Stabile Software in volatilen Umgebungen

eCommerce Camp Jena

Kore Nordmann (@koredn / @qafoo)

March 17, 2017

Hi, I'm Kore (@koredn)

Well Aged Online Shop

Our Project

SproxWare

- Development worked like a charm
- Ressons to change something:
  - New (security) release of shop software
  - New feature requirements
  - New scaling requirements → **Microservices**

After just one change

Reasons For Slow Updates

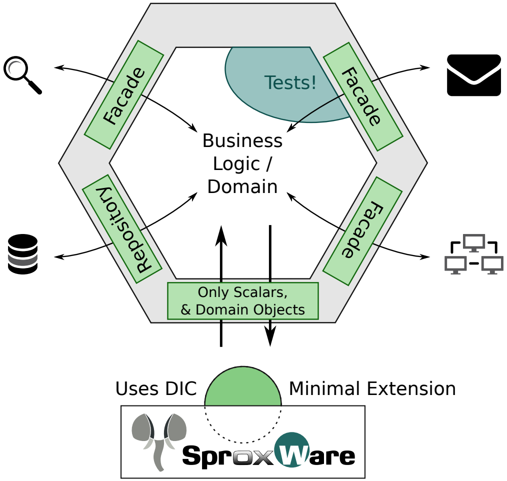
- Changing APIs in vendor software
- Side effects (session, global scope, class scope) between “modules”
- Wrong abstractions – not embracing the next change

Project Development

OK – what can we do?

- We aim for project lifetimes longer then three years
- We aim for fearless releases and upgrades
- We aim for consistently fast bug fixes and feature development

Design Stable Modules / Extensions



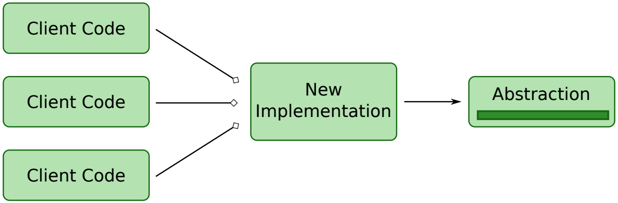
The Entry Point

- May be an ugly mess
- **Must not** contain logic, beside:
  - Basic input conversion
  - Exception handling
  - Basic out preparation
- Access your Dependency Injection Container (Application configuration) statically, if necessary
- Do not test with Unit Tests (unless it is a certification requirement)

The Domain

- Separate between “Newables” (Data Objects) and “Injectables” (Services)
- Only add “Eternal Truth” to Newables – all other logic goes into Injectables
- No Newable may aggregate an Injectable
- No Injectable may aggregate an Newable – only use as parameters

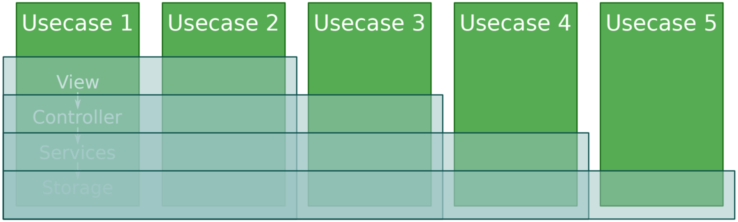
Finding The Domain (Branch By Abstraction)



Do Not Abstract (Embrace Chnage)

Split vertically instead of horizontally

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



Do **Not** Abstract

- **Never** let fellow developers come up with technical abstractions
    - **No** custom Object Relational Mapper
    - **No** custom Request, Routing
    - **No** custom Form Handler
    - **No** custom Configuration Handlers
    - **No** custom Template Systems
    - **No** custom Logger
    - **No** custom . . .
- Use the amazing components which are out there and **tested**

Do Not Abstract

Your developers do not develop SproxWare

Summary

- Extract a framework independent domain
- Test your decoupled domain
- Sensible Domain Driven Design (just) consists of sensible Domain Objects in 99% (no CQRS, no Event Sourcing, ...)
- Embrace change
- Define an Extended Definition Of Done with design rules[1]

[1] `https://qafoo.com/blog/097_extended_definition_of_done.html`

passion for software quality

THANK YOU

Rent a quality expert
qafoo.com