Microsoft

Agile Japan 2016

文化の壁をぶち壊せ！
日本でも出来る
本物の DevOps ジャーニー

マイクロソフト　コーポレーション
シニアテクニカルエバンジェリスト DevOps

牛尾　剛

株式会社カラダメディカ
部長代理

野澤　英歩

牛尾　剛 | @sandayuu

DevOps シニア テクニカル エバンジェリスト

DevOps/アジャイルの組織導入 15+年
インターナショナルチーム文化

技術・英語勉強系コミュニティ

著者　オブジェクト脳、英語勉強法

ヴォーカリスト

1994 年 IT 業界へ、2015 年マイクロソフト入社

DX DevOps Technical Working Group
グローバル チームメンバ

DevOps ハッカソン

国内外カンファレンスでの講演経験も多数

Team DevOps – David Tesar and Me

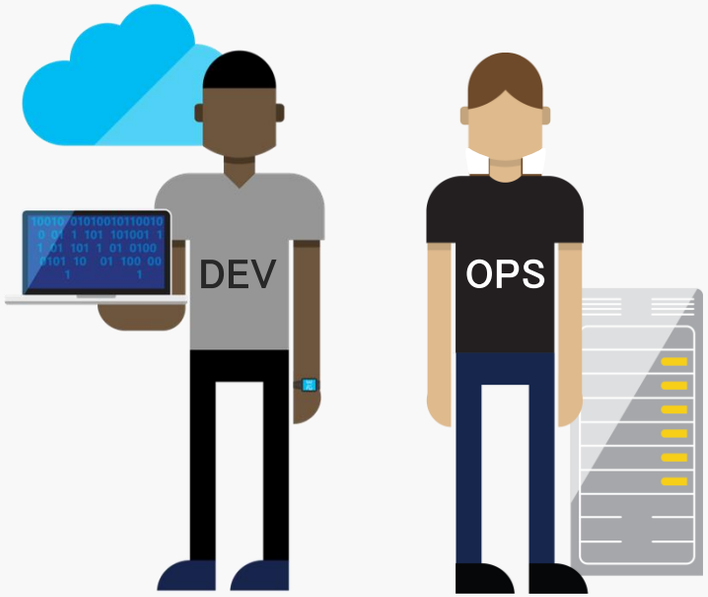
Live DevOps in Japan
ブログ、DevOpsインタビュー（日・英）
Channel 9 動画（日・英）
メソッド屋のブログ （生産性と文化）

http://blogs.technet.com/b/livedevopsinjapan/
https://channel9.msdn.com/Blogs/livedevopsinjapan/
http://simplearchitect.hatenablog.com/

新しいソフトウェア技術や
考え方の導入を
USと同じ速度にする

DevOps とは、人・プロセス・プロダクトの集合体で継続的にエンドユーザに価値を提供することである

– Donovan Brown



1	人

2	プロセス

3	プロダクト

DevOps とは、人・プロセス・プロダクトの集合体で、
継続的にエンドユーザに価値を提供することである

– Donovan Brown

10 デプロイ／日も可能

1	人
2	プロセス
3	プロダクト

DevOps のメリット

出典: https://puppetlabs.com/

コードのデプロイ速度 30倍

リードタイム 1 / 200

エラー 1 / 60

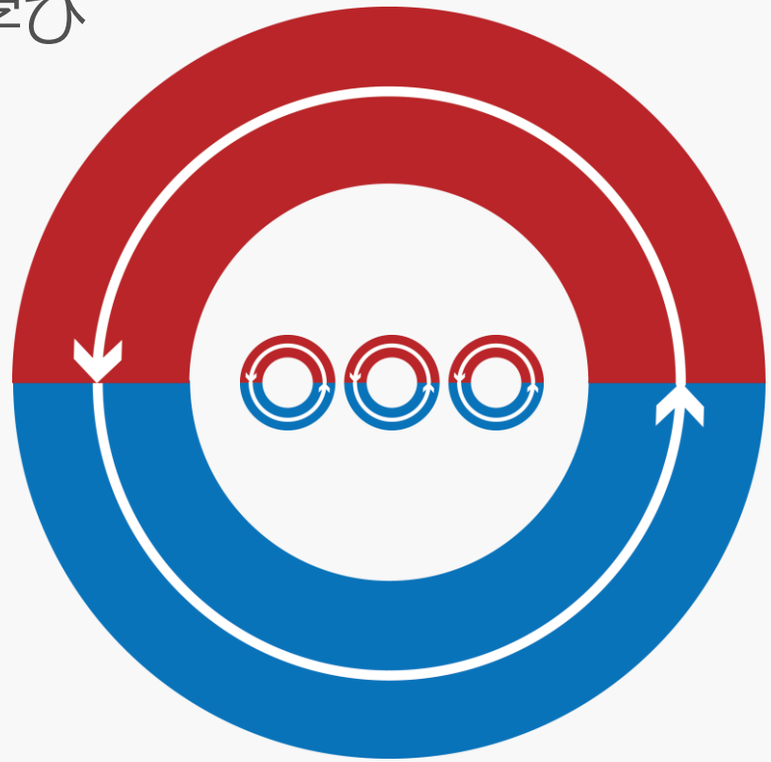
エラーからの復旧 168倍

標準的な DevOps のビュー

(aka Gene Kim's "Three Ways")

1 way： リードタイムの短縮
2 way： 本番環境 / ユーザからのフィードバック
3 way： 継続的な実験と学び

Development
Business



Operations
Customer

【DevOps Enterprise 2015 参加レポート】第 2 回 - DockerによるDevOpsソフトウェアサプライチェーンの改善
http://blogs.technet.com/b/livedevopsinjapan/archive/2015/11/13/devops-enterprise-2015-2.aspx

DevOps プラクティスの一覧

主要なプラクティス

- Infrastructure as Code (IaC)
- 継続的インテグレーション
- 自動テスト
- 継続的デプロイ
- リリースマネジメント
- アプリ パフォーマンスの監視
- ロード テストと自動スケーリング

サブプラクティス

- 可用性監視
- 変更/構成管理
- 機能フラグ（フューチャーフラグ）
- 環境へのプロビジョニングの自動解除
- セルフサービス環境
- 自動回復 (ロールバックとロールフォワード)
- 仮説に基づく開発
  - 運用環境でのテスト
  - フォールトインジェクション
  - 使用状況監視/ユーザー テレメトリ

http://www.itproguy.com/devops-practices/

DevOps スタータキット

DevOps プラクティス紹介動画

プレゼンテーション資料

DevOps プラクティスをいつでも学べる

「メソッド屋のブログ」で今すぐ検索！

日本はどの程度遅れているか？

8年ぐらい
遅れてるわ

Alex Vinyar, Chef

Capital, HP, IBM のDevOps を支援

5年遅れ
でんな

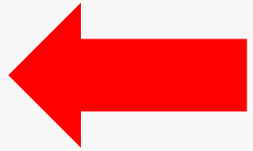
Sam Guckenheimer, Microsoft

マイクロソフトのDevOps ソートリーダー

日本文化をぶち壊して DevOps をインストールする方法

日本向け DevOps 導入ステップ

1. DevOps プレゼンテーション＆デモ
2. Value Stream Mapping
3. 文化とギャップ要素のインストール
4. DevOps ハックフェスト
5. モニタリングと継続的改善



Value Stream Mapping

リードタイムに関係するプロセスの可視化と共有、課題の共有

Value Stream Map の例

文化の違いによる Agile 導入難度

Agile / DevOps の日本導入はそのままでは難しい

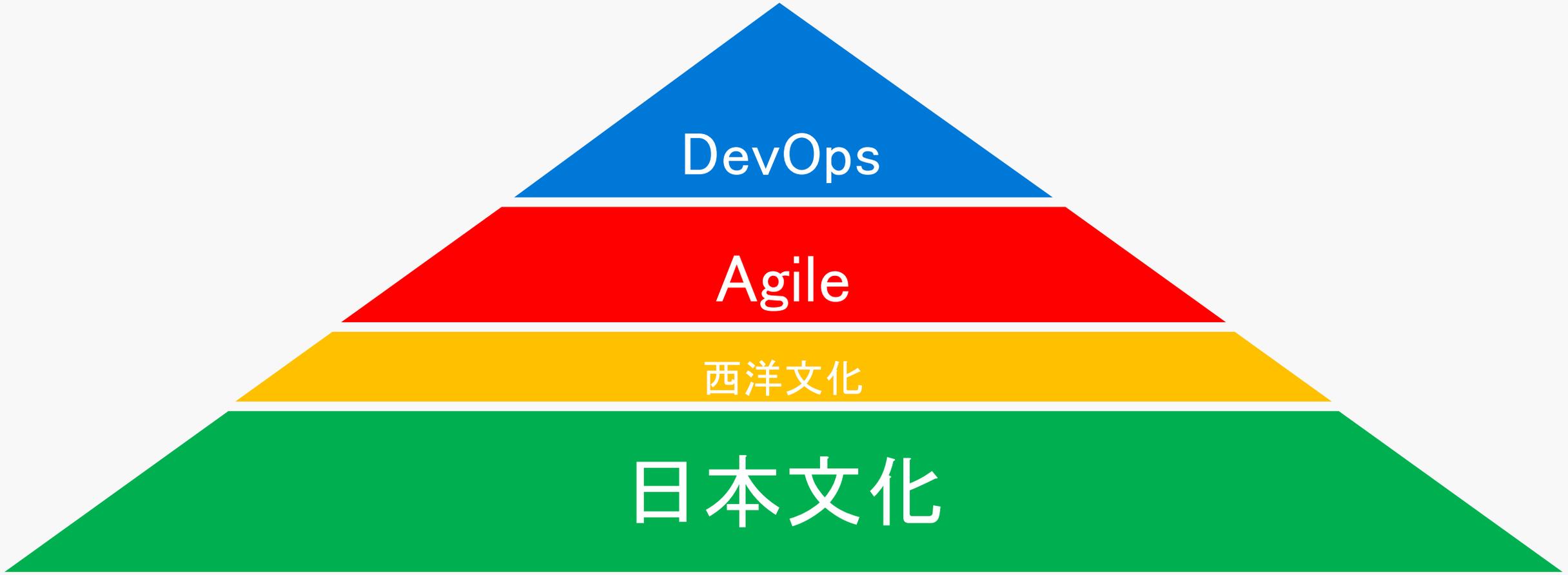
	Agile 導入難度	権力の差 (PDI)	個人の自立 (IDV)	男性社会 (MAS)	不確実性忌避 (UAI)	長期指向 (LTO)
日本	**80**	54	46	**95**	**92**	**80**
フランス	66	**68**	71	43	86	
イタリア	65	50	76	70	75	
アメリカ	49	40	**91**	62	46	29
イギリス	45	35	89	66	35	25
ドイツ	55	35	67	66	65	31

http://www.clearlycultural.com/geert-hofstede-cultural-dimensions/power-distance-index/
Alistair Cockburn Twitter: https://twitter.com/TotherAlistair/status/731591878777417729

文化をインストールする

DevOps は 西洋文化の上に成り立っている

「文化」をインストールすれば良い



インターナショナルチームでの気づき

物事を高速にこなしているのではなく、同じ価値に対する物量が少ない。

お客様のやりたいことが明確で、やり取りも物量も少ない。

意思決定・ゴールに無理がなくロジカル。

あるがままの自分を受け入れてもらえて「常識」が存在しない。

KPIがあるのみで、あとは自分で考える。上下関係・指示・承認がほぼ無い。

バカバカしい質問でも気軽に質問する

「楽しんでいるか？」が最も重視される。

米国と比較した日本の単位時間当たりの生産性比較

対

米国　100%

日本　62%

日本の生産性の動向 2015年版　http://www.jpc-net.jp/annual_trend/

インターナショナルチームの生産性の秘密

物量が違う

Be Lazy

Drew Robbins

Be Lazy

より少ない時間で、成果を最大化する

書籍「エッセンシャル思考」より

	非エッセンシャル思考	エッセンシャル 思考
考え方	みんな・すべて ・やらなくては ・どれも大事だ ・全部こなす方法は？	より少なく、しかしより良く ・これをやろう ・大事なことは少ない ・何を捨てるべきか？
行動	やることをでたらめに増やす ・差し迫ったものからやる ・反射的に「やります」 ・期限が迫ると根性で頑張る	やることを計画的に減らす ・本当に重要なことを見極める ・大事なこと以外は断る ・あらかじめ障害を取り除いてお
結果	無力感 ・何もかも中途半端 ・振り回されている ・何かがおかしい ・疲れ切っている	充実感 ・質の高い仕事ができる ・コントロールしている ・正しいことをやっている ・毎日を楽しんでいる

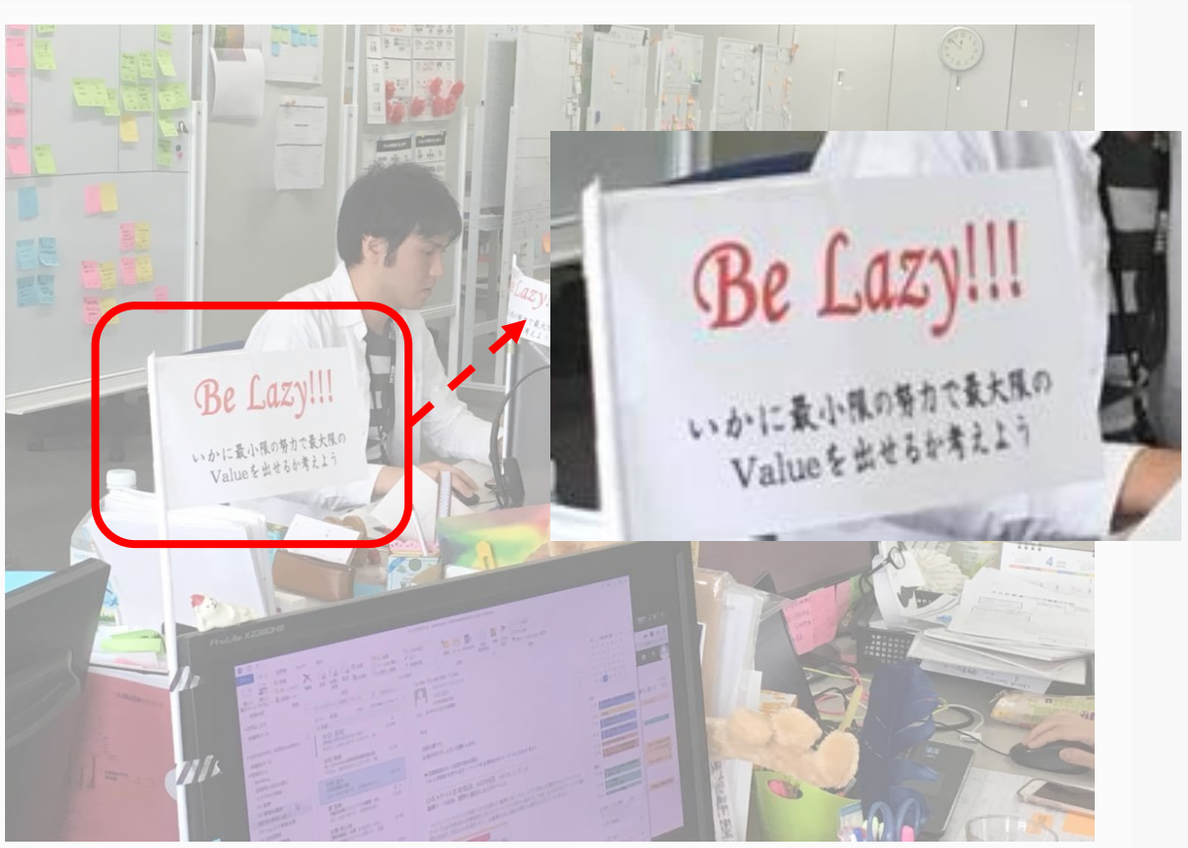
世の中のほとんどのことは「ノイズ」である

Ask for Help

Aleks, Julien, and Damien

Karadamedica

VSM・自動化でリードタイム 13日が3日にMS技術フル活用！
チーム/PO/UCD 自ら考え、協力し行動して成果を出す文化に



詳細は後半をお楽しみに！

文化インストールの詳細に関してはこちら

メソッド屋のブログ

http://simplearchitect.hatenablog.com/

ソフトウェア生産性の向上と、
インターナショナル文化と
DevOps の考え方に関するブログ

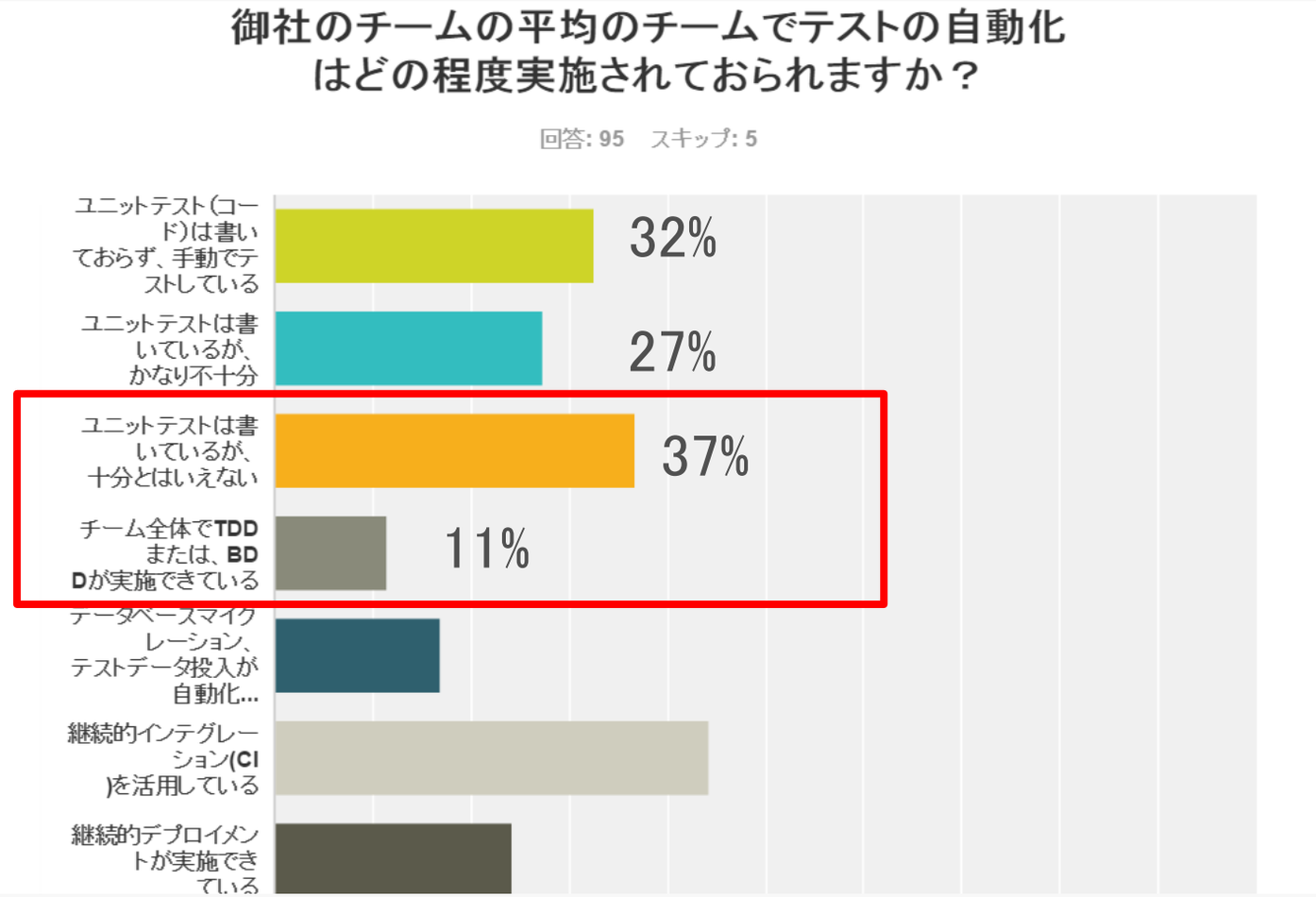
DevOps 導入ギャップ要素のインストール

上位マネージメント / 関係者の巻き込み

イケてるベンダ / アジャイルコーチの選定

アジャイル開発の実施

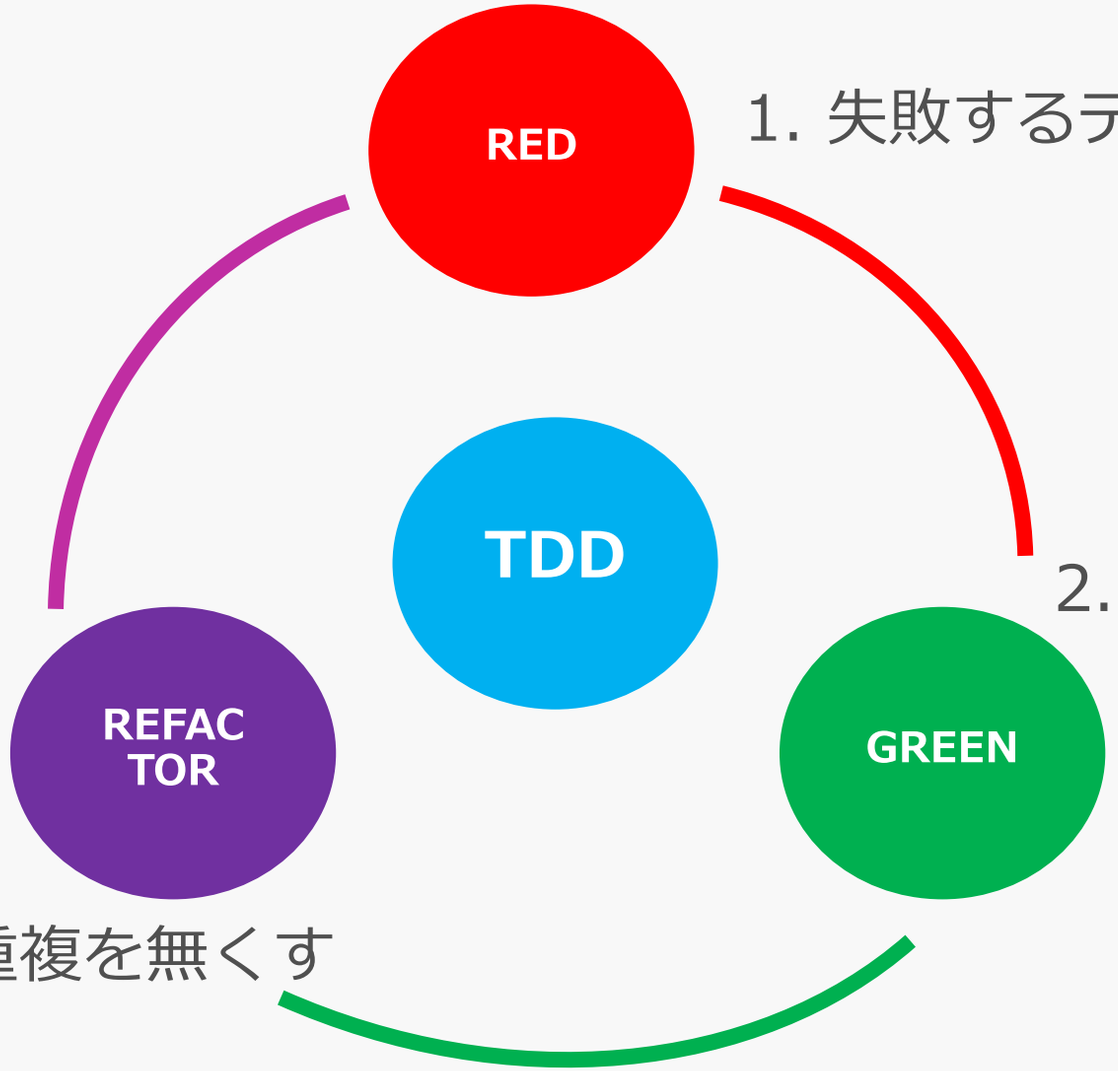
アジャイルプロジェクトの自動テスト率



テストをある程度ちゃんと書けている
アジャイルプロジェクトの割合

48%

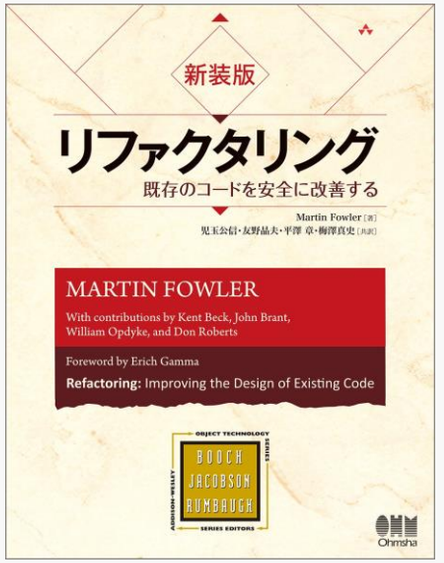
テスト駆動開発



1. 失敗するテストを書く

2. コードを動くようにする

3. 重複を無くす



http://labs.nintex.com/code-dojo-test-driven-development/

テスト駆動開発を知る

6 Minutes DevOps：
テスト駆動開発

https://channel9.msdn.com/Blogs/livedevopsinjapan/6min-DevOps-10

DevOps ハックフェスト

エキスパートと一緒に自動化するハッカソン実施

ハックフェスト in Microsoft

ペアプログラミング

定期的に実施し、DevOps Dojo として、いつでも学べる場にするのもよい

モニタリングと継続的改善

定期的にリードタイム等の改善状況を共有する

Demo day

記念撮影

西洋文化を学んで実践すれば
アジャイルは上手くいく！

Karadamedica 事例

自己紹介

野澤　英歩（のざわ　ひでゆき）

株式会社カラダメディカ
プロダクト企画部　部長代理

株式会社カラダメディカって？

医師、薬剤師、栄養士、
カウンセラーといった
専門化に健康に関する悩みを
365日24時間相談できる
Webサービスを運営しています

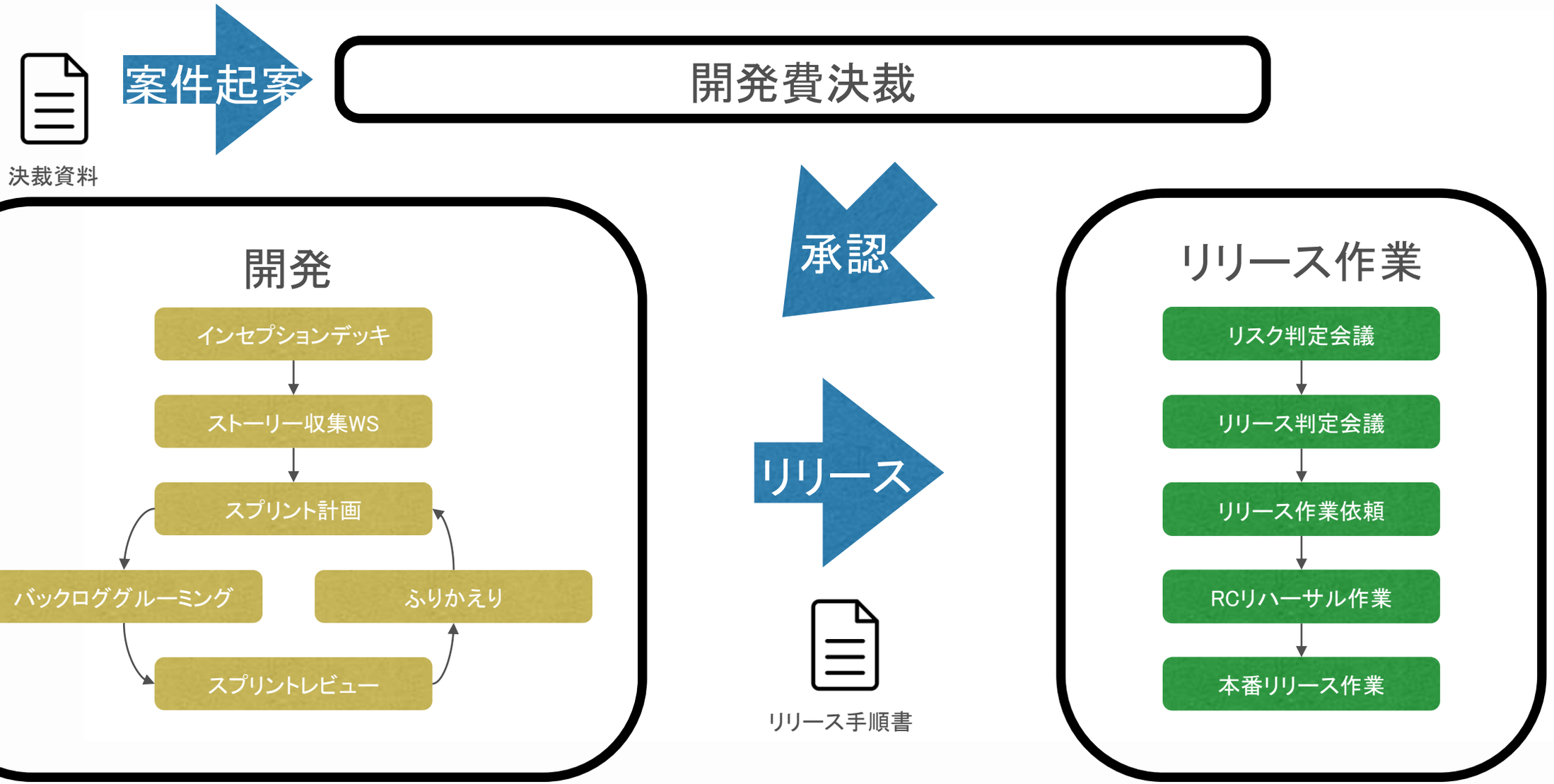
+KARADAmedica
カラダメディカ

DevOps 導入の経緯

- 非効率な作業をしている気がする
- その原因がわからない
- どうしていいかわからない

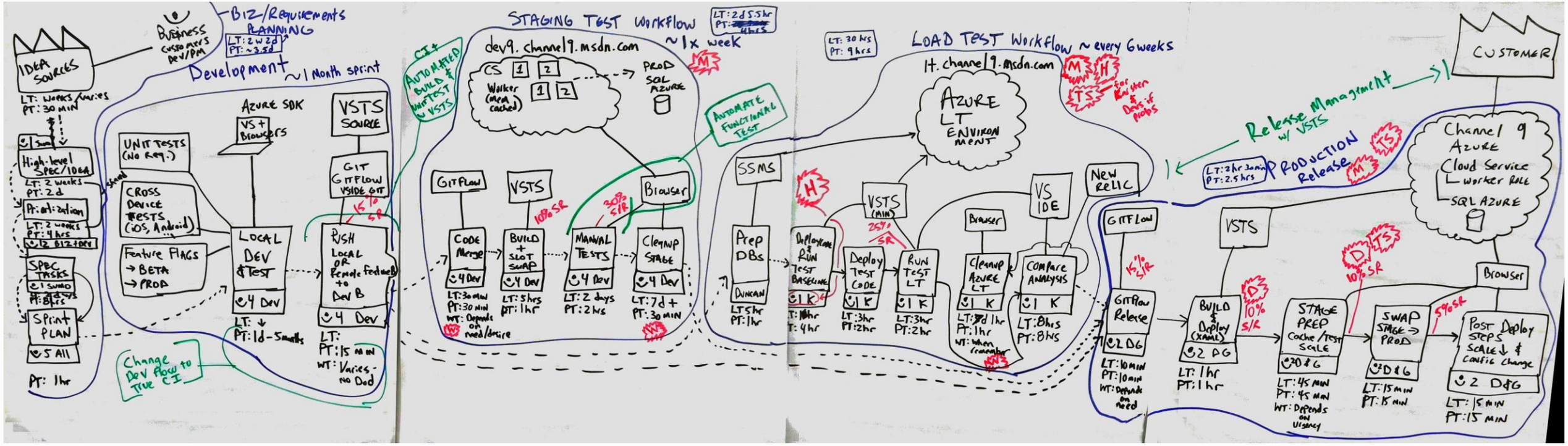
この現状を何とかしたい！

カラダメディカのリリースサイクル



バリューストリームマップで問題点の抽出

バリューストリームマップとは、開発終了-リリースまでの全工程を見える化して所要時間、問題点を洗い出す手法です。



Value Stream Map の例

検出された問題点

1. リリース作業が手作業メイン
2. リリースに必要な資料が多い
3. リリースの手順が多すぎる
4. リリースの決定権が現場にない
5. インフラ担当が複数部署兼任

でも・・・

今までもずっとそうやってきたし、そもそも**会社の方針**を変えるなんて無理でしょ？
品質も納期も全部大事だから時間がかかるのはしょうがないでしょ。

ただ、それって本当に会社の方針なの？

ということで**社長**に聞いてみました

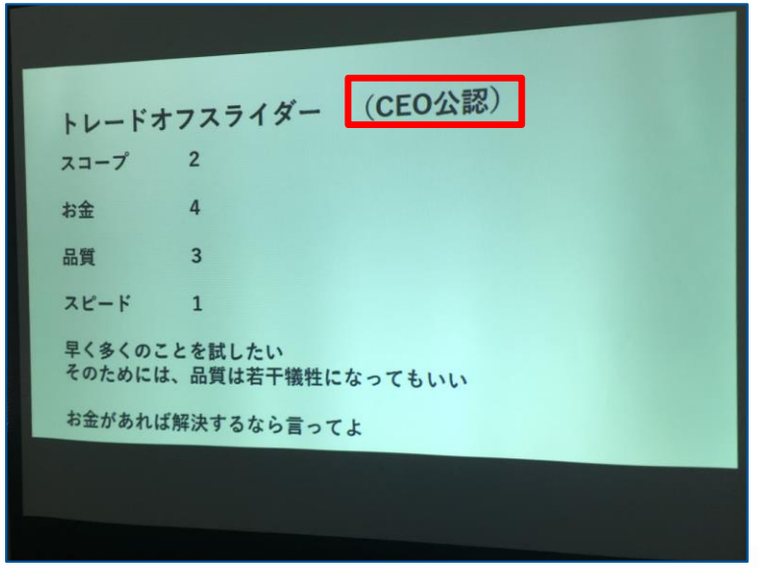
現場
「社長にとって
一番重要な事って
何なんですか？」

社長
「リリースサイクルを回すスピード**です」**

現場
「え…！そうなんですか？」

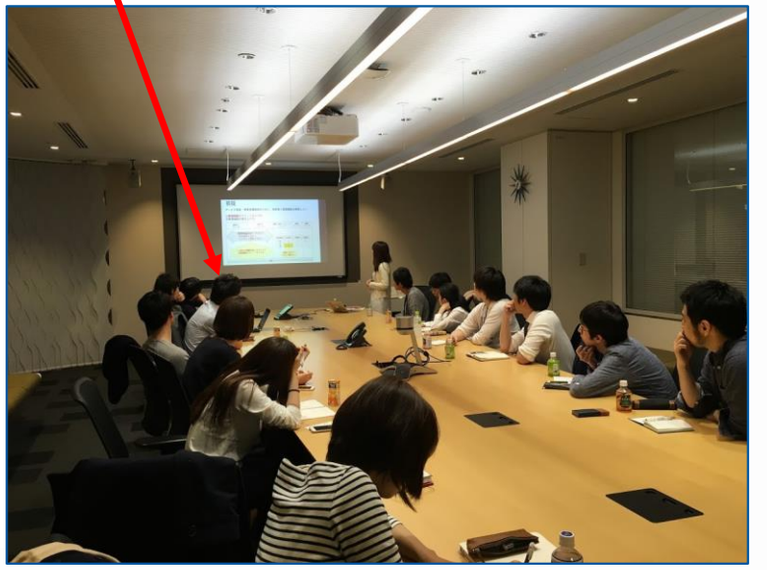
もっと早く巻き込んでおけばよかった…

それからというもの事あるごとに社長を巻き込みました



社長

社長



さあ、優先順位が決まれた後は改善です

一番大事な事は何かを常に考えながら色々改善していきました

改善①

リリース作業が手作業メイン

リリース作業はほとんどが定型作業ですので機械に任せた方が早い上に間違いがありません！

手作業=安全という幻想は捨ててもらい
「自動化」する為の時間を捻出しました！

改善②

リリースに必要な資料が多い

作って終わりの資料が多すぎる。
「テスト報告書」？「レビュー記録票」？
バグ発生時にそれ見て何か解決しますか？

作っても誰も見ない資料は作る必要がないと判断しましたので、リリース時にはリリースノートのみ作成するようにしました。

改善③

リリースの手順が多すぎる

リスクを必要以上に心配し過ぎるから手間だけが増えていくんです。運営を揺るがすような問題はそうそう発生しません！

そもそもバグの発生を0にすることは不可能なので、バグが発生することは前提に考えてもらうようにしました。（ただし、重要なバグはすぐに直せる体制にする）

改善④

リリースの決定権が現場にない

現場の事は**現場が一番良く知っています**。
そもそもリリース前の**出口チェックでは後戻り作業を誘発**して非効率です。

リリースの承認を**決裁時**に行ってもらうようにしました。また、**大幅な仕様変更や新たなリスク**が顕在化したら、**リアルタイムに再承認**をもらうようにする形にしました。

改善⑤

インフラ担当が**複数部署兼任**

兼任していることによって、**作業の手待ち**が発生したり、**チームに所属できない**等の**非効率な事態が発生**しています。

定型作業を自動化することで**捻出された時間をチームに所属する時間に割り当てる**ように調整しました。

結果

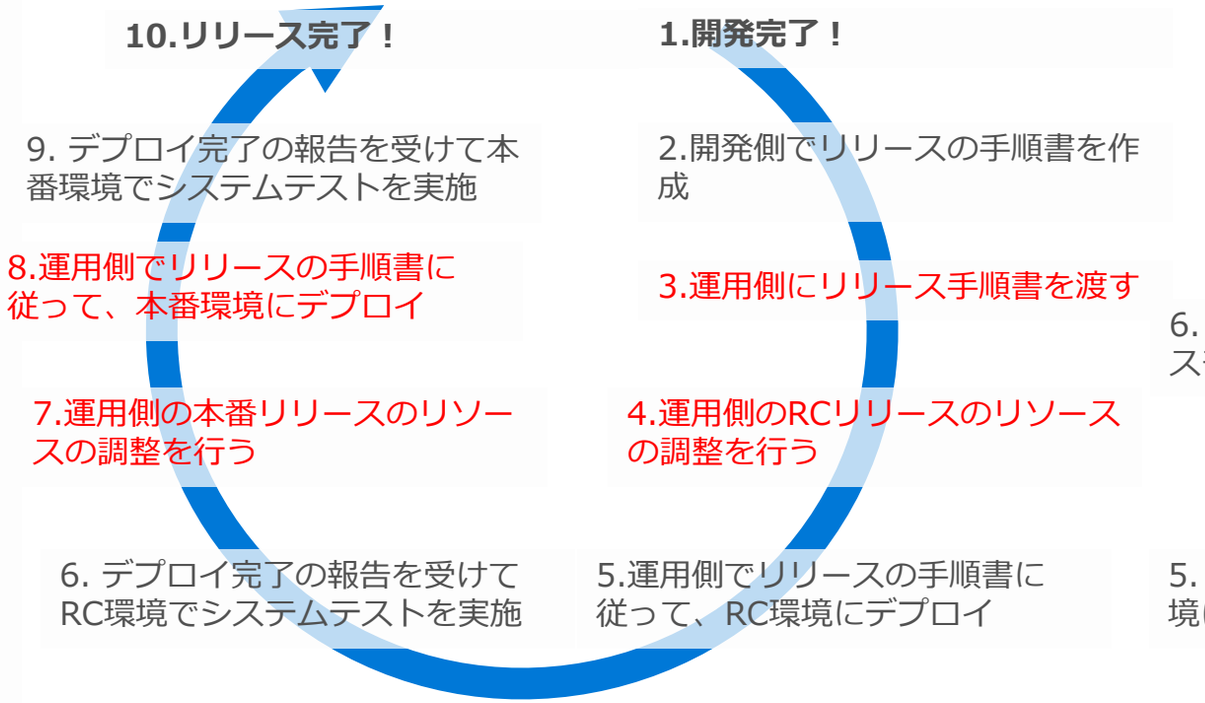
これらの活動により、開発完了～リリースまでのリードタイムが13日→3日に短縮されました！

さらなる改善

ですがまだまだ改善の余地はあります。マインド、プロセス面の強化だけでなく、**定型作業の自動化**の取り組みも進めております。

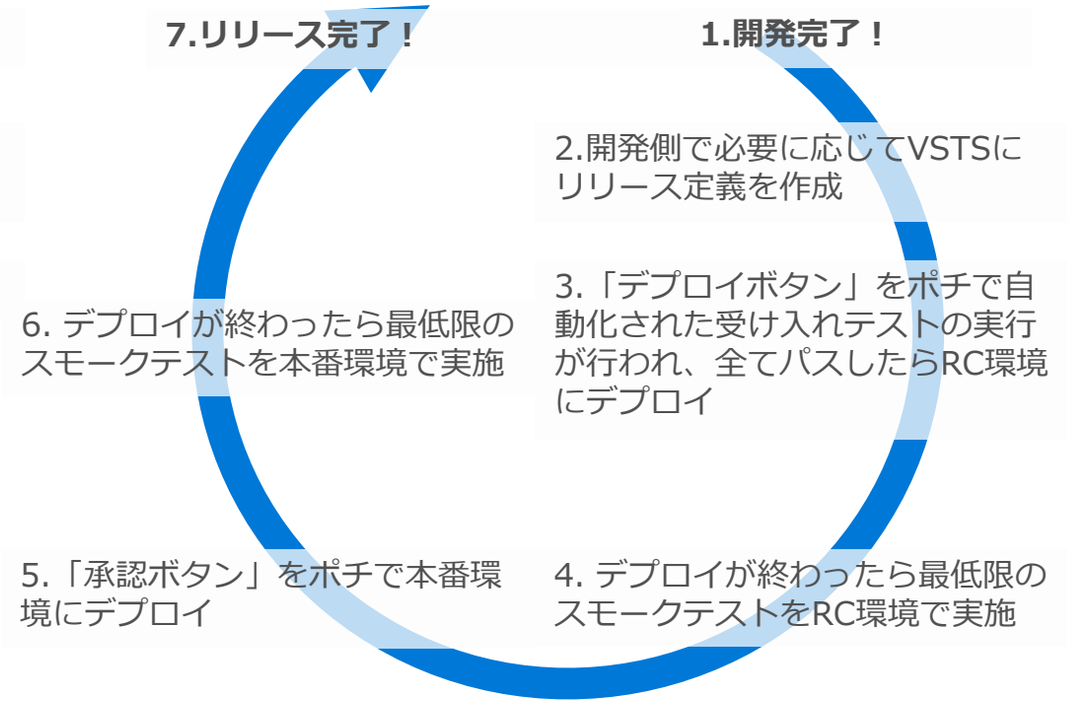
例:リリース作業の自動化

改善前



赤字：Hand-Off（手渡し）作業

改善後



でも・・・

それってカラダメディカみたいなスタートアップの企業だからできたんでしょ？

いいえ、そんなことはありません！

我々もトップダウンの気質が強い会社でした。
ただ、今までのやり方を続けていても何も変わらない事に気づいて、**社長含むメンバー全員で変えていこうという決断**をしました。

大事なもの

- 高い目標
- 変えたいという気持ち
- 出来ると思う気持ち
- Be Lazyのマインド

心がけたこと

1. 上を巻き込む（活動の承認を得る）
2. トレードオフを意識する（全てを救おうとしない）
3. とにかくやってみる
4. 全員が参加する
5. 焦らずに一つ一つ確実に終わらせる
6. やりづらいことは続けないですぐに改善
7. 常にムダを意識して不要なことはやらない
8. 悩んだらすぐに周りに相談
9. 失敗を恐れない（むしろ喜ぶ）
10. 複雑なことはシンプルにする

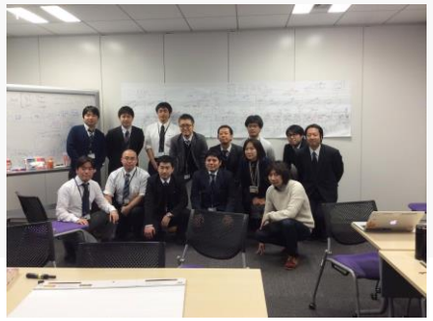
Appendix：VSTSおススメです！

VSTSとはVisual Studio Team Servicesの略でアジャイルでの開発に特化した統合管理環境です。
カンバン、ソース管理、タスク管理、バグ管理、課題管理、リリース管理（CI&CD）、テスト管理といったあらゆるものを一元管理できます。
また、強力なクエリ機能でVSTS上のあらゆるデータのトラッキングが行えます。（Excelとの連携もできます）
更に5人のユーザーまで無償で使えるのも魅力の一つです。
手作業やExcelでのタスク管理に手詰まりを感じているのであれば、是非一度お試しいただくことをおススメいたします。
何より、ブラウザとマイクロソフトアカウントさえあれば
何でもできるというのが最大の魅力です。

最後にお知らせ

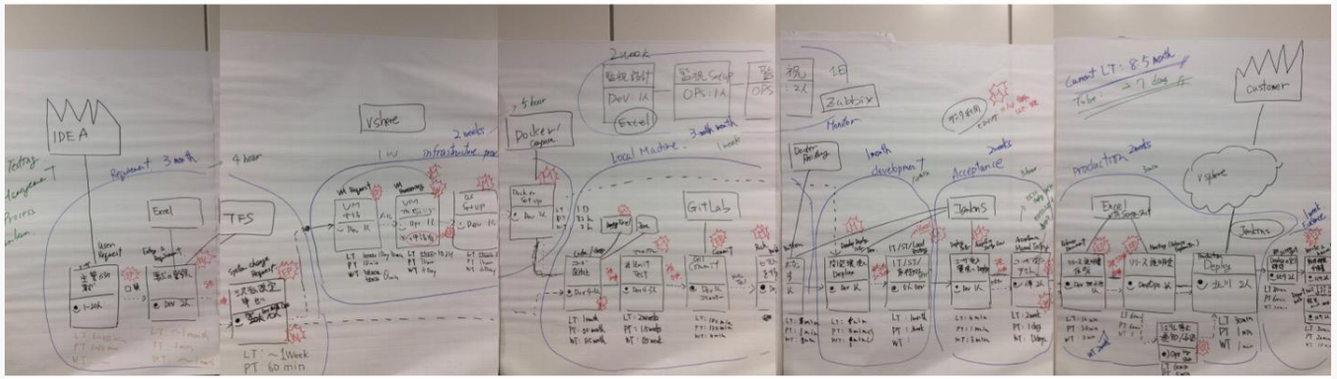
NEC Solution Innovators

エンプラ文化を変革　リードタイム　8.5カ月 ->1週間に
Docker / Compose / Terraform / Serf + VSTS / Azure を用いたハイブリッドクラウドストーリ

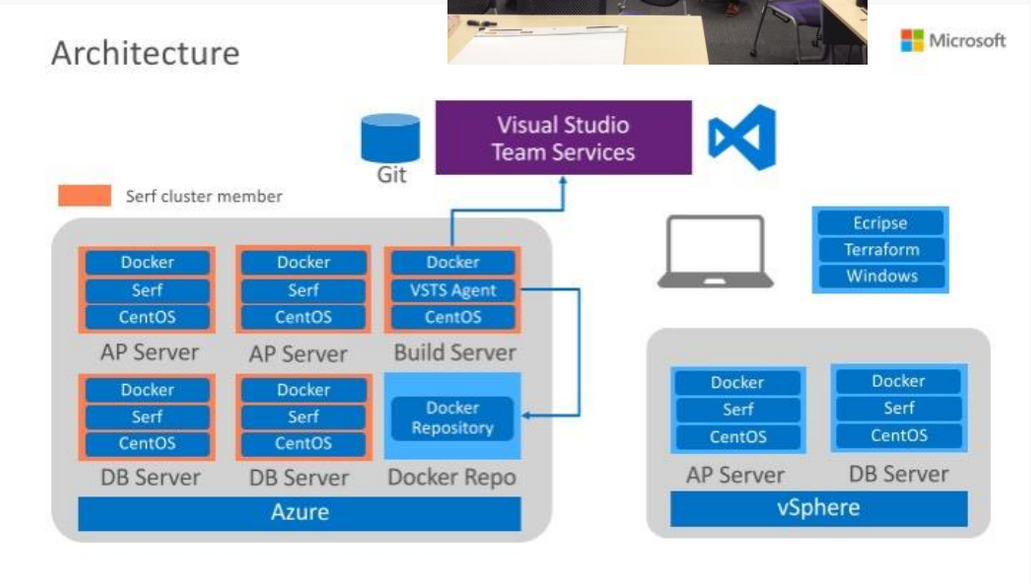


Microsoft

Value Stream Map



NEC Solution Innovators



「今まで、1990年代の開発だったのが、2017年の開発をしている気分です」– 福井様

DevOps ハッカソン

6/11, 12 土、日
DevOps ハッカソン
入門編

6/18, 19 土、日
アドバンスド DevOps ハッカソン
黒船襲来スペシャル編

DevOps ポーズを決める黒船の皆さん

案内ページ： http://devopsjp.connpass.com/