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
株式会社カラダメディアク 株式会社カラダメディアク 部長代理
牛尾 剛
野澤 英歩
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ー
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

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）

Team DevOps - David Tesar and Me

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

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

出典: https://puppetlabs.com/
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

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
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/
「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

5年遅れでんな

Alex Vinyar, Chef
Capital, HP, IBM のDevOps を支援

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 ハックフェスコト
5. モニタリングと継続的改善
Value Stream Mapping

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

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

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
インターアショナルチームでの気づき

物事は高速にこなしているのでなく、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

米国 100%  対  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

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

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！
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

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/
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. コードを動かすようにする
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
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

案件起案 → 開発費決裁

決裁資料

開発

インセプションデッキ
ストーリー収集WS
スプリント計画
バックロググルーミング
ふりかえり
スプリントレビュー

承認

リリース

リスク判定会議
リリース判定会議
リリース作業依頼
RCリハーサル作業
本番リリース作業

リリース作業

リリース手順書
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。
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う？
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う。
ただ、それって本当に会社の方針なのか？
ということでお世話になりまして、
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
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
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ないんです！

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！
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」？バグ発生時にそれを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。
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）
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。
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。
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

1. 開発完了！
2. 開発側でリリースの手順書を作成
3. 運用側にリリース手順書を渡す
4. 運用側でリリースの手順書に従って、RC環境にデプロイ
5. デプロイ完了の報告を受けてRC環境でシステムテストを実施
6. デプロイが終わったら最低限のスモークテストをRC環境で実施
7. デプロイが終わったら最低限のスモークテストを本番環境で実施
8. 運用側でリリースの手順書に従って、本番環境にデプロイ
9. デプロイ完了の報告を受けて本番環境でシステムテストを実施
10. リリース完了！

改善後

1. 開発完了！
2. 開発側で必要に応じてVSTSにリリース定義を作成
3. 「デプロイボタン」をポチで自動化された受け入れテストの実行が行われ、全てパスしたらRC環境にデプロイ
4. デプロイが終わったら最低限のスモークテストをRC環境で実施
5. 「承認ボタン」をポチで本番環境にデプロイ
6. デプロイが終わったら最低限のスモークテストを本番環境で実施
7. リリース完了！

赤字: Hand-Off (手渡し) 作業
でも・・・
それってカラダメ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
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
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ー

「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/