Hands-on Axon Framework

AxonFramework

AxonIQ

Who's who

Steven van Beelen

lead dev Axon Framework

Frans van Buul

evangelist at AxonIQ

Exercises

chat getting started

- Build your first Axon application by completing missing parts.
- Begin here if you don't have previous Axon Framework experience.

chat scaling out

- Scale out your Axon application.
- Two options (which you can do both):
  1. Fully open source, using RabbitMQ and JGroups
  2. Using AxonIQ's AxonHub and AxonDB

Agenda

- Quick introduction Axon Framework

  (optional, but recommended)

- Do the exercises

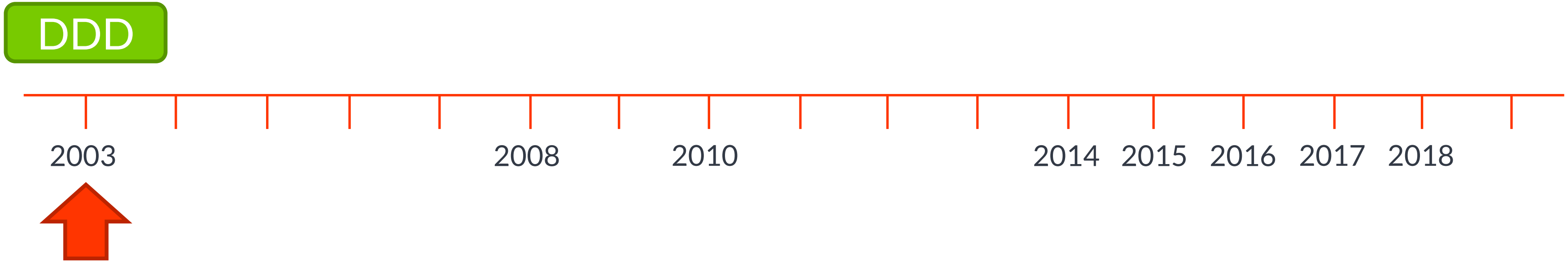
Getting started quickly

- Clone or copy https://github.com/AxonIQ/devoxx-london-labs/
- Have a look at the main README.md
- Choose between "Getting started" and "Scaling out"
- Have a look at the specific README.md in that folder

Note: there are USB sticks which contain a copy of this repository and other dependencies you may need.

2003 2008 2010 2014 2015 2016 2017 2018

History and introductions



DDD

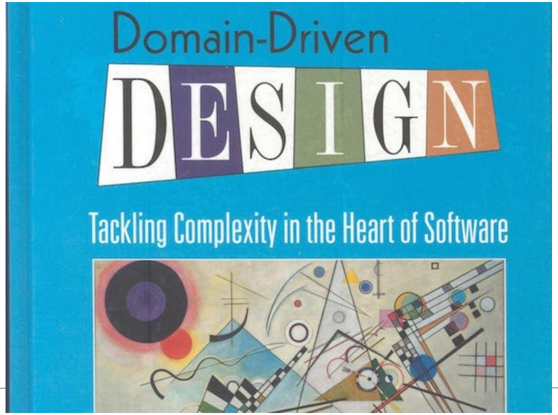
2003

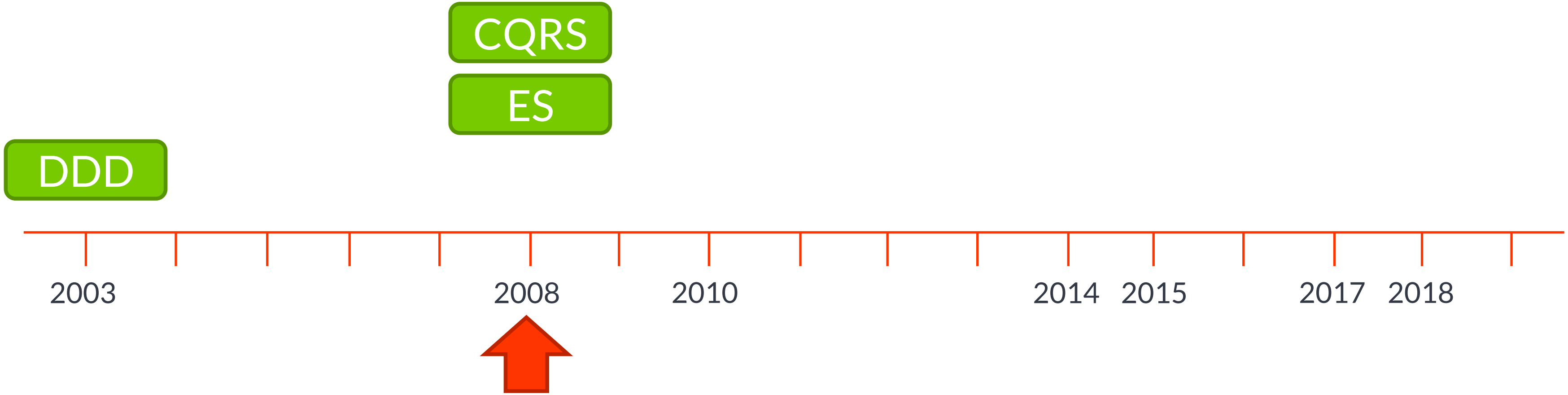
2008

2010

2014 2015 2016 2017 2018

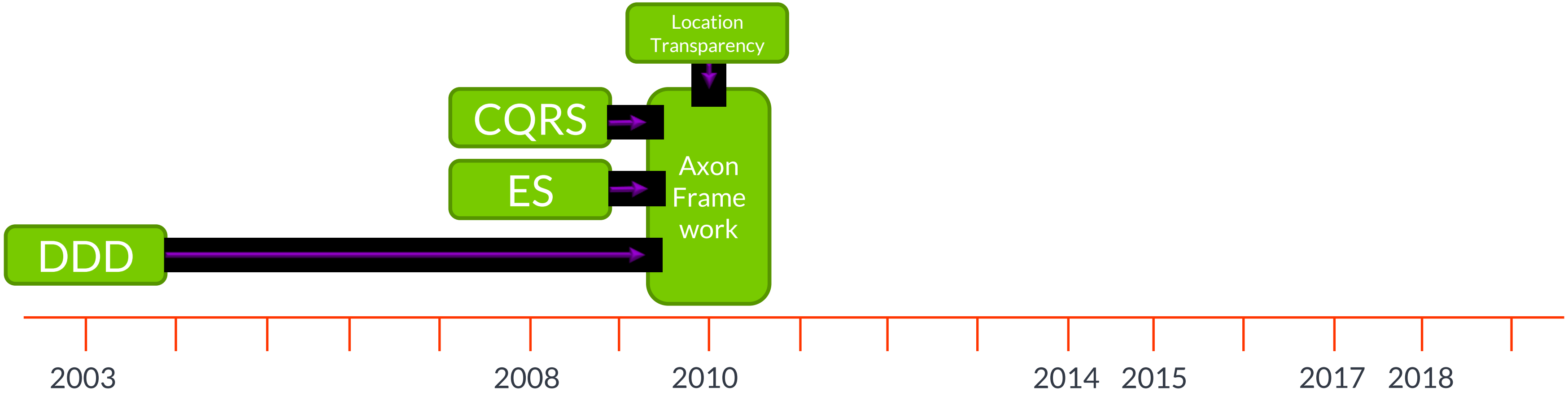
Eric Evans





"Command-Query Responsibility Segregation" (CQRS) coined by Greg Young

Often used in conjunction with "Event Sourcing" (ES)



Location Transparency

CQRS

ES

DDD

Axon Frame work

2003

2008

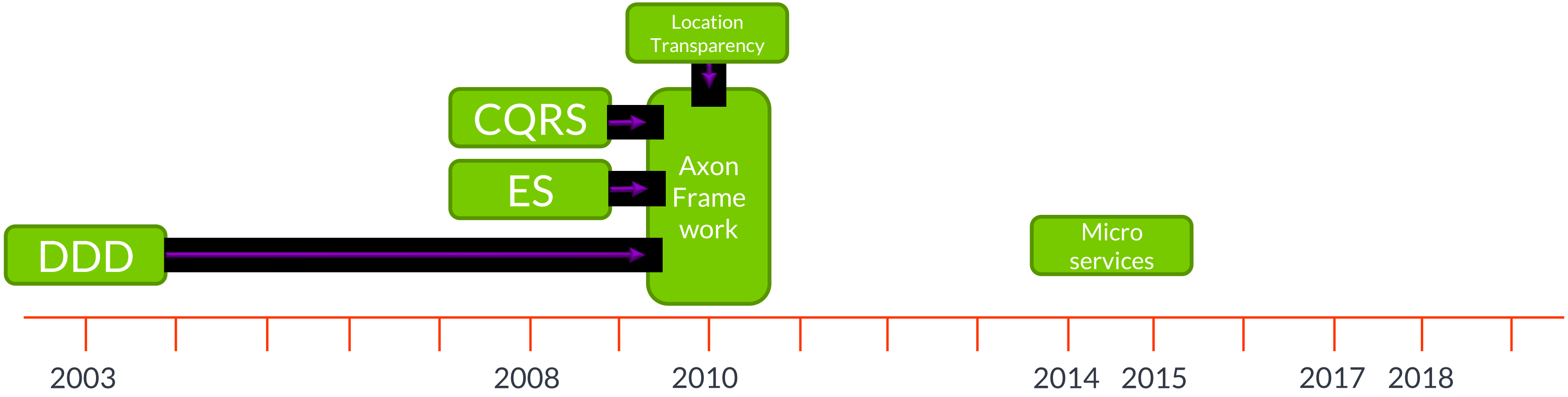
2010

2014 2015

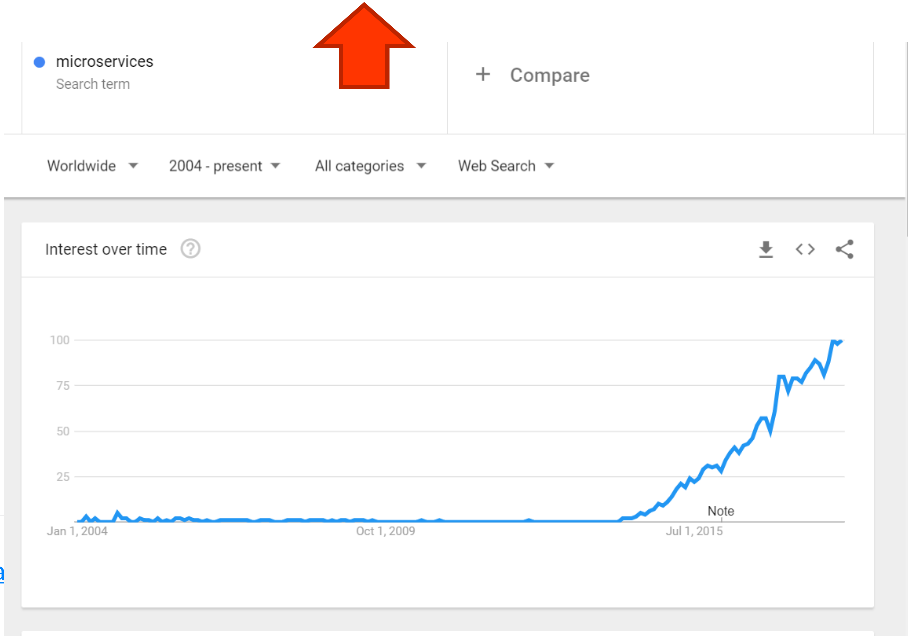
2017 2018

Allard Buijze started development of Axon Framework, CQRS being the core use case.

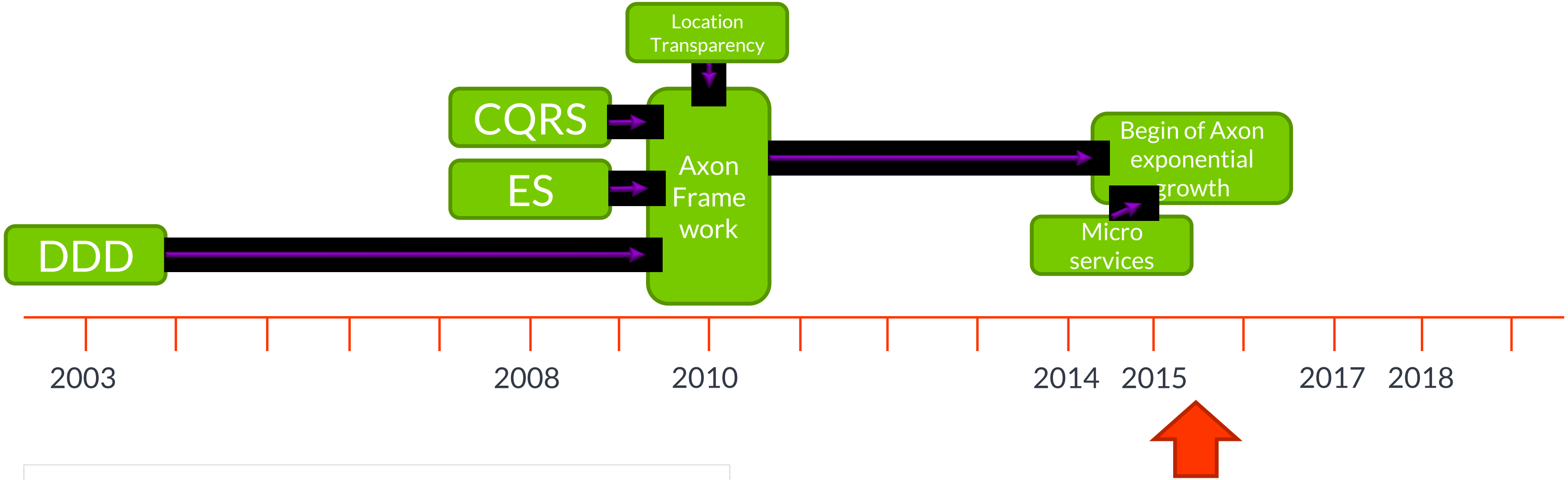
"Location Transparency" added to the core ideas of DDD, CQRS and ES

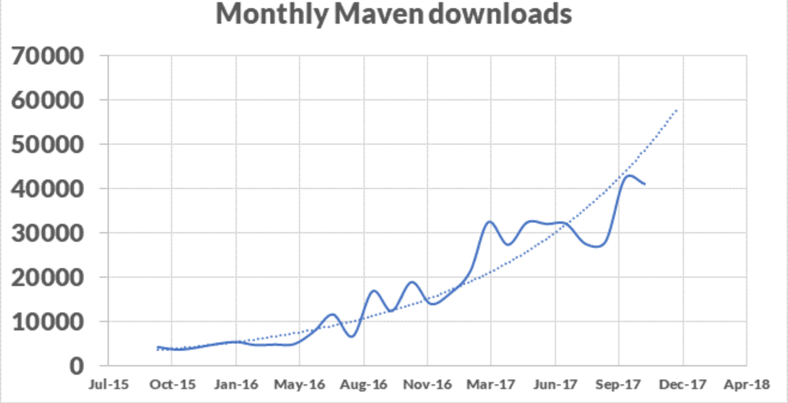


Around 2014/2015,
"microservices" become a topic.



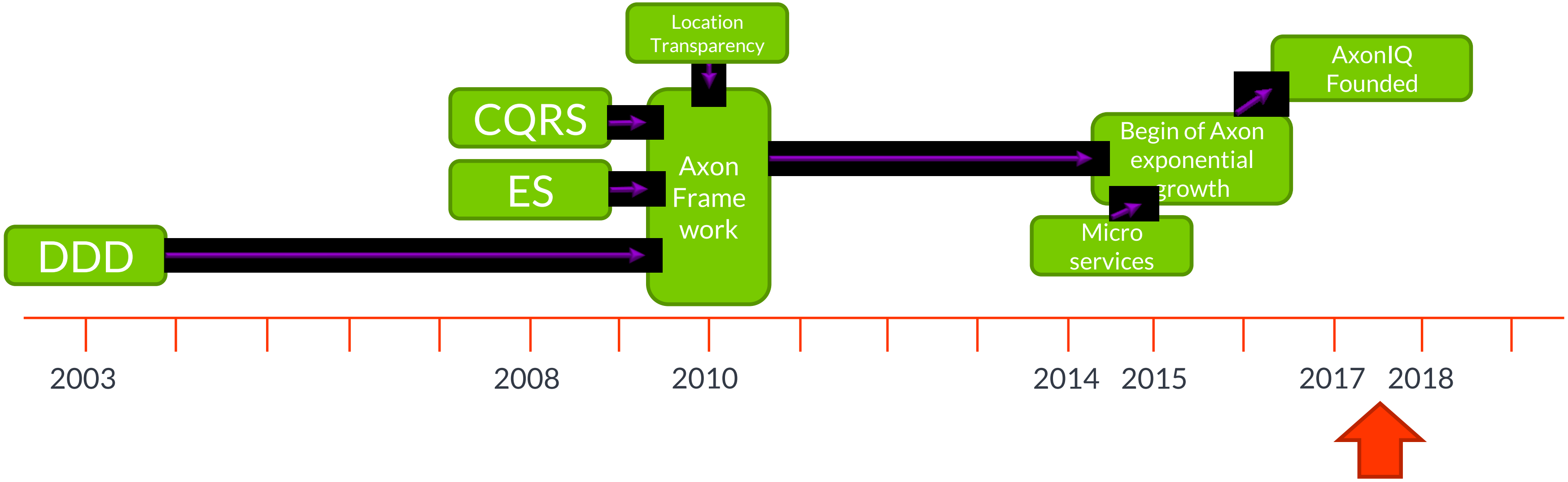
Worldwide ▾ 2004 - present ▾ All categories ▾ Web Search ▾





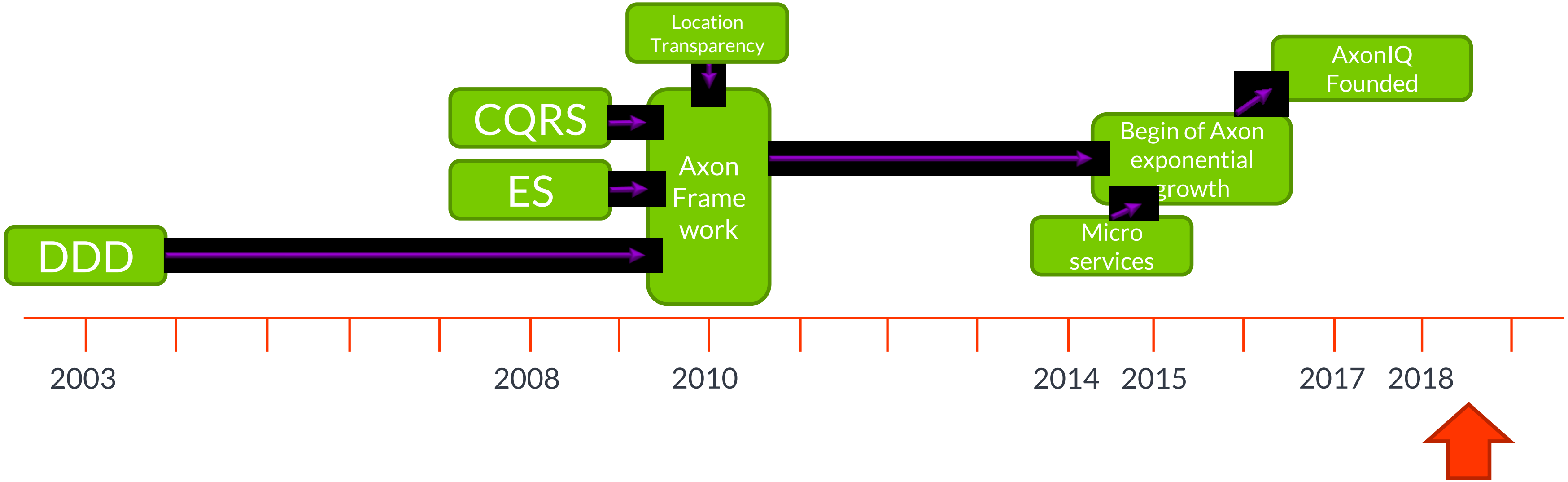
From 2015, Axon usage starts growing exponentially.

DDD, CQRS, ES work great with Microservices!



AxonIQ founded in July 2017:

- to meet increasing demand for Axon services
- to realize a vision on additional Axon products



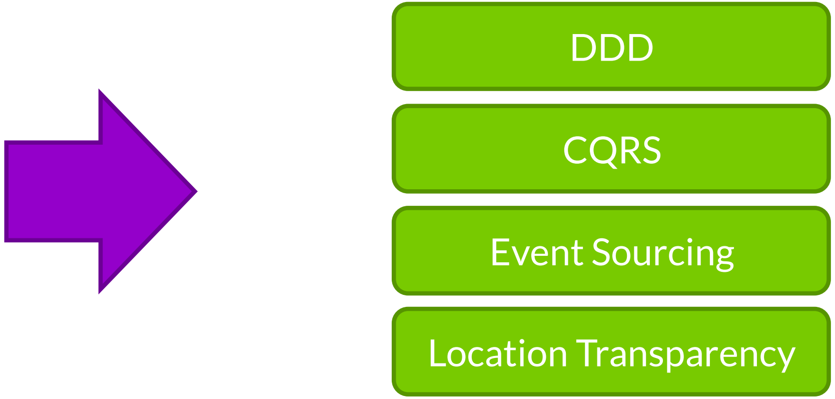
Axon Framework

750k downloads total, 60k per month
50% in last 6 months
Is and will remain open source
(Apache 2 license)

AxonIQ

15 employees
HQ in Amsterdam, also in US, Serbia, Italy
Commercial products: AxonHub, AxonDB, GDPR Module
Support and training services

BARCLAYS TOYOTA SOCIETE GENERALE FedEx ING NOMURA



DDD

CQRS

Event Sourcing

Location Transparency

1. What is it?
2. Why is it useful?
3. What does it look like in Axon?

DDD

CQRS

Event Sourcing

Location Transparency

Domain-Driven Design

- Approach to the software design process.
- Key purpose is to keep complex software manageable.

- Operates at two levels:
  1. **Strategic level**
     Domain model, Ubiquitous Language, Bounded Context, …
  2. **Tactical level**
     Aggregate, Repository, Service, Value Object, Entity, Anti-Corruption Layer, …

DDD

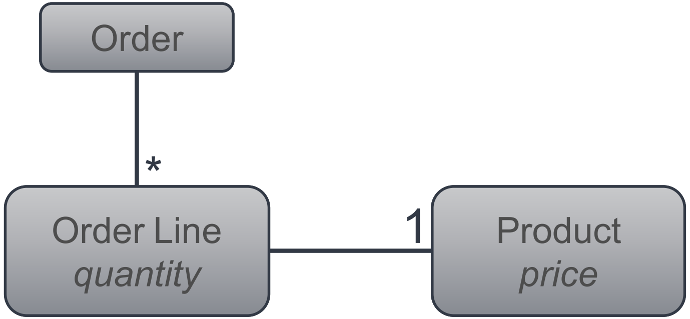
CQRS

Event Sourcing

Location Transparency

Example

Customer Order



DDD

CQRS

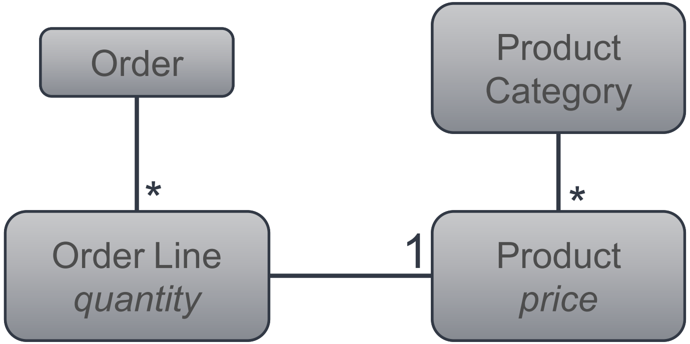
Event Sourcing

Location Transparency

Example

Customer Order perspective

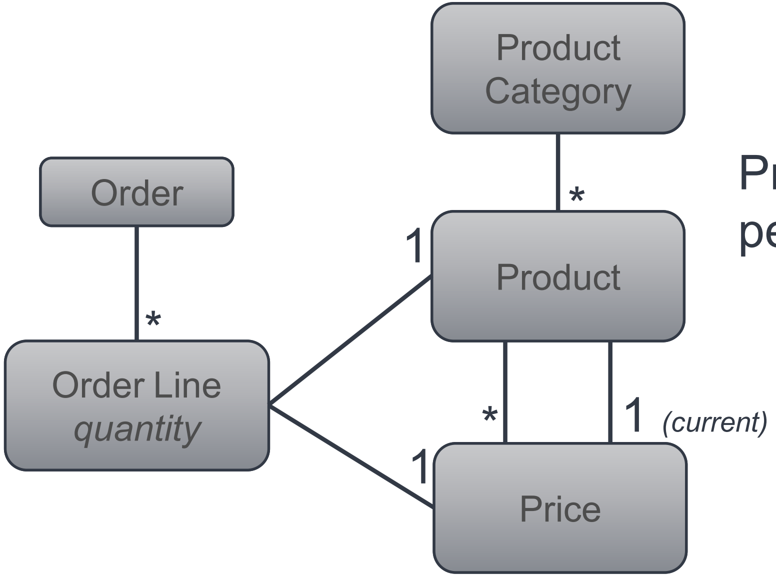
Product catalogue perspective



Example

- DDD
- CQRS
- Event Sourcing
- Location Transparency

Customer Order perspective



Product catalogue perspective

DDD

CQRS

Event Sourcing

Location Transparency

Aggregate

A cluster of associated objects that are treated as a unit for the purpose of data changes.

Source: Evans, DDD

DDD

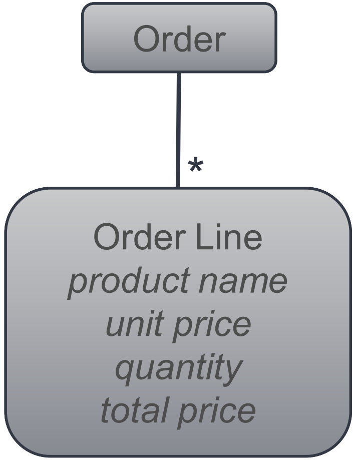
CQRS

Event Sourcing

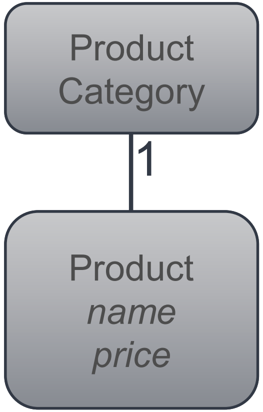
Location Transparency

Revisiting the example

Order aggregate



Orders and Products are separated **aggregates**



Product aggregate

DDD

CQRS

Event Sourcing

Location Transparency

In Axon Framework:

- Aggregates are setup with the the @Aggregate annotation. Axon will inspect handlers, set up a repository etc.
- An @AggregateIdentifier field is needed.
- Business logic code should generally be *in* the Aggregate itself. In DDD, you avoid *anemic* domain models (Martin Fowler). No "service layer" vs. "data layer".

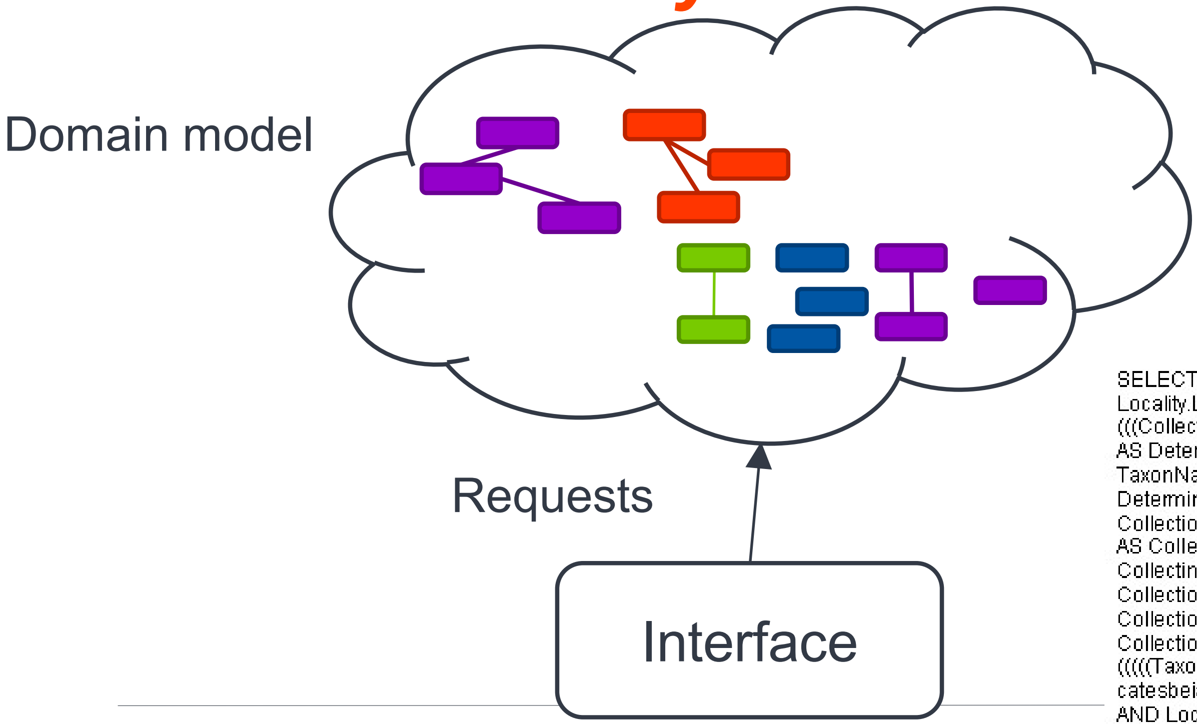
DDD

CQRS

Event Sourcing

Location Transparency

"Normal" system



```
SELECT TaxonName2.FullTaxonName, Locality.Latitude,
Locality.Longitude FROM (CollectionObject) INNER JOIN
(((CollectionObject AS CollectionObject2) INNER JOIN ((Determination
AS Determination2) INNER JOIN (TaxonName AS TaxonName2) ON
TaxonName2.TaxonNameID = Determination2.TaxonNameID) ON
Determination2.BiologicalObjectID =
CollectionObject2.CollectionObjectID) INNER JOIN ((CollectingEvent
AS CollectingEvent2) INNER JOIN (Locality) ON Locality.LocalityID =
CollectingEvent2.LocalityID) ON CollectingEvent2.CollectingEventID =
CollectionObject2.CollectingEventID) ON
CollectionObject2.CollectionObjectID =
CollectionObject.CollectionObjectID  WHERE
(((((TaxonName2.FullTaxonName IN('Rana blairi', 'Rana
catesbeiana'))))) AND (((Locality.Latitude BETWEEN 37.8 AND 39.4
AND Locality.Longitude BETWEEN 94.8 AND 96.0))))
```

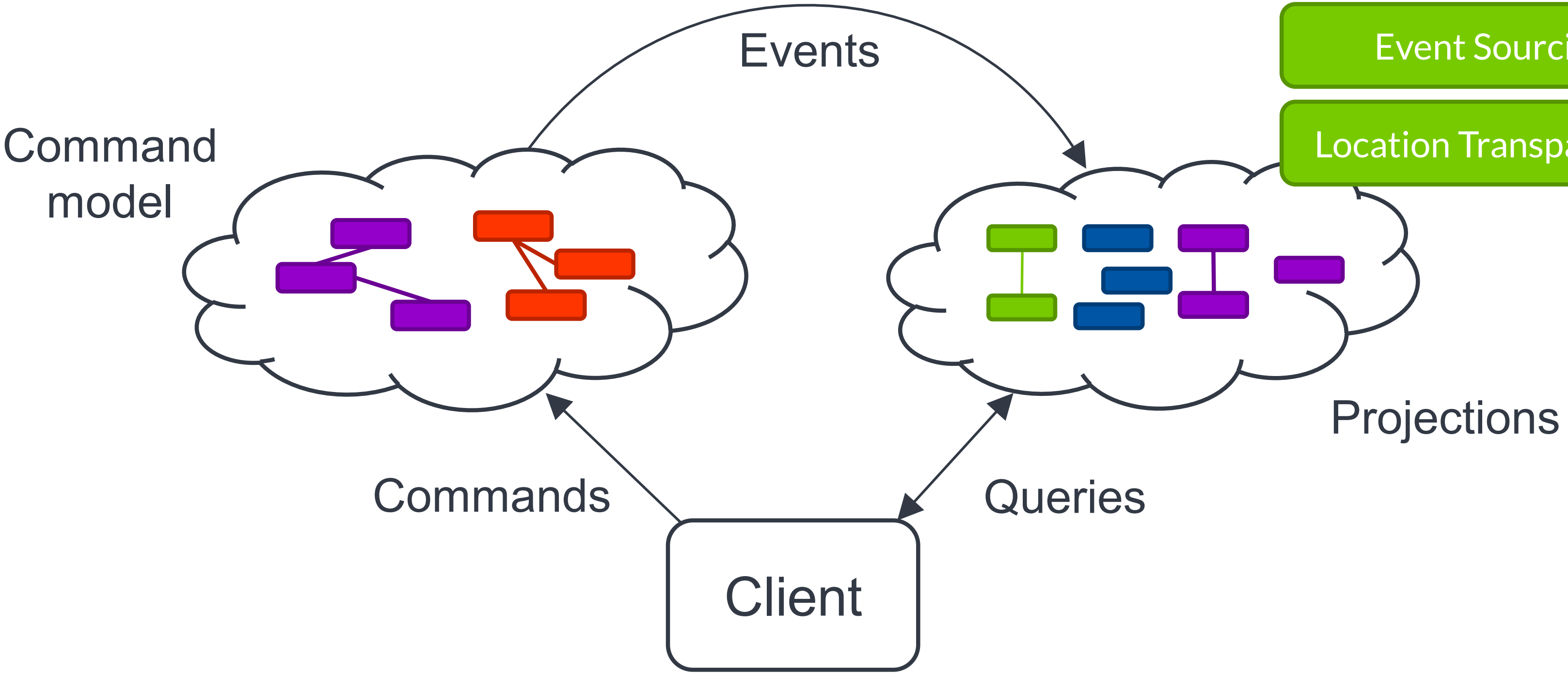
Command Query Responsibility Segregation

DDD

CQRS

Event Sourcing

Location Transparency



DDD

CQRS

Event Sourcing

Location Transparency

In Axon Framework:

- @CommandHandler methods on the write side – typically on an @Aggregate class.
- Only keep data in an @Aggregate that you need for validating commands.
- @QueryHandler methods on the read side (new in 3.2)
- @EventHandler methods on the read side to get events from the write side and update the read model accordingly (BTW, this is *not* "event sourcing" yet!)

DDD

CQRS

Event Sourcing

Location Transparency

Event sourcing

Traditional persistence

- Store the current state of an aggregate (cmd side) directly to database (e.g. by using JPA).

Event Sourcing

- All state change through events.
- Events are distributed *and* persisted to an event store.
- To access current state of an aggregate, read all relevant events and *replay* them ("micro-replay").

Why use event sourcing?

- DDD
- CQRS
- Event Sourcing
- Location Transparency

- Fully cover all changes
- Perfect audit trail
- Analytics, machine learning
- Debugging
- Replay into new read models
- Capturing the intent of change

DDD

CQRS

Event Sourcing

Location Transparency

In Axon Framework:

- Event sourcing is the default, and event sourcing repositories will be created automatically.
- Don't do any aggregate state change in @CommandHandler. Instead apply() an event.
- Change state in @EventSourcingHandler methods.
- Don’t do business logic or side effects in the @EventSourcingHandlers.
- Aggregates need the empty constructor.

DDD

CQRS

Event Sourcing

Location Transparency

Location transparency

- Command, Events and Queries seen as messages.
- Components do not know whether they are communicating to a local or remote component – they're sending a message anyway.

Benefits:

- Enables scaling out.
- Enables an evolutionary approach to microservices.

DDD

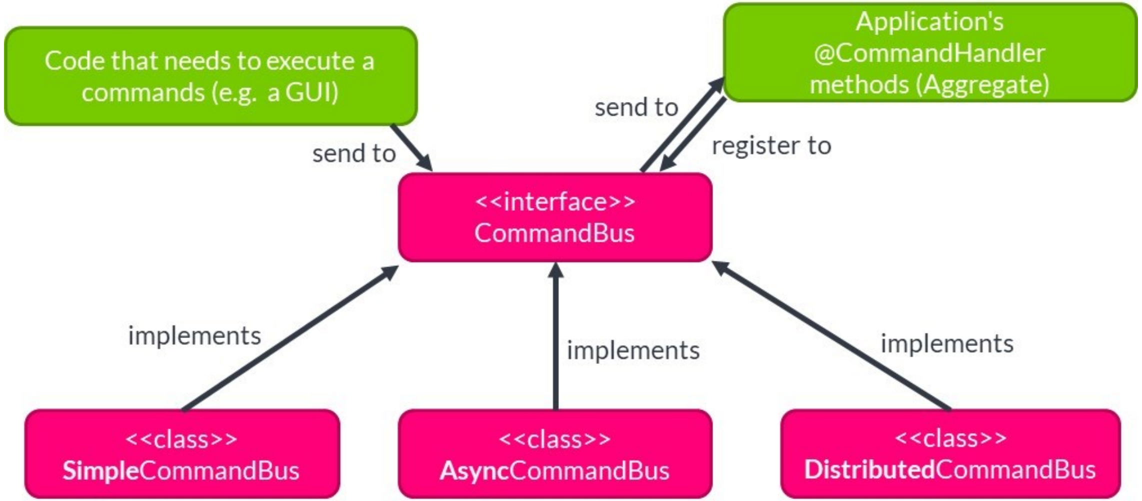
CQRS

Event Sourcing

Location Transparency

Location transparency in Axon

Commands as illustration – Queries and Events work similarly



Code that needs to execute a commands (e.g. a GUI)

send to

Application's @CommandHandler methods (Aggregate)

send to

register to

<<interface>> CommandBus

implements

implements

implements

<<class>> **Simple**CommandBus

<<class>> **Async**CommandBus

<<class>> **Distributed**CommandBus

Also:
DisruptorCommandBus
AxonHubCommandBus

Getting started

- Clone or copy https://github.com/AxonIQ/devoxx-london-lab/
- Have a look at the main README.md
- Choose between “Getting started” and “Scaling out”
- Have a look at the specific README.md in that folder

Note: there are USB sticks which contain a copy of this repository and other dependencies you may need.