When To Abstract

International PHP Conference 2017

Tobias Schlitt (@tobySen) & Kore Nordmann (@koredn)

25th October 2017

passion for software quality

Hi, we're Toby & Kore

Tobias Schlitt
@tobySen

Kore Nordmann
@koredn

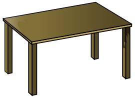
@qafoo

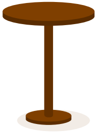
@tideways

You: A Pattern Recognition Machine

Fast Category Learning*

Look:
A table





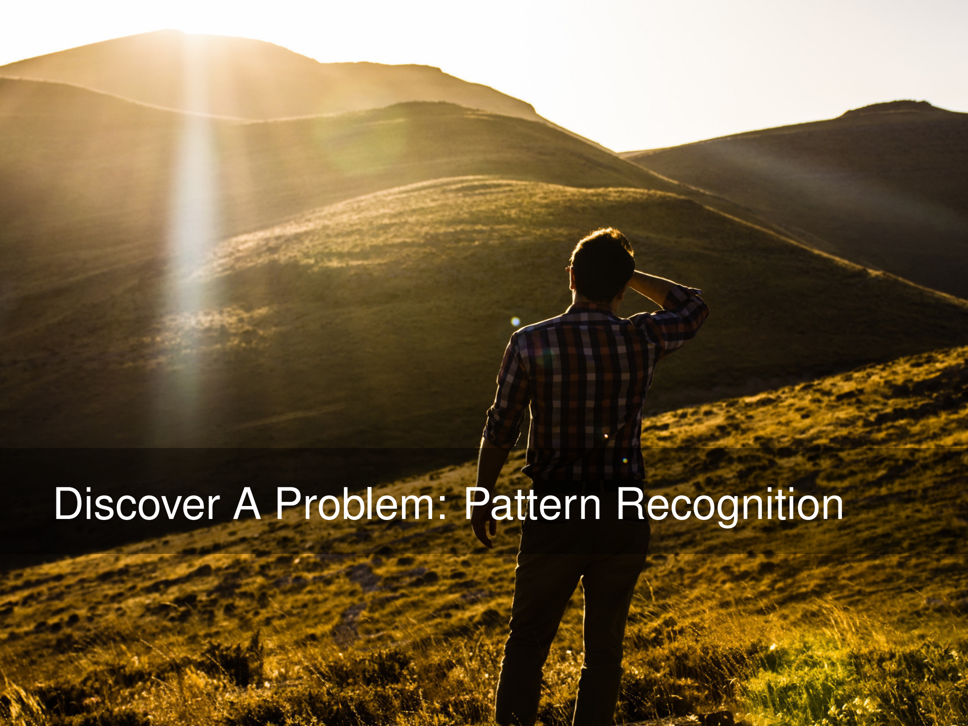
Table?

Table.

Table!

*Psychology of Learning and Motivation (Band 43)
Academic Press, 04.09.2003

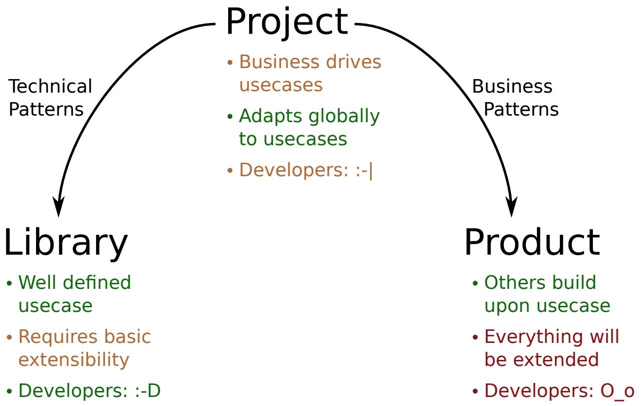
Qafoo
passion for software quality



Discover A Problem: Pattern Recognition

When To Stop?

Types Of Development



Project

- Business drives usecases
- Adapts globally to usecases
- Developers: :-|

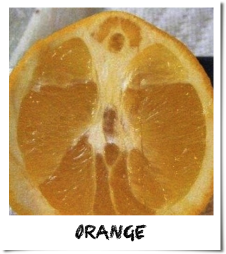
Library

- Well defined usecase
- Requires basic extensibility
- Developers: :-D

Product

- Others build upon usecase
- Everything will be extended
- Developers: O_o

Pareidolia



Discover Business Patterns

Project Development

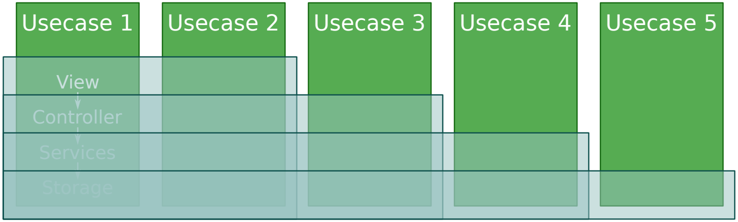
OK – what should we do?

Business Requirements Will Never Converge

Embrace Change

Split vertically instead of horizontally

- How much code can be re-used for that new user registration through Facebook?



Embrace Change

Avoid Pareidolia when creating abstractions

- Never abstract before you have seen the second / third usage of something
- “Duplication is better than the wrong abstraction”[1]

[1] `https://medium.com/@rdsubhas/10-modern-software-engineering-mistakes-bc67fbef4fc8`

Embrace Change

Logic in Value Objects should model **eternal truth**

- There is almost no eternal truth in products
- There might be some eternal truth in libraries
- There is much global truth encoded in your project already

Embrace Change

Facades are good if they **simplify** multiple cases of usage of externals

- Do not implement the next generic query / mapper / . . .

The Joy Of Refactoring

The Joy Of Refactoring

“Legacy Code” already contains all the “stable” business logic

- Let the patterns emerge
- Rewrites would, as always, forget some (40%) of the domain rules
- When refactoring, a developer can do Domain Driven Design on its own

Qafoo
passion for software quality

The Joy Of Refactoring

Refactoring is default behaviour

- Just like testing, you refactor code when it makes sense
- The Domain changes – so does the code

The Joy Of Refactoring

When shall I refactor?

- The more you wait the more you know
- The more technical dept there is, the harder it gets
- Same as testing: Keep the crucial parts clean & tested

Summary

Library	Project	Product
Extensibility +	Extensibility -	Extensibility +++
API-Stability +++	API-Stability -	API-Stability +
Modifiability +	Modifiability +++	Modifiability ++
Optimize for stability with customizations	Optimize for change	Optimize for modfications with stable core

Qafoo
passion for software quality

Summary

Do **not** abstract!*

You are not Symfony

* Except you develop a product or library or it makes sense.

https://qafoo.com/newsletter

Qafoo
passion for software quality

THANK YOU

Rent a quality expert
qafoo.com