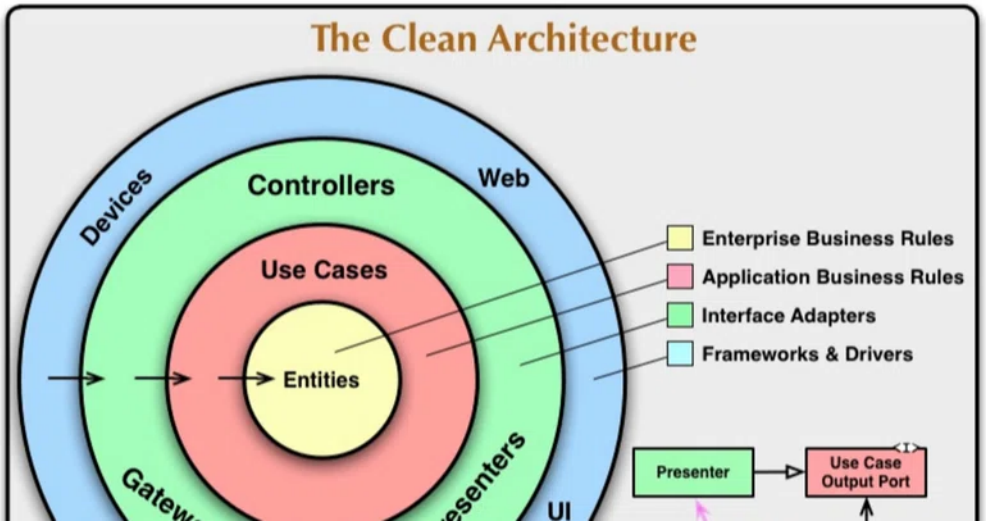

さよならアーキテクチャ議論

Seiji Takahashi@ベースマキナ
2020年6月29日 08:43

ポエム。

つまり？

予算やチームのリテラシーに合わせて最速で作れて、チーム内で「俺ら高凝集低結合だなー」と思えるなら、アーキテクチャはなんでもいいと思えてきました。

前提

・まだ割と収益が安定してないプロジェクトでの話です。**お金があるなら好きにやりましょう。**Go Bold。

・DDDやクリーンアーキテクチャがダメとは言ってないです。むしろ自分は直近そこまで厳格ではないクリーンアーキテクチャでAPI書いてます。

・以前こういうポスト書くくらいにはアーキテクチャのこと試行錯誤してました。

Goのパッケージ構成の失敗遍歴と現状確認

“Goのパッケージ構成の失敗遍歴と現状確認” is published by timakin.

medium.com

アーキテクチャ導入議論への疲労

以前僕は、**DDDやクリーンアーキテクチャを導入するという話が出ると積極的に顔を出すようにしていました。でも、最近は「導入しましょう」「既に適用してあるのでキャッチアップしてください」**などの議論をするのに少し疲れてしまい、足が重くなったように感じます。**もうおじいちゃんなので体力がないんじゃ……**

疲労感を感じる理由は以下の３つです。

1. 事業成功に占めるアーキテクチャという要素の小ささ
2. チームでの共通認識を作るコスト
3. レイヤー分けという行為そのものへの疑問

アーキテクチャは事業を成功に導くか

導かないと思うんですよ。もちろん、**優れたアーキテクチャがコードを書くときの認知コストを下げ、スピードが上がった結果事業成果に寄与するのは当然**で、それは僕も実感するところです。

なのですが、その状況って**売り上げがまず立つことや、チームのリテラシーや人数がスケールすることが大前提**じゃないですか。しかもその**大前提の要素の方が影響度が大きすぎて、実はどういったアーキテクチャを選択するかは、枝葉末節**なのではないか、と思えてなりません。

売り上げは全てを癒すけど、アーキテクチャは全ては癒してくれないんですよ。

チームにアーキテクチャが浸透するまで

DDD、クリーンアーキテクチャに関しては**実際の各社のコードを見ても、完全にレイヤーの区分が一致していることがほぼない**んですよ。しかも**永遠に正解を議論**している。

その状況下で**チームに複雑度の高いアーキテクチャを採用すると、まず間違いなく共通認識を作るためのコストが発生します。**アーキテクチャの議論を積極的にする人ならまだしも、**「別にそんなの興味なくて、とっとと機能ができれば良い」というスタンスの人がいた場合、ドキュメント作成のコストが跳ね上がります。**

しかも、先ほど書いた通りこれは「終わりのない旅」なので、**みんなでスクラムを組んで正解を模索し続けないと、少数の興味ある人間が開発チーム全体の障害点になりうるわけですね。**なので、**開発者のスタンスが相互に異なる場合、そもそも難易度の高いアーキテクチャって採用すべきではないと思うんです。**

さらに、模範的回答が定めやすいという意味で、「アーキテクチャ」と「フレームワーク」は違うんですよ。よくチームで技術選定時にフレームワークについて過敏に反応する事例がありますが、**実はアーキテクチャの方が採用コストがでかく、慎重になるべき**だと感じてます。

とはいえ、FiNCさんみたいに**レビュー体制を敷けるならアーキテクチャのアップデートとか維持もしていけそう**だし、こういうレビューの仕組みも解の一つかもしれないですね。（これはマイクロサービスの話ですけど）

どこまで細かくコードを分割すべきか

Go言語の事例を挙げますが、アーキテクチャを決める、ということはつまり、以下の流れを辿るということです。

・ざっくり「フラット」にすべきか「レイヤーを分ける」べきかという派閥に分かれる
・その後**レイヤーをさらに理想的な分割方法に持っていくために、DDDやクリーンアーキテクチャが採用候補に上がる**

階層の深さとその分割基準を決める、ということですね。僕は割と細かくレイヤー分割したい派閥で、かつ基準点が欲しいのでDDDとかクリーンアーキテクチャを採用しがちです。

そんな自分ですが、フラットにしたい派閥の人の気持ちをずっと考えておりまして、最近やっと気付いたのですが、この**両派閥間では、コードを読むときの思考の流れが決定的に違う**ようです。

「フラット派」は「ドメインというまとまりでコードを読む」人が多く、
「レイヤー分割派」は「構造体やインターフェースのまとまりでコードを読む」人が多い
ように感じられます。（かなり主観入ってます）

ここで重要になってくるのは、**どっちの読み方が正解だ、という話を僕がしていないことでして、ある種多様性を認めるという意味で、どっちも許容可能にした方がいい**と思うんですよ。**人やチームによって凝集度や結合度の定義って実は変わるんだ**と思ってます。もちろん、それをリテラシーって言葉で片付けてしまうこともあるんですけど、相互に歩み寄ろうと努めるとそういう結論になってしまいました。

だからDDDだのクリーンアーキテクチャだのを採用したところで、**コード認知の観点で多様性を否定することにつながり、書き手読み手の労力が減ってなかったらなんの意味もない**んですよね。

究極こんなの、チームメンバーの大半がどの書き方に手慣れてるか、で決めていいと思うんですよ。
GoだとMVCはマイノリティな印象がありますが、別にMVCでいいですよね、みんなが慣れてるなら。Swift書くときだって、クリーンアーキテクチャに慣れてればそれでいいし、そうでもないならMVVMでいいですよ。最近はこの問題に関して、**現在のスキルセットと目標開発スピードをベースに合議制で決めればオッケー**だと思ってます。

まとめ

色々書きましたが、正直ちょっと寂しいです。どういうコードだったら頭が整理されやすいかなって考えるのは、楽しくもあり苦しくもあり。

でも、**その労力に比べて、別にチームが幸せになっている度合いも、事業の成功確度も、別に上がってないんじゃないかなあと**、少なくとも僕は感じてしまったんですよね。

なので、僕にとっての楽しさの力点が、チーム全体の生産性や事業の伸びに一層移っていくことを願って、僕はアーキテクチャの議論に、やや距離を取ることとします✋ (´･_･`)