Agileとテスト自動化導入の勘所

Dec 7, 2018

Ayana Yoshida, Kotaro Ogino

Rakuten, Inc.

Rakuten

JaSST’ 18 Tokyoのクロージングパネルから始まるAIとQAの話は

クロージングパネル

セッション A8 ▶概要

「アジャイル・自動化時代のテストの現場のリアル」

モデレータ：
荻野 恒太郎 （楽天）
Download 講演資料 (PDF : 913KB)

パネリスト：
John Micco （Google）
Download 講演資料 (PDF : 148KB)

天野 祐介 （サイボウズ）
Download 講演資料 (PDF : 4,389KB)

松尾 和昭 （クックパッド）
Download 講演資料 (PDF : 1,118KB)

山口 鉄平 （ヤフー）
Download 講演資料 (PDF : 164KB)

JaSST’18 Tokyo レポート

http://www.jasst.jp/symposium/jasst18tokyo/report.html

2018/11/13 アクセス

Test Automation テスト自動化とか品質とか勉強会とかやってるエンジニア（Automation Architect）のブログです。

2018-12-06

アジャイルもテスト自動化も当たり前？！ ~AIがテスト設計をする日が来るかも~

はじめに

このエントリは、ソフトウェアテスト #2 Advent Calendarの６目の記事として書いています。

ソフトウェアテスト #2 Advent Calendar 2018 - Qiita
ソフトウェアテスト Advent Calendar 2018 が埋まってしまいましたので、2を作りました。 テストにまつわることなら何でもありです！ ソフトウェアテスト Advent Calendar 2018 https://qiita.co...
qiita.com

qiita.com

モチベーション：JaSST'18 Tokyo 振り返り

JaSST' 18 Tokyo のクロージングパネルでは、実は当初予定していたモノから内

Profile

id:kokotatata PRO
テスト自動化とか品質とか勉強会とかやってるエンジニアのブログです。
読者になる 35

Search

記事を検索

Recent Entries

2018-12-06
アジャイルもテスト自動化も当たり前？！ ~AIがテスト設計をする日が来るかも~

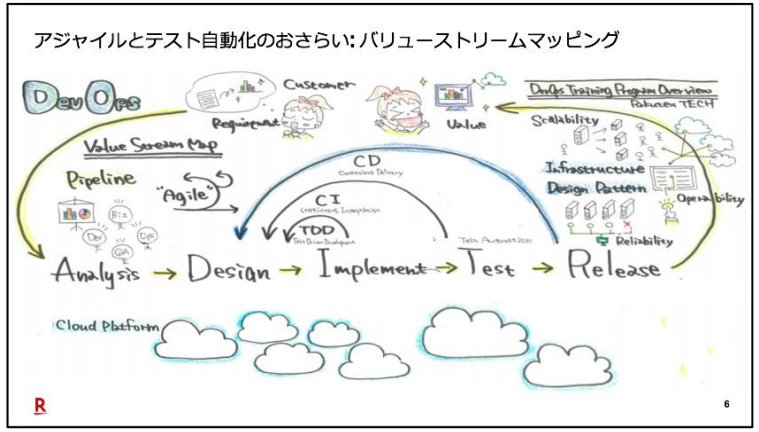
アジャイル　テスト自動化　当たり前

検索

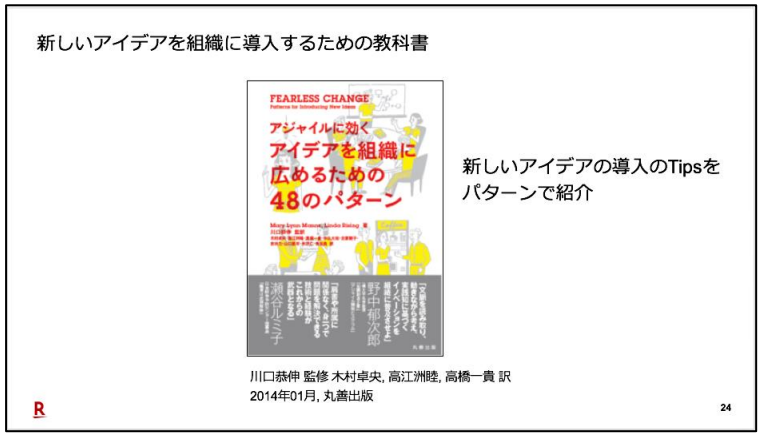
本日お持ち帰りいただきたいこと

本日お持ち帰りいただきたいこと

① おさらい

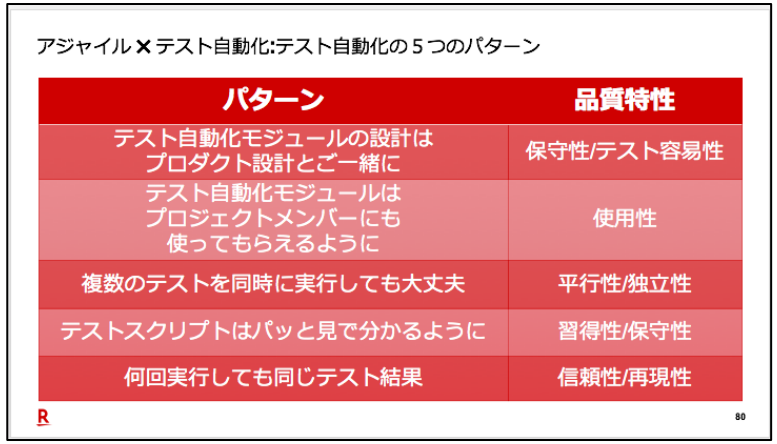


② 導入の勘所



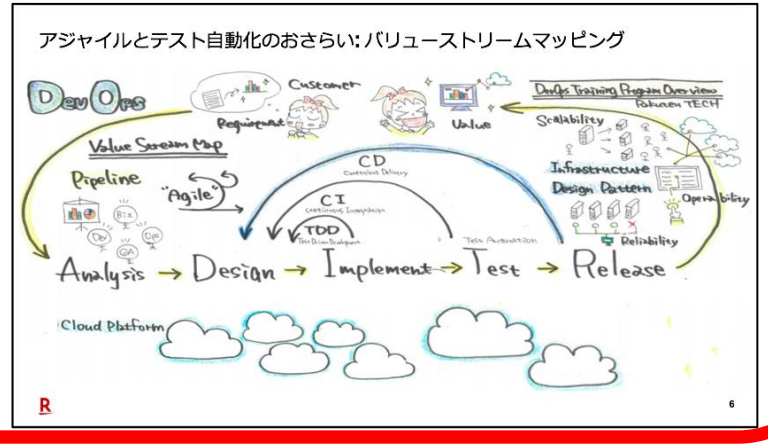
③ 導入事例（文化面）

④ 導入事例（技術面）

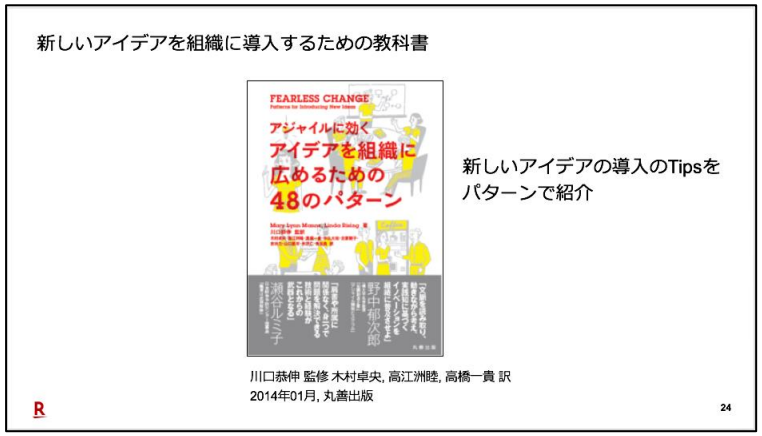


本日お持ち帰りいただきたいこと

① おさらい

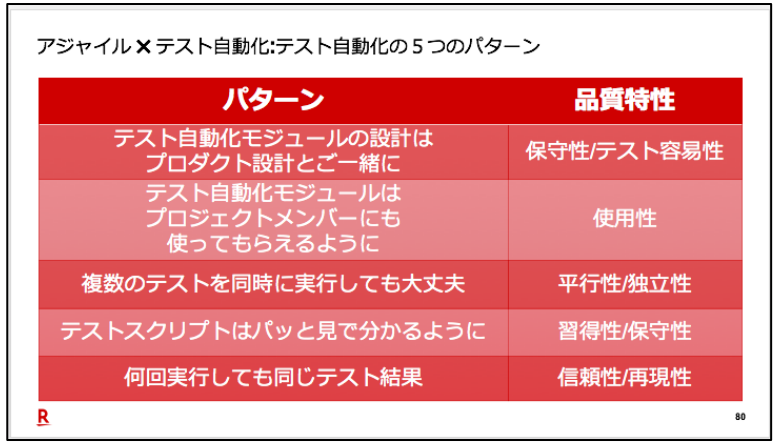


② 導入の勘所

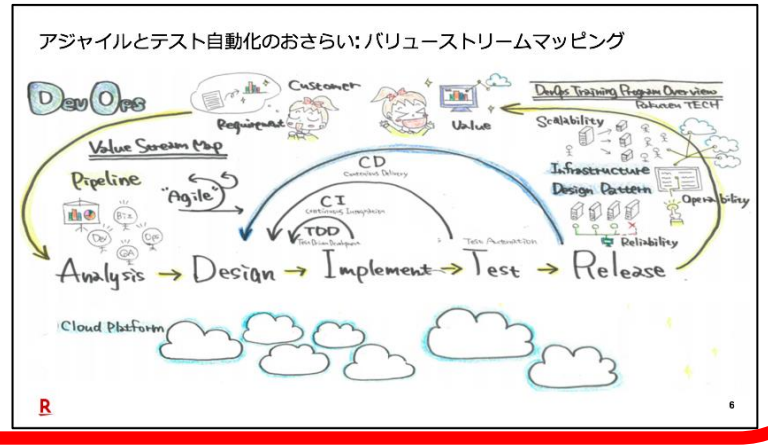


③ 導入事例（文化面）

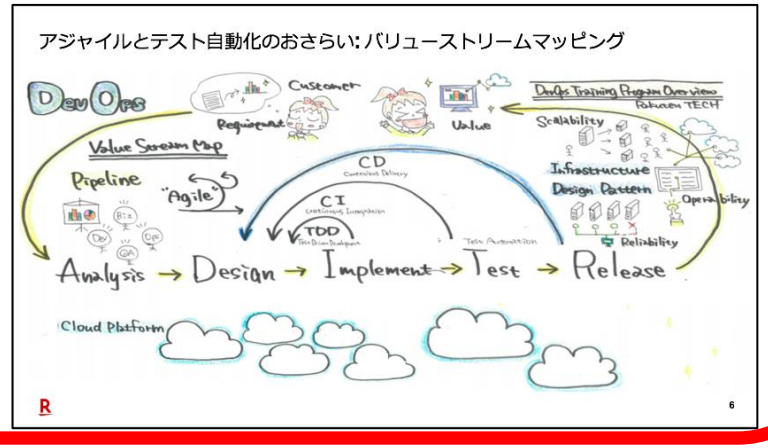
④ 導入事例（技術面）



① おさらい


アジャイルとテスト自動化のおさらい：バリューストリームマッピング
DevOps
Customer
Requirement
Value
DevOps Training Program Overview
Fukuoka TECH
Value Stream Map
Pipeline
"Agile"
CD
CI
TDD
Scalability
Infrastructure
Design Pattern
Operability
Reliability
Analysis → Design → Implement → Test → Release
Cloud Platform
6

① おさらい

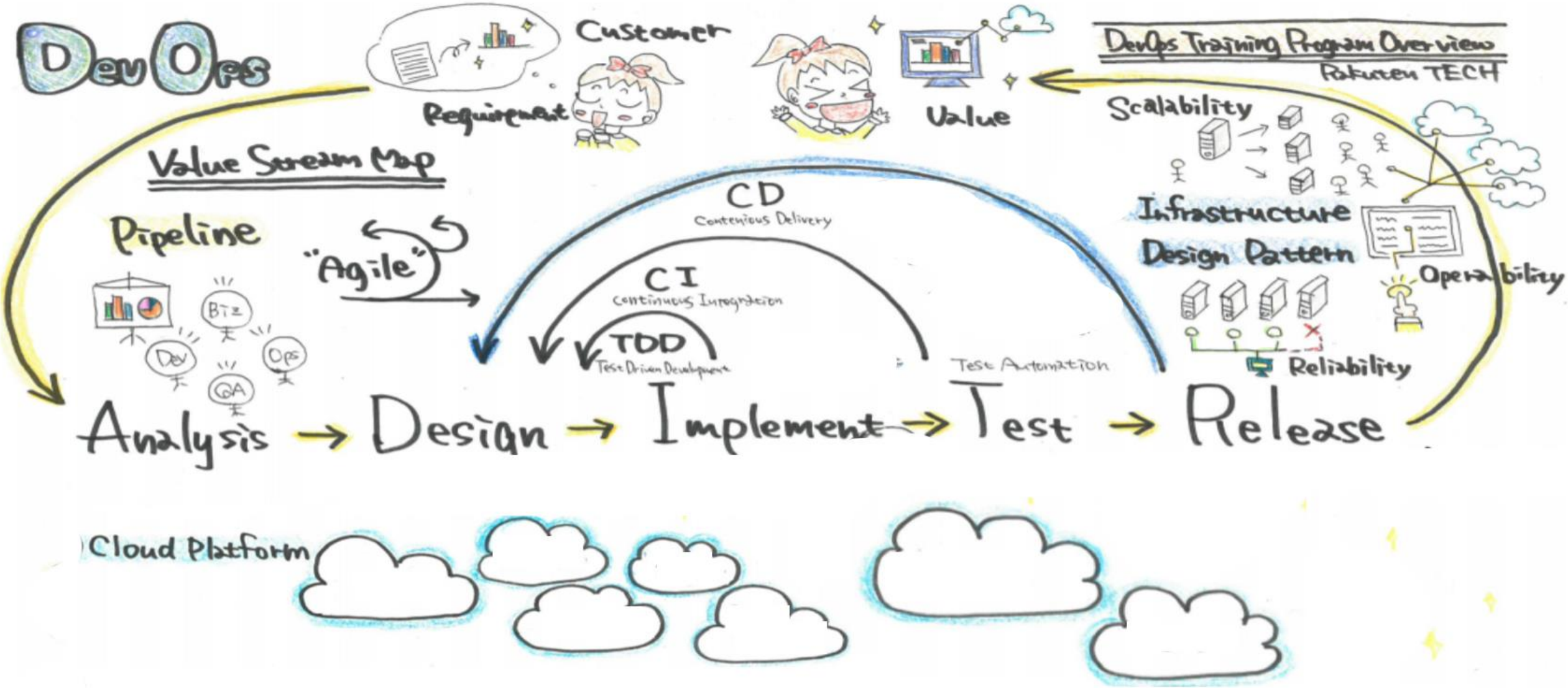


- アジャイルとは何か？

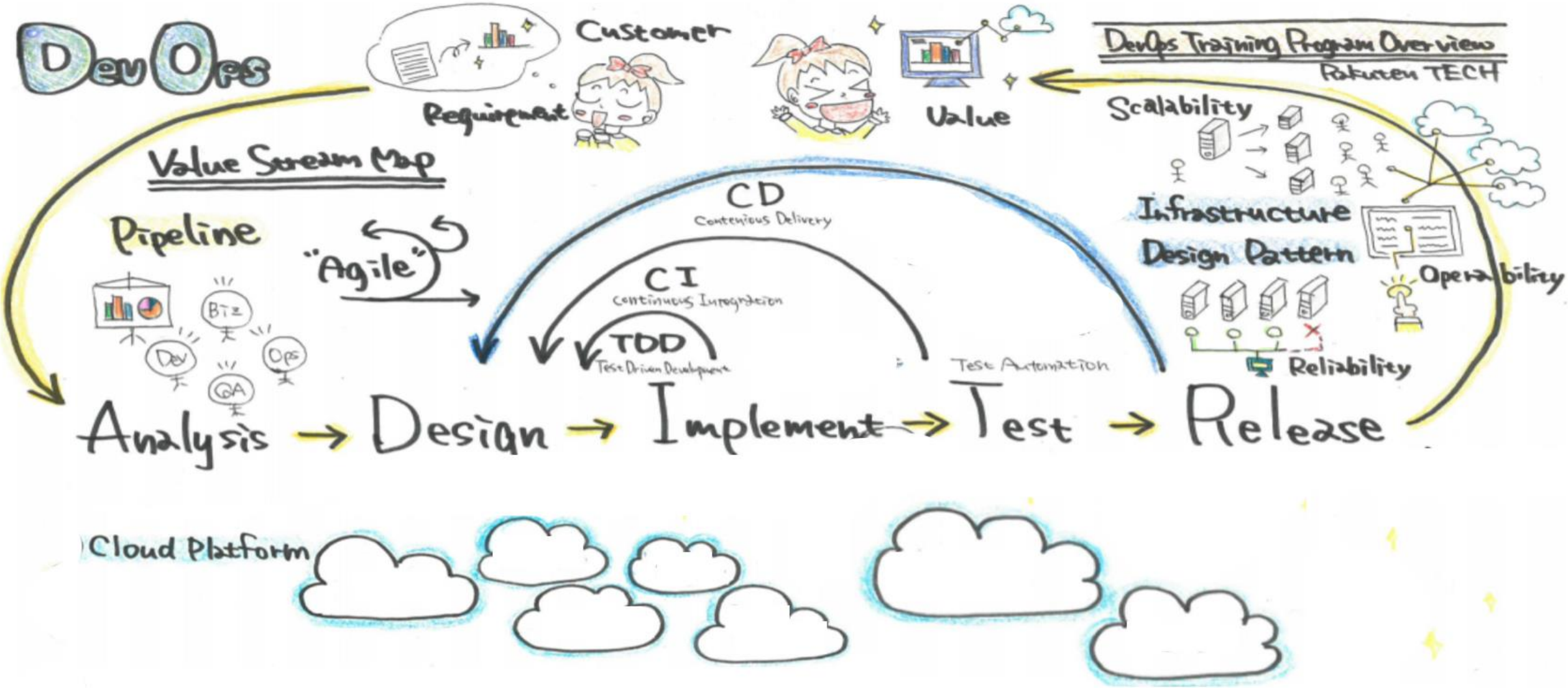
- アジャイルでのテスト自動化の役割

アジャイルとテスト自動化のおさらい: バリューストリームマッピング

アジャイルとテスト自動化のおさらい: バリューストリームマッピング

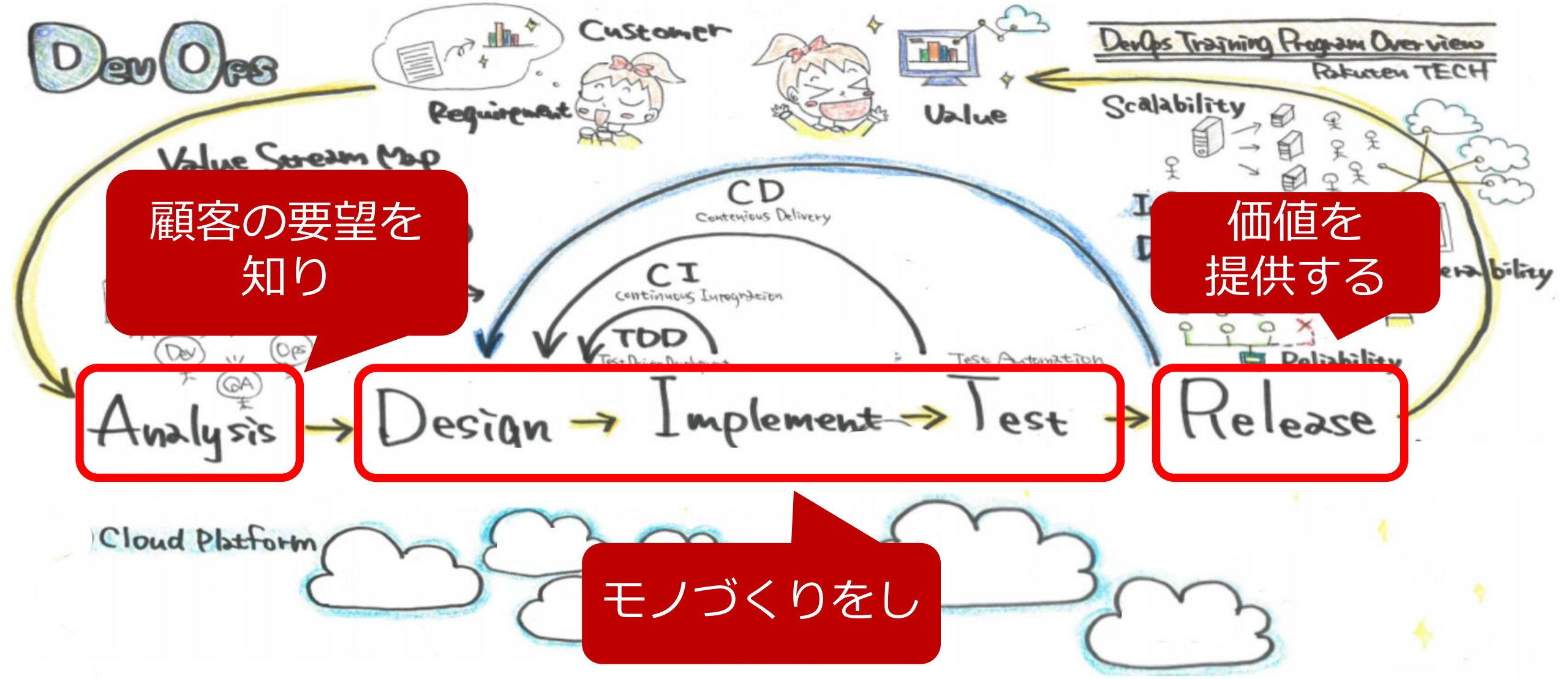


アジャイルとテスト自動化のおさらい:アジャイルとは何か？



アジャイルとテスト自動化のおさらい:アジャイルとは何か？

アジャイルとテスト自動化のおさらい:アジャイルとは何か？



- 顧客の要望を知り
- モノづくりをし
- 価値を提供する

- 顧客の要望を知り
- モノづくりをし
- 価値を提供する

ソフトウェア開発に共通

- 顧客の要望を知り
- モノづくりをし
- 価値を提供する

ソフトウェア開発に共通

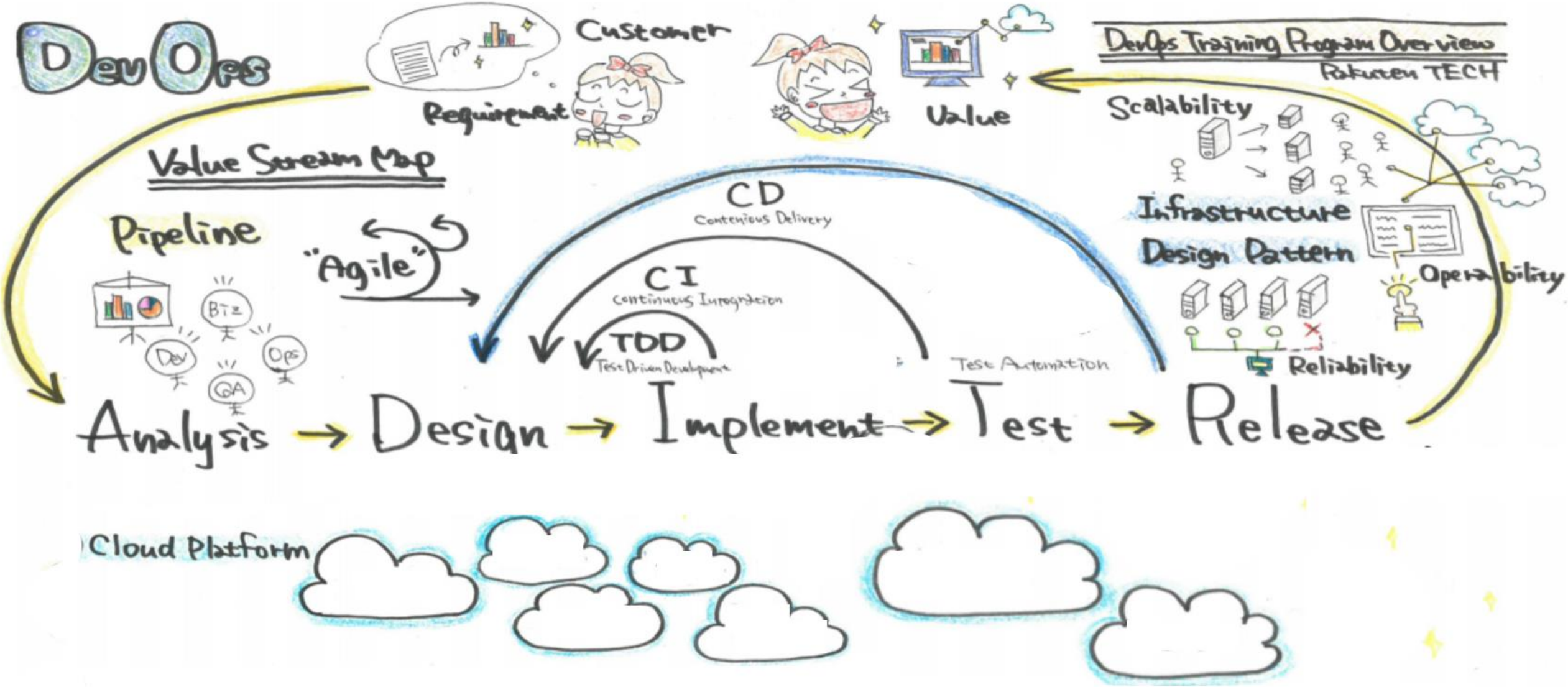
×

- 少しずつ
- 素早く

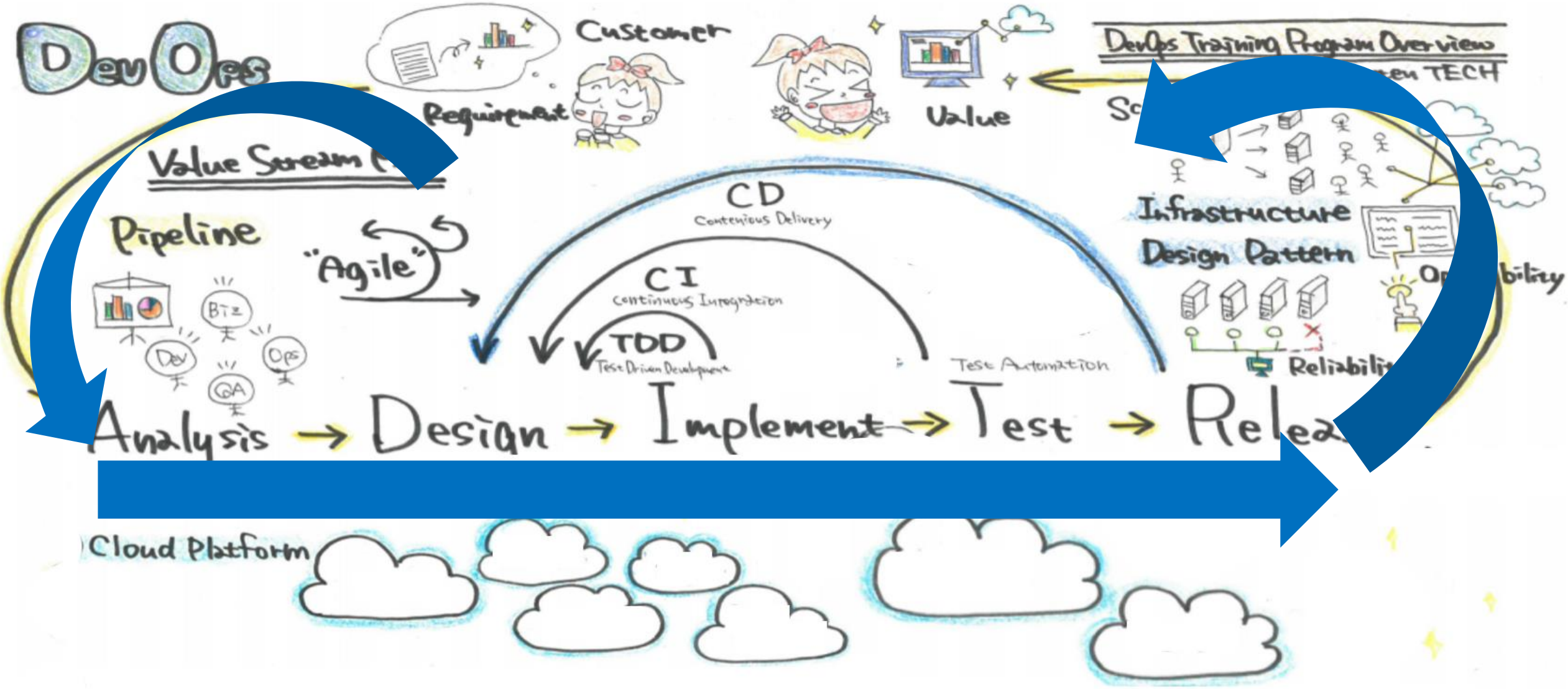
アジャイルの特徴

アジャイルとテスト自動化のおさらい:アジャイルとは何か？（少しずつ、素早く）

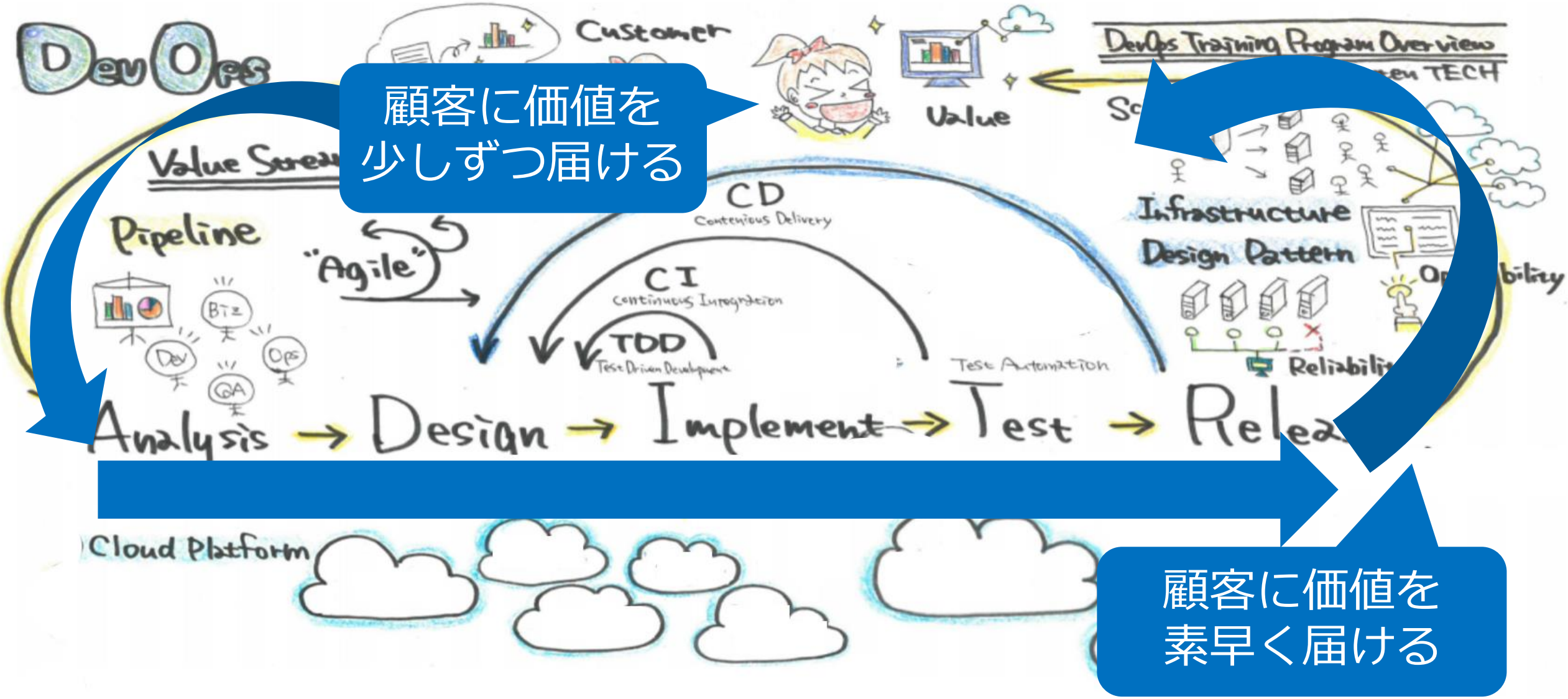
アジャイルとテスト自動化のおさらい:アジャイルとは何か？（少しずつ、素早く）



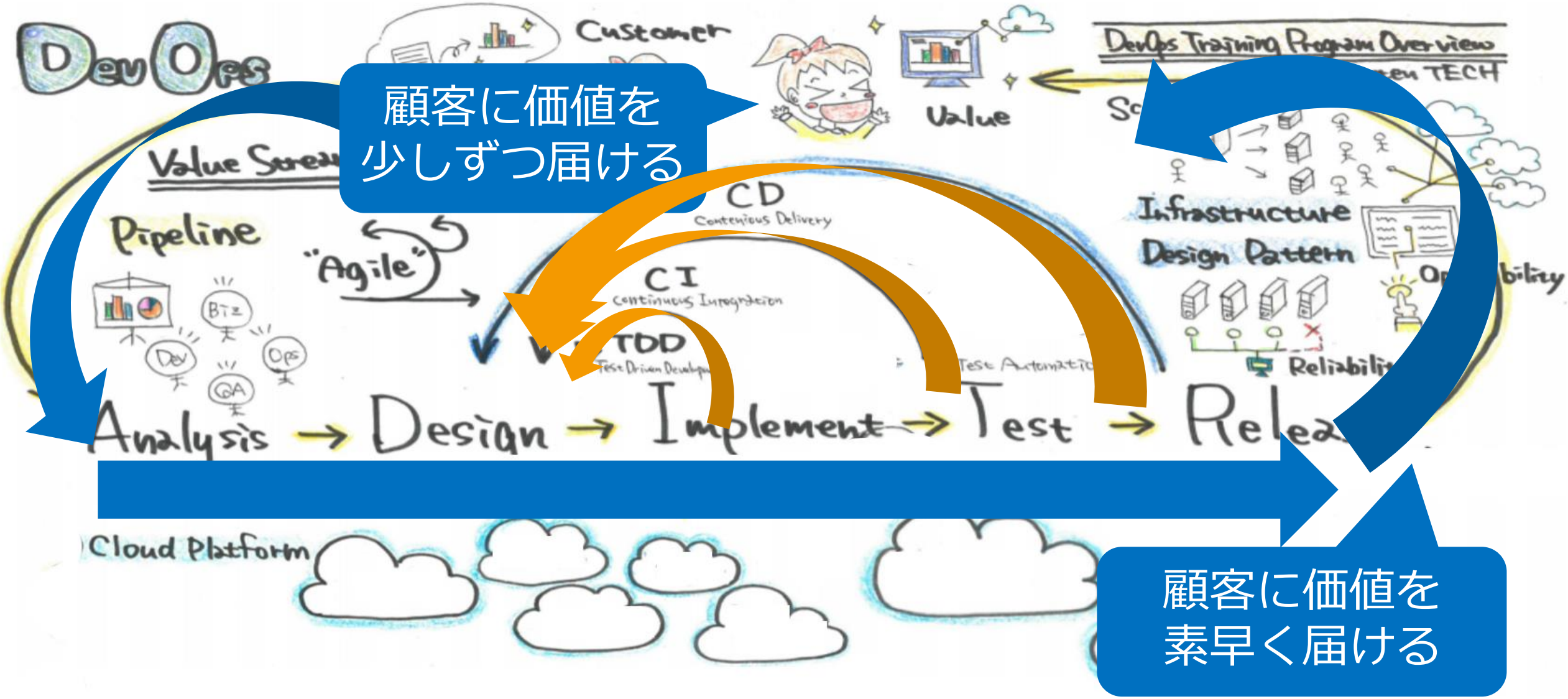
アジャイルとテスト自動化のおさらい:アジャイルとは何か？（少しずつ、素早く）



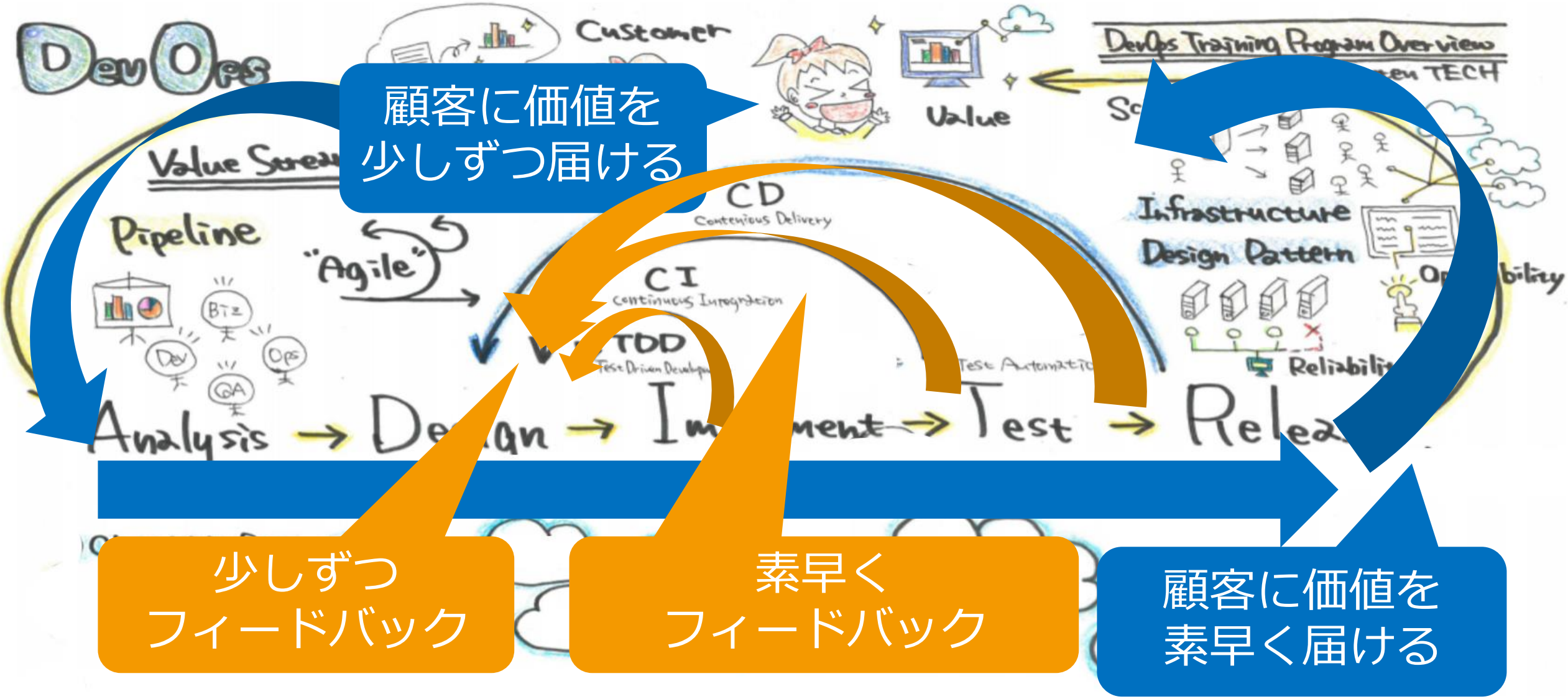
アジャイルとテスト自動化のおさらい:アジャイルとは何か？（少しずつ、素早く）



アジャイルとテスト自動化のおさらい:アジャイルとは何か？（少しずつ、素早く）



アジャイルとテスト自動化のおさらい:アジャイルとは何か？（少しずつ、素早く）



アジャイルとテスト自動化のおさらい:アジャイルとは何か？（少しずつ、素早く）

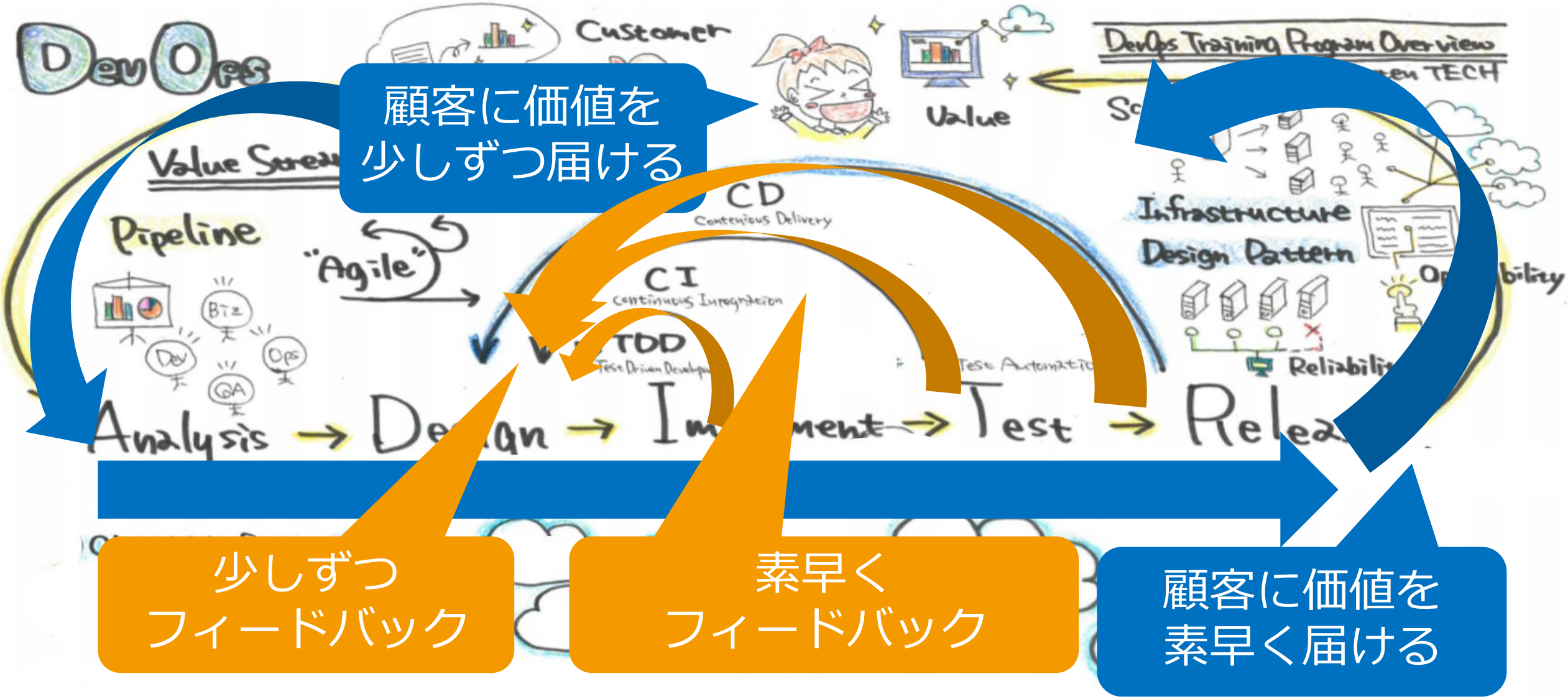
アジャイルとテスト自動化のおさらい:アジャイルにおけるテスト自動化の役割

役割	アジャイルのプラクティス
顧客に価値を 少しずつ・素早く届ける	• スクラム • ユーザーストーリーマッピング • リーンスタートアップ
チームにフィードバックを 少しずつ・素早く届ける	• スクラム • テスト駆動開発 • 継続的インテグレーション • 継続的デリバリー

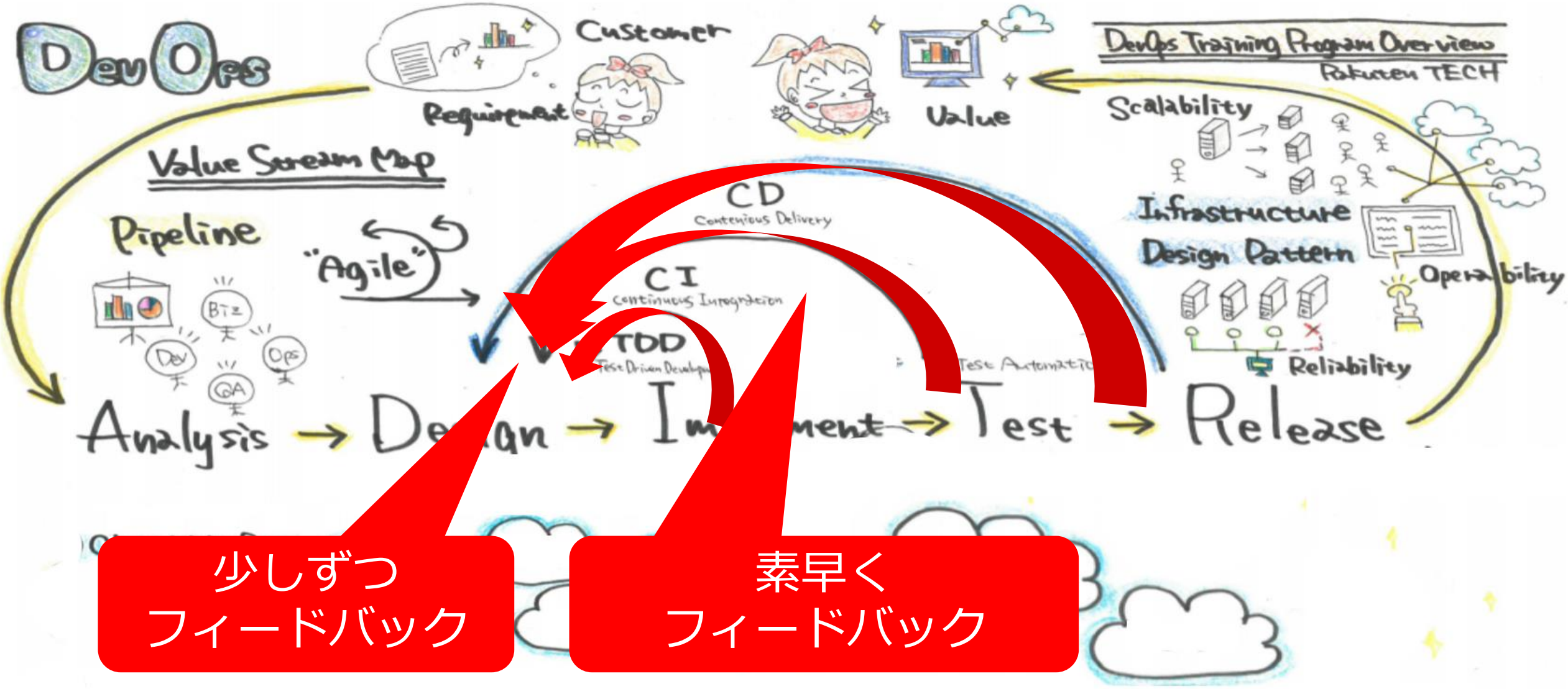
アジャイルとテスト自動化のおさらい:アジャイルにおけるテスト自動化の役割

役割	アジャイルのプラクティス
顧客に価値を 少しずつ・素早く届ける	• スクラム • ユーザーストーリーマッピング • リーンスタートアップ
チームにフィードバックを 少しずつ・素早く届ける	• スクラム • テスト駆動開発 • 継続的インテグレーション • →テスト自動化！ • 継続的デリバリー • →テスト自動化！

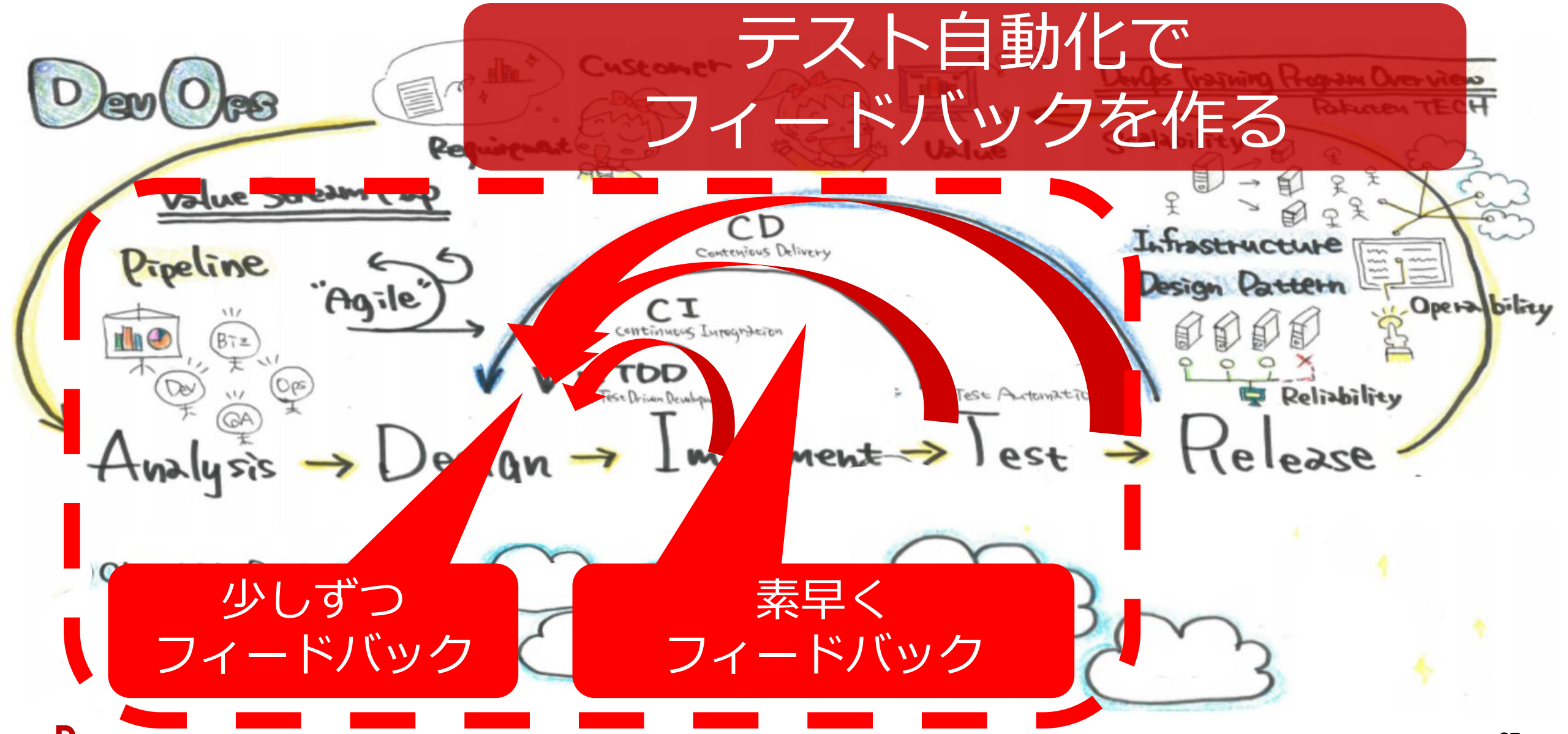
アジャイルとテスト自動化のおさらい:アジャイルにおけるテスト自動化の役割



アジャイルとテスト自動化のおさらい:アジャイルにおけるテスト自動化の役割



アジャイルとテスト自動化のおさらい:アジャイルにおけるテスト自動化の役割

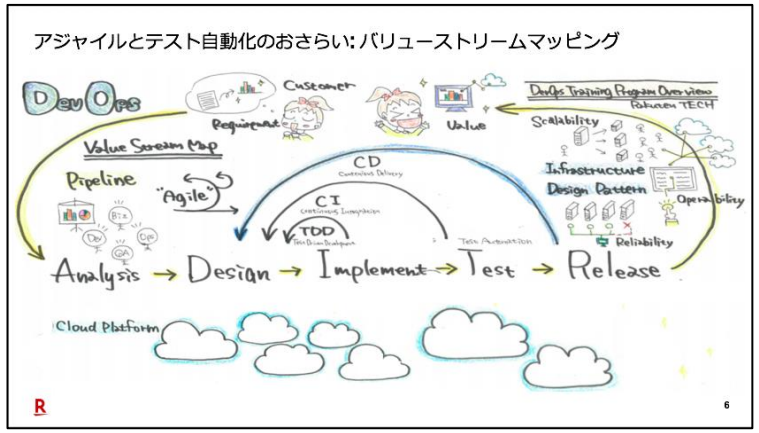


アジャイルとテスト自動化のおさらい:アジャイルにおけるテスト自動化の役割

少しずつ、素早くチームに
フィードバックするための仕組み

まとめ①

① おさらい

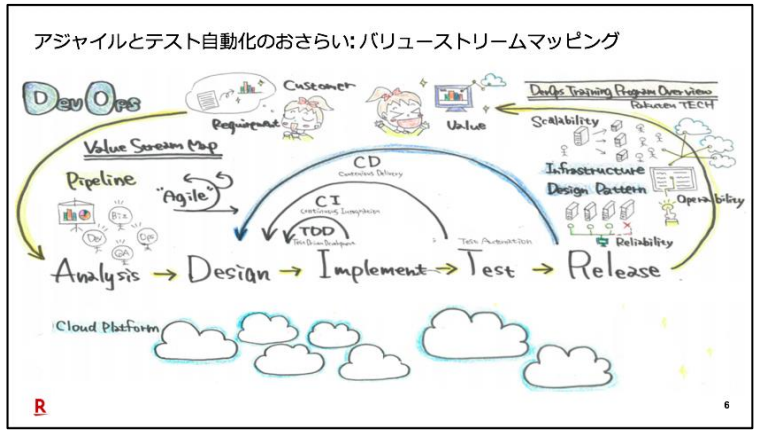


- アジャイルとは何か？

- アジャイルでのテスト自動化の役割

まとめ①

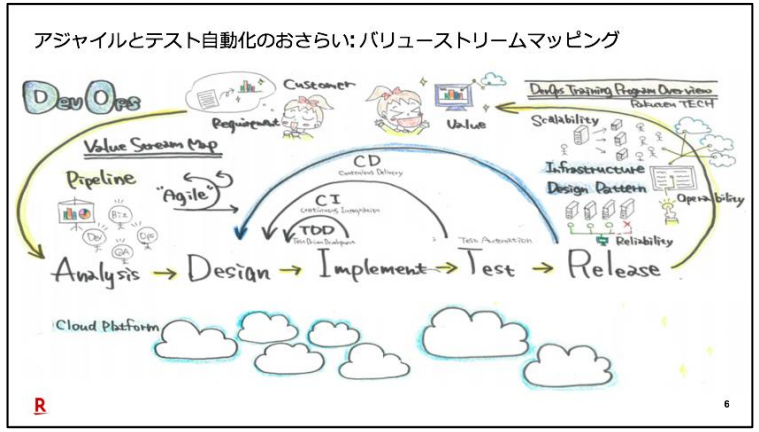
① おさらい



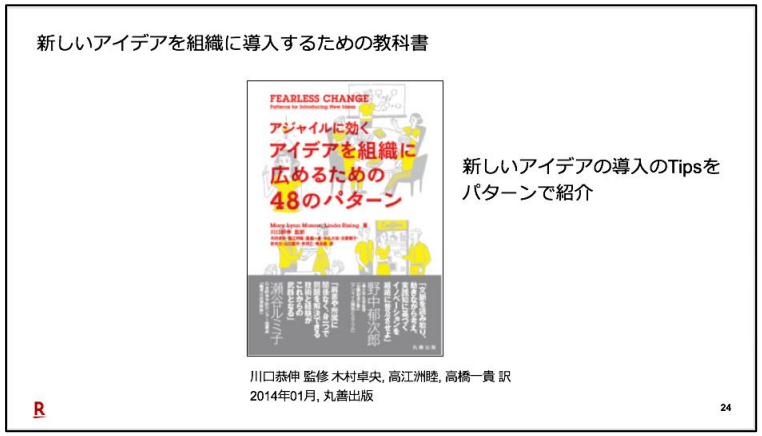
- アジャイルとは何か？
  →少しずつ、素早く
- アジャイルでのテスト自動化の役割
  →チームにフィードバック

本日お持ち帰りいただきたいこと

① おさらい

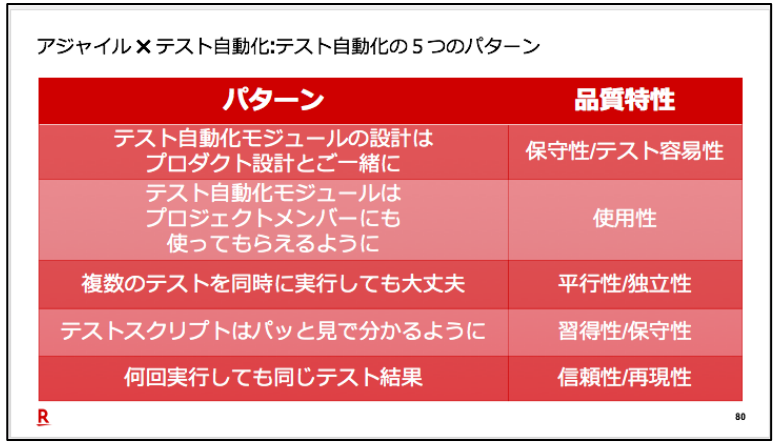


② 導入の勘所



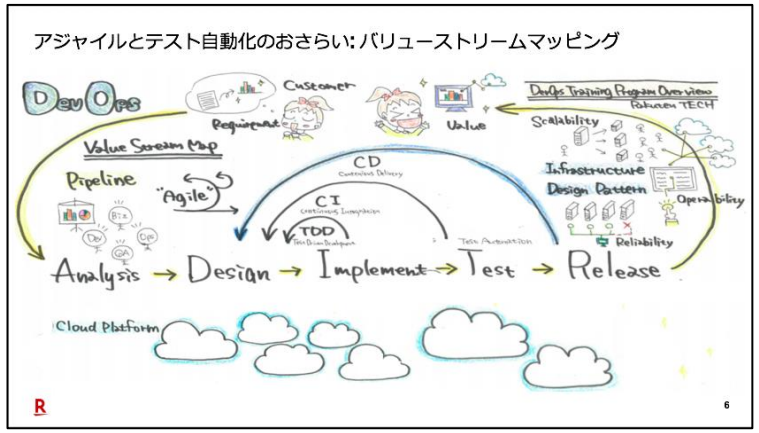
③ 導入事例（文化面）

③ 導入事例（技術面）

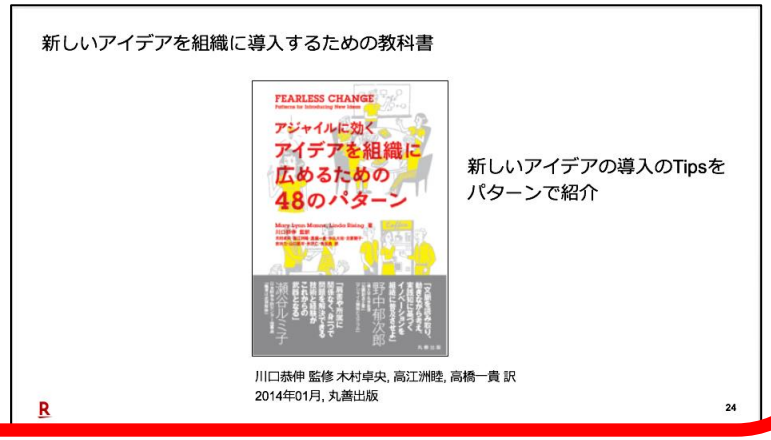


本日お持ち帰りいただきたいこと

① おさらい

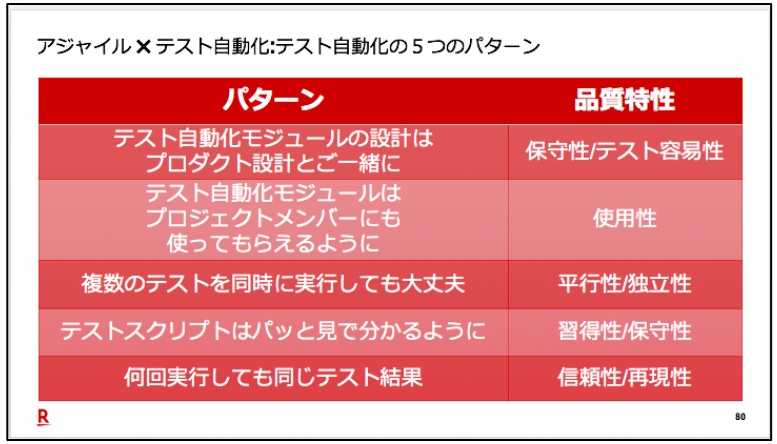


② 導入の勘所



③ 導入事例（文化面）

③ 導入事例（技術面）



② 導入の勘所

新しいアイデアを組織に導入するための教科書

新しいアイデアの導入のTipsを
パターンで紹介

川口恭伸 監修 木村卓央, 高江洲睦, 高橋一貴 訳
2014年01月, 丸善出版

R

24

② 導入の勘所

新しいアイデアを組織に導入するための教科書

新しいアイデアの導入のTipsを
パターンで紹介

川口恭伸 監修 木村卓央, 高江洲睦, 高橋一貴 訳
2014年01月, 丸善出版

R

24

・新しいアイデア導入のTips

導入の勘所：

導入の勘所：

ここまで見てきて良いことづくめのアジャイルとテスト自動化ですが、

導入の勘所：新しいアイデアの導入には障壁がある

導入の勘所：新しいアイデアの導入には障壁がある

新しいアイデアの導入では一般的に

・チームに経験者がいないこと

・周りに仲間がいないこと

・もともとあるルール

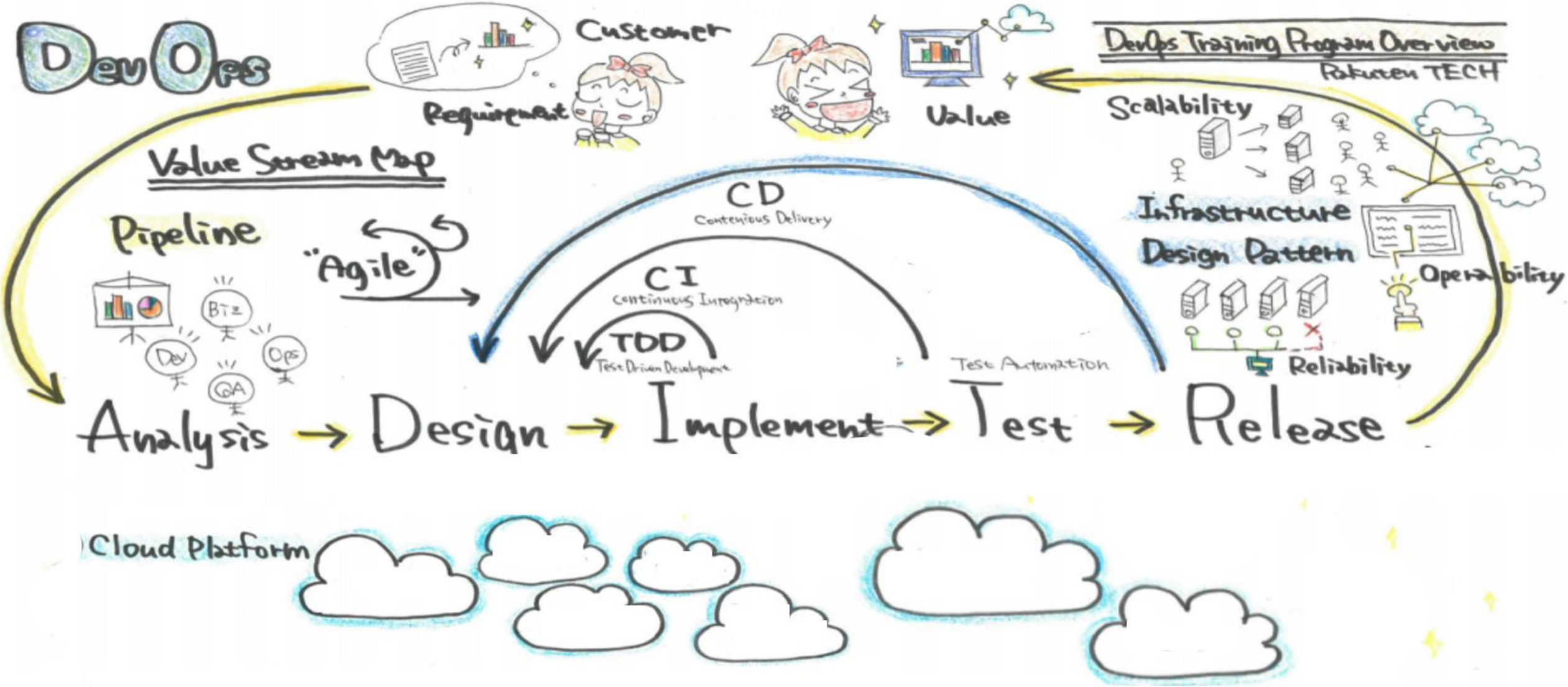
などが障壁に

アジャイル・テスト自動化の導入でも

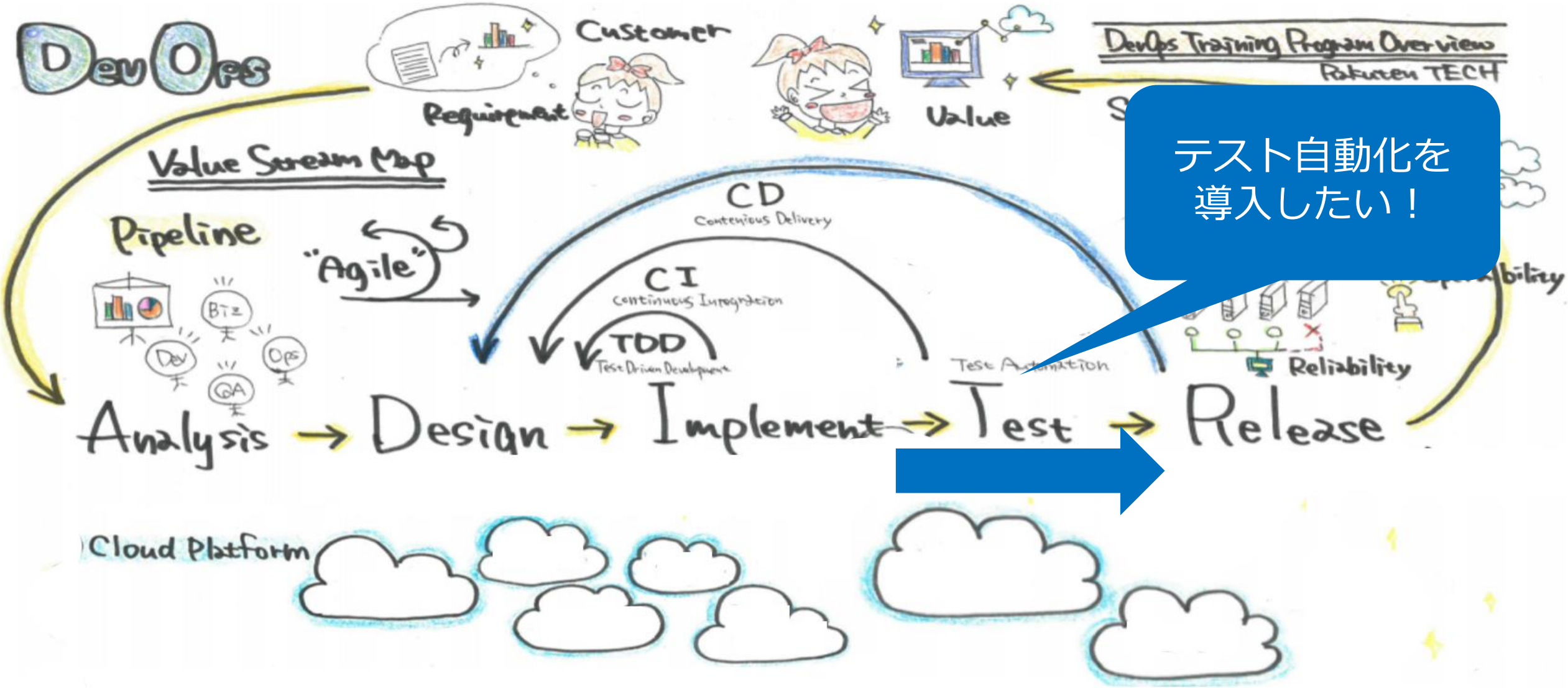
これらは障壁に成りうる

導入の勘所：障壁をテスト自動化の例で考えると

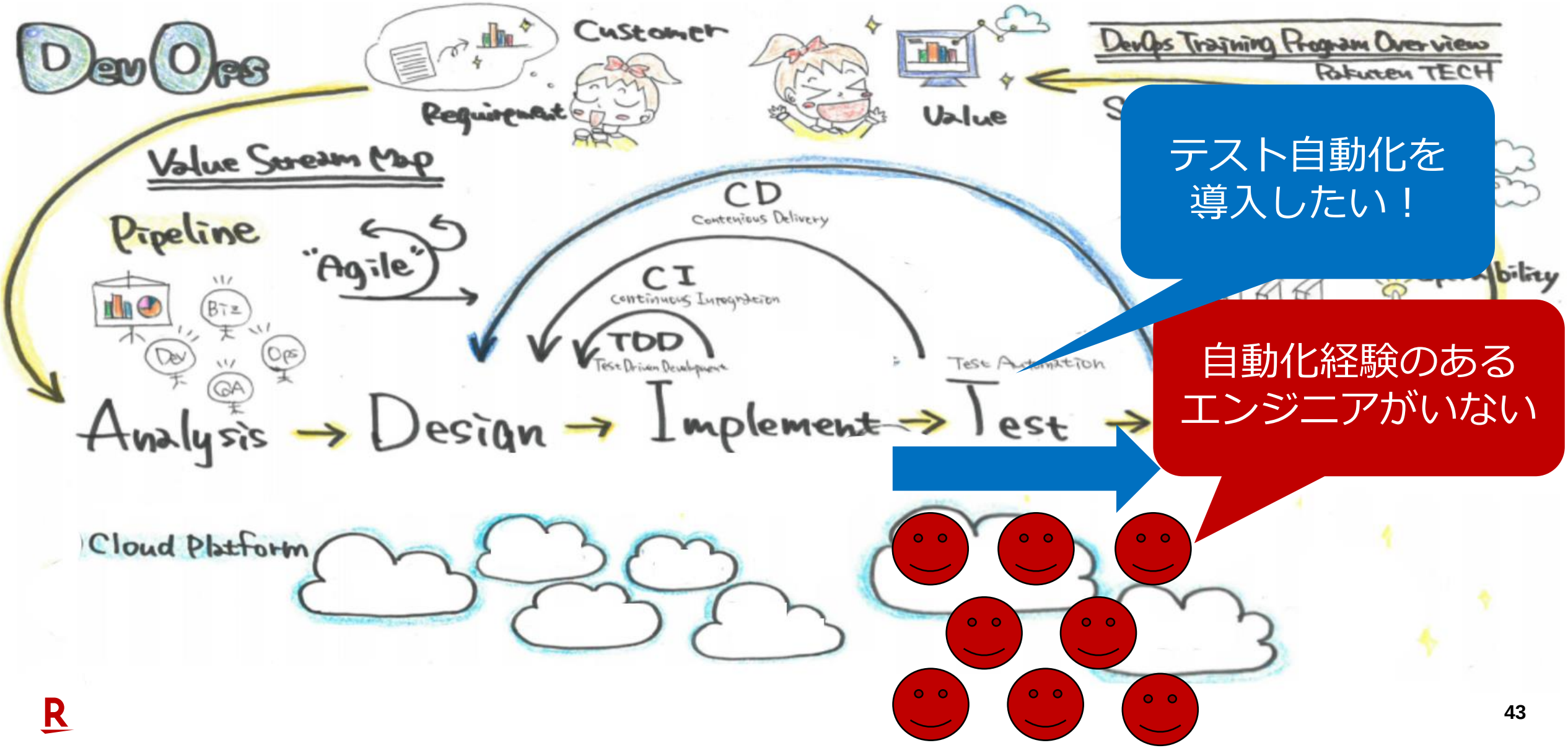
導入の勘所：障壁をテスト自動化の例で考えると



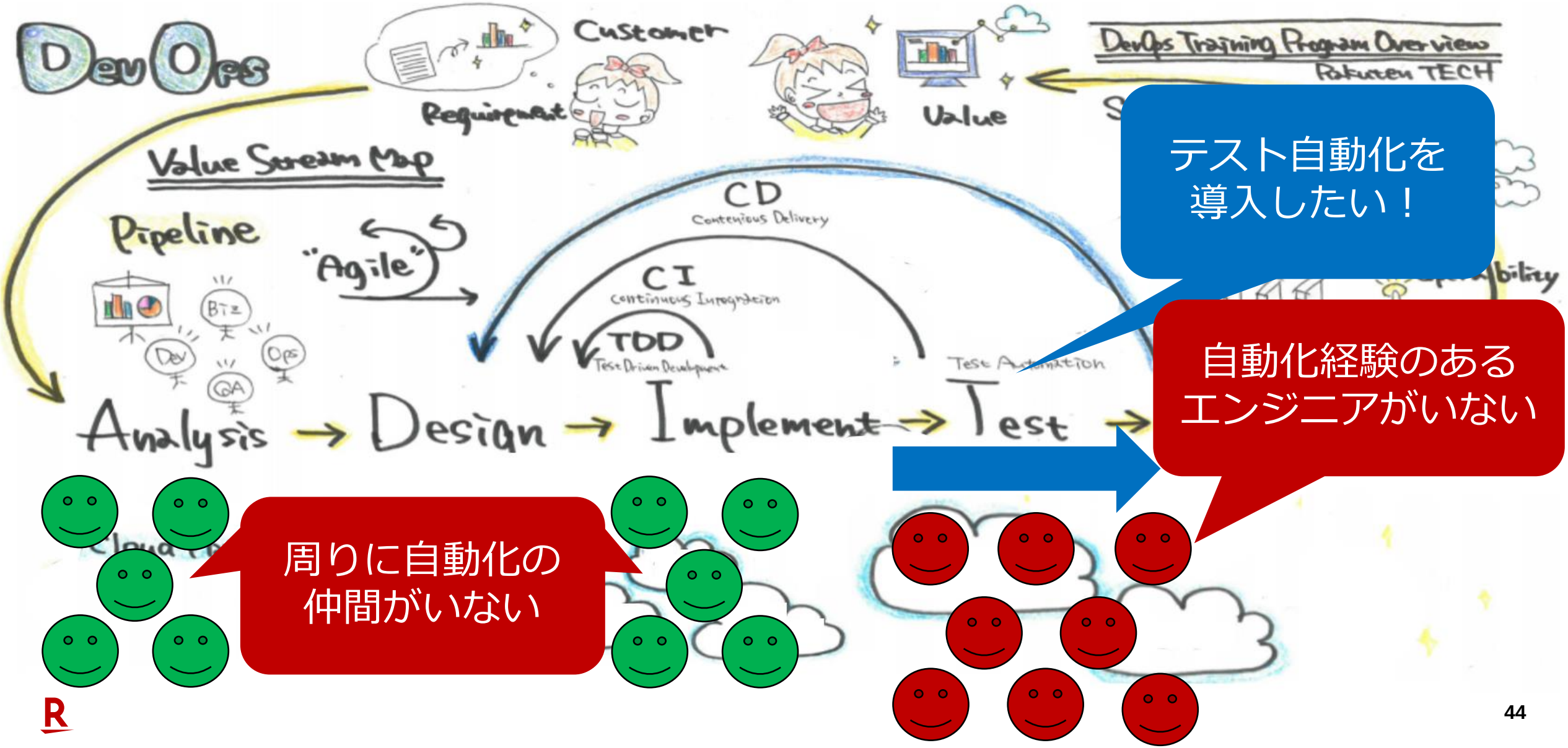
導入の勘所：障壁をテスト自動化の例で考えると



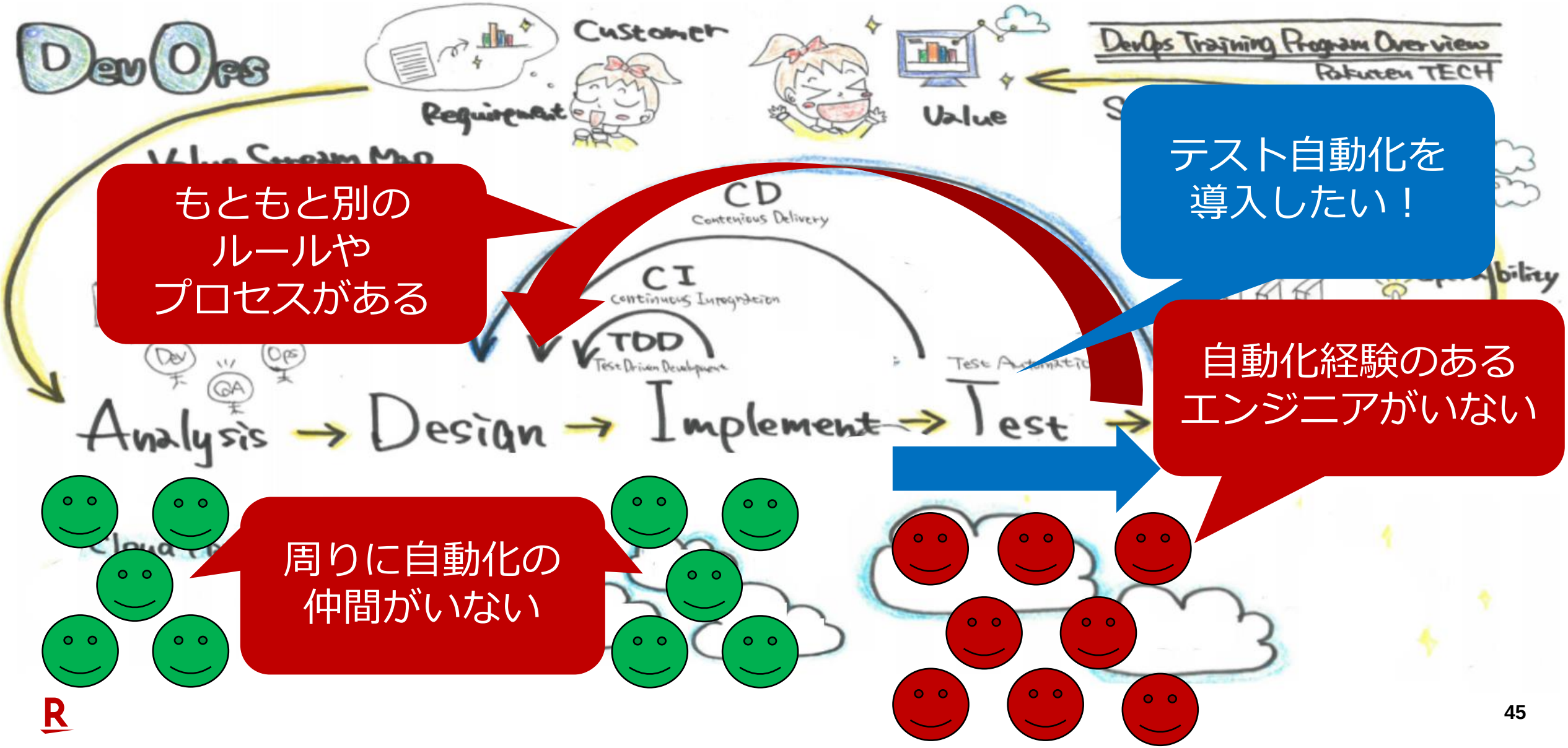
導入の勘所：障壁をテスト自動化の例で考えると



導入の勘所：障壁をテスト自動化の例で考えると



導入の勘所：障壁をテスト自動化の例で考えると



導入の勘所：新しいアイデアの導入には障壁がある

障壁がある場合
どうすれば乗り越えられるのか？

導入の勘所：新しいアイデアを組織に導入するための教科書

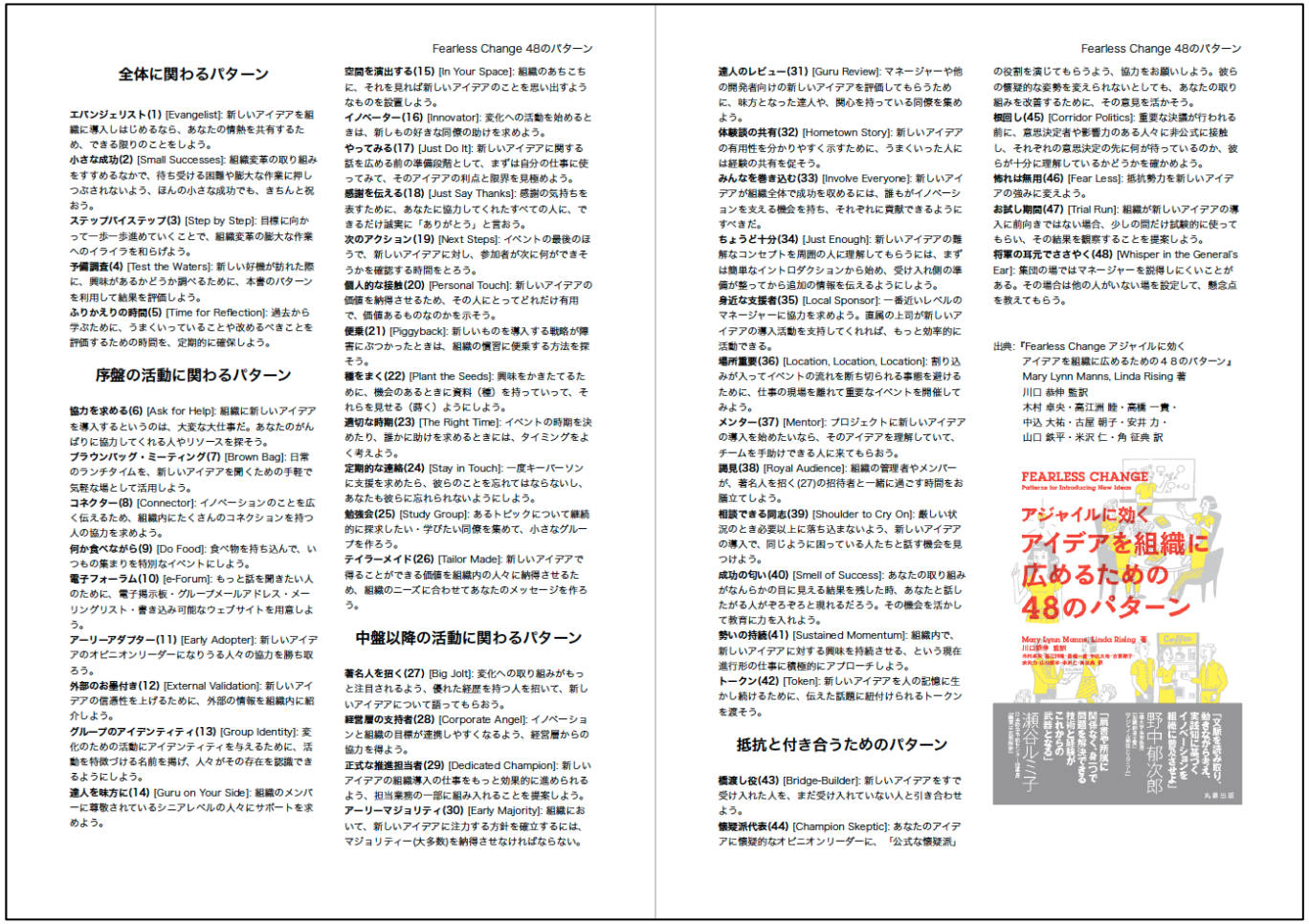
導入の勘所：新しいアイデアを組織に導入するための教科書

川口恭伸 監修 木村卓央, 高江洲睦, 高橋一貴 訳
2014年01月, 丸善出版

吉羽龍太郎、原田騎郎 訳
2017年01月, 技術評論社

導入の勘所： Fearless Change

導入の勘所： Fearless Change



Fearless Change 48のパターン

全体に関わるパターン

エバンジェリスト(1) [Evangelist]: 新しいアイデアを組織に導入しはじめるなら、あなたの情熱を共有するため、できる限りのことをしよう。
小さな成功(2) [Small Successes]: 組織変革の取り組みをすすめるなかで、待ち受ける困難や膨大な作業に押しつぶされないよう、ほんの小さな成功でも、きちんと祝おう。
ステップバイステップ(3) [Step by Step]: 目標に向かって一歩一歩進めていくことで、組織変革の膨大な作業へのイライラを和らげよう。
予備調査(4) [Test the Waters]: 新しい好機が訪れた際に、興味があるかどうか調べるために、本書のパターンを利用して結果を評価しよう。
ふりかえりの時間(5) [Time for Reflection]: 過去から学ぶために、うまくいっていることや改めるべきことを評価するための時間を、定期的に確保しよう。

序盤の活動に関わるパターン

協力を求める(6) [Ask for Help]: 組織に新しいアイデアを導入するというのは、大変な大仕事だ。あなたのがんばりに協力してくれる人やリソースを探そう。
ブラウンバッグ・ミーティング(7) [Brown Bag]: 日常のランチタイムを、新しいアイデアを聞くための手軽で気軽な場として活用しよう。
コネクター(8) [Connector]: イノベーションのことを広く伝えるため、組織内にたくさんのコネクションを持つ人の協力を求めよう。
何か食べながら(9) [Do Food]: 食べ物を持ち込んで、いつもの集まりを特別なイベントにしよう。
電子フォーラム(10) [e-Forum]: もっと話を聞きたい人のために、電子掲示板・グループメールアドレス・メーリングリスト・書き込み可能なウェブサイトを用意しよう。
アーリーアダプター(11) [Early Adopter]: 新しいアイデアのオピニオンリーダーになりうる人々の協力を勝ち取ろう。
外部のお墨付き(12) [External Validation]: 新しいアイデアの信憑性を上げるために、外部の情報を組織内に紹介しよう。
グループのアイデンティティ(13) [Group Identity]: 変化のための活動にアイデンティティを与えるために、活動を特徴づける名前を掲げ、人々がその存在を認識できるようにしよう。
達人を味方に(14) [Guru on Your Side]: 組織のメンバーに尊敬されているシニアレベルの人々にサポートを求めよう。
空間を演出する(15) [In Your Space]: 組織のあちこちに、それを見れば新しいアイデアのことを思い出すようなものを設置しよう。
イノベーター(16) [Innovator]: 変化への活動を始めるときは、新しもの好きな同僚の助けを求めよう。
やってみる(17) [Just Do It]: 新しいアイデアに関する話を広める前の準備段階として、まずは自分の仕事に使ってみて、そのアイデアの利点と限界を見極めよう。
感謝を伝える(18) [Just Say Thanks]: 感謝の気持ちを表すために、あなたに協力してくれたすべての人に、できるだけ誠実に「ありがとう」と言おう。
次のアクション(19) [Next Steps]: イベントの最後のほうで、新しいアイデアに対し、参加者が次に何ができそうかを確認する時間をとろう。
個人的な接触(20) [Personal Touch]: 新しいアイデアの価値を納得させるため、その人にとってどれだけ有用で、価値あるものなのかを示そう。
便乗(21) [Piggyback]: 新しいものを導入する戦略が障害にぶつかったときは、組織の慣習に便乗する方法を探そう。
種をまく(22) [Plant the Seeds]: 興味をかきたてるために、機会のあるときに資料（種）を持っていって、それらを見せる（蒔く）ようにしよう。
適切な時期(23) [The Right Time]: イベントの時期を決めたり、誰かに助けを求めるときには、タイミングをよく考えよう。
定期的な連絡(24) [Stay in Touch]: 一度キーパーソンに支援をもとめたら、彼らのことを忘れてはならないし、あなたも彼らに忘れられないようにしよう。
勉強会(25) [Study Group]: あるトピックについて継続的に探求したい・学びたい同僚を集めて、小さなグループを作ろう。
テイラーメイド(26) [Tailor Made]: 新しいアイデアで得ることができる価値を組織内の人々に納得させるため、組織のニーズに合わせてあなたのメッセージを作ろう。

中盤以降の活動に関わるパターン

著名人を招く(27) [Big Jolt]: 変化への取り組みがもっと注目されるよう、優れた経歴を持つ人を招いて、新しいアイデアについて語ってもらおう。
経営層の支持者(28) [Corporate Angel]: イノベーションと組織の目標が連携しやすくなるよう、経営層からの協力を得よう。
正式な推進担当者(29) [Dedicated Champion]: 新しいアイデアの組織導入の仕事をもっと効果的に進められるよう、担当業務の一部に組み入れることを提案しよう。
アーリーマジョリティ(30) [Early Majority]: 組織において、新しいアイデアに注力する方針を確立するには、マジョリティー(大多数)を納得させなければならない。

Fearless Change 48のパターン

達人のレビュー(31) [Guru Review]: マネージャーや他の開発者向けの新しいアイデアを評価してもらうために、味方となった達人や、関心を持っている同僚を集めよう。
体験談の共有(32) [Hometown Story]: 新しいアイデアの有用性を分かりやすく示すために、うまくいった人には経験の共有を促そう。
みんなを巻き込む(33) [Involve Everyone]: 新しいアイデアが組織全体で成功を収めるには、誰もがイノベーションを支える機会を持ち、それぞれに貢献できるようにすべきだ。
ちょうど十分(34) [Just Enough]: 新しいアイデアの難解なコンセプトを周囲の人に理解してもらうには、まずは簡単なイントロダクションから始め、受け入れ側の準備が整ってから追加の情報を伝えるようにしよう。
身近な支援者(35) [Local Sponsor]: 一番近いレベルのマネージャーに協力をまめよう。直属の上司が新しいアイデアの導入活動を支持してくれれば、もっと効率的に活動できる。
場所重要(36) [Location, Location, Location]: 割り込みが入ってイベントの流れを断ち切られる事態を避けるために、仕事の現場を離れて重要なイベントを開催してみよう。
メンター(37) [Mentor]: プロジェクトに新しいアイデアの導入を始めたいなら、そのアイデアを理解していて、チームを手助けできる人に来てもらおう。
謁見(38) [Royal Audience]: 組織の管理者やメンバーが、著名人を招く(27)の招待者と一緒に過ごす時間をお膳立てしよう。
相談できる同志(39) [Shoulder to Cry On]: 厳しい状況のとき必要以上に落ち込まないよう、新しいアイデアの導入で、同じように困っている人たちと話す機会を見つけよう。
成功の匂い(40) [Smell of Success]: あなたの取り組みがなんらかの目に見える結果を残した時、あなたと話したがる人がぞろぞろと現れるだろう。その機会を活かして教育に力を入れよう。
勢いの持続(41) [Sustained Momentum]: 組織内で、新しいアイデアに対する興味を持続させる、という現在進行形の仕事に積極的にアプローチしよう。
トークン(42) [Token]: 新しいアイデアを人の記憶に生かし続けるために、伝えた話題に紐付けられるトークンを渡そう。

抵抗と付き合うためのパターン

橋渡し役(43) [Bridge-Builder]: 新しいアイデアをすでに受け入れた人を、まだ受け入れていない人と引き合わせよう。
懐疑派代表(44) [Champion Skeptic]: あなたのアイデアに懐疑的なオピニオンリーダーに、「公式な懐疑派」の役割を演じてもらうよう、協力をお願いしよう。彼らの懐疑的な姿勢を変えられないとしても、あなたの取り組みを改善するために、その意見を活かそう。
根回し(45) [Corridor Politics]: 重要な決議が行われる前に、意思決定者や影響力のある人々に非公式に接触し、それぞれの意思決定の先に何が待っているのか、彼らが十分に理解しているかどうかを確かめよう。
怖れは無用(46) [Fear Less]: 抵抗勢力を新しいアイデアの強みに変えよう。
お試し期間(47) [Trial Run]: 組織が新しいアイデアの導入に前向きではない場合、少しの間だけ試験的に使ってもらい、その結果を観察することを提案しよう。
将軍の耳元でささやく(48) [Whisper in the General's Ear]: 集団の場ではマネージャーを説得しにくいことがある。その場合は他の人がいない場を設定して、懸念点を教えてもらう。

出典：『Fearless Change アジャイルに効く アイデアを組織に広めるための４８のパターン』
Mary Lynn Manns, Linda Rising 著
川口 恭伸 監訳
木村 卓央・高江洲 睦・高橋 一貴・中込 大祐・古屋 朝子・安井 力・山口 鉄平・米沢 仁・角 征典 訳

“48のパターンのチートシートを作りました”
http://kawaguti.hateblo.jp/entry/20140228/1393522489
2018/11/13 アクセス

知ってもらう

- 勉強会(25)
- 著名人を招く(27)
- トークン(42)

仲間を増やす

- コネクター(8)
- 相談できる同志(39)

説得する

- 達人を味方に(14)
- 将軍の耳元でささやく(48)
- 経営層の支持者(28)

導入の勘所： Fearless ChangeからのTips

新しいアイデアを導入するには

・知ってもらう

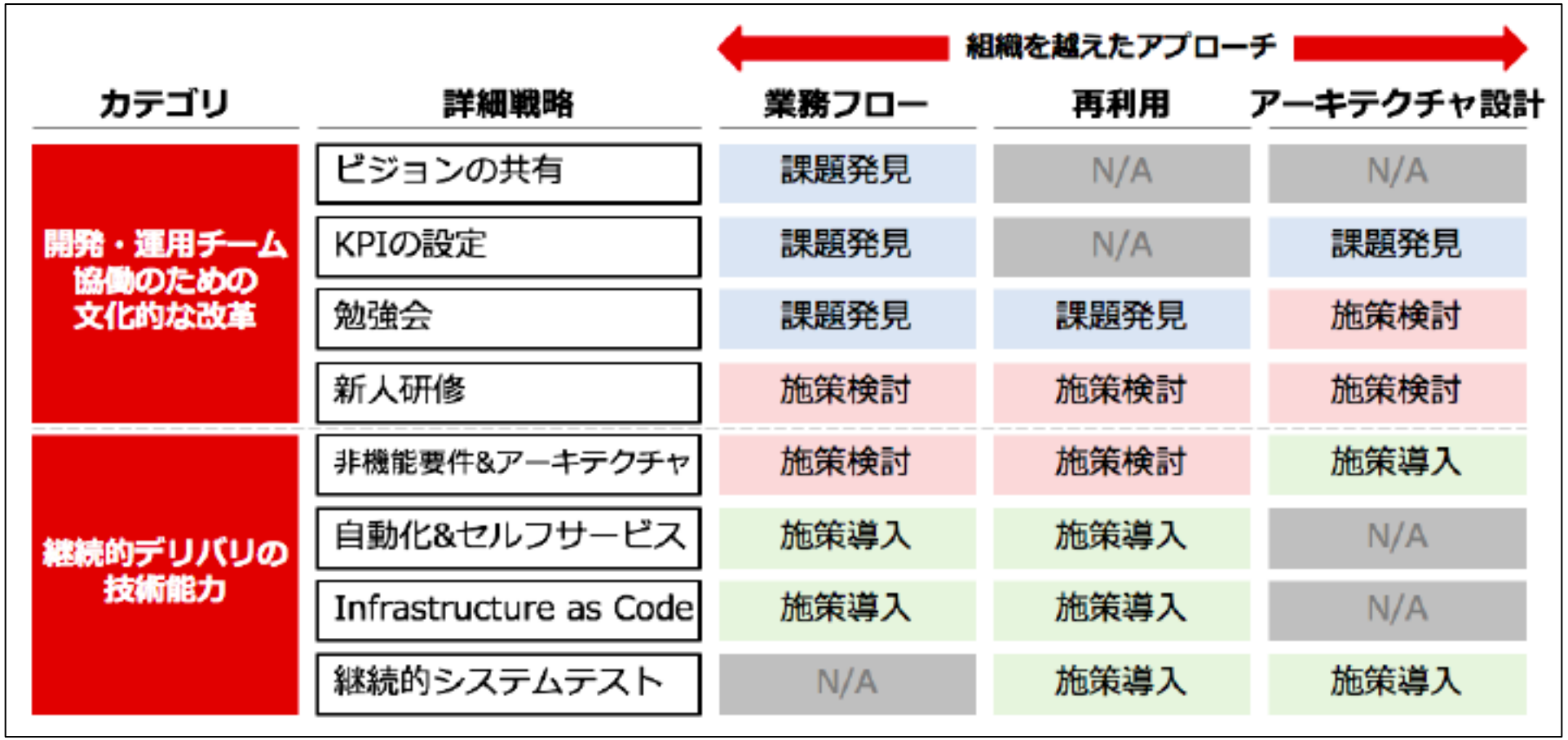
・仲間を増やす

・説得する

などの啓蒙活動が重要になる

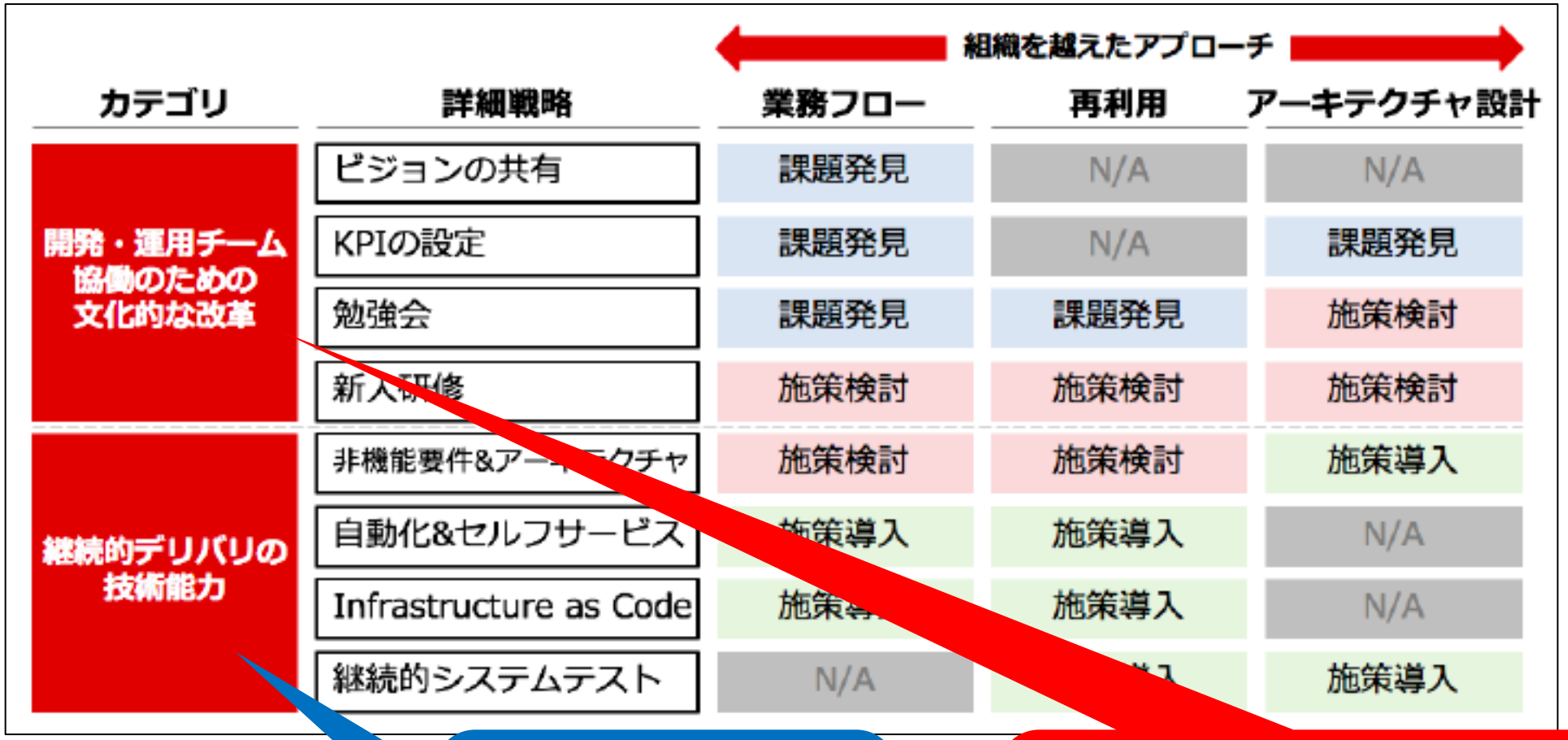
導入の勘所： 変革の軌跡の付録”楽天のDevOpsエンジニアのストーリー”

導入の勘所： 変革の軌跡の付録”楽天のDevOpsエンジニアのストーリー”



		組織を越えたアプローチ →		
カテゴリ	詳細戦略	業務フロー	再利用	アーキテクチャ設計
開発・運用チーム協働のための文化的な改革	ビジョンの共有	課題発見	N/A	N/A
	KPIの設定	課題発見	N/A	課題発見
	勉強会	課題発見	課題発見	施策検討
	新人研修	施策検討	施策検討	施策検討
継続的デリバリの技術能力	非機能要件&アーキテクチャ	施策検討	施策検討	施策導入
	自動化&セルフサービス	施策導入	施策導入	N/A
	Infrastructure as Code	施策導入	施策導入	N/A
	継続的システムテスト	N/A	施策導入	施策導入

導入の勘所： 変革の軌跡の付録”楽天のDevOpsエンジニアのストーリー”



組織を越えたアプローチ

カテゴリ	詳細戦略	業務フロー	再利用	アーキテクチャ設計
開発・運用チーム協働のための文化的な改革	ビジョンの共有	課題発見	N/A	N/A
	KPIの設定	課題発見	N/A	課題発見
	勉強会	課題発見	課題発見	施策検討
	新人研修	施策検討	施策検討	施策検討
継続的デリバリの技術能力	非機能要件&アーキテクチャ	施策検討	施策検討	施策導入
	自動化&セルフサービス	施策導入	施策導入	N/A
	Infrastructure as Code	施策導入	施策導入	N/A
	継続的システムテスト	N/A	施策導入	施策導入

自動化などの技術面

勉強会や研修などの文化面

導入の勘所： 変革の軌跡からのTips

新しいアイデアを導入するには

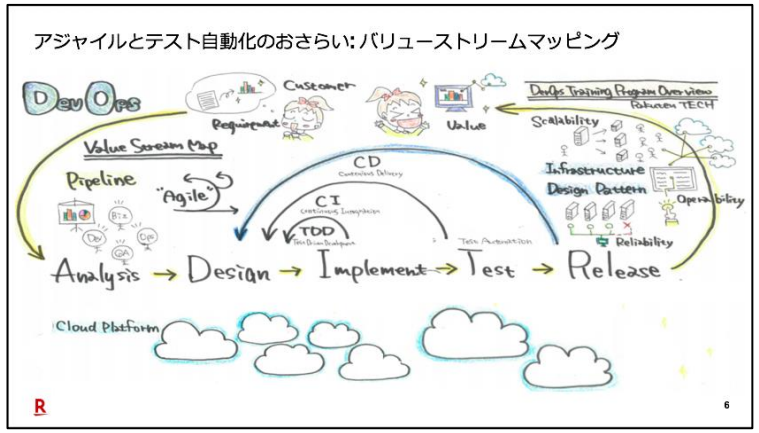
- 文化面
- 技術面

両面からの活動が重要になる

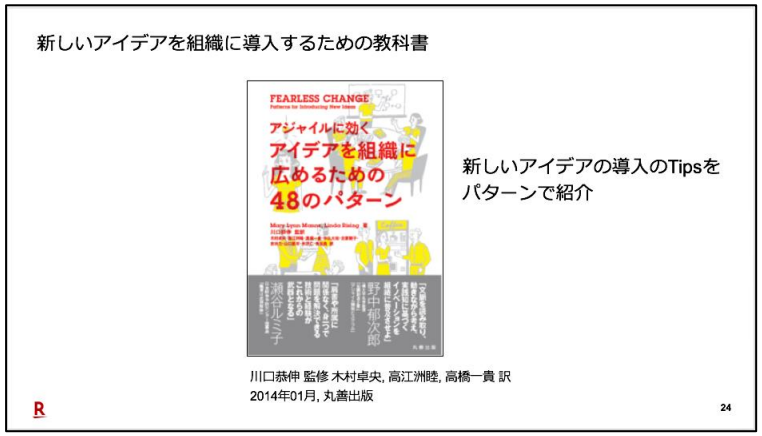
ということで、、、

ということで、、、

① おさらい

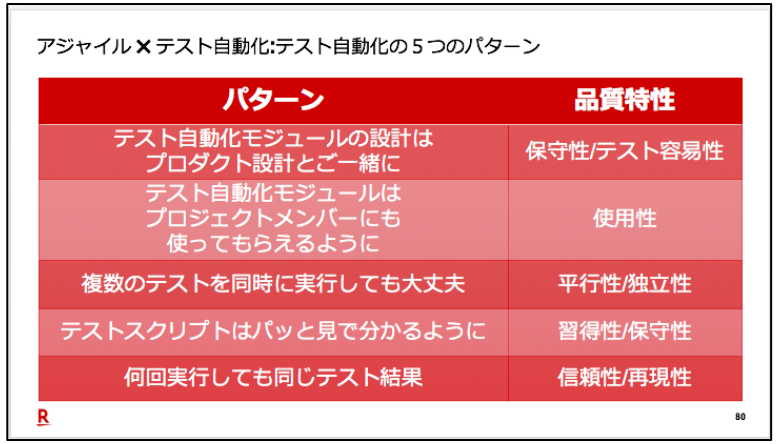


② 導入の勘所



③ 導入事例（文化面）

④ 導入事例（技術面）



ということで、、、

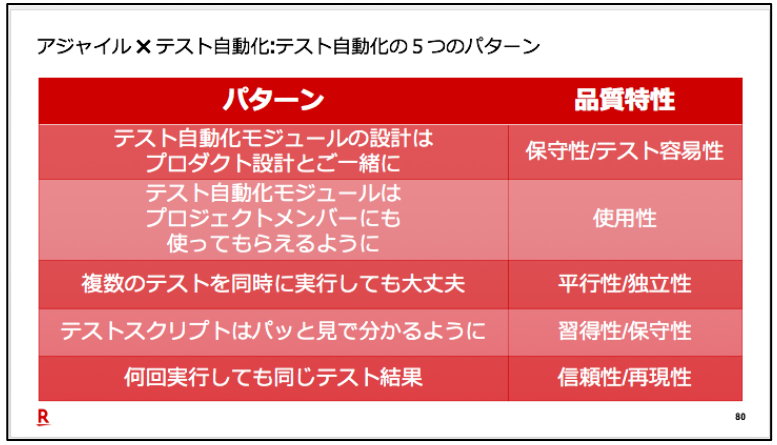
① おさらい

② 導入の勘所

本日の残り時間は、文化面と技術面での導入事例のお話をします

③ 導入事例（文化面）

④ 導入事例（技術面）



まとめ②

② 導入の勘所

新しいアイデアを組織に導入するための教科書

新しいアイデアの導入のTipsを
パターンで紹介

川口恭伸 監修 木村卓央, 高江洲睦, 高橋一貴 訳
2014年01月, 丸善出版

R

24

- **新しいアイデア導入のTips**

まとめ②

② 導入の勘所

新しいアイデアを組織に導入するための教科書

新しいアイデアの導入のTipsを
パターンで紹介

川口恭伸 監修 木村卓央, 高江洲睦, 高橋一貴 訳
2014年01月, 丸善出版

R

24

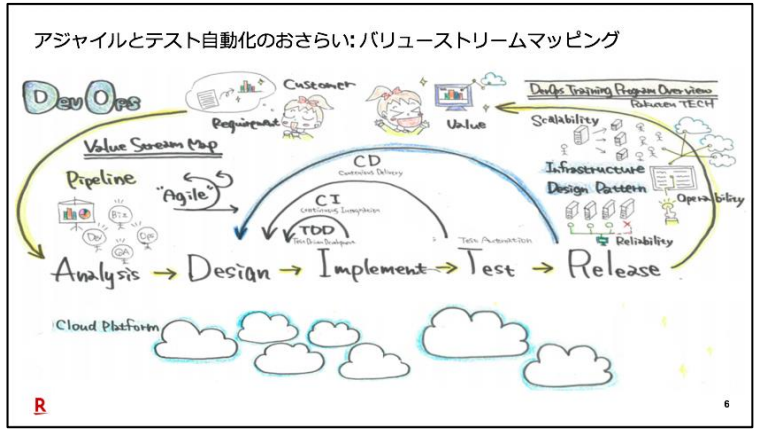
- **新しいアイデア導入のTips**

→仲間を増やす、説得するなど

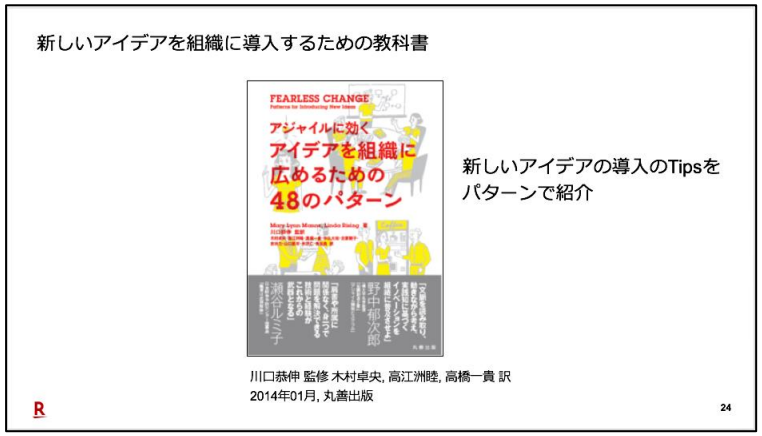
→文化面、技術面の両面から

本日お持ち帰りいただきたいこと

① おさらい

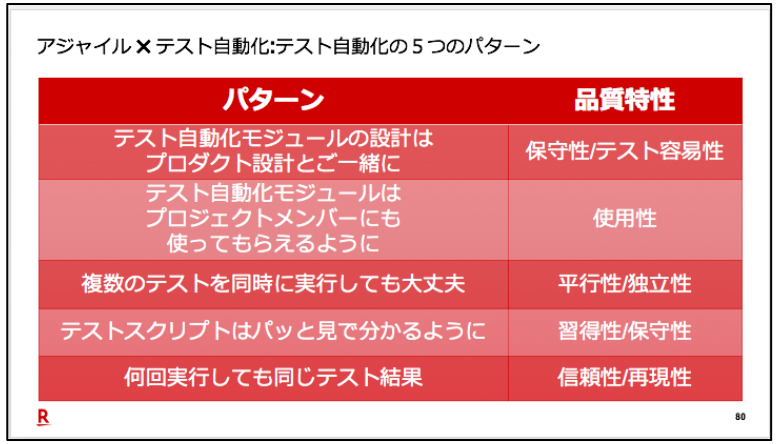


② 導入の勘所



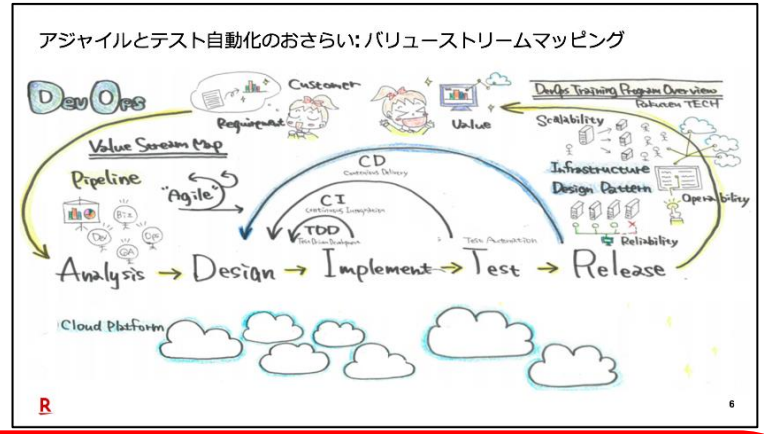
③ 導入事例（文化面）

④ 導入事例（技術面）

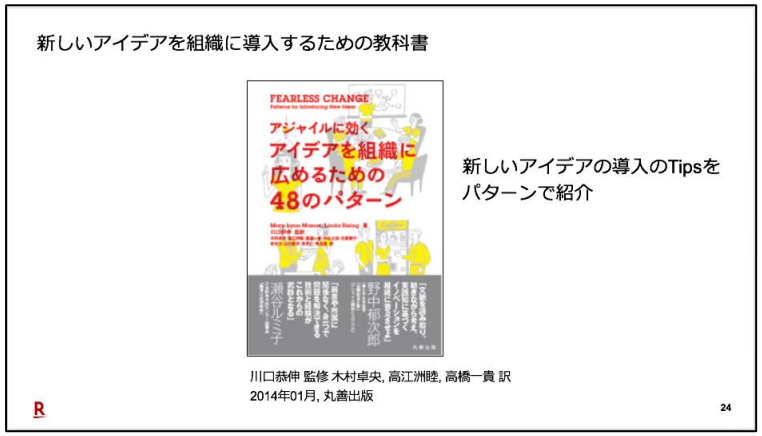


本日お持ち帰りいただきたいこと

① おさらい

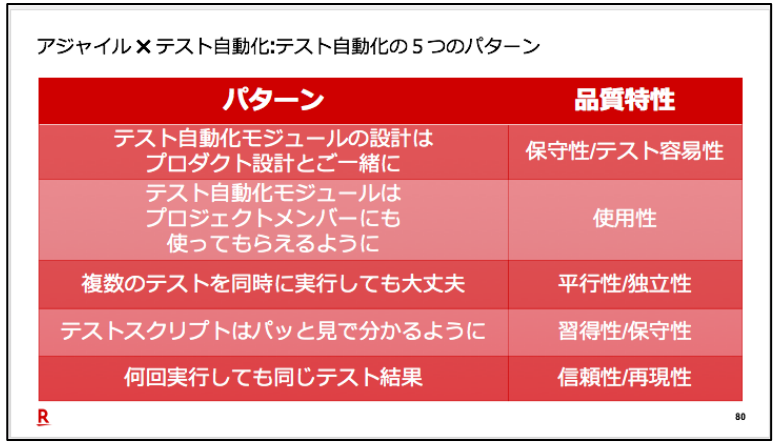


② 導入の勘所



③ 導入事例（文化面）

④ 導入事例（技術面）



③ 導入事例（文化面）


2018 エンジニアでもない私が達成したこと
DevOps
Training
Program
Rakuten
Technology
Conference

・組織を変える啓蒙活動

③ 導入事例（文化面）

導入事例(文化面)

恐れ知らずのブルドーザー改革

1. 小さなことからやってみる
2. やってみてダメなら変える
3. 巻き込めそうな人をどんどん巻き込む
4. XXは後からついてくる

2018 エンジニアでもない私が達成したこと

DevOps Training Program

Rakuten Technology Conference

DevOps研修の成果を表す数字たち

20 個以上の**研修**を約半年間で作成

30 人以上の技術/知識を持つエンジニアに**講師**を依頼

400 名以上が研修に**参加** (東京/札幌/仙台/大阪)

Tech Confの成果を示す数字たち

70 以上の**スピーカー**が集結

180 以上の楽天エンジニアが**ボランティア**で活躍

2000 以上の**参加者**

(東京/札幌/仙台/大阪/名古屋/仙台/中国/シンガポール/インド/アメリカ)

ここに至るまでの3年間

TECH HR英語研修担当
TOEIC/英会話レッスンの運営

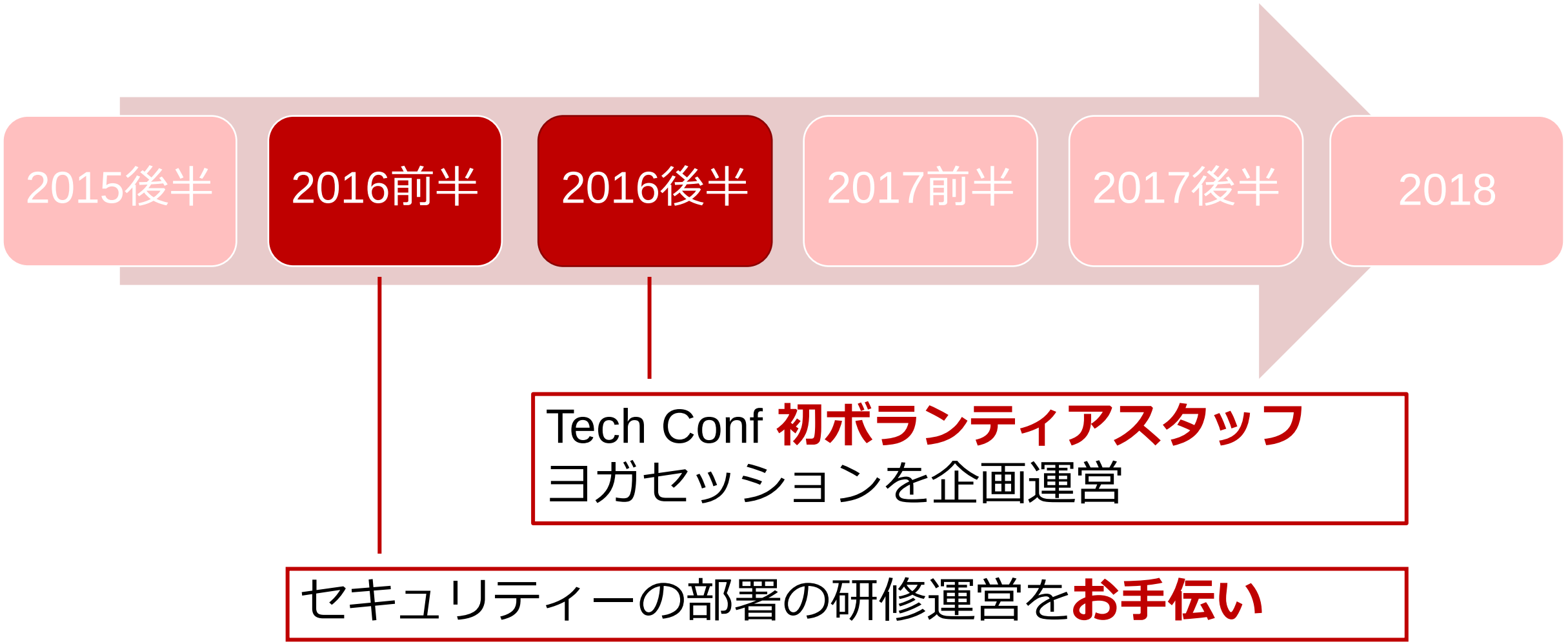
DevOps研修 企画運営
Tech Conf 2018 企画運営

アジャイルやテスト自動化、それ以外にも何か新しいことを
皆さんの組織に取り入れたいときの参考になることを祈りつつ、
この間、どのように業務内容や組織を変えて来たかをお話します。

恐れ知らずのブルドーザー改革

1. 小さなことからやってみる
2. やってみてダメなら変える
3. 巻き込めそうな人をどんどん巻き込む
4. XXは後からついてくる

2016 - 小さなことからやってみた



このとき大事にしてたこと

コラボ精神

セキュリティー研修

自分にできることを考えた

- HRオペレーションの提供
- HRデータの提供/管理

ヨガセッション

自分が巻き込める人を考えた

- インストラクター
- ドリンクスポンサー
- 空間デザイン(アロマ)

1:1のコラボレーションから
複数のコラボレーションへ

コラボを見た周囲の反応

「なんか盛り上がってるね！」

「楽しそうだね！」

「クオリティー高いね！」

→ **やりがいを感じる**

DevOps研修ができるまで１

テクニカルスキル研修の部署ができ、
協力・情報交換するようになる

→ **やりたいことが明確になる**

DevOps研修ができるまで２

テクニカルスキル研修の部署に**異動**

→ **エンジニアとコラボして研修を作成するスタイルが確立**

DevOps研修ができるまで３

DevOps研修担当になる　(Docker, テスト自動化, 継続的デリバリー, インフラデザインパターンの研修を企画運営)
Docker研修時には**ミートアップも開催**

→　**外部にも盛り上がりが広がり始める**

恐れ知らずのブルドーザー改革

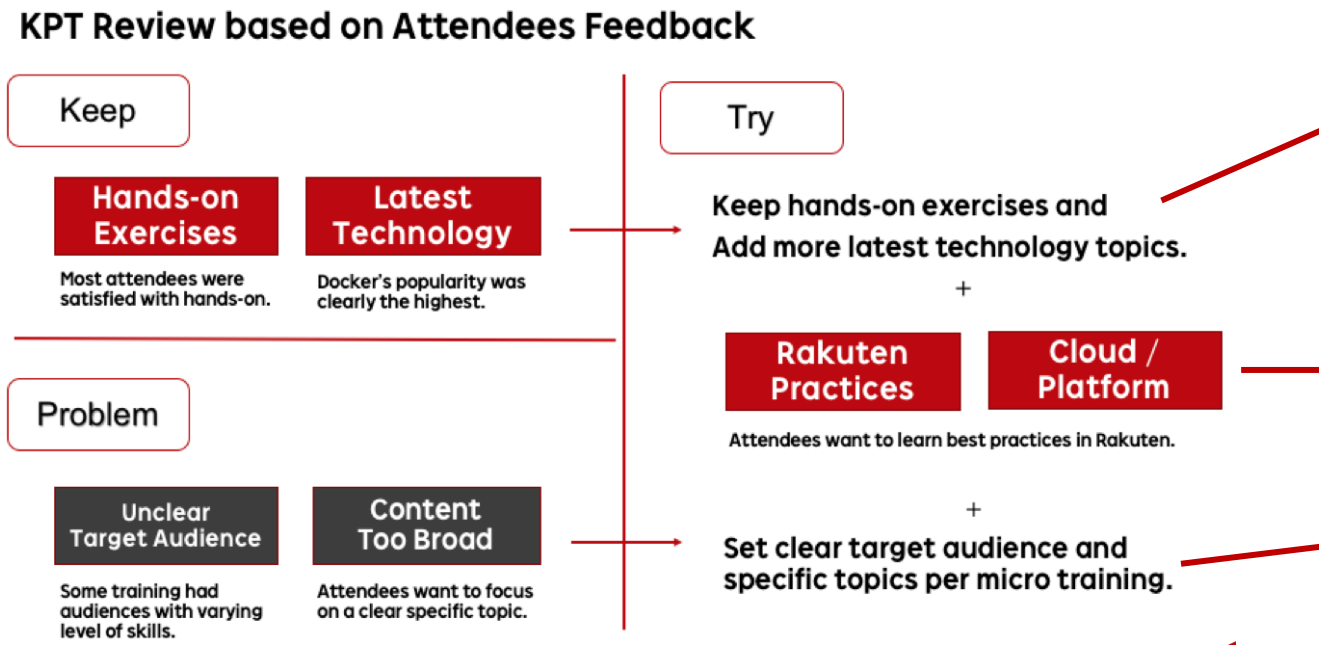
1. 小さなことからやってみる
2. やってみてダメなら変える
3. 巻き込めそうな人をどんどん巻き込む
4. XXは後からついてくる

DevOps研修ができるまで4

たくさん開発できたものの、参加者アンケート結果はイマイチ…

よりよい研修を目指して2018年、**研修計画を立て直す！**

2018Q1作成の報告書資料より抜粋



社員が求めていたのは
最新のトピックとハンズオン
に加えて、

- 楽天での**事例**/使用方法
- **クラウド**研修
  - パブリッククラウドのスキル
  - 社内でのクラウド使用方法
- 研修内容/参加対象者の**細分化**

DevOps研修ができるまで5

やるべきことを決めたものの・・・ **ハードルだらけ！**

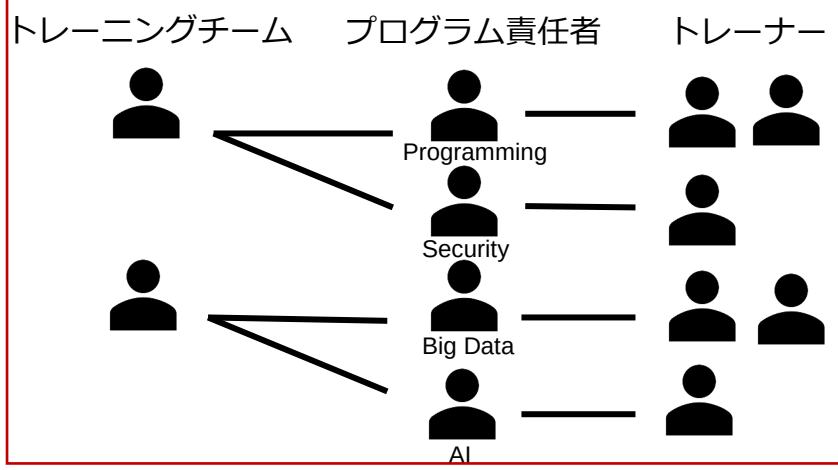
楽天での事例/使用方法 → 社内講師(または社内を知る講師)が必要

クラウド研修 → 社内にクラウド7種類

研修内容/参加対象者の細分化 → 細分化により研修数増加

→ 20以上の研修作成

研修プログラムの組織編成



研修作成者は2,3名
通常、3～6ヶ月かけて1つの研修を作成する

→ 一体何年かかる!???

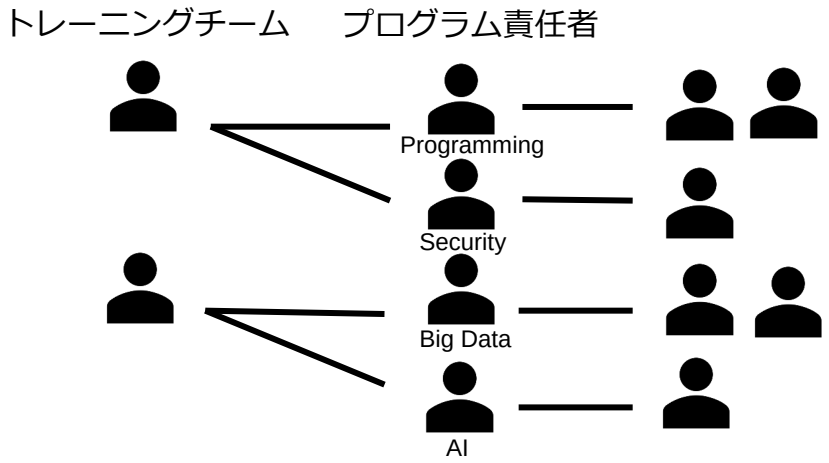
恐れ知らずのブルドーザー改革

1. 小さなことからやってみる
2. やってみてダメなら変える
3. **巻き込めそうな人をどんどん巻き込む**
4. XXは後からついてくる

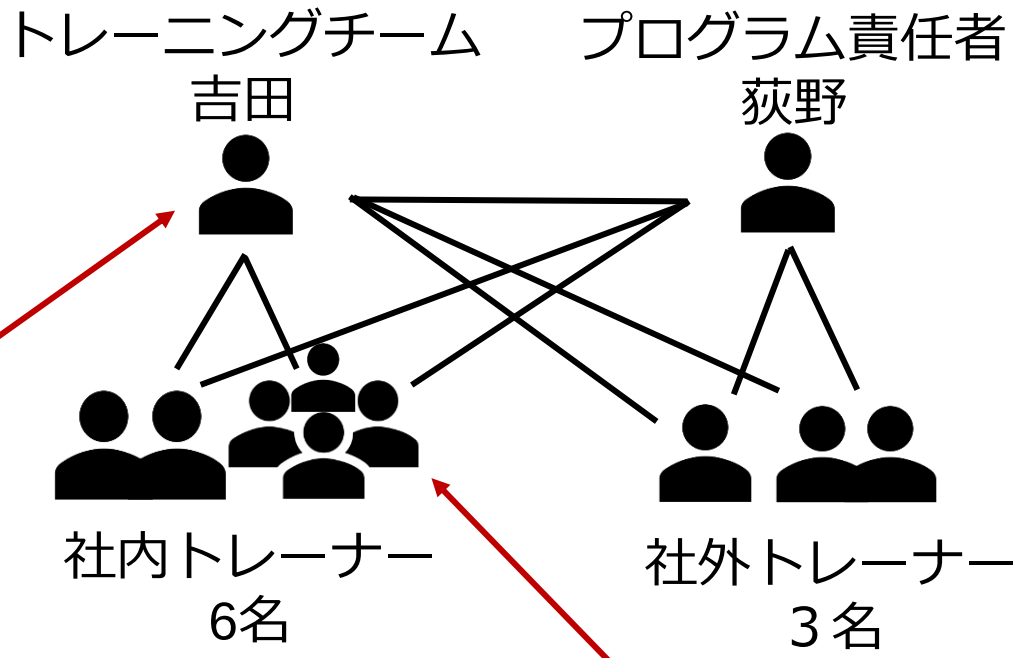
2017年DevOps研修プログラム組織編成

他のプログラムよりも**多くの人を巻き込み３ヶ月で４つ作成**

研修プログラムの組織編成



2017 DevOps研修プログラムの組織編成

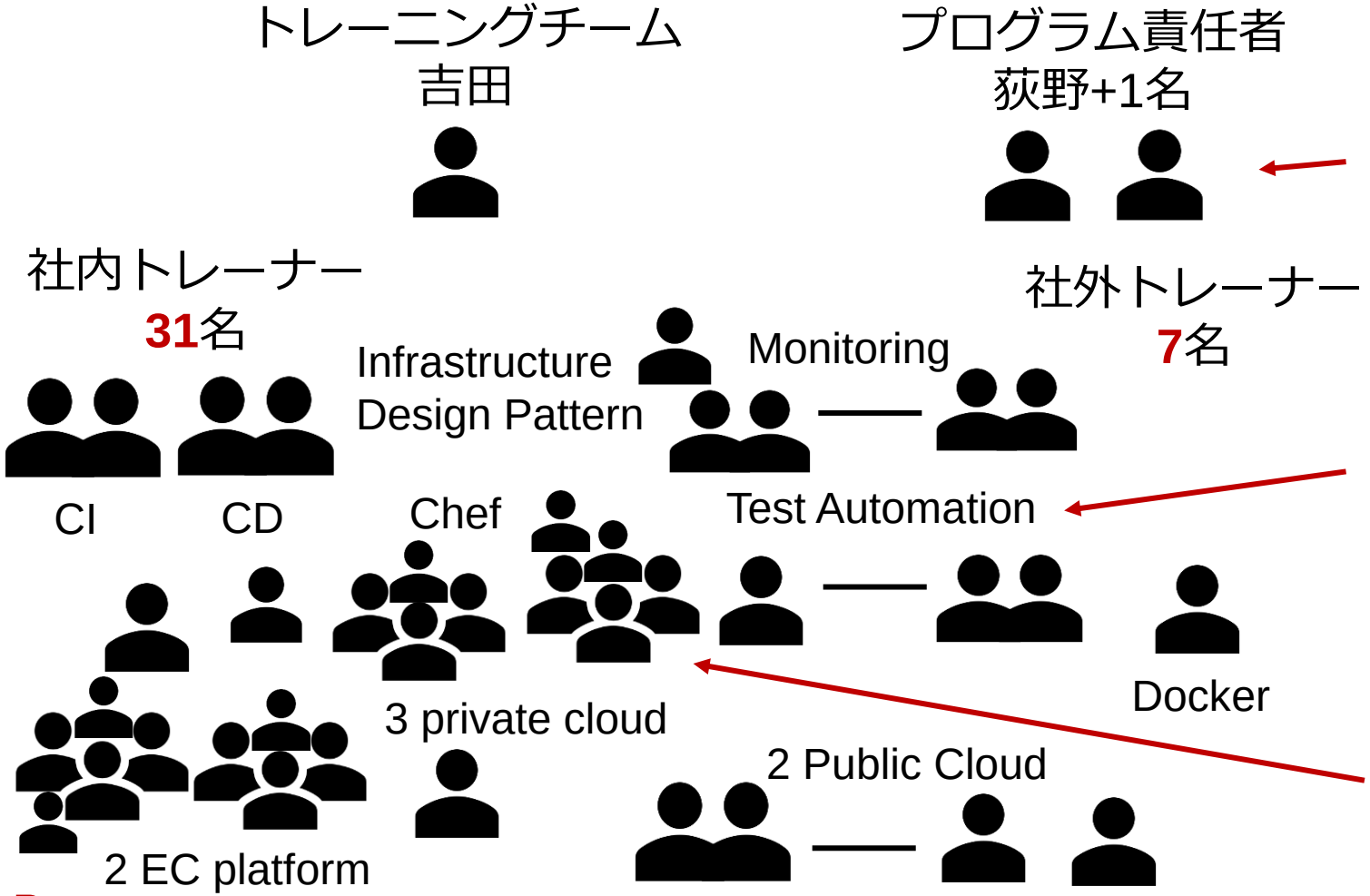


ポイント１
どの研修も**企画会議を運営**

ポイント２
トレーナーが**ひとりぼっちじゃない**

2018年DevOps研修プログラム組織編成

さらに多くの人を巻き込み半年で20個作成

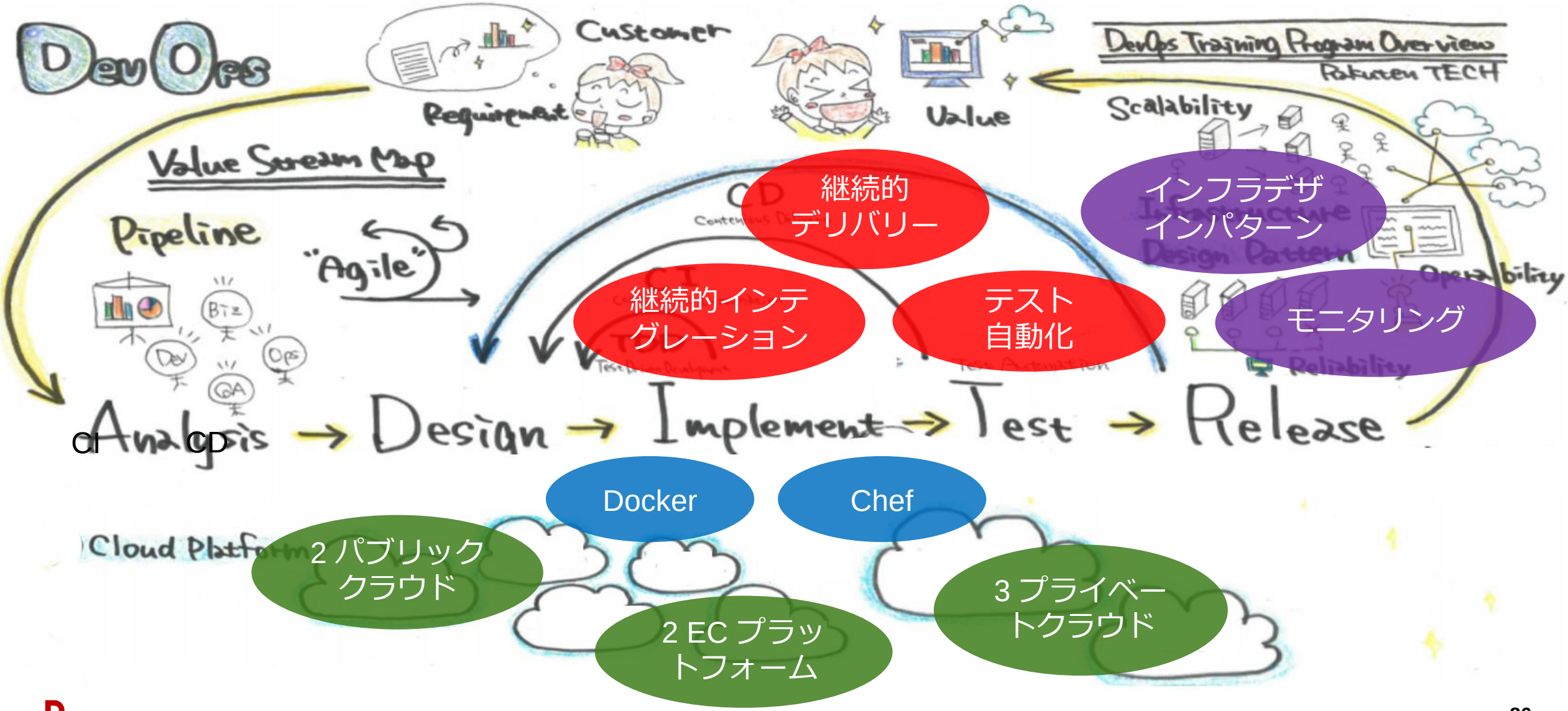


ポイント１
相方も増やした

ポイント２
社外の方ともコラボ

ポイント３
もちろん**ぼっちじゃない**

2018年DevOps研修プログラム



恐れ知らずのブルドーザー改革

1. 小さなことからやってみる
2. やってみてダメなら変える
3. 巻き込めそうな人をどんどん巻き込む
4. XXは後からついてくる

組織変更

- HRが扱う研修範囲の変化
- 他部署とのコラボ
- 社外講師とのコラボ
- トレーナー30名体制

人望

- 研修依頼の増加
- 質問/依頼の増加
- この人とやればできると言われることが増えた
- 他のことも一緒にやるようになった
- 何やら大きい話が入ってくるようになった

恐れ知らずのブルドーザー改革

1. 小さなことからやってみる
2. やってみてダメなら変える
3. 巻き込めそうな人をどんどん巻き込む
4. 組織変更・人望は後からついてくる

振り返ってみて

やってきたことを
まとめてみたら

Fearless Change

でした…！

全体に関わるパターン

エバンジェリスト(1) [Evangelist]: 新しいアイデアを組織に導入しはじめるなら、あなたの情熱を共有するため、できる限りのことをしよう。
小さな成功(2) [Small Successes]: 組織変革の取り組みをすすめるなかで、待ち受ける困難や膨大な作業に押しつぶされないよう、ほんの小さな成功でも、きちんと祝おう。
ステップバイステップ(3) [Step by Step]: 目標に向かって一歩一歩進めていくことで、組織変革の膨大な作業へのイライラを和らげよう。
予備調査(4) [Test the Waters]: 新しい好機が訪れた際に、興味があるかどうか調べるために、本書のパターンを利用して結果を評価しよう。
ふりかえりの時間(5) [Time for Reflection]: 過去から学ぶために、うまくいっていることや改めるべきことを評価するための時間を、定期的に確保しよう。

序盤の活動に関わるパターン

協力を求める(6) [Ask for Help]: 組織に新しいアイデアを導入するというのは、大変な大仕事だ。あなたのがんばりに協力してくれる人やリソースを探そう。
ブラウンバッグ・ミーティング(7) [Brown Bag]: 日常のランチタイムを、新しいアイデアを聞くための手軽で気軽な場として活用しよう。
コネクター(8) [Connector]: イノベーションのことを広く伝えるため、組織内にたくさんのコネクションを持つ人の協力を求めよう。
何か食べながら(9) [Do Food]: 食べ物を持ち込んで、いつもの集まりを特別なイベントにしよう。
電子フォーラム(10) [e-Forum]: もっと話を聞きたい人のために、電子掲示板・グループメールアドレス・メーリングリスト・書き込み可能なウェブサイトを用意しよう。
アーリーアダプター(11) [Early Adopter]: 新しいアイデアのオピニオンリーダーになりうる人々の協力を勝ち取ろう。
外部のお墨付き(12) [External Validation]: 新しいアイデアの信憑性を上げるために、外部の情報を組織内に紹介しよう。
グループのアイデンティティ(13) [Group Identity]: 変化のための活動にアイデンティティを与えるために、活動を特徴づける名前を掲げ、人々がその存在を認識できるようにしよう。
達人を味方に(14) [Guru on Your Side]: 組織のメンバーに尊敬されているシニアレベルの人々にサポートを求めよう。

空間を演出する(15) [In Your Space]: 組織のあちこちに、それを見れば新しいアイデアのことを思い出すようなものを設置しよう。
イノベーター(16) [Innovator]: 変化への活動を始めるときは、新しもの好きな同僚の助けを求めよう。
やってみる(17) [Just Do It]: 新しいアイデアに関する話を広める前の準備段階として、まずは自分の仕事に使ってみて、そのアイデアの利点と限界を見極めよう。
感謝を伝える(18) [Just Say Thanks]: 感謝の気持ちを表すために、あなたに協力してくれたすべての人に、できるだけ誠実に「ありがとう」と言おう。
次のアクション(19) [Next Steps]: イベントの最後のほうで、新しいアイデアに対し、参加者が次に何ができそうかを確認する時間をとろう。
個人的な接触(20) [Personal Touch]: 新しいアイデアの価値を納得させるため、その人にとってどれだけ有用で、価値あるものなのかを示そう。
便乗(21) [Piggyback]: 新しいものを導入する戦略が障害にぶつかったときは、組織の慣習に便乗する方法を探そう。
種をまく(22) [Plant the Seeds]: 興味をかきたてるために、機会のあるときに資料（種）を持っていって、それらを見せる（蒔く）ようにしよう。
適切な時期(23) [The Right Time]: イベントの時期を決めたり、誰かに助けを求めるときには、タイミングをよく考えよう。
定期的な連絡(24) [Stay in Touch]: 一度キーパーソンに支援を求めたら、彼らのことを忘れてはならないし、あなたも彼らに忘れられないようにしよう。
勉強会(25) [Study Group]: あるトピックについて継続的に探求したい・学びたい同僚を集めて、小さなグループを作ろう。
テイラーメイド(26) [Tailor Made]: 新しいアイデアで得ることができる価値を組織内の人々に納得させるため、組織のニーズに合わせてあなたのメッセージを作ろう。

中盤以降の活動に関わるパターン

著名人を招く(27) [Big Jolt]: 変化への取り組みがもっと注目されるよう、優れた経歴を持つ人を招いて、新しいアイデアについて語ってもらおう。
経営層の支持者(28) [Corporate Angel]: イノベーションと組織の目標が連携しやすくなるよう、経営層からの協力を得よう。
正式な推進担当者(29) [Dedicated Champion]: 新しいアイデアの組織導入の仕事をもっと効果的に進められるよう、担当業務の一部に組み入れることを提案しよう。
アーリーマジョリティ(30) [Early Majority]: 組織において、新しいアイデアに注力する方針を確立するには、マジョリティー(大多数)を納得させなければならない。

達人のレビュー(31) [Guru Review]: マネージャーや他の開発者向けの新しいアイデアを評価してもらうために、味方となった達人や、関心を持っている同僚を集めよう。
体験談の共有(32) [Hometown Story]: 新しいアイデアの有用性を分かりやすく示すために、うまくいった人には経験の共有を促そう。
みんなを巻き込む(33) [Involve Everyone]: 新しいアイデアが組織全体で成功を収めるには、誰もがイノベーションを支える機会を持ち、それぞれに貢献できるようにすべきだ。
ちょうど十分(34) [Just Enough]: 新しいアイデアの難解なコンセプトを周囲の人に理解してもらうには、まずは簡単なイントロダクションから始め、受け入れ側の準備が整ってから追加の情報を伝えるようにしよう。
身近な支援者(35) [Local Sponsor]: 一番近いレベルのマネージャーに協力を求めよう。直属の上司が新しいアイデアの導入活動を支持してくれれば、もっと効率的に活動できる。
場所重要(36) [Location, Location, Location]: 割り込みが入ってイベントの流れを断ち切られる事態を避けるために、仕事の現場を離れて重要なイベントを開催してみよう。
メンター(37) [Mentor]: プロジェクトに新しいアイデアの導入を始めたいなら、そのアイデアを理解していて、チームを手助けできる人に来てもらおう。
謁見(38) [Royal Audience]: 組織の管理者やメンバーが、著名人を招く(27)の招待者と一緒に過ごす時間をお膳立てしよう。
相談できる同志(39) [Shoulder to Cry On]: 厳しい状況のとき必要以上に落ち込まないよう、新しいアイデアの導入で、同じように困っている人たちと話す機会を見つけよう。
成功の匂い(40) [Smell of Success]: あなたの取り組みがなんらかの目に見える結果を残した時、あなたと話したがる人がぞろぞろと現れるだろう。その機会を活かして教育に力を入れよう。
勢いの持続(41) [Sustained Momentum]: 組織内で、新しいアイデアに対する興味を持続させる、という現在進行形の仕事に積極的にアプローチしよう。
トークン(42) [Token]: 新しいアイデアを人の記憶に生かし続けるために、伝えた話題に紐付けられるトークンを渡そう。

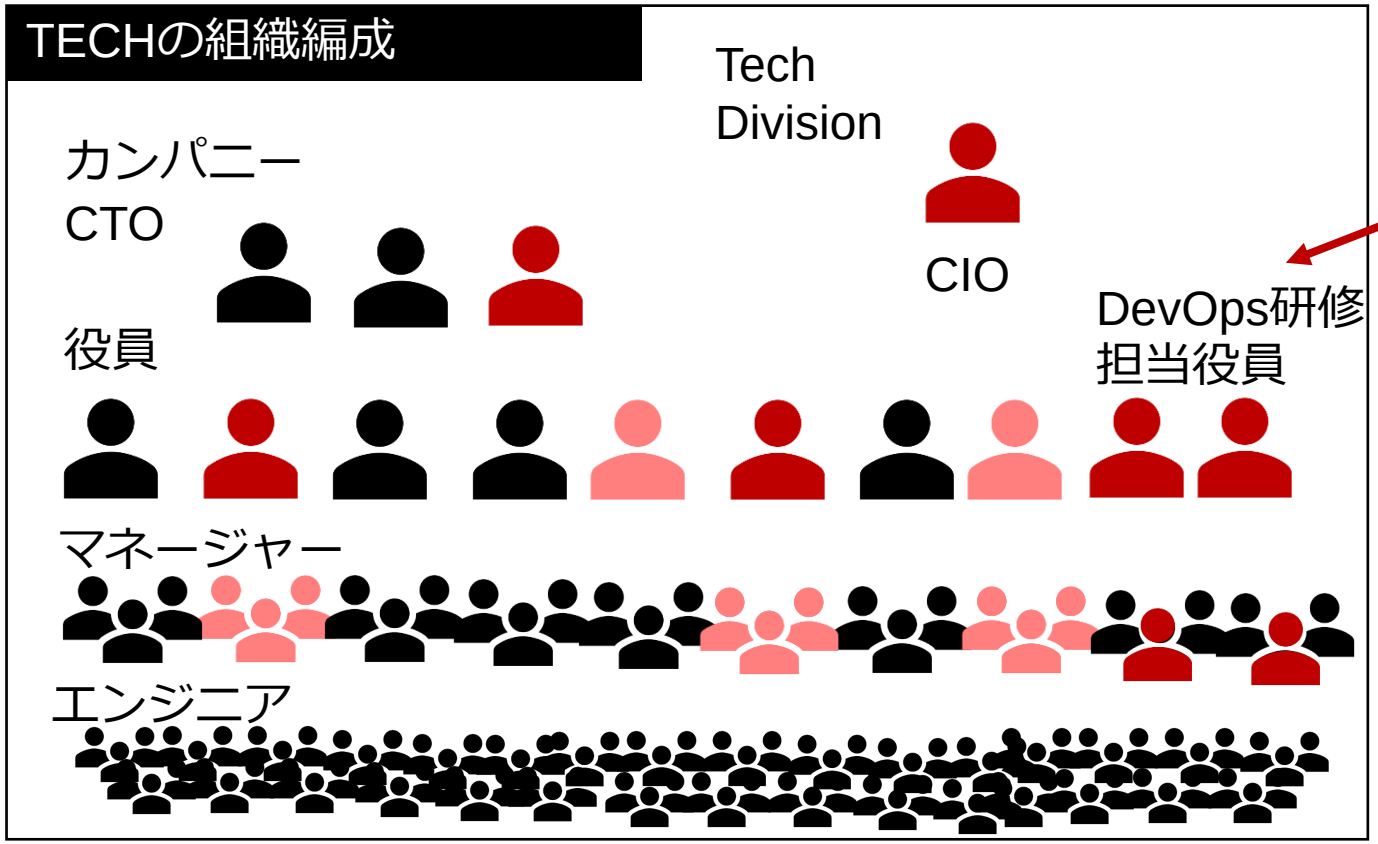
抵抗と付き合うためのパターン

橋渡し役(43) [Bridge-Builder]: 新しいアイデアをすでに受け入れた人を、まだ受け入れていない人と引き合わせよう。
懐疑派代表(44) [Champion Skeptic]: あなたのアイデアに懐疑的なオピニオンリーダーに、「公式な懐疑派」

の役割を演じてもらうよう、協力をお願いしよう。彼らの懐疑的な姿勢を変えられないとしても、あなたの取り組みを改善するために、その意見を活かそう。
根回し(45) [Corridor Politics]: 重要な決議が行われる前に、意思決定者や影響力のある人々に非公式に接触し、それぞれの意思決定の先に何が待っているのか、彼らが十分に理解しているかどうかを確かめよう。
怖れは無用(46) [Fear Less]: 抵抗勢力を新しいアイデアの強みに変えよう。
お試し期間(47) [Trial Run]: 組織が新しいアイデアの導入に前向きではない場合、少しの間だけ試験的に使ってもらい、その結果を観察することを提案しよう。
将軍の耳元でささやく(48) [Whisper in the General's Ear]: 集団の場ではマネージャーを説得しにくいことがある。その場合は他の人がいない場を設定して、懸念点を教えてもらう。

出典:『Fearless Change アジャイルに効く
アイデアを組織に広めるための４８のパターン』
Mary Lynn Manns, Linda Rising 著
川口 恭伸 監訳
木村 卓央・高江洲 睦・高橋 一貴・
中込 大祐・古屋 朝子・安井 力・
山口 鉄平・米沢 仁・角 征典 訳

組織全体から理解を得るために(マネージャーレベル)



アピールした！

達人を味方に(14)

種をまく(22)

将軍の耳元でささやく(48)

経営層の支持者(28)

相方を作った！

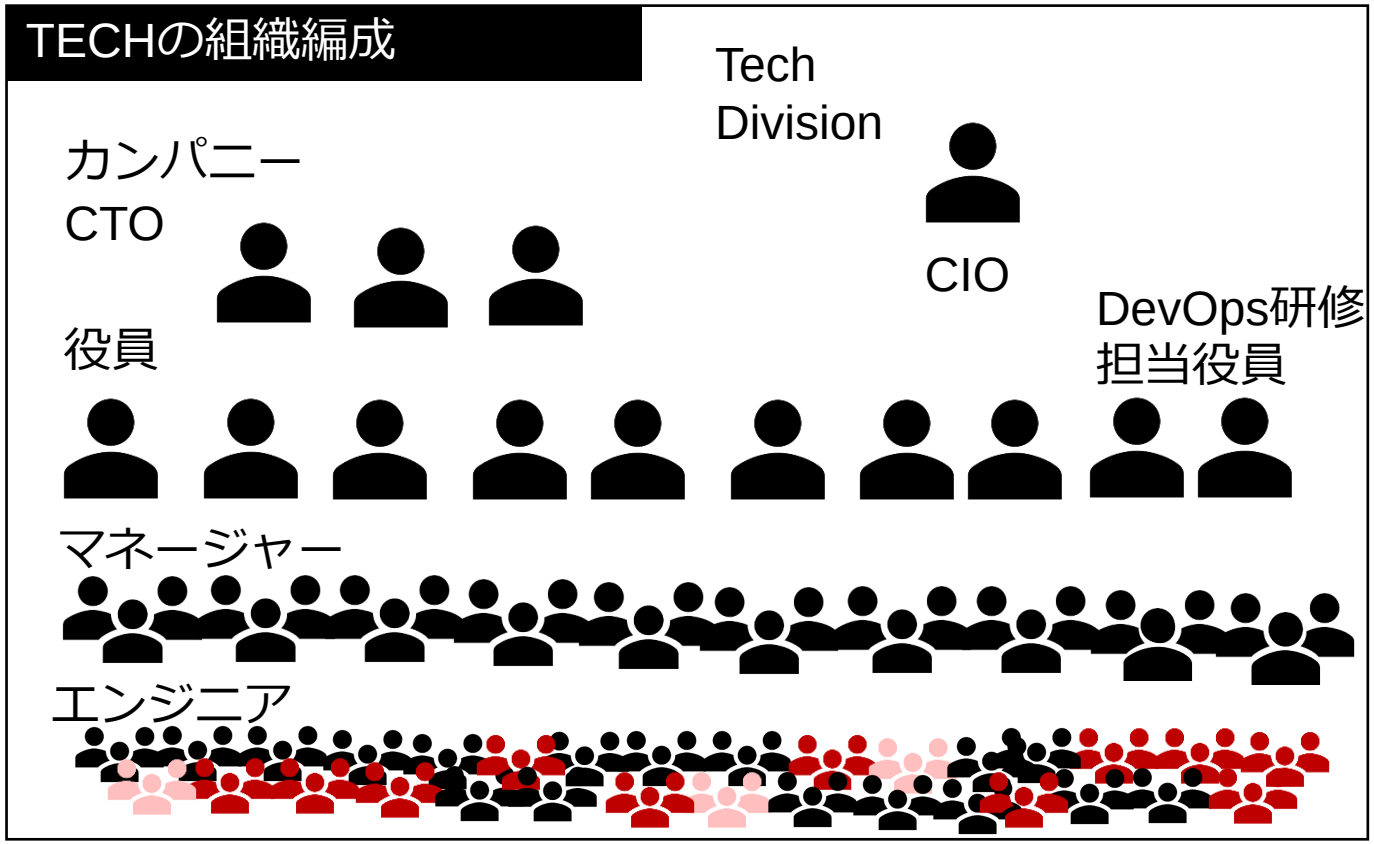
コネクター(8)

達人のレビュー(31)

相談できる同志(39)

実はこんなことをしていました2/4

組織全体から理解を得るために(エンジニアレベル)



マーケティングした！

電子フォーラム(10)

トークン(42)

イベントやった！

勉強会(25)

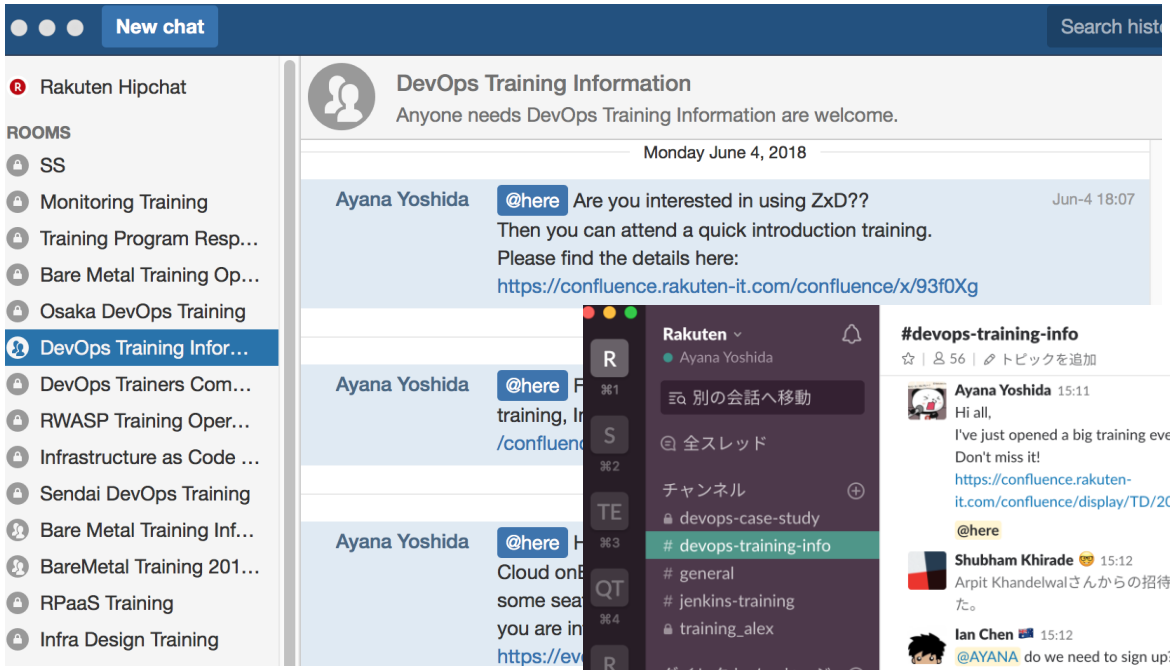
著名人を招く(27)

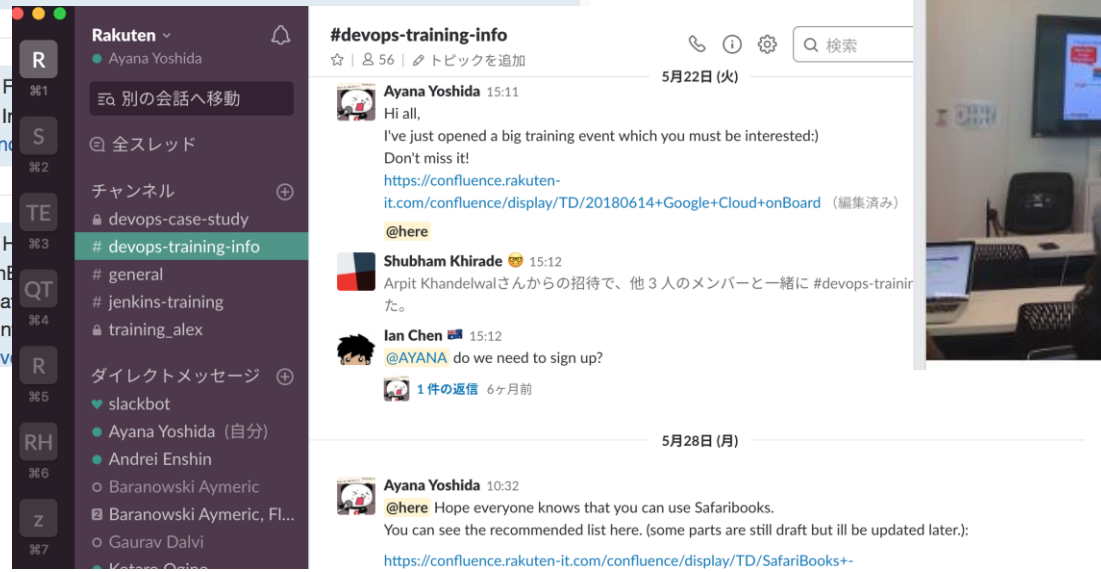
マーケティング事例

Facebookはもちろん**こまめに投稿**→

↓**あらゆる**コミュニケーションツールを駆使

社内掲示板, Viber, Yammer, Hipchat, Slack, …





イベント事例

ハッシュタグ ： #dockertokyo

募集内容	参加枠1 無料	先着順（抽選終了） 323/100人

イベントの説明

Docker Meetup Tokyo #19 (DockerCon EU 17 updates)

応募枠増席しました（2017/10/6 17:40）

(正直なところ)滅多に来日しないDocker社の中の人が来日するのでちょうど良いのでMeetupをやっちゃおうよ。ということで Docker Meetup Tokyo を楽天さんの会場をお借りして開催することになりました。

https://dockerjp.connpass.com/event/68798/

2018/11/23 アクセス

実はこんなことをしていました 3/4

たくさんの人を巻き込むために

飲みに誘いまくった！

みんなを巻き込む(33)

恐れは無用(46)

何か食べながら(9)

協力を求める(6)

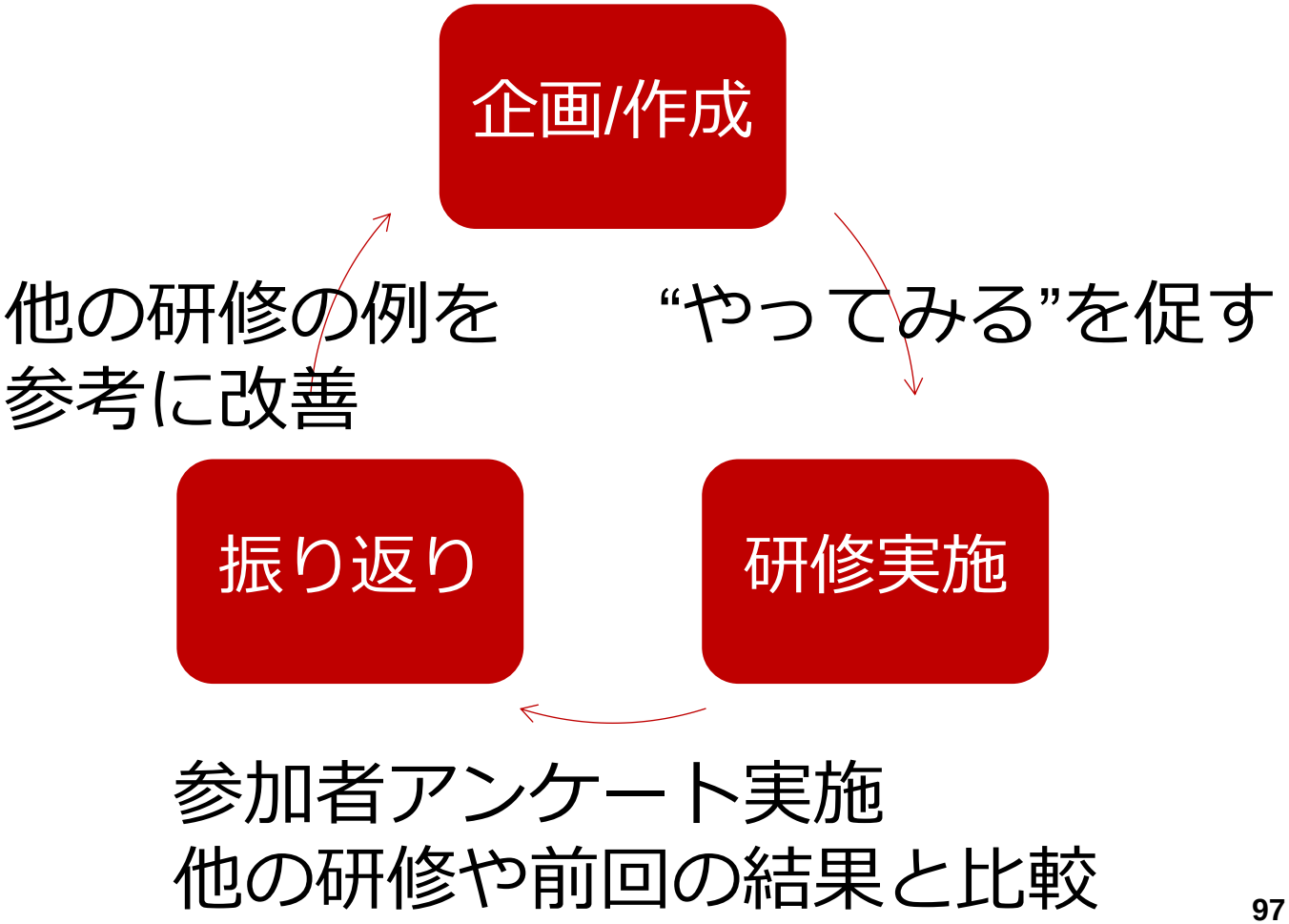
グループのアイデンティティ(13)

ミーティングより深い話
トレーナー同志も繋がる

巻き込まれた人たちがやりがいを感じられるように

仕組みを作った！

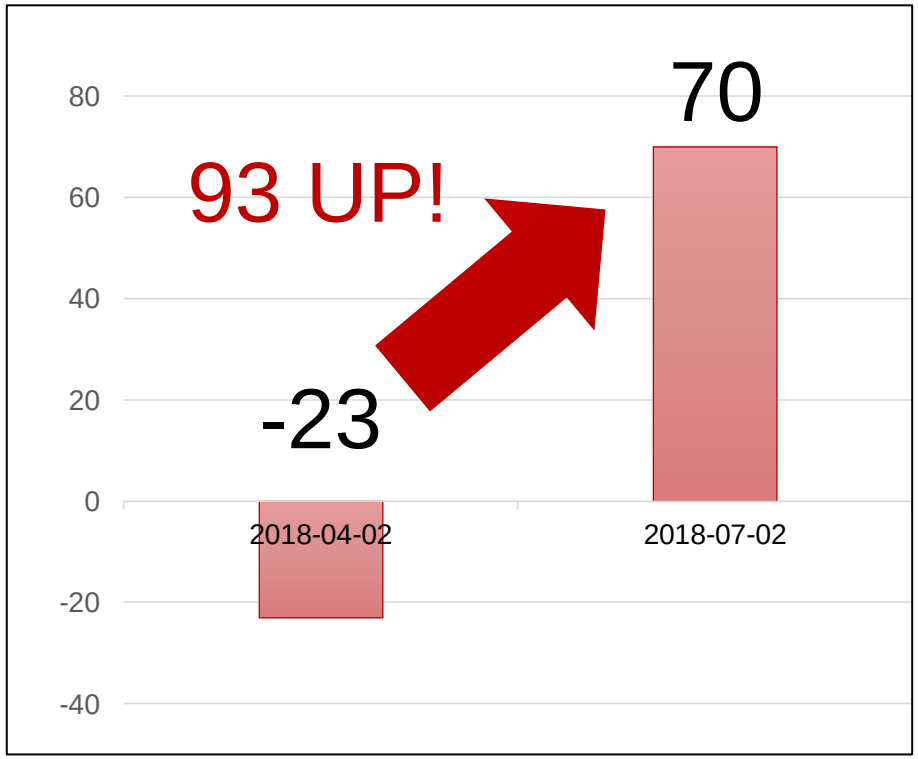
やってみる(17)
振り返りの時間(5)
小さな成功(2)
次のアクション(19)
体験談の共有(32)

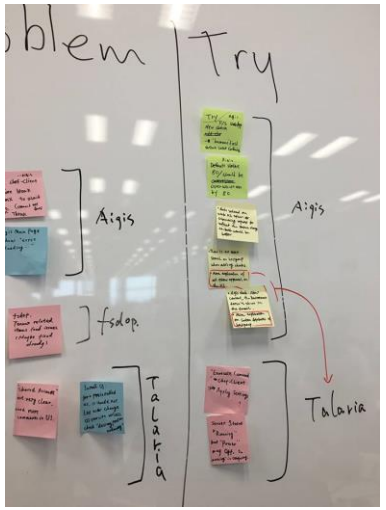


NPSが信じられないくらい上がった事例

*NPS＝参加者満足度を表す数値(-100から+100)
参加者の満足度はトレーナーの自信に不可欠

とある研修で１回目と２回目の結果が大幅UP





やったこと

1) 問題点の洗い出し
2) 他の研修のいい例を提案
3) 自部署対象に**トライアル**
4) 振り返りミーティング
5) 改善点の洗い出し

振り返ってみて

Fearless Change以外にも、まだあった！！

Fearless Change

Fearless Change 48のパターン

全体に関わるパターン

エバンジェリスト(1) [Evangelist]: 新しいアイデアを組織に導入しはじめるなら、あなたの情熱を共有するため、できる限りのことをしよう。

小さな成功(2) [Small Successes]: 組織変革の取り組みをするなかで、待ち受ける困難や膨大な作業に押しつぶされないよう、ほんの小さな成功でも、きちんと祝おう。

ステップバイステップ(3) [Step by Step]: 目標にむかって一歩一歩進めていくことで、組織変革の膨大な作業へのイライラを和らげよう。

予備調査(4) [Test the Waters]: 新しい好機が訪れた際に、興味があるかどうか探べるために、本書のパターンを利用して結果を評価しよう。

ふりかえりの時間(5) [Time for Reflection]: 過去から学ぶために、うまくいっていることや改めるべきことを評価するための時間を、定期的に確保しよう。

序盤の活動に関わるパターン

協力を求める(6) [Ask for Help]: 組織に新しいアイデアを導入するというのは、大変な大仕事だ。あなたのがんばりに協力してくれる人やリソースを探そう。

ブラウンバッグ・ミーティング(7) [Brown Bag]: 日常のランチタイムを、新しいアイデアを聞くための手軽で気軽な場として活用しよう。

コネクター(8) [Connector]: イノベーションのことを広く伝えるため、組織内にたくさんのコネクションを持つ人の協力を求めよう。

何か食べながら(9) [Do Food]: 食べ物を持ち込んで、いつもの集まりを特別なイベントにしよう。

電子フォーラム(10) [e-Forum]: もっと話を聞きたい人のために、電子掲示板・グループメールアドレス・メーリングリスト・書き込み可能なウェブサイトを用意しよう。

アーリーアダプター(11) [Early Adopter]: 新しいアイデアのオピニオンリーダーになりうる人々の協力を勝ち取ろう。

外部のお墨付き(12) [External Validation]: 新しいアイデアの信憑性を上げるために、外部の情報を組織内に紹介しよう。

グループのアイデンティティ(13) [Group Identity]: 変化のための活動にアイデンティティをうえるために、活動を特徴づける名前を掲げ、人々がその存在を認識できるようにしよう。

達人を味方に(14) [Guru on Your Side]: 組織のメンバーに尊敬されているシニアレベルの人々にサポートを求めよう。

空間を演出する(15) [In Your Space]: 組織のあちこちに、それを見れば新しいアイデアのことを思い出すようなものを設置しよう。

イノベーター(16) [Innovator]: 変化への活動を始めるときは、新しものの好きな同僚の協力を求めよう。

やってみる(17) [Just Do It]: 新しいアイデアに関する話を広める前の準備段階として、まずは自分の仕事に従ってみて、そのアイデアの利点と限界を見極めよう。

感謝を伝える(18) [Just Say Thanks]: 感謝の気持ちを表すために、あなたに協力してくれたすべての人に、できるだけ誠実に「ありがとう」と言おう。

次のアクション(19) [Next Steps]: イベントの最後のほうで、新しいアイデアに対し、参加者が次に何ができるかを確認する時間をとろう。

個人的な接触(20) [Personal Touch]: 新しいアイデアの価値を納得させるため、その人にとってどれだけ有用で、価値あるものなのかを示そう。

便乗(21) [Piggyback]: 新しいものを導入する戦略が障害に出つかったときは、組織の慣習に便乗する方法を探そう。

種をまく(22) [Plant the Seeds]: 興味をかきたてるために、概念のあると言に資料（種）を持っていって、それらを見せる（蒔く）ようにしよう。

適切な時期(23) [The Right Time]: イベントの時期を決めたり、誰かに助けを求めるときには、タイミングをよく考えよう。

定期的な連絡(24) [Stay in Touch]: 一度キーパーソンに支援を求めたら、彼らのことを忘れてはならない。あなたも彼らに忘れられないようにしよう。

勉強会(25) [Study Group]: あるトピックについて継続的に学びたい・学びたい同僚を集めて、小さなグループを作ろう。

テイラーメイド(26) [Tailor Made]: 新しいアイデアで得ることができる価値を組織内の人々に納得させるため、組織のニーズに合わせてあなたのメッセージを作ろう。

中盤以降の活動に関わるパターン

著名人を招く(27) [Big Jolt]: 変化への取り組みがもっと注目されるよう、優れた経歴を持つ人を招いて、新しいアイデアについて話してもらおう。

経営層の支持者(28) [Corporate Angel]: イノベーションと組織の目標が達成しやすくなるよう、経営層からの協力を得よう。

正式な推進担当者(29) [Dedicated Champion]: 新しいアイデアの組織導入の仕事をもっと効果的に進められるよう、担当業務の一部に組み入れることを提案しよう。

アーリーマジョリティ(30) [Early Majority]: 組織において、新しいアイデアに注力する方針を確立するには、マジョリティー（大多数）を納得させなければならない。

Fearless Change 48のパターン

達人のレビュー(31) [Guru Review]: マネージャーや他の開発者向けの新しいアイデアを評価してもらうために、味方となった達人や、関心を持っている同僚を集めよう。

休憩時の共有(32) [Hometown Story]: 新しいアイデアの有用性を分かりやすく示すために、うまくいった人には経験の共有を促そう。

みんなを巻き込む(33) [Involve Everyone]: 新しいアイデアが組織全体で成功を収めるには、皆もがイノベーションを支える機会を持ち、それぞれに貢献できるようにすべきだ。

ちょうど十分(34) [Just Enough]: 新しいアイデアの最低なコンセプトを周囲の人に理解してもらうには、まずは簡単なイントロダクションから始め、受け入れ側の準備が整ってから追加の情報を伝えるようにしよう。

身近な支援者(35) [Local Sponsor]: 一番近いレベルのマネージャーに協力を求めよう。直属の上司が新しいアイデアの導入活動を支持してくれれば、もっと効率的に活動できる。

場所重要(36) [Location, Location, Location]: 割り込みが入ってイベントの流れを断ち切られる事態を避けるために、仕事の現場を離れて重要なイベントを開催してみよう。

メンター(37) [Mentor]: プロジェクトに新しいアイデアの導入を始めたいなら、そのアイデアを理解していて、チームを手助けできる人に実てもらおう。

視見(38) [Royal Audience]: 組織の管理者やメンバーが、著名人を招く(27)の招待者と一緒に過ごす時間をお届けしよう。

相談できる同志(39) [Shoulder to Cry On]: 新しい状況のときの愛以上に落ち込まないよう、新しいアイデアの導入で、同じように困っている人たちと話す機会を見つけよう。

成功の匂い(40) [Smell of Success]: あなたの取り組みがなんらかの目に見える結果を残した時、あなたと話したがる人がそそらと現れるだろう。その機会を活かして教育に力を入れよう。

弱いの持続(41) [Sustained Momentum]: 組織内で、新しいアイデアに対する興味を持続させる、という現在進行形の仕事に積極的にアプローチしよう。

トークン(42) [Token]: 新しいアイデアを人の記憶に生かし続けるために、伝えた話題に結付けられるトークンを渡そう。

抵抗と付き合うためのパターン

橋渡し役(43) [Bridge-Builder]: 新しいアイデアをすで受け入れた人を、まだ受け入れていない人と引き合わせよう。

懐疑派代表(44) [Champion Skeptic]: あなたのアイデアに懐疑的なオピニオンリーダーに、「公式な懐疑派」の役割を演じてもらうよう、協力をお願いしよう。彼らの懐疑的な姿勢を変えられないとしても、あなたの取り組みを改善するために、その意見を活かそう。

根回し(45) [Corridor Politics]: 重要な決議が行われる前に、意思決定者や影響力のある人々に非公式に接触し、それぞれの意思決定の先に何が待っているのか、彼らが十分に理解しているかどうかを確かめよう。

怖れは無用(46) [Fear Less]: 抵抗勢力を新しいアイデアの強みに変えよう。

お試し期間(47) [Trial Run]: 組織が新しいアイデアの導入に前向きではない場合、少しの間だけ試験的に使ってもらい、その結果を観察することを提案しよう。

有事の耳元でささやく(48) [Whisper in the General's Ear]: 重大の事ではマネージャーを説得しにくいことがある。その場合は他の人がいない場を設定して、懸念点を教えてもらう。

出典：『Fearless Change アジャイルに効く アイデアを組織に広めるための４８のパターン』 Mary Lynn Manns, Linda Rising 著 川口 恭伸 監訳 木村 卓央・吉江 貴 絵・高橋 一貴・中込 大祐・古屋 朝子・安井 力・山口 敏宇・米沢 仁・角 征典 訳

FEARLESS CHANGE
アジャイルに効く
アイデアを組織に広めるための48のパターン
Mary Lynn Manns, Linda Rising 著
瀬谷リミ子 中俣公郎 訳

□

Fearless bulldozer 1/4

構わず突進

- 返事なくてもまたメールする
- チャットもする
- 席にも行く
- 定例設定＆毎回ファシリテート
- 完成を待つだけじゃなく進捗を確認する
- 問い合わせが熱そうならヒアリングを設定
- 頼られたときは求めてる以上のサポートを

無邪気な無知

- 知らないことを隠さない
- たくさん質問して教えてもらう
- でも自分が正しいと思えば意見も言う

Kotaro Ogino

5月30日 ·

今作成中のDevOps研修のレビューで、 Ayana Yoshida さんが研修講師に「これはValue Stream Mappingじゃない」と突っ込んでたのが今日のハイライト。

もう僕いなくても、いい感じで研修作成は回ってく気がするww

相方絶対論

- 陰口に負けない
- 何か変えようとすると必ず抵抗勢力は現れると悟る
- 抵抗勢力と戦うよりも味方を増やすことに力を入れる
- 絶対的に信頼できる相方を作る
    - 自分にはまらない相手もはまることが多い
    - 自分への文句は相方にいくようになる
    - フォローしてくれるし後で教えてくれる

    → ひとりで頑張るよりも頑張れる！

Fearless bulldozer 4/4 - Core

一番大事にしているのは、初心に返って、

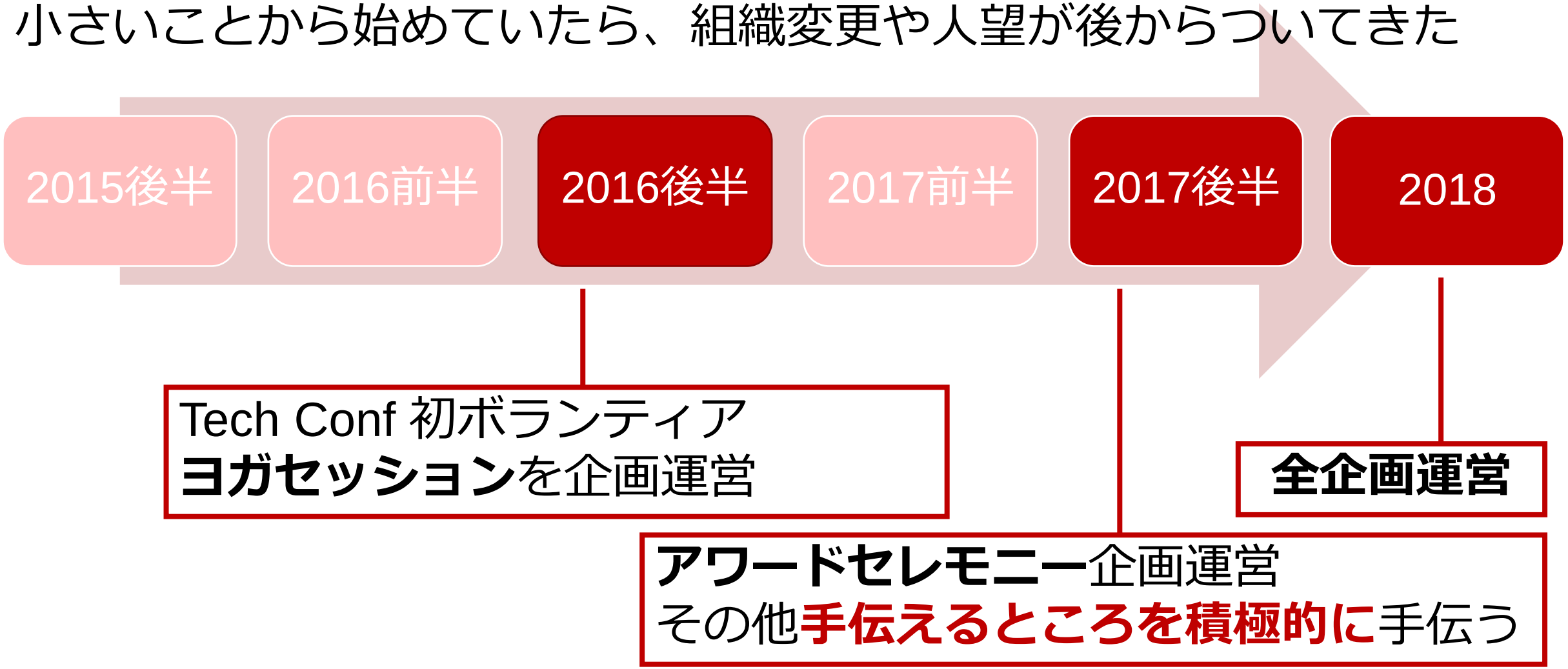
コラボ精神

- １人でやるよりも複数でやる方が大きいことができる
- たくさんのアイディアとたくさんのスキル
- 同じタイプの人も違うタイプの人も
- 興味を持ってくれたらとにかく誘ってみる

→ そして、みんなにプラスになるようにやってみる

Technology Conferenceも同じ

小さいことから始めていたら、組織変更や人望が後からついてきた



2018 Tech Conf x Fearless Bulldozer

いろいろやってみた！　たくさん巻き込んだ！

恐れ知らずのブルドーザー改革

1. 小さなことからやってみる
2. やってみてダメなら変える
3. 巻き込めそうな人をどんどん巻き込む
4. 組織変更や人望は後からついてくる

→ Fearless Change

+ Fearless Bulldozer

- 構わず突進
- 無邪気な無知
- 相方絶対論
- コラボ精神

まとめ③

・組織を変える啓蒙活動

③ 導入事例（文化面）

まとめ③

・組織を変える啓蒙活動

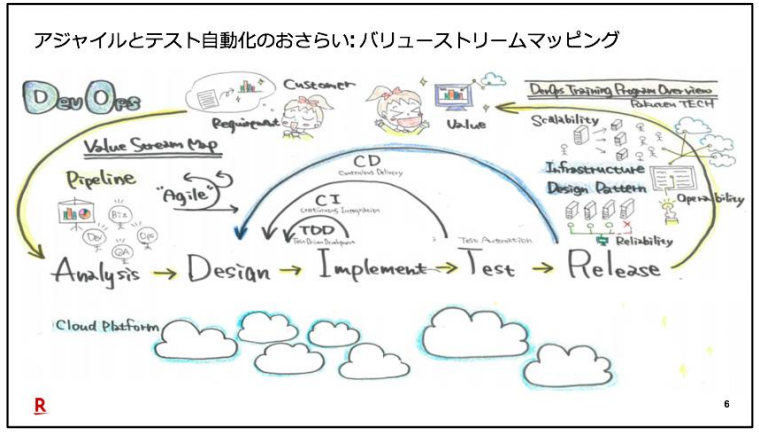
→構わず突進　→無邪気な無知

→相方絶対論　→コラボ精神

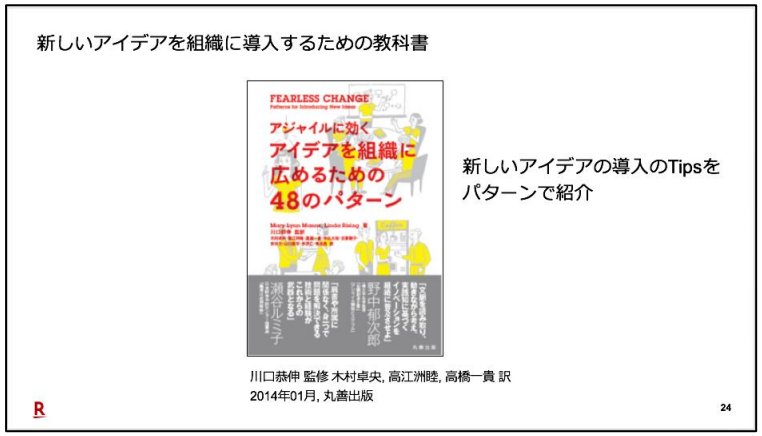
③ 導入事例（文化面）

本日お持ち帰りいただきたいこと

① おさらい

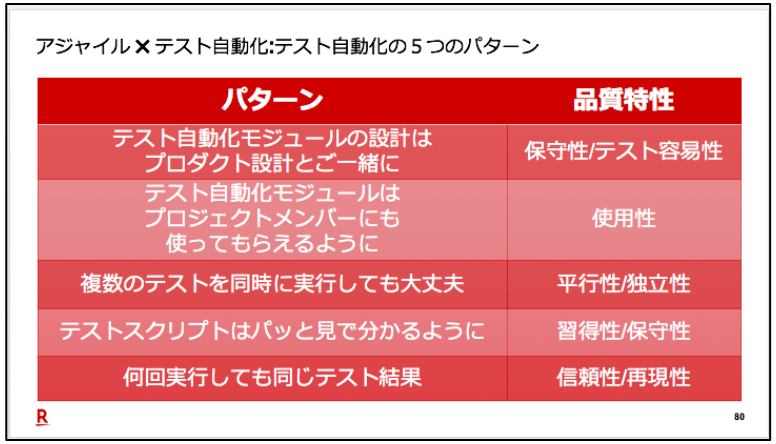


② 導入の勘所



③ 導入事例（文化面）

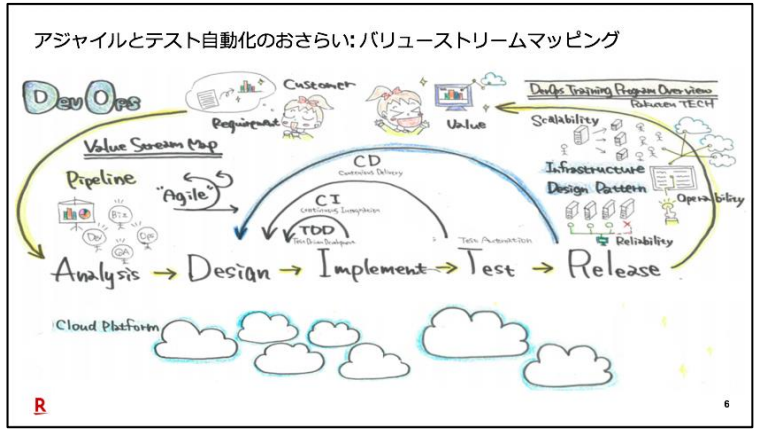
④ 導入事例（技術面）



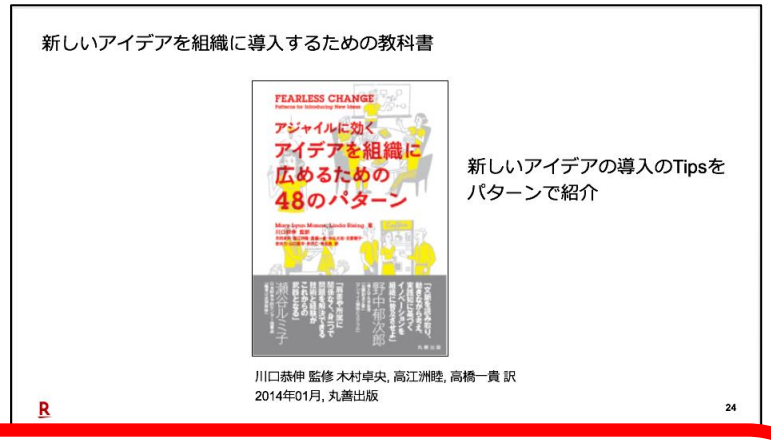
パターン	品質特性
テスト自動化モジュールの設計はプロダクト設計とご一緒に	保守性/テスト容易性
テスト自動化モジュールはプロジェクトメンバーにも使ってもらえるように	使用性
複数のテストを同時に実行しても大丈夫	平行性/独立性
テストスクリプトはパッと見で分かるように	習得性/保守性
何回実行しても同じテスト結果	信頼性/再現性

本日お持ち帰りいただきたいこと

① おさらい

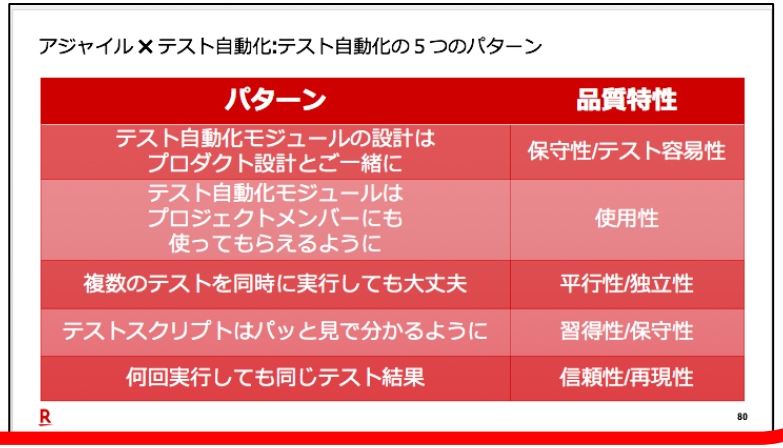


② 導入の勘所

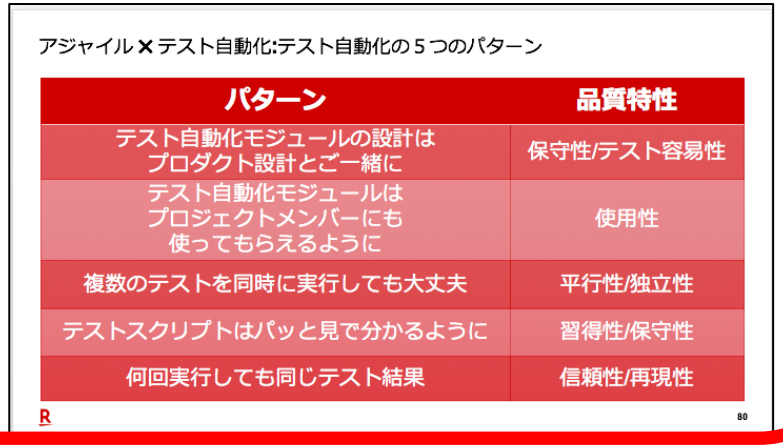


③ 導入事例（文化面）

④ 導入事例（技術面）



④ 導入事例（技術面）



アジャイル × テスト自動化:テスト自動化の５つのパターン

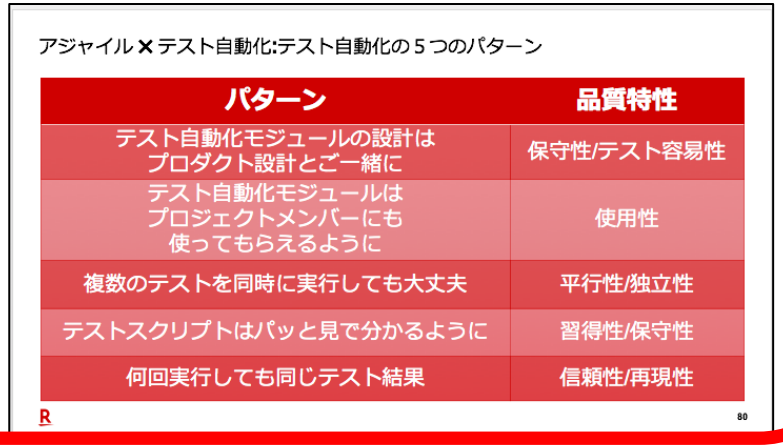
パターン	品質特性
テスト自動化モジュールの設計は プロダクト設計とご一緒に	保守性/テスト容易性
テスト自動化モジュールは プロジェクトメンバーにも 使ってもらえるように	使用性
複数のテストを同時に実行しても大丈夫	平行性/独立性
テストスクリプトはパッと見で分かるように	習得性/保守性
何回実行しても同じテスト結果	信頼性/再現性

R

80

・テスト自動化の導入事例
・導入事例の考察
・テスト自動化の５つのパターン

④ 導入事例（技術面）



アジャイル×テスト自動化:テスト自動化の５つのパターン

パターン	品質特性
テスト自動化モジュールの設計は プロダクト設計とご一緒に	保守性/テスト容易性
テスト自動化モジュールは プロジェクトメンバーにも 使ってもらえるように	使用性
複数のテストを同時に実行しても大丈夫	平行性/独立性
テストスクリプトはパッと見で分かるように	習得性/保守性
何回実行しても同じテスト結果	信頼性/再現性

R

80

・テスト自動化の導入事例
・導入事例の考察
・テスト自動化の５つのパターン

・テスト自動化の導入事例
・導入事例の考察
・テスト自動化の５つのパターン

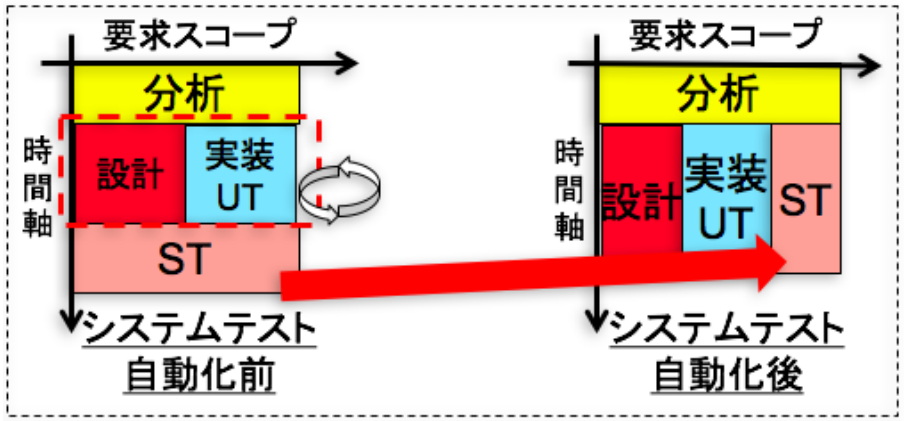
①検索基盤開発チームでのシステムテスト自動化

状況：スクラム/CIなどの
アジャイル手法は導入済み

問題：システムテストで見つかった
バグ修正の手戻り(5日)

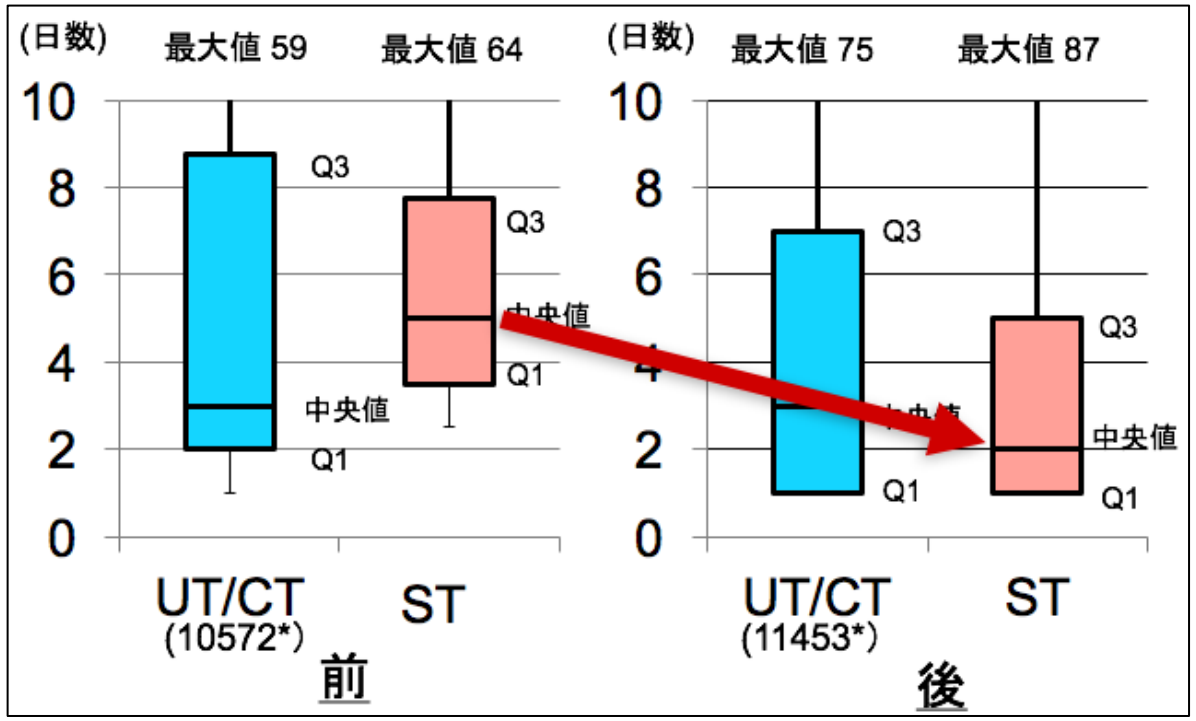
要因：システムテストを最後に実行

解決：継続的システムテスト



行動：①自動化専門チーム
②テスト自動化コストの削減
③開発とテストの協働

結果：バグ修正日数が５日から２日に



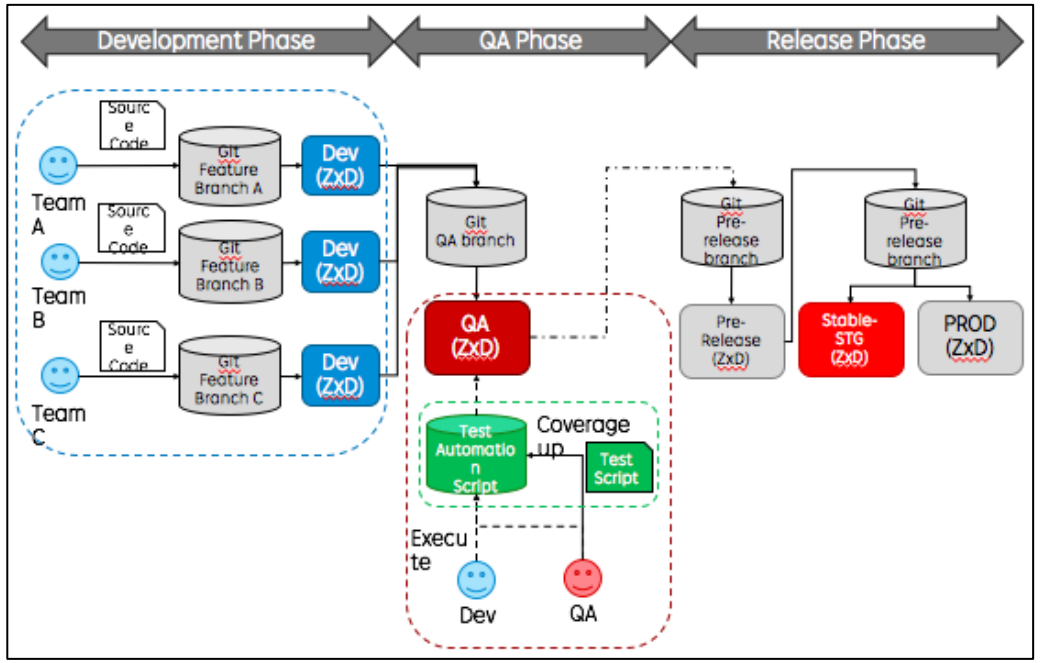
②ID基盤開発チームでのDevOps導入

状況：4拠点・100名以上の開発体制

問題：結合バグが大量に発生

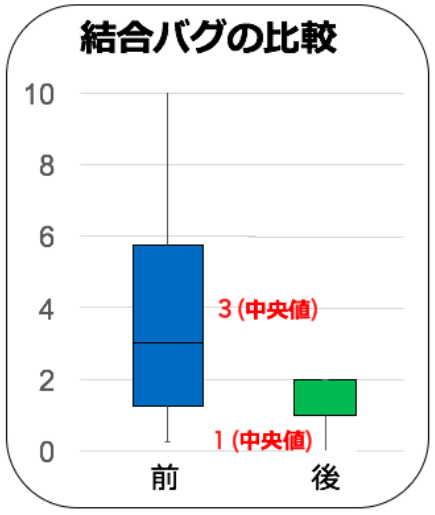
要因：個々のチームが独自に結合

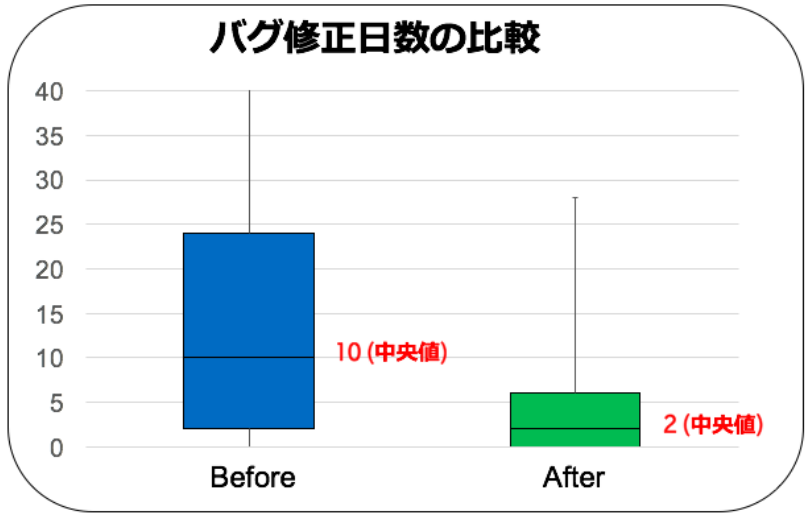
解決：パイプラインの自動化



行動：①セルフサービスQA環境
②カバレッジ改善
③スケーラブル開発環境

結果：結合バグの改善(3/週→1/週)
バグ修正日数の改善(10日→2日)

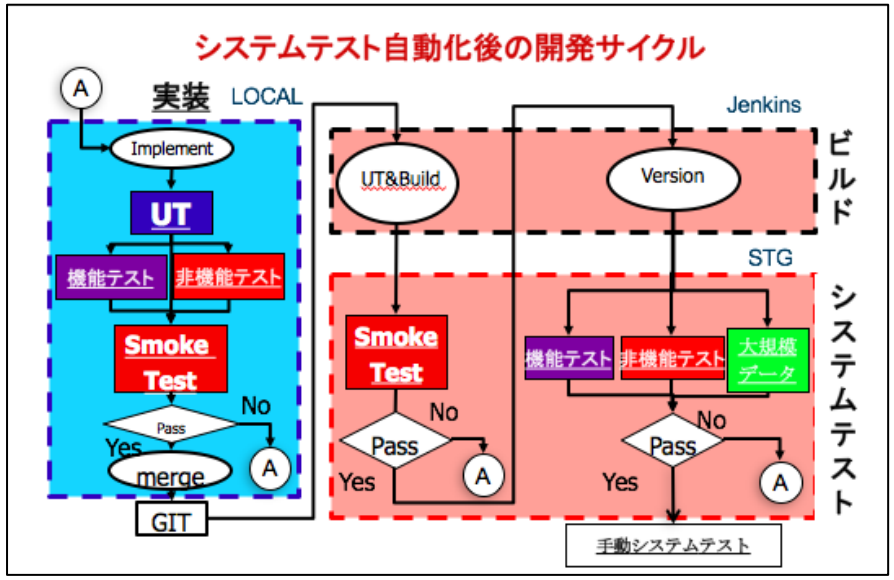


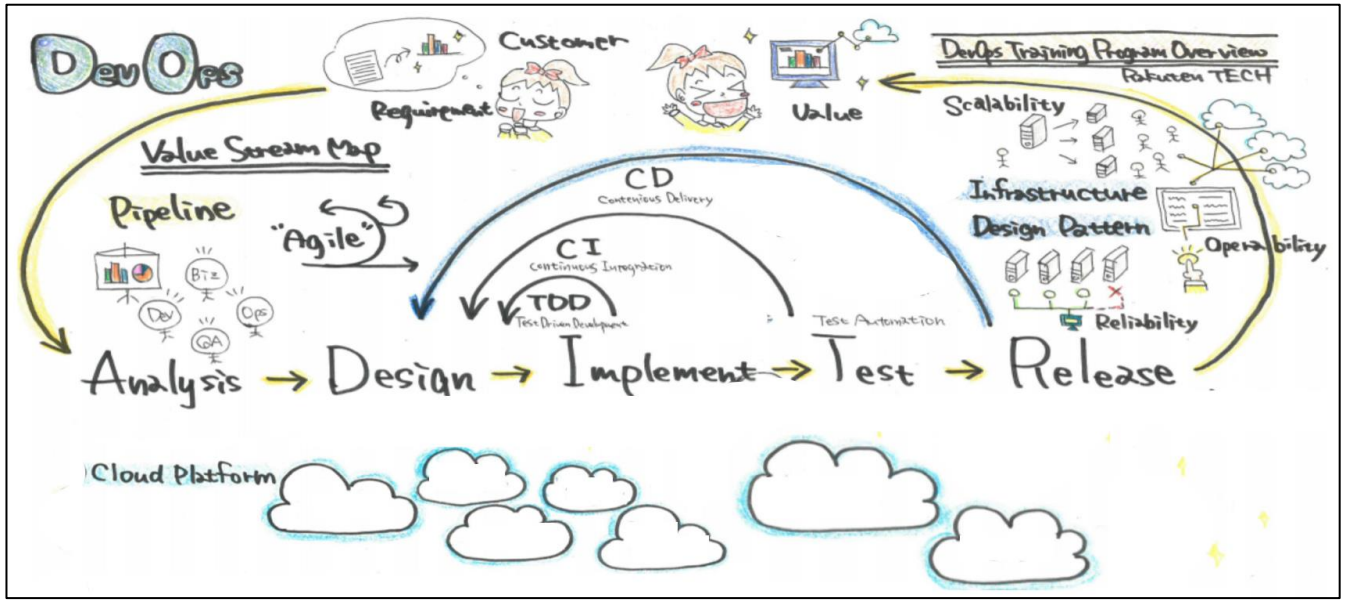


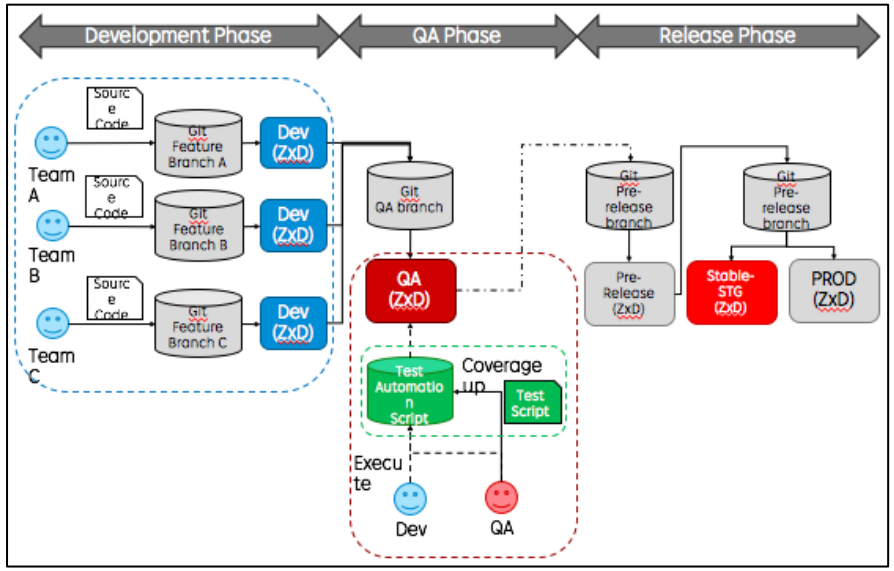
・テスト自動化の導入事例
・導入事例の考察
・テスト自動化の５つのパターン

・テスト自動化の導入事例
・導入事例の考察
・テスト自動化の５つのパターン

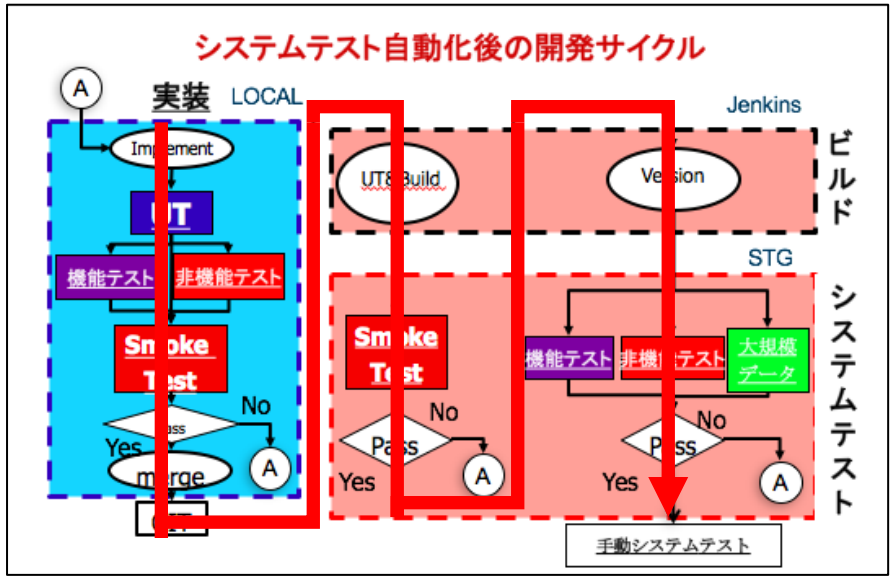
導入事例（技術面）:導入事例の考察 ~頻繁なフィードバック~

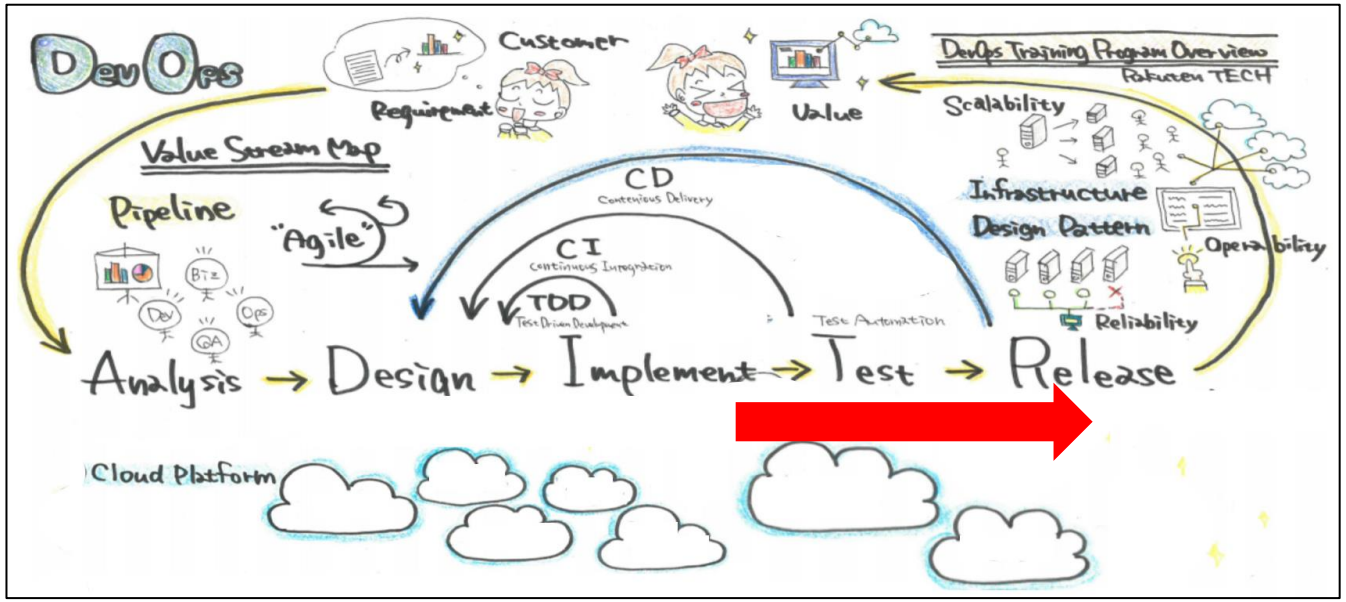


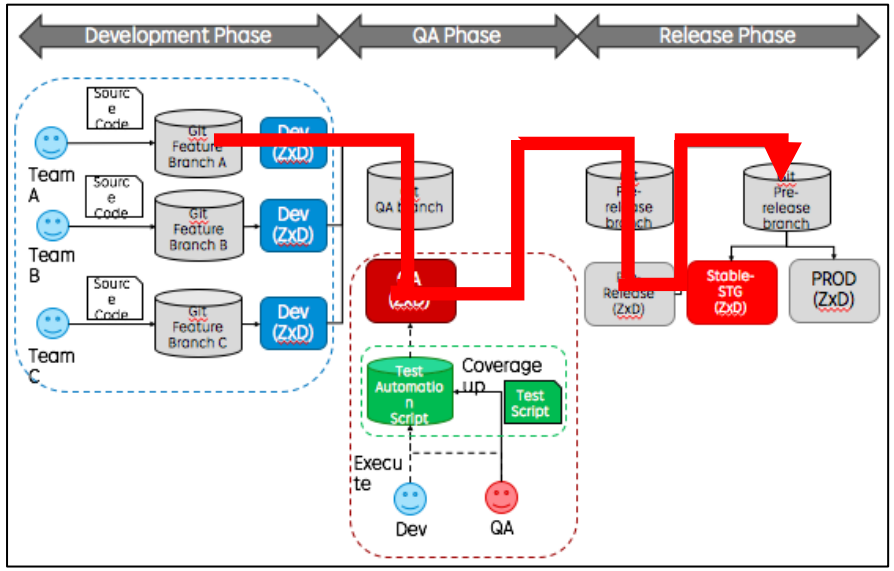




導入事例（技術面）:導入事例の考察 ~頻繁なフィードバック~

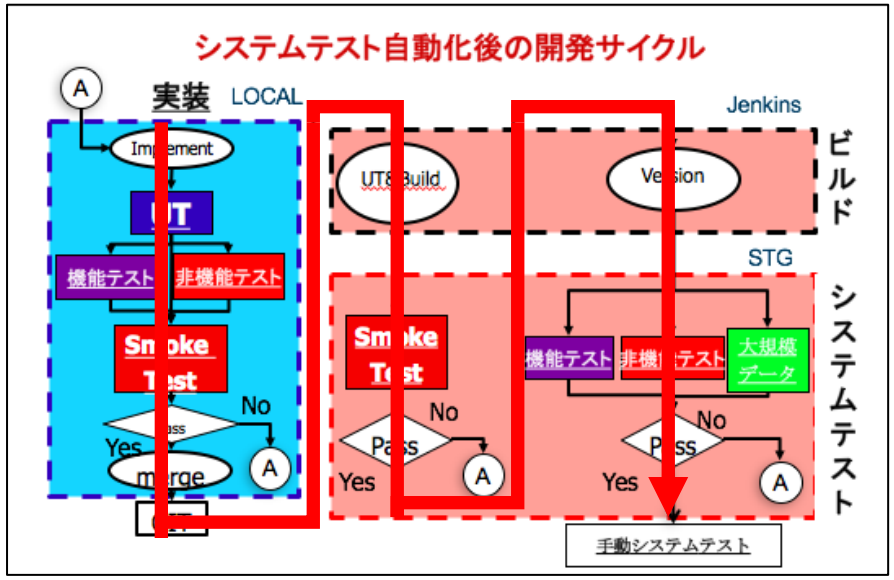


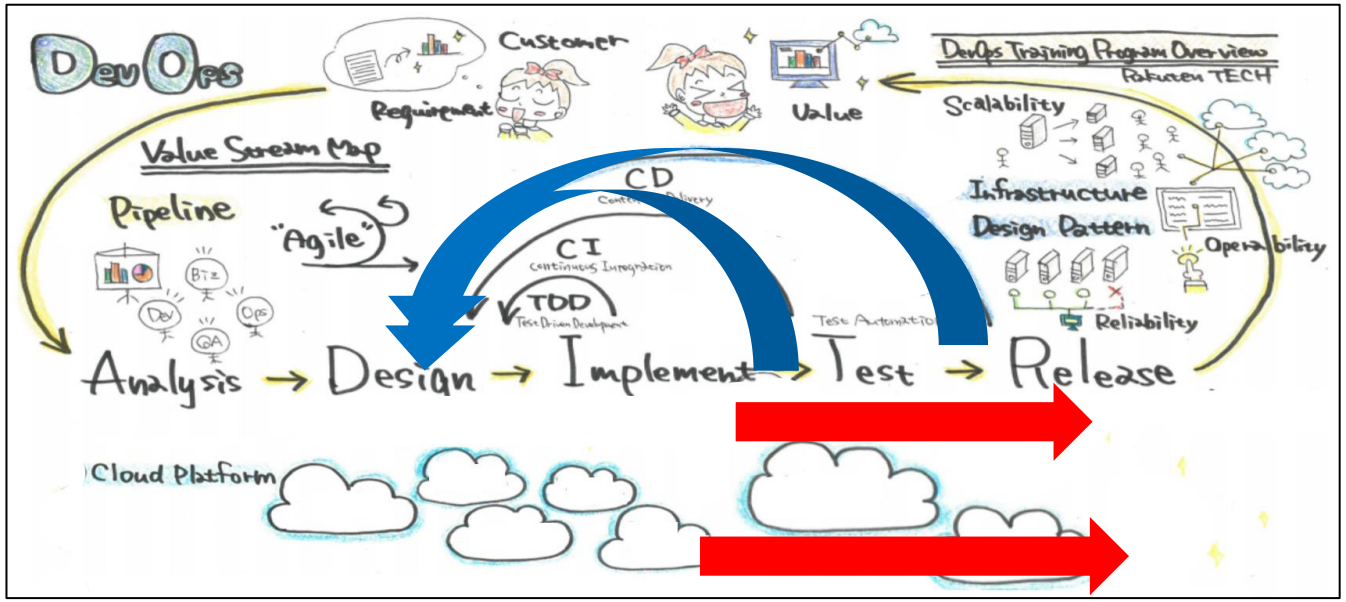


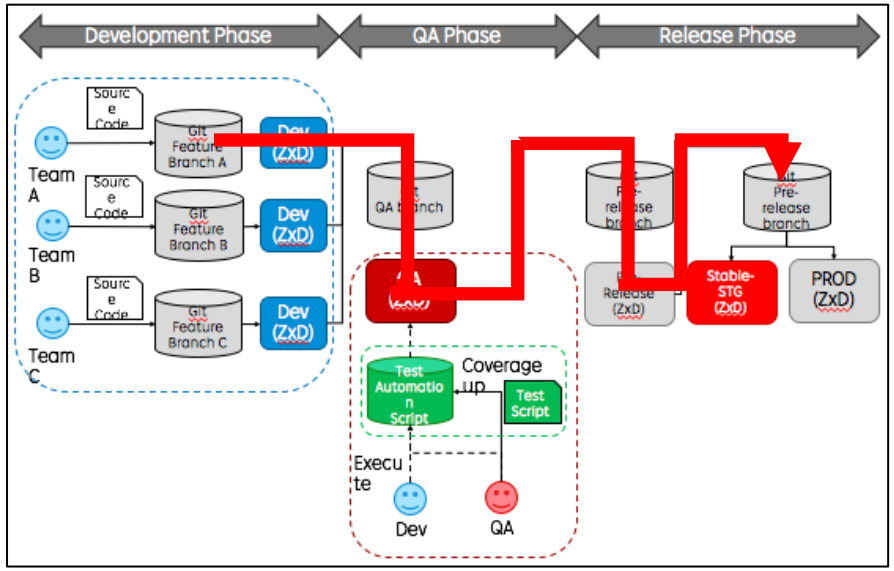


パイプライン（テスト）の自動化
=リードタイムの改善（最短1日）

導入事例（技術面）:導入事例の考察 ~頻繁なフィードバック~



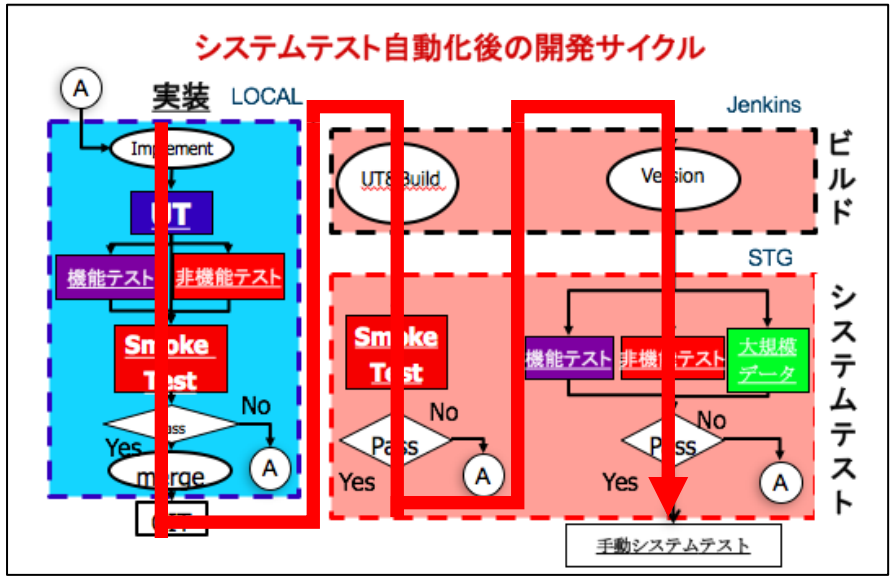


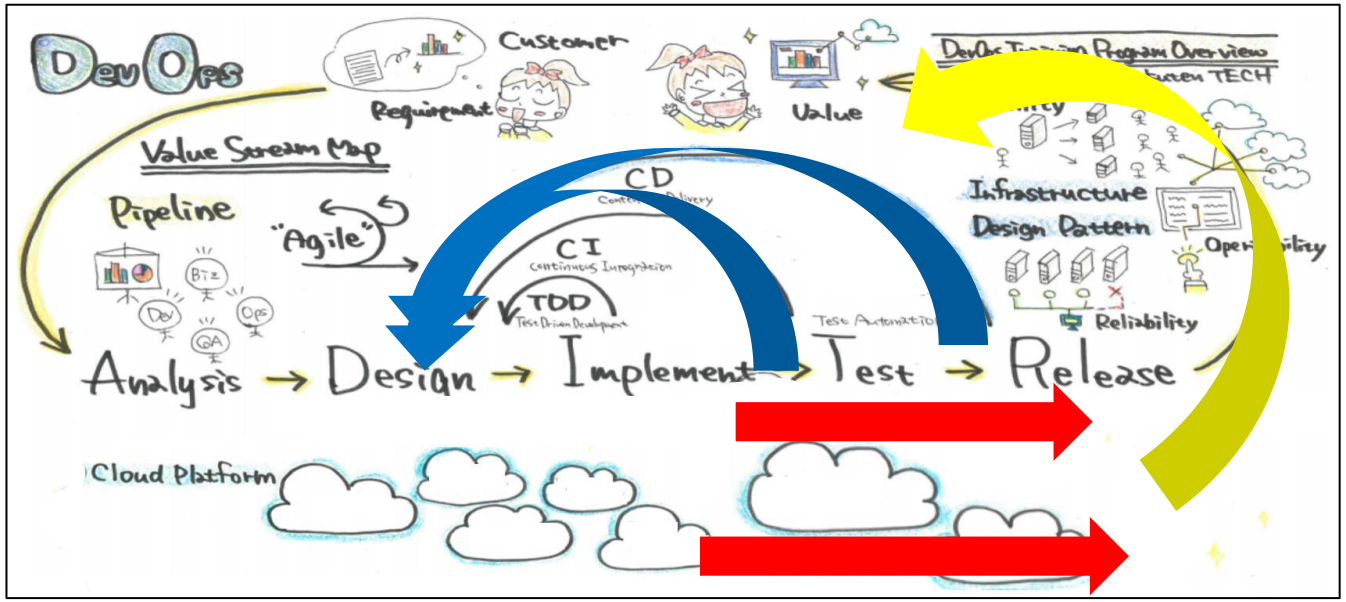


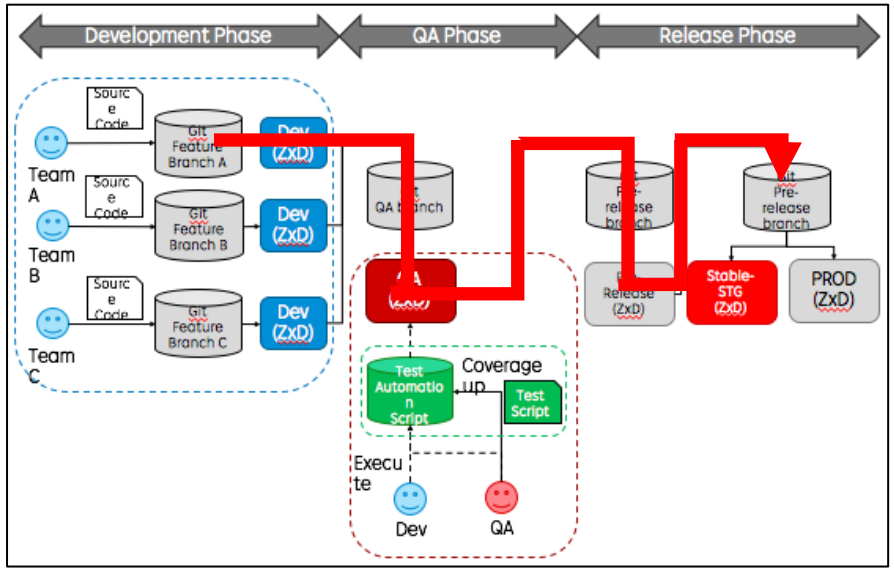
パイプライン（テスト）の自動化
=リードタイムの改善（最短1日）

頻繁なテストから設計への
フィードバック

導入事例（技術面）:導入事例の考察 ~頻繁なフィードバック~







パイプライン（テスト）の自動化
=リードタイムの改善（最短1日）

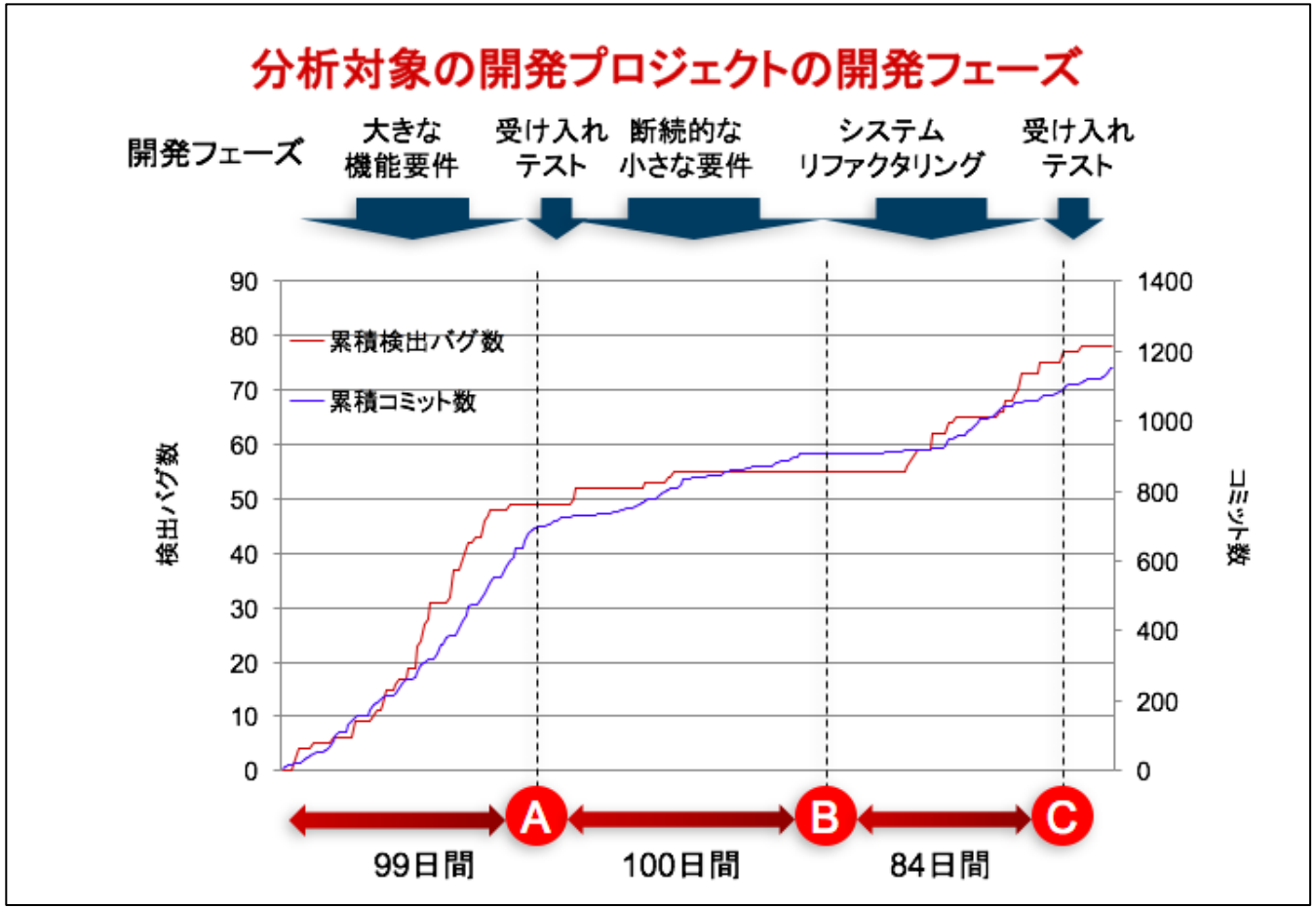
頻繁なテストから設計への
フィードバック

品質とリードタイムの改善

テスト自動化は
頻繁なフィードバックを作る

導入事例（技術面）:導入事例の考察 ~頻繁なフィードバック①~

毎日システムテストを実行しているプロジェクトのバグカーブ



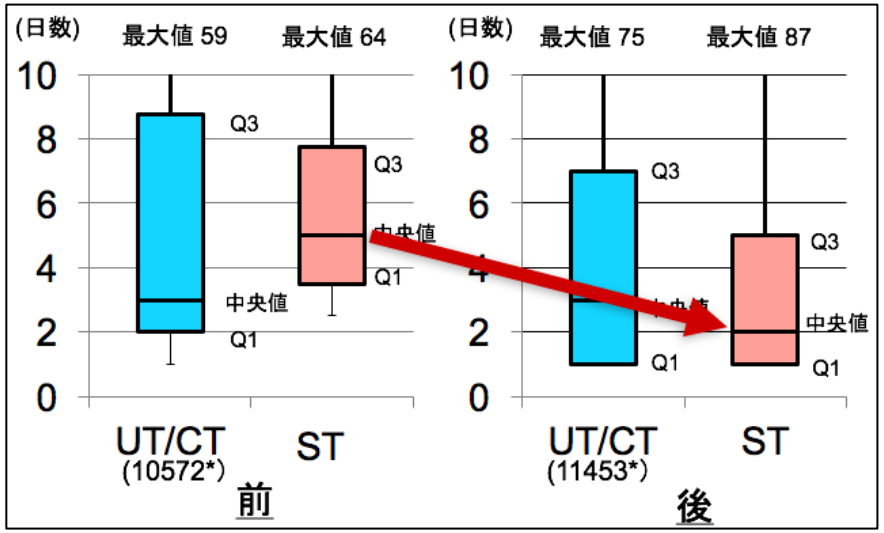
青線

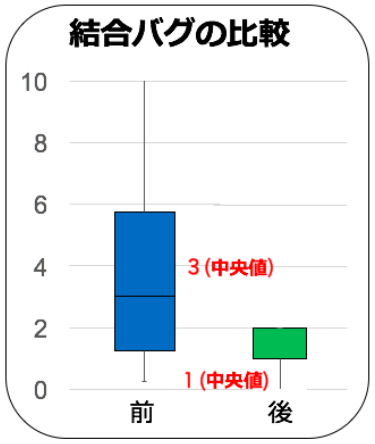
毎日ソースコード変更が可能に

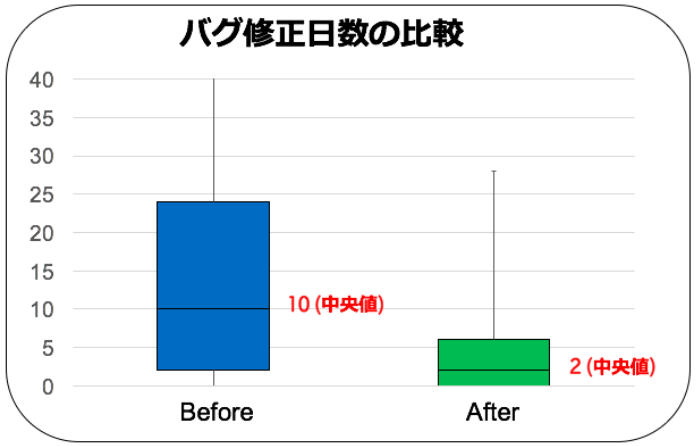
赤線

毎日バグが見つかる

導入事例（技術面）:導入事例の考察 ~頻繁なフィードバック②~







頻繁なフィードバックは バグ修正期間を大幅に短縮する

特に欠陥特定期間が大幅に短縮

理由：

- ソースコード変更からテスト実行までの期間が短い（多くの場合1日）
- 欠陥特定のための調査範囲を最小化することができる

・テスト自動化の導入事例
・導入事例の考察
・テスト自動化の５つのパターン

・テスト自動化の導入事例
・導入事例の考察
・テスト自動化の５つのパターン

アジャイル ✕ テスト自動化:テスト自動化の 5 つのパターン

アジャイル ✕ テスト自動化:テスト自動化の５つのパターン

パターン	品質特性
テスト自動化モジュールの設計は プロダクト設計とご一緒に	保守性/テスト容易性
テスト自動化モジュールは プロジェクトメンバーにも 使ってもらえるように	使用性
複数のテストを同時に実行しても大丈夫	平行性/独立性
テストスクリプトはパッと見で分かるように	習得性/保守性
何回実行しても同じテスト結果	信頼性/再現性

パターン①: テスト自動化モジュールの設計はプロダクト設計とご一緒に

状況：これから自動テストを導入する/外部から自動テストを導入した

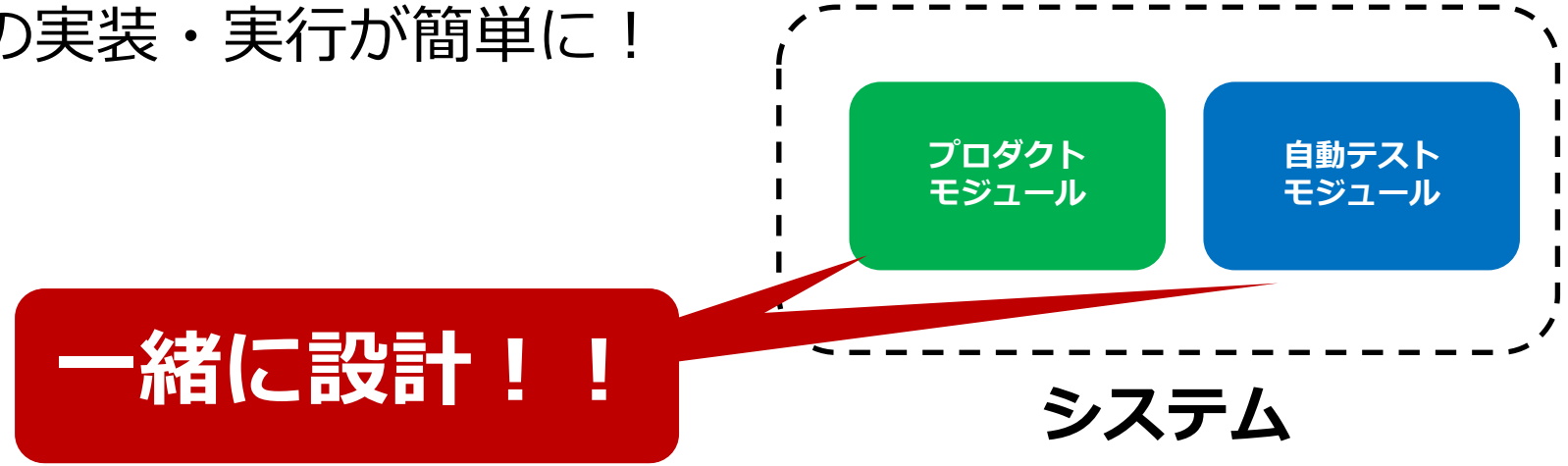
問題：プロダクトの設計に自動テストが振り回される

要因：独立したシステムとして自動テストが設計されている

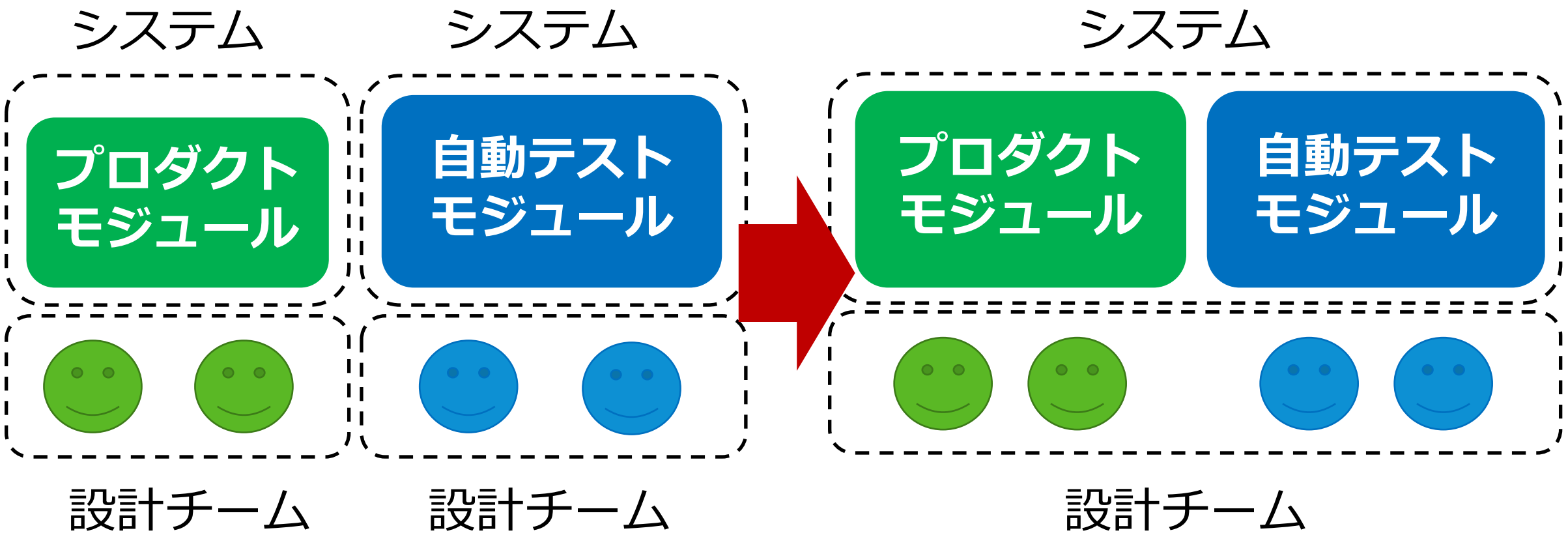
解決：プロダクトと自動テストを一つのシステムとみなす

行動：設計チームの統合とFeature Flagなどの実装

結果：テストの実装・実行が簡単に！



パターン① ：補足



プロダクトから自動テスト設計を改善：ABテストなど
テストからプロダクト設計を改善：Feature Flagなど

パターン②:テスト自動化モジュールは、プロジェクトメンバーにも使ってもらえるように

状況：テストチームに自動テストを導入した

問題：自動テスト導入の効果がテストのコスト削減に限定

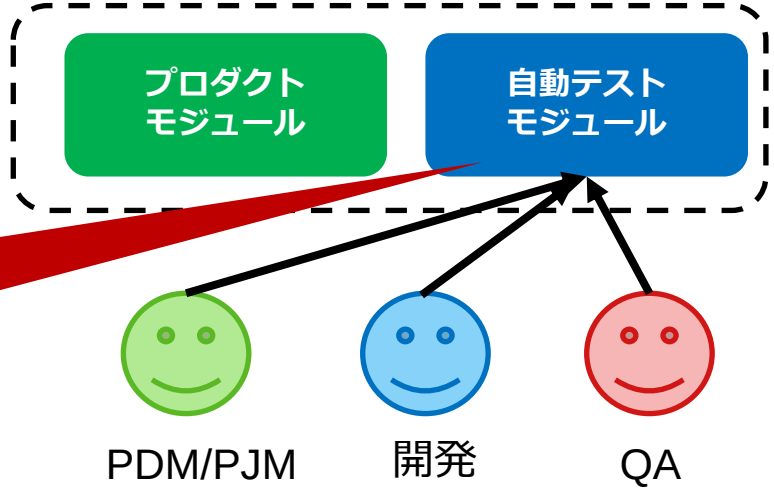
要因：プロジェクトメンバーから自動テストが実行不可能

解決：プロジェクトメンバーからも自動テストを実行可能に

行動：テスト実行とテストレポートUIの用意

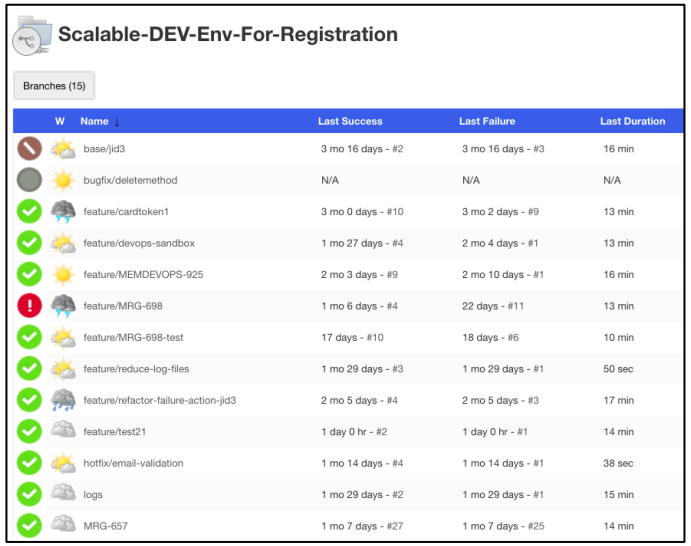
結果：プロジェクト全体で生産性が改善！

**プロジェクトメンバー
みんなが使えるように**

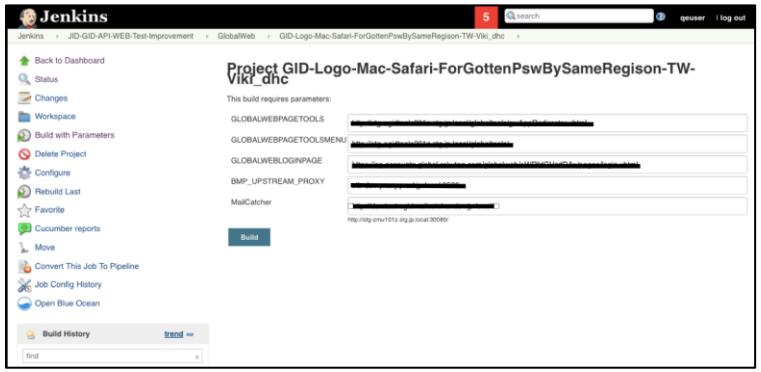


パターン②：補足

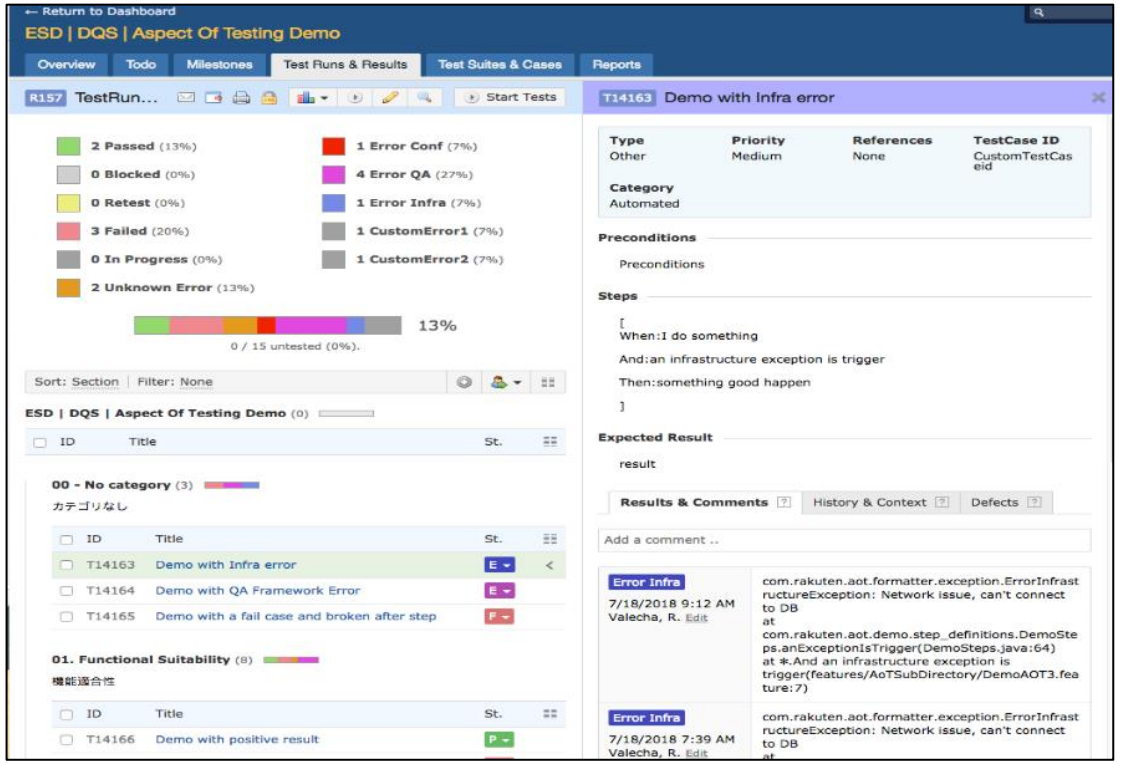
テスト環境構築のUI



自動テスト実行のUI



テストレポート



エンジニア以外でも
誰でもテストが実行可能に

パターン③:複数のテストを同時に実行しても大丈夫

状況：結構な量のテストを自動化した

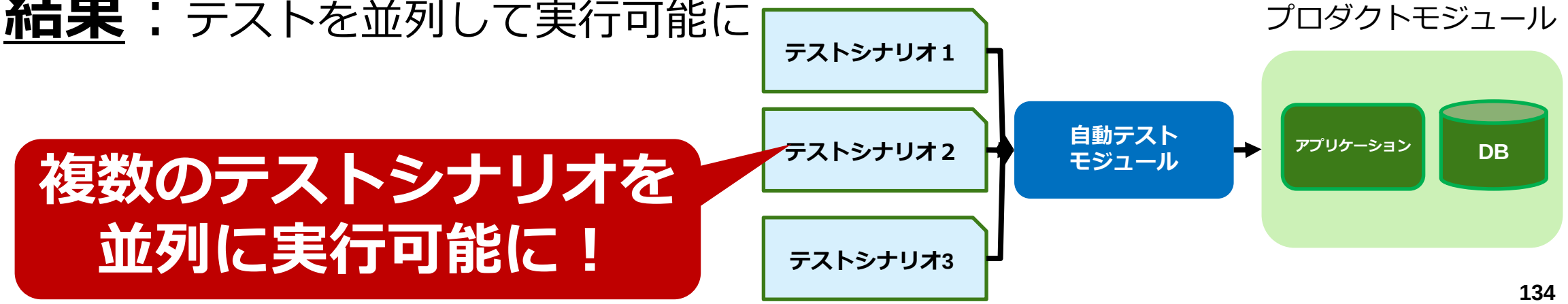
問題：複数のテストを同時に実行すると問題が起きる

要因：テストシナリオやテスト環境の独立性が担保されていない

解決：テストの独立性を担保する

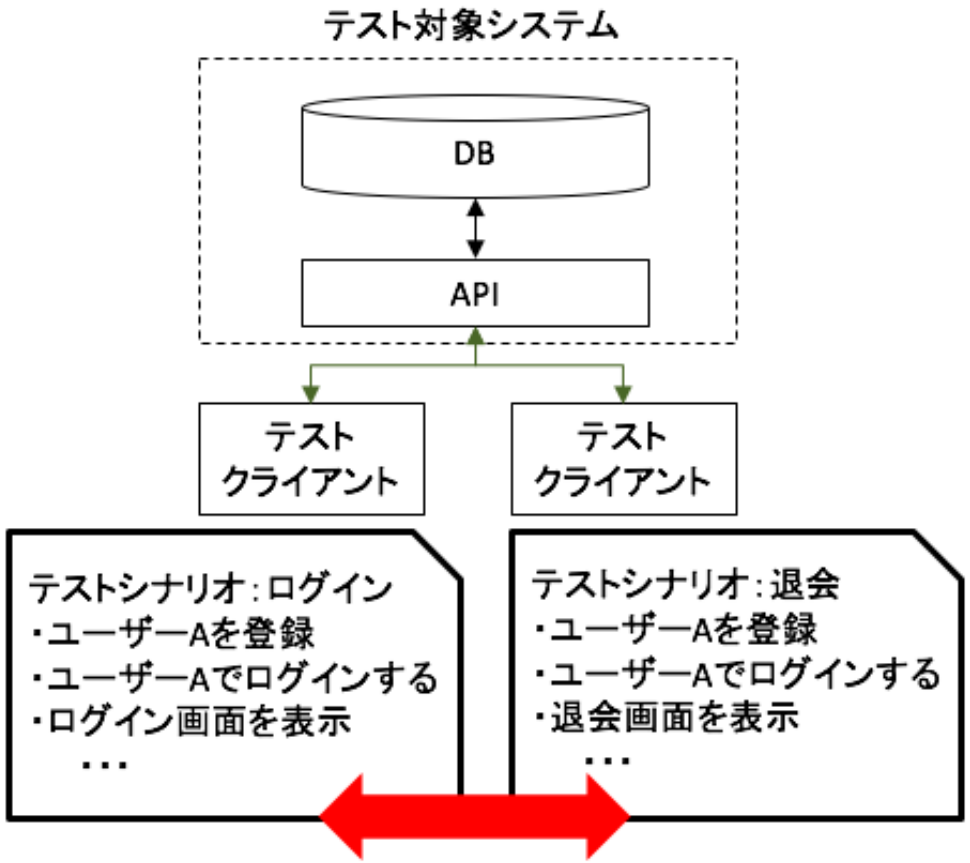
行動：テストシナリオとテストデータの独立性

結果：テストを並列して実行可能に



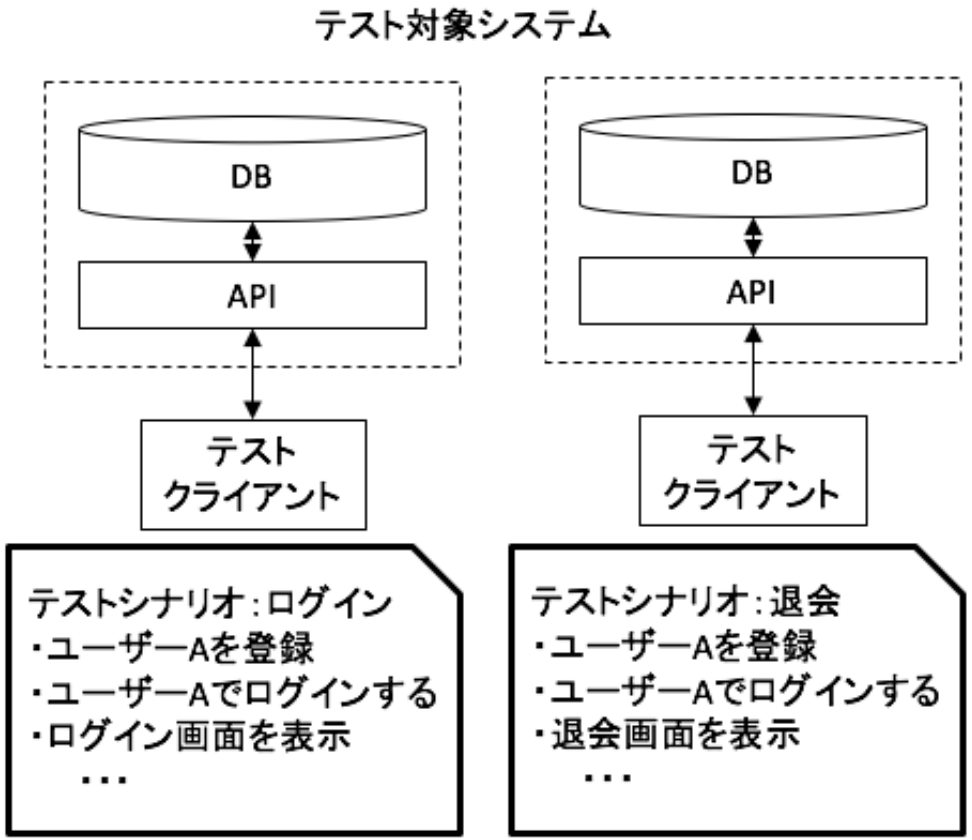
パターン③ ： 補足

テスト対象システムを共有する場合



同じユーザーをテストシナリオで共有すると
並列に実行できない

テスト対象システムを共有しない場合



同じユーザーをテストシナリオで共有しても
並列に実行できる

パターン④:テストスクリプトはパッと見で分かるように

状況：結構な量のテストが自動化されている

問題：テスト結果からバグを特定するのに時間がかかる

要因：テスト観点と条件がテストスクリプトからわかりにくい

解決：Behavior Driven Development(BDD)などの記述を採用する

行動：BDDの採用とテスト条件を明確に

結果：テスト結果の分析が簡単に！

キレイなテスト設計と自動テストは異なる？！

```
テストシナリオ：ログイン機能
    事前条件設定：
        このテストシナリオ専用のテスト対象環境を構築する
        アプリケーションとデータベースが正常に起動していなければテストをエラー終了する
        ユーザAのパスワードをxxxで登録する
        ユーザAがデータベースに登録されていなければテストをエラー終了する
    テスト条件：
        ユーザー名とパスワードを入力しログインボタンをクリック
    実際値と期待値の比較：
        表示されるページのメッセージを期待値と比較する
    事後操作：
        ユーザAがデータベースに登録されていなければテストをエラー終了する
        ユーザAを削除
        テスト対象環境を廃棄
    テスト条件：
        テストケース番号   ユーザ名   パスワード   期待値
        001                A          xxx          ログインできました
        002                A          yyy          ユーザ名かパスワードが間違っています
        003                B          xxx          ユーザ名かパスワードが間違っています
        ・・・
```

パタンーン④ :補足1(BDDで記述する)

シナリオ例：ログイン

テストシナリオ：ログイン機能
　　事前条件設定：ユーザーがシステムに登録されていること
　　テスト条件：
　　　　ユーザー名とパスワードを入力しログインボタンをクリック
　　実際値と期待値の比較：
　　　　表示されるページのメッセージを期待値と比較する

BDDでの記述の例

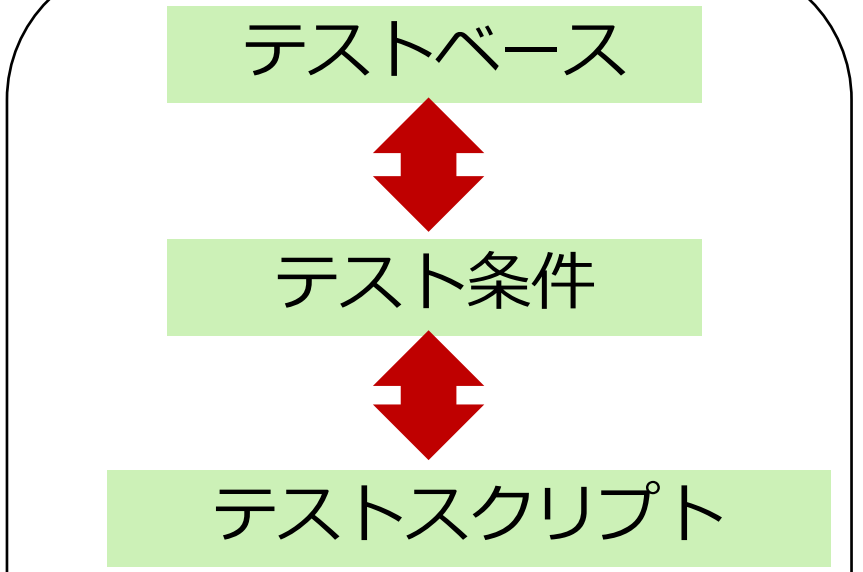
Scenario：login
　　Given the user is registered in the system
　　When the user inputs username and password then clicks login button
　　Then the user should see the specific message on the screen

パタンーン④ :補足2(テスト条件を明確に)

ログイン機能のテスト条件の例

ソースコード

```
If ( user is registered)
    if ( password is correct)
        change login status to logged in
        print “ログインできました”
    else
        print “ユーザー名かパスワードが間違っています”
else
    print “ユーザー名かパスワードが間違っています”
```



テストベースから分析した
テスト条件を
テストスクリプトに
そのまま記述する

テスト条件：

テストケース番号	ユーザ名	パスワード	期待値
001	A	xxx	ログインできました
002	A	yyy	ユーザ名かパスワードが間違っています
003	B	xxx	ユーザ名かパスワードが間違っています

パターン⑤:何回実行しても同じテスト結果

状況：結構な量のテストを自動化した

問題：実行するたび異なる結果になるテストが存在する (Flaky Test)

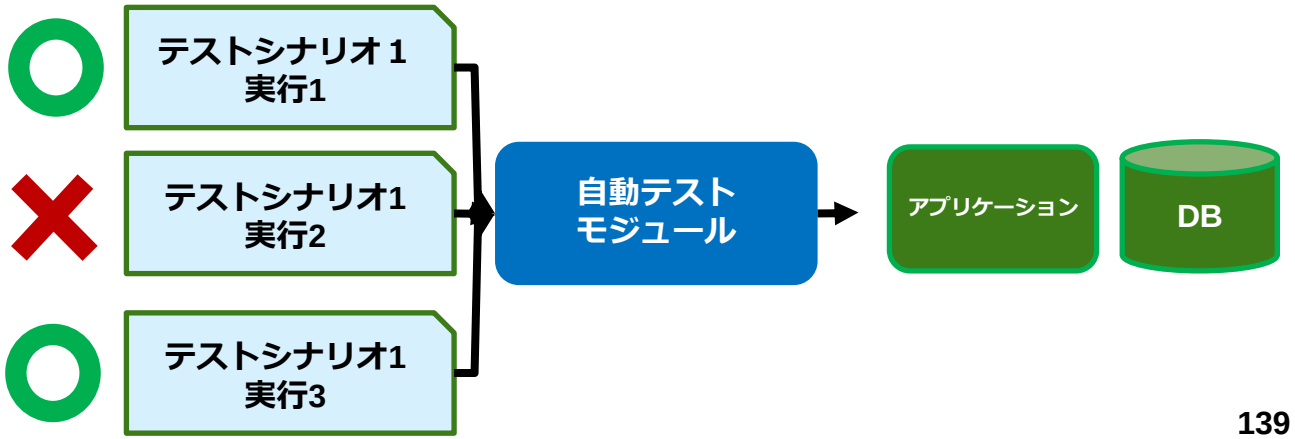
要因：テストの実行のたびに、テスト環境に変更が加えられている

解決：テスト環境をクリーンに保つ

行動：前処理・後処理とエラー処理の自動化

結果：信頼出来るテスト結果！

実行のたびに結果が異なると調査が大変



パターン⑤ ：補足

テストシナリオ：ログイン機能

事前条件設定：

このテストシナリオ専用のテスト対象環境を構築する

テスト対象環境が正常に起動していなければテストをエラー終了する

ユーザAのパスワードをxxxで登録する

ユーザAがデータベースにテストをエラー終了する

テスト条件：

ユーザー名とパスワードを入力しログインボタンをクリ

実際値と期待値の比較：

表示されるページのメッセージを期待値と比較する

事後操作：

ユーザAがデータベースに登録されていなければテストをエラー終了する

ユーザAを削除

テスト対象環境を廃棄

前処理

エラー処理

後処理

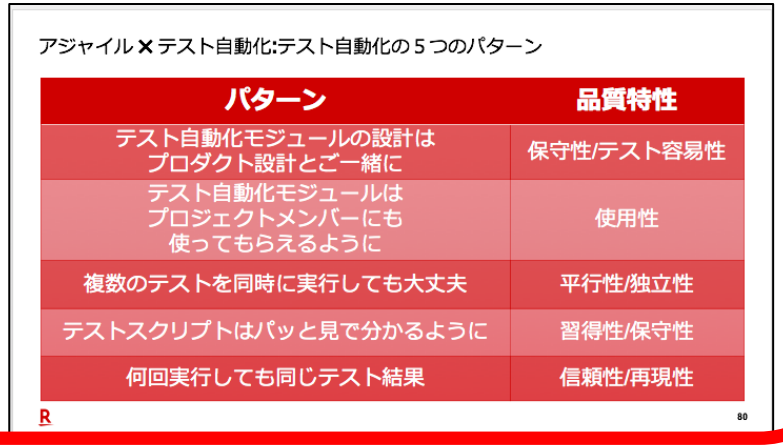
アジャイル ✕ テスト自動化:テスト自動化の５つのパターン

パターン	品質特性
テスト自動化モジュールの設計は プロダクト設計とご一緒に	保守性/テスト容易性
テスト自動化モジュールは プロジェクトメンバーにも 使ってもらえるように	使用性
複数のテストを同時に実行しても大丈夫	平行性/独立性
テストスクリプトはパッと見で分かるように	習得性/保守性
何回実行しても同じテスト結果	信頼性/再現性

まとめ④

- テスト自動化の導入事例
- 導入事例の考察
- テスト自動化の５つのパターン

④ 導入事例（技術面）



アジャイル×テスト自動化:テスト自動化の５つのパターン

パターン	品質特性
テスト自動化モジュールの設計は プロダクト設計とご一緒に	保守性/テスト容易性
テスト自動化モジュールは プロジェクトメンバーにも 使ってもらえるように	使用性
複数のテストを同時に実行しても大丈夫	平行性/独立性
テストスクリプトはパッと見で分かるように	習得性/保守性
何回実行しても同じテスト結果	信頼性/再現性

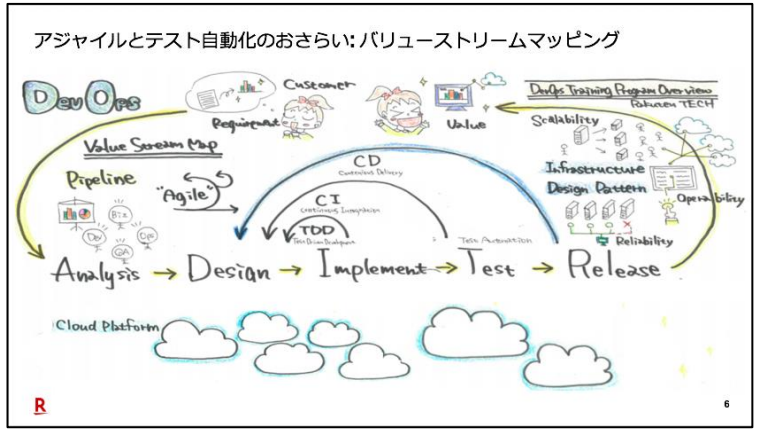
R

80

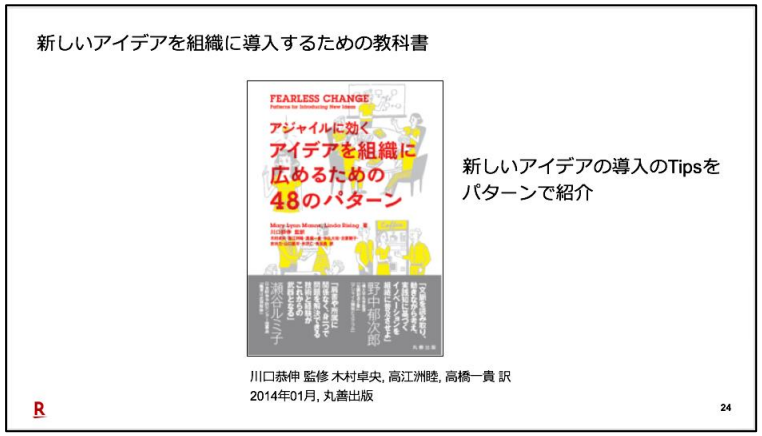
総まとめ

総まとめ

① おさらい

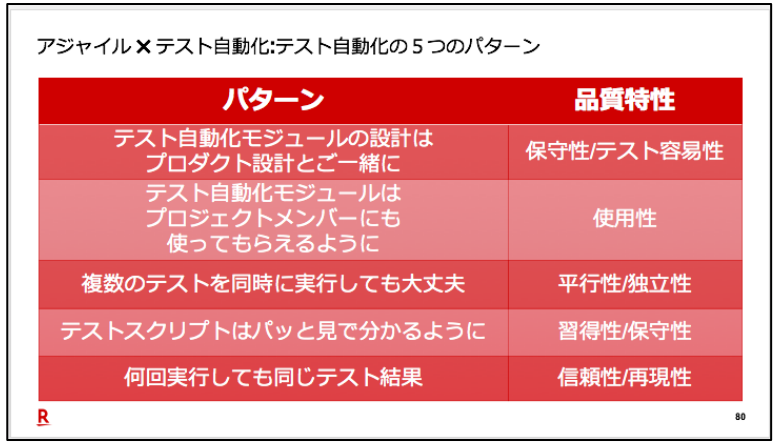


② 導入の勘所



③ 導入事例（文化面）

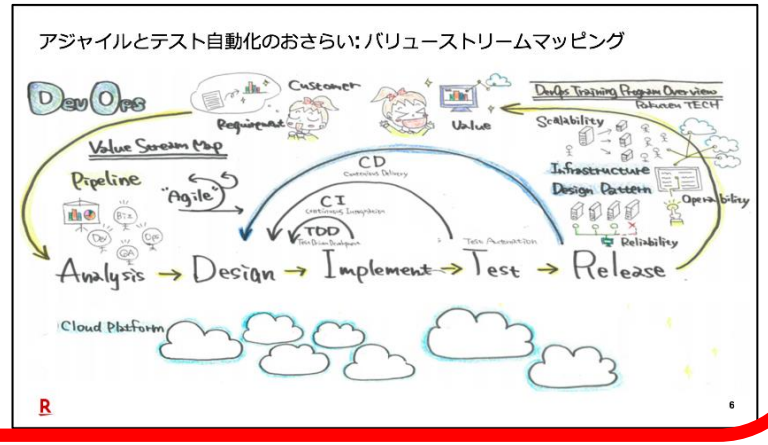
④ 導入事例（技術面）



パターン	品質特性
テスト自動化モジュールの設計はプロダクト設計とご一緒に	保守性/テスト容易性
テスト自動化モジュールはプロジェクトメンバーにも使ってもらえるように	使用性
複数のテストを同時に実行しても大丈夫	平行性/独立性
テストスクリプトはパッと見で分かるように	習得性/保守性
何回実行しても同じテスト結果	信頼性/再現性

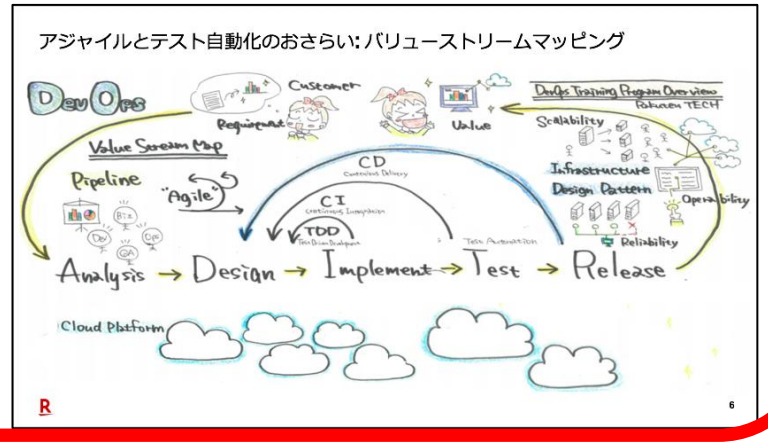
総まとめ

① おさらい


アジャイルとテスト自動化のおさらい：バリューストリームマッピング
DevOps
Customer
Requirement
Value
DevOps Training Program Overview
Fukuwen TECH
Value Stream Map
Pipeline
"Agile"
CD
CI
TDD
Scalability
Infrastructure
Design Pattern
Operability
Reliability
Analysis → Design → Implement → Test → Release
Cloud Platform
R
6

総まとめ

① おさらい



- アジャイルとは何か？
  →少しずつ、素早く
- アジャイルでのテスト自動化の役割
  →チームにフィードバック

総まとめ

② 導入の勘所

新しいアイデアを組織に導入するための教科書

新しいアイデアの導入のTipsを
パターンで紹介

川口恭伸 監修 木村卓央, 高江洲睦, 高橋一貴 訳
2014年01月, 丸善出版

R

24

総まとめ

② 導入の勘所

新しいアイデアを組織に導入するための教科書

新しいアイデアの導入のTipsを
パターンで紹介

川口恭伸 監修 木村卓央, 高江洲睦, 高橋一貴 訳
2014年01月, 丸善出版

R

24

- **新しいアイデア導入のTips**
  - **→仲間を増やす、説得するなど**
  - **→文化面、技術面の両面から**

総まとめ

③ 導入事例（文化面）

総まとめ

・組織を変える啓蒙活動

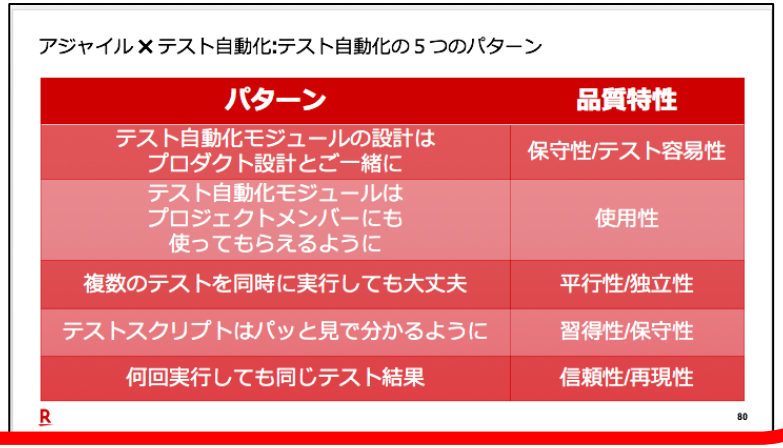
→構わず突進　→無邪気な無知

→相方絶対論　→コラボ精神

③ 導入事例（文化面）

総まとめ

④ 導入事例（技術面）



アジャイル × テスト自動化:テスト自動化の５つのパターン

パターン	品質特性
テスト自動化モジュールの設計は プロダクト設計とご一緒に	保守性/テスト容易性
テスト自動化モジュールは プロジェクトメンバーにも 使ってもらえるように	使用性
複数のテストを同時に実行しても大丈夫	平行性/独立性
テストスクリプトはパッと見で分かるように	習得性/保守性
何回実行しても同じテスト結果	信頼性/再現性

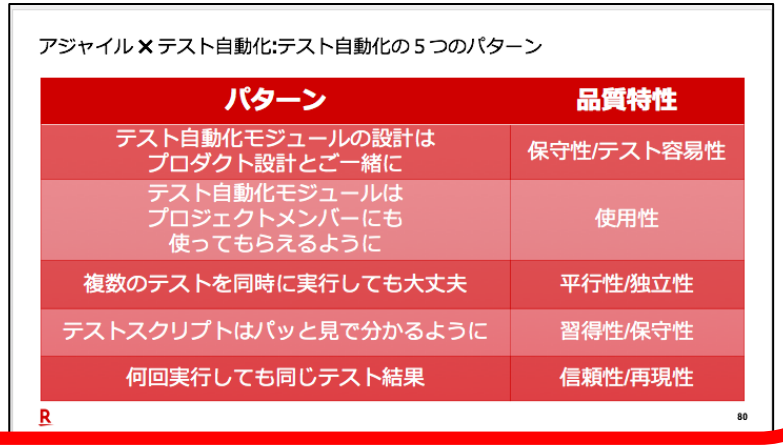
R

80

総まとめ

- **テスト自動化の導入事例**
- **導入事例の考察**
- **テスト自動化の５つのパターン**

④ 導入事例（技術面）



アジャイル×テスト自動化:テスト自動化の５つのパターン

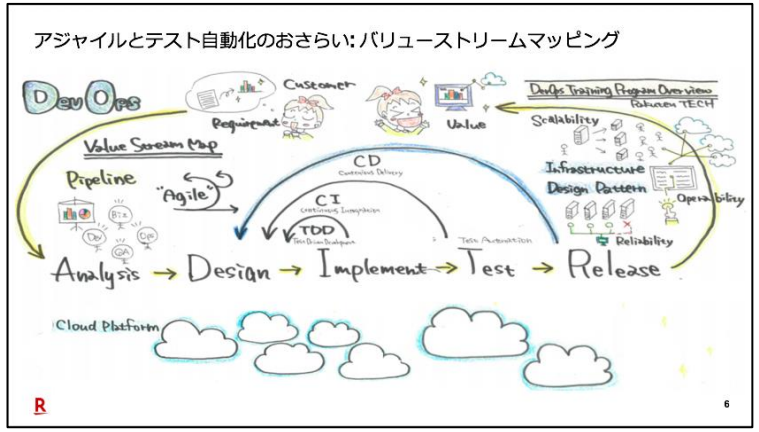
パターン	品質特性
テスト自動化モジュールの設計は プロダクト設計とご一緒に	保守性/テスト容易性
テスト自動化モジュールは プロジェクトメンバーにも 使ってもらえるように	使用性
複数のテストを同時に実行しても大丈夫	平行性/独立性
テストスクリプトはパッと見で分かるように	習得性/保守性
何回実行しても同じテスト結果	信頼性/再現性

R

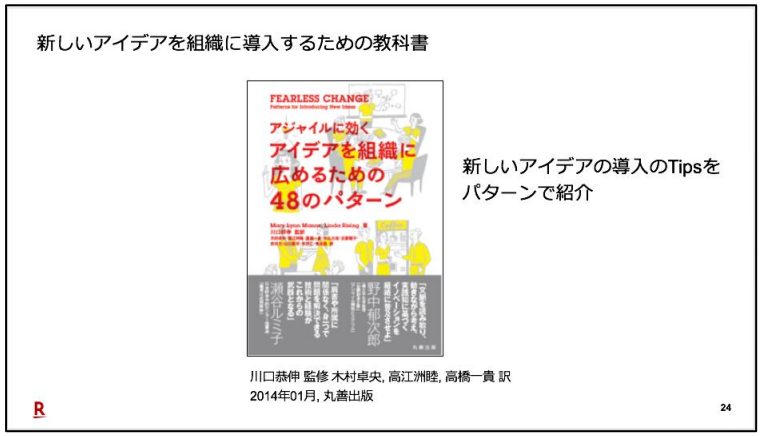
80

総まとめ

① おさらい

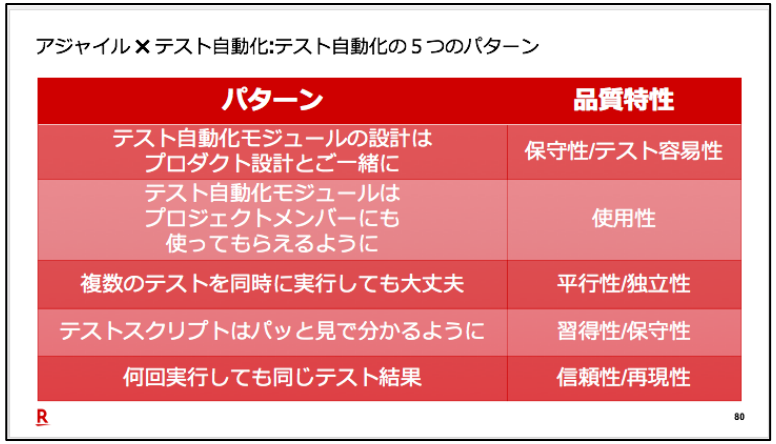


② 導入の勘所



③ 導入事例（文化面）

④ 導入事例（技術面）



パターン	品質特性
テスト自動化モジュールの設計はプロダクト設計とご一緒に	保守性/テスト容易性
テスト自動化モジュールはプロジェクトメンバーにも使ってもらえるように	使用性
複数のテストを同時に実行しても大丈夫	平行性/独立性
テストスクリプトはパッと見で分かるように	習得性/保守性
何回実行しても同じテスト結果	信頼性/再現性


Rakuten

付録

なぜアジャイルとテスト自動化が”今“必要なのか？

JaSST’ Tokyo 18

基調講演

“JaSST’18Tokyo_ご報告書(ご協賛)”より

クロージングパネル

クロージングパネル

セッション A8 ▶概要

「アジャイル・自動化時代のテストの現場のリアル」

モデレータ：
荻野 恒太郎 （楽天）
Download 講演資料 (PDF：913KB)

パネリスト：
John Micco （Google）
Download 講演資料 (PDF：148KB)

天野 祐介 （サイボウズ）
Download 講演資料 (PDF：4,389KB)

松尾 和昭 （クックパッド）
Download 講演資料 (PDF：1,118KB)

山口 鉄平 （ヤフー）
Download 講演資料 (PDF：164KB)

JaSST’18 Tokyo レポート

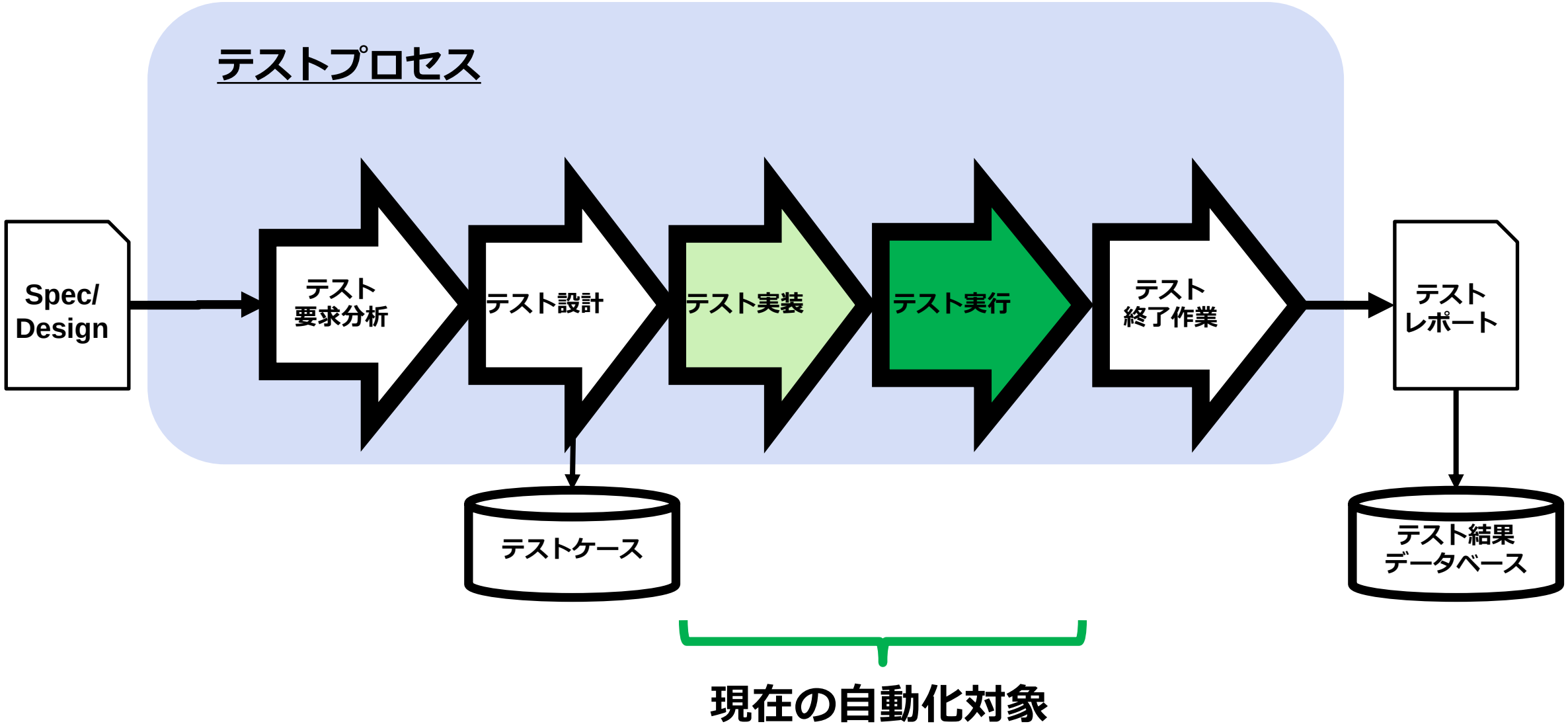
http://www.jasst.jp/symposium/jasst18tokyo/report.html

2018/11/13 アクセス

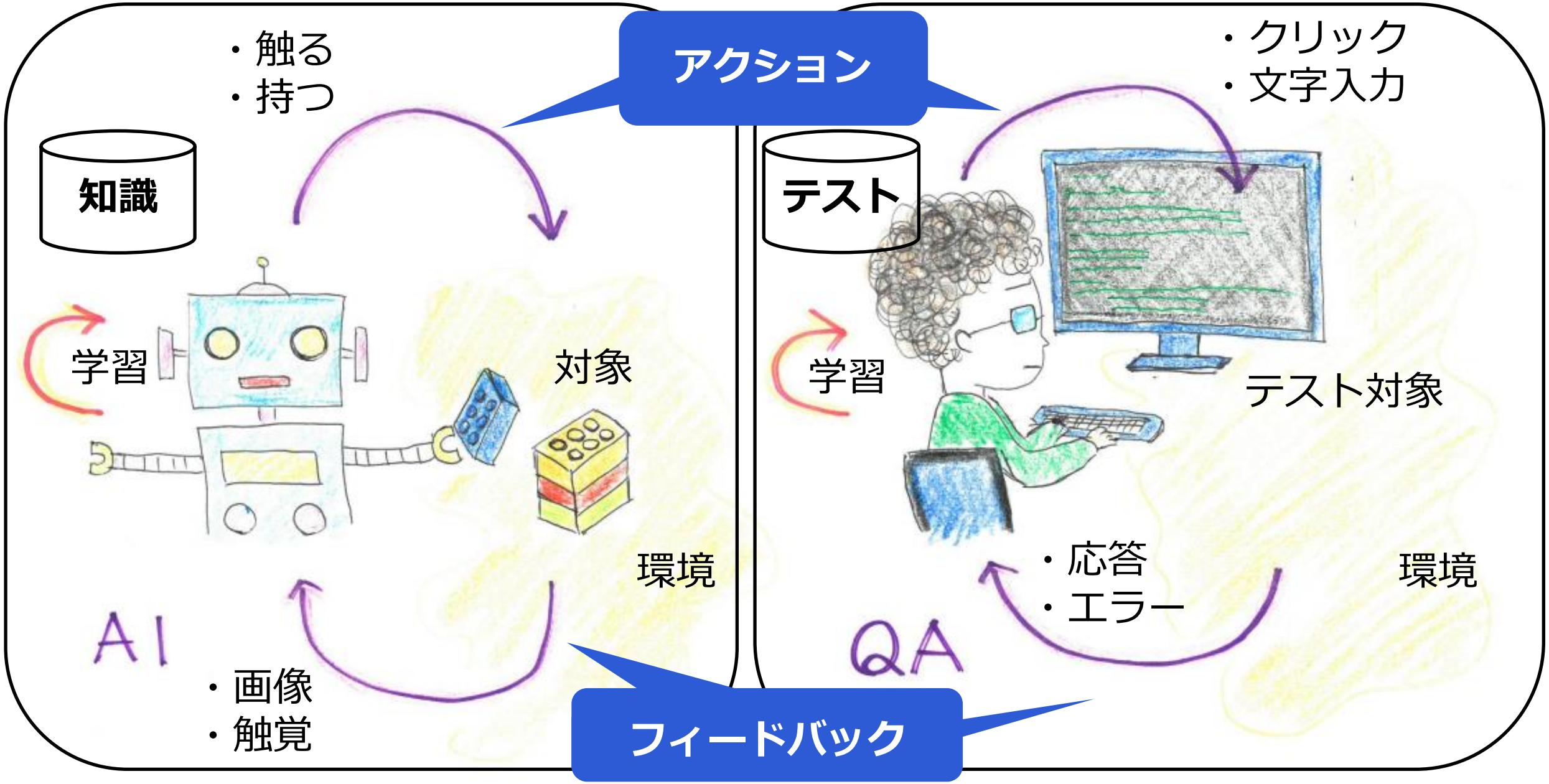
JaSST’ Tokyo 18でのギャップ

- **GoogleのJohn Miccoの基調講演の内容**
  - -(**Agileの導入やテスト実行自動化は100%完了**している前提)
  - -数百万件～数億件のテストが自動化されている世界で、
    テスト結果の確認を機械にやらせたい（自動化したい）
- **会場質問から感じたギャップ**
  - -「本当に100%自動化されているのか？」と繰り返される質問に
    自動化しない言い訳を探しているように感じた
- **パネルディスカッションで伝えたかった事**
  - -Googleみたいな企業では**テスト実行の自動化は当たり前**
  - -**テスト実行以外の領域の自動化**に乗り出している

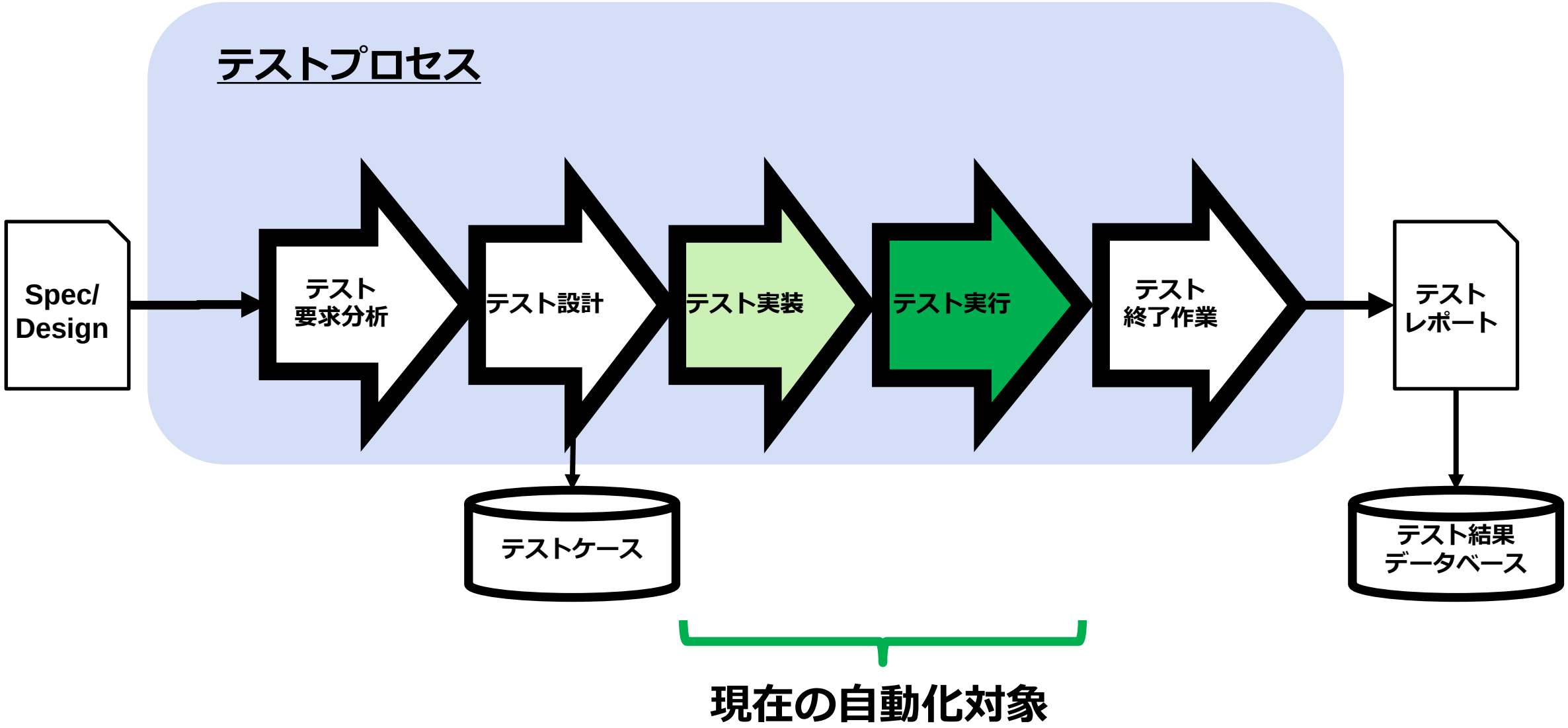
テスト自動化の対象領域



AIとQAのモデル: 仮定=“AI は素早くテスト設計やテスト結果を学習可能”



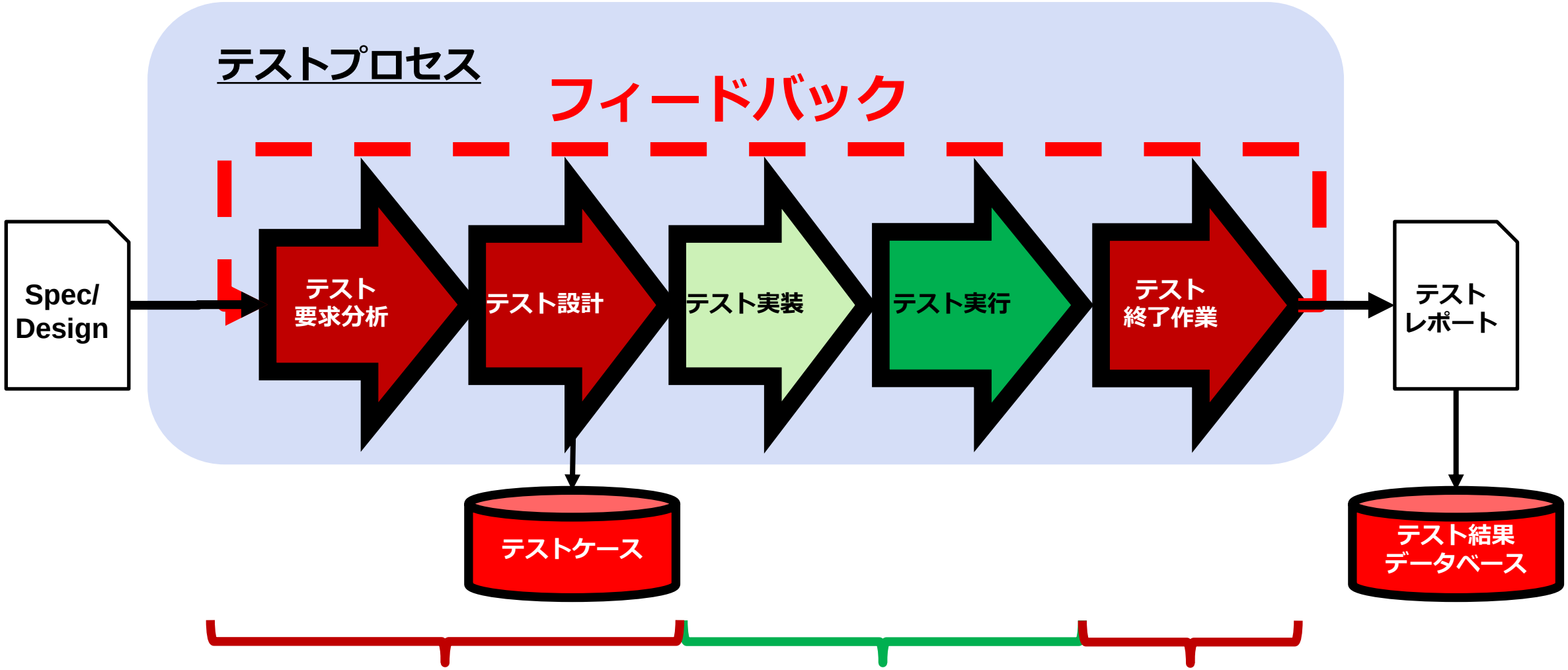
テスト自動化の対象領域



現在の自動化

・テスト実行の生産性向上
-コスト削減
-工数の削減
-カバレッジ向上

“次のテスト自動化=テストでのAIの利用”の対象領域とフィードバック



現在の自動化

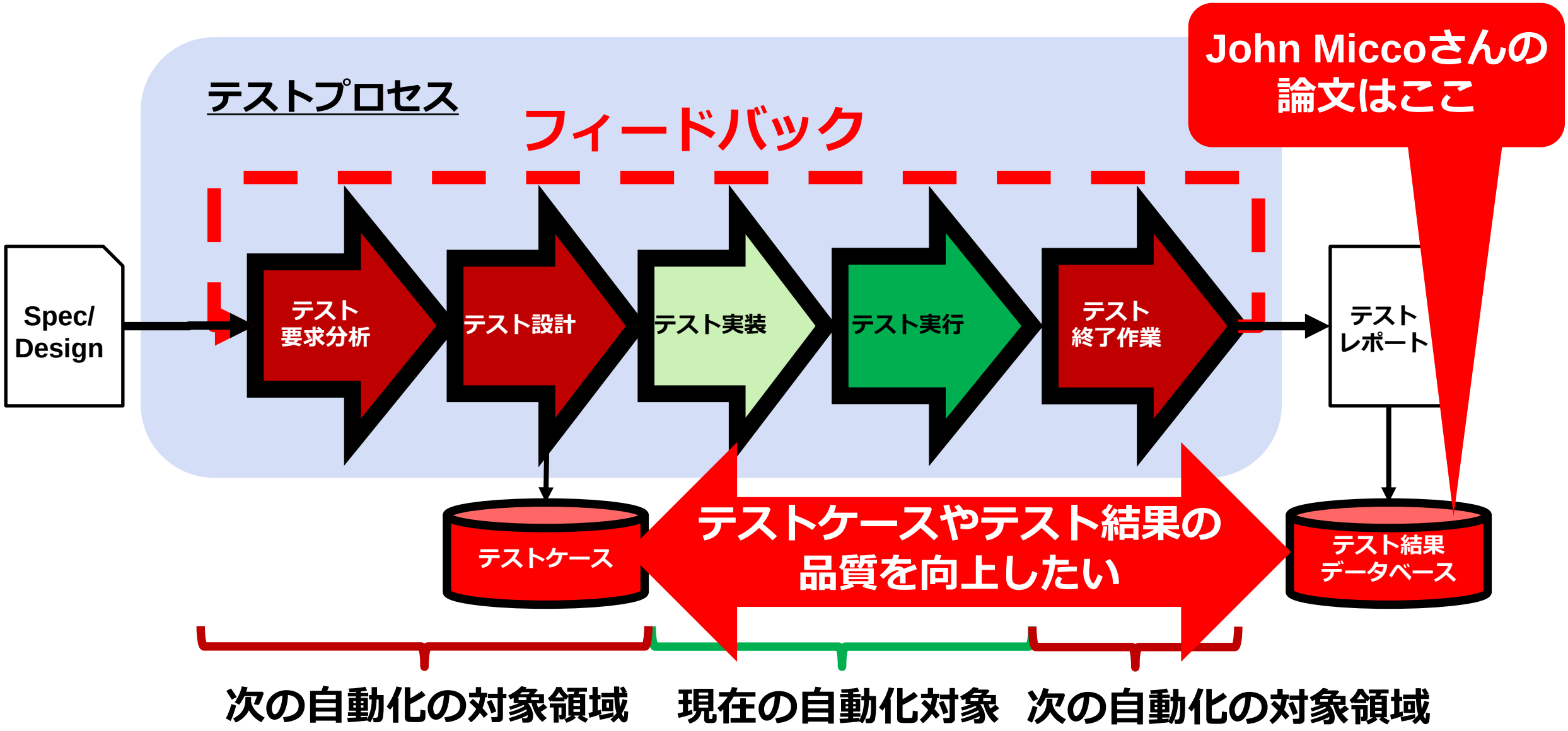
・テスト実行の生産性向上
-コスト削減
-工数の削減
-カバレッジ向上

次の自動化

・テスト設計の改善
・テスト終了作業の改善

⇒テストプロセスに
フィードバックループを
作ることで可能に

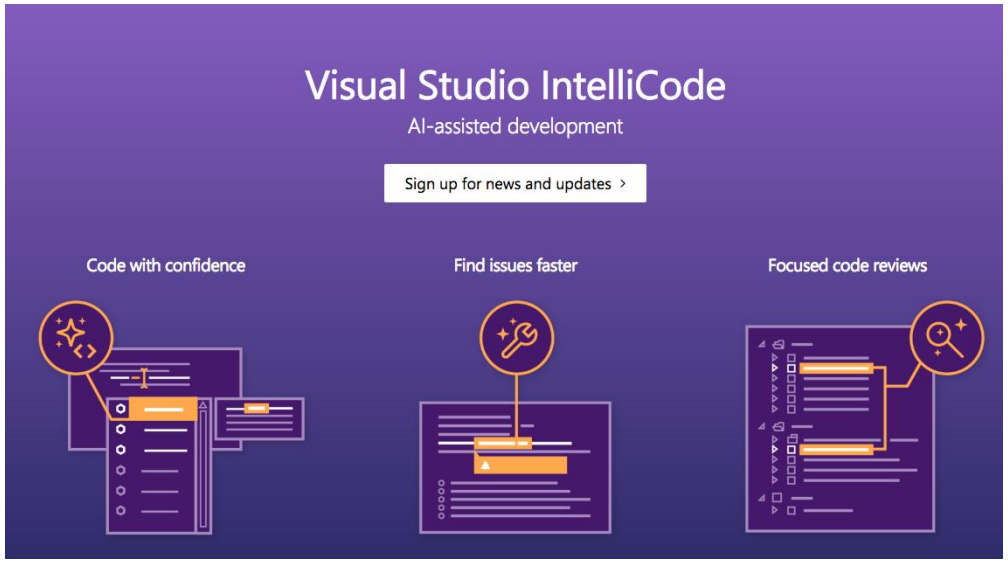
“次のテスト自動化=テストでのAIの利用”の対象領域とフィードバック



“次のテスト自動化=テストでのAIの利用”領域の事例

Microsofot: Visual Studio IntelliCode

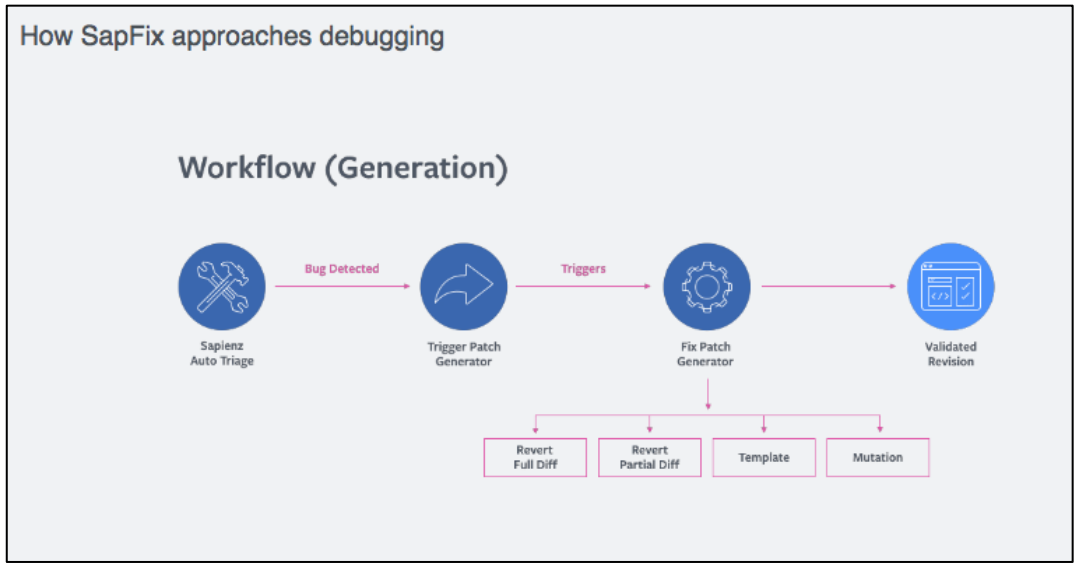
・コードフォーマッターの自動化
・少しスマートな静的解析



Visual Studio IntelliCode
https://visualstudio.microsoft.com/services/intellicode/
2018/11/13 アクセス

Facebook: SapFix and Sapienz

・バグ修正の自動化



“Finding and fixing software bugs automatically with SapFix and Sapienz”
https://code.fb.com/developer-tools/finding-and-fixing-software-bugs-automatically-with-sapfix-and-sapienz/
2018/11/13 アクセス

”次のテスト自動化=テストでのAIの利用”の実施可能要件

・フィードバックループを可能にする
逐次的なテストプロセス

・テストケースとテスト結果コーパス構築を可能にする
テスト自動実行技術

・AI技術利用を促進する
AIとテスト技術に精通したエンジニア

自動化しないための言い訳を探している場合ではない！！

”次のテスト自動化=テストでのAIの利用”の実施可能要件

・フィードバックループを可能にする
逐次的なテストプロセス → **アジャイル**

・テストケースとテスト結果コーパス構築を可能にする
テスト自動実行技術 → **テスト自動化**

・AI技術利用を促進する
AIとテスト技術に精通したエンジニア