

# Domain Driven Design applied to Frontend

**Milena Suarez**  
2019-11-09



<BoyaConf>

# Milena Suarez

Senior Frontend engineer. I have worked in academic, retail and financial industries with Javascript, Typescript, Angular.

I love the console, travels, wine and cats.



@milena\_suarezl



@milenasuarezl





<BoyaConf>

# Agenda

1. What is DDD?
2. Why DDD?
3. How to do DDD?
4. Strategic Design
5. Tactical Design
6. Case Study
7. Advantages
8. Caveats



<BoyaConf>

# What is Domain driven design?

DDD is a **software development approach**.

*"DDD reminds you of common-sense rules, and it offers a set of ideas, principles, and patterns **to make it happen.**"*

Vaughn Vernon

Domain-Driven Design Tackling Complexity  
in the Heart of Software. 2003. Eric Evans



<BoyaConf>

# Why DDD?

*"The economy of no design is a fallacy"*

*Vaughn Vernon*

- Effective design
- Improve your **craft**
- increase your success in projects
- **Model unique** business needs
- Help your business compete at new heights



< BoyaConf >

# How to do DDD?

“Embrace to business people; They’ll enjoy explaining their **jobs and problems**, you’ll love listening and **finding solutions.**”

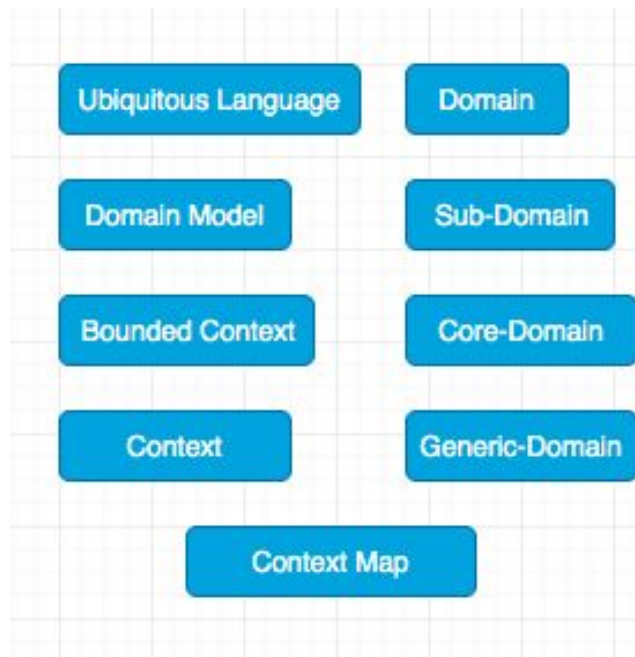
**"In order to go fast, we must go well."**

*khalil stemmler*

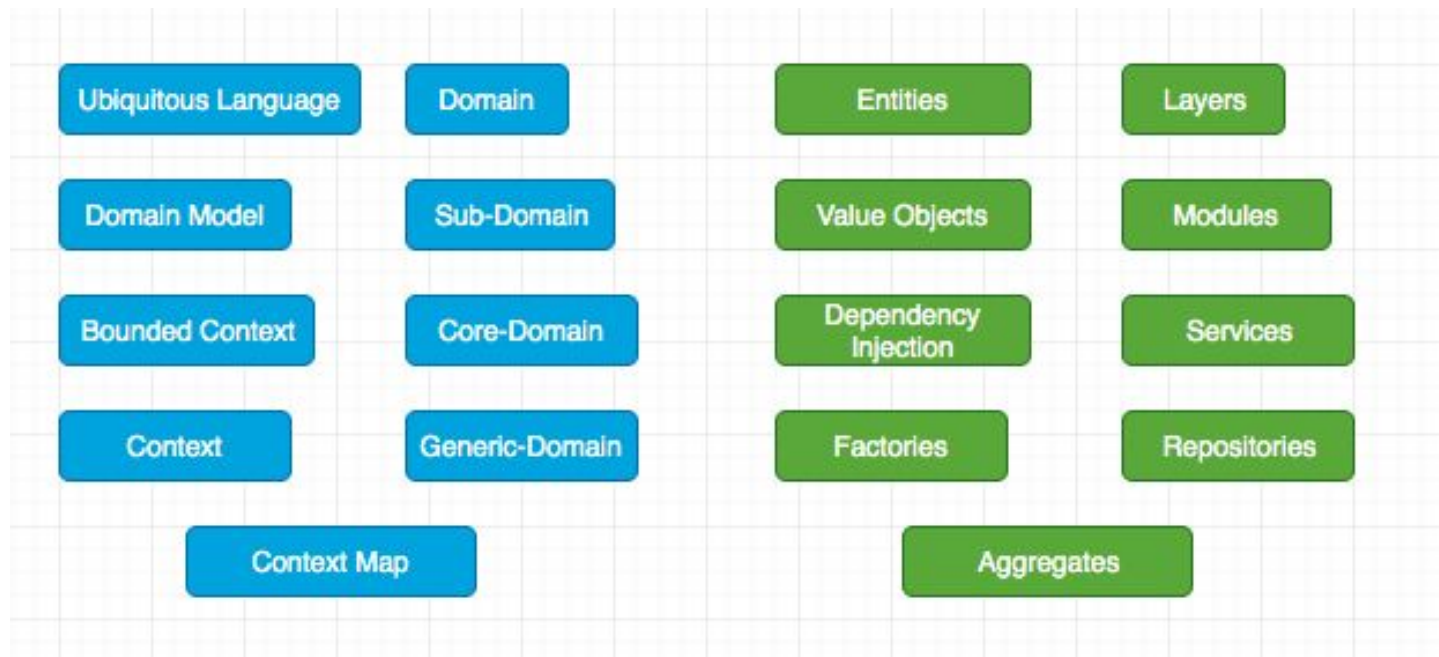


<BoyaConf>

## Strategic Design



## Tactical patterns









< BoyaConf >

# Strategic Design

- Identify the core competencies
- Introduction to new staff
- Ubiquitous language



< BoyaConf >

## Ubiquitous language

“The names should be agreed on with the business.”

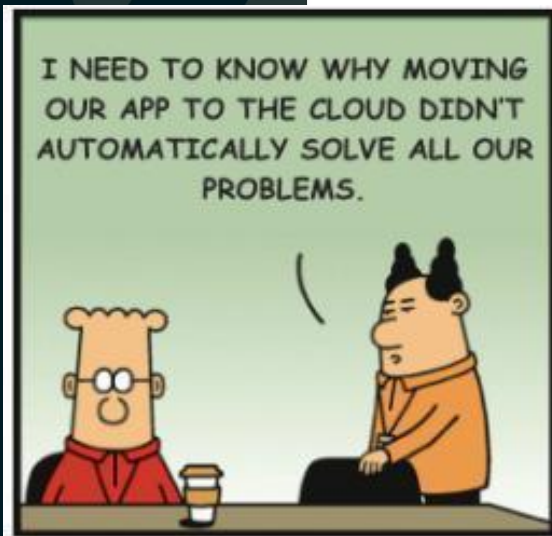
Why? Because in the future, when you talk about a new feature or a bug report the domains expert understand you"

**Think and write meaningful names.**



<BoyaConf>

## Communication between domain experts and developers.



@ScottAdamsSays

Dilbert.com



11-06-17 © 2017 Scott Adams, Inc./Dist. by Andrews McMeel





< BoyaConf >

# Bounded Context

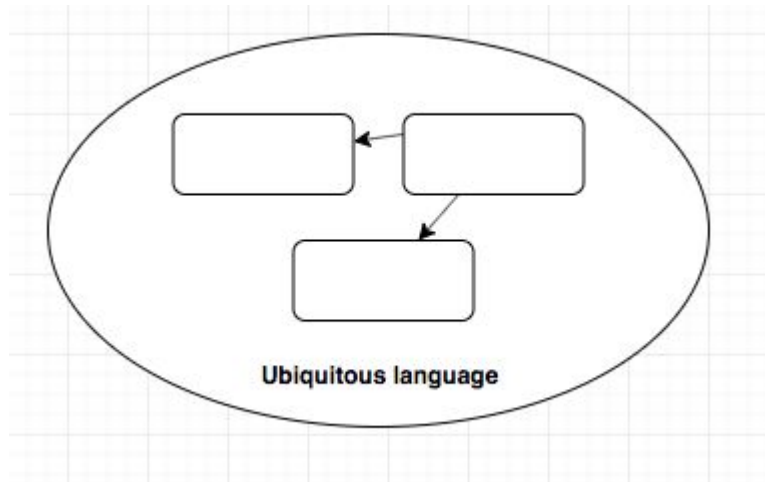
*Define the context for business people.*

## Why is the Core?

### Subdomains

- Core Domain
- Generic Domain

## Context Mapping





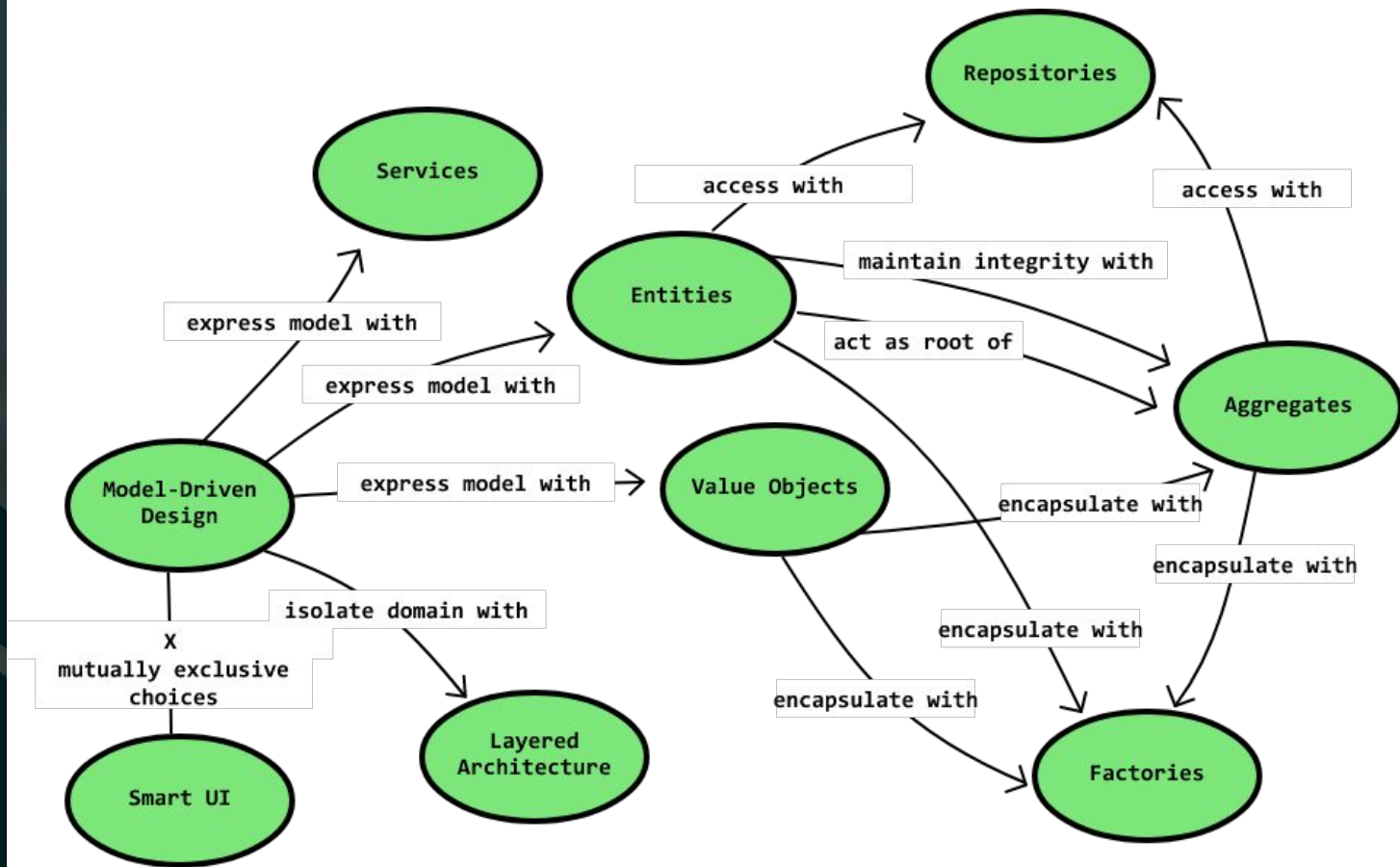
<BoyaConf>

# Tactical design

- Applying the tactical patterns
- Protecting the domain layer
- Anemic Domain Model

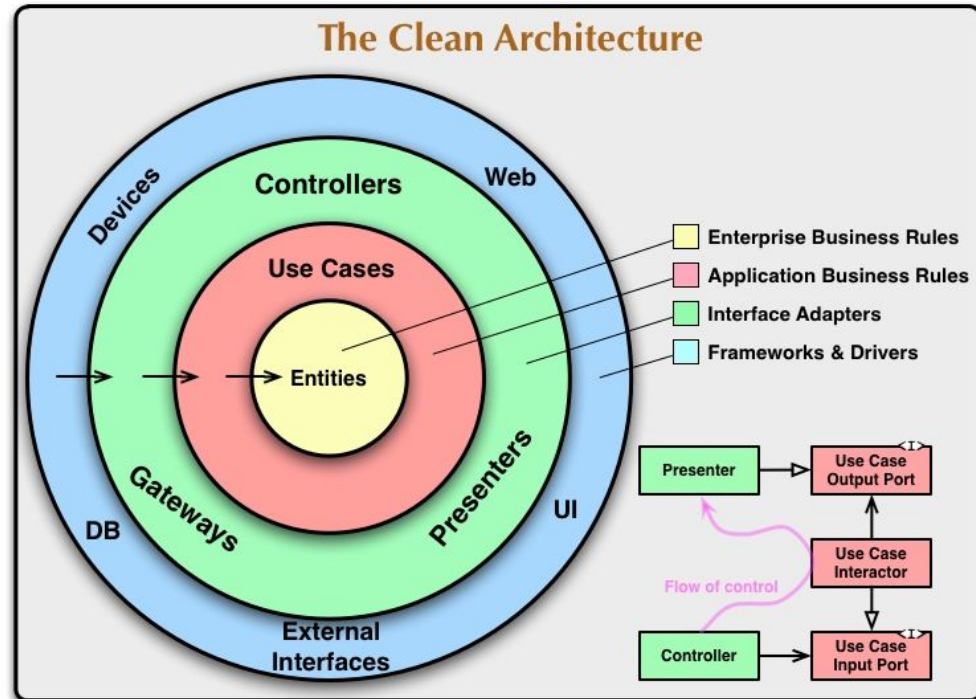


<BoyaConf>



Taken from khalilstemmler.com

# Protecting Domain Layer - Hexagonal architecture





< BoyaConf >

## Anemic Domain Model

Employee
+ identification: string + name: string + hoursWorked: number
+ save(identification) + updated(identification, data)

Employee
+ identification: string + name: string + hoursWorked: number
+ save(identification) + updated(identification, data) + calculatePay(identification) + calculateHoliday(identification)

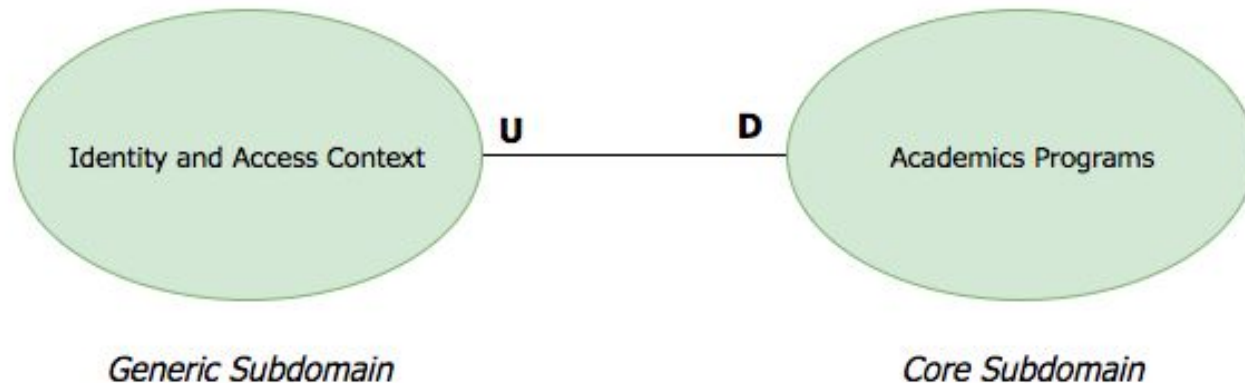






<BoyaConf>

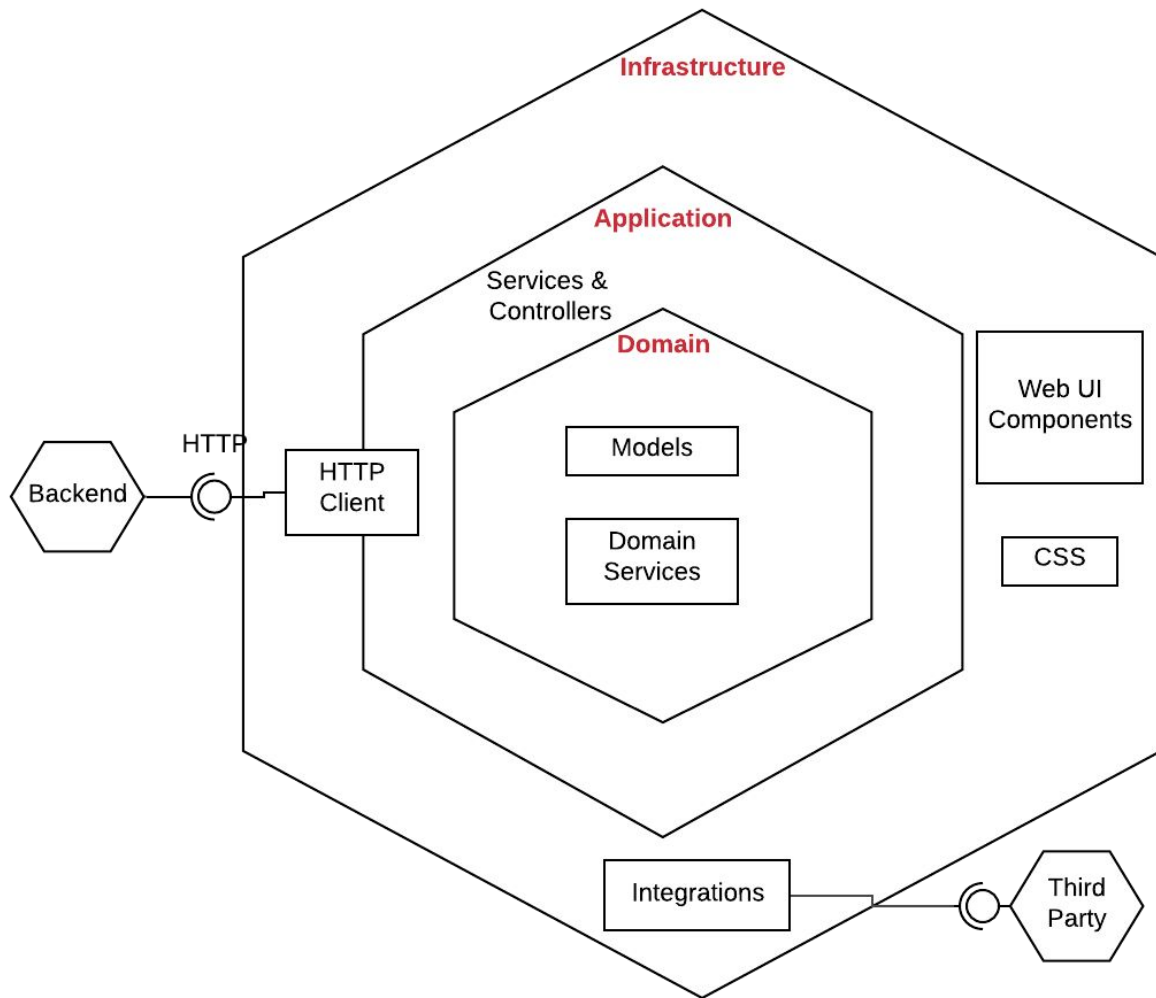
## Case Study

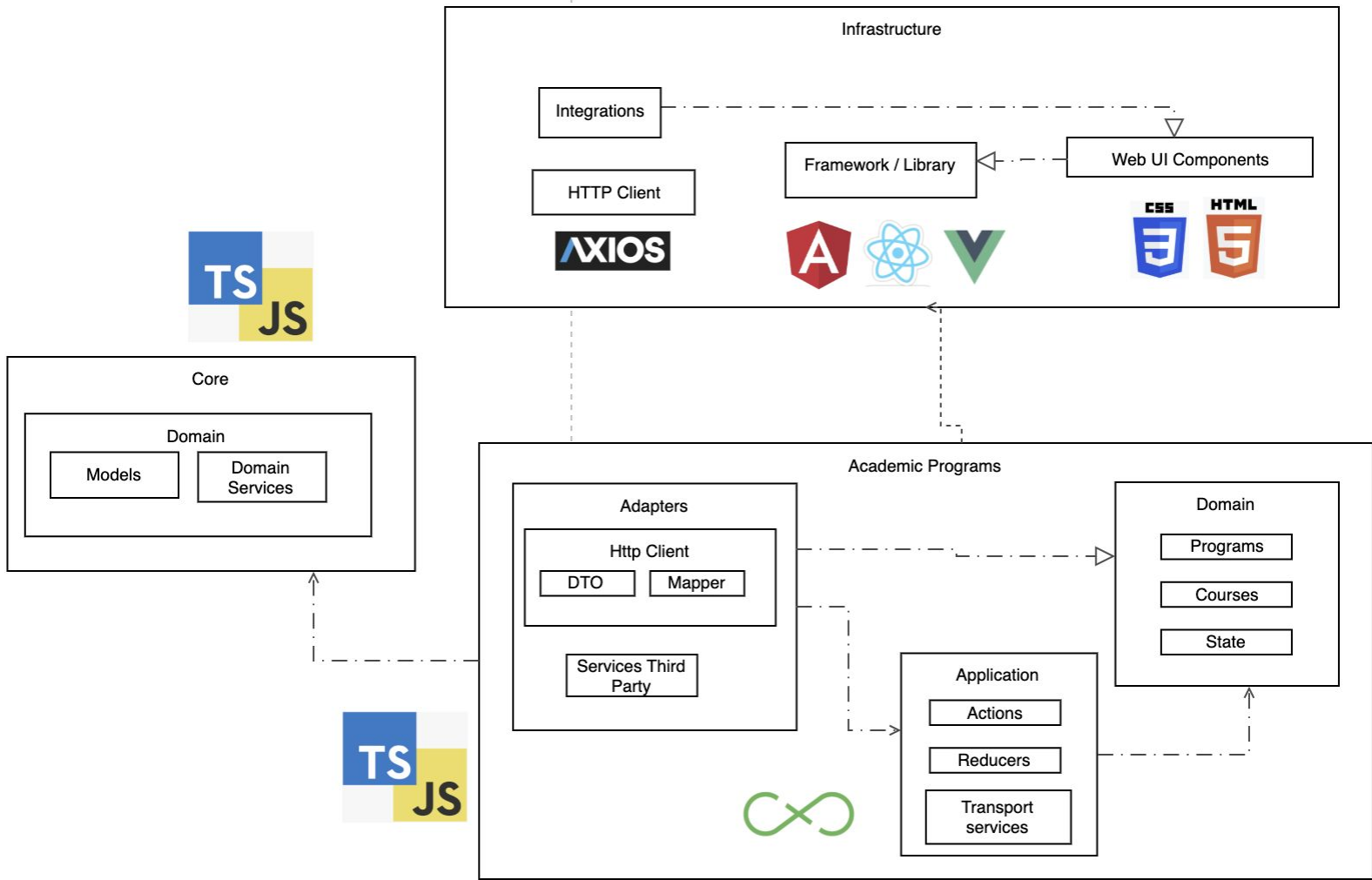


U	Upstream	Service exhibitor
D	DownStream	Service consumer



<BoyaConf>







<BoyaConf>

# Advantages

- A better understanding of business by domains experts and developers
- Business logic **agnostic** to the development tool
- Controlled team **growth**
- Large scale applications a lot more **friendly** and **maintainable**
- Write meaningful unit **tests** that covered **the business rules**





< BoyaConf >

# Caveats

*"The software development is considered a cost center rather than a profit center"*

*Vaughn Vernon*

- Acquired **knowledge** of all members
- It is an **iterative** approach
- Required effort between **team members**



<BoyaConf>

## Conclusion

*Domain Driven Design allows **being better developers** according to needs the business*

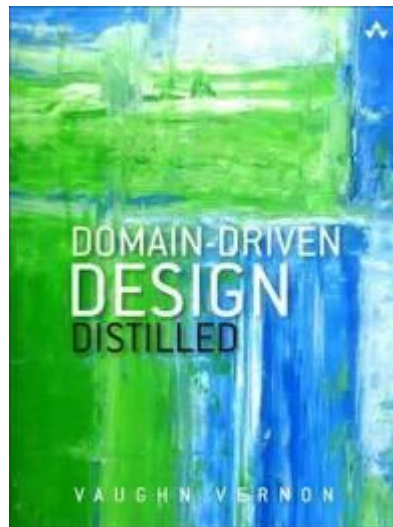
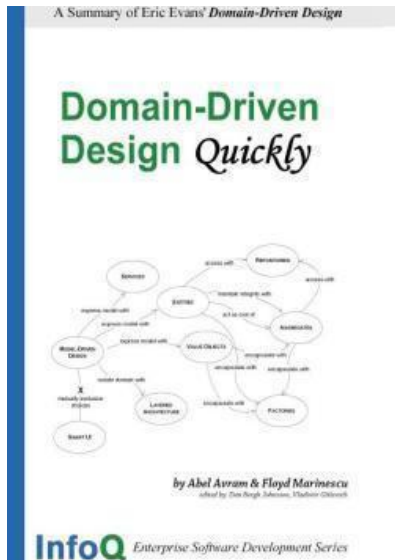
*Grow into the company*

*The frontend developments is not only html and css, also it is really important a effective design before to write code.*



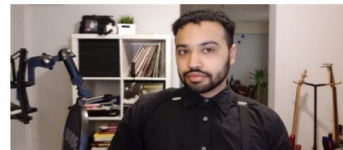
<BoyaConf>

# Interests resources



khalil stemmler

## About



Howdy 🤖

I'm Khalil Stemmler.

I'm a Developer Advocate at [Apollo GraphQL](#).



I'm also a Canadian software developer, writer, and (occasional) musician living in Southern Ontario.



**THANK YOU!**



EVERYDAYIDERP.TUMBLR.COM