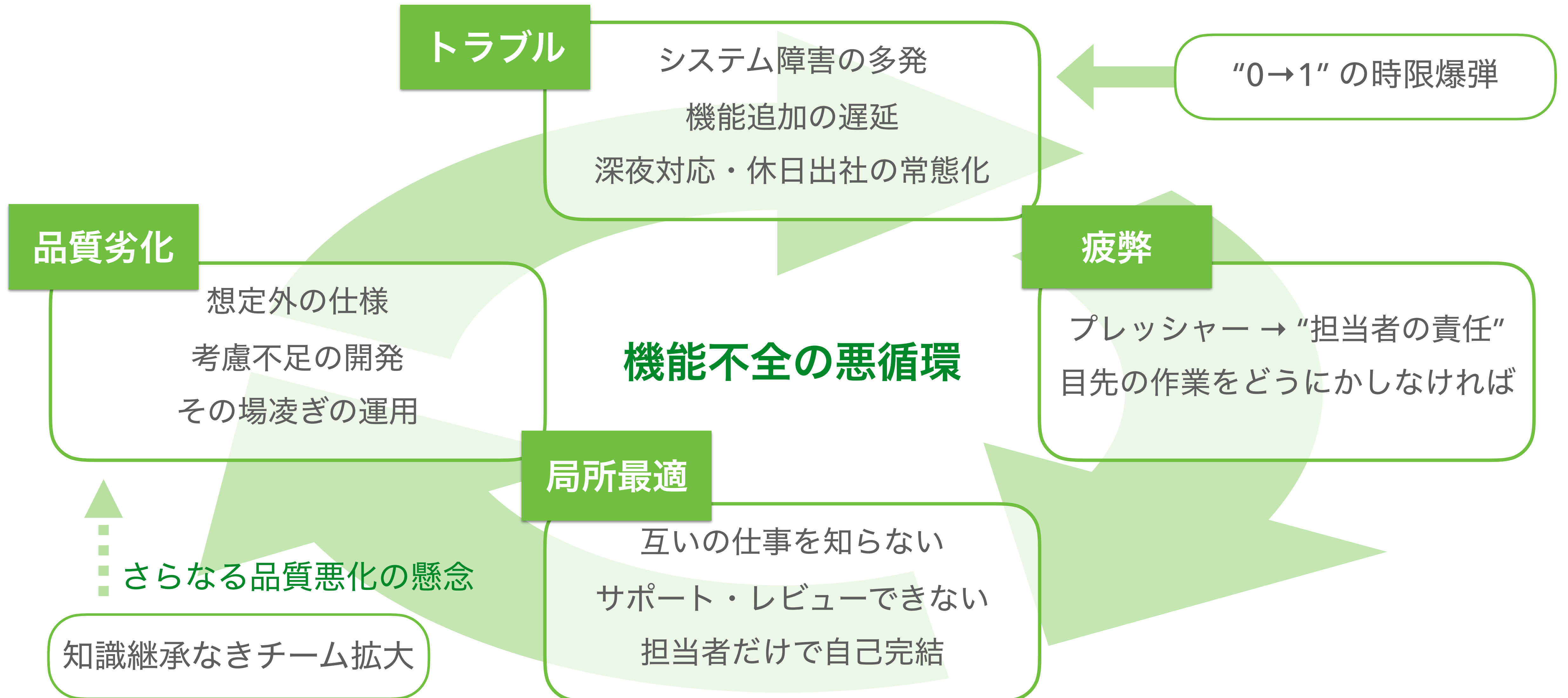
1. 背景・課題

2. 改善活動

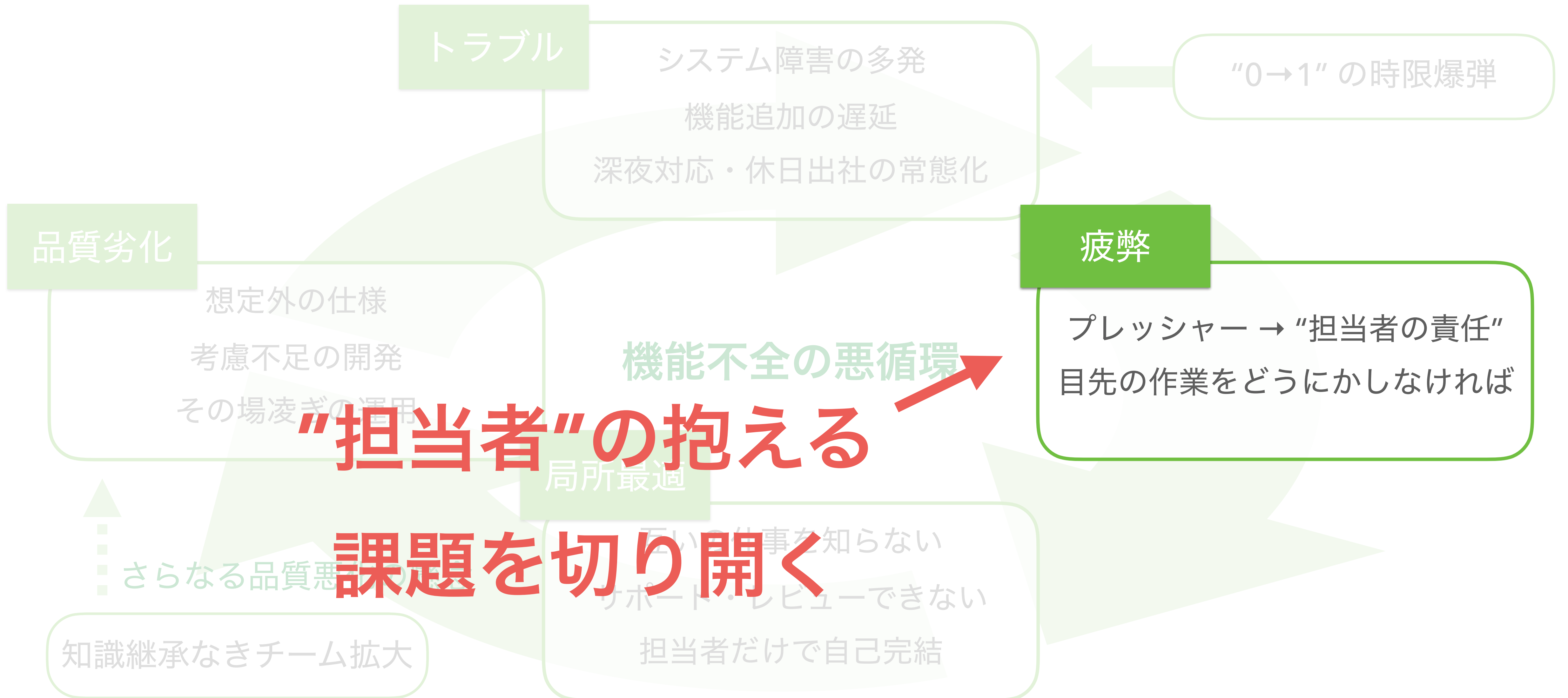
①サービスレベル ②セレモニー ③バリューストリーム ④ドキュメント

3. 振り返り

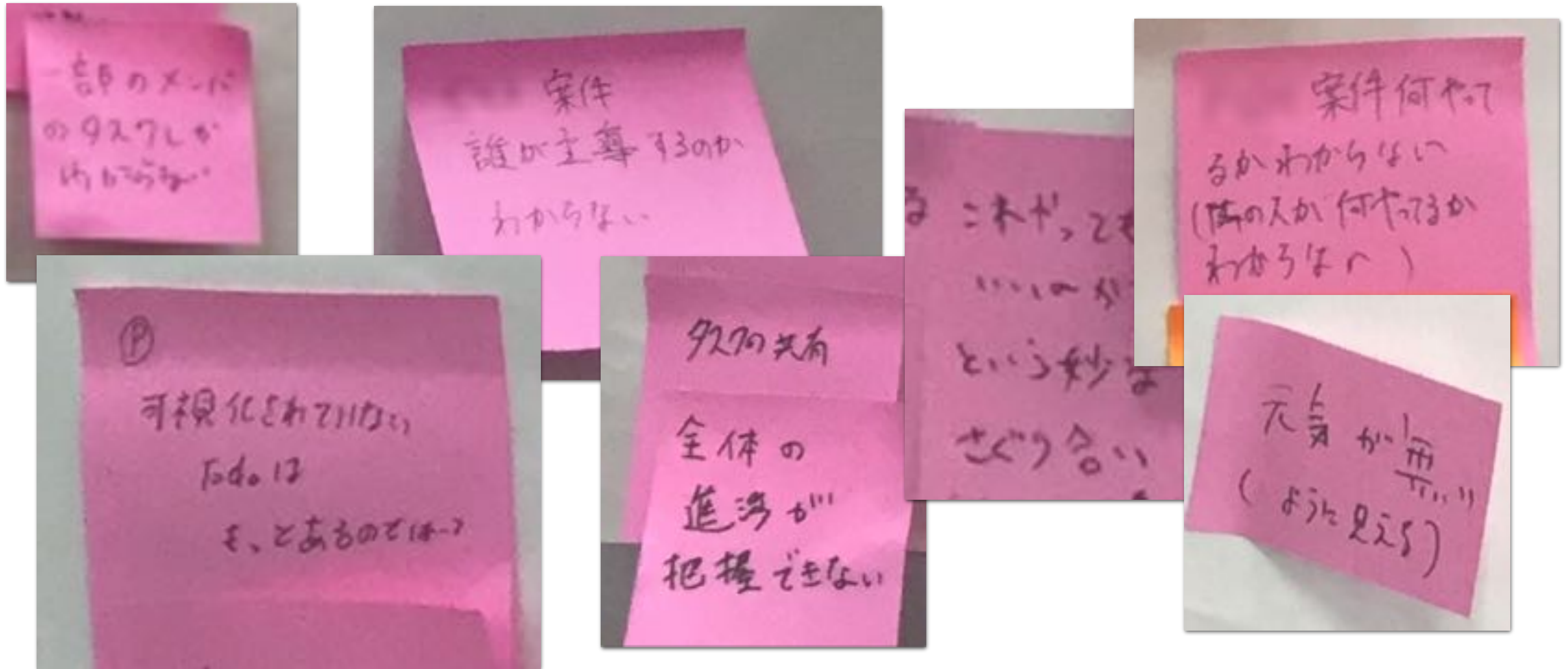
Dev&Opsで生じた痛み



ボトルネック



“担当者”だけが課題に対峙する

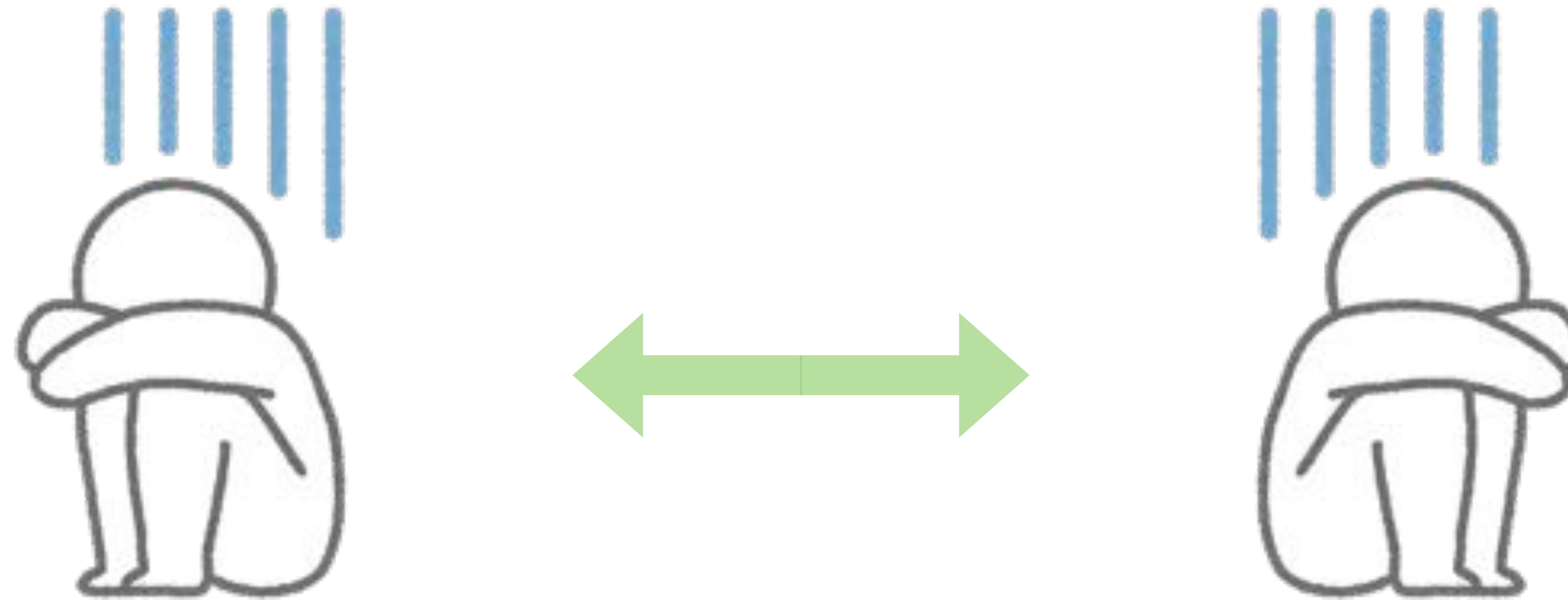




1on1にてヒアリング

<https://www.pexels.com/photo/adult-beard-beverage-blur-590516/>

継続的な課題の検知ができていない



- メンバーごとに異なる課題意識を抱えている
- その“問題”をお互いに発信しあえていない
- チームとして“問題”の検知・解決ができていない

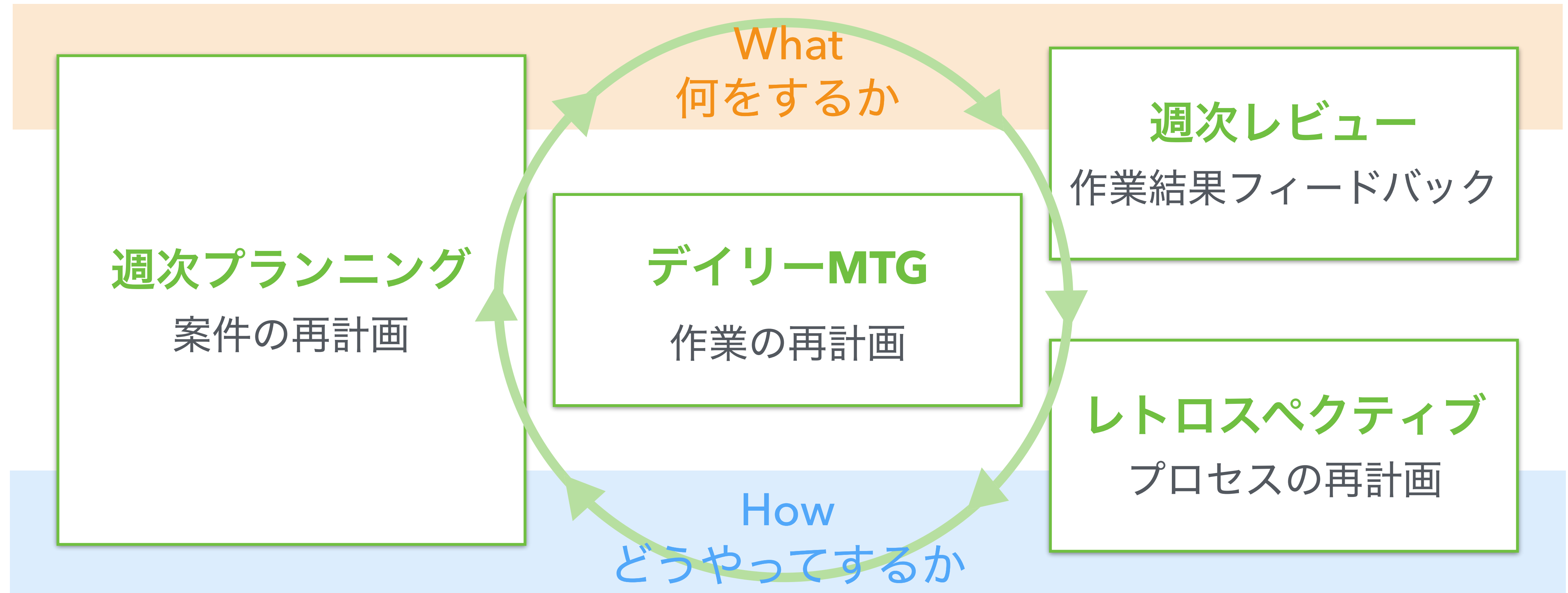
優先順位に応じて作業できていない

[illegible]

			高			
			高			
			低			
			高			
			低			
			低			

- 「高い優先度」だらけの計画
- 「高い優先度」だらけの割り込み作業
- “担当者”が稼働を増やすことで計画に実態を合わせようとした

スプリントセレモニーを開催

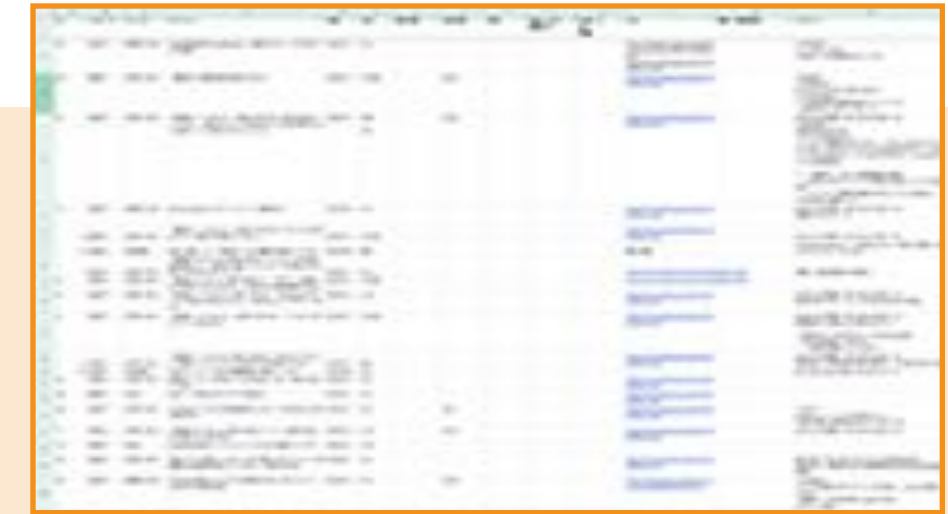


チーム内部の会話を活性化 → 透明性を保つ → 検査・適応のサイクルを回す

セレモニーに伴うアイテムの整備

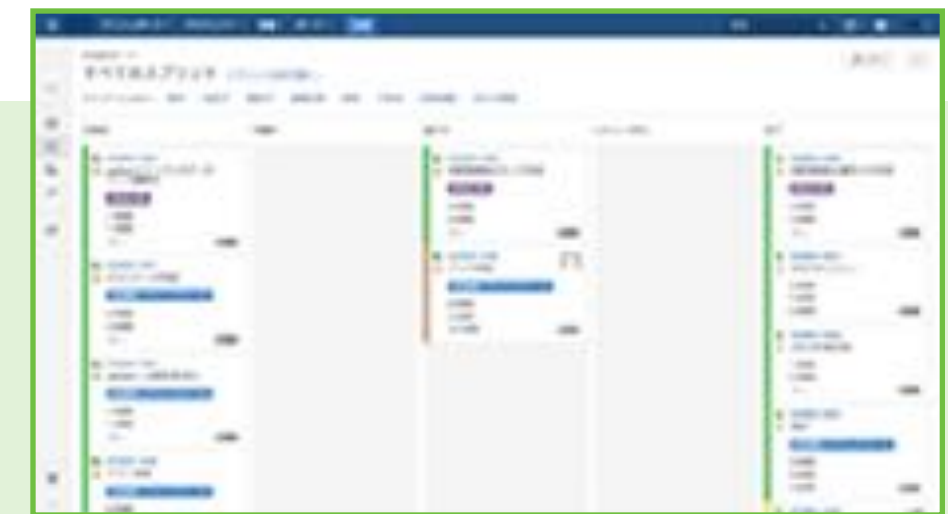
プロダクトバックログ

限られた人員で優先順位に応じて作業するキュー



スプリントバックログ

現状の作業進捗を可視化するカンバン



障害物リスト

レトロスペクティブで可視化された課題



KPTによる振り返りから着手



Keep

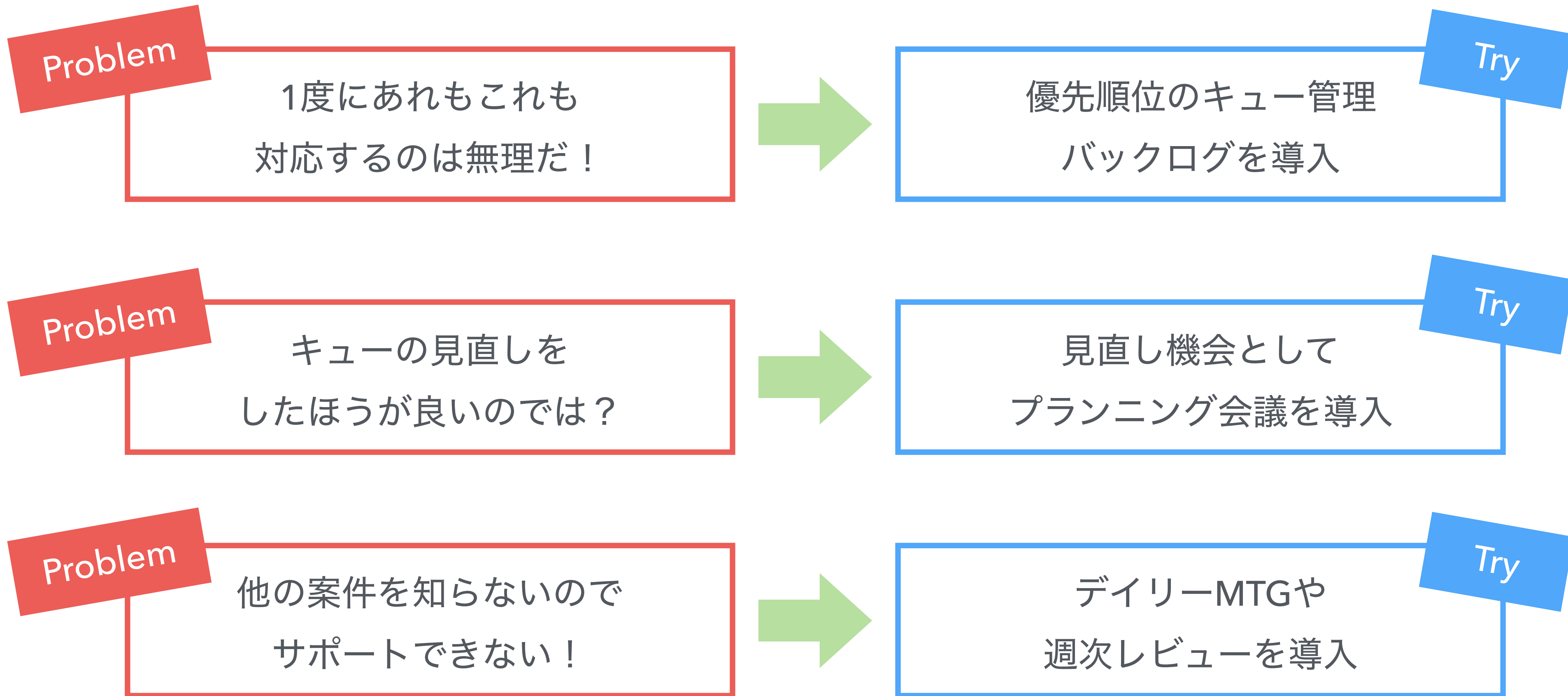
厳しい状況だからこそ発揮された
良い点をしっかりと称え合った*

Problem

（その上で）各メンバーが
感じている課題を洗い出した

*例 「あの鬼電の中でも手順書を作ってレビューできた！二次災害が起きないように動けたのは良かった！」

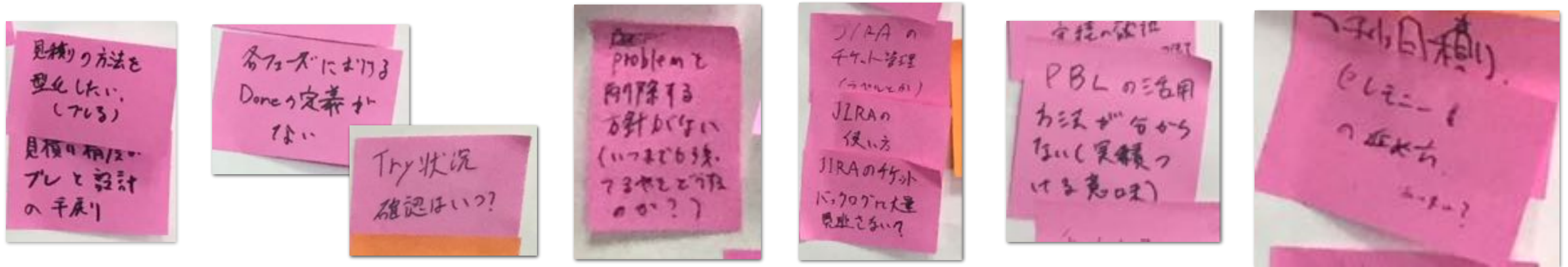
Tryとしてプラクティスを徐々に導入



現場が無理なく受け入れられる順番でアジャイル手法を適用（いきなりスクラムだと身構えられる）

「やり方がコロコロ変わる」問題

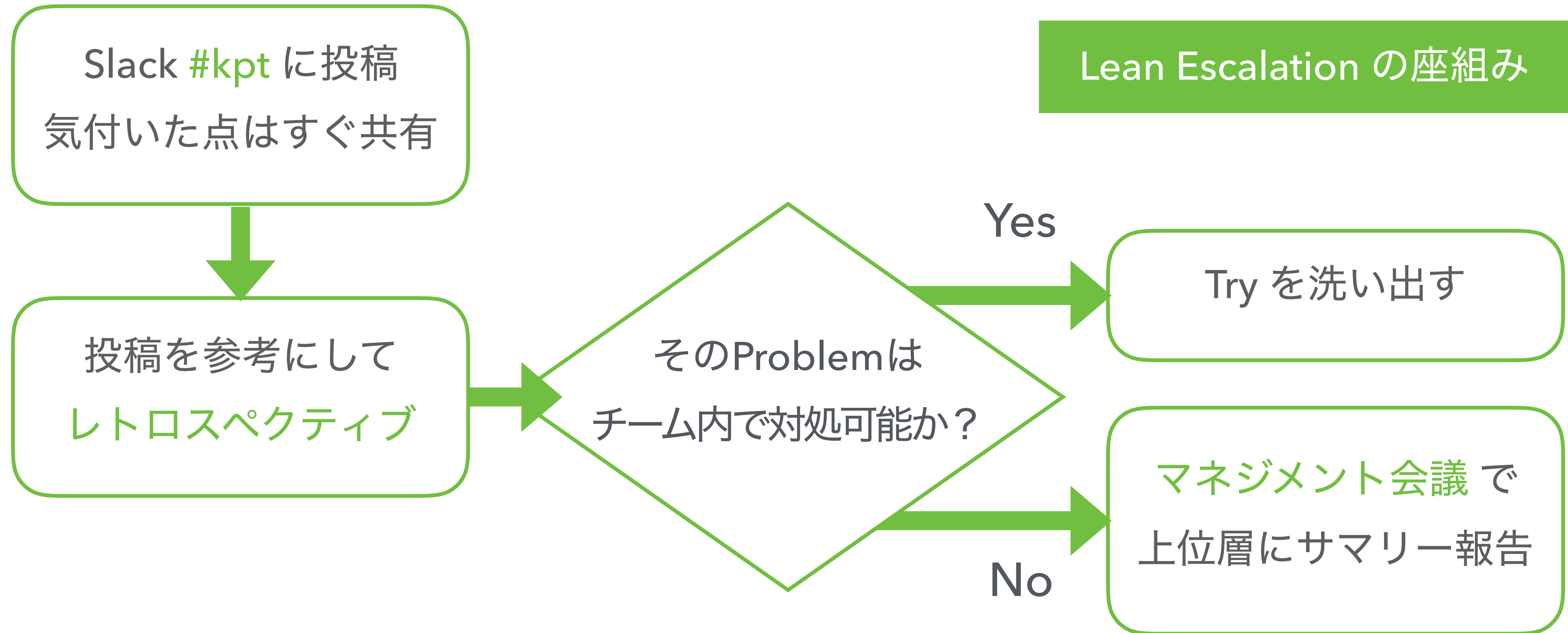
プラクティスの適用が急すぎると、現場のメンバーが咀嚼できない



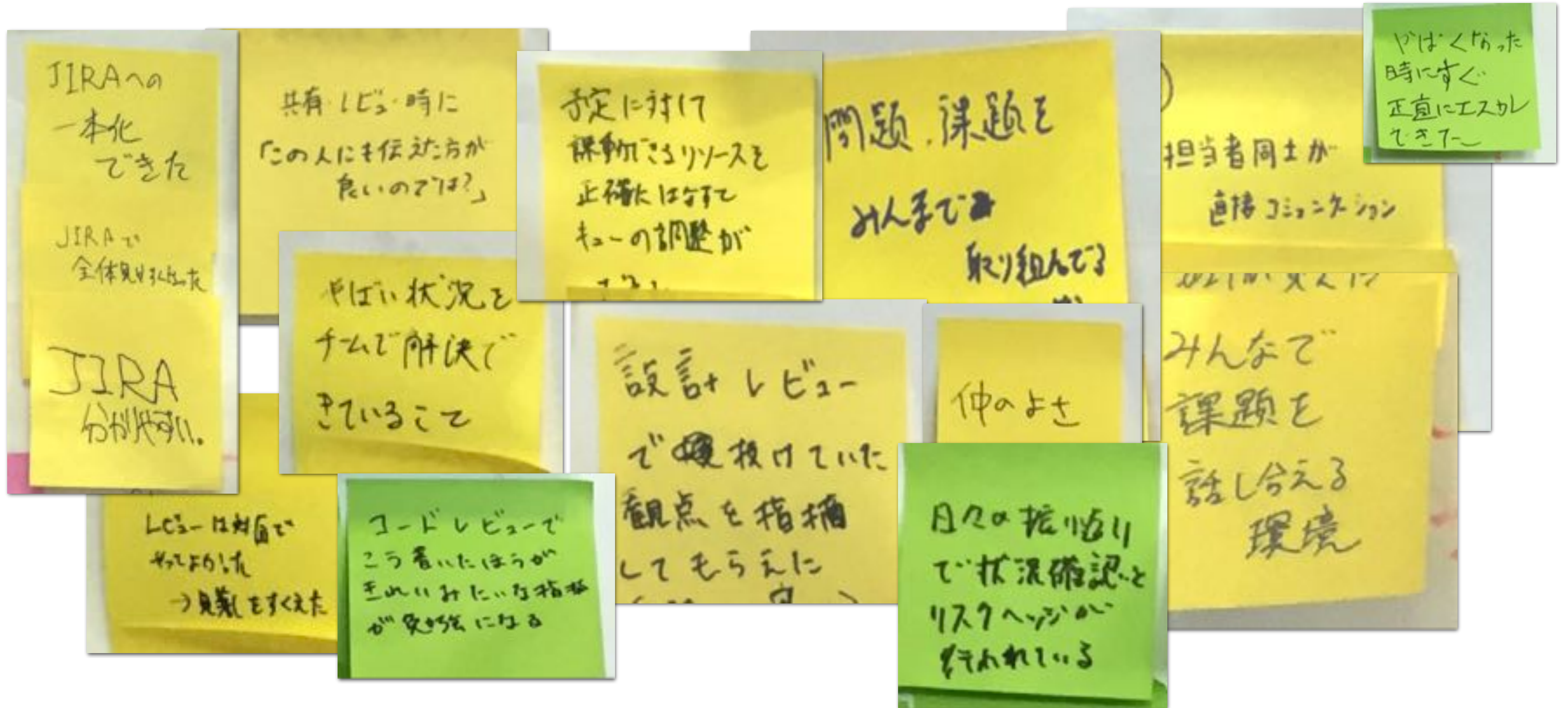
- 納得できる範囲を聞き出す → 全員が納得できる部分から導入
- このチームにはまだ早すぎた → 「やっぱりやめましょう」で切り戻し
- 中途半端な適用ゆえのProblemが挙がる → Tryでさらに本格適用
- 「根本の考え方を知りたい」という意見 → 希望者に対して参考図書の案内

“現場の振り返り”を基点に上位層連携

Lean Escalation の座組み



チームとして課題と対峙する



アジェンダ

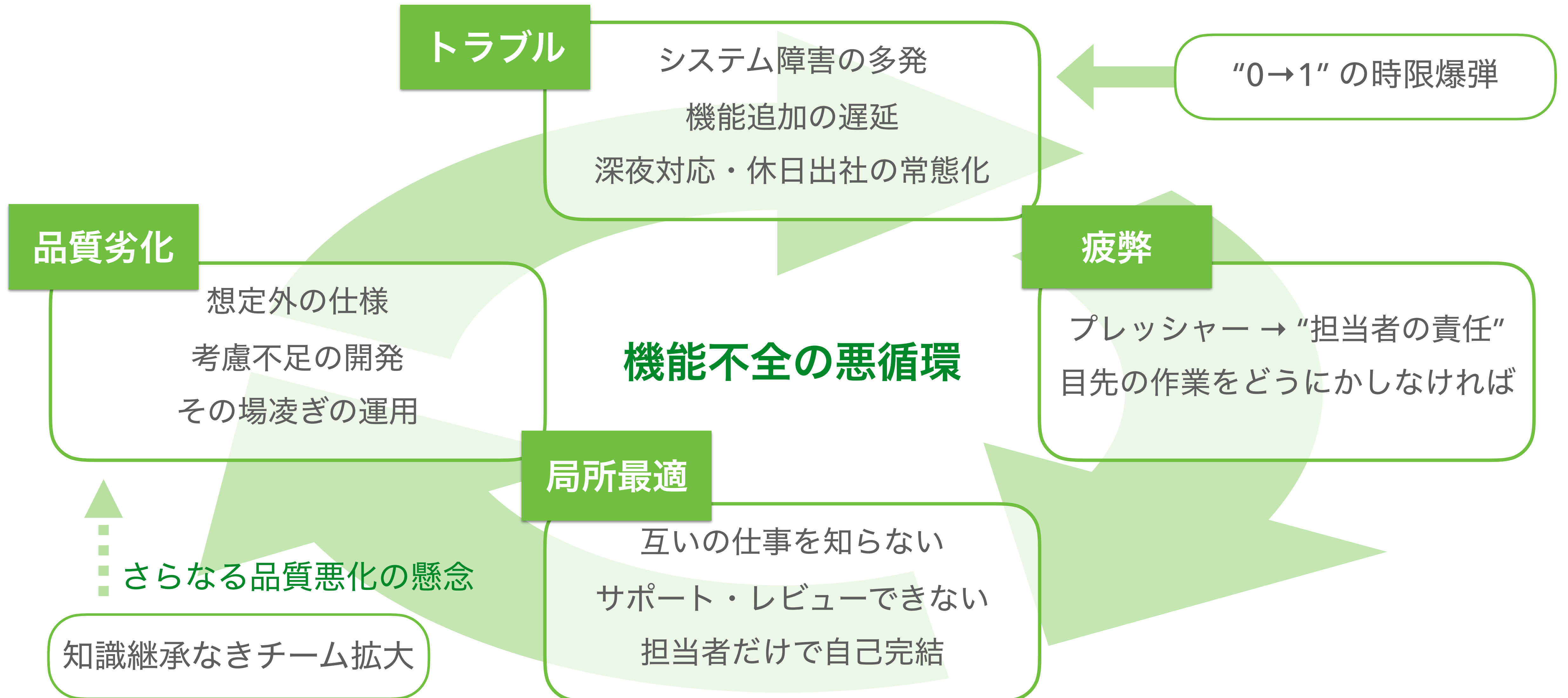
1. 背景・課題

2. 改善活動

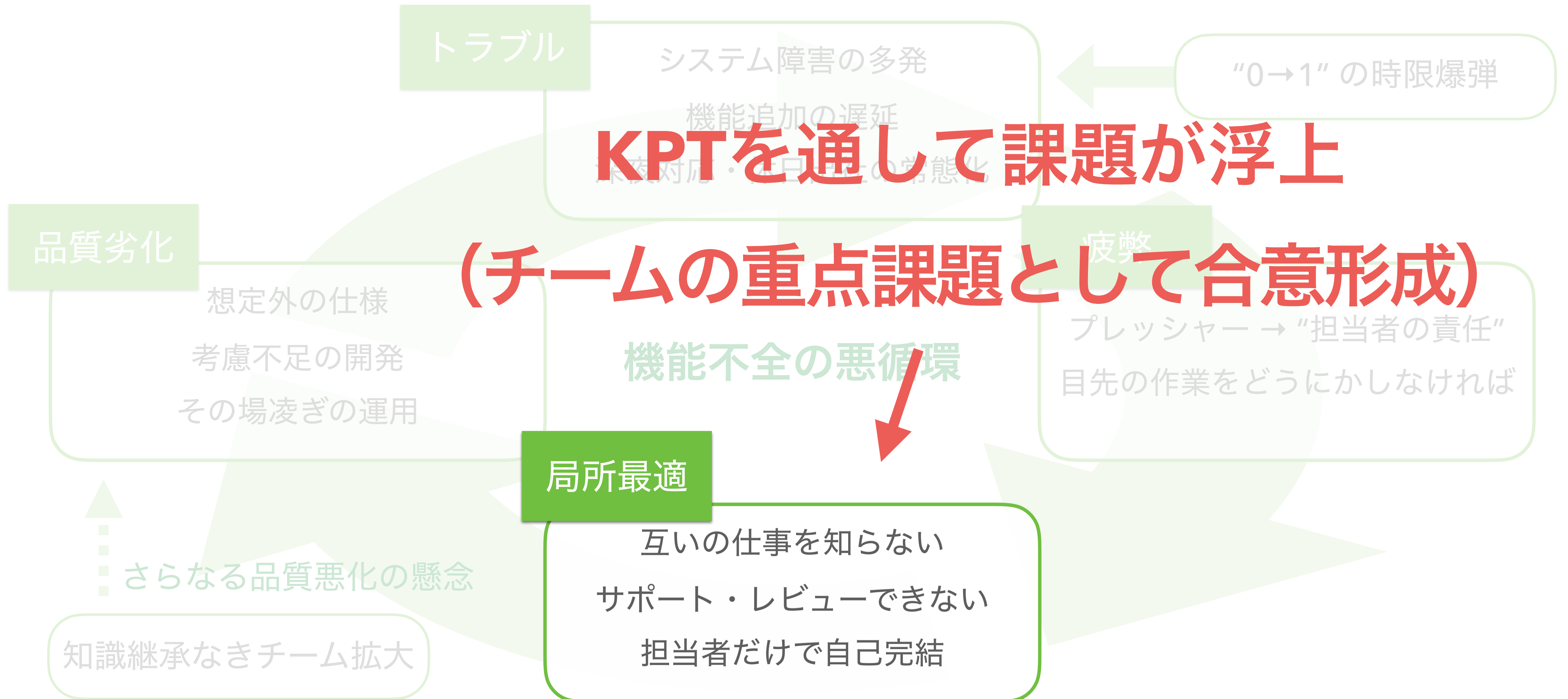
①サービスレベル ②セレモニー ③バリューストリーム ④ドキュメント

3. 振り返り

Dev&Opsで生じた痛み



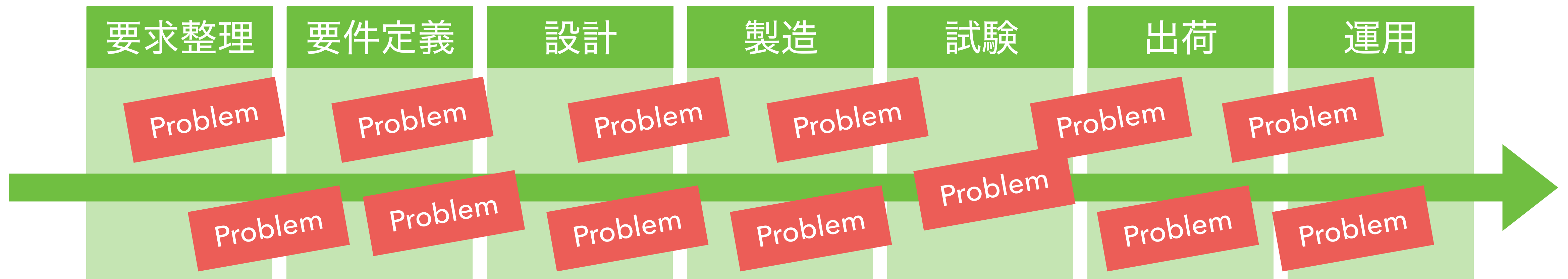
ボトルネック



局所最適の誘因

- 全体像を全く知らない
- 全体像にアクセスできない

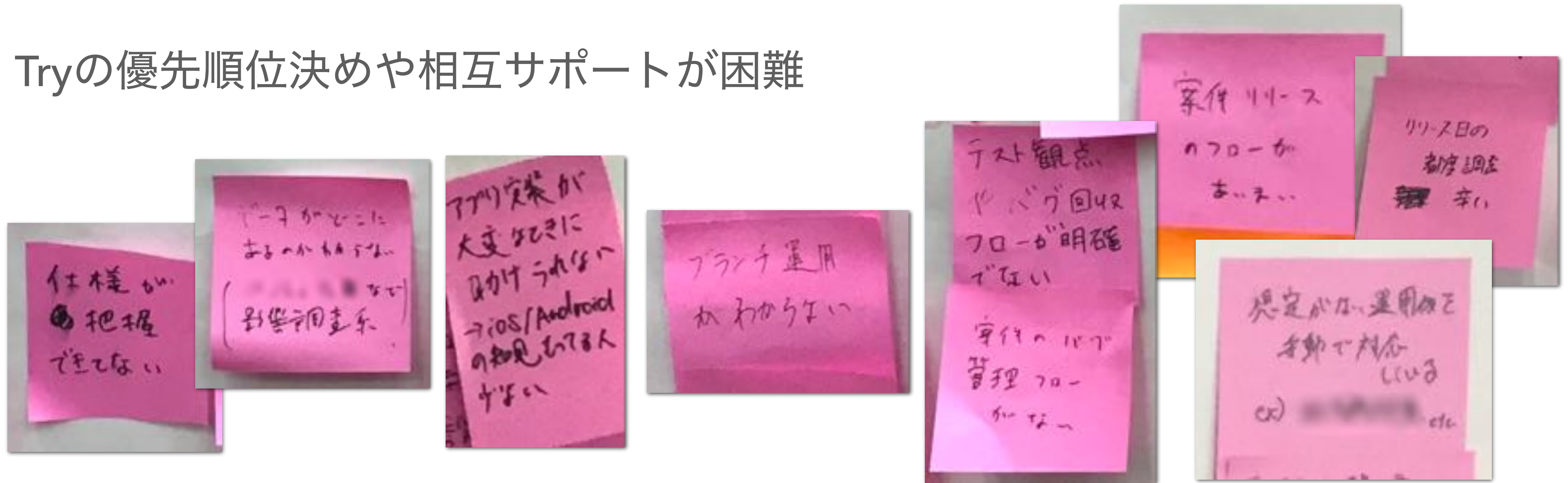
「何となく」の作業だらけ



あらゆる開発・運用業務の、あらゆる工程に
あらゆるProblemがマッピングされた

全体像が不透明

Tryの優先順位決めや相互サポートが困難



- 担当者以外は（担当者自身でさえ）各々の業務フローを把握できない
- 業務内容のどこに、どのようなムダ・ムラ・ムリが生じるか見えない

バリューストリームマップ（VSM）を作成



作業時間・待ち時間や手戻りの実態を洗い出す
ムダ・ムラ・ムリが生じる箇所を特定

各業務ごとに別のVSMを作る



機能追加の開発

システム障害対応



他部署からの
作業依頼対応

カスタマーからの
問い合わせ対応



システム保守業務
例：EOSL対応

システム運用業務
例：キャパシティ計測



見えにくい部分にこそ問題が潜む = 全業務を可視化することが大切

全員参加型ワークショップから始める

1. 納得してもらう

- 問題意識・打ち手を説明
- 社内でのVSM活用事例を紹介



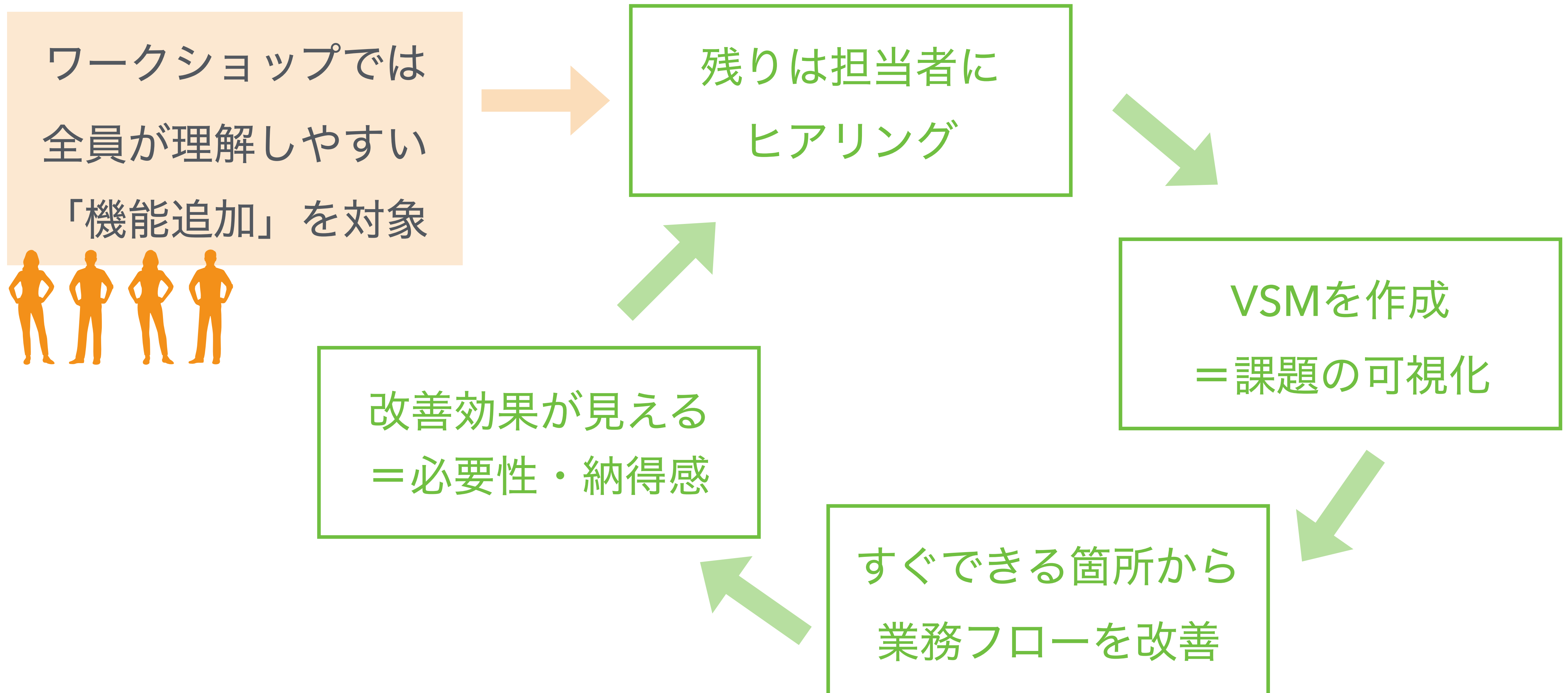
実物は汚いです……

2. 全員でVSMを作成

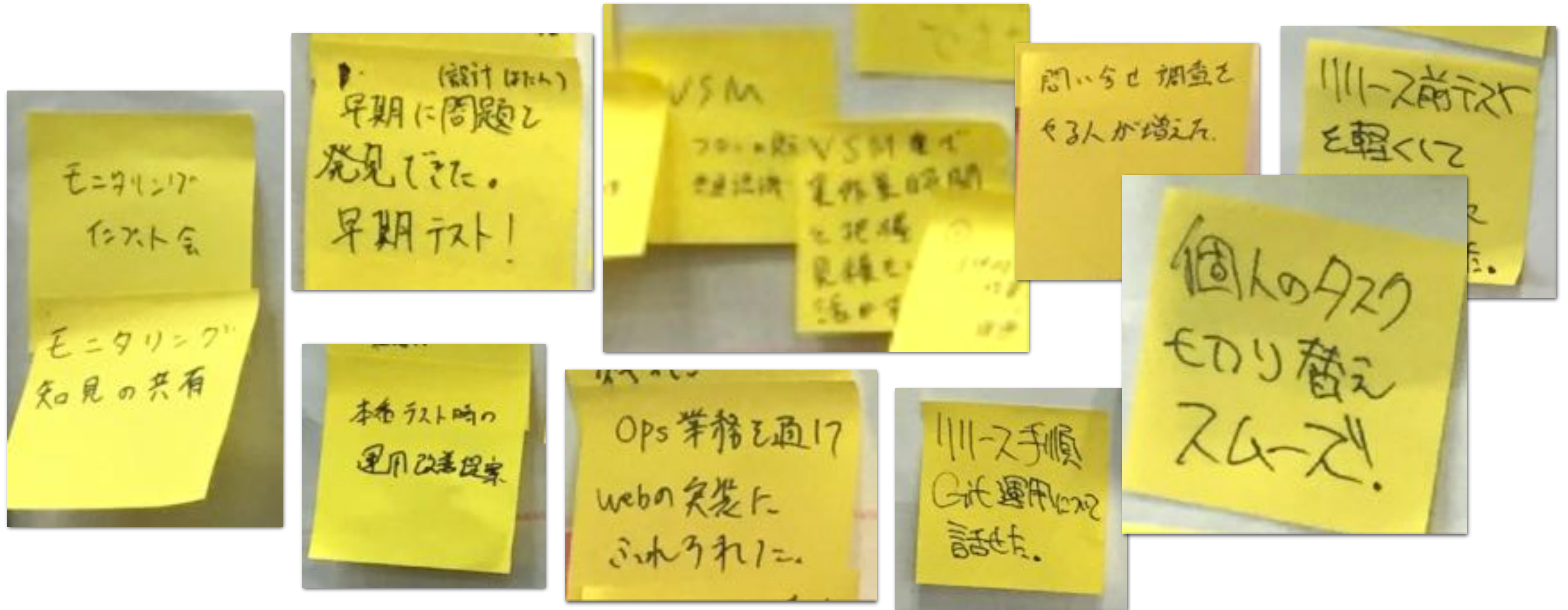
- 各担当者が説明しながら業務の流れを書く
- “ムダ・ムラ・ムリ” や “メンバー間の認識の差分” を可視化
- まさに進行中だった案件のリスクをその場で検知できた（成功体験）



「拘束時間が長すぎる」問題



業務の全体像を知る



アジェンダ

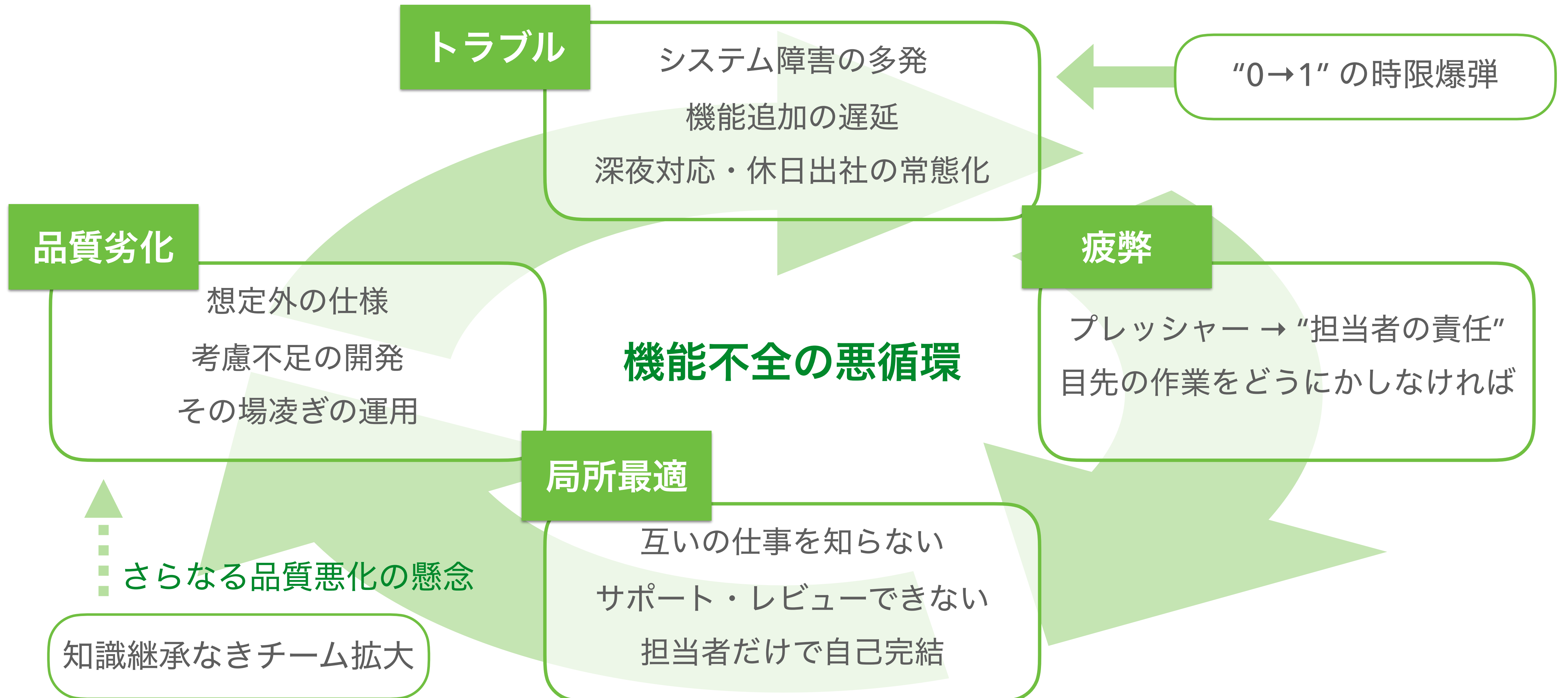
1. 背景・課題

2. 改善活動

①サービスレベル ②セレモニー ③バリューストリーム ④ドキュメント

3. 振り返り

Dev&Opsで生じた痛み



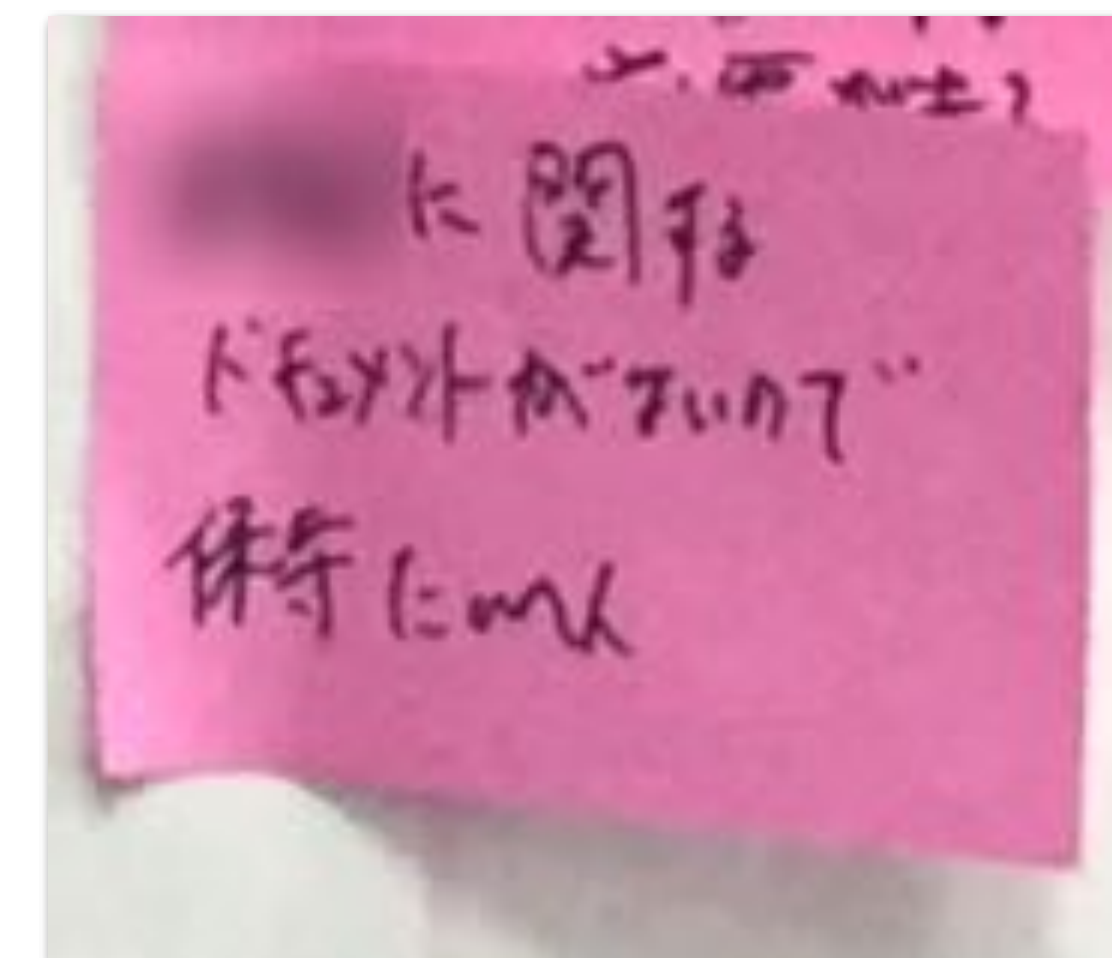
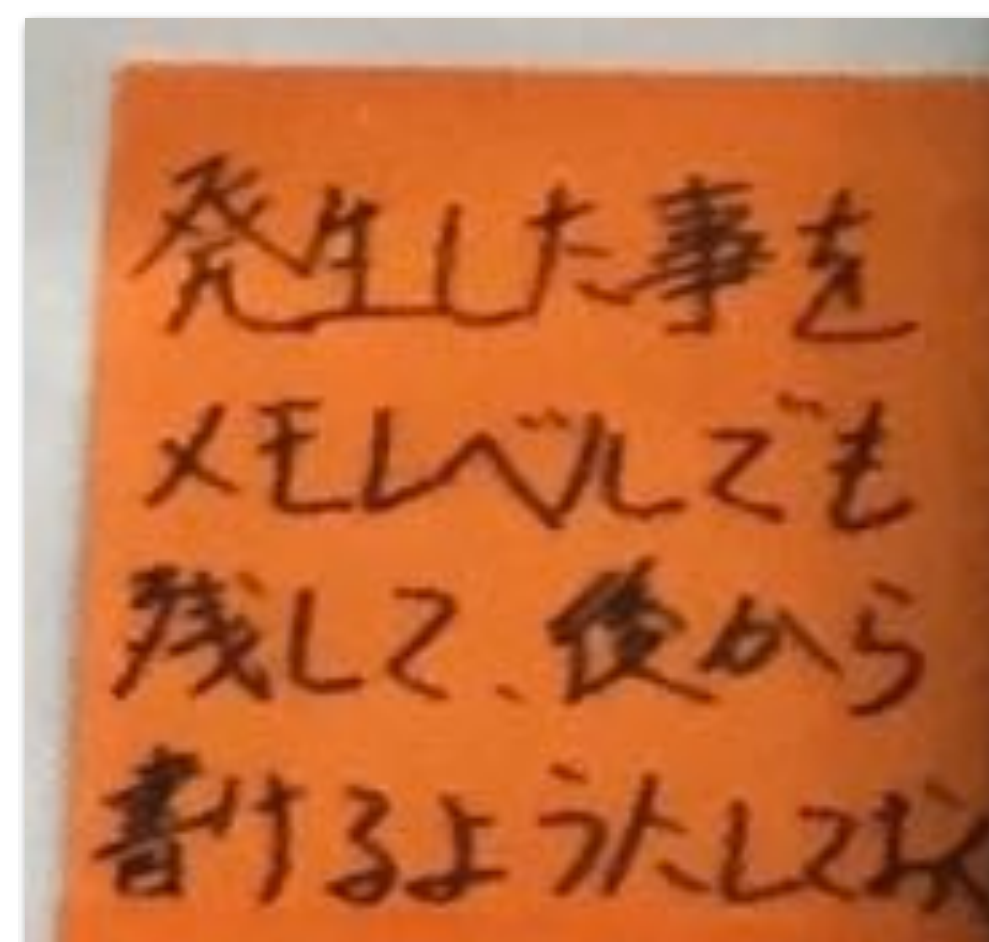
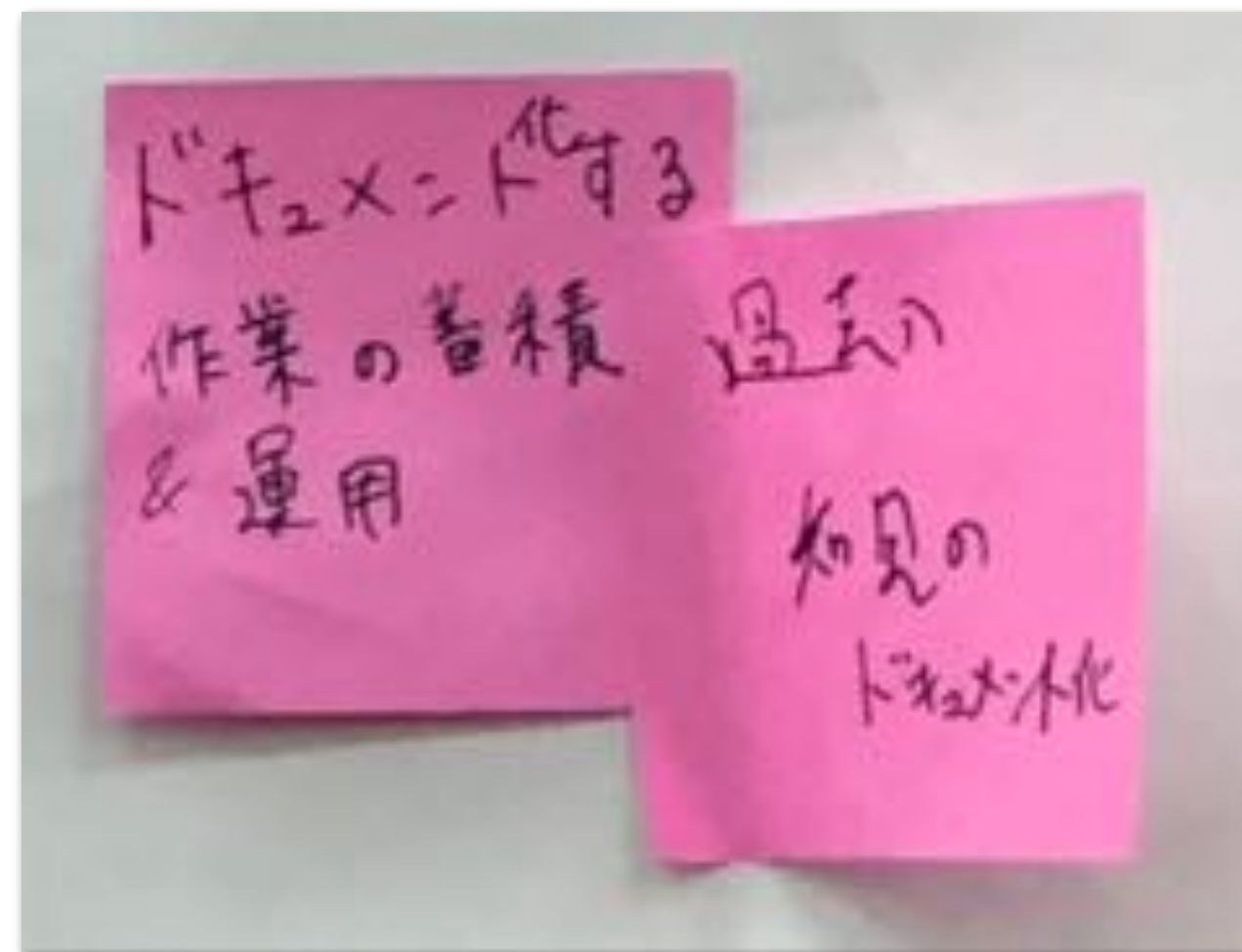
ボトルネック



局所最適の誘因

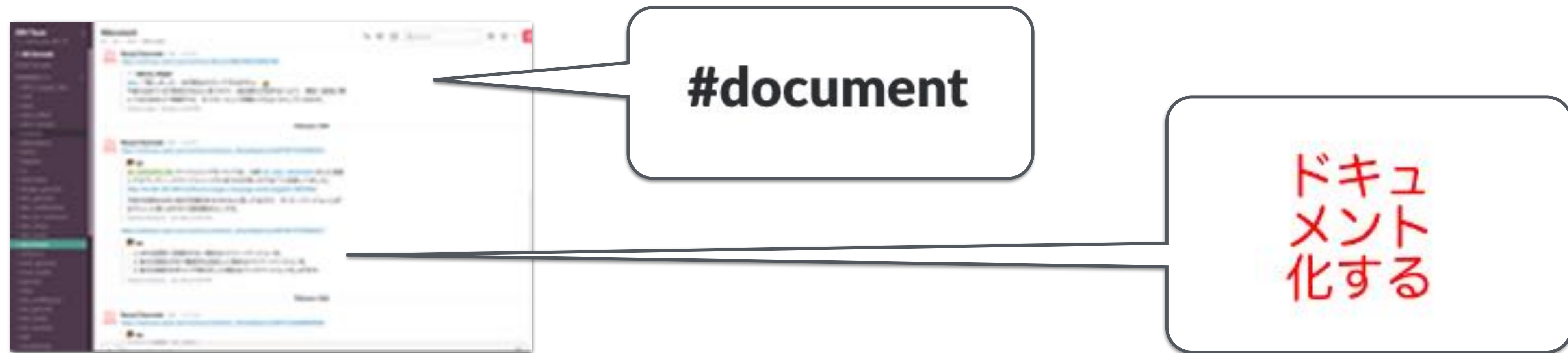
- 全体像を全く知らない
- 全体像にアクセスできない

「どこを見れば分かる？」 問題



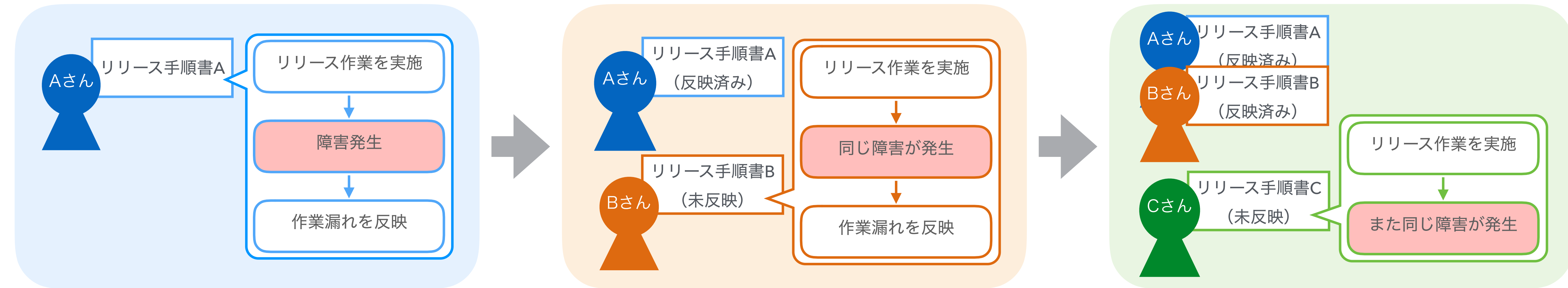
改善活動を通して急激にルール・プロセスが決まっていくが
意思決定のスピードにドキュメント化が間に合わない

チーム全員に推進者として振舞ってもらおう



- ドキュメント化＋更新し続けることの重要性はKPTで強調
- 日々のコミュニケーションで「ぜひ文書化してください」「更新してください」
- 促進案を募る → メンバー主体で促進botやチケット運用の試行錯誤
(案自体は続かなかったがメンバーが自発的に文書化するようになった)

「どれが正しい？」 問題



- メンバーが主体的に文書化 → 別の場所に似たドキュメントが重複

実例: Aさんが障害を受けてリリース手順書を更新 → Bさんは別の手順書で同じミス → Cさんも

- 迷ったときに、どこを見れば (=どこに書けば) 良いか分からない

レガシードキュメント

Write

修正がつい後回し
何を書けばいいか迷う
どこに書けばいいか迷う

Read

見つからない
情報が古い・誤り
似た内容が散在している

まさにレガシーコードと同じ

対象が自然言語（もしくは表・図）になっただけ！



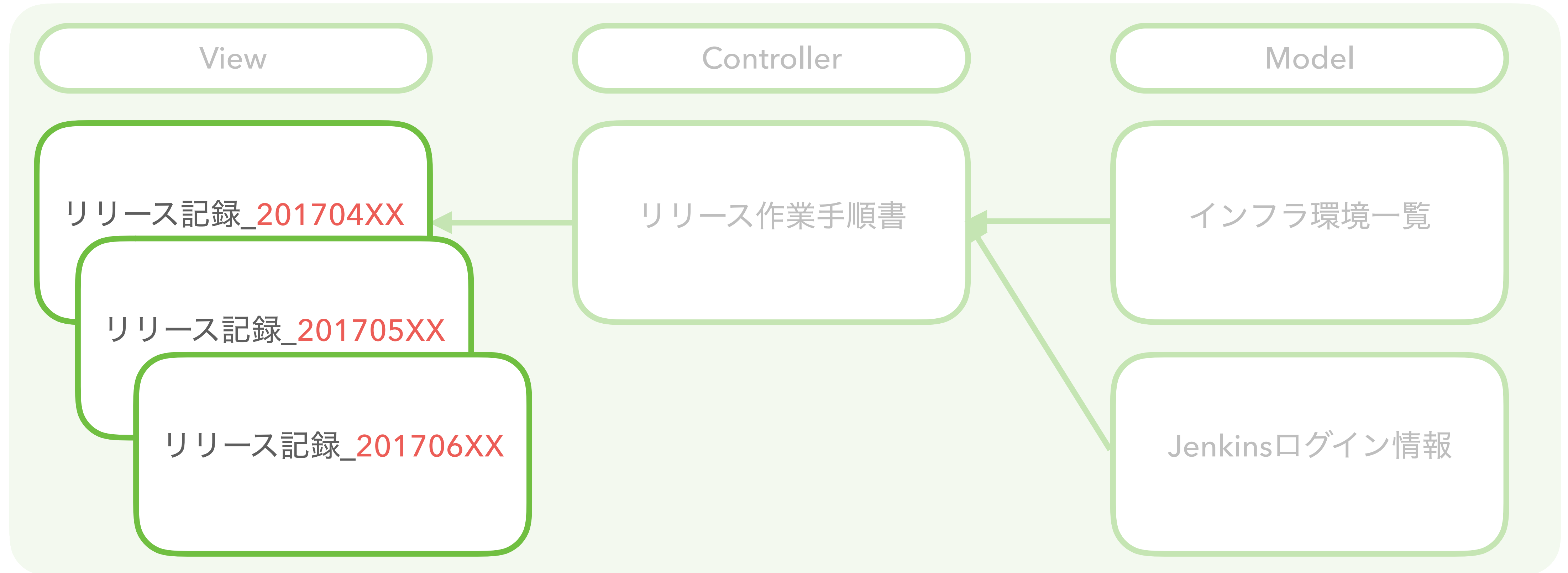
設計・修正のやり方もプログラミングと同じでは？

MVCアーキテクチャの適用



依存関係に基づいたドキュメント設計

Iteratorパターン(GoF)の適用



繰り返しページは一括管理

Document as Code - ドキュメントをソースコードのように扱う

どこにどのドキュメントがあるべきかを

コーディングに例えて会話する



考え方や作業の進め方にも当てはまる



- ボーイスカウト原則：訪れたメンバーが加筆・修正

文書を見る前よりも、見た後のほうが綺麗な状態となっているのが望ましい

- コードレビュー：構成に悩んだら相談

「こういう意図でここに書こうと思う！どう思う？」を相談するSlackチャンネル = レビュー活性化

ドキュメントをコードのように扱う16のパターン (Document Design Pattern)

体制のパターン

- アーキテクトチーム
- コンウェイの法則

文化のパターン

- 技術的負債
- ポートフォリオ
- コーディングを促す
- コードレビュー
- ボーイスカウト
- Rule of Three

構成のパターン

- MVCモデル
- GoFデザインパターン
- 継続的デリバリー /
イテレーション

作業のパターン

- ユースケース分析
- リファクタリング
- ペアプログラミング
- モブプログラミング
- バージョン管理

※体制や文化はデザイン（設計）に影響を与えるのでデザインパターンに含めています

情報のアクセシビリティ

「コンフリクト」
整頓されて
見やすくなってる

Confluence
に情報を集約
UPDATE かんたん

「コンフリクト」
何かはある
カバー・テスト
〜など

ドキュメント残ってる

アジェンダ

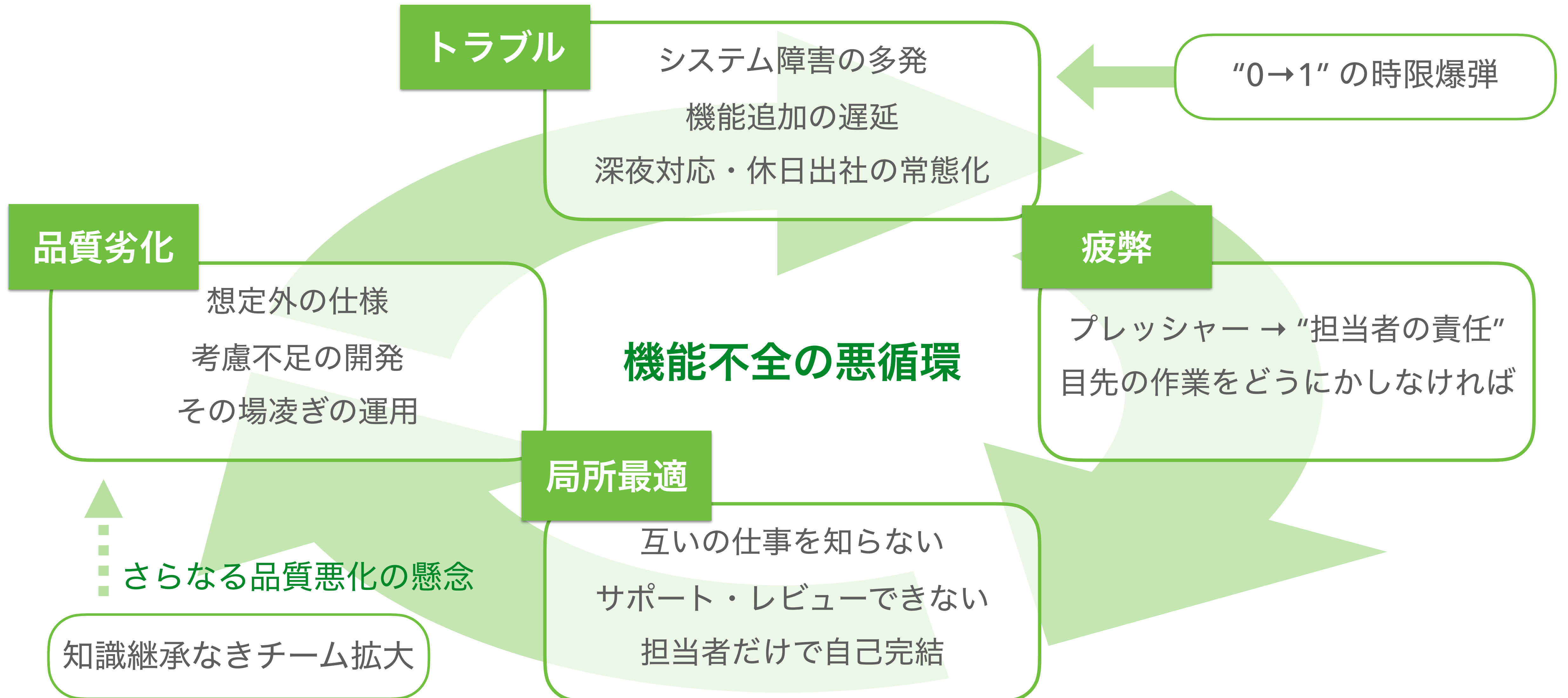
1. 背景・課題

2. 改善活動

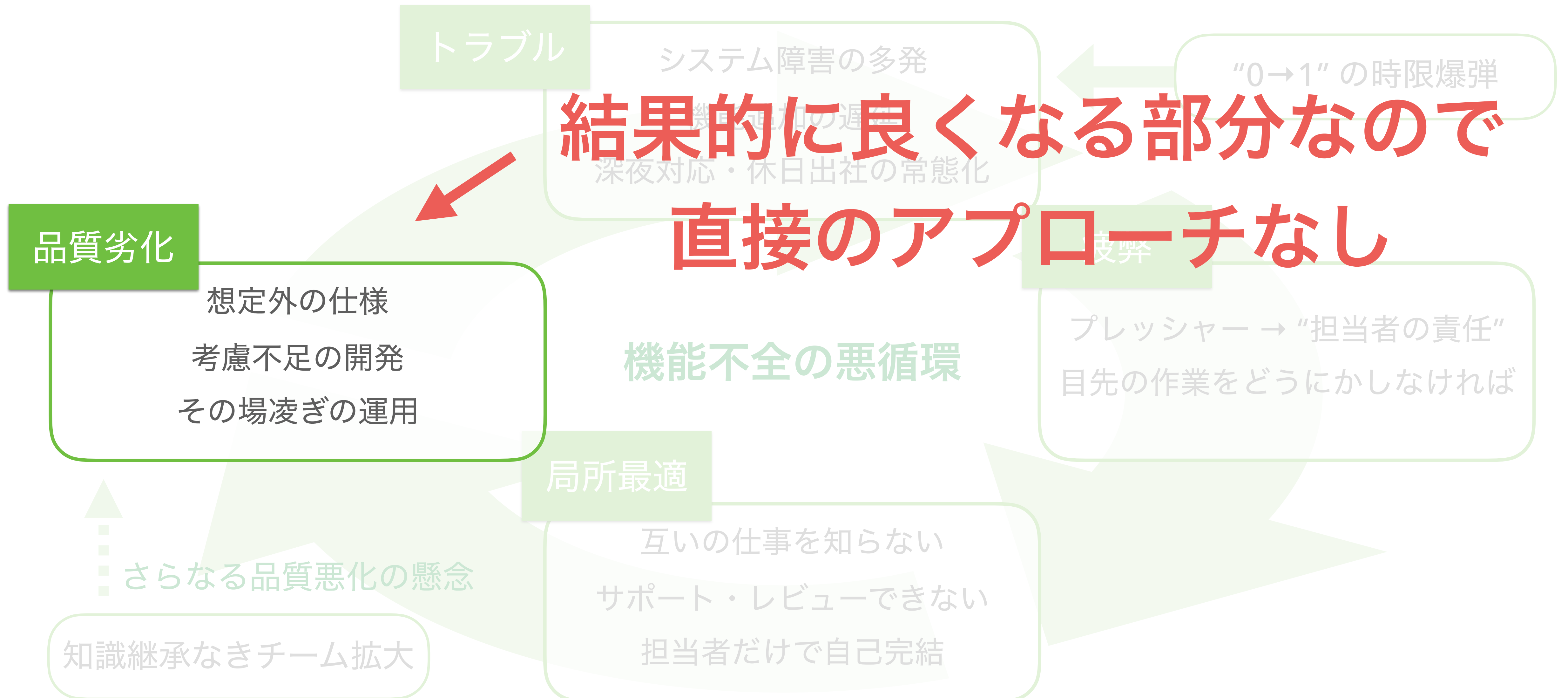
①サービスレベル ②セレモニー ③バリューストリーム ④ドキュメント

3. 振り返り

Dev&Opsで生じた痛み



ボトルネック？



アジェンダ

1. 背景・課題

2. 改善活動

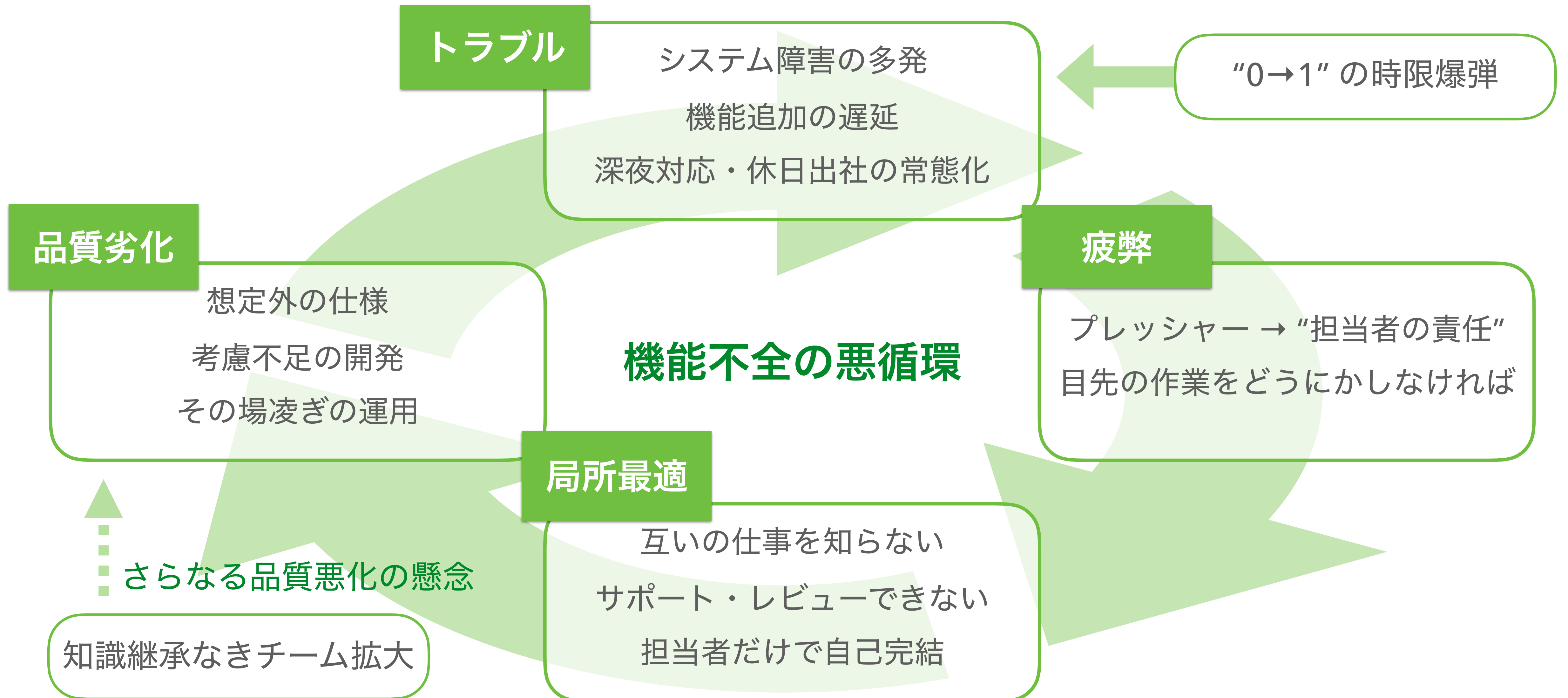
①サービスレベル ②セレモニー ③バリューストリーム ④ドキュメント

3. 振り返り

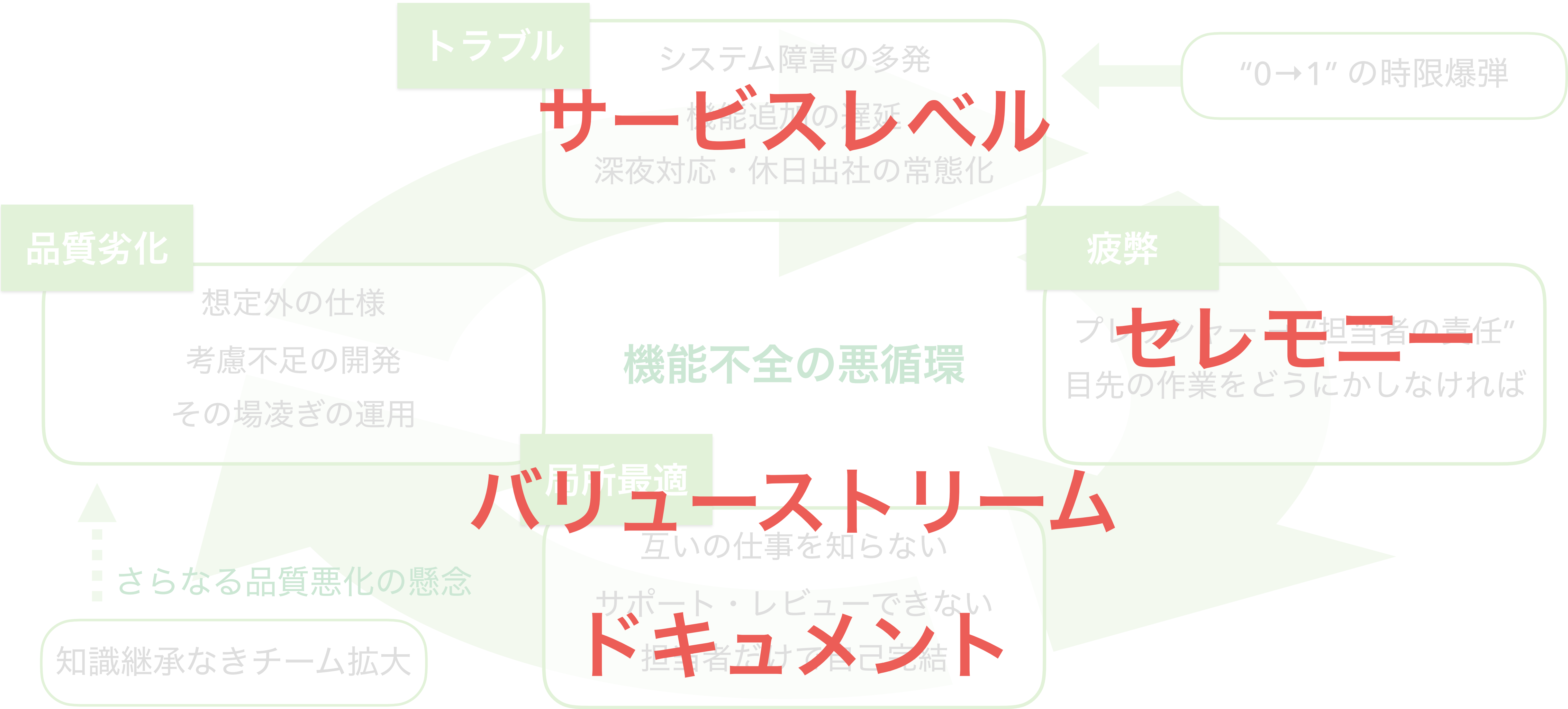
ミッション

- 開発・運用チームを立て直すこと
- さらなる成長の土台を構築すること

Dev&Opsで生じた痛み



痛みを抑制する

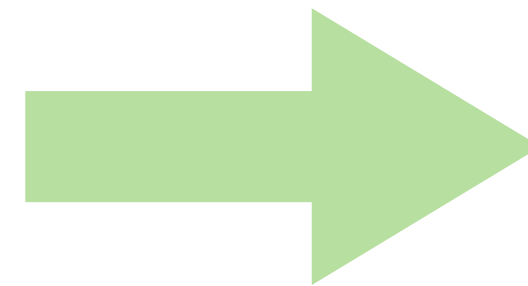


グロースを持続可能にする好循環

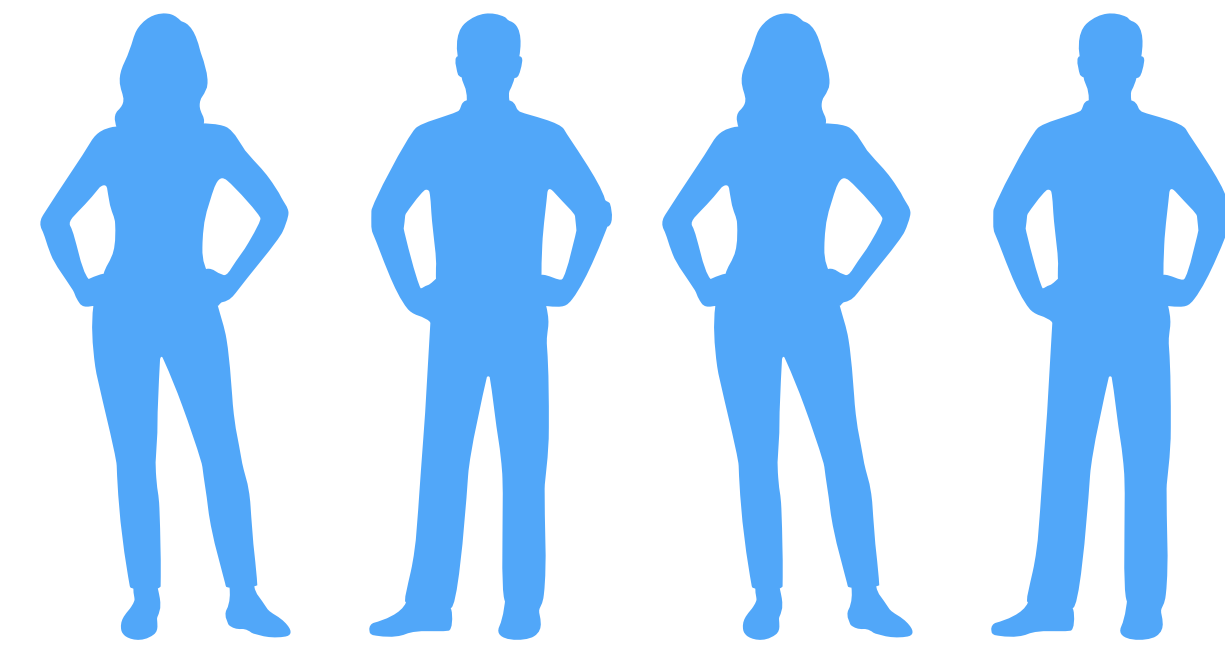


新規参画者の声 @KPTでの会話

4Q Joinメンバー



1Q Joinメンバー

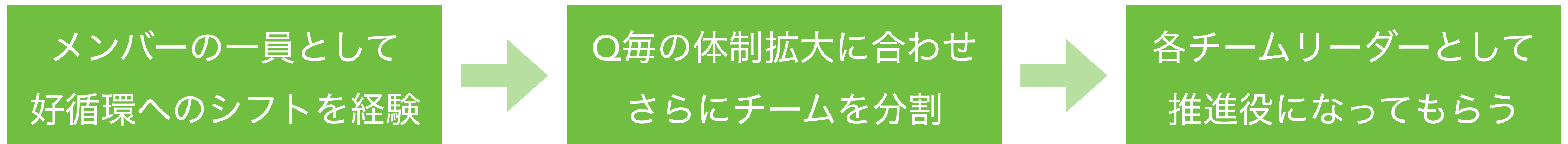


何か聞いたら全部「決まっていない」
こんなに何も決まっていない現場は初めて

ドキュメントやサポートが充実していた
こんなにスムーズに立ち上がった現場は初めて

スケールに向けた推進役

1. 早い段階で優秀な若手社員を何名か他部署から引き抜いた



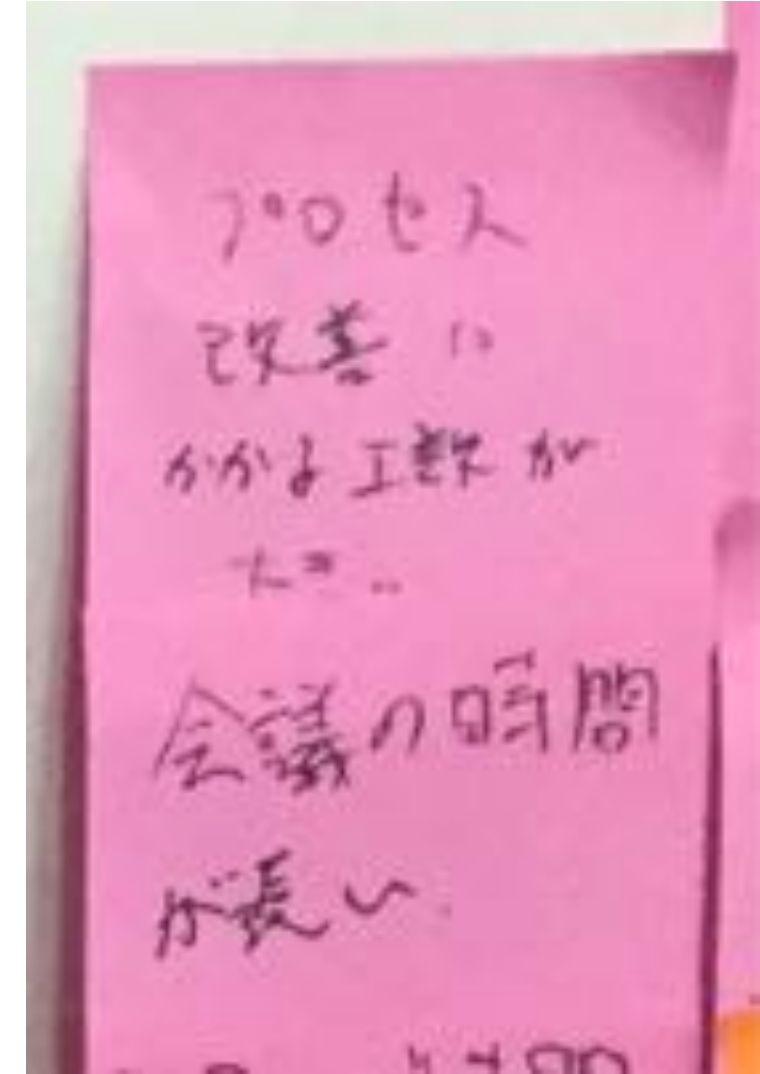
2. 判断基準とした人材要件 = 馬力があること

- 問題提起・改善推進できる自発性 このアサインは推進役を増やすための打ち手なので
- 職能横断志向 トラブルは横断箇所で起きるため（例：iOSとWebAPIを両方見る必要がある）
- 何かしら1つの言語・FWの業務利用経験 1つスキルのベースがあれば他のスキルは学べるため
- ソースコードに触れずくすぶっていること 嫌でも毎日ソースコードを読み書きできる修羅場を提供できるよ！

好循環を回すために費やしたコスト

現場メンバーのコミュニケーションコストを大幅に費やした

- 短期的にはデリバリーにマイナス影響（に見える：前工程の品質担保で結果的にプラス）
- どのみち悪循環を止めるために必要なコストだと上位組織・発表者は認識している

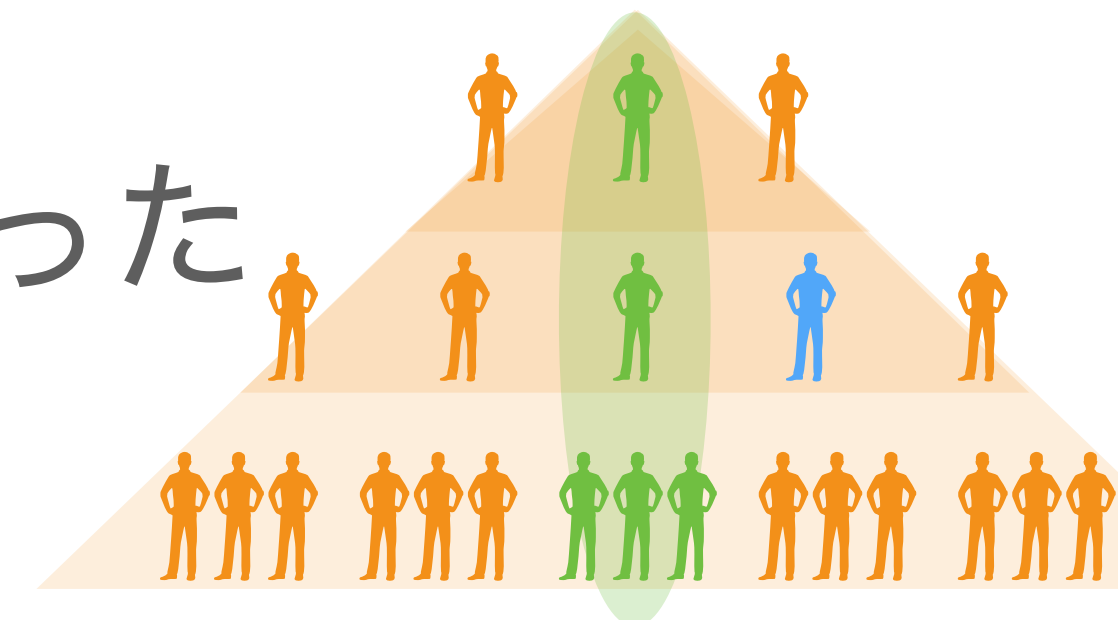


ステークホルダー全体との握りは弱かった

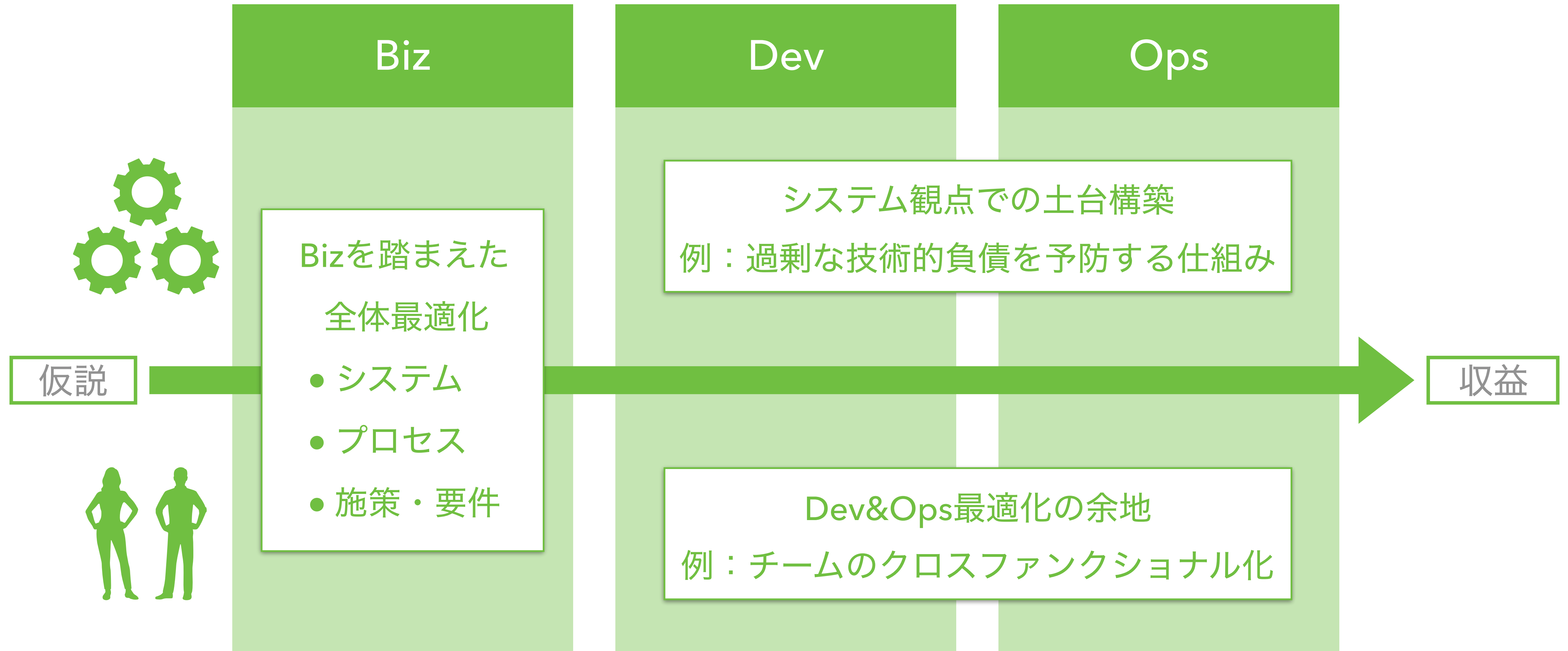
- 状況が状況なので「チーム内の課題を切り開くこと」「上位組織に伝達すること」が最優先
- 外からの見え方は「障害を出した・開発が遅れているのに、手を止めてワークショップを始めるチーム」
- 嫌味を言われたのは1度や2度ではない → この圧力が悪循環の真因 → 守り手としてのエンジニア部署の意義

上位組織&他部署メンバに裏でフォローいただく形となった

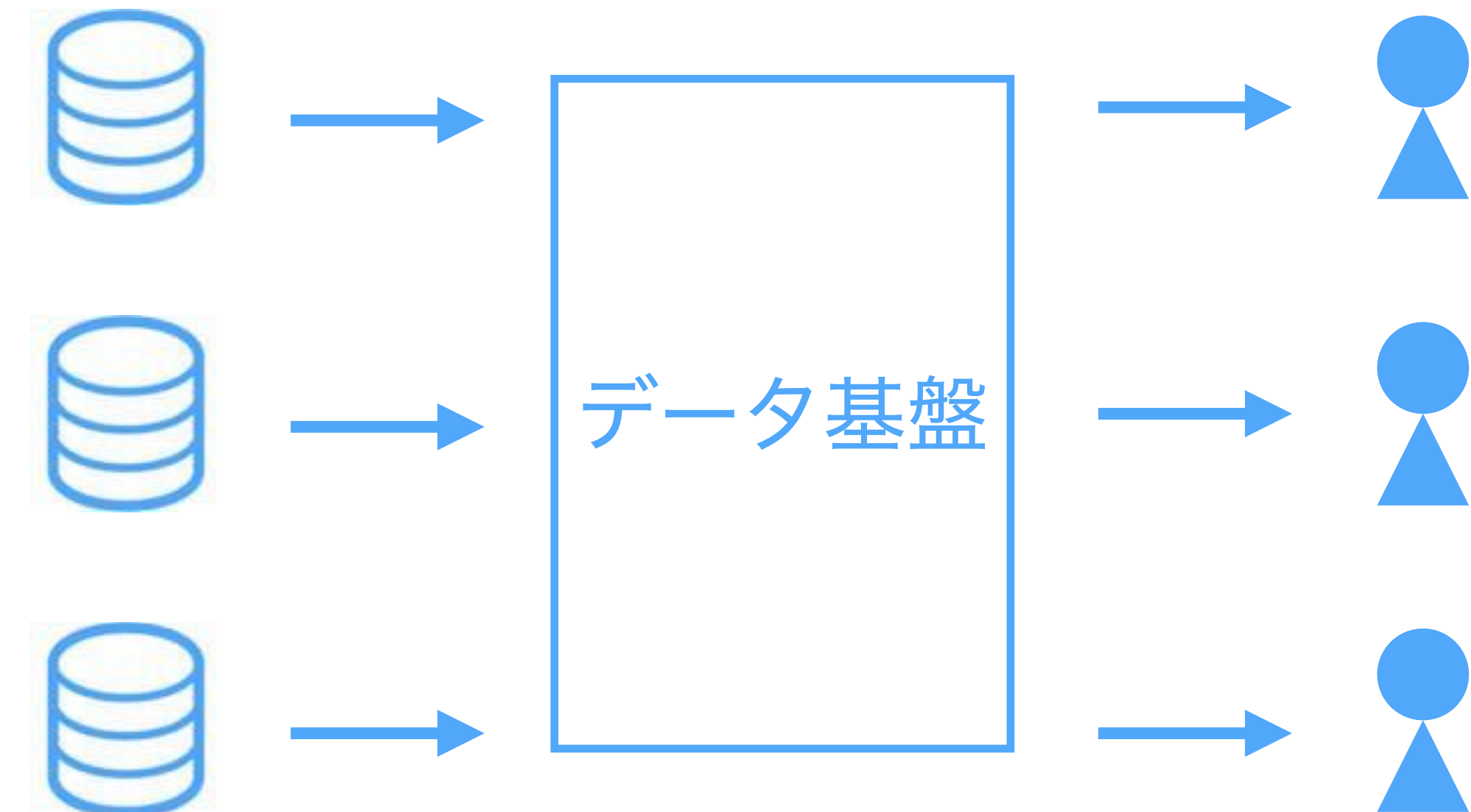
- 特に「開発経験あり」「ビジネス部門に所属」のメンバーが緩衝材となったのが大きい



さらなる成長のために



BizDevOpsに向けた取り組み



“部署・職務を超えたメトリクスの共有” と “データ活用文化の装着”

ビジネス価値に向けた改善

何だかんだで
TRYを著実に
完結している

よりハッキリ論
を ちゃんと
話せる。

開発と事業
と一緒に
案件の企画から

事業と開発
で一緒にやって
る感（チーム感）
（開発と事業）
に関して、それぞれで
やるかを分けて
案件を再設計して
くれている。

最後に

Product Growth

A background image of a young child jumping joyfully in a grassy field. The child is wearing a white long-sleeved shirt, a patterned scarf, blue jeans, and a dark cap. Their arms are raised high, and they are in mid-air, with their legs bent. The background is a soft-focus green field under a bright sky.

スピード感を大事にする

- 爆弾を避けながら前に走り続けるということ
- 爆弾に自分から突っ込むことではない → 負傷するほど遅くなる

エンジニアだからこそ気付けること

- 「ここは焦ってはダメだ！」 「爆弾だ！丁寧に進むぞ！」という場面
- 手を動かす立場（かつて手を動かした経験者）でないと分からない
- 急成長プロダクトではその場面がひたすら続く

逆説のエンジニアリング

ベストプラクティスは反直感的

- クオリティを高めるために、即時対応を減らす
- アジリティを高めるために、手を止めて話し合う

考えると当たり前のこと

- 焦ったまま仕事を続けて上手く行くわけがない
寝不足の操縦士が担当する飛行機に乗りたいか？ パニック状態の外科医に手術されたいか？
グロースフェーズだからといって悪循環を放置していい理由にはならない
- チームには透明性と検査と適応が必要

「これっておかしくないですか？」
誰かが声を上げることが、改善の第一歩

「これってどうなっていますか？」
誰かに声を掛けることが、透明性の第一歩

チームの一員の役割です
今ここにいる私たち自身の役割です

チームとしてプロセスを改善しよう

チームとしてシステムを改良しよう

チームとしてプロダクトを成長させよう

チームの安定の上にもこそ
継続的な顧客価値の提供が
(その結果としてビジネスの成長が)
実現できるのだと思っています



ご清聴ありがとうございました

presented by @yuzutas0

Appendix

- 試行錯誤・失敗事例
- 参考資料
- Special Thanks

改善役が最もコードを書いていた

夜中に1人でコーディングする羽目になった

- 最初の頃はメンバーは自身の担当範囲しか分からなかった
- サブシステム横断で設計・実装できる人材がいない状態

3,179 contributions in the last year



背中を見せることの光と陰

- 着任早々に難易度の高い設計・実装をやってのけたからこそ、メンバーの信頼を得た側面もある
- 改善に力を割けず、人は育たず、チームが機能しない（プレイングマネージャーのアンチパターン）

自走を促して徐々に健全化

- 若手社員の引き抜き + 会話の活性化 + 継続的な作業内容のフィードバック（週次レビュー）

BizDevOpsの促進 - 早すぎた施策

1. チームビジョン、クレド、開発KPIの策定

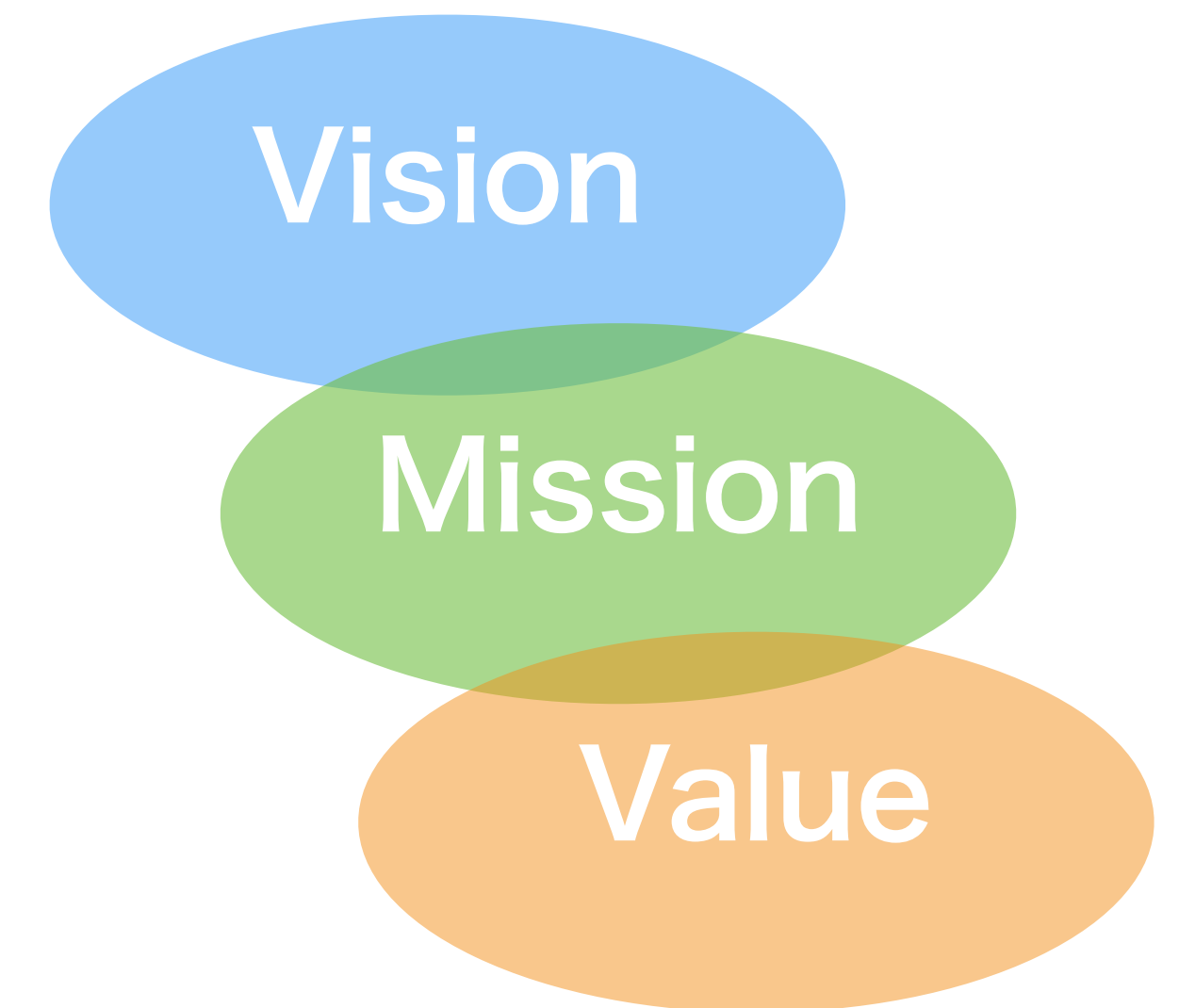
- 仮説：目標像や基準があれば改善の方向性が見えてメンバーが自発的に動ける
- 結果：マイナスをゼロにしている途中なので、将来の絵空事・現実味がない綺麗事

2. ビジネス構造・状況についての勉強会

- 仮説：担当外の業務や分野の話を聞くことでBizへの染み出しを後押しできる
- 結果：面白そうだけど今はそれどころではないのでフラストレーション

→ 最近になって再チャレンジ

- 「2歩先を見据えた上で、1歩先を促すくらいがちょうどいい」 by メンター



参考資料

書籍・PDF

『リーン開発の本質 - ソフトウェア開発に活かす7つの原則』 『スクラムガイド - スクラム完全ガイド: ゲームのルール』
『組織パターン - チームの成長によりアジャイルソフトウェア開発の変革を促す』 『エクストリームプログラミング - Embrace Change』
『ウェブオペレーション - サイト運用管理の実践テクニック』 『強い会社はこうして作られる！ - ITIL実践の鉄則』

スライド

- 結果的にスクラムになってる！なのがいいと思う！ <https://speakerdeck.com/bufferings/jie-guo-de-nisukuramuninatuteru-nafalsegaiitosi-u-number-rsgt2017>
- 効果的な自動化を目指す！ Value Stream Mapping 実践ワークショップ <https://docs.com/ushio-tsuyoshi/8263>
- 新規事業が対峙する現実からエンジニアリングを俯瞰する <https://www.slideshare.net/i2key/devsumib>
- サービスレベル：設計と運用のプラクティス <http://yuzutas0.hatenablog.com/entry/2017/05/23/073000>
- エスカレを支える技術 - アジャイルな報連相と情報流通 <http://yuzutas0.hatenablog.com/entry/2017/08/10/090000>
- DevOpsとドキュメントデザインパターン <http://yuzutas0.hatenablog.com/entry/2017/07/06/083000>
- JupyterとBigQueryによるデータ分析基盤のDevOps <http://yuzutas0.hatenablog.com/entry/2017/09/12/203000>



Special Thanks

彼らの心強い支えがあったからこそ、チーム立て直しに専念することができました。

毎日の 泣き事 相談に粘り強く対応し続けてくださったことを感謝しております。

Satoshi Uejima 至らなかつた点を裏でフォローいただきました。「もう無理っす」の度に、見苦しい言い訳の1つ1つと丁寧に向き合ってくれた最高の上司でした。

Itsuki Kuroda チームの実情や文脈を踏まえた上で、改善の打ち手・進め方について本質的かつ生産的なアドバイスをいただきました。

Kenichi Takahashi 組織面での惜しみないバックアップを通して
基盤整備や案件開発の軌道が乗るようにサポートいただきました。