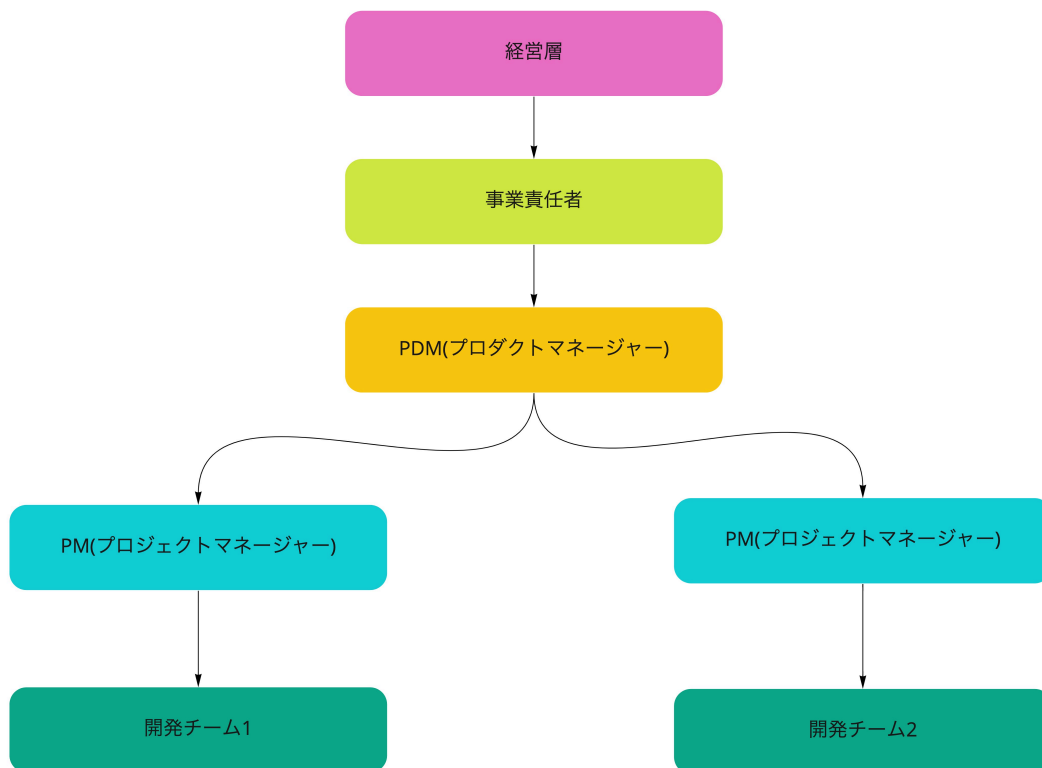


# プロダクト開発の進め方 (S○ORES編)

現職側の組織構成図



## 各組織の役割

### 経営層

プロダクト個別ではなく、会社としてどういった価値を提供していきたいかの方向性を決めている

### 事業責任者

プロダクトの運営そのものに責任を持っている(あまり良く分かっていない)

## **PDM(プロダクトマネージャー)**

どんな施策をどういったスパンで打つか、開発の優先順位決め、どの領域に攻め込むか 等々を考える プロダクト運営の一番のブレイン 現職ではPMと兼任したりもする

## **PM(プロジェクトマネージャー)**

各プロジェクトを進めるための責任者 PM1人につき、開発チームが1チーム付く  
PM, 開発チームのセット分だけ同時にプロジェクトを走らせることが出来る

## **開発チーム**

プロジェクトの遂行に必要な機能開発を行うチーム 複数チームあるが、余程のこと  
がない限り兼任することはない  
1チーム当たり、7~8人くらいがMAX

## **施策開発の進み方**

### **プロダクト開発方針の言語化**

例えばこんな感じのやつ

これから自分たちのプロダクトをどういった方向性に強化していくのかのをしっかりと言語化する

これがないと各Prj単位でバラバラな方向を向いた動きをしてしまい、アプリとしても組織としても迷走することになる

(たぶん今のアングラズはこの状態 一つの組織として同じ方向を向いているのではなく、各Prj単位で別々に俺たちが考える最高のアングラズ像に突っ走っている)

アングラズに対する個人的な感想として、Prjごとの優先順位が見え辛いなと感じる

各Prjのリーダー全員が自分のところが最優先だと思ってそうに思える

それで数少ないエンジニアをPrjごとに取り合ったり、強引に同時並行で走らせて開発スピードを落としている

開発スピードが落ちる == アプリの進化の速度が落ちる なので結果的に競争力を失って会社としても悲しい結果になってしまう

なので優先順位付けが大事

#### プロダクト開発方針

- 釣果がないと利用できない インフルエンサーならInstagramでいいアプリから脱却するため、釣りトークは最優先で対応
- タックルボックスは釣具を効率よく探したいユーザーを得るために有効なため、2023年度中に提供を目指す



社長(PdMを兼任している場合) or PdM(Prjリーダーの上の存在)がしっかりと決めて組織で同じ方向を向けるように音頭を取るべき

#### 数年先を見据えたリリース計画

ここは理想論でいい

が、口だけで数年後はこうなりたいと言うのではなく、実現可能性を感じさせるロードマップは敷いておくべき



これを月1単位等で実際の開発の進捗具合等と照らし合わせながらメンテナンスしていく

ここまでが、アングラーズというプロダクトをどうしていきたいか そこに向かって  
 こういったスパンで何を開発していくかの未来予想図を立てる部分

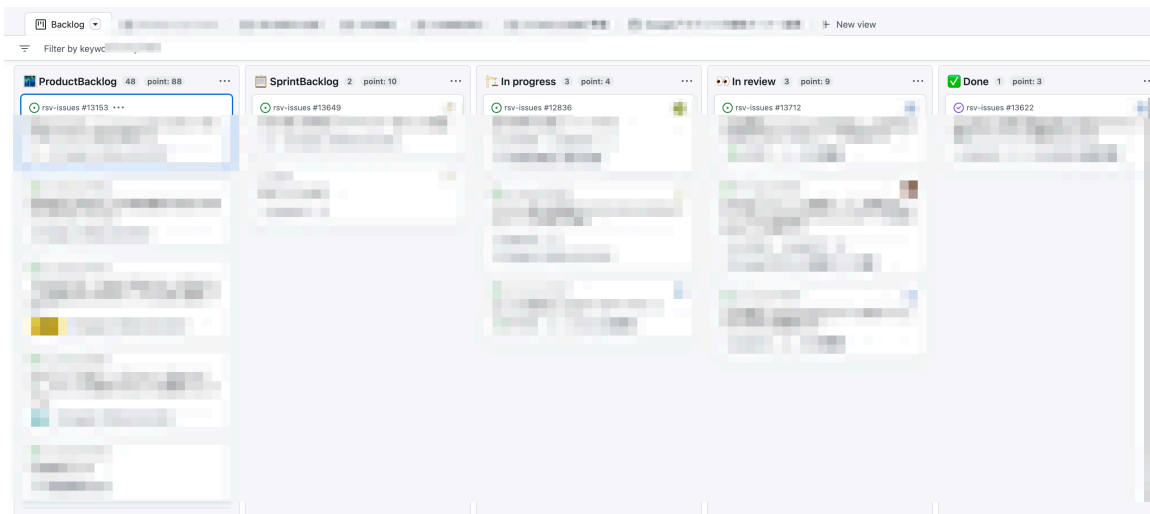
## 開発の進め方

この機能(細かい仕様はまだ未調整)を作り出すというのは決まった前提

- デザイナーに導線を含めたモックを作ってもらう
- モックを見ながらPM, デザイナー, エンジニアで細かい仕様を詰める
  - エンジニアからは今のシステムの現状を踏まえた上でこれは厳しい、これだと実装が複雑だから代わりにこうするのはどうか...etc を提案しつつ調整
  - PMはリリースしたい時期と照らし合わせながら判断
    - リリースはなるべく早くしたいからここは妥協しよう、ここは妥協したくないからリリース遅れるの承知で頑張ってもらおう...etc
  - アングラーズは特にここが不十分

- なので後からやっぱこれも、ここはどうするんだっけ? Biz側に聞かなくちゃが多くなる
  - Bizがエンジニアを巻き込む姿勢、エンジニアが介入しにくい姿勢 どちらも足りていない
    - Bizは開発をお願いしたら終わりではない。エンジニアに思った通りに作ってもらえないと自分等の目的が達成できないはず
    - エンジニアは後戻りが多くて開発工数が膨らんでしまったり、リリース後に不具合多くてBizから疎まれたり運用負荷が上がってしまう
  - 見積もり
    - ここでやっと思見積もり
    - 各タスクが1~2日以内で終わるような分量になるまでタスクを細分化する
- レコメンドフィード機能で例えた場合、以下のようにタスク分割が行える
- サーバーサイド(Rails)
    - レコメンドを登録するAPIの実装
    - レコメンドを取得するAPIの実装
    - 管理画面の実装
      - レコメンドタイトル作成画面の実装
      - レコメンド期間設定画面の実装
      - レコメンド投入画面の実装
      - 釣果検索画面の改修
        - マイスターの釣果のみを検索できるようにする
  - クライアント(RN)
    - レコメンド表示用コンポーネントの作成
    - レコメンド取得処理の実装

これらは全部issueとして切り、終了し次第、Closeしていく  
管理用の看板を別途作成しておくとし進捗がみえやすい



- 細分化したタスクに対してそれぞれ実装工数を見積もる
  - この場合、人月単位で見積もるのではなく、ポイントで見積もるのが好ましい
    - 参考: <https://www.atlassian.com/ja/agile/project-management/estimation>
    - そのためには普段から自分達が特定期間(1週間単位等)でどれだけの開発が行えるのかを計測しておく必要がある
- 合計の工数を基に内部的な仮のリリース日とする
  - ここでの見積もりは基本的にはズレるものと認識しておく必要がある
    - ここでの見積もりを元にユーザーに対してリリース日を告知するのはバカ
    - 実際に作り出さないと見えてこない部分が多い
      - 簡単にできそうだったけど、既存機能との整合性を担保するのに想像以上に苦戦する など
    - これ以降は、見積もり合計ポイントに対する進捗がどれくらいかを見ながら開発完了目処を調整する
    - 参考: 不確実性コーンの話  
<https://blog.takuros.net/entry/2017/05/18/121145>



プロジェクトの初期段階では不確定要素が多いので見積もりには4倍くらいの誤差があり、プロジェクトの進行とともに不確実な部分が少なくなってきて誤差は収束に向かうという話です。

- 最近いい記事が出てたので読んでみることをおすすめします
  - <https://logmi.jp/tech/articles/327563>
- 開発がある程度進んで目処が立ったタイミングでリリース日を本決めする
  - 別に請負開発ではないので、完全に出来上がってからの告知でも問題ないはず
  - とはいえだらだらいくらでも後ろにずらせる姿勢もよくないので、開発終盤ではリリース日をここと決めてしまったほうがいい
- ダメだったらズラせば済む

## 質疑応答編

半分くらいは↑読んで で終わりそうな内容なので、触れて無さそうなものに対して個別にPickしながら答えていく

- こちら、クレームになってる部分は、時間かかっても対応できている認識です
  - クレームに対して全部対応するのではなく、それに対応することが本当にアプリにとって大事なのかを判断してから対応するべき
  - クレームを出した本人しか幸せにならないものは対応するべきではない
- エンジニアは何が企画されているのかわからない...
  - これは組織の問題
  - プロダクト開発方針や、開発ロードマップを敷かずに行き当たりばったりで進めているからわからなくなる
    - たぶんエンジニアだけでなく、Bizサイドのメンバーも自分達のPrj以外のことは把握できてなさそうに思える
- そもそもBiz側が進めている企画を全部オンタイムでできるリソースがあるのかどうか...

- ないなら調整すべき 以上！
- タスクの流れを各Prj→エンジニアチームではなく。Prjのチームにエンジニアも含まれて、その単位で見積もりをするべきである
  - おそらく今のアングラズの開発人員を顧みると、同時に複数Prjを走らせる余裕はないはず
    - なので、今タックルボックス、船宿、釣りトーク、SEOと何個か走らせているが、既に破綻している(本来よりも開発速度が落ちた状態で開発が進んでいる)認識を持った方がいいのかと
- 納期が短い状態で仕事が割り振られることが多い。
  - そもそも他社ステークホルダーがいない自社の開発において、納期を決めてから開発が始まることがおかしい
    - ロードマップ上このくらいでリリースしたい といったラインを決めるが、実際のリリーススケジュールは開発を初めてから現実的に実現可能なものを定めるべき
- 結構Bizチームも検証に関わらないと、バグが多くて、リリースが延びるということが発生してそうです
  - むしろ加わってなかったんですか???
  - お願いしたものが想定通りに出来上がっているか 誰が確認するのでしょうか？
    - エンジニア内で確実に想定通りに出来上がっている保証が持てない限り、エンジニアからも動作確認をお願いすべきなので、次からはBizチームによるQAフェーズも初めから工数に加えておくのがいいと思います。
- 実装作業スケジュールにバッファはどのくらい設けていますか？
  - これを見積もるためには、常日頃から実際の開発タスク以外にどれくらいのタスク(差し込みタスクと呼ばれたりする)が発生するのかを計測する必要がある
  - それを基に、見積もりの合計ポイントに対して普段発生する平均的な差し込みタスクのポイントを合計するとそれがちょうどバッファ分になります。
  - 参考: <https://www.ryuzee.com/faq/0083/>
- ANGLERSでは特定の機能を開発したあとにすぐ次の開発へと向かい、機能の改善やグロースがされていない気がします。メジャーアップデートが多いとエンジ



ニアも気持ち的に消耗しないか心配です。

- まずは開発した機能に対してKPIを定めて計測することをBiz側の動きとして根ざさせることが重要
- 機能をスリム化していく動きも大事だと思います。pjを終わらせる、機能の削除はどの様に検討されてますか？また削除にもエンジニアの工数が必要だと思いますが、どの程度のリソース配分されてますか？
  - 機能のスリム化に関しては、明らかに足を引っ張っているものでない限りは放置してしまっていたりする
    - 周辺領域に対して追加開発を加える際に、あれこれいらなくね？ほんとに使われてる？と思ったものに対して都度、削除判断を行っていたりする
  - 定常的に行えなくても、どこかのタイミングで一度、一気に棚卸しする期間を設けるのはアリだと思う
- UIデザイナーがHTML・CSSを書いて実装し、エンジニアさんに連携する手法はもうレガシーだと思いますか？
  - レガシーではないと思う
    - が、最近ではSPA化してたりで、単純にHTML, CSSを組んで終わることが少ないので、storybookを用意して、そこに対して追加してもらうとかのほうが良いかもしれない
  - Figmaでデザイン組んでもらえれば、ある程度は吸い出したCSSで実装できるので、それだけでありがたい
- 急に納期の短い重い仕事が割り振られるからプライベートの予定が崩壊する
  - 無理です と断った上でプライベートを優先してください！ 以上！
  - そもそもプロダクト開発における納期はBiz/Eng間で合意が取れたものを設定すべきであり、一方的に設定されるものではありません。
    - 唯一の例外は、利用ユーザーに対してビジネス的な機会損失を発生させてしまうタイプの不具合が発生している時 です。
      - このパターンの納期は なるはや です。