Building an event sourced system in C#

Hannes Lowette

AXXES

Who am I?

- Father of Arne (11), Joren (7) and Marit (5)
- Partner of Barbara (?)
- Head of L&D @ Axxes
- .NET backend dev
- Loves knowledge sharing
- Amateur guitar builder
- Guitarist @
  Dylan Beattie & the Linebreakers
- Mountain biker
- Bad chess player
- Microsoft MVP

Who are you?

- What's your name?
- What is your background?
- What are you hoping to learn?
- Tell me 1 cool fact about you!

… or if you don't feel like sharing, just say:

I'm X and I'm here to see what happens.

AXXES

Disclaimer (what to expect)

I am here for you, not the other way around

This means:

- If you have questions, ask!
- If you feel we can do things better/differently, speak up!

→ This workshop goes differently every time

AXXES

PSA

1. Need a break? Others probably need one too.
   → Let me know! I tend to ramble on.

2. Learning happens best when you feel comfortable.
   → Eat snacks, drink, go to the bathroom, …

3. I can talk for 2 days, but my voice gets hoarse.
   → Let's turn the coding parts into a conversation!
   → You learn, I learn something too

4. Need me to commit and push?

AXXES

Agenda

1 _ DDD

2 _ Event Sourcing

3 _ CQRS

4 _ What are we building?

5 _ Optimizations

AXXES_

DDD

–

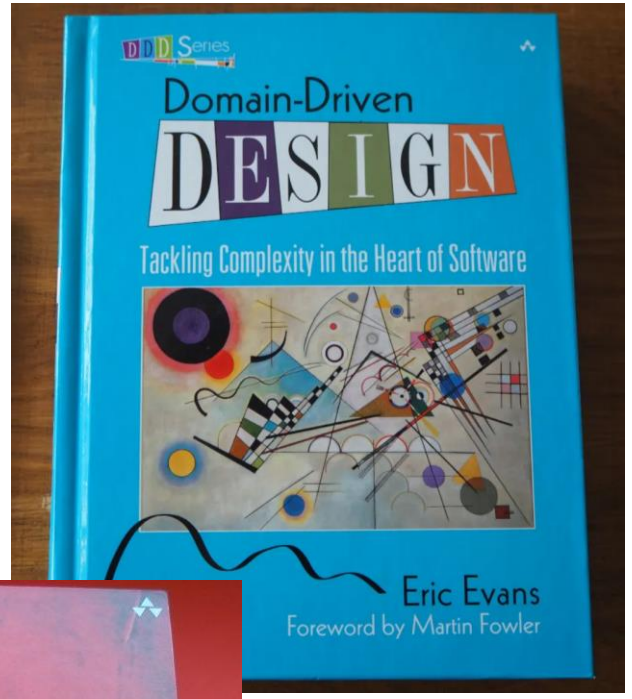
Domain Driven Design in a nutshell

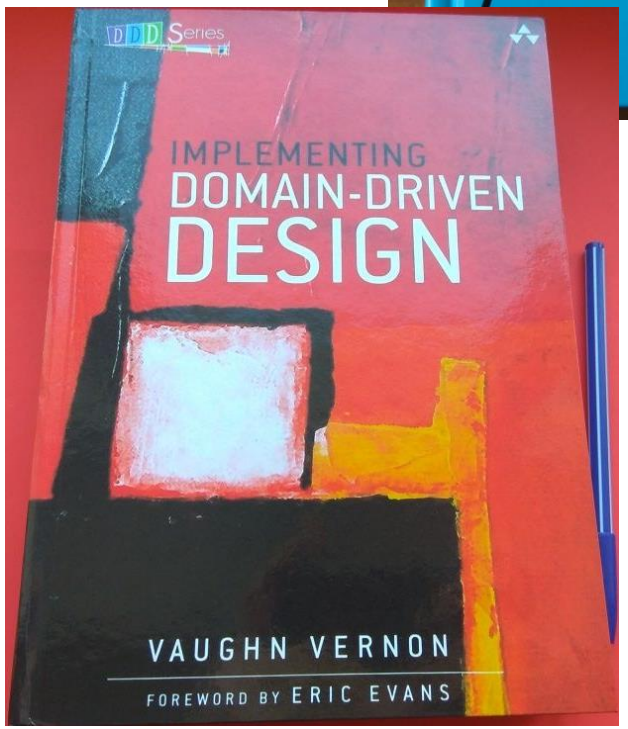
AXXES_

Where it started for me

- Eric Evans = the bible
- Vaugn Vernon = easier to digest

→ Written from an OOP perspective





How it often goes

Business need

→Analyst Interpretation

→Developer Interpretation

→Software

AXXES

To remove interpretations,
we must share a language

AXXES

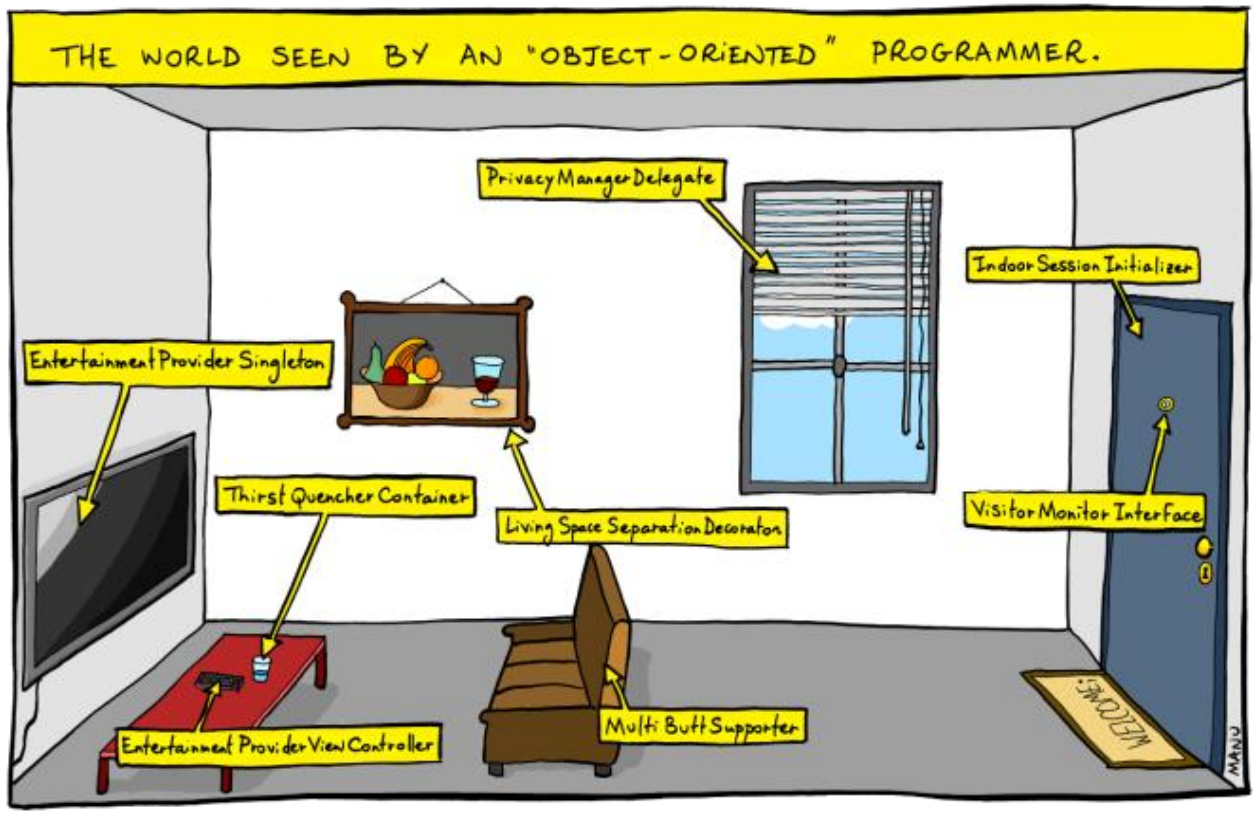
Ubiquitous Language

A ubiquitous language is ***a vocabulary shared by everyone involved in a project****, from domain experts to stakeholders, to project managers, to developers*

AXXES

Ubiquitous Language

- Names in code
  = names in conversations
- Strip meaningless terms
- Strip implementation details



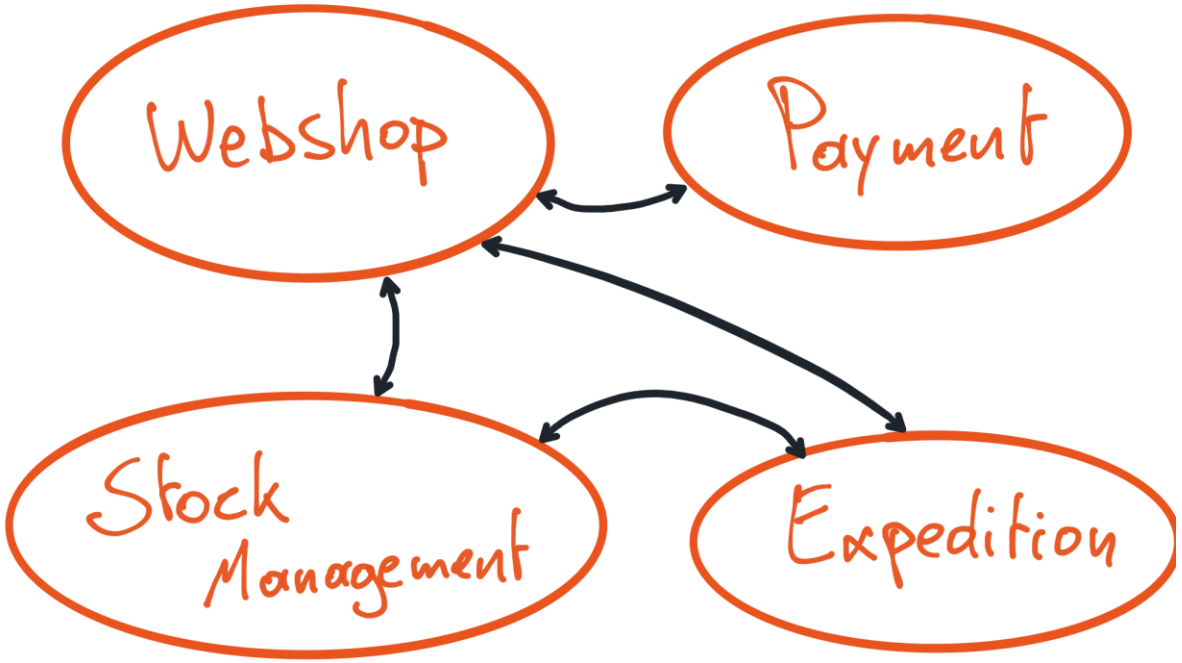
AXXES

Organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations

Melvin Conway

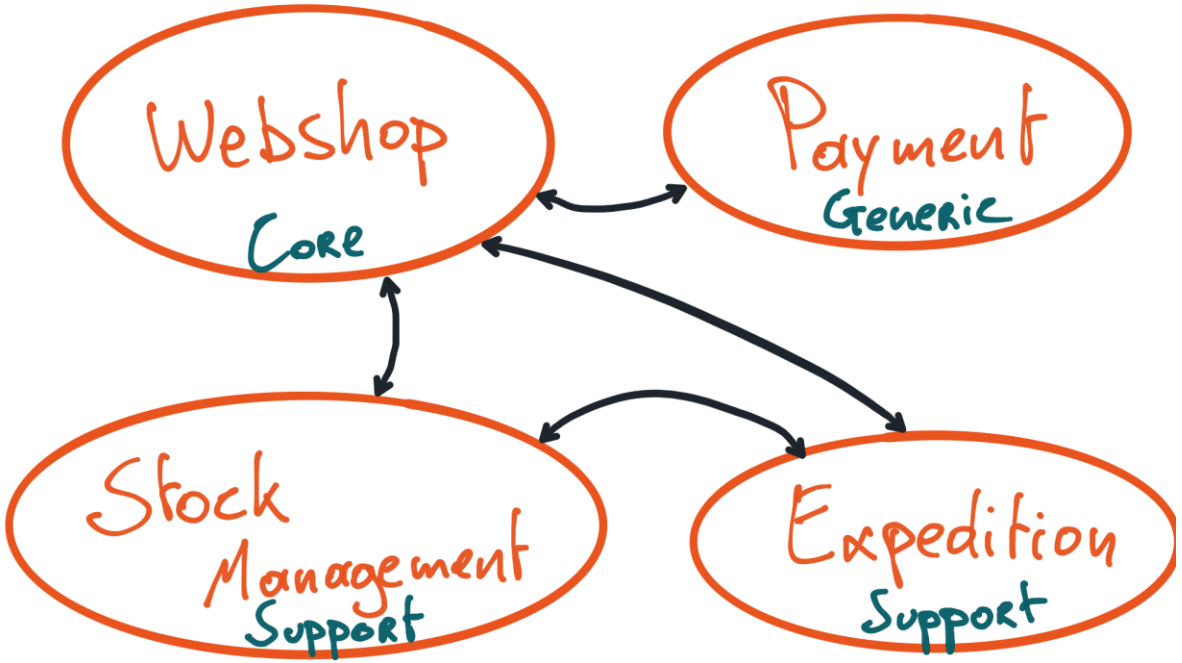
AXXES

Bounded Context



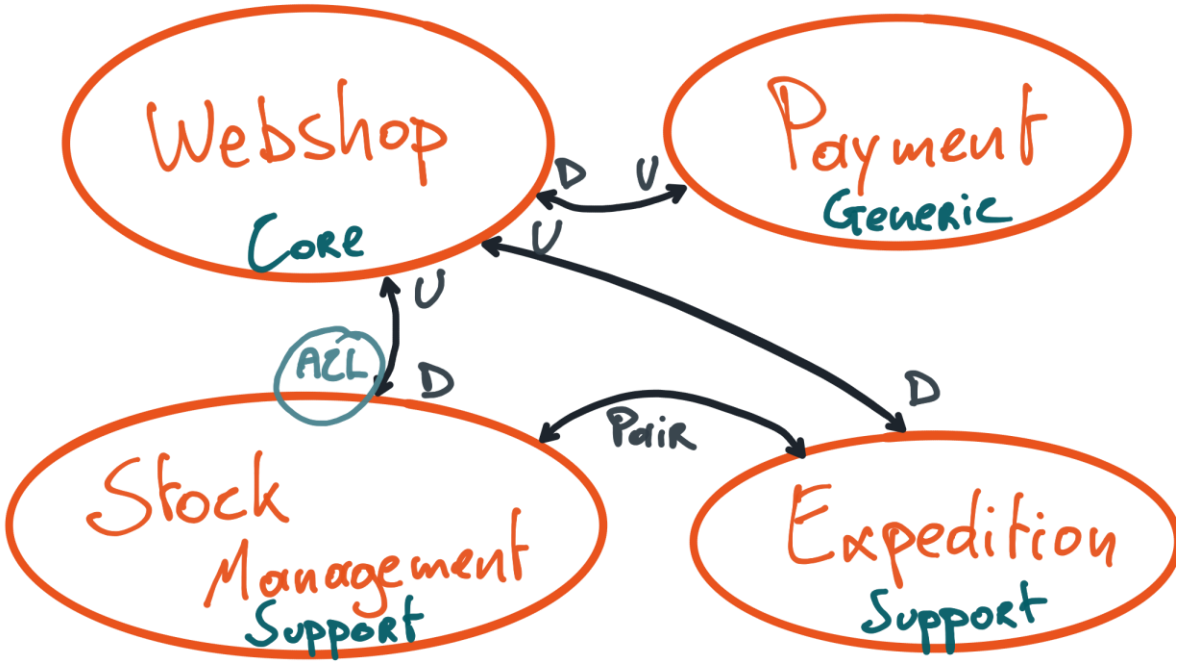
AXXES

Bounded Context



AXXES

Bounded Context



AXXES

Commands & Events

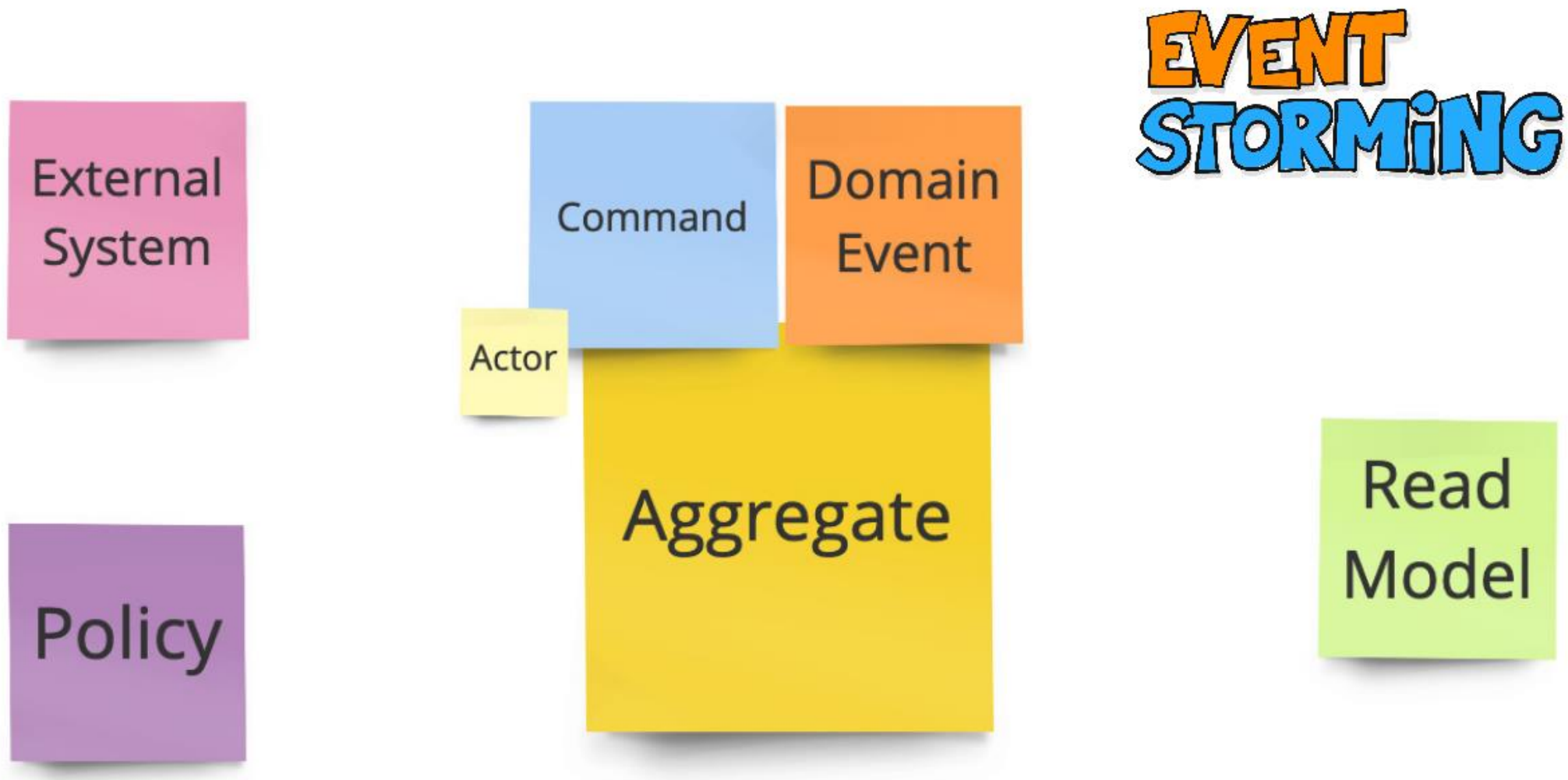
Commands

- Imperative form
- Request to perform an action
- *E.g. SendBeer*

Events

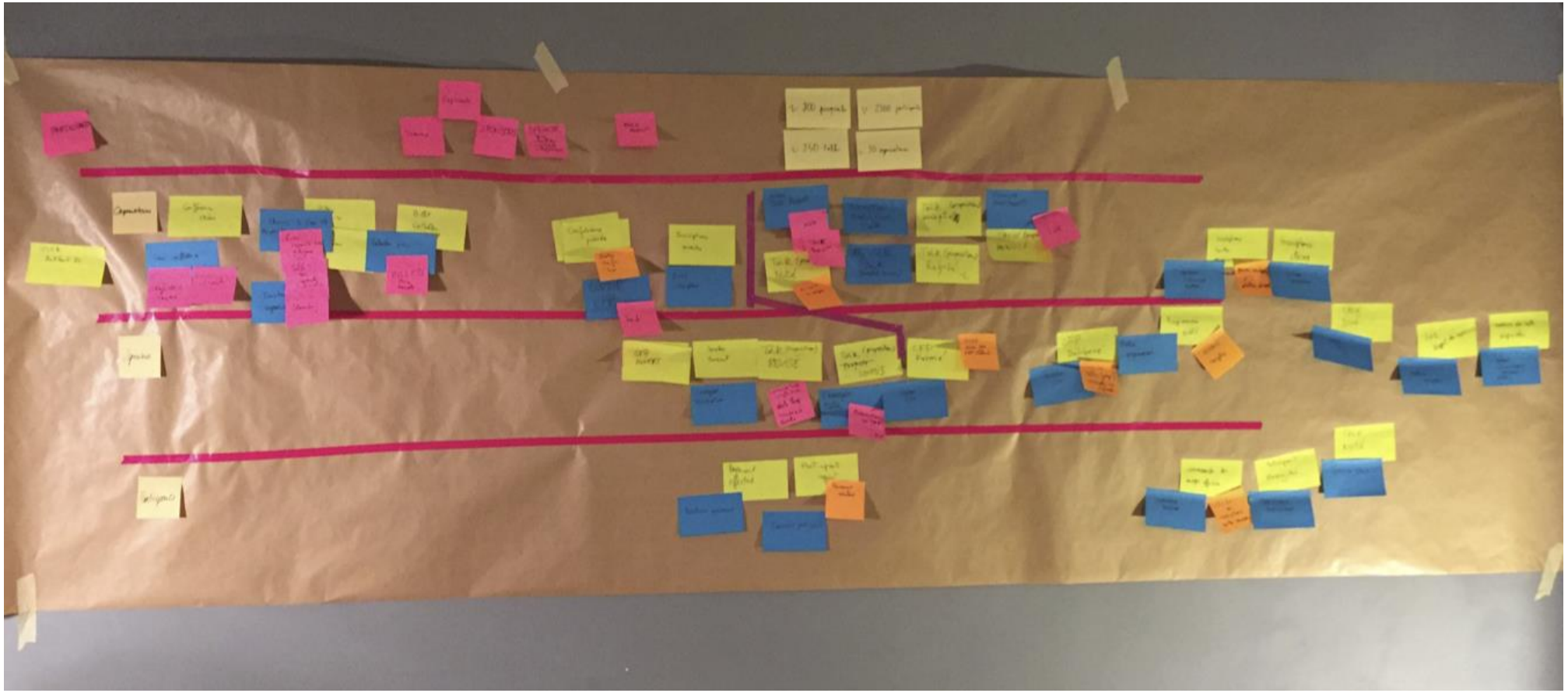
- Past tense
- Communicates that an action has been performed
- *E.g. BeerReceived*

AXXES


EVENT
STORMING
External System
Command
Domain Event
Actor
Aggregate
Policy
Read Model

AXXES

Event Storming



AXXES

Aggregate

- Handles Commands
- Raises Events
- Is the root of reasoning (sometimes called Aggregate root)
- Holds all the info required to respond to Commands & Events
- Is small enough to reason about

After bounded contexts, aggregates are the hardest to define.

Value Types

- Help in bringing Ubiquitous language to Code
- Make the implicit explicit
- Group logic with data
- Easier to reason about
- Easier to test

Now with C# records!

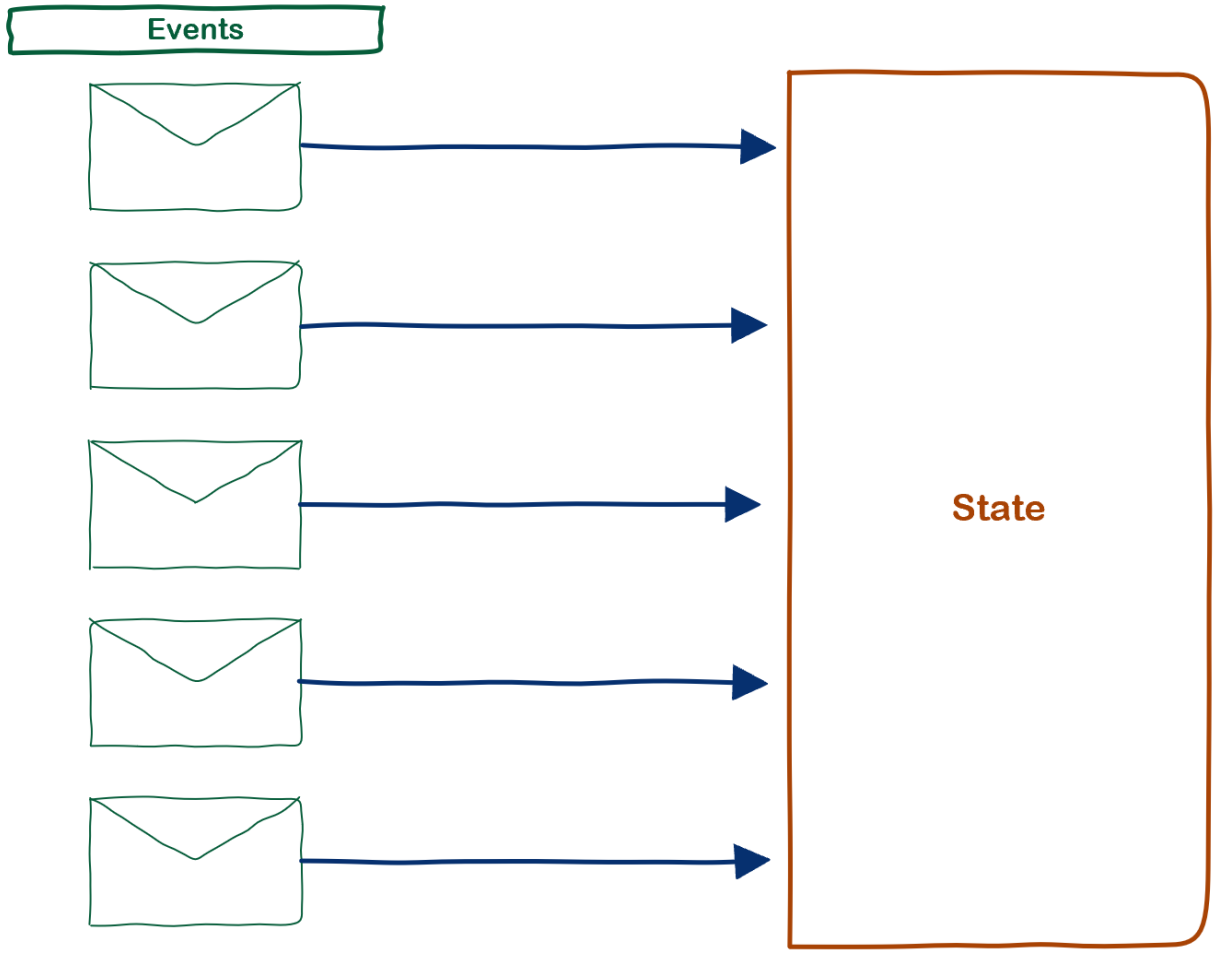
Event Sourcing

–

Keep track of our history

AXXES_

Event Sourcing



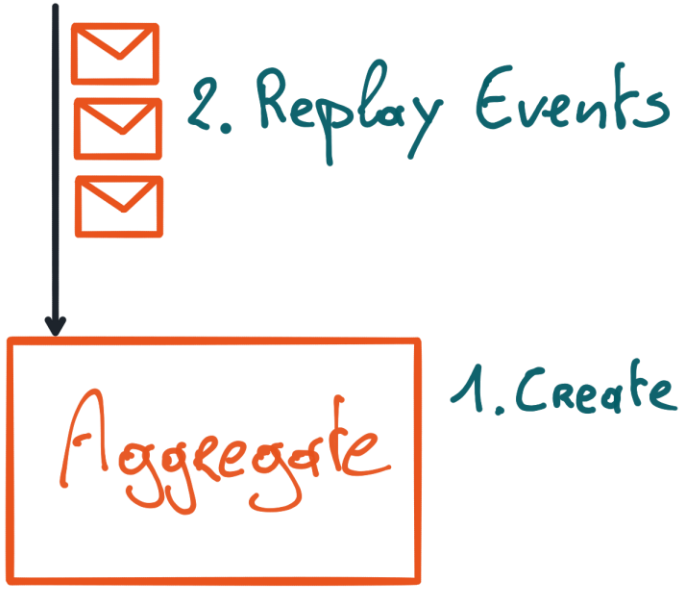
- Don't store state
- Store events
- Project state when needed

AXXES

Aggregate life cycle

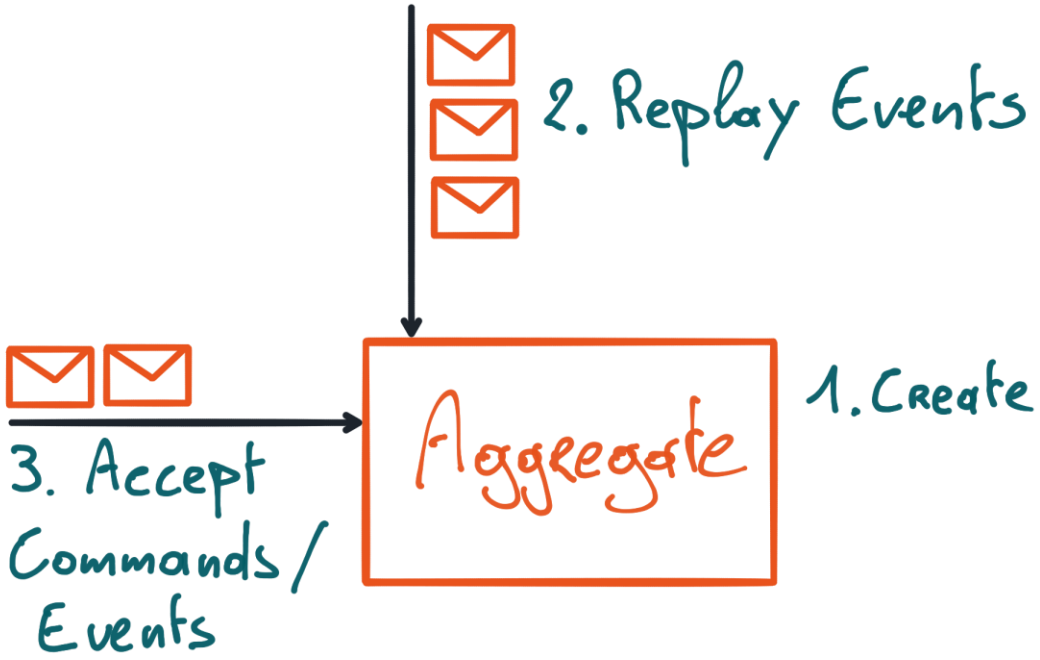
AXXES

Aggregate life cycle



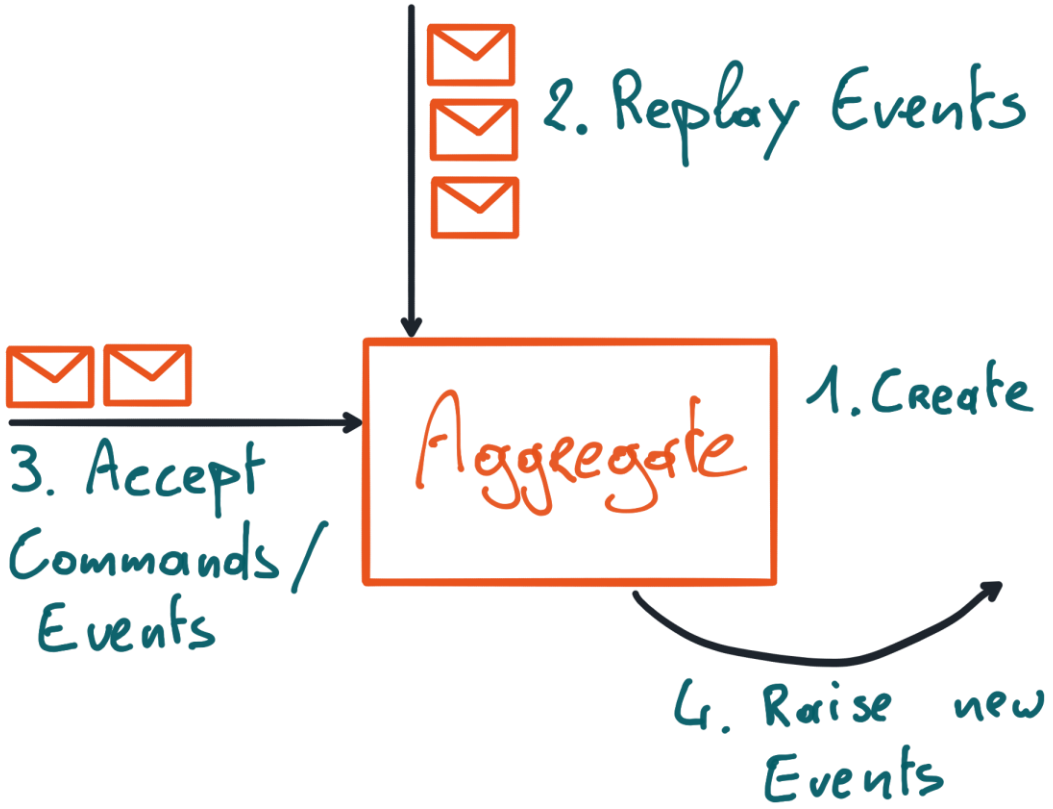
AXXES

Aggregate life cycle



AXXES

Aggregate life cycle



AXXES

CQRS

–

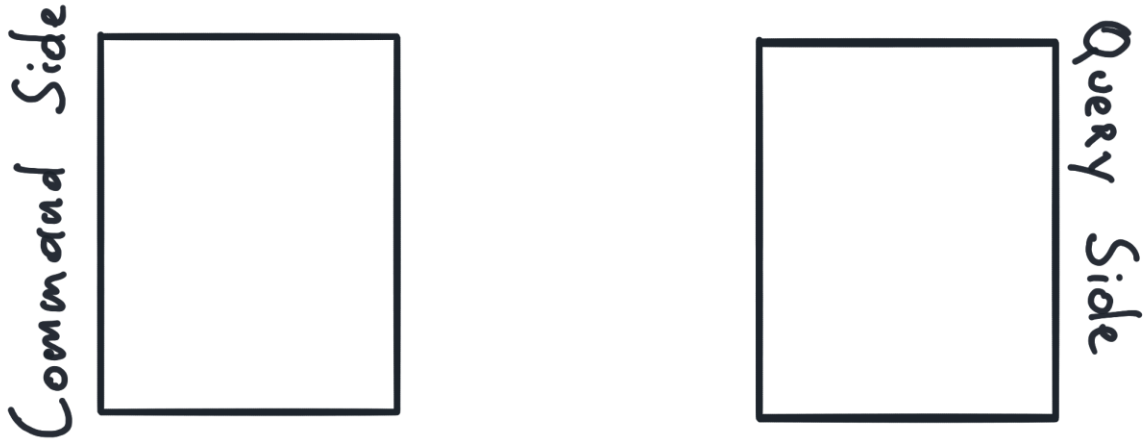
Command – Query Responsibility Segregation

AXXES

CQRS

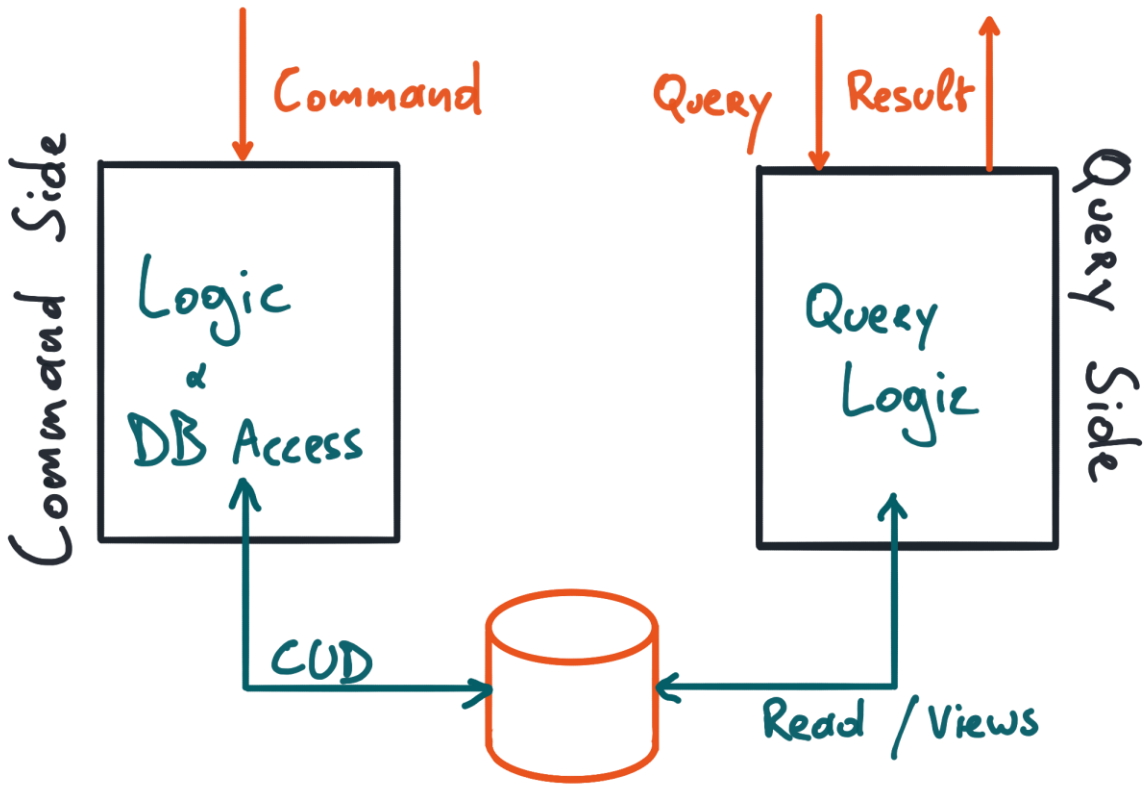
- Separate the write from the read side
- Command side (write) responds to commands
- Query side (read) doesn't change state
- This enables:
  - Performance tuning sides separately
  - Eventual consistency
  - Keep history (when used with ES)
  - …

CQRS



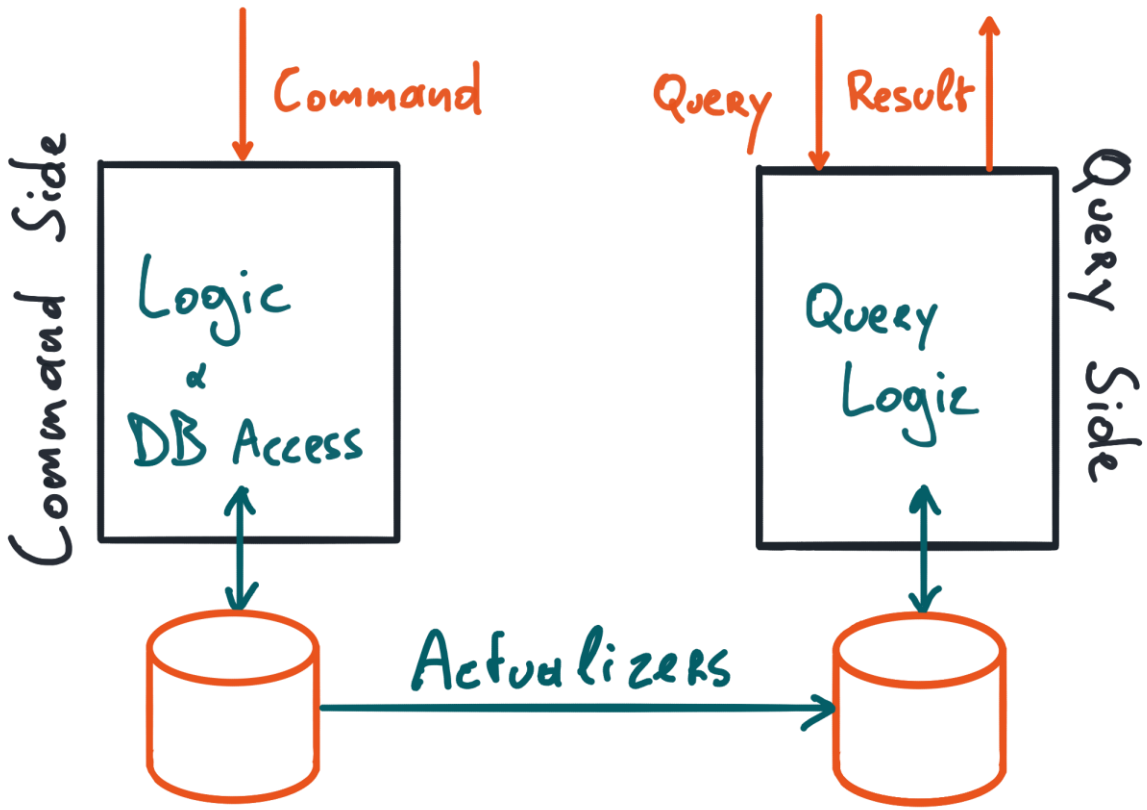
AXXES

CQRS


Command
Query
Result
Command Side
Logic
&
DB Access
Query Side
Query
Logic
CUD
Read / Views

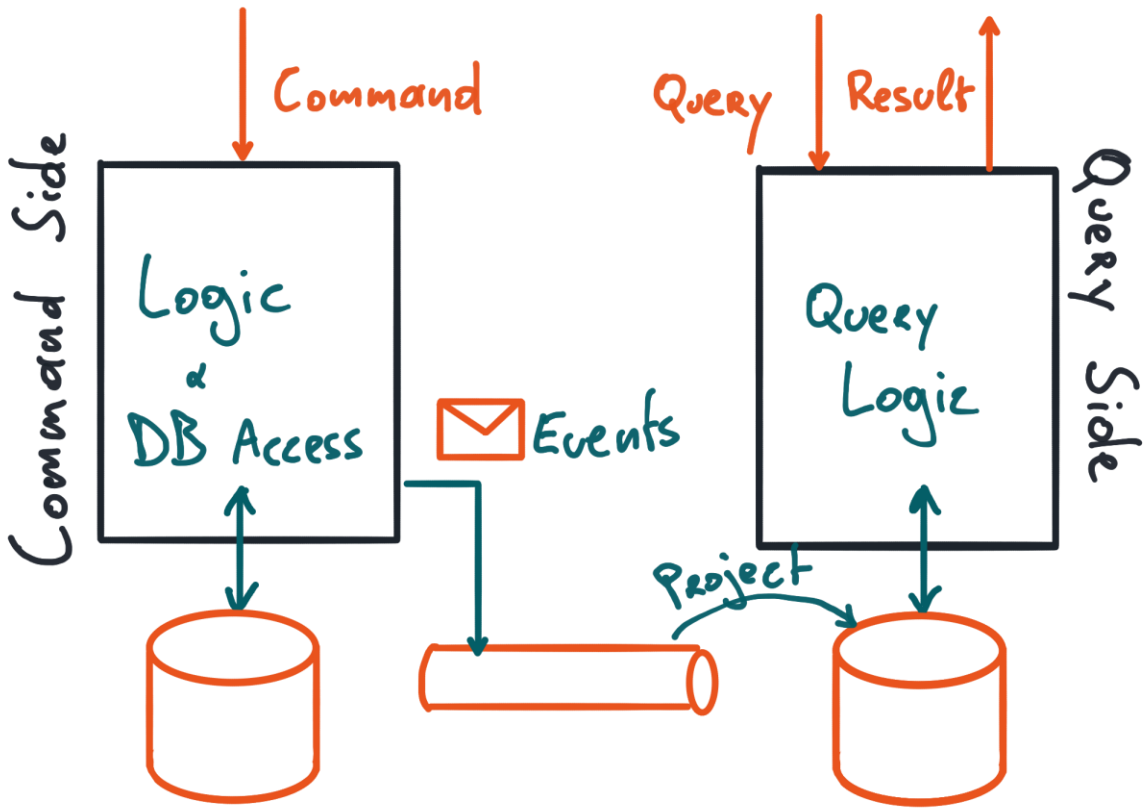
AXXES

CQRS


Command
Query
Result
Command Side
Logic
&
DB Access
Query
Logic
Query Side
Actualizers

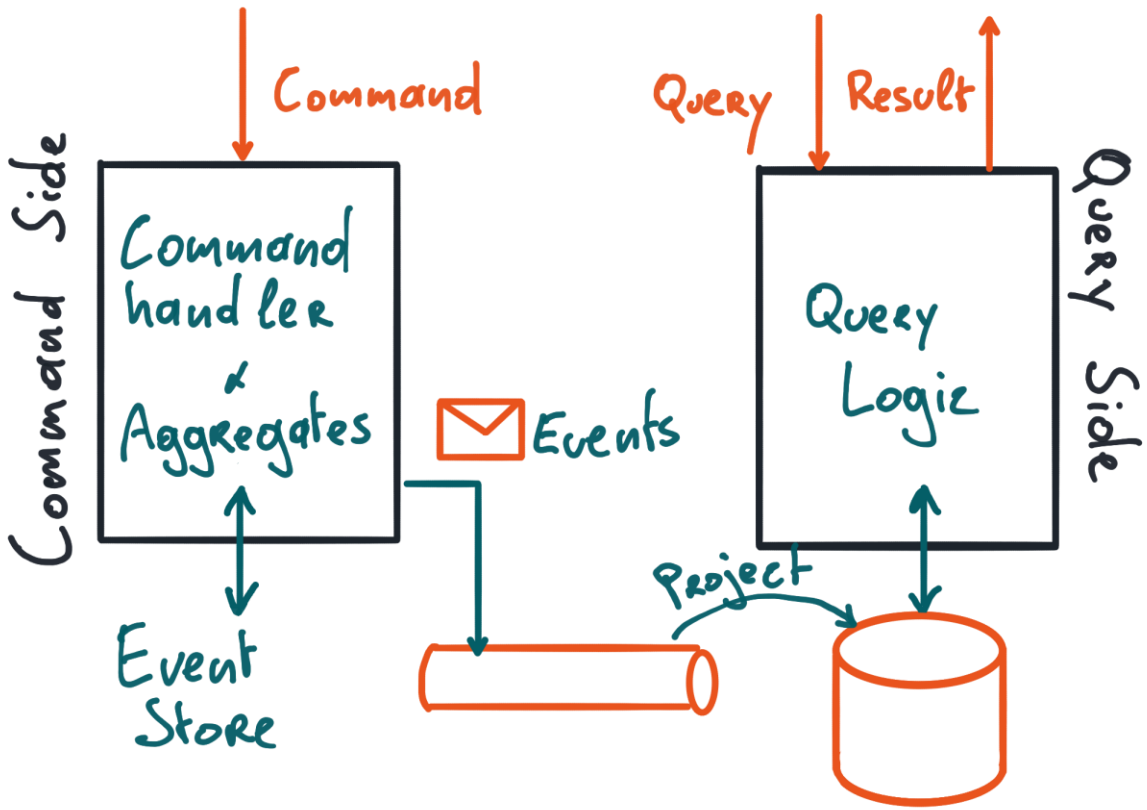
AXXES

CQRS


Command
Query
Result
Command Side
Logic
&
DB Access
Events
Query
Logic
Query Side
Project

AXXES

CQRS


Command
Query
Result
Command Side
Command handler & Aggregates
Events
Query Logic
Query Side
Project
Event Store

AXXES

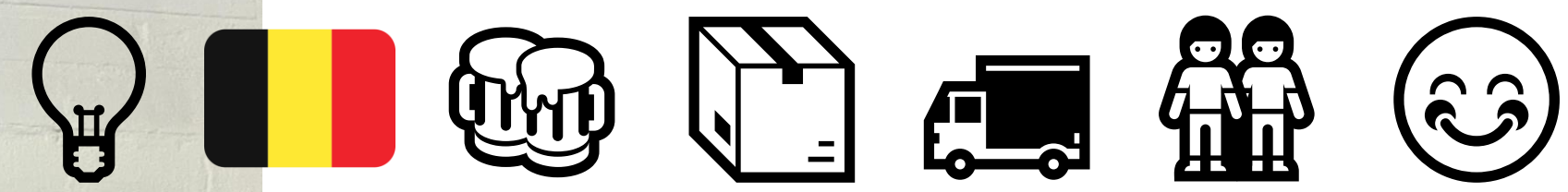
What are we building?

–

BeerSender.NET

AXXES_

Lockdown boredom



AXXES

Execution

AXXES

Expectation

Reality

12/01/2022 15:41	**Returned** Returned to shipper MOL, BE
11/01/2022 19:07	**Returning to Sender** UPS initiated contact with receiver or importer for clearance information. Once received, UPS will submit for clearance. / The package will be returned to the sender. Lummen, Belgium

11/01/2022 13:02	UPS initiated contact with the sender to obtain clearance information. Once received, UPS will submit for clearance. / The information requested has been obtained and the hold has been resolved. Lummen, Belgium
10/01/2022 22:22	UPS initiated contact with the sender to obtain clearance information. Once received, UPS will submit for clearance. Lummen, Belgium
10/01/2022 22:21	A missing commercial invoice is causing a delay. We are currently waiting for information from the sender. Lummen, Belgium
10/01/2022 22:20	Warehouse Scan Lummen, Belgium
10/01/2022 19:40	Origin Scan Lummen, Belgium
10/01/2022 14:00	Collection Scan Lummen, Belgium
08/01/2022 11:41	Your parcel is currently at the UPS Access Point™ and is scheduled to be tendered to UPS. Lummen, Belgium
08/01/2022 11:41	Drop-Off Lummen, Belgium
07/01/2022 22:21	Your parcel is pending release from a Government Agency. Once they release it, your parcel will be on its way. Lummen, Belgium
07/01/2022 22:21	Your parcel is on the way
07/01/2022 22:20	Shipper created a label, UPS has not received the package yet. Belgium

AXXES

Ended up being

- Tracking
- Providing extra customs info
- Emails
- UPS helpdesk 😠
- Packages returned
- Recuperating shipping costs
- Re-sending
- No delivery attempts
- Re-re-sending

AXXES

Sending beer is tricky

- Making & filling boxes is fun
- The rest, not so much
- Follow-up is key!

→ We need an app for that!

AXXES

BeerSender.NET

-

Because it's not written in VB6!

AXXES

Let’s start building!

tinyurl.com/dotnetdaysro-miro

Password: ???

tinyurl.com/dotnetdaysro-github

AXXES

Optimizations

–

How can we optimize this system further?

AXXES

3 Types of projections

1. Immediate:
   Completes together with the Command
   Projected data is immediately available
   Slows down Commands
2. Eventual:
   Events get placed on a bus
   Projections are processed ASAP
   Slows down the availability
   Speeds up Commands
3. From source events:
   Queries are projected as required
   No unnecessary processing
   Might be slower with many source events

AXXES

Caching Aggregates

Keeping aggregates in memory

=

lightning fast command processing

AXXES

Snapshots

When aggregates get MANY events:

- Group the aggregate's state in a state object
- Save state sporadically
- Replay events from snapshot moment

AXXES

Query indexing & caching

Step 1:

- Every query hits an index
- Every query hits a single table (ideally)

Step 2:

- API output caching