ES/CQRS & Axon

Overview

Event Sourcing (ES)

Command Query Responsibility Segregation (CQRS)

Building Blocks

Axon Framework

Hands-On Coding Session

ES & CQRS

Architectural patterns

Orthogonal concepts

- playing well together
- don’t depend on each other

Paradigm shift in both

- Stream of events (ES)
- Read/Write models (CQRS)

Event Sourcing

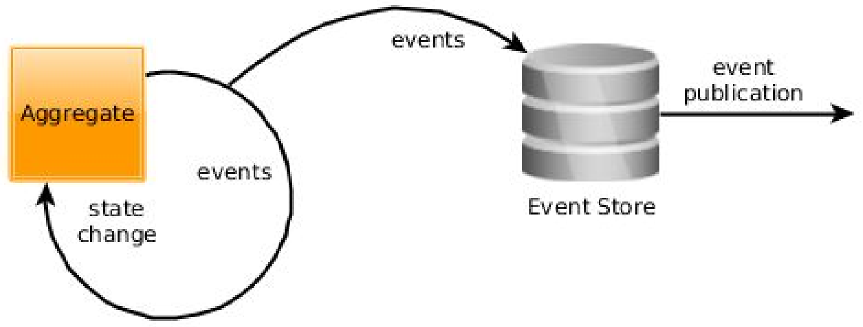
Classical architecture

storing snapshot of current applications state in data store

Event sourcing

all changes to application state stored as stream of events

Event Sourcing



Aggregate: “cluster of associated objects that we treat as a unit for the purpose of change” -- Evans

Event Sourcing (Pros)

Additional business value

Storing events: append-only operation

O/R impedance mismatch removed

Temporal queries

Audit Log for free

Event Sourcing (Cons)

Restoring state (performance)

    solved by Snapshotting or Read model

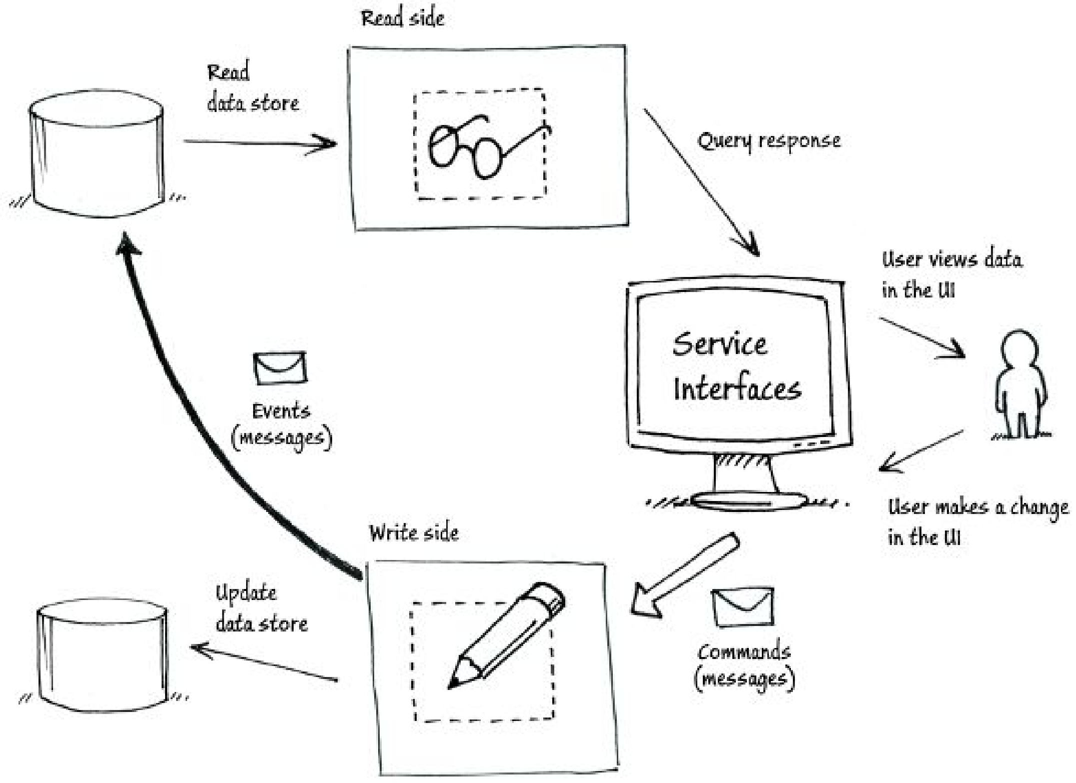
Not suited for complex queries

Evolving models & code changes

Side effects on External Systems

CQRS

CQS:	divide an object's methods into Queries & Commands (class/component level)
CQRS:	“is simply the creation of two objects where there was previously only one” — Greg Young (bounded context level)


Read side
Read data store
Query response
User views data in the UI
Service Interfaces
User makes a change in the UI
Events (messages)
Write side
Update data store
Commands (messages)

CQRS

Scalability
- imbalance between number of reads and number of writes
- performance optimizations

Reduced Complexity
- separating business logic for writes from query logic
- changes to read logic without any impact to business logic

Why ES/CQRS?

Strategic advantages for

- complex domain models (behaviour above state)
- long living applications with many expected changes

Gaining additional business value from historic events

Auditing / temporal queries

Building Blocks

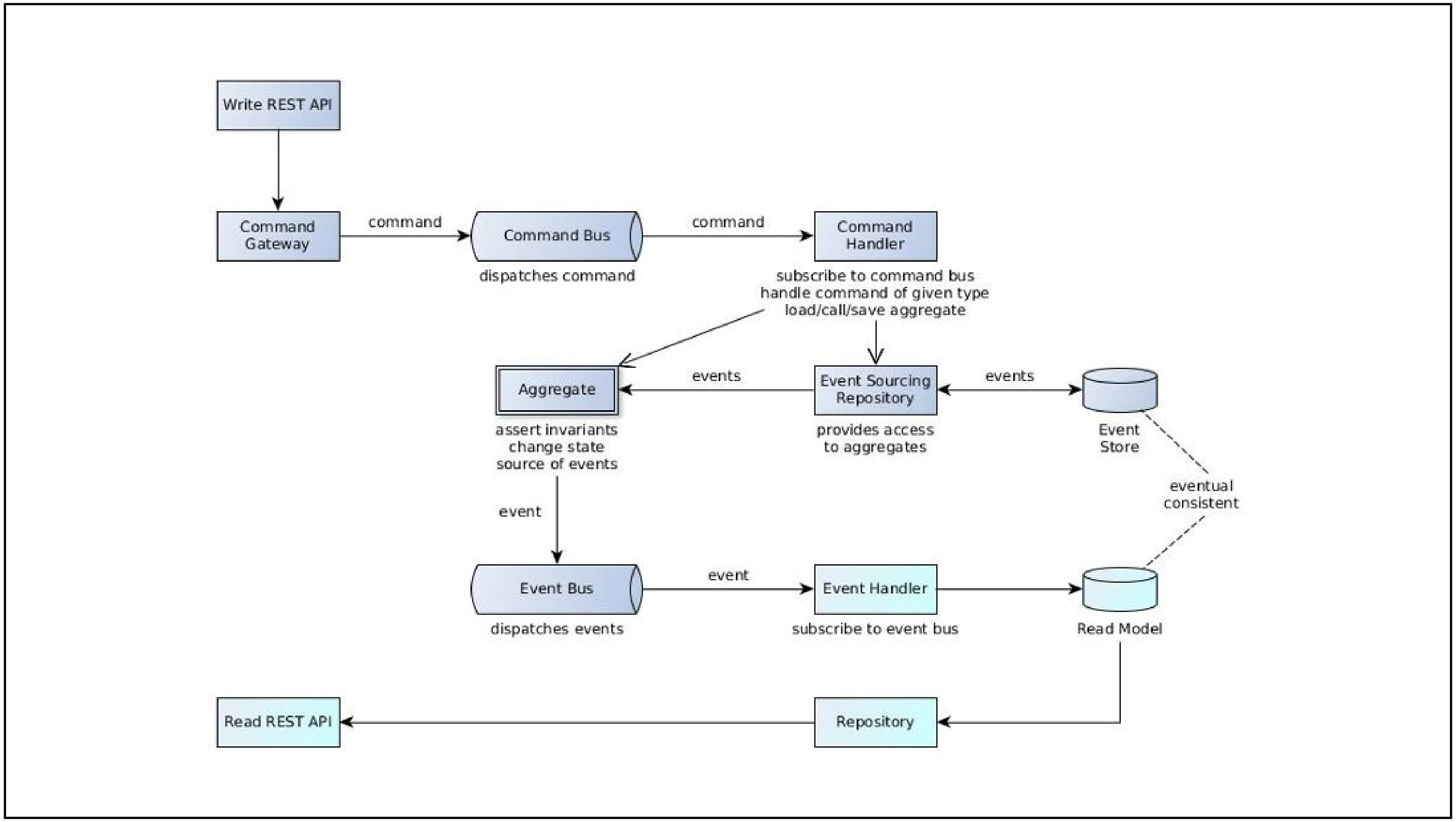
Command Handler, Command, Aggregate

Event Processor, Event Handler, Event

Command Bus, Event Bus

Event Store, Repository

Interceptors, ...


Write REST API
Command Gateway
command
Command Bus
dispatches command
command
Command Handler
subscribe to command bus
handle command of given type
load/call/save aggregate
Aggregate
events
Event Sourcing Repository
events
Event Store
provides access to aggregates
assert invariants
change state
source of events
eventual consistent
event
Event Bus
dispatches events
event
Event Handler
subscribe to event bus
Read Model
Read REST API
Repository

Axon Framework

Framework for scalable, high performance applications

Supports developers to apply CQRS

Provides most important building blocks

Annotation support

Support for Spring Boot AutoConfiguration

Coding Session

Domain: agile project
Aggregates: sprint, backlog item

Exercise: implement the following requirements

1. create a sprint (already done)
2. create a backlog item
3. commit a backlog item to a sprint
4. before committing: revoke previous commit
5. Tests (with Axon Test module)