

The bounded context is not enough!

Alberto Acerbis – Alessandro Colla



Disclaimer

The "Bounded context" is needed, but in our opinion is not just a business transactional boundary anymore. We must keep in consideration its evolution and the static coupling with the infrastructure.

github.com/BrewUp/DDD-Explore-2024

All the code you will see today

Awful monolith

Show me the code





Big ball of mud

Lack of a clear architecture, leading to a system that is haphazardly structured and difficult to understand or modify. Tight Coupling / Cyclic dependencies between components

Components are heavily interdependent, making changes in one part of the system likely to break other parts.





Lack of modularity

Failure to properly encapsulate different functionalities, leading to a system where changes in one module ripple through others.

Shared data model

Different modules or functionalities directly accessing and modifying a shared data model, leading to high risk of data corruption and conflicts.





Monolithic database design

A single database for all application needs, creating a bottleneck and complicating any attempt to scale or separate concerns.

What is the impact of changes?



Application-level change scope

(Triangle represents where change occurs)

where estimpromotiheck() {

Net ratingPromptbisabled = mait (meter water

if((ratingPromptDisabled) {
 /s Check if the timer is set or not se
 let ratingPromptTimer = sett Cade, get(ratingPromptTimer = set);
}

tf(!ratingPromptTimer) {
 Gebe.set('ratingPromptTimer', __mumi) { 'MMP(
 belse
 M(ratingPromptTimer SA __mumi) | 'MMP - ratingPromptTimer SA __mumi) | 'MMP - ratingPromptTimer SA __mumi) | 'MMP - ratingPromptTimer =
 this.displayRatingPrompt());
 }
}

Exercise

How can we evolve it?

"Developers are drawn to complexity, like moths to a flame, often with the same outcome" - Neal Ford



Architecture decisions

We must understand the principles/key concepts to make effective decisions

- Service: a cohesive collection of functionality deployed as an independent executable
- Coupling: a change in one service might require a change in an another to maintain proper functionality
- **Component**: building block of the application that does some sort of business or infrastructure function (e.g.: namespace or package)
- Synchronous communication: the caller wait for the response before proceeding
- Asynchronous communication: the caller does not wait for the response
- Orchestrated coordination: it includes a service that coordinate the workflow
- Choreographed coordination: it lacks an orchestrator
- Atomicity: A workflow is atomic if all parts maintain a consistent state at all times (the opposite is eventual consistency)
- **Contract**: the interface between two software parts

Modular (good) monolith

Show me the code

What is the impact of changes?



(Triangle represents where change occurs)

What we are missing next?

What we can add to make modules fully indipendent?



while ratingProsptCheck() { In Check (F the rating prospt is a

het ratingPromptDisabled = anoit Contempor

int(instingPromptDisabled) {
 Is Oneck if the timer is set or not w
 Tet ratingPromptTimer = awiit Code, get(not oppromptTimer =
}

tf(!ratingPromptTimer) {
 Cache.set('ratingPromptTimer', __mmal) | 'mmn;
 letse
 tf(ratingPromptTimer 54 __mmal) | 'mmn = contemporate
 this_displayRatingPrompt();

Exercise

Let's work together with some live coding

Modular monolith with events

Show me the code

Fitness functions

The book «Building Evolutionary Architectures» defined the concept of an architectural fitness function

"any mechanism that performs an objective integrity assessment of some architecture characteristics or combination of architecture characteristics."

Microservices

Show me the code

What is the impact of changes?



Function-level change scope

(Triangle represents where change occurs)

Well...How do I split my monolithic DB?



"Software architecture it's abstract by nature and we must ground it with some implementation details to make it concrete"

Architecture quantum | quanta

An architecture quantum measures several aspects of both topology and behavior in software architecture related to how parts connect and communicate with one another.

It is an independently deployable artifact with high functional cohesion, high static coupling, and synchronous dynamic coupling.

Bounded context is not enough

- Transactional boundary is not enough as a concept.
- We have to keep in consideration static and dynamic coupling of our components (e.g. DB, service bus, event store, etc.).
- We like to see it as «components cohesion» like the concept of functional cohesion

Components Cohesion



Components Cohesion



Last responsibility moment

«Delay decisions as long as you can, but not longer. Maximize the information you have. Minimize technical debt from complexity.»

Postel's law or Robustness principle

"Be conservative in what you do, be liberal in what you accept from others" Only validate what you need

Trade-off everywhere

«Don't try to find the best design in software architecture. Instead, strive for the least worst combination of trade-offs»

Bibliografia

• Implementing Domain-Driven Design (V. Vernon)

https://www.amazon.it/Implementing-Domain-Driven-Design-Vaughn-Vernon-ebook/dp/B00BCLEBN8/

- Introducing Event storming (A.Brandolini)
- REST in Practice: Hypermedia and Systems Architecture (J.Webber)
- Enterprise Integration Patterns (G. Hohpe) https://www.amazon.it/Enterprise-Integration-Patterns-Designing-Deploying/dp/0321200683/
- Learning Domain-Driven Design: Aligning Software Architecture and Business Strategy (Vlad Khononov)

https://www.amazon.com/Learning-Domain-Driven-Design-Aligning-Architecture/dp/1098100131

- Hands-On Domain-Driven Design with .NET Core (A. Zimarev) https://www.amazon.it/Hands-Domain-Driven-Design-NET/dp/1788834097
- Software Architecture: The Hard Parts (N. Ford, M. Richards, P. Sadalage, Z. Dehghani)

https://www.amazon.com/Software-Architecture-Trade-Off-Distributed-Architectures/dp/1492086894









Thank

You

You can find us at

✓ @aacerbis m https://www.linkedin.com/in/aacerbis/ □ alberto.acerbis@intre.it

@collaalessandro
 alessandrocolla
 alessandro.colla@evoluzione.agency



