Blazor Conf 2022

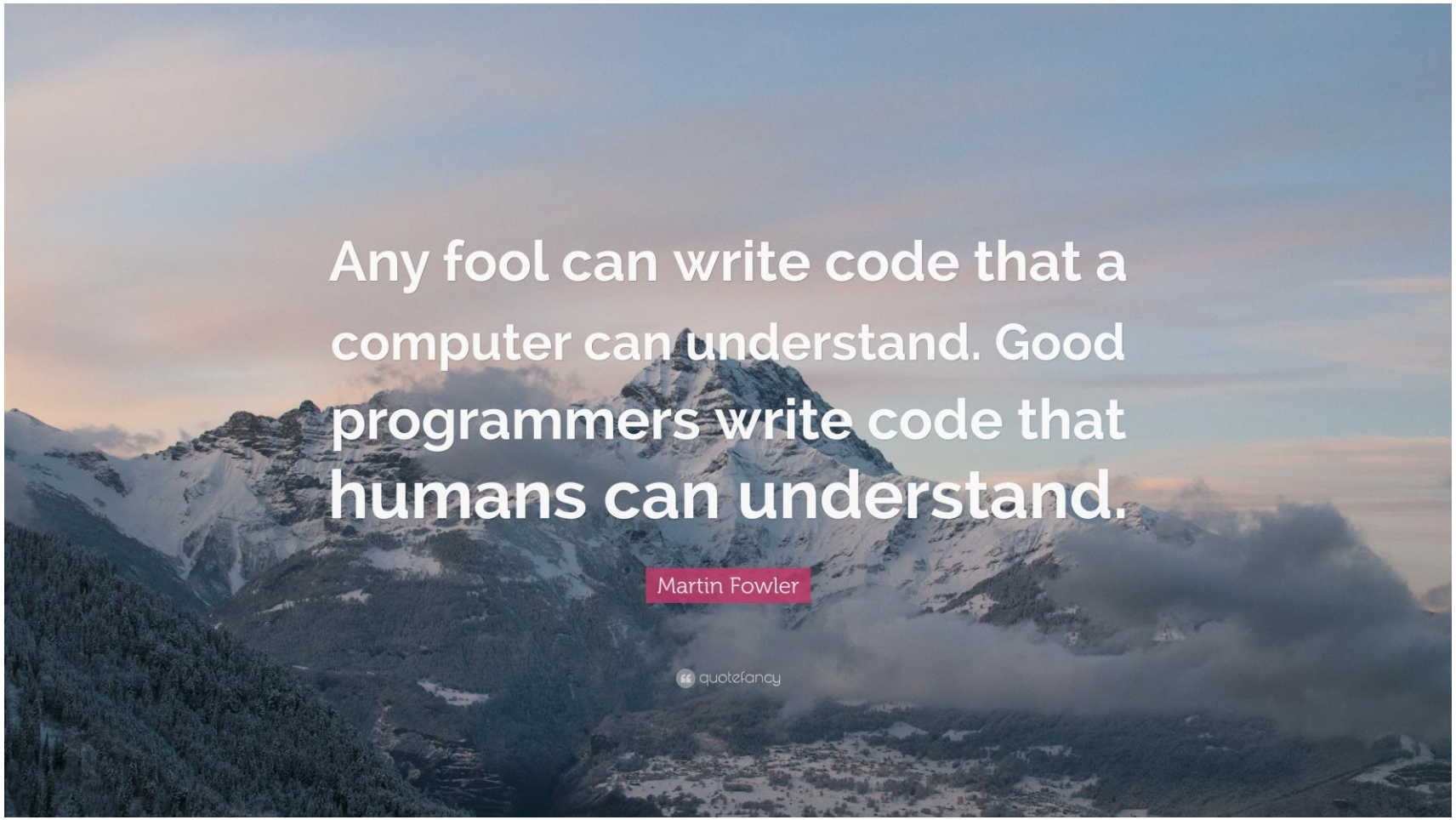
Micro-Frontends

Ovvero: come DDD sbarca nella UI

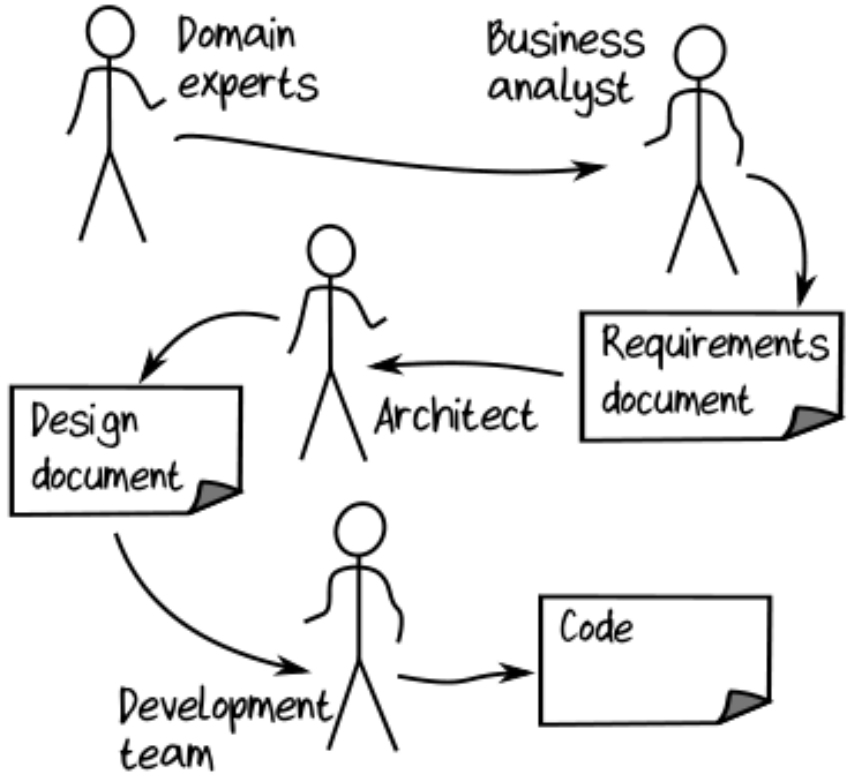
Un grazie agli sponsor

E alle community che ci hanno supportato

Good Software



Like Facebook … but better



We Like to Brew (and Drink) Beer

Suppliers

Production

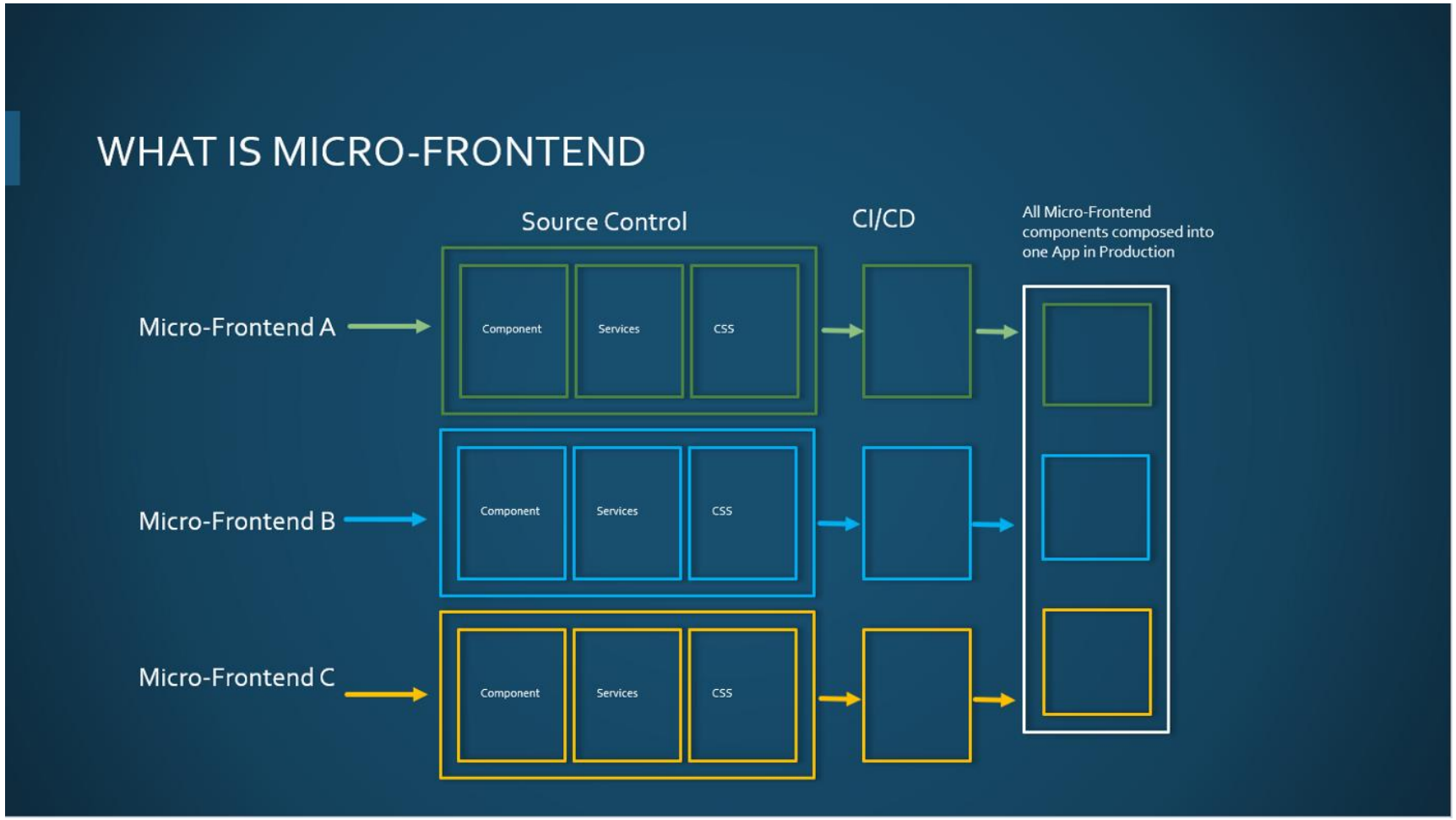
Pubs

Blazor

Blazor Conf 2022

Talk is Cheap … Show me the code

Micro-Frontends



Not a new concept

ThoughtWorks Technology Radar 2016

Self-Contained Systems

- Autonomous Web Application
- Each SCS is owned by One Team
- Asynchronous Communication
- Each SCS has its own API
- Each SCS include Data and Logic
- No Shared UI
- No Shared Business Code
- Shared Infrastructure can be minimized

Decision Framework

- Defining what a micro-frontend is in your architecture
- Composing micro-frontend
- Routing micro-frontend
- Communicating between micro-frontend

Micro-Frontends ways

- Micro-apps with a shared session and parameters
- Routing
- Blazor as a component in an existing project
- Shared components or a Razor class library

Introduction to Micro-frontends with Blazor web assembly (ifourtechnolab.nl)

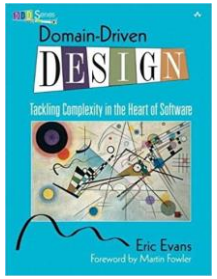
What are Micro-Frontends?

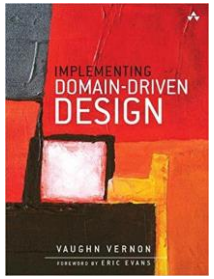
Micro-frontends are the technical representation of a business subdomain, they allow independent implementations with the same, or different, technology choices.

Finally they should avoid sharing logic with other subdomains and they are own by a sigle team.

Let me think

DDD – A bit of story





Tackling complexity in the heart of software

E. Evans – V. Vernon

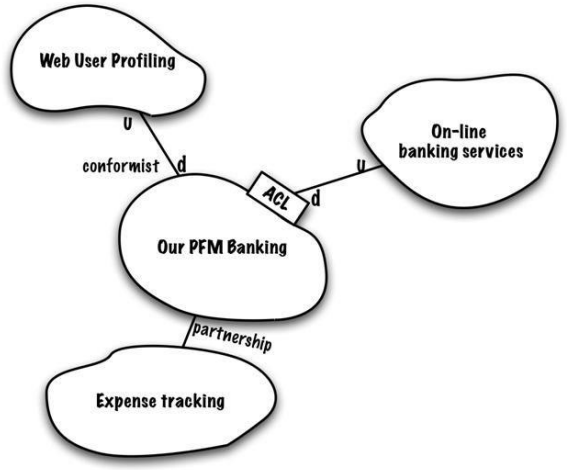
Qualcuno più vicino a noi …

D. Esposito – A. Santarello

Qualcosa dalla community italiana

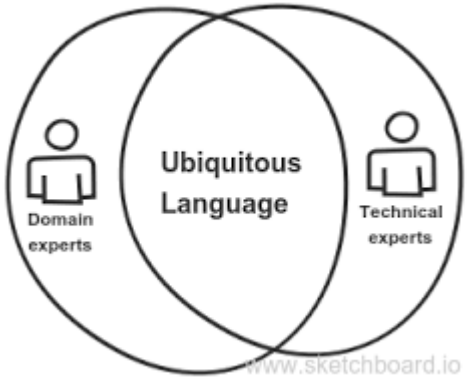
DDD – Strategic Patterns

Context Mapping

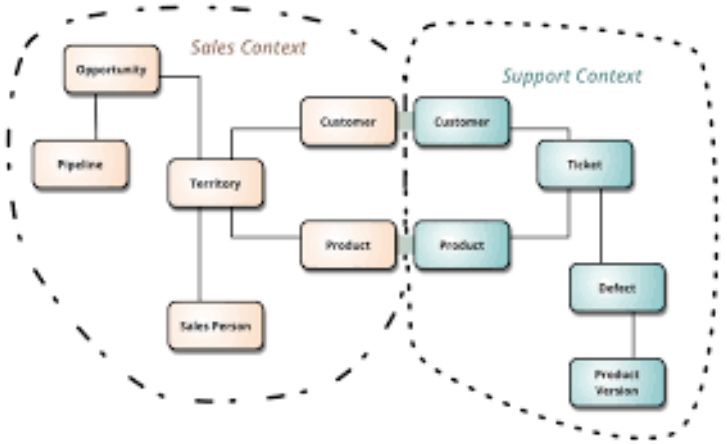


https://www.infoq.com/articles/ddd-contextmapping/

Ubiquitous Language

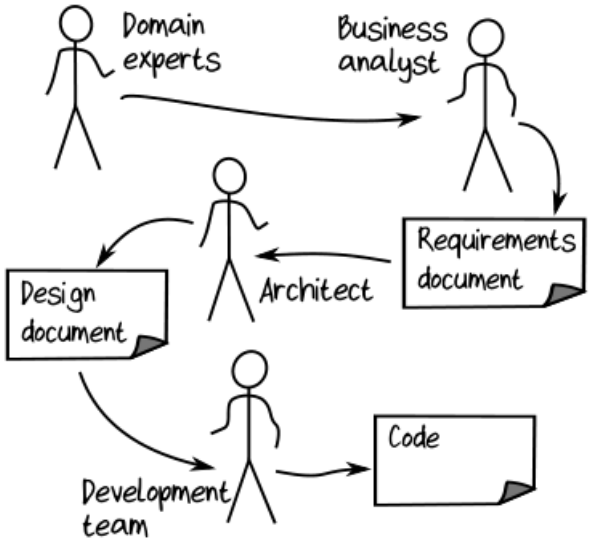


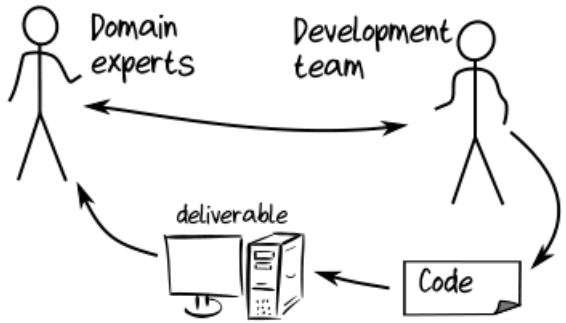
Bounded Context

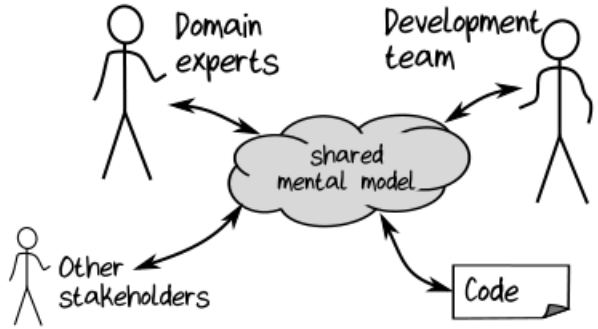


https://martinfowler.com/bliki/BoundedContext.html

DDD – Shared Model







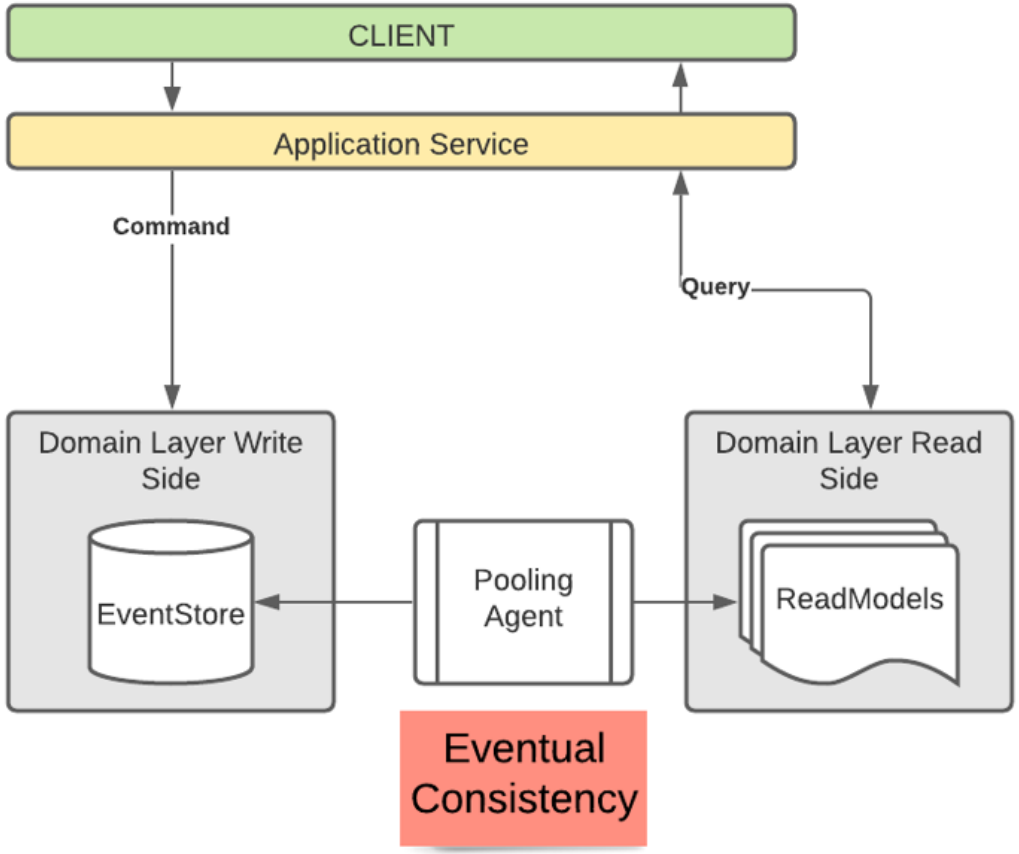
Blazor Conf 2022

Talk is Cheap … Show me the code

Bounded Context – Micro-Services – Micro-Frontends

Feature	Bounded Context	Microservices	Microfrontends
Organized around Business Capabilities	It is implicitly understood in the very concept of Ubiquitous Language, which is the main pattern for identifying a Bounded Context	Cross-Functional Teams specific to a business functionality	Each SCS is owned by One Team
Decentralized Governance	A shared model for each purpose	Local choices, which must be independent, are favored/encouraged.	Autonomous WebApp
Decentralized Data Management	Private persistence is critical for language consistency, but especially necessary for the safe and independent evolution of the model	Each microservice must persist its data in a private database! Otherwise, it will be unable to evolve independently from others	Each SCS has its own API Include Data and Logic
Evolutionary Design	Each model can, and must, evolve independent of the others	Key feature	No Shared UI No Shared Business Code Shared infrastructure can be minimized
Smart endpoints and dumb pipes	Recommended as a strategic model	Key feature. SOA docet!	Asynchronous Communication
Language Consistency	Ubiquitous Language	Key feature	Private Logica and Data

Eventual Consistency



Thank You

- Alberto Acerbis

alberto.acerbis@intre.it

@aacerbis

LinkedIn

https://github.com/brewup

https://github.com/cqrs-muflone