芯を通す開発を目指して

ー アジャイル"ファン"が
本気でアジャイル開発に取り組んだ2年間 ー

ギルドワークス株式会社
前川博志 a.k.a @Posaune

今日話したいこと

- 開発者として、より良いプロダクトを作っていくための学びと気づきの共有
- 私の話を聞いて、明日から開発者としてサービスに関わる姿勢に少しでも影響を与えられればと思っています

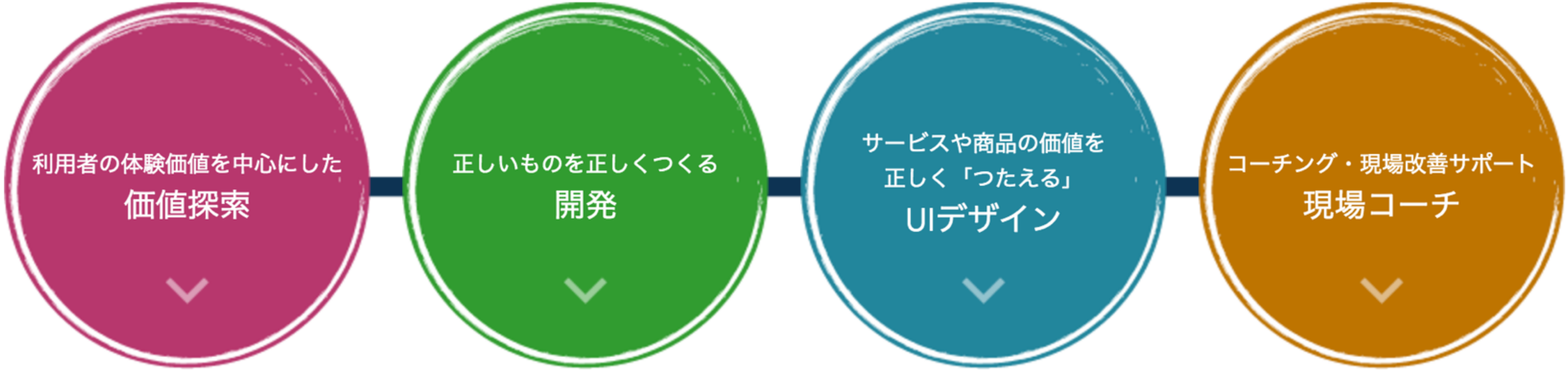
自己紹介

- 前川博志 a.k.a @Posaune （33）
- ギルドワークス株式会社所属のエンジニア（今年から3年目）

- Microsoft MVP for Visual Studio and Development

ギルドワークスについて

ギルドワークスについて



アジャイル “ファン” からアジャイル開発者へ

- “ファン” = 「素振り」がどうしても多くなりがち
- 前職は大企業で、大幅に変えるのが難しく、小さなプラクティスを導入するのが精一杯

- そんな状況から、ギルドワークス中村さんに声をかけて頂き、転職
- アジャイル開発者として、実践を積んでいくことに

ギルドワークスの開発スタイル

- アジャイル + リモートな開発スタイル。開発者は基本リモート（基本は同席しない）
- 1スプリント = 1週間 サイクル でイテレーティブにプロダクトを作っていく
- ユーザーストーリーとして要件をとりまとめ、それを開発者が実装していく

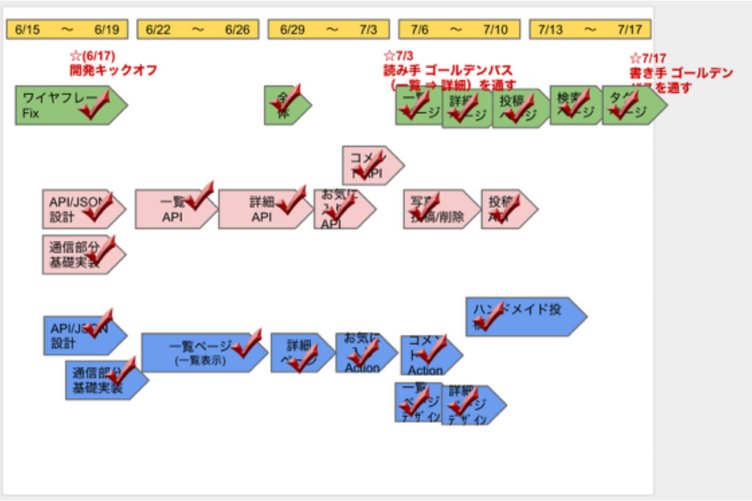
ギルドワークスの開発スタイル

Pivotal Tracker

ユーザーストーリーの管理

Basecamp

Google Docs



開発用ドキュメント・記録

GitHub

ソースコード管理・レビュー

circleci

継続的インテグレーション

ギルドワークスのミッション

正しいものを正しくつくる

「正しくつくる」難しさ

どこまでストーリーに書き込む？

- あるストーリーの機能追加をレビューしたが、こちらの想定とは違う挙動をする箇所があった
- 開発者が「あるべき」として判断して追加実装した箇所
- レビュー指摘で直ぐに修正依頼したが、一週間スプリントでは1日の手戻りも結構インパクトは大きい

- 詳細までしっかりと書き込んで「この通り作って」とするべきなのか？？

「正しさ」と「スピード」どちらを優先する？

- 開発佳境な時期、あるプルリクエストで気になるデータマイグレーションが。
- ただしその変更をマージしないと後続の機能の開発が止まってしまう。

- 一旦変更点としては見送って、改善ストーリーとして積むべき？気になるところは、ここでも止める判断をすべき？

開発を支える2つの芯

アジャイル開発に必要な2つの芯

- プロダクトがユーザーに「どう使われるか」について、チーム全員で意識を合わせること

  = プロダクトの分析

- プロダクトがユーザーに「どう使われるか」が、コード上の機能の実現方法と乖離していないこと

  = プロダクトの設計

なぜ、プロダクトの分析が必要か？

- ユーザーストーリーには「〜したい」という言葉しかない。
- 実際にプロダクトの細かい動きがどうあるべきかを考えるには、「どう使われるのか」まで開発チームとして合わせておく必要がある。

- 「どう分析したか？」の意識を合わせておけば、詳細の決定を開発者に渡すことができる

「どう使われるか」が合意できていなかったら...①

- ユーザーストーリーには大まかな指針でしかない。実際には開発時に多くの事を決定していかなければいけない
- 例：

  ユーザーとして、自分の情報の新規登録を行いたい。

  なぜなら、このサービスで同じ情報を何度も入力したくないからだ

「どう使われるか」が合意できていなかったら...①

- このストーリーだけ開発者は当然ながら実装できない
  - 「必須項目を明示して下さい」
  - 「バリデーション方法を...」
  - 「項目の並び順を...」

- ⇒ すべてをガチガチに明記するのも一つの方法だが、

  本当にどうあるべきかを考えられるのは開発者
- ⇒ 開発者がそれを判断できるだけの情報 = 「どう使われるか」

「どう使われるか」が合意できていなかったら...②

- 受発注システムの発注書出力機能を開発

- ユーザーストーリー：

  管理者として、受注者に発注書を発行したい

- 開発チームは、管理側で発注支持を出した時に、受注側が発注書をダウンロードできるような仕組みを作成し、実装した

「どう使われるか」が合意できていなかったら...②

- しかし実際は、受発注システムで出力したファイルを、必要に応じて加工し、発注者に郵送するのが、現時点のユースケース。

  （発注者が直接ダウンロードするのはありえない）

- 開発者が必要だと考えて実装した機能は、ユーザーの使い方から考えると、想定できないものだった（結果消すことになった）
- ⇒ プロダクトの分析をし、使われ方をチーム内でもっと話せていれば、このような勘違いは起こらなかった

なぜプロダクトの設計が必要か

- ユーザーのフィードバックの数は、「どう使われるか」の基本線上に乗っているものほど多い
- 今の「どう使われるか」にと、プロダクトの作り方を合わせておけば、フィードバックにうまく適用できる

- 逆に、小手先の対応でのごまかしが多くなってくると非常に危険な兆候
- 簡単な変更がうまく実装できないような事態になる

「どう使われるか」通りに作れていなかったら...

- やりたかったこと： CSV出力のソート順変更（降順 ⇒ 昇順）
- 機能的にはソートはあったのに、モデル・コード上ではソート順を表すものは無かった！

- 対応として、一覧IDのセレクトをを逆順で返す対応をした

  ⇒ テストで山ほど影響範囲が見つかり、思いの外苦戦

  ⇒ モデルとしてソート順を示すオブジェクトを結局追加

芯を形作る取り組み

アジャイル開発に必要な2つの芯

- プロダクトがユーザーに「どう使われるか」について、チーム全員で意識を合わせること

  = プロダクトの分析

- プロダクトがユーザーに「どう使われるか」が、コード上の機能の実現方法と乖離していないこと

  = プロダクトの設計

分析の「芯」を作るための取り組み（Try中）

- 詳細を細かく指示するのではなく、前提条件を与えた上で
  あるべき姿を考えてもらう

- ⇒ 「ここは必須として下さい」という詳細の指示ではなく、
  「今のデータの持ち方から行って、これは無くても大丈夫？」
  「業務的にこれが無いと破綻しないかな？」という問いかけへ

分析の「芯」を作るための取り組み（Try中）

- ストーリーの背景に潜む業務はどんなものか、という観点で開発メンバーとディスカッションを行う

- ⇒「こういうことを実現してください」だけではなく、

  「全体の業務としてはこうなっています」「こういうことをしたいので必要です」という流れを、チームに説明する。

分析の「芯」を作るための取り組み（やってみたい）

- ストーリーがある程度決まった時点で、システムテストをみんなで書いてみる

- ⇒ やりたいこと別にシナリオをつくり、そのシナリオに沿ってまずはゴールデンパスを埋めていく、ようなイメージ
- ⇒ 言葉ベースのディスカッションを、せっかくならテストケースという再利用可能な形で残しておきたい
- ⇒ 自動テストとして整備されていると言うことなし

アジャイル開発に必要な2つの芯

- プロダクトがユーザーに「どう使われるか」について、チーム全員で意識を合わせること

  = プロダクトの分析

- プロダクトがユーザーに「どう使われるか」が、コード上の機能の実現方法と乖離していないこと

  = プロダクトの設計

設計の「芯」を作るための取り組み（Try中）

- まずは「名前」に着目する

- クラス名・パッケージ名レベルで、ネーミングをレビューする。
  実際の業務や動きがイメージできるような名前になっているか
  どうかを考える

「名前」のリファクタリング

posaunehm on 12 Jan

[IMO]24時間なので `EXPIRE_TIME_IN_HOUR` なり、単位を明記したほうが後で読みやすそう。

posaunehm on 19 Jan

ここで追加した項目ではないと思うのだけれど、MasterIdって何でしたっけ？？
ちょっと名前が汎用的過ぎるような。

- 一番違和感を表明しやすいのが「名前」
- 違和感をみんなで表明して、意識を揃えていくことも必要

設計の「芯」を作るための取り組み（Try中）

- モデリングの「向き直り」を定期的に行う
- 「ふりかえり」ではなく、今で来ているモデルを受け止めた上で、本来の業務との差分を考えて、どう変えるべきかを考える
- モデルは、DBのデータモデルから、コード上のオブジェクトまで、色々なレイヤーで見直す
- 2〜3ヶ月に一度くらいのペース

モデルの「向き直り」例

- まず “AsIs” を描き、そこに感じている違和感を”ToBe”として出していく

なぜ芯を通す開発をするのか

アジャイル開発に必要な2つの芯

- プロダクトがユーザーに「どう使われるか」について、チーム全員で意識を合わせること

  = プロダクトの分析

- プロダクトがユーザーに「どう使われるか」が、コード上の機能の実現方法と乖離していないこと

  = プロダクトの設計

芯を通す開発で得られるもの

- 開発者が、「どう使われるか」について深く理解し、

  その理解と実際に動くコードを連携させている。

  ⇒ 開発者の越境

- 変更コストの低減
- コミュニケーションの効率化
- etc...

芯を通す開発で得られるもの

- プロの開発者としての自信とプライド

- このプロダクトがどう使われるか分かり、
- それに応じて詳細を自らの意思で決めていき、
- その理解をコードに落とし込む。

- ⇒ そのプロダクトを自分のものにする

ギルドワークスのミッション

正しいものを正しくつくる

「正しいものを正しくつくる」

- 自分自身のプロダクトと思わないと、「正しい」と確信を持ち、それを実現するために「正しく」つくることなんかできない

- 皆さんも、そして私も、もっとプロダクトに向き合っていきませんか？

- 実際に動いているコードは、プロダクトオーナーでもスクラムマスターでもなく、開発者の皆さんが書いたものです。