UZABASE

2022.11.05

ペアプロでテストを書く

Awaken a world of play in business, with our insights.

自己紹介

・藤原 考功 (@trickmrbiz)

・テストエンジニア@ユーザベース
・JSTQB技術委員

・佐賀には初めて来ました！
・その土地の地形を見るのが趣味です

ペアプロはいいぞ、という話をします。

ペアプロとは

ペアプログラミング

2人でプログラミングする

2人が1台のマシンを使ってプログラミングする開発スタイル。キーボードを触っている人をドライバー、そうでない人をナビゲーターと呼びます。

いろいろなスタイルがあるようですが、常に2人で相談しながら進めるので、集中を保ちやすかったり、外部からの割り込みに強かったり(残った一人で作業を進めることができる)といったメリットがあります。

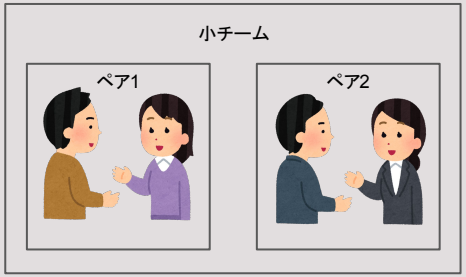
画像は「"ペアプログラミング"」『フリー百科事典　ウィキペディア日本語版』より引用
2020年5月25日 (月) 06:47 UTC
URL: https://ja.wikipedia.org/wiki/%E3%83%9A%E3%82%A2%E3%83%97%E3%83%AD%E3%82%B0%E3%83%A9%E3%83%9F%E3%83%B3%E3%82%B0

わたしたちのチーム

わたしたちのチーム

4~5名規模のチーム

事業全体では100人程度のエンジニアがおり、
プロダクト(機能)単位で4~5名規模の小チームに分かれて開発をしています。
小チーム内では、2ペアが並行して開発を進めます。
ペア内では、ドライバーとナビゲーターの交代を随時行います。
(ペアの組み合わせも、一日に何度か変更します)



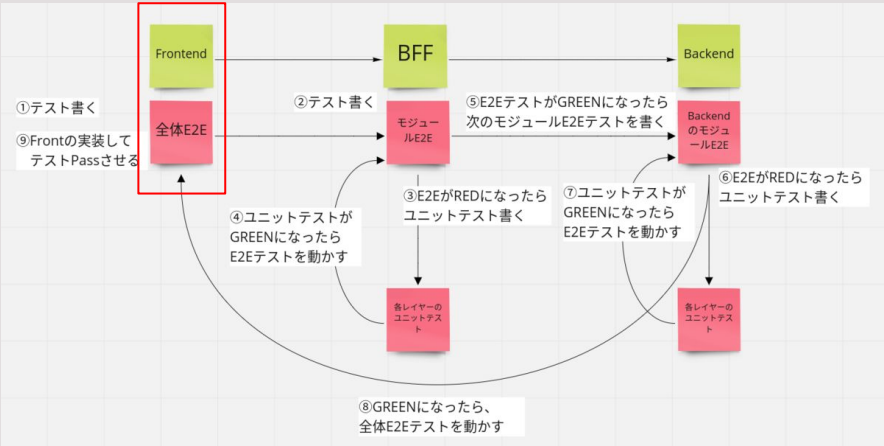
いつでもペアプロ、もといペアワーク

大人数でのミーティングを除いては、なんでもペアでやっています。
プログラミング以外の作業もペアでやるので、ペアワークという方が適切かもしれません。
テストもペアで書いています。

わたしたちのチーム

受け入れテスト駆動開発(ATDD) / テスト駆動開発(TDD)

各ユーザーストーリーの受け入れ基準となるテストを起点に開発を進めます。
テストを使って、いまからやるべき開発のスコープを決めるイメージです(弊社テックブログ記事)。



わたしたちのチーム

受け入れテストの例

```
## ユーザーは、サービスにログインできる
* サービスに、ユーザー"○○"でログインする
* ログイン後のページに"ようこそ、○○さん"と表示される
```


```
@Step("サービスに、ユーザー<userName>でログインする")
fun login(userName: String) {
    loginAs(userName)
}

@Step("ログイン後のページに<welcomeMessage>と表示される")
fun displayWelcomeMessage(welcomeMessage: String) {
    `$`(".welcome-message").shouldHave(exactext(welcomeMessage)
}
```

ペアプロによる恩恵

こんなことありませんか？　その1

重大なテストの漏れに、本番リリースしてから気付いた

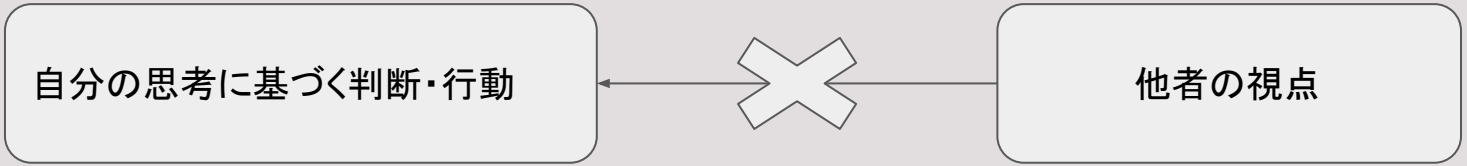
自分なりにしっかりテストしたつもりだったのに、
重大なバグを見つけることができなかった。
ちょっと考えれば気付けたはずなのに、なぜだろう…？

重大なテストの漏れに、レビューを受けて初めて気付いた

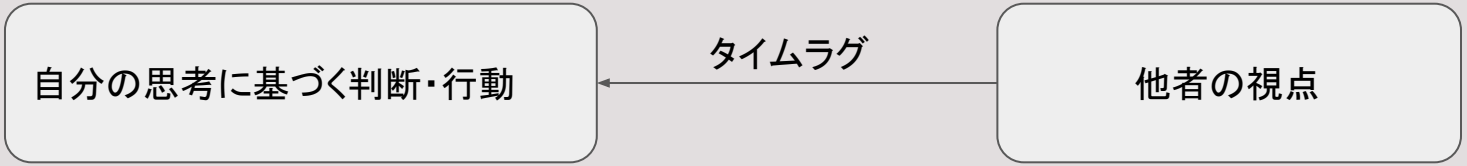
しっかり自己レビューしたつもりだったのに、
指摘を受けるまで自分では気付くことができなかった。なぜだろう…？

原因と思われるもの

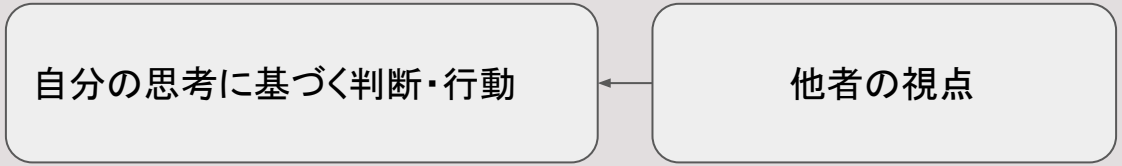
外部(他者)からのフィードバックがない



フィードバックがあったとしても、(大きな)タイムラグがある



ペアプロなら



ペアプロでテストを書く具体例

ペアプロでテストを書く

たとえばこんな風に、テストを書き始めます

```
## 権限Aを持つユーザーは、企業の勘定科目Cを見ることができる
```

👨‍🦱「まずは、いま思っているテスト条件を書いてみますね。」
👩‍🦰「お願いします！」

ペアプロでテストを書く

対話が始まります

```
## 権限Aを持つユーザーは、企業の勘定科目Cを見ることができる
```

👩「『権限A』は限定的な表現ですけど、ここでは例えば権限Bでも表示して良いと思うので、
それらをまとめた『属性D』と表現した方が良いんじゃないでしょうか?」
👨「確かに、そのほうがより正確ですし、意図が明確になりますね。」
👩「じゃあ書き換えちゃいますね。」
〜〜〜〜〜
👩「勘定科目CってどのAPIが持ってるんでしたっけ？すぐに思い出せなくて、すみません。」
👨「E-APIですね。あ、いま気付いたんですけど、属性Fの企業の場合はF-APIですね。
つまり、属性Eの企業はE-API、属性Fの企業はF-APIです。」
👩「了解です！テストを分けた方が良さそうですね。」
👨「他に考慮すべき属性がないか不安なので、一緒に確認してもらってもいいですか？」

ペアプロでテストを書く

対話が始まります

```
## 権限Aを持つユーザーは、企業の勘定科目Cを見ることができる
```

👩「『権限A』は限定的な表現ですけど、ここでは例えば権限Bでも表示して良いと思うので、
それらをまとめた『属性D』と表現した方が良いんじゃないでしょうか?」
👨「確かに、そのほうがより正確ですし、意図が明確になりますね。」
👩「じゃあ書き換えちゃいますね。」

質問や確認による気付き

～～～～～

👩「勘定科目Cってどの API が持ってるんでしたっけ？すぐに思い出せなくて、すみません。」
👨「E-APIですね。あ、いま気付いたんですけど、属性Fの企業の場合はF-APIですね。
つまり、属性Eの企業はE-API、属性Fの企業はF-APIです。」
👩「了解です！テストを分けた方が良さそうですね。」

問題の各個撃破

👨「他に考慮すべき属性がないか不安なので、一緒に確認してもらってもいいですか？」

ペアプロでテストを書く

対話が始まります

権限Aを持つユーザーは、企業の勘

・言語化

👩「『権限A』は限定的な表現ですけ　　して良いと思うので、
それらをまとめた『属性D』と表現　　？」
👨「確かに、そのほうがより正確です
👩「じゃあ書き換えちゃいますね。」
～～～～～
👩「勘定科目Cってどの API が持ってるんでしたっけ？すぐに思い出せなくて、すみません。」
👨「E-APIですね。あ、いま気付いたんですけど、属性Fの企業の場合はF-APIですね。
つまり、属性Eの企業はE-API、属
👩「了解です！テストを分けた方が良
👨「他に考慮すべき属性がないか不安なので、一緒に確認してもらってもいいですか？」

・相互フィードバック
・即時フィードバック
・問題の細分化
・学びや気付き

・早期解決

ペアプロでの心がけ

こんなことありませんか？　その2

容赦ないマサカリ(という名のマウント)

「機能Aを変更するってことは機能Bにも影響あるでしょ？(圧)」
「フォーム入力のテストでなんでバリデーション考えてないの？(圧)」
「同値分割したのはいいけど、無効同値のパターンはないの？(圧)」

原因と思われるもの

(マサカリを投げる瞬間は)(意識的でなくとも)相手を下に見ている

自分の思考に基づく判断・行動 ＞ 相手の思考に基づく判断・行動

事実として正しいことを言えば十分だと思っている

客観的な事実 → 推敲なしに言葉にしてしまう

ただペアを組めば良いわけじゃない

ペアプロでテストを書く

さきほどの対話

```
## 権限Aを持つユーザーは、企業の勘定科目Cを見ることができる
```

👩「『権限A』は限定的な表現ですけど、ここでは例えば権限Bでも表示して良いと思うので、
それらをまとめた『属性D』と表現した方が良いんじゃないでしょうか?」
👨「確かに、そのほうがより正確ですし、意図が明確になりますね。」
👩「じゃあ書き換えちゃいますね。」
〜〜〜〜〜
👩「勘定科目CってどのAPIが持ってるんでしたっけ？すぐに思い出せなくて、すみません。」
👨「E-APIですね。あ、いま気付いたんですけど、属性Fの企業の場合はF-APIですね。
つまり、属性Eの企業はE-API、属性Fの企業はF-APIです。」
👩「了解です！テストを分けた方が良さそうですね。」
👨「他に考慮すべき属性がないか不安なので、一緒に確認してもらってもいいですか？」

ペアプロでテストを書く

さきほどの対話

```
## 権限Aを持つユーザーは、企業の勘定科目Cを見ることができる
```

👩「『権限A』は限定的な表現ですけど、ここでは例えば権限Bでも表示して良いと思うので、
それらをまとめた『属性D』と表現した方が良いんじゃないでしょうか?」

👨「確かに、そのほうがより正確ですし、意図が明確になりますね。」

👩「じゃあ書き換えちゃいますね。」

〜〜〜〜〜

👩「勘定科目CってどのAPIが持ってるんでしたっけ？すぐに思い出せなくて、すみません。」

👨「E-APIですね。あ、いま気付いたんですけど、属性Fの企業の場合はF-APIですね。
つまり、属性Eの企業はE-API、属性Fの企業はF-APIです。」

👩「了解です！テストを分けた方が良さそうですね。」

👨「他に考慮すべき属性がないか不安なので、一緒に確認してもらってもいいですか？」

リスペクトをもって
言葉を選ぶ

発言しやすい空気を作る

ペアプロでテストを書く

さきほどの対話

権限Aを持つユーザーは、企業の

・相互リスペクト

・発言のハードル低下

👩「『権限A』は限定的な表現ですけど、ここでは例えば権限Bでも表示して良いと思うので、
それらをまとめた『属性D』と表現した方が良いんじゃないでしょうか?」

👨「確かに、そのほうがより正確です

👩「じゃあ書き換えちゃいますね。」

〜〜〜〜〜

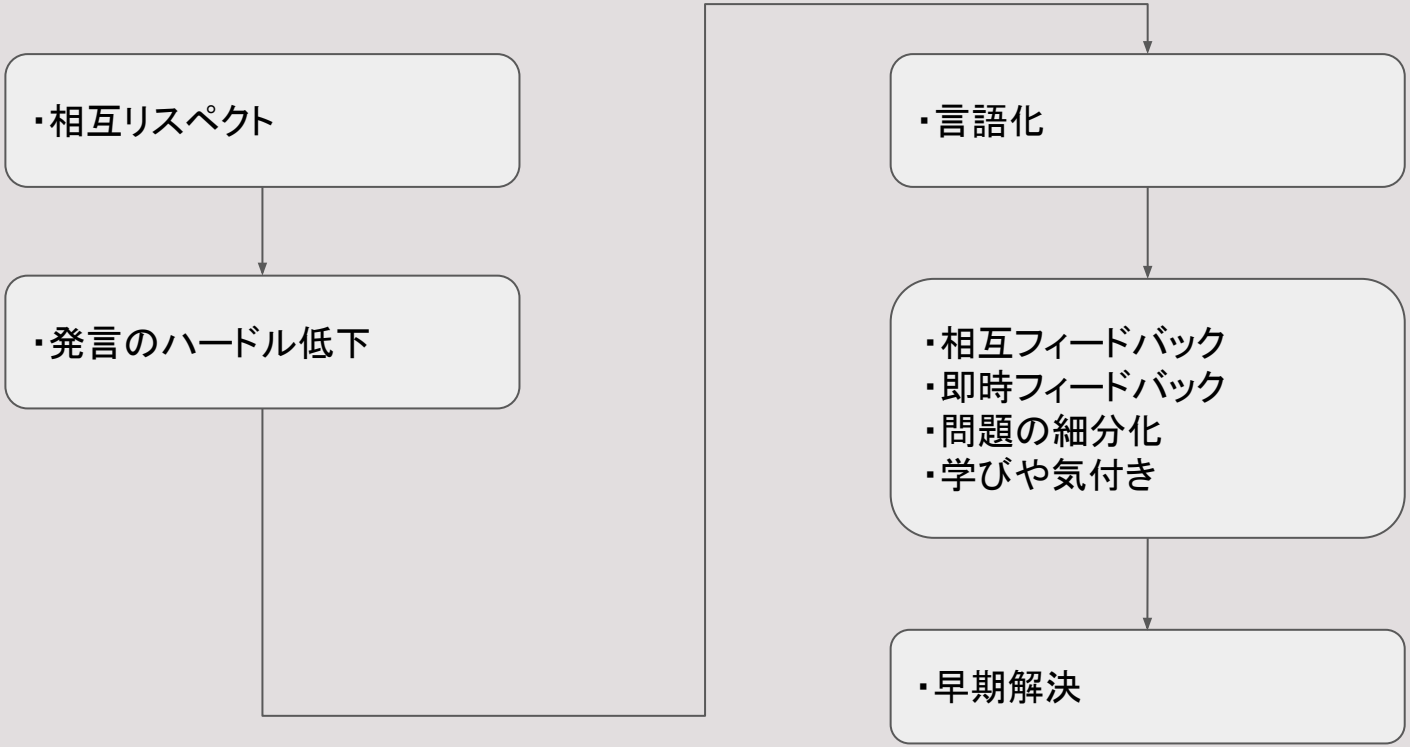
👩「勘定科目CってどのAPIが持ってるんでしたっけ？すぐに思い出せなくて、すみません。」

👨「E-APIですね。あ、いま気付いたんですけど、属性Fの企業の場合はF-APIですね。
つまり、属性Eの企業はE-API、属性Fの企業はF-APIです。」

👩「了解です！テストを分けた方が良さそうですね。」

👨「他に考慮すべき属性がないか不安なので、一緒に確認してもらってもいいですか？」

対話を促すためにも、まずはリスペクトを



おまけ

自然言語でのテストを書ききった状態

```
## 属性Dを持つユーザーは、属性Eを持つ企業の勘定科目Cを見ることができる
* 属性Dを持つユーザーでログインする
* 属性Eを持つ企業を選択し、画面Bを開く
* 勘定科目Cに“1,000”百万円と表示される
```

まとめ

まとめ

ペアプロは、対話を促す仕組み

言語化することで、相互に気付きや学びを得ることができる

相互にリスペクトを心がける