



REX on Event Sourcing

Nizar KEFI

@nizarazu9

Haythem ZAYANI

@HaythemZayani



SOCIÉTÉ
GÉNÉRALE

Event Sourcing ?

I know
someone that
...

DDD ?

What's wrong
with my
implem ?

How to do it
right ?

Do I really
need it ?

Why is it so
complicated ?

Is it killing my
project?

Where should
I begin ?

Here THE way
to do it !

CQR... what ?

Event Sourcing ?

Don't look after **THE** way to do it

What works for others **may not** work for you

It's all about **tradeoffs**

Who are we?



Nizar
Developer
@nizarazu9



Haythem
Developer
@HaythemZayani



Why are we talking about Event Sourcing ?

The Financing Platform is a platform that
originates, syndicate and control
corporate loan opportunities

Microservices

(communicating with
legacy applications)

Domain
Driven Design

(as much as possible)

Event
Sourced

(With CQRS)

Message
Driven
architecture

Why do we use Event Sourcing

Audit (regulatory compliance)

Behavior analysis

Debugging and testing

How will we proceed?

Domain
Events

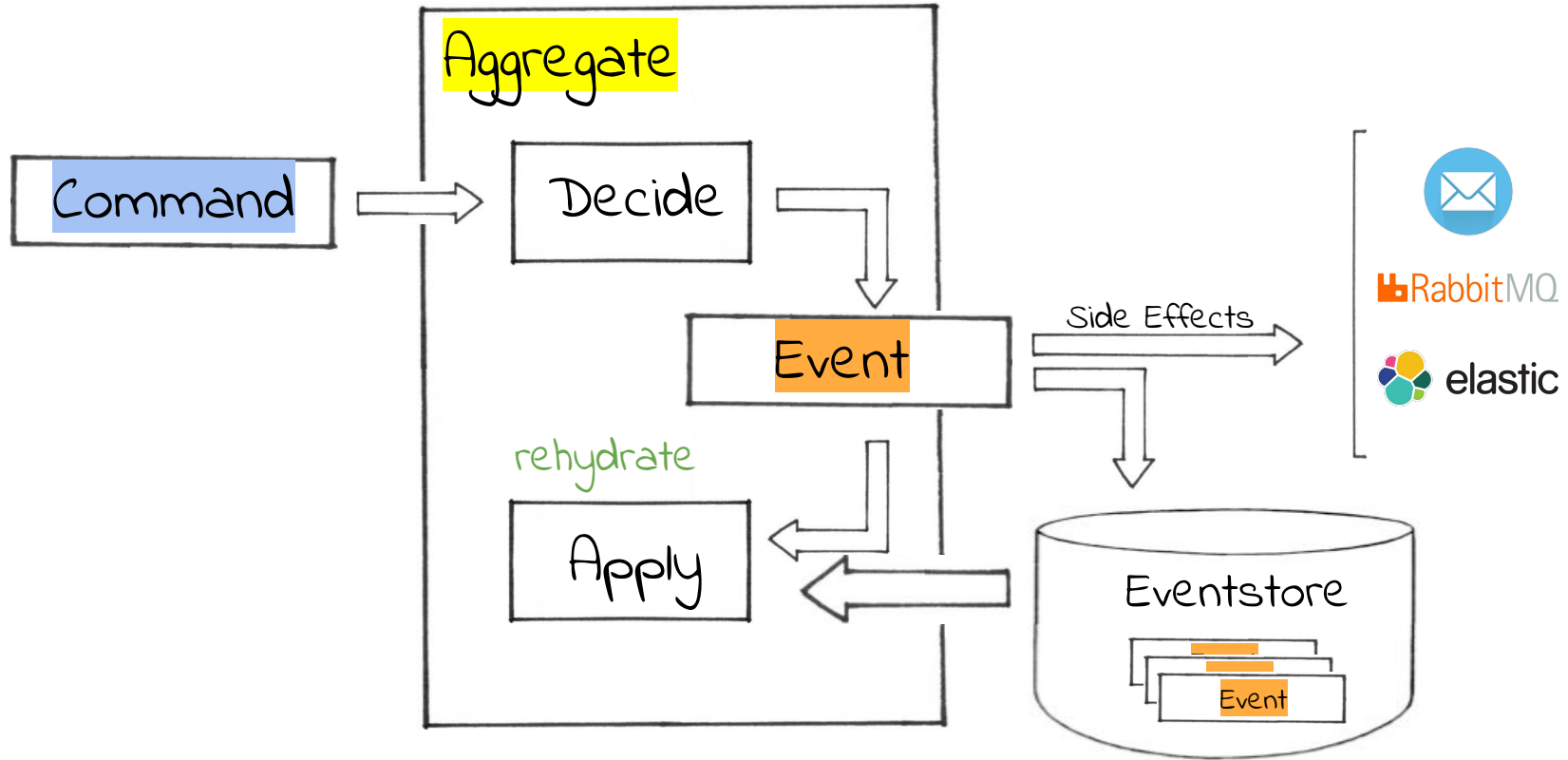
Boundaries
&
logic

Architecture
&
Mechanisms

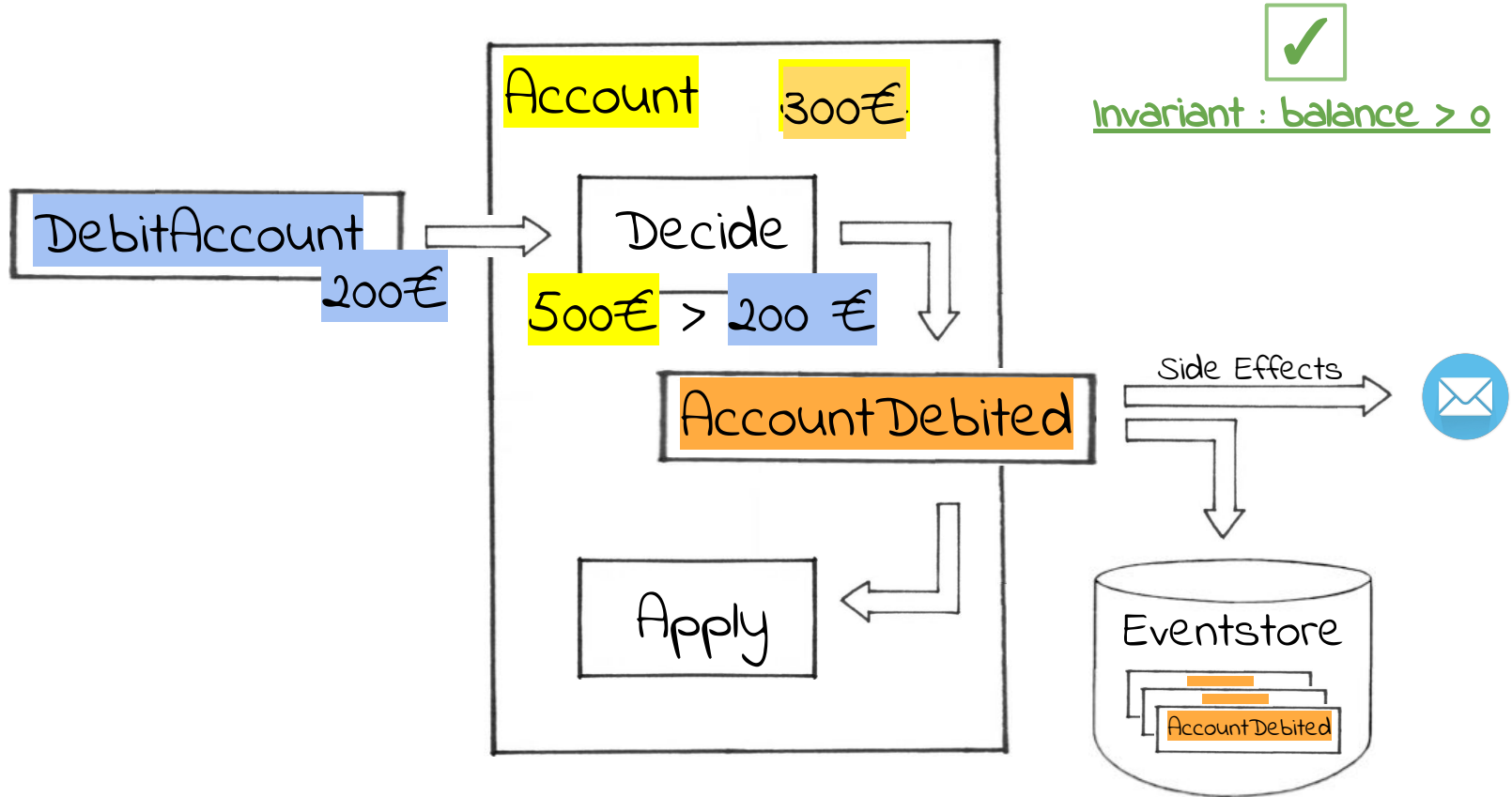
Q&A

Event Sourcing Definitions

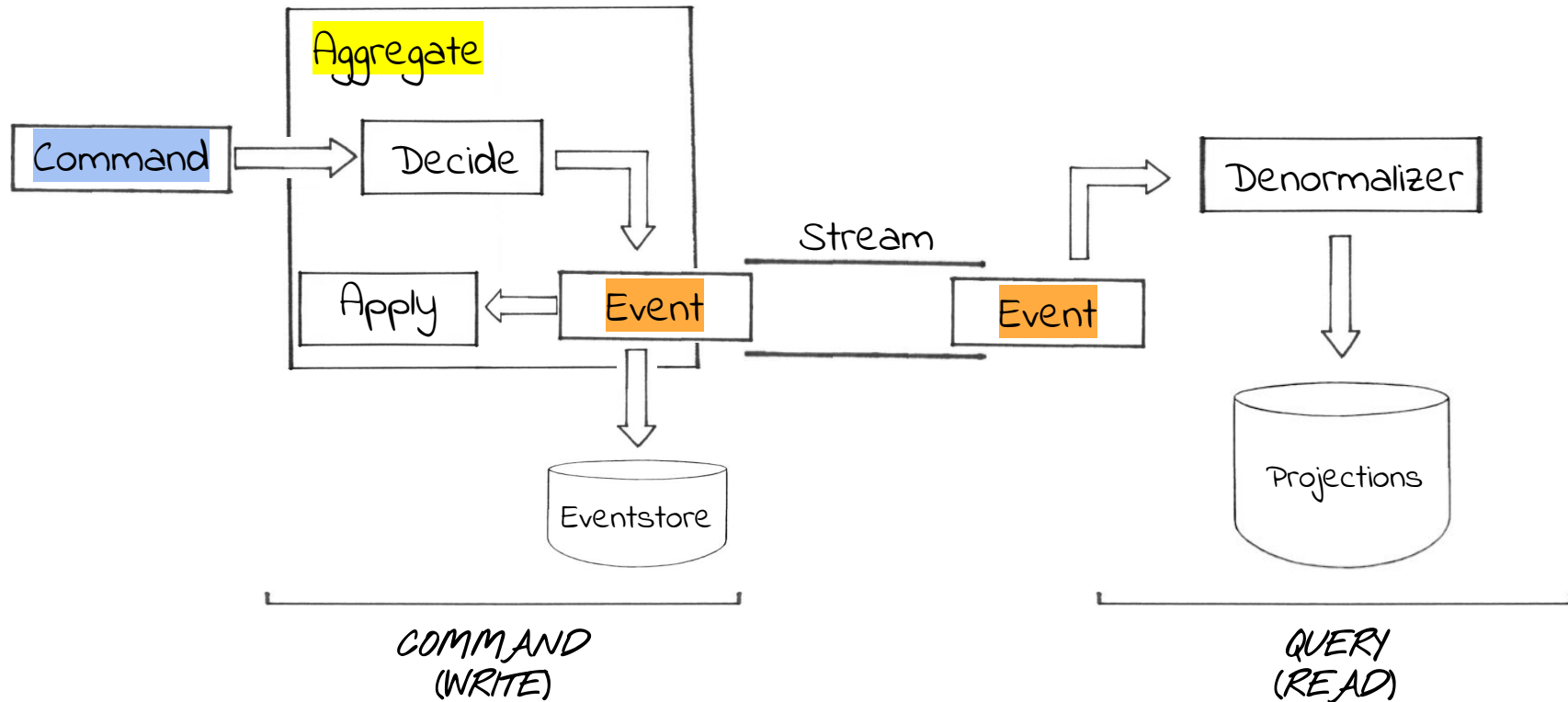
Event Sourcing : Under the Hood



Event Sourcing : Example



Event Sourcing & CQRS



Event Sourcing & CQRS

Event Sourcing doesn't require CQRS

Eventual Consistency

Domain Events

How to define Events ?

Event Storming, Behavior Driven Development, ...

Focus on the domain behaviors

Identify Bounded Contexts

Pay attention at the words and meanings

Domain Events : Don't be CRUDe !



#1 AccountCreated

#2 AccountUpdated

#3 AccountUpdated

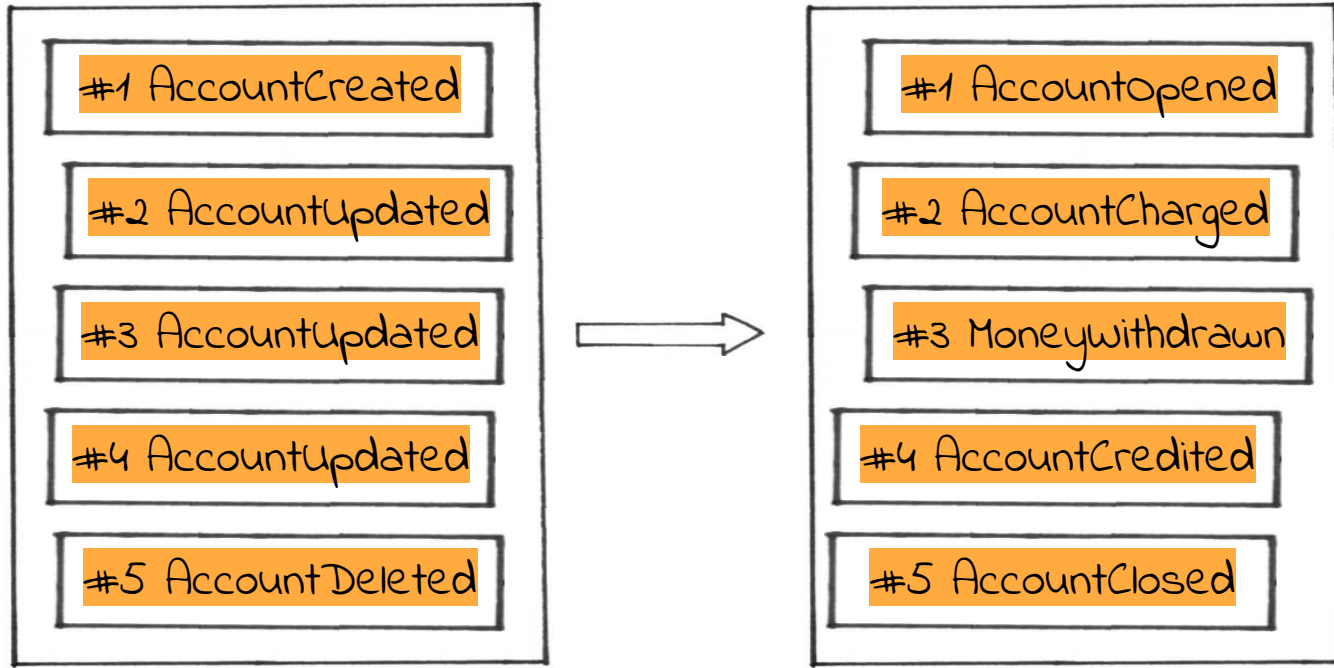
#4 AccountUpdated

#5 AccountDeleted

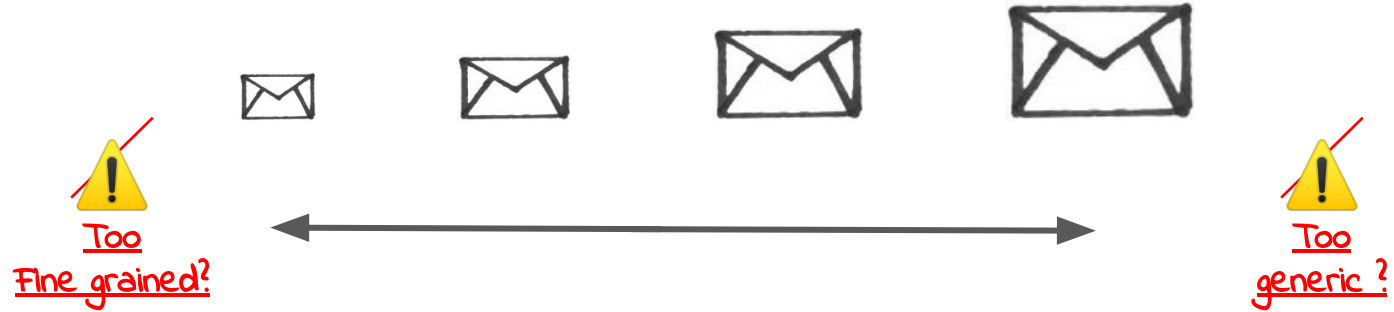
✓ use ubiquitous
Language

✓ Challenge
Domain Experts

Domain Events : Don't be CRUDe !



Domain Events : Granularity



Domain Events : Upcaster



Dealing with
many versions

New Version
or
Split
or
Merge

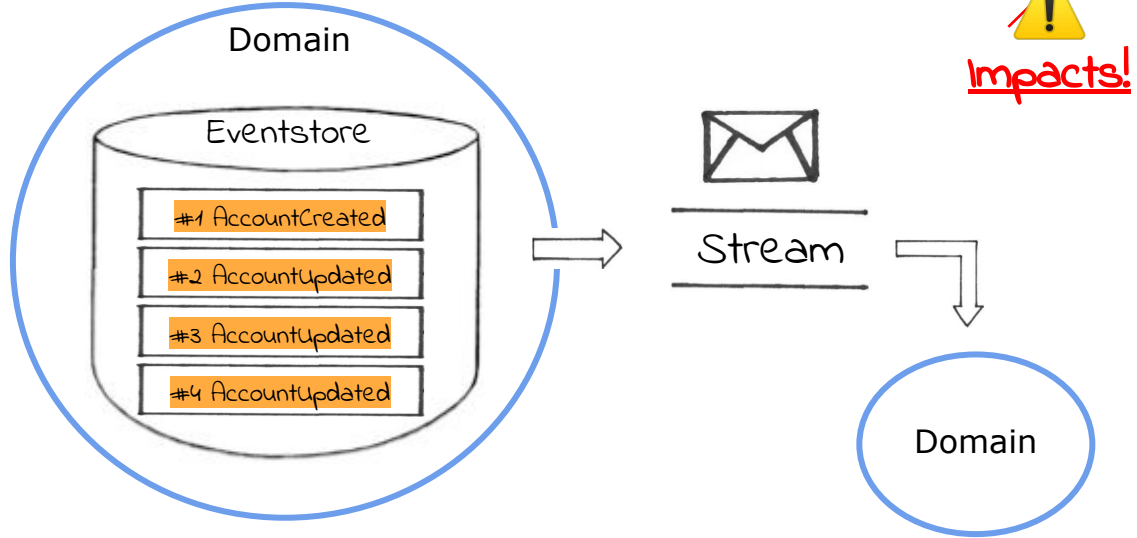


Hard to maintain

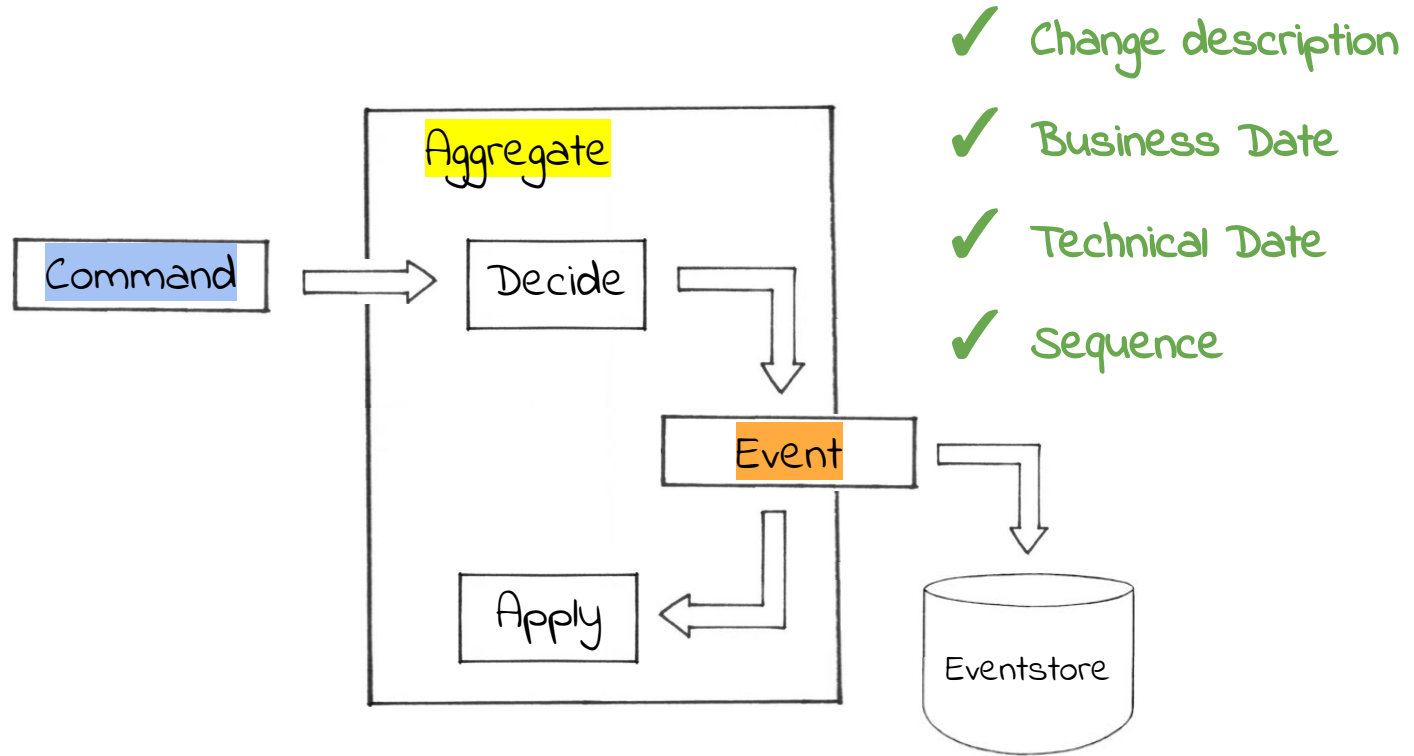
Runtime
vs
persisted



Became very
complex



Domain Events : Content

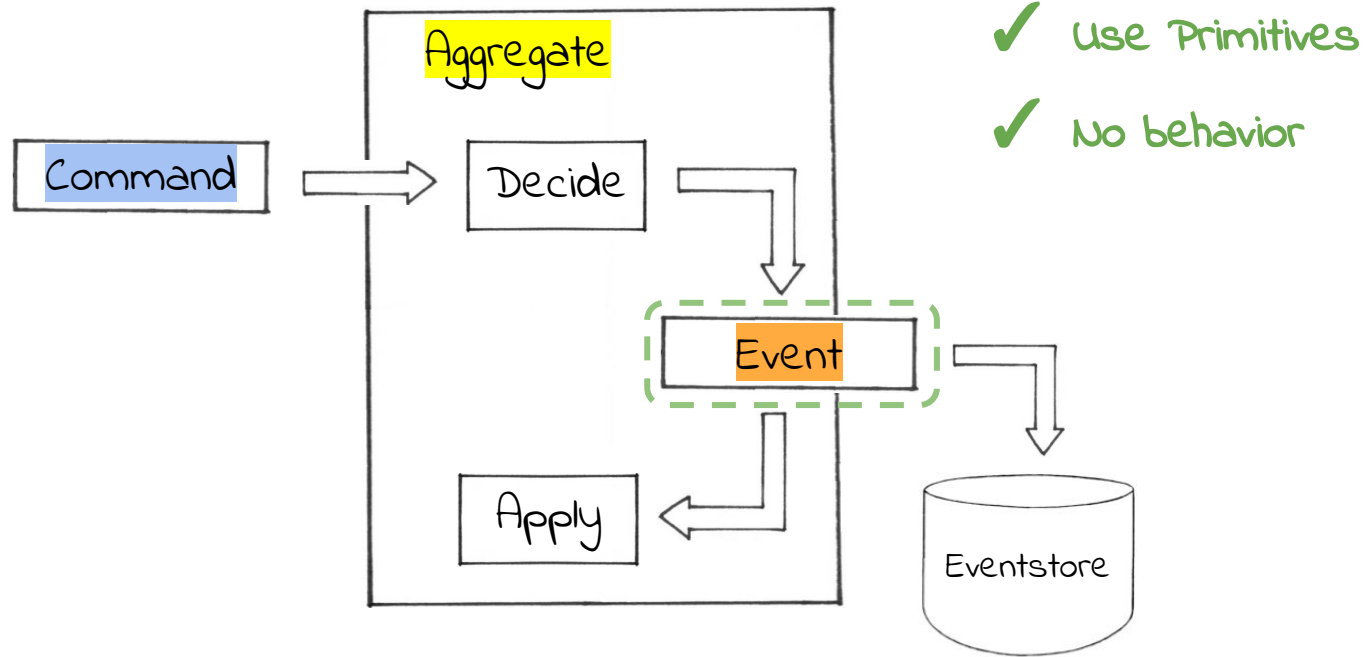


Domain Events : Use Primitives

Value Object

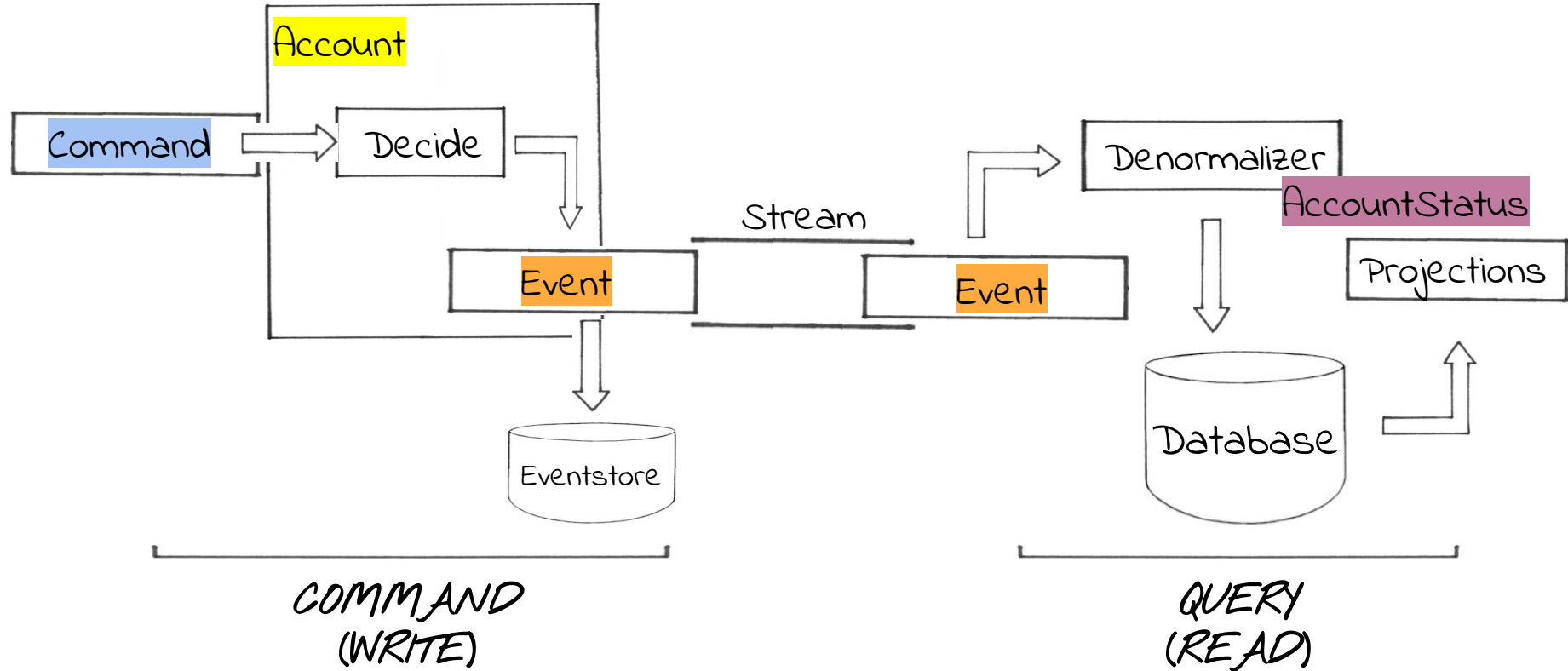
Example :
Amount, Currency, Age...

Domain Events : Use Primitives



Aggregates

Logic out of the Aggregate

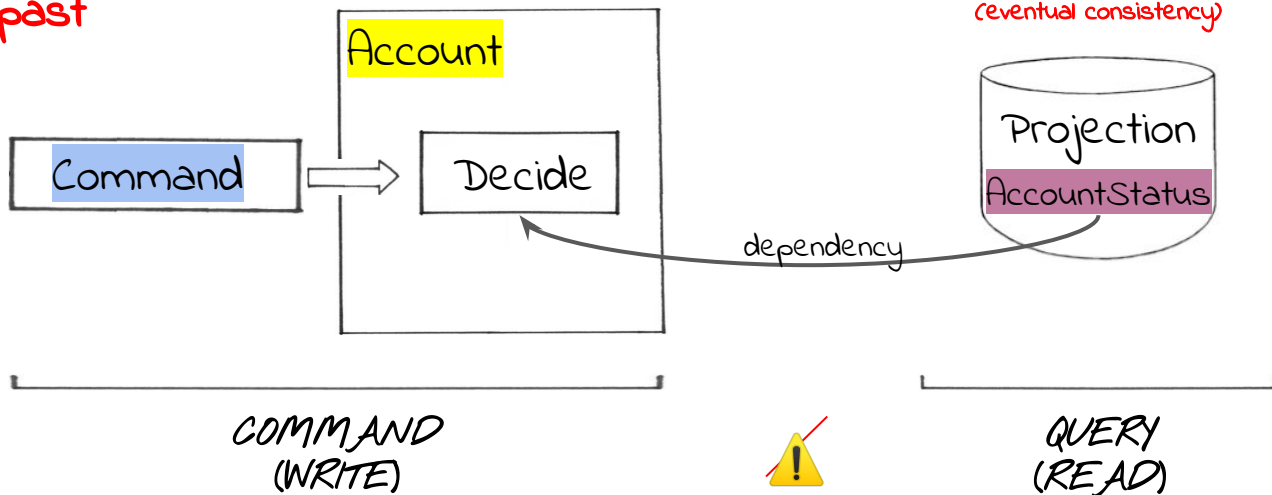


Logic out of the Aggregate

! May change the past

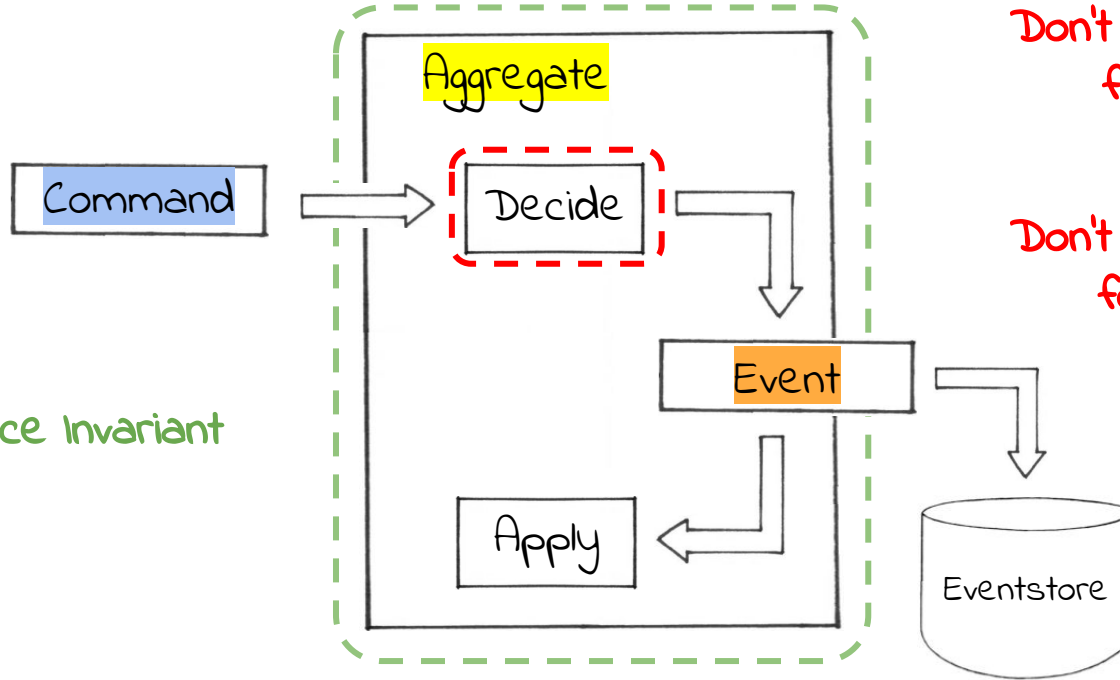
! Prevent auditing

! Projection may not be up to date
(eventual consistency)



Logic out of the Aggregate

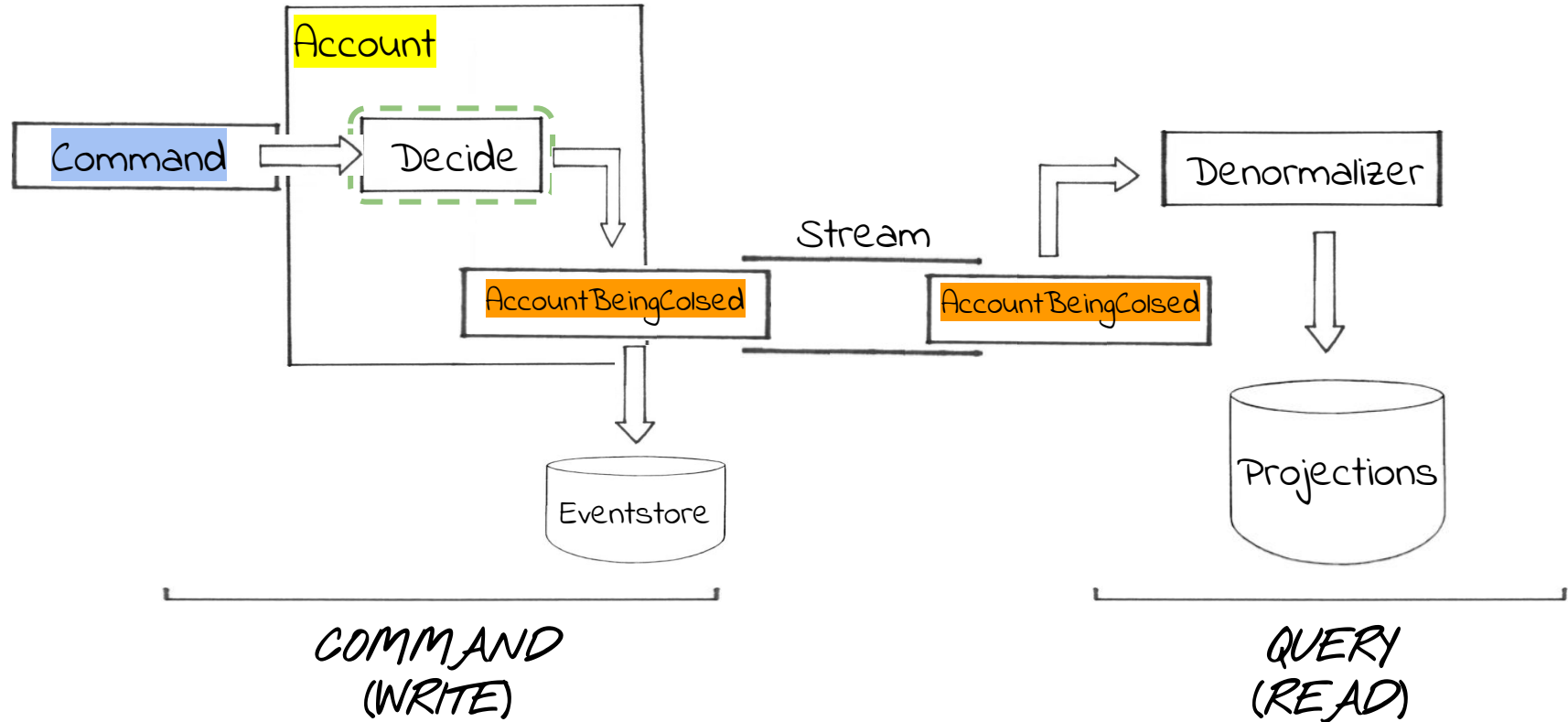
✓ Enforce Invariant



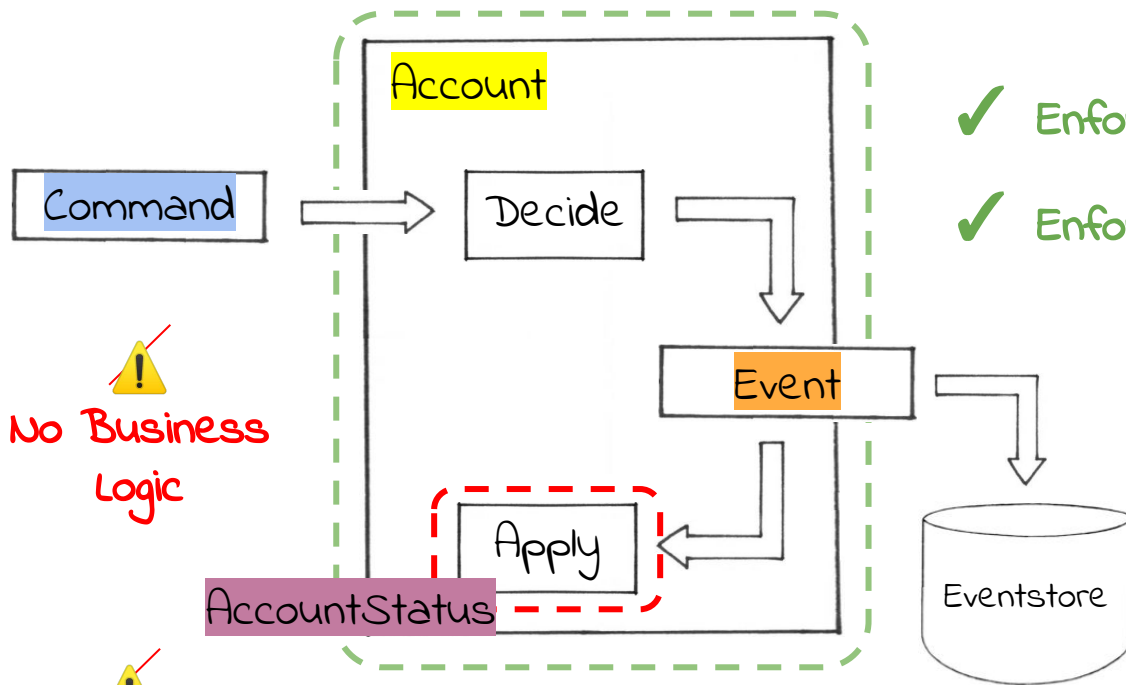
⚠ Don't use Projections for Decision

⚠ Don't use Projections for Validation

The logic should be in the Aggregate



Aggregates : Apply Function



- ✓ Enforce Invariant
- ✓ Enforce Consistency

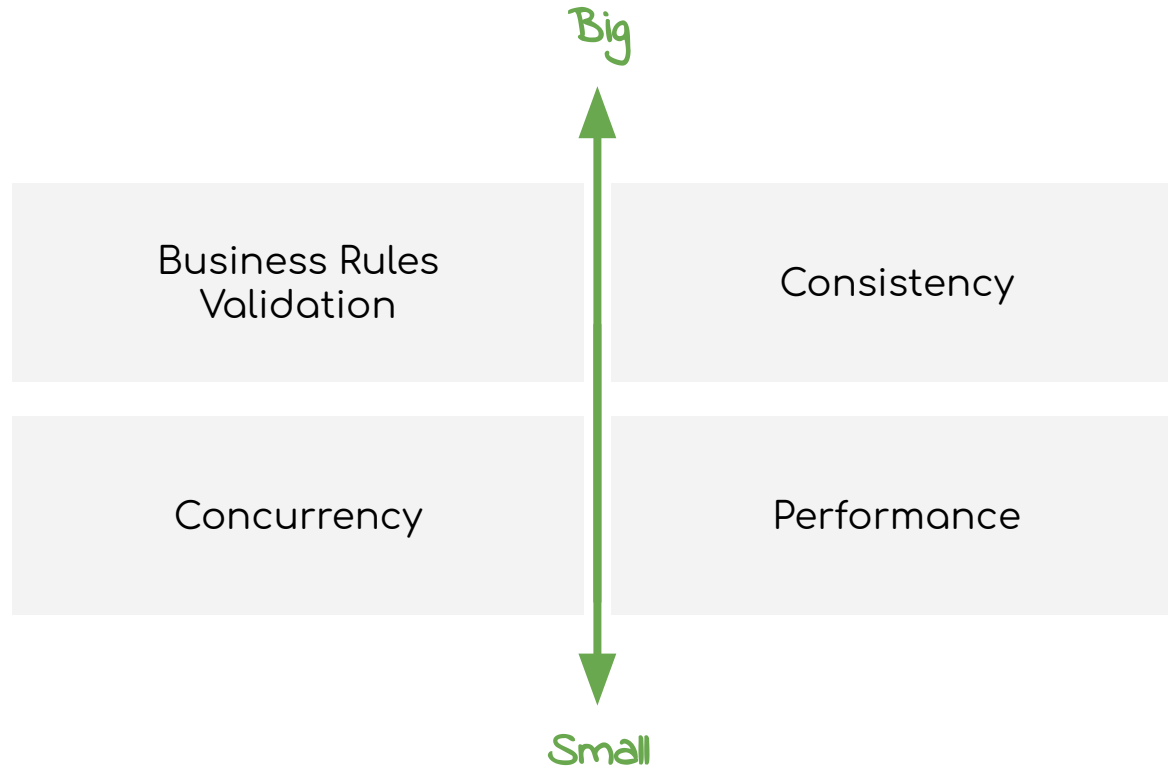
⚠
Prevent
auditing

⚠
No Business
Logic

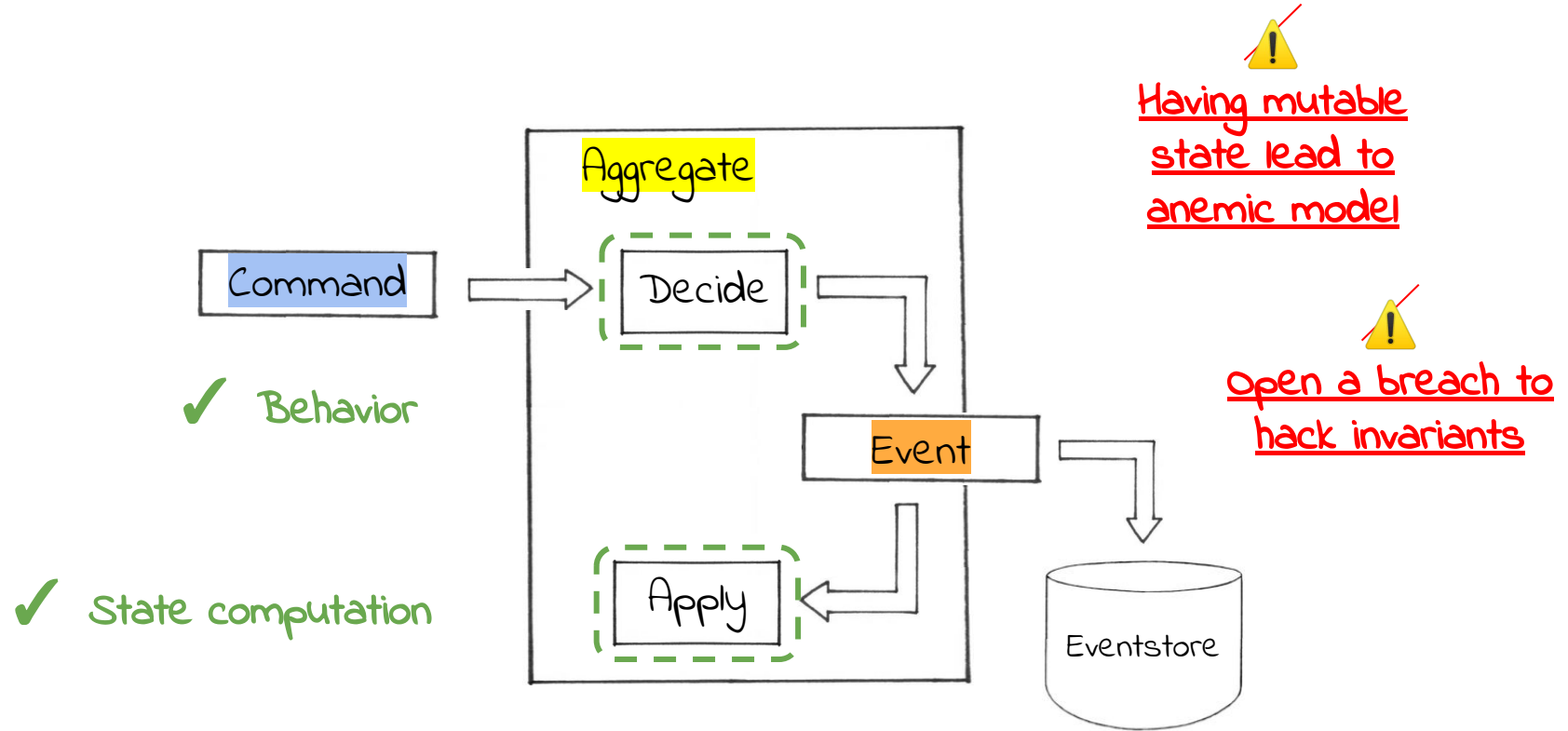
⚠
May break old
aggregate
rehydration

⚠
May change
the past

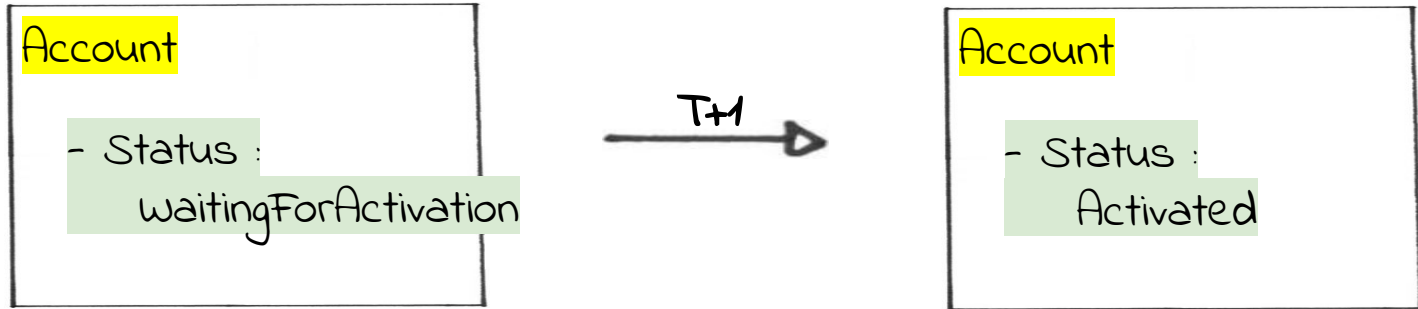
Aggregates : Granularity



Aggregates : No Setter



Aggregates : No Default Value



Change the past

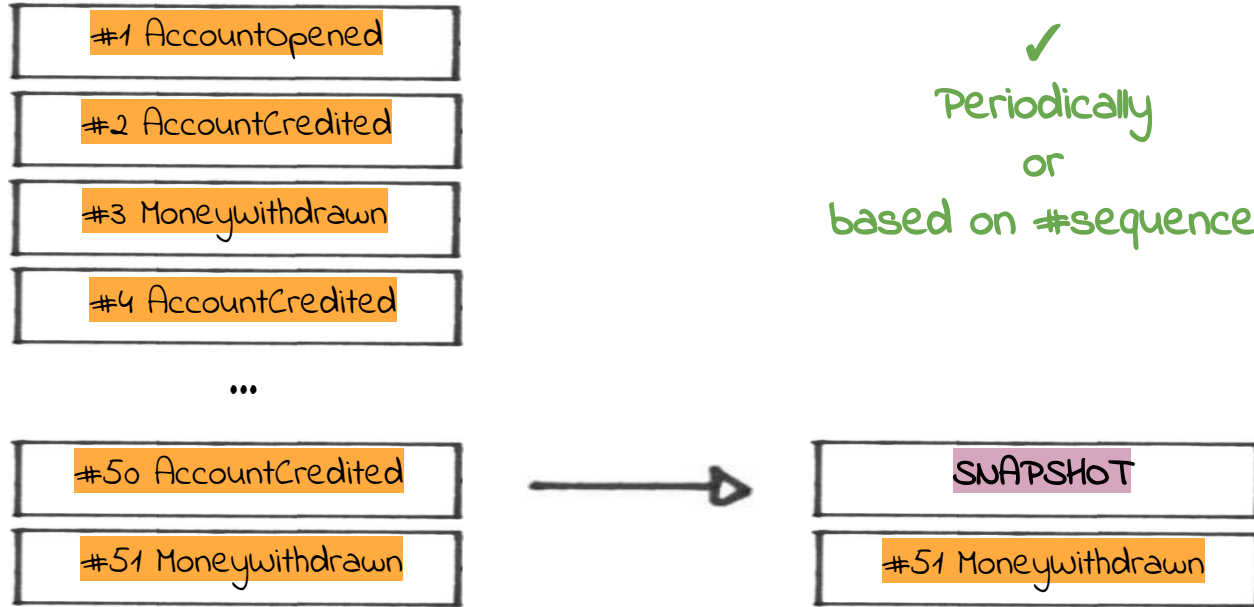
Aggregates : Tests

Given {
Event,
Event
}

When {
Command
}

Then {
Event
}

Aggregates : Snapshot

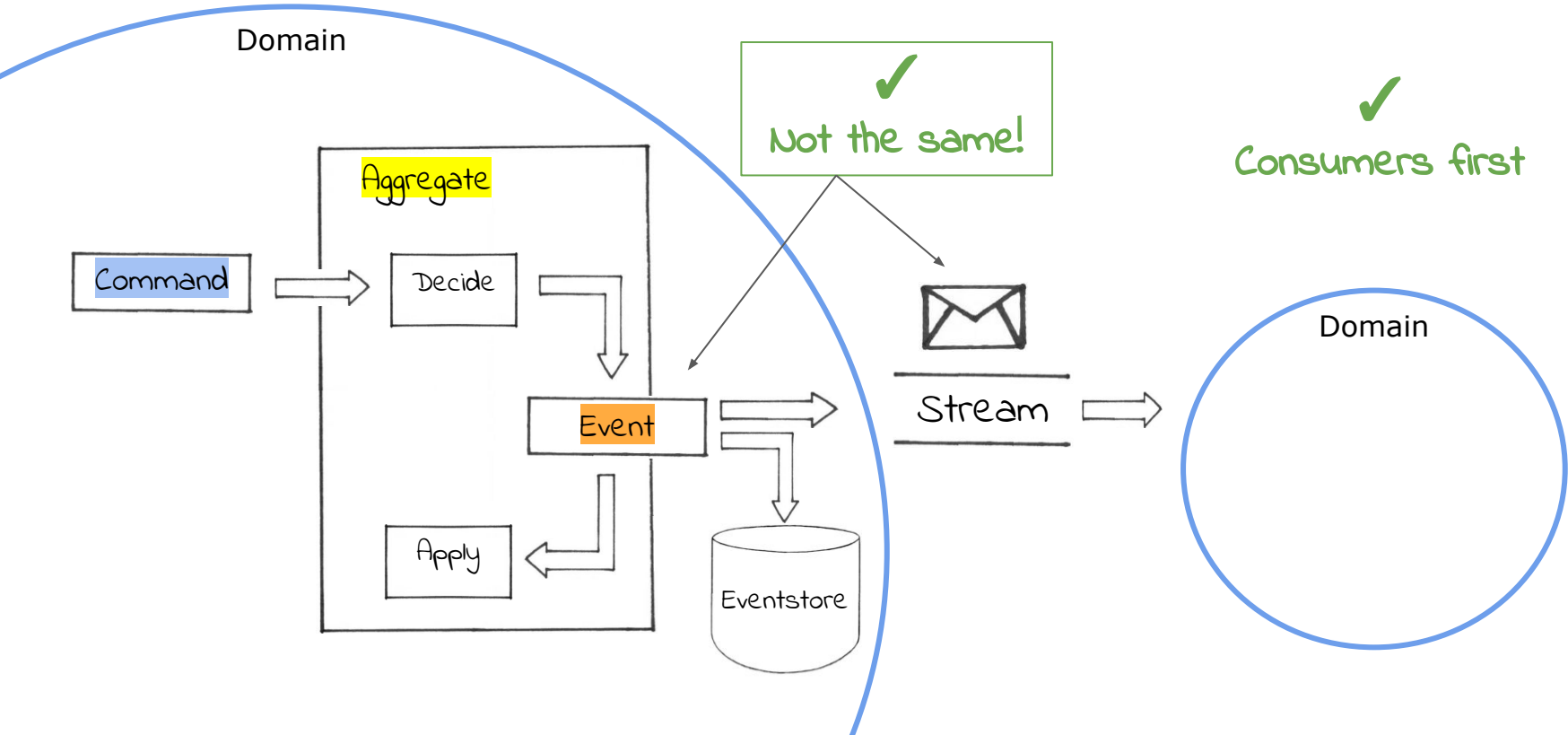


Event
Sourcing
And more...

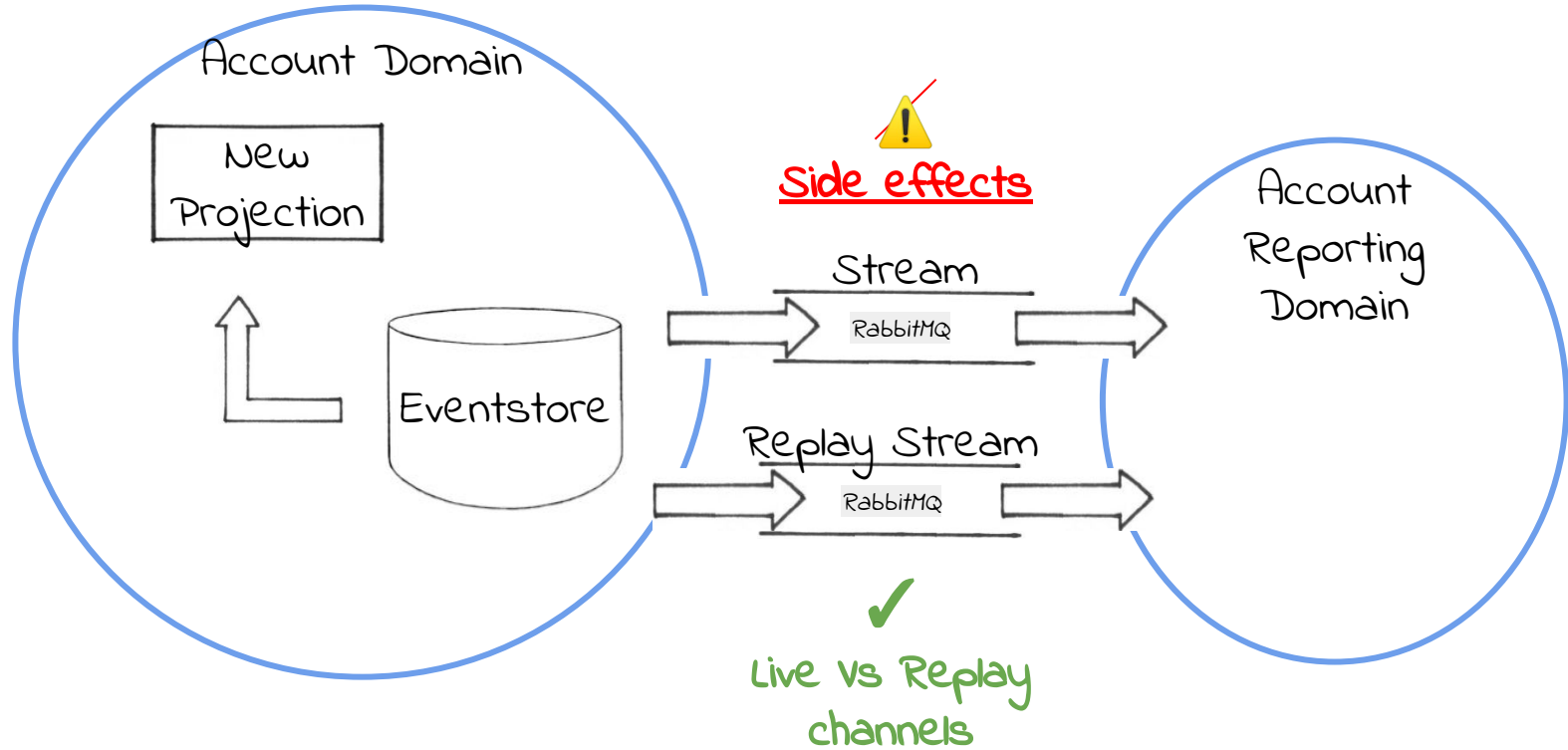
Framework (Axon) Vs No Framework

	Pros	Cons
Framework (Axon)	<p>Rapid implementation</p> <p>Take benefits from the community work</p> <p>Make the tests easier (Axon DSL)</p>	<p>Intrusive (broke hexagonal approach)</p> <p>May inhibit the architecture understanding</p>
No Framework	<p>Better understanding of the implementation and how it works</p> <p>Increases the concepts appropriation</p>	<p>Reinvent the wheel</p> <p>Fall into known traps</p>

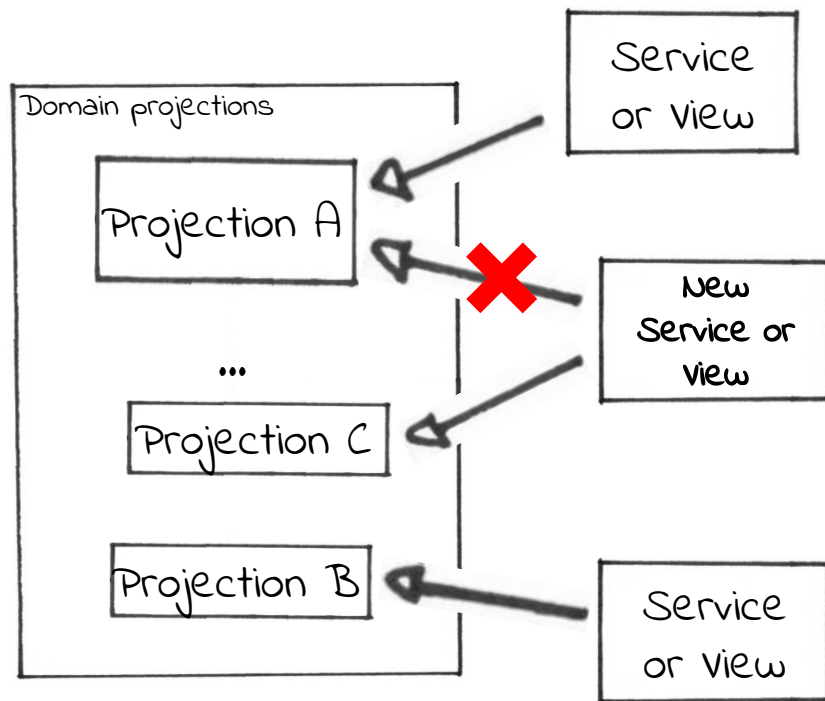
Domain Events : Internal Vs External



Replay



Projections



Monitoring

Infra

Machines/VMs/nodes

Topics/Queues

Databases/Stores

...

Business services

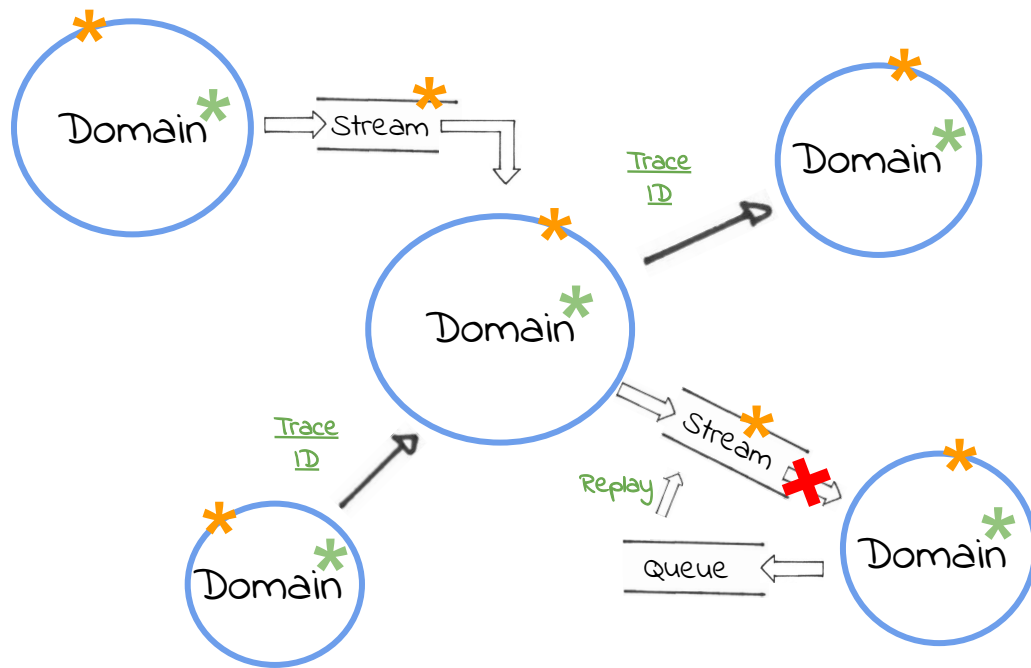
Services Health

Failed event/messages

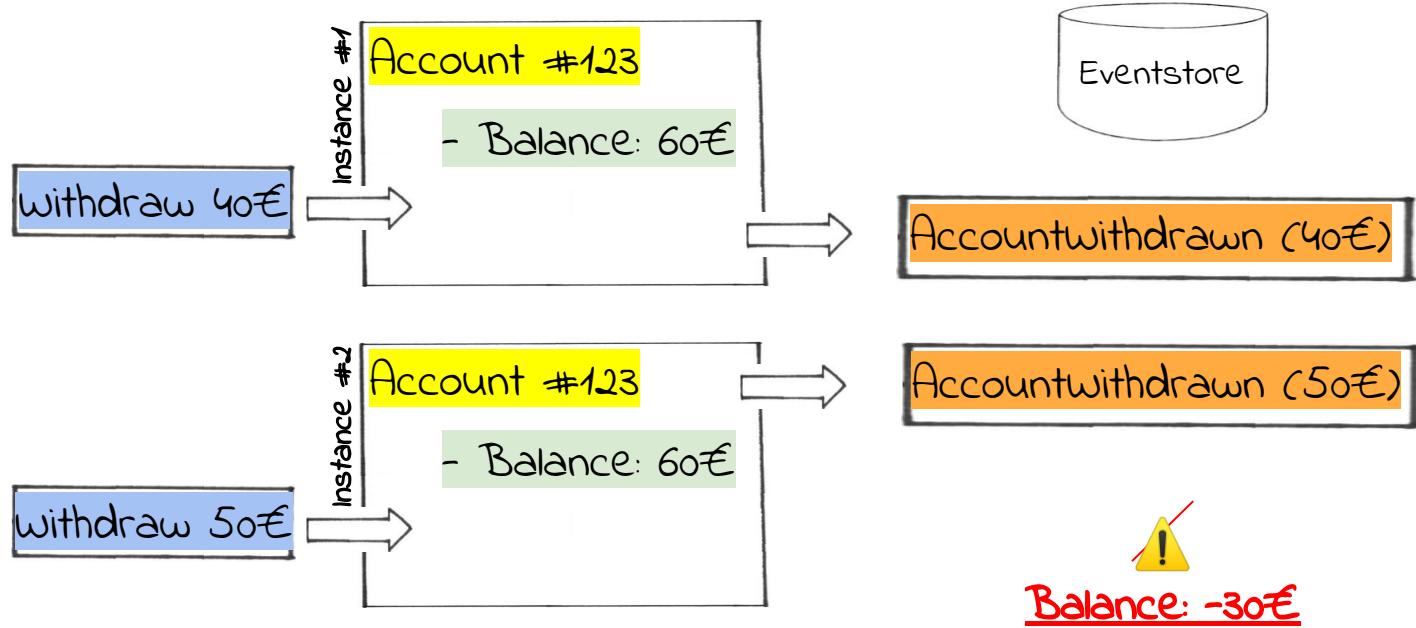
Replay

Distributed Tracing

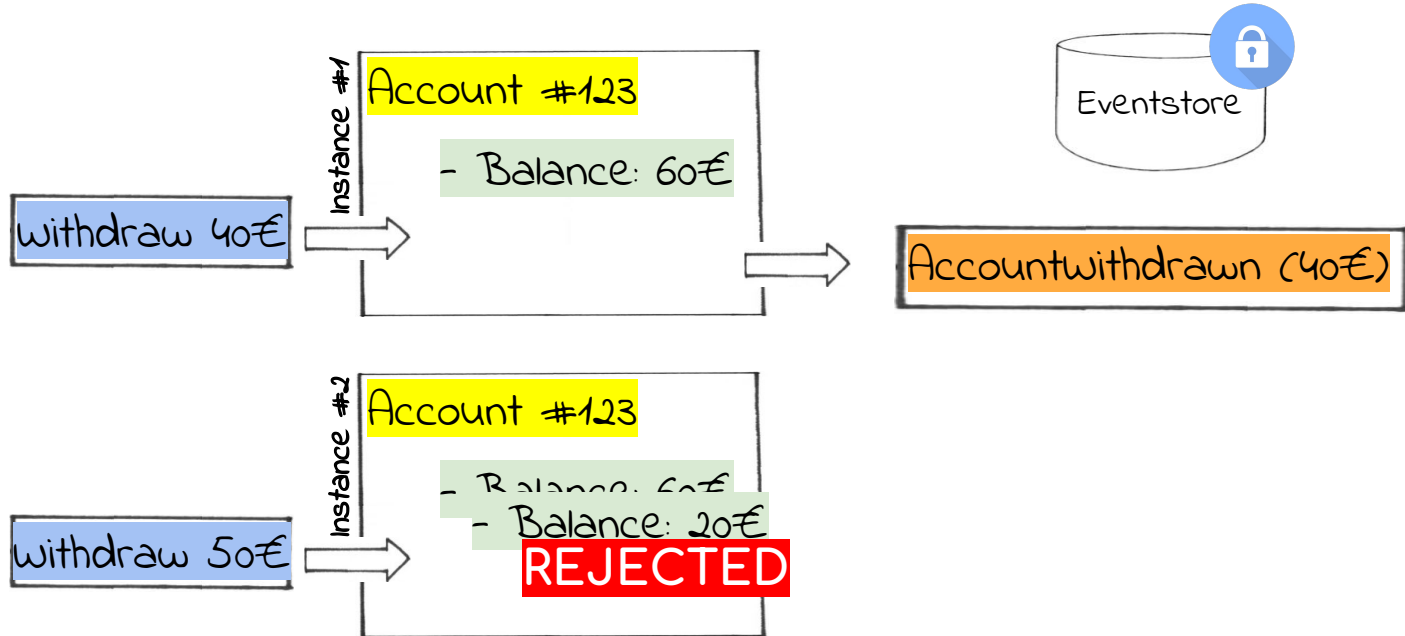
...



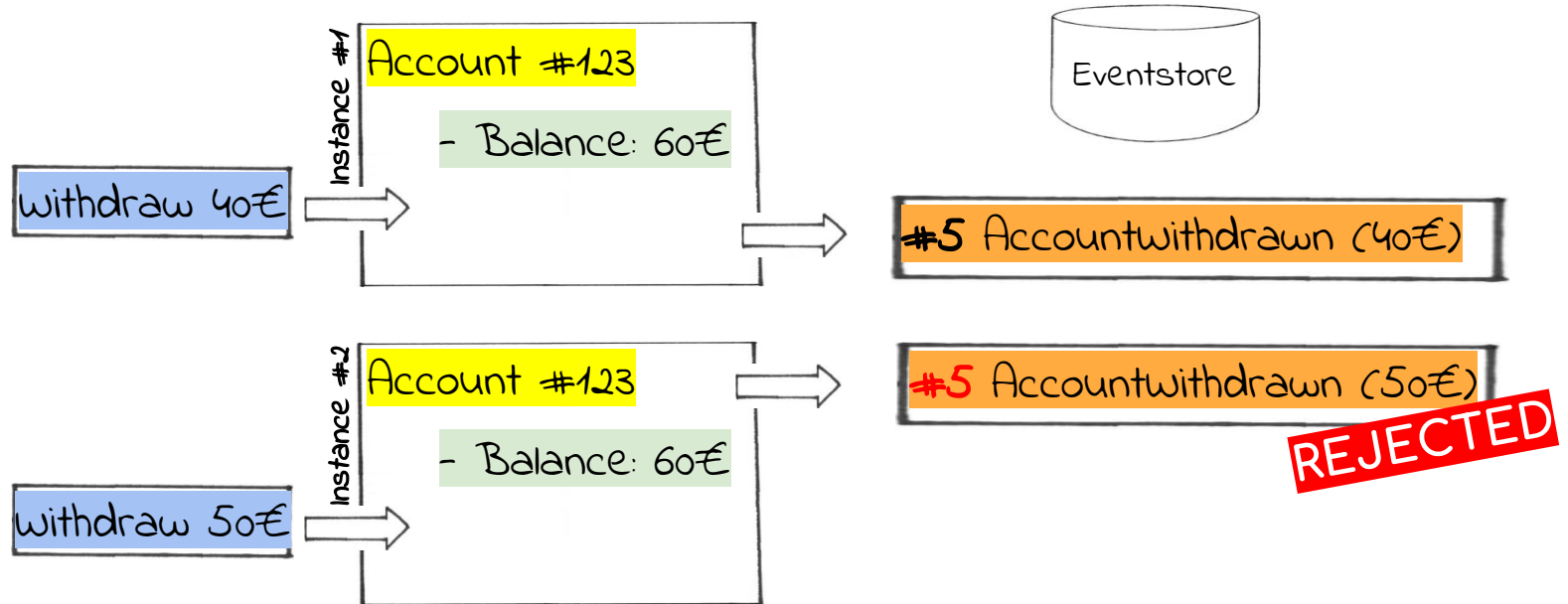
Scaling : Why it matters ?



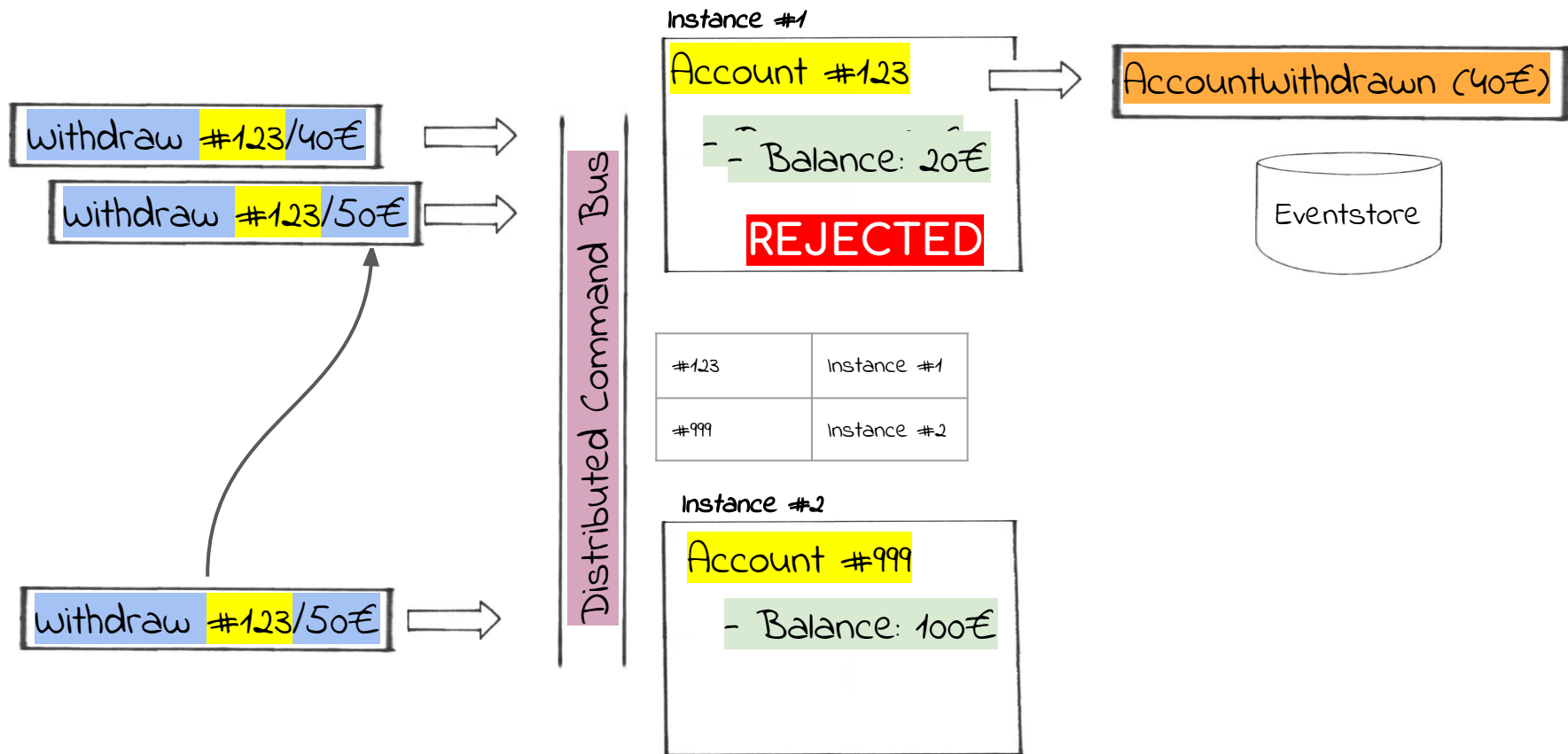
Scaling : Pessimistic Lock



Scaling : Optimistic Lock



Scaling : Multi-Instances with Axon/JGroups

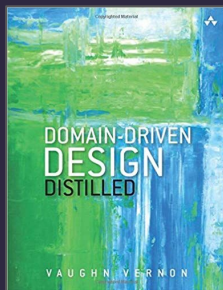
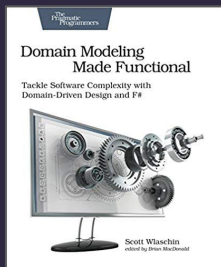
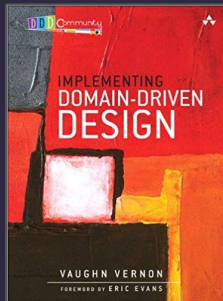
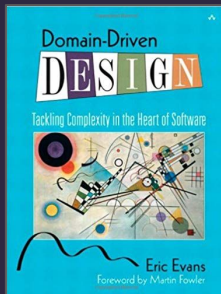


Finally

Fit with any domain but depending on what really needed!	Keep it simple!	Always challenge the architecture	Require a good understanding of the global system
Monitor & document the way your system is working	May be complex	Require Business & IT collaboration	Fix and refactor continuously
Revert if needed!	Keep learning	Pay attention on namings and responsibilities	Do it iteratively

Thank you!





The anatomy of Domain Event

<https://blog.arkency.com/2016/05/the-anatomy-of-domain-event/>

1 Year of Event Sourcing and CQRS

<https://hackernoon.com/1-year-of-event-sourcing-and-cqrs-fb9033ccd1c6>

Some CQRS and Event Sourcing Pitfalls

<https://dzone.com/articles/some-cqrs-and-event-sourcing-pitfalls>

The unfinished guide to Event-Sourcing, CQRS and DDD

<https://medium.com/@vladimirmetnew/the-unfinished-guide-to-event-sourcing-cqrs-and-ddd-unreleased-row-ff798b554837>

Jeremie Chassaing

<https://thinkbeforecoding.com/>



Clement Heliou

https://www.youtube.com/watch?v=zxa4y6eJj_g

Nick Tune

<https://medium.com/@ontcoding>

REX on
Event Sourcing

by  @nizarazu9
 @HaythemZayani