

アジャイルプラクティスガイドブックを携え、 チームで現場を変えていく

2023/8/24

エンタープライズアジャイル勉強会

常松祐一



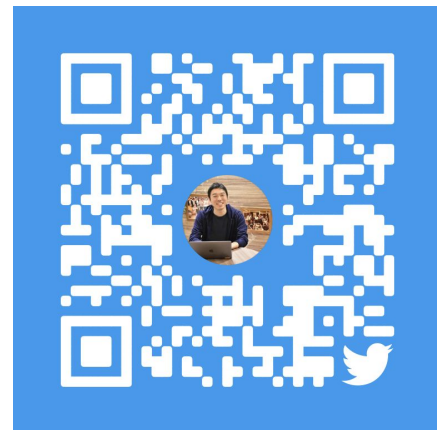
常松祐一

Retty株式会社

プロダクト部門長 執行役員 VPoE

立ち食い蕎麦担当

顧客にとって価値のあるプロダクトを、チーム一丸となって協力し、短期間にリリースする開発体制のあり方を模索しています。



アジャイル開発(大規模スクラム LeSS)や エンジニアリングマネジメントに関する発信をしています

「1プロダクトをみんなで作る！」
Rettyでの大規模スクラム (LeSS)導入記

エンジニアリングマネージャーとしてどんなことをしているのか？

Retty Tech Blog

2019-12-08

「1プロダクトをみんなで作る！」Rettyでの大規模スクラム(LeSS)導入記

#組織 #アジャイル開発

64 ツイート 159 シェア

この記事は Retty Advent Calendar 2019 8日目の記事です。

Retty Advent Calendar 2019 - Qiita
Rettyのエンジニアが書くAdventCalendarです。

Qiita

qiita.com

- 目次 -

- はじめに
- LeSSを選択した背景
- LeSS展開のプロセス
 - 1チームスクラム期(4月~6月)

Hatena Blog tuneの日記

tuneの日記

2021-09-05

エンジニアリングマネージャーとしてどんなことをしているのか？

開発プロセス 育成 組織 採用 マネジメント

プロフィール

tune

このブログについて

注目記事

スプリント期間中にバックログアイテムを消化し切ったらどうする？PBLの上から取る？ #SDNonline

帆船のふりかえり / Sailboat Retrospective

【目次】

- はじめに
- エンジニアリングマネージャーとは？
- メンバーのサポート・育成・評価
 - メンバーの状態観察
 - 自覚設定・人事評価
 - 後進の育成
 - 日常の労務管理
- 開発
 - プロダクトマネジメント
 - エンジニアリングのリーダーシップ
- 採用・採用広報・アドバイザーの招徠
 - 採用
 - 採用広報
 - アドバイザーの招徠
 - 他社との情報交換
- 終わりに

はじめに

自身のアジャイル開発経験

- ・メーカーに13年在籍し、最後3年は新規事業開発に従事
- ・組織上の職位は主任研究員 (=課長代理)
- ・開発プロセスはスクラム
スクラムでの役割は当初 PO → 後に SM へ

メーカー
新規事業

Retty

- ・グルメメディアの開発・運用に4年間従事
- ・組織上の職位はマネージャー → VPoE
- ・開発プロセスはスクラム (大規模スクラム (LeSS))
スクラムでの役割は社内コーチ、ステークホルダーの 1人

アジャイルプラクティスガイドブックの紹介

アジャイルプラクティスガイドブック チームで成果を出すための開発技術の実践知



チーム・組織にプラクティスを導入し、根付かせるために!

116の手法を一冊にまとめた“実践”の手引き

著者 : 常松 祐一(著)、川口 恭伸(監修)、松元 健(監修)

発売日 : 2023年7月20日

定価 : 2,860円(本体2,600円+税)

出版社 : 翔泳社

書籍目次

第1章 アジャイルな開発を支えるプラクティス

- 1.1 プラクティスの実践
- 1.2 高速に石橋を叩いて渡る
- 1.3 広く知られたアジャイル開発手法とプラクティス
- 1.4 プラクティス理解に役立つ考え方

第2章「実装」で活用できるプラクティス

- 2.1 実装方針 / 2.2 ブランチ戦略 / 2.3 コミット
- 2.4 コードレビュー / 2.5 協働作業 / 2.6 テスト
- 2.7 長期的な開発／運用ができるソースコード

第3章「CI/CD」で活用できるプラクティス

- 3.1 継続的インテグレーション
- 3.2 継続的デリバリー
- 3.3 継続的テスト

第4章「運用」で活用できるプラクティス

- 4.1 デプロイ／リリース
- 4.2 モニタリング
- 4.3 ドキュメント

第5章「認識合わせ」で活用できるプラクティス

- 5.1 関係者との認識合わせ
- 5.2 開発内での認識合わせ
- 5.3 計画の継続的な見直し

第6章「チーム連携」で活用できるプラクティス

- 6.1 チームの基本単位
- 6.2 属人化の解消
- 6.3 パフォーマンスの測定
- 6.4 円滑なコミュニケーションのアイデア
- 6.5 意識を揃えるワークショップ

アジャイルプラクティスガイドブックの エレベーターピッチ

アジャイルプラクティスガイドブックは悩めるマネージャーと開発者の皆さんに向けて、アジャイル開発とその周辺の技術プラクティスについて説明を試みたもの。

マネージャーとして実際に技術プラクティスを組織に導入する仕事をしている常松が実体験に基づき書いた、技術プラクティスを「チーム全体で」学習するための道標となる書籍。

伝統的な組織で取り入れるために

皆さんが抱えてそうな悩み事

1. 社内の経験者が少ない
2. システムアーキテクチャがアジャイルに向いていない
3. 契約や社内ルールの制約を乗り越えるのが大変
4. 変化に対する一般的な抵抗
5. マネジメントの理解・支援がない、弱い

※エンタープライズアジャイル勉強会 セミナーアンケート2023年4月分より

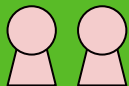


過去に従事したアジャイル開発事例の紹介

Web技術を使ったオンプレの画像解析システム

- PoC・展示会向けに試作したものを製品化
- PjM・開発・QA含め最盛期は30人前後が参加

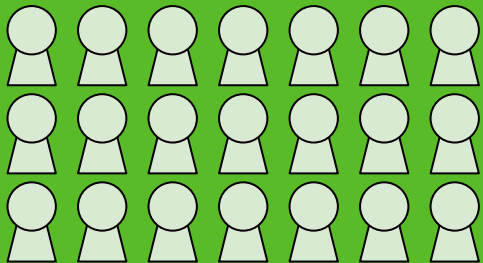
PjM



Webフロントエンド：
Angular



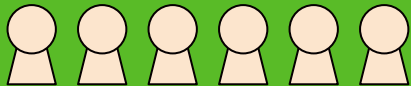
Dev



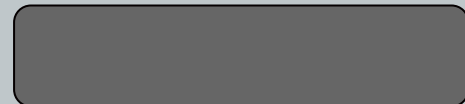
Webバックエンド：
ASP.Net Core / ASP.Net



QA



画像解析：
C#, C++



事例紹介を通じて伝えたいこと

- 経験者がいなくても、“アジャイル開発を学びながら”取り組める。
- 伝統的な組織であっても“少しずつ”変えていける
- “事業ドメインや採用技術を問わず”アジャイルな技術プラクティスの導入はできる

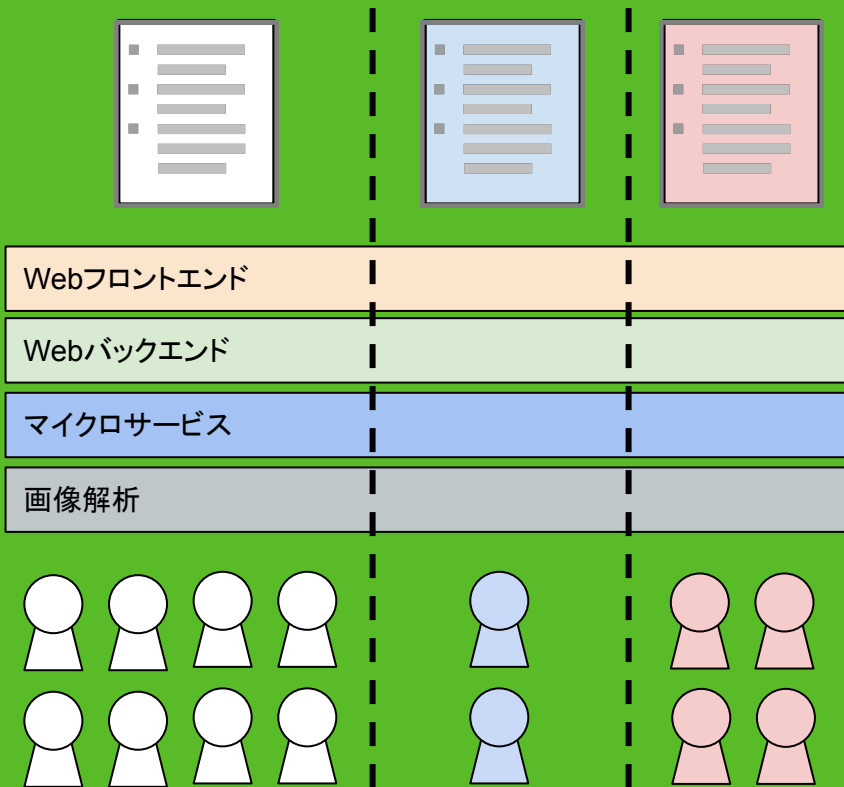


1. チーム分け

- フィーチャーチーム (書籍 6.1)

チームをどう分けるか？

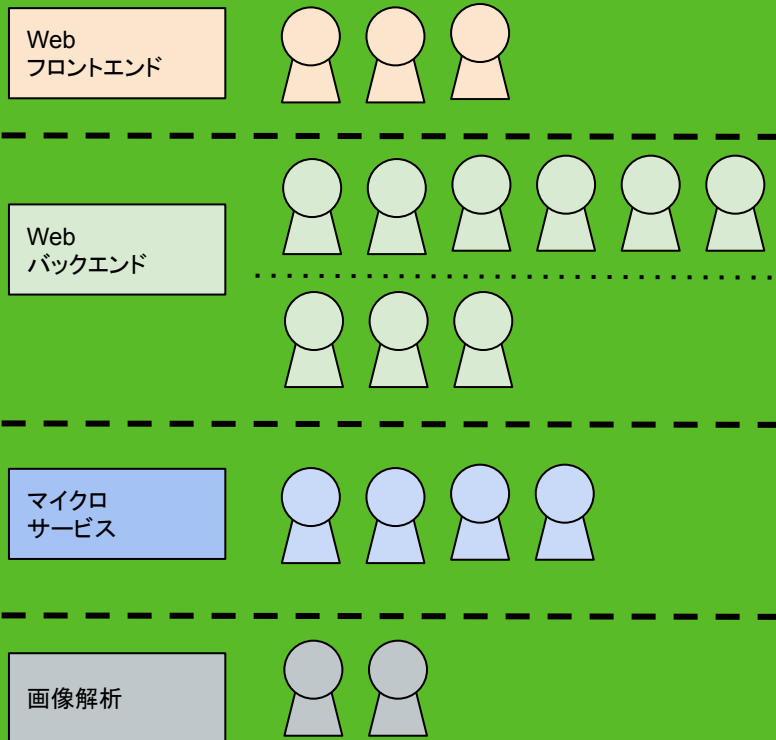
機能・役割でチームを分けてみたが・・・



- 「機能追加」と「技術負債返却」チームができたが、大変不評
- 機能追加の枠を見直してみたが、機能名がそのままチーム名になり、新規の取り組みを始めようとすると、どこが担当するかで揉めてしまった。

チームをどう分けるか？

職能ごとにチームを分ける提案を受けたが...

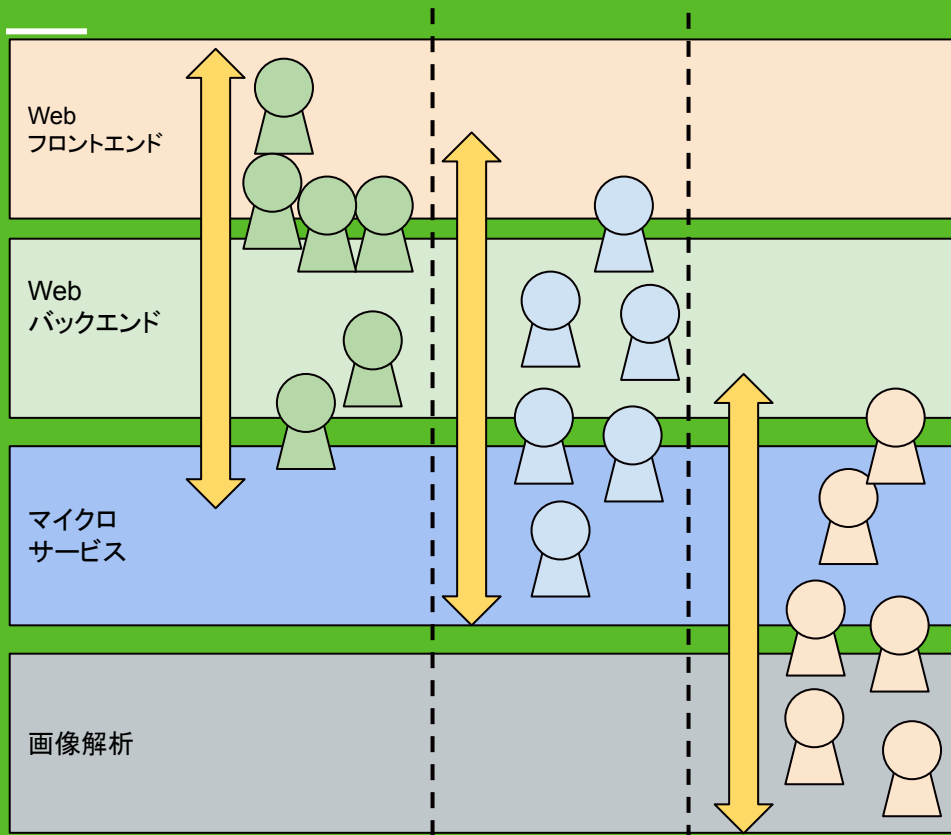


要件が固まったのでコンポーネントごとに人を当て、開発を加速してほしい。

- 要件固まってない...
- コード行数は増えるかもしれないが、どこかが遅れると機能がリリースできない。
- 会議・調整が増える

チームをどう分けるか？

▷フィーチャーチームを構成し、チーム名をつける



- チーム単独で顧客に価値が提供できる体制にする。
- 機能と関係無いチーム名をつけ、チームの結束を促す。

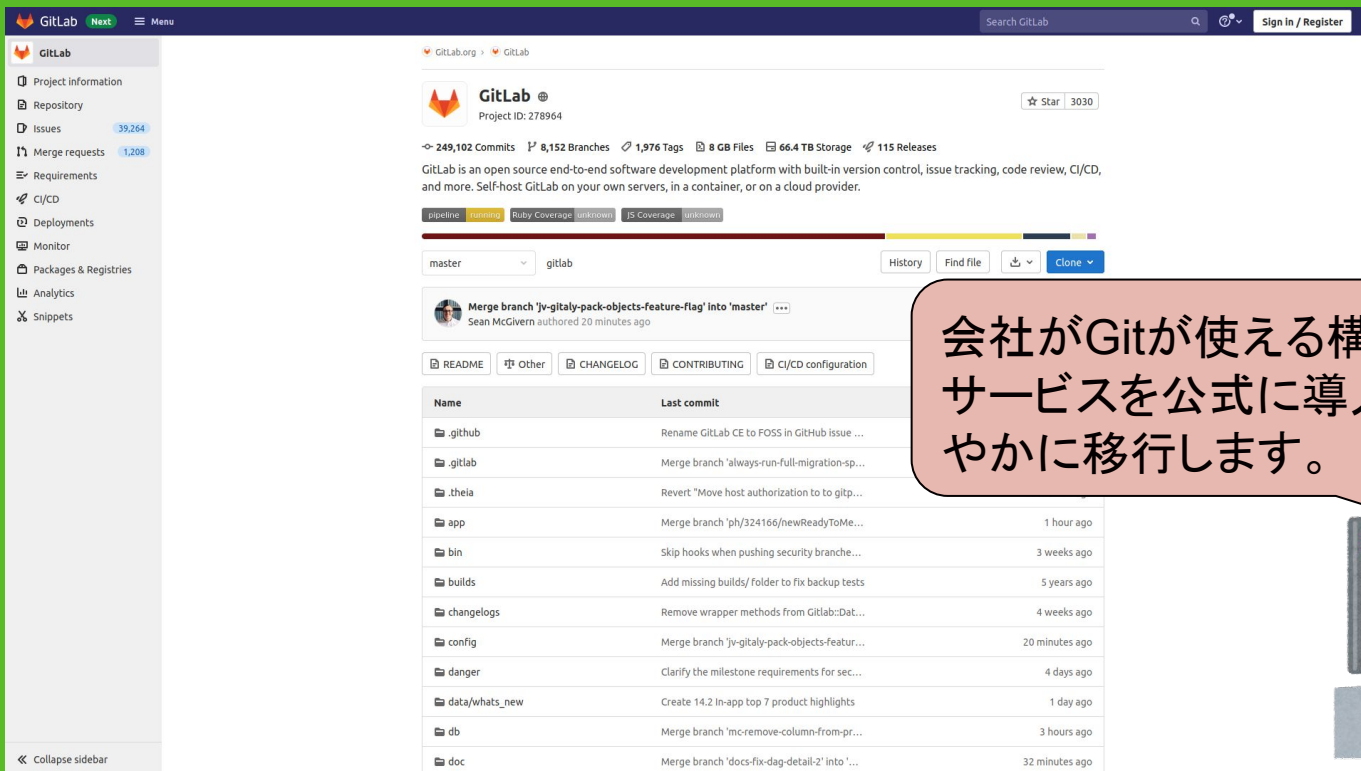
※フルスタックエンジニアが必要という話ではない
※プロダクトの全領域を全チームがカバーしなければならないという意味でもない

2. 実装

-
- GitLabを自前管理のサーバーで建てる
- トランクベース開発 (書籍 2.2)

Gitが使える構成管理サービスが無かった

▷GitLabを自前管理のサーバーで建てる



The screenshot displays the GitLab web interface for the 'gitlab' project. The top navigation bar includes the GitLab logo, a search bar, and a 'Sign In / Register' button. The left sidebar contains a navigation menu with options like 'Project information', 'Repository', 'Issues', 'Merge requests', 'Requirements', 'CI/CD', 'Deployments', 'Monitor', 'Packages & Registries', 'Analytics', and 'Snippets'. The main content area shows the project's overview, including the GitLab logo, project ID (278964), and statistics: 249,102 Commits, 8,152 Branches, 1,976 Tags, 8 GB Files, 66.4 TB Storage, and 115 Releases. A description states: 'GitLab is an open source end-to-end software development platform with built-in version control, issue tracking, code review, CI/CD, and more. Self-host GitLab on your own servers, in a container, or on a cloud provider.' Below this, there are tabs for 'pipeline', 'testing', 'Ruby Coverage', and 'JS Coverage'. A commit history table is visible, showing a merge branch 'jv-gitaly-pack-objects-feature-flag' into 'master' by Sean McGovern 20 minutes ago. The table lists various files and their last commit details.

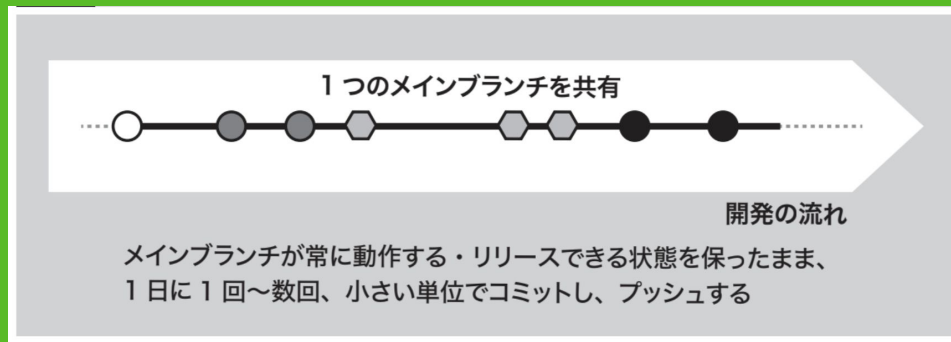
Name	Last commit	Time ago
.github	Rename GitLab CE to FOSS in GitHub issue ...	
.gitlab	Merge branch 'always-run-full-migration-sp...	
.theia	Revert "Move host authorization to gitl...	
app	Merge branch 'ph/324166/newReadyToMe...	1 hour ago
bin	Skip hooks when pushing security branche...	3 weeks ago
builds	Add missing builds/ folder to fix backup tests	5 years ago
changelogs	Remove wrapper methods from Gitlab:Dat...	4 weeks ago
config	Merge branch 'jv-gitaly-pack-objects-featur...	20 minutes ago
danger	Clarify the milestone requirements for sec...	4 days ago
data/whats_new	Create 14.2 In-app top 7 product highlights	1 day ago
db	Merge branch 'mc-remove-column-from-pr...	3 hours ago
doc	Merge branch 'docs-fix-dag-detail-2' into '...	32 minutes ago

会社がGitが使える構成管理サービスを公式に導入したら速やかに移行します。



マージで衝突が多発し時間が取られる

▶トランクベース開発へ移行



- 開発中のソースコードが最新版から離れる時間が短くなり、コードのコンフリクトが発生しにくくなる。
- 最新版が常に結合された状態となり、並行開発によるデグレなどの問題を早期に気づくことができる。
- 自身も当初否定派でしたが1週間で慣れました。

3. CI/CD

- Jenkinsを自前管理のサーバーで建てる
- 継続的デリバリー (書籍 3.2)
- 品質評価の巻き込み

CI/CDを動かすサーバーが無かった ▶Jenkinsを自前管理のサーバーで建てる



The screenshot shows the Jenkins dashboard with a table of jobs. The table has columns for Status (S), Webhook (W), Name, Last Success, Last Failure, and Last Duration. The jobs listed are 'Freestyle project', 'Github Org project', 'Multibranch', and 'Other'. The 'Freestyle project' job shows a successful build with a duration of 41 ms. The 'Github Org project' job shows a failed build with a duration of 2.9 sec. The 'Multibranch' and 'Other' jobs show no recent activity.

S	W	Name	Last Success	Last Failure	Last Duration
		Freestyle project	29 sec - #1	N/A	41 ms
		Github Org project	N/A	20 sec - log	2.9 sec
		Multibranch	N/A	N/A	N/A
		Other	N/A	N/A	N/A

余っているPCを活用して、ビルド・静的解析・テストをガンガン実行させよう！



最新版をドッグフーディングしたい

▷継続的デリバリー

社内に最新版の動作環境(≒ステージング環境)を用意し、機能開発の最後に更新するルールとした。

1. 当初はエンジニアが自前でビルドし、手動で更新する。
2. デプロイ回数が増え面倒になり、Jenkinsでビルドして実行ファイル一式をダウンロードできるようにする。
3. ビルド結果を共有フォルダに置き、バッチファイル実行で自動更新する。
4. 製品の一機能としてインストーラーを作成し、インストーラー自体のドッグフーディングを実施。

製品リリース時にチーム外の品質チェックが必要

▷品質部門の巻き込み

- 製品開発する事業部の外に品質を担保する組織が存在
 - 製品開発完了→品質チェックと進めるとスケジュールが長くなってしまふ
- 開発途中の段階から品質部門を巻き込む
 - 開発スタイル(アジャイル・スクラム)の説明
 - 必要なチェック項目全体の整理
 - 開発中からチェックできる項目と、リリース前にチェックする項目を分ける

4. 監視・運用

- モニタリング(書籍4.2)

使っていると不調がいろいろ見つかる

▷モニタリング

- ログ出力を充実させ、定期的に確認する
- OSSの活用 (Prometheus + Grafana)



5. 認識合わせ

- アジャイルと言わない説明

アジャイルを皆が受け入れてくれたわけではない

▷アジャイルと言わない説明

- 組織が感じていたであろう悩み
 - アジャイル開発の成功体験はない
 - 一方で新事業開発の良い進め方もわからない
- 「アジャイル開発だからこうやる」ではなく、目的に沿って議論し落とし所を探っていく
 - チームの分け方、開発計画の立て方、進捗報告の内容、リーダー(責任者)の立て方、スクラムイベントの運営コスト、スケジュール遅れに対する打ち手 などなど



現場で少しずつ
変化を起こすためのヒント

アジャイルにやれていたが、アジャイルにやりたい
メンバーばかりが集まったわけではない

- アジャイルを好きになった人もいる
- 開発に向き合えていることに満足な人もいる
- ゴール(≡新規事業の成功)に向き合えていることに満足な人もいる
- こういうやり方の現場なのだと従ってくれていた人もいたと思う



アジャイルな開発の実現は長距離走

▶ 目的に則して小さな打ち手を積み重ねていく

- 変えられるものも、変えられないものもある。
- 成功した打ち手の裏にはもっとたくさんの失敗もある。



うまく行った取り組みは記録しておきましょう

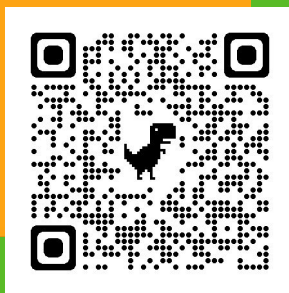
※スクラムフェス大阪 2022登壇資料より抜粋



覗いてみよう！現場のスクラムチーム

Scrum Fest Osaka 2022
2022.06.18

Retty株式会社
今井貴明



Retty

スプリントバックログ

Retty

- MiroのKanbanでタスク管理
- アイテムごとにswimlaneを作る



スプリントプランニング(後半)

Retty

- アイテムをタスクに分解する

スプリント終了までの日割りのタスク量をイメージできるようにラインをひく



対応内容の解像度が高い時はどのクラスにどんなメソッドを作るかまで分解したりもする

タスク分解をした上で改めてアイテムをスプリント中に終わらせられそうか判断する

まとめ

事例紹介を通じて伝えたいこと

- 経験者がいなくても、“アジャイル開発を学びながら”取り組める。
- 伝統的な組織であっても“少しずつ”変えていける
- “事業ドメインや採用技術を問わず”アジャイルな技術プラクティスの導入はできる



おわりに

- 書籍で紹介したアジャイルプラクティスで、あなたのチームを変えていきましょう。
- 書籍で紹介したものは、常松が7年ほどアジャイル開発に取り組む中で取捨選択したものをまとめました。
- 書籍を踏み台に、より良いプラクティスを見つけてください。私もそれを知りたいです！