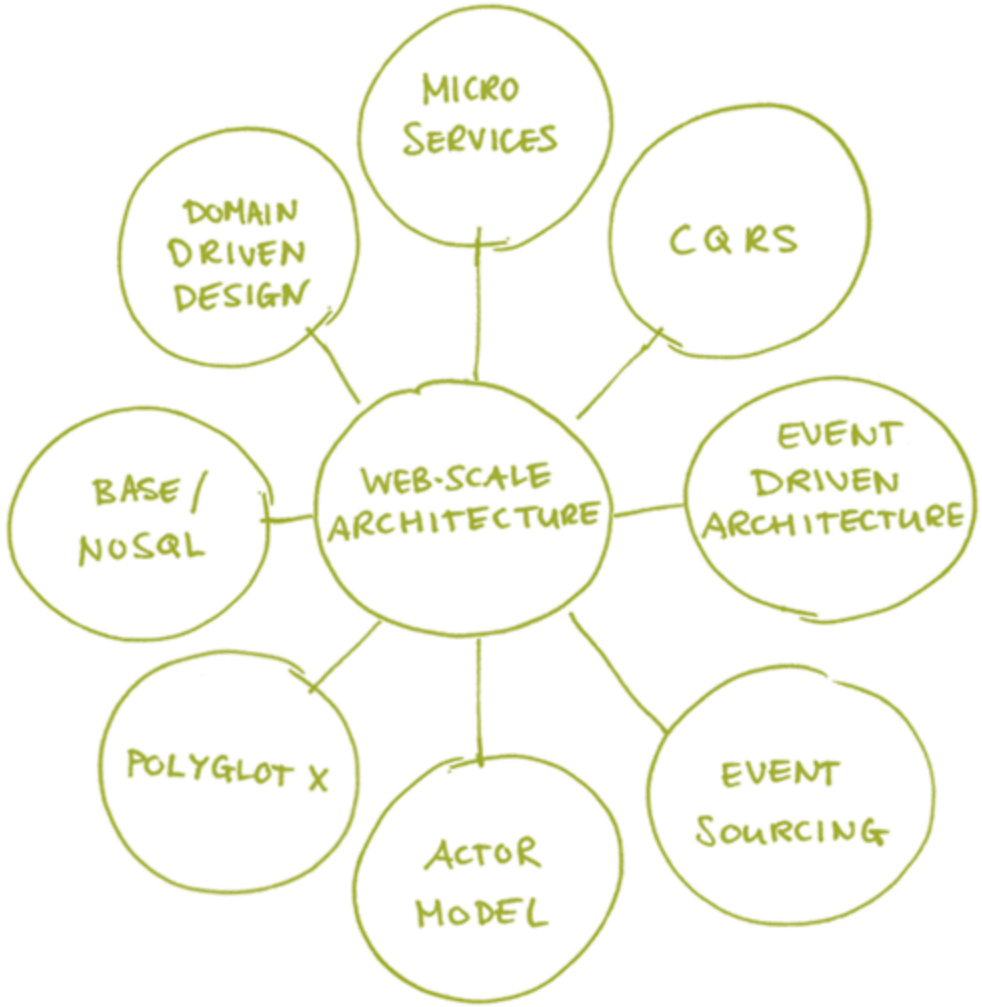
Web-scale architecture

All cool stuff!

Solid Innovator

Recap


MICRO SERVICES
DOMAIN DRIVEN DESIGN
CQRS
BASE / NOSQL
WEB-SCALE ARCHITECTURE
EVENT DRIVEN ARCHITECTURE
POLYGLOT X
ACTOR MODEL
EVENT SOURCING

Four Event-driven architectures

Four forms of event-driven architectures

- Event notification
- Event-carried State Transfer
- Event Sourcing
- CQRS

-Martin Fowler (GOTO 2017; https://www.youtube.com/watch?v=STKCRSUsyP0)
https://martinfowler.com/articles/201701-event-driven.html

Event notification

(something has changed)

+ decouple receiver from sender

- no statement of overall behavior

Event-carried state transfer

(this particular thing has changed)

+ Even more decoupling
+ reduced load on supplier
- replicated data
- eventually consistency

Event sourcing

+ Audit
+ Debugging
+ Historic state
+ Alternative state
+ Memory Image

- Unfamiliar
- External systems
- Event Schema (changes)
- identifiers
- Asynchrony?
- Versioning?

CQRS

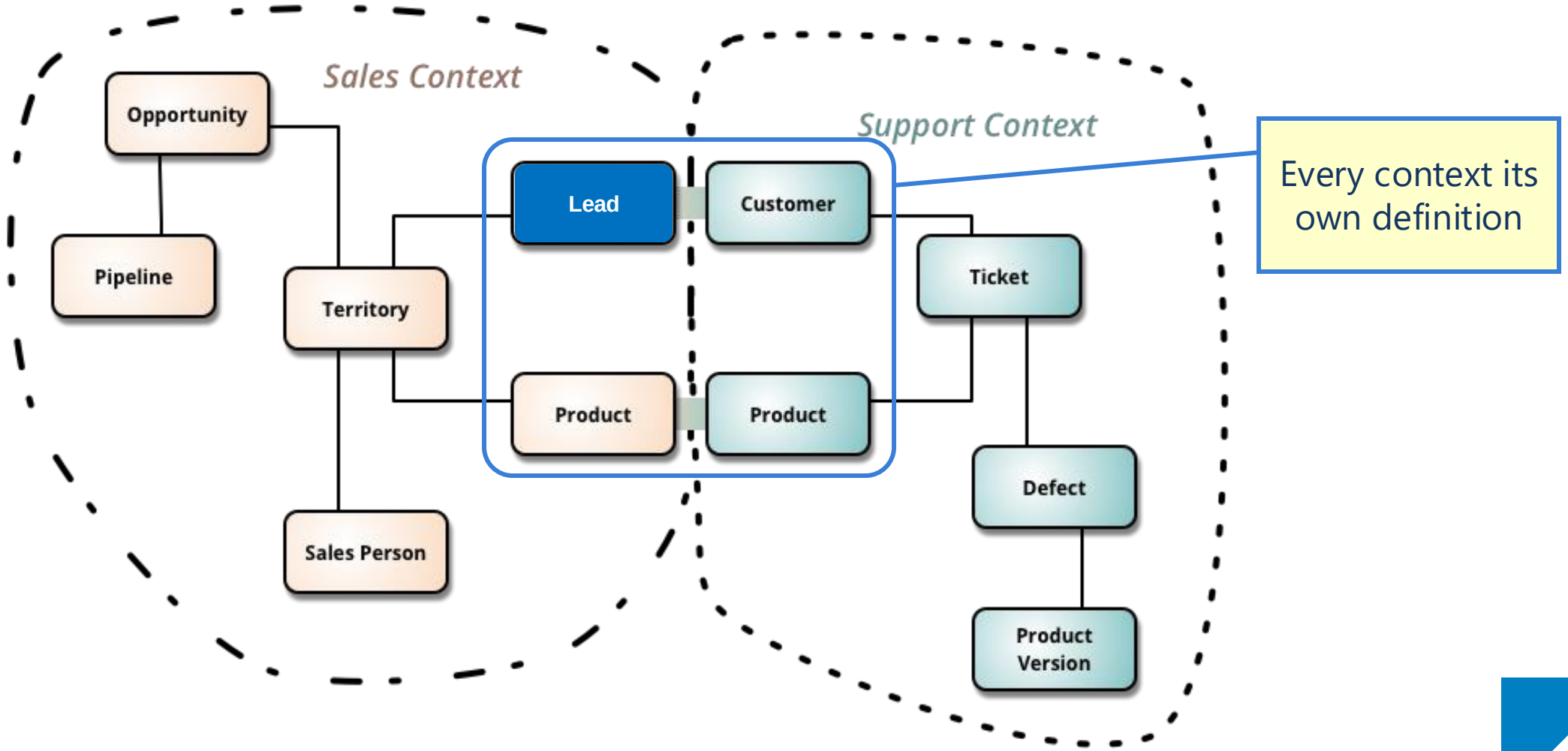
- Martin Fowler says: do not use this much

Autonomy over Authority

Autonomy over Authority

- **Sharing data** between BCs / services is not evil (if done right!)
- An autonomous service and team can **deliver more value**
- Can drastically **reduce chatty service-interactions**
- Can drastically improve **availability**
- Can improve **cloud readiness**
- Can be used for **BI / Reporting**

"Local" domain-model definition per context



http://martinfowler.com/bliki/BoundedContext.html

Autonomy over Authority

Customer Management

CustomerRegistered

string CustomerId;
string Name;
string Address;
string PostalCode;
string City;
string TelephoneNumber;
string EmailAddress;

Workshop Management

CustomerRegistered

string CustomerId;
string Name;
string TelephoneNumber;

Notifications

CustomerRegistered

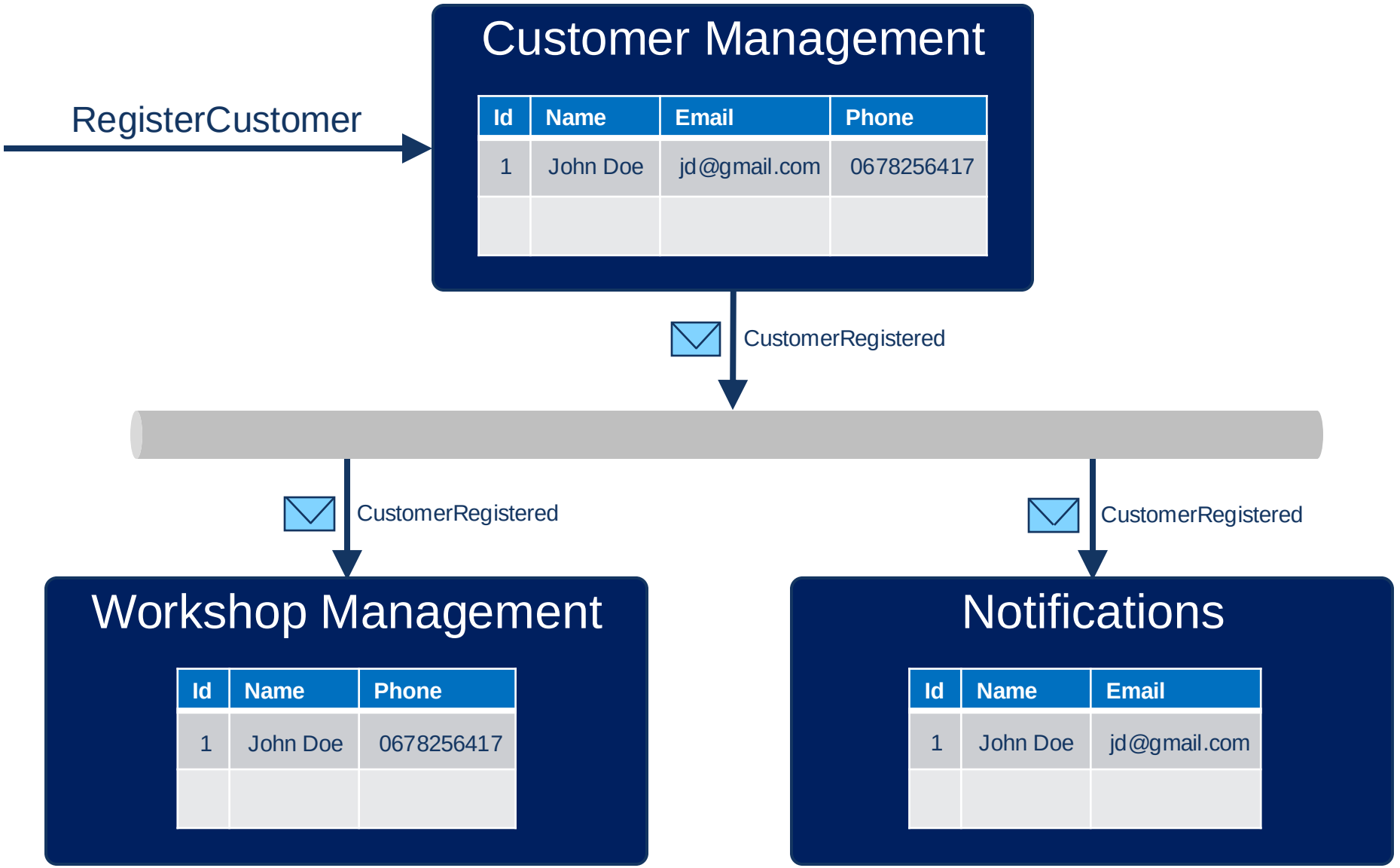
string CustomerId;
string Name;
string EmailAddress;

Invoicing

CustomerRegistered

string CustomerId;
string Name;
string Address;
string PostalCode;
string City;

Autonomy over Authority



Autonomy over Authority principles

- **Less == more!**
- Shared data is always a **read-only cache**
- Make sure you know the **maximum staleness-period** of the data
- Share data using **ETL or Events** (or both)
- Make sure you can **detect and handle missed events**

Eventual consistency

Eventual Consistency

- For distributed systems the **CAP theorem** applies

- **C**onsistency
  - All nodes in the system see the same data at a certain moment in time

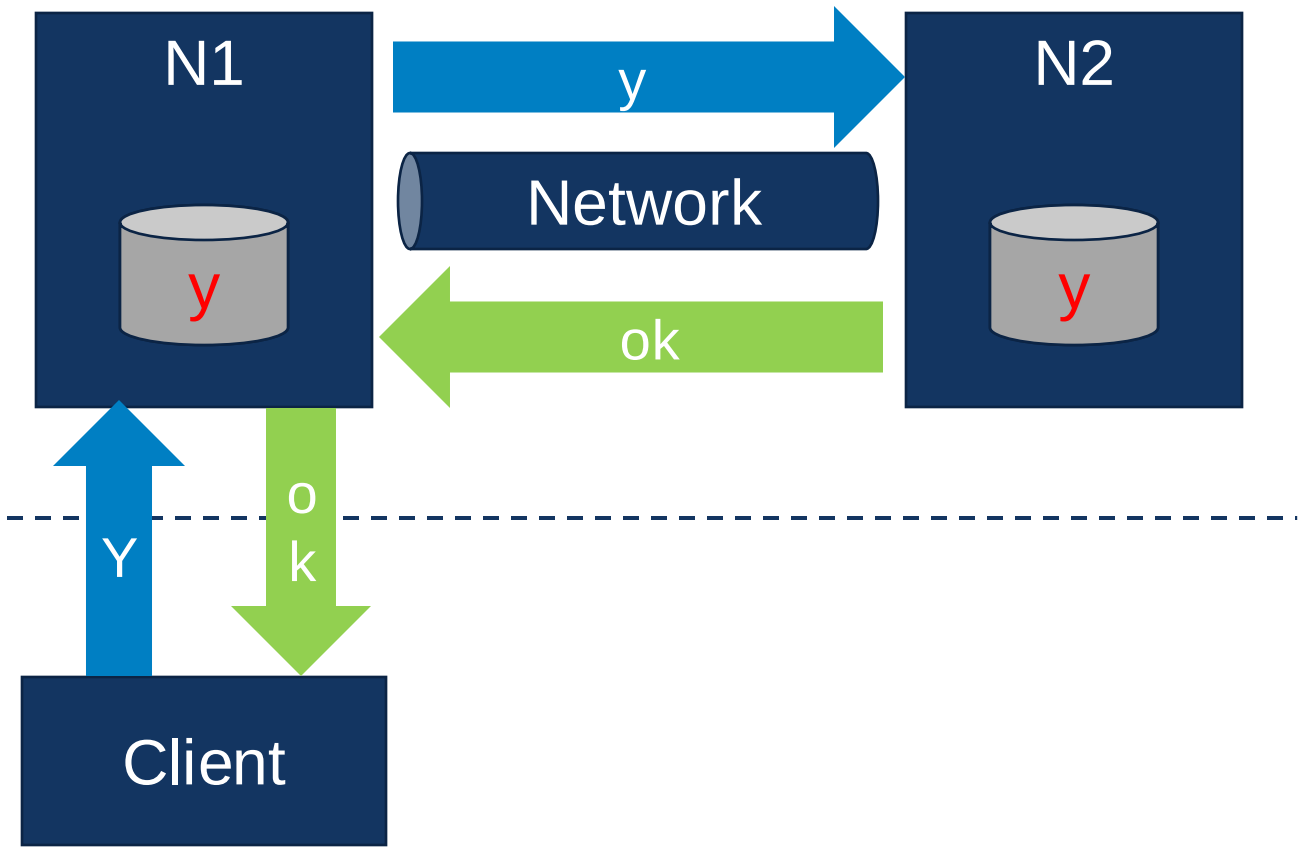
- **A**vailability
  - A node will always return a useful response (no exception or time-out)

- **P**artition Tolerance
  - The system gracefully handles broken connection between nodes in a system (network failure / crash / …)

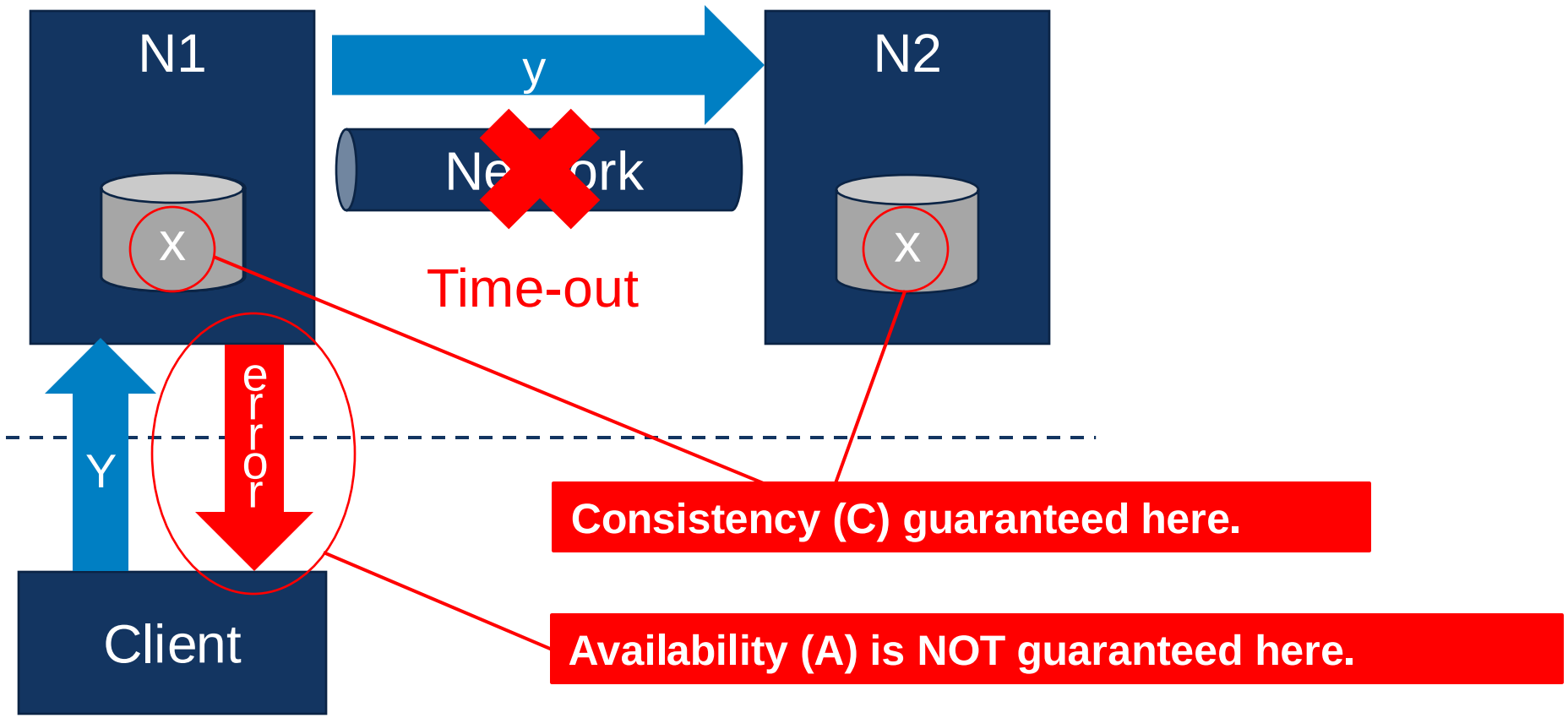
CAP theorem

- According to the CAP theorem, in a distributed system its only possible to adhere to two conditions at the same time - not to all three
- Since networks are not reliable by nature, we MUST be “partition tolerant” (**P**)
- So we need to choose for either consistency (**C + P)** or availability (**A + P)**

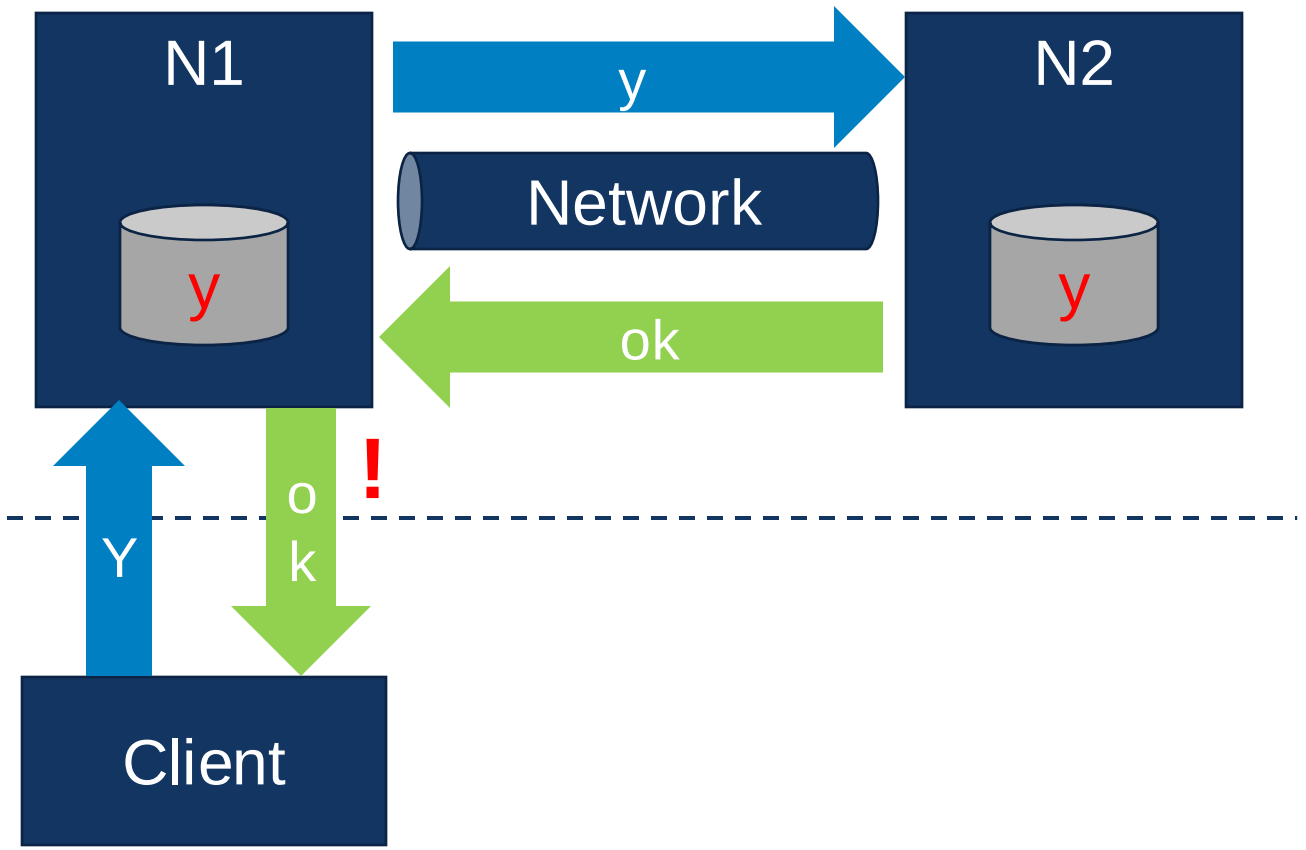
Consistency



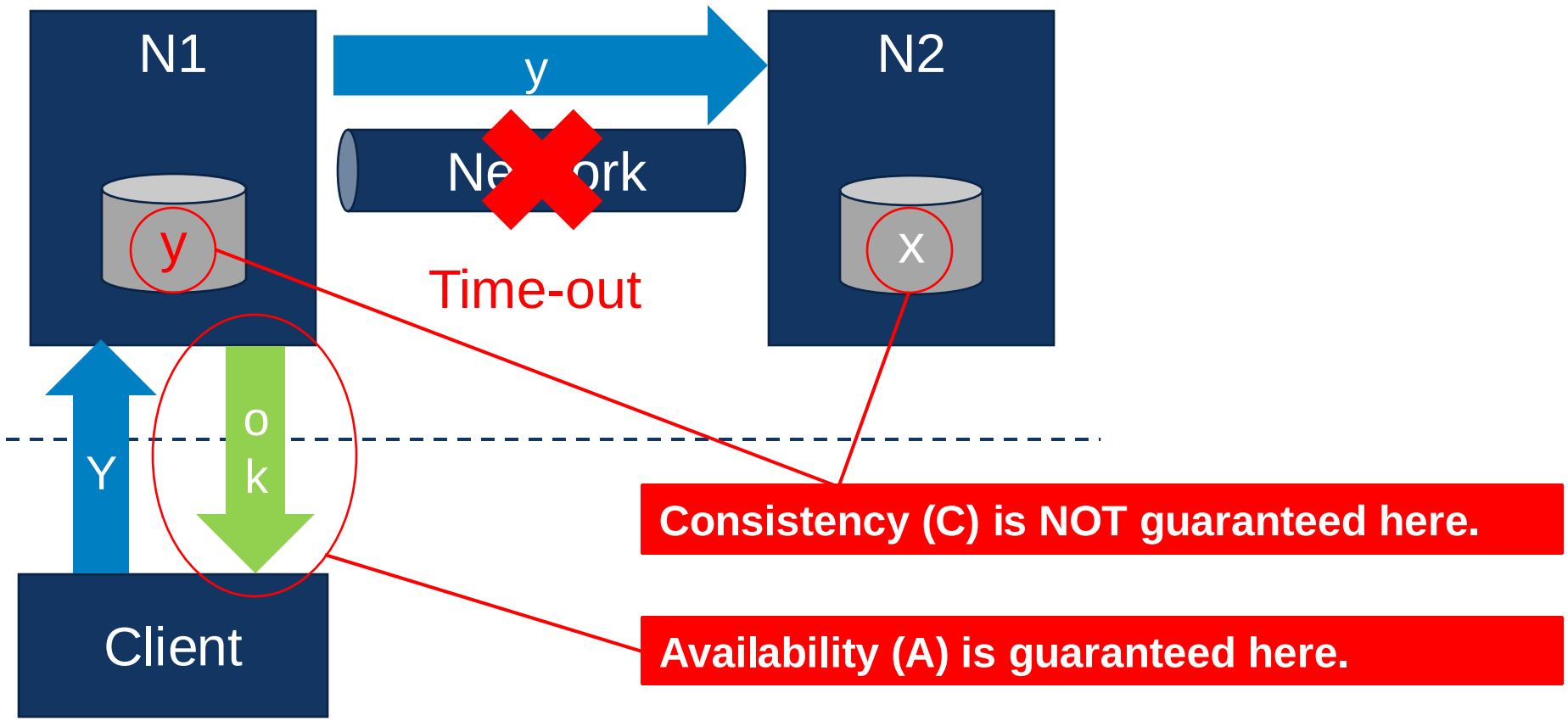
Consistency



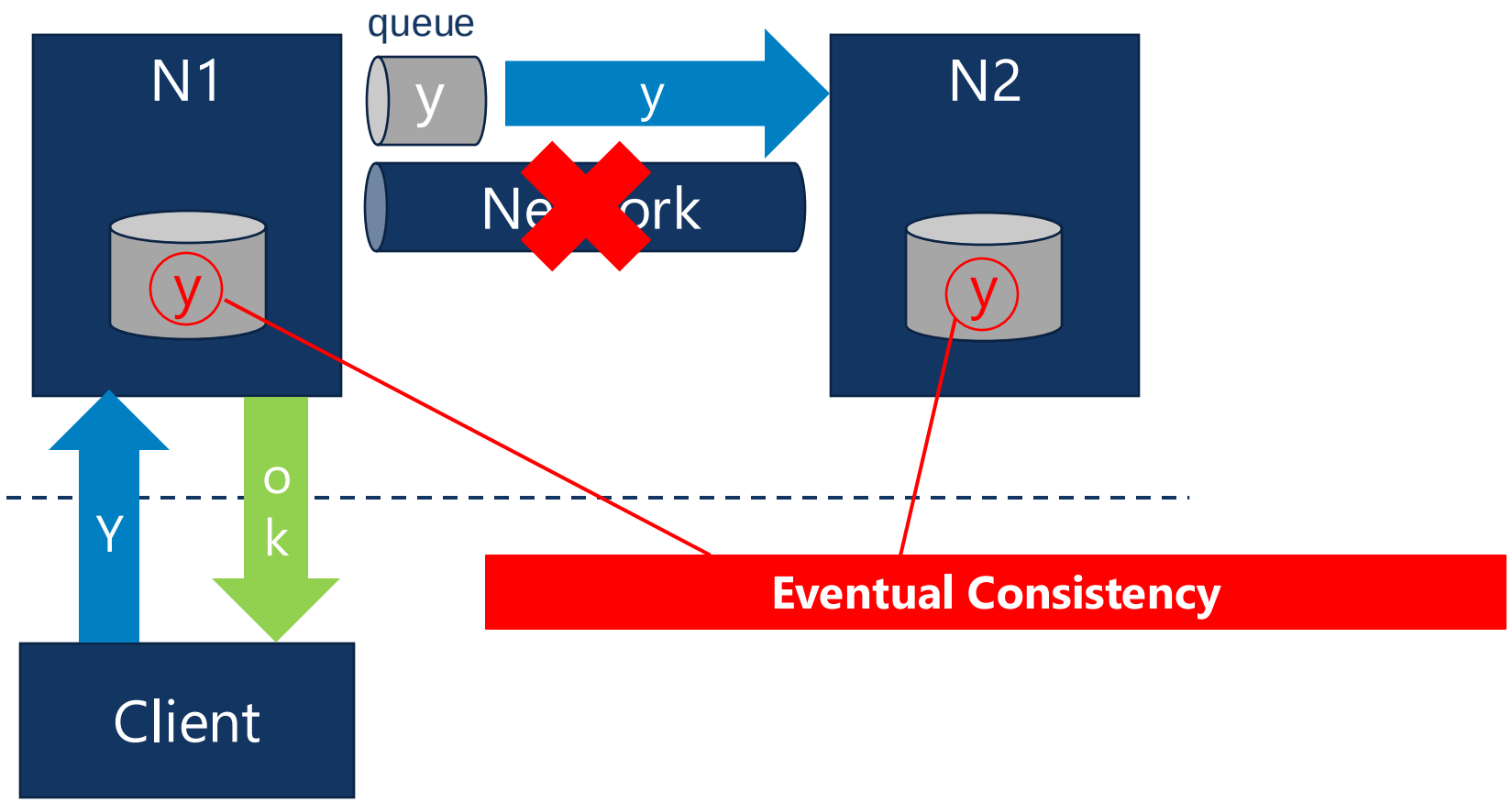
Availability



Availability



Availability + Eventual Consistency



Eventual Consistency

- EC is often not easily accepted
  - “And what about “ACID” and 2PC?”
- Yet, in the “real” world almost every proces is EC
  - Consider whether you really need full consistency when automating business processes
  - Users tend to “get” EC a lot better than we think
  - EC can save you a lot of complexity and trouble (and $)
  - Compensating actions vs. 2PC

CQRS

Solid Innovator

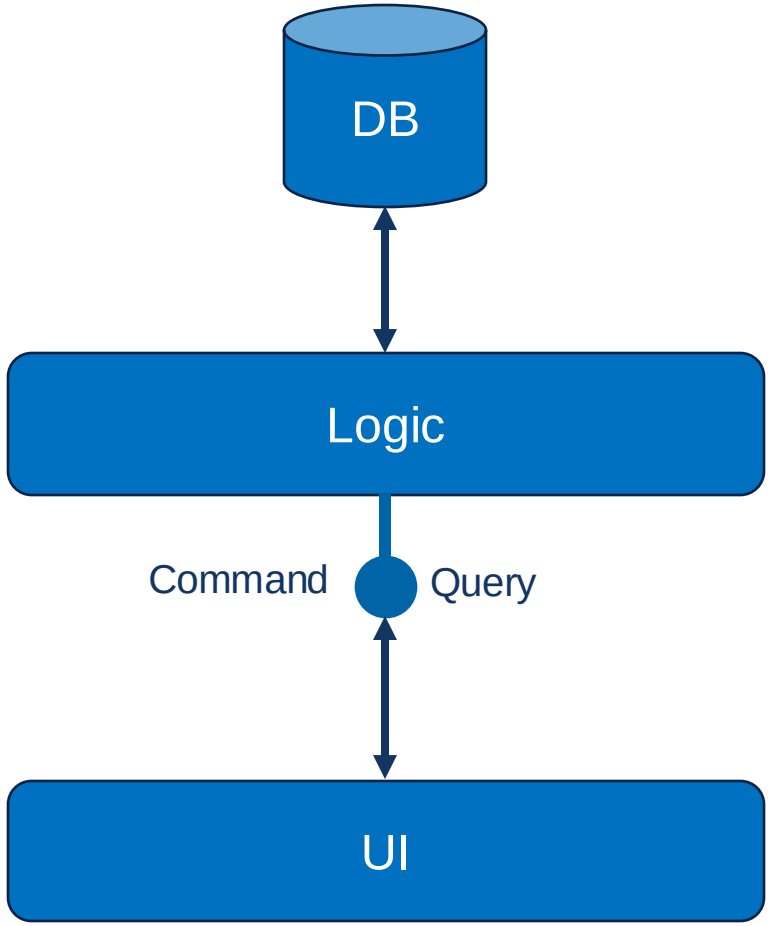
CQRS

- Command Query Responsibility Segregation

- Pattern that embodies separating updates and queries in a system
  - Scale the update and query parts independently
  - Decreases coupling between systems
  - Enables a task oriented approach for your system (commands)

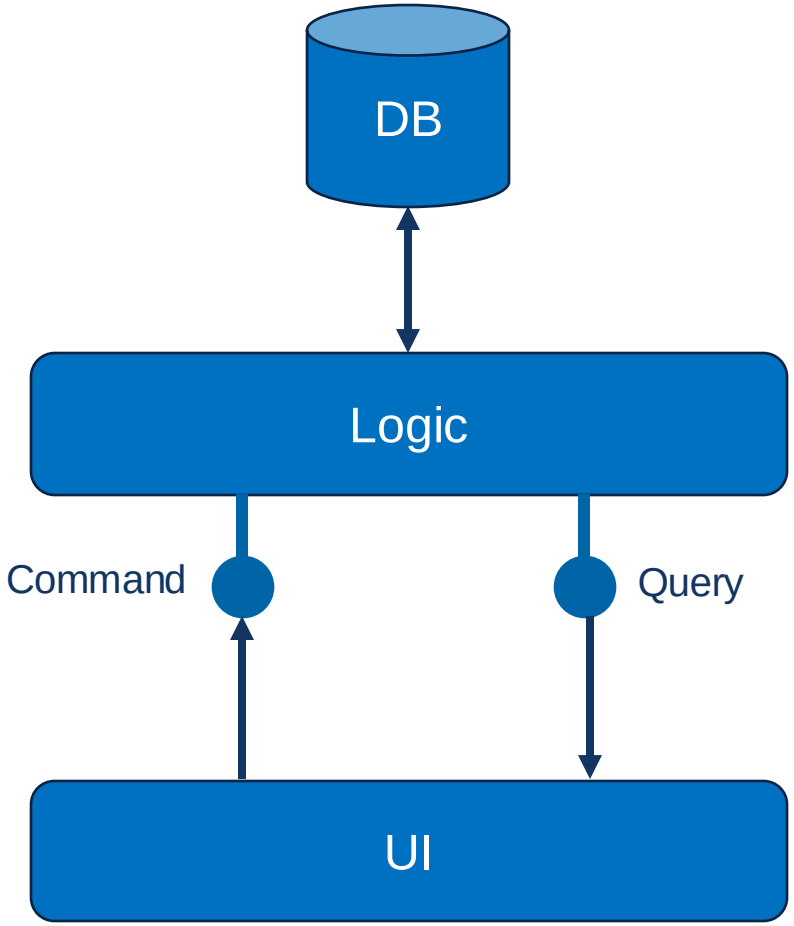
Evolution from SOA to CQRS

Traditional Architecture

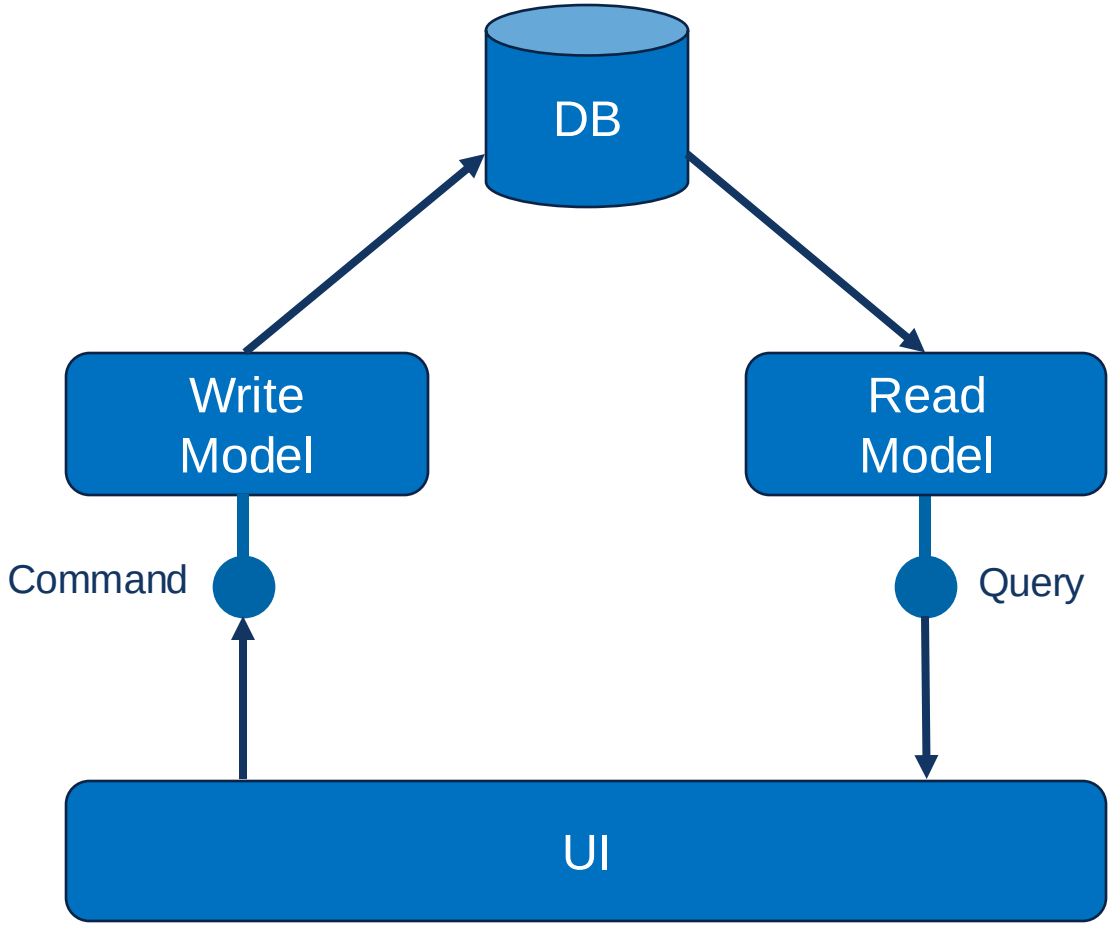


Evolution from SOA to CQRS

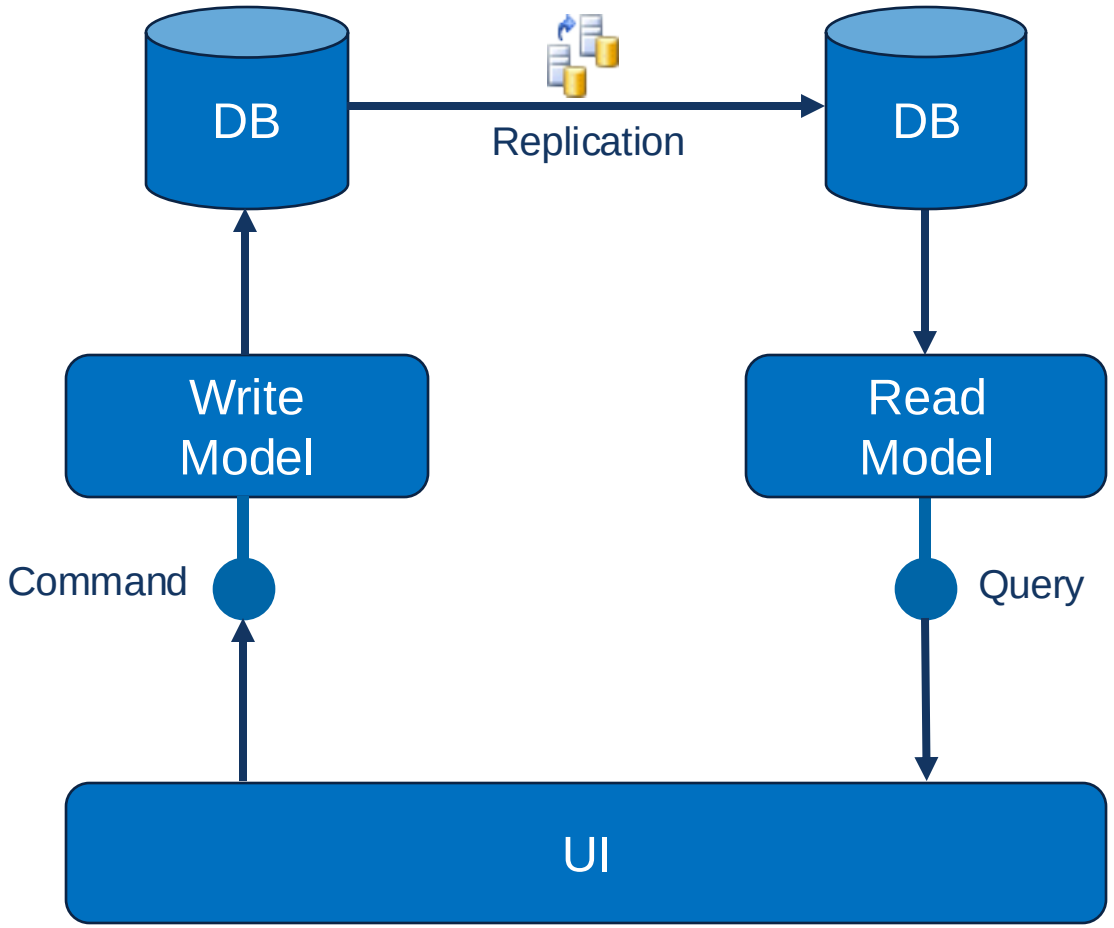
CQS



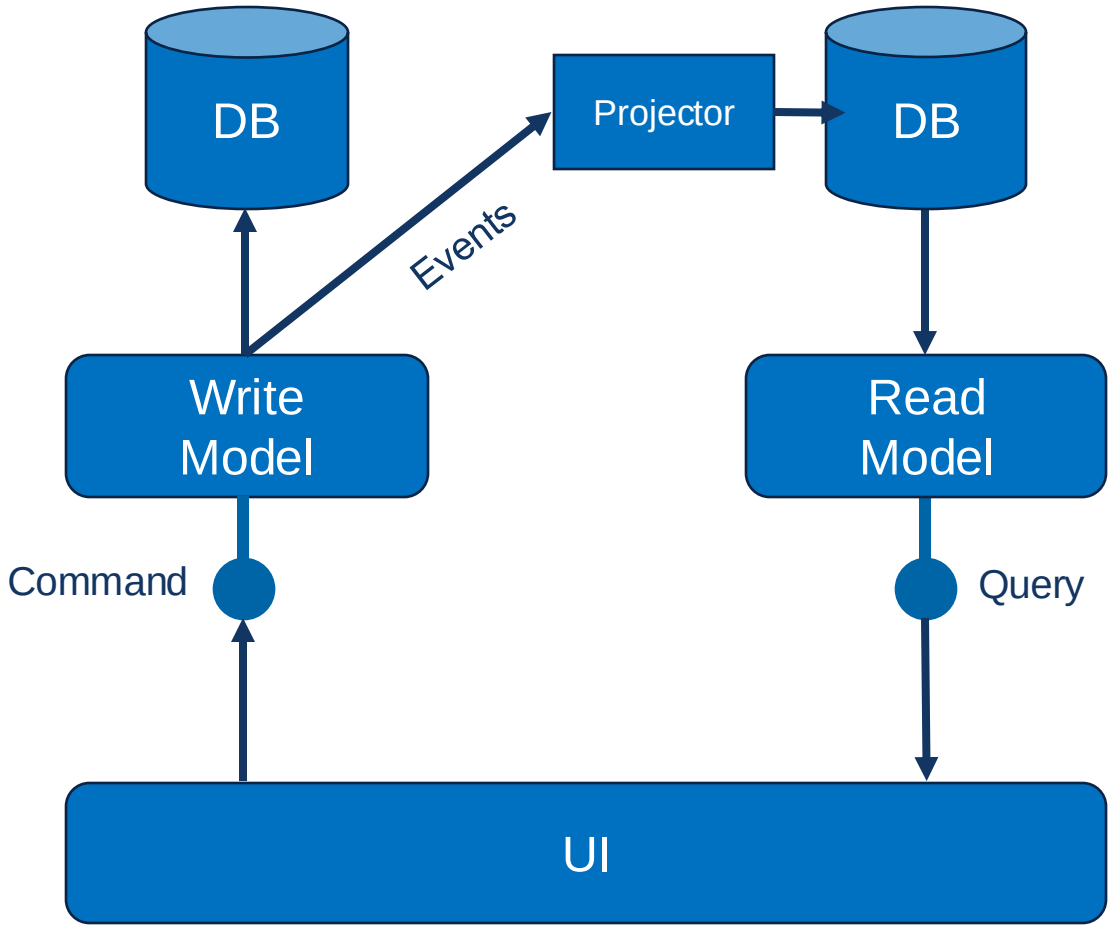
CQRS



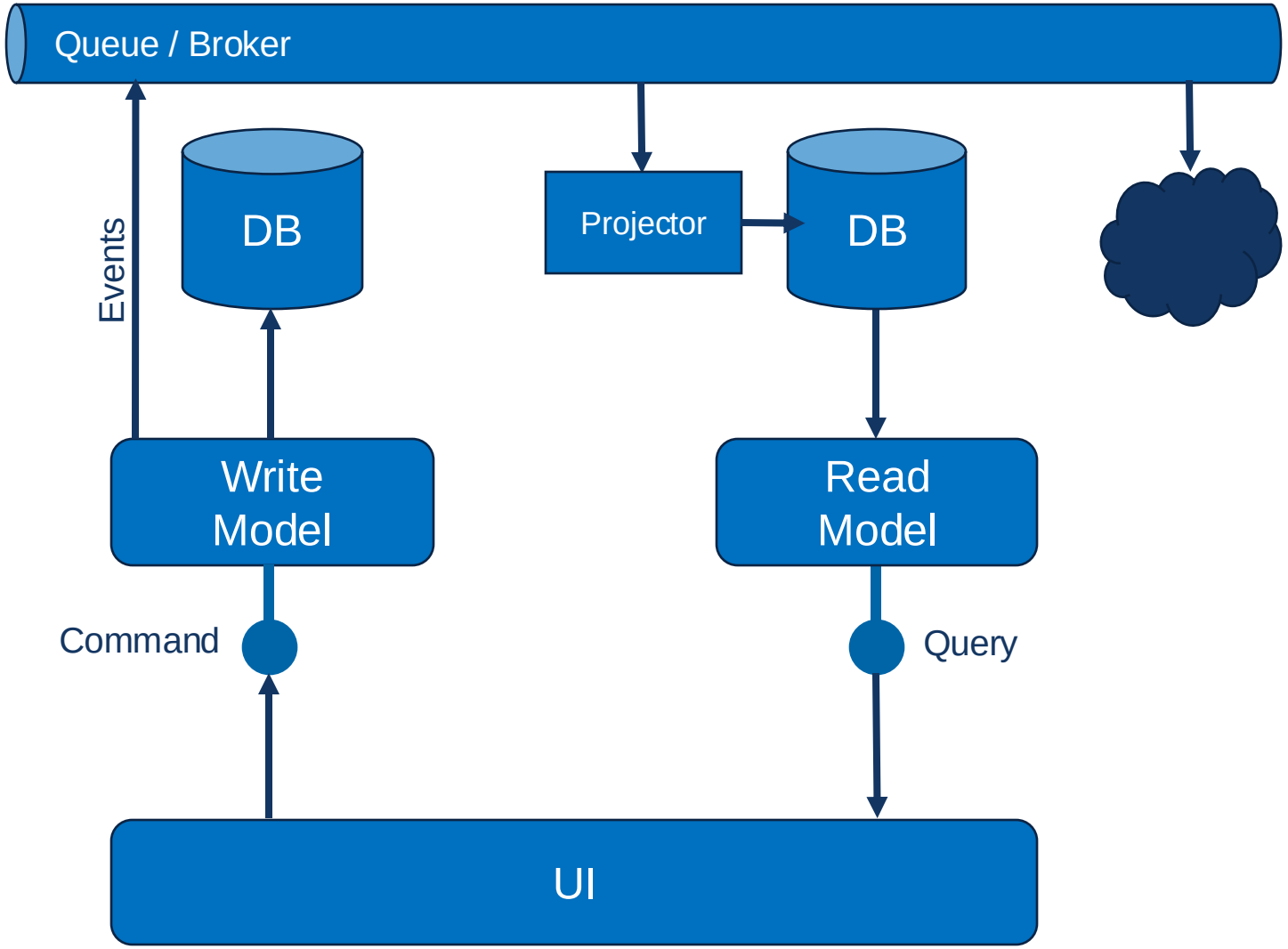
CQRS



CQRS


DB
Projector
DB
Events
Write
Model
Read
Model
Command
Query
UI

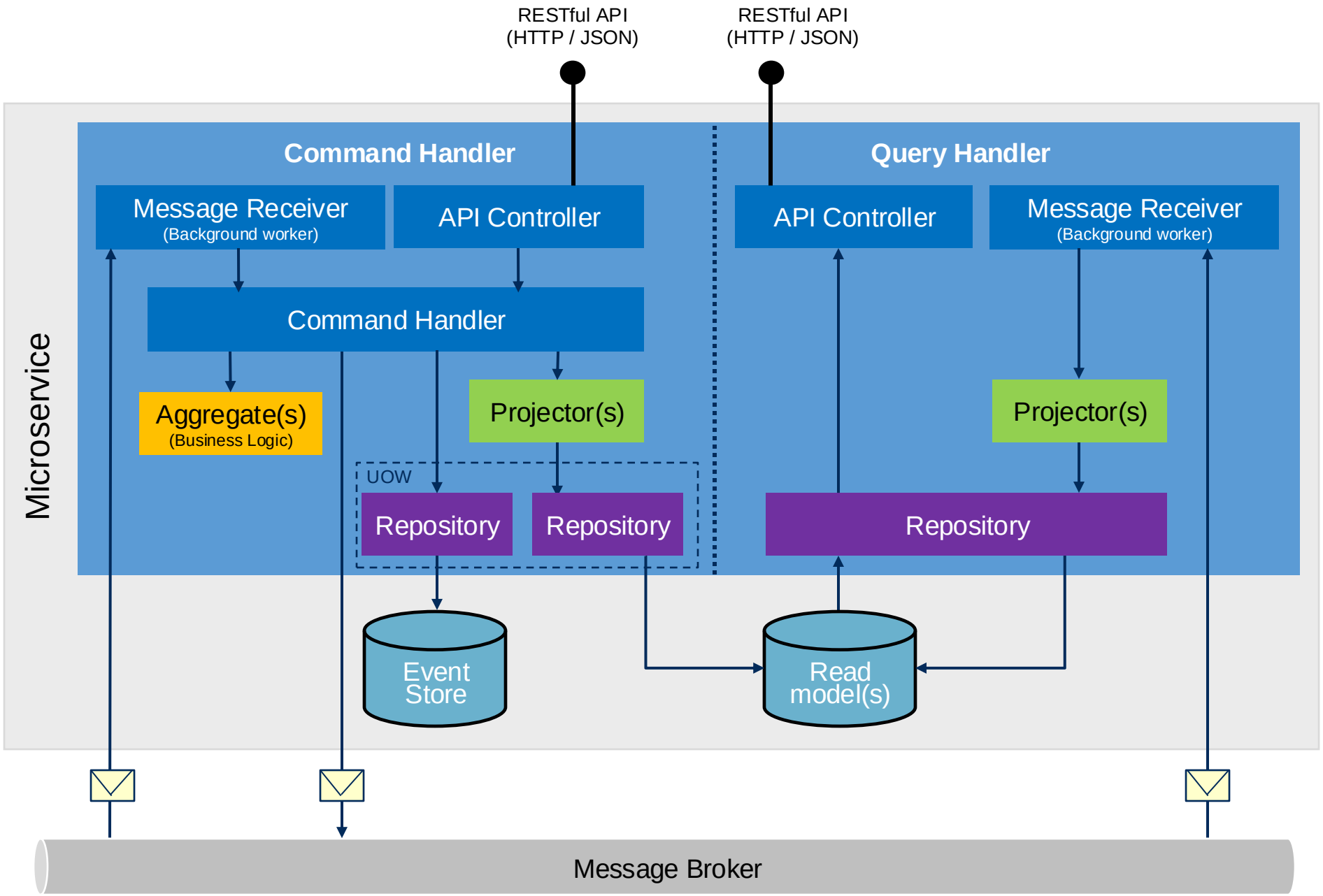
CQRS


Queue / Broker
Events
DB
Projector
DB
Write Model
Read Model
Command
Query
UI

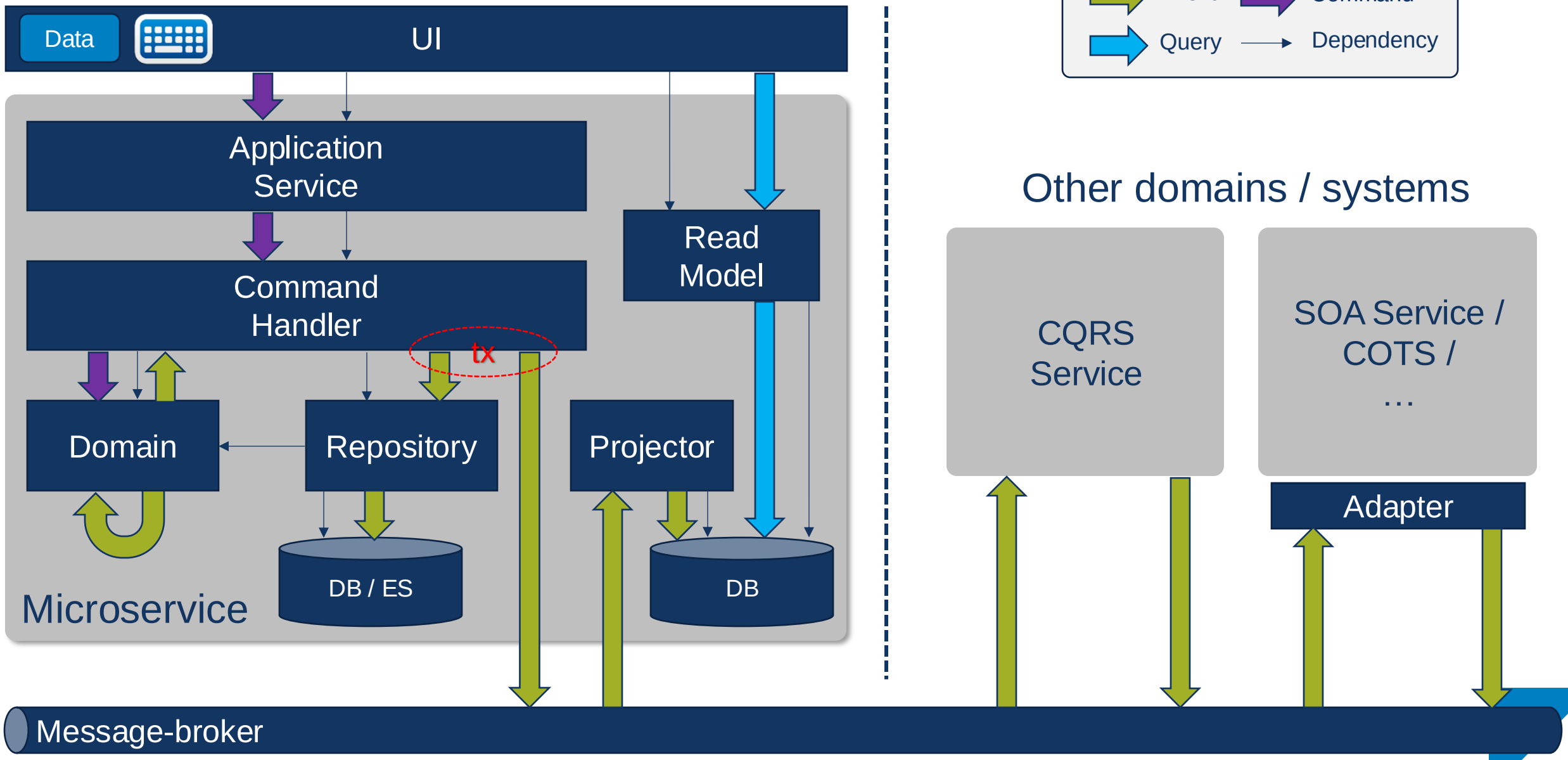
CQRS - Commands & Events

- Commands are the things that need to be executed
  - Must state business intent
    - So not “UpdateInventory” but “CheckOutItem”
  - Always in the form <Verb><Noun>
  - Can fail (because of business rule / invariants checks)

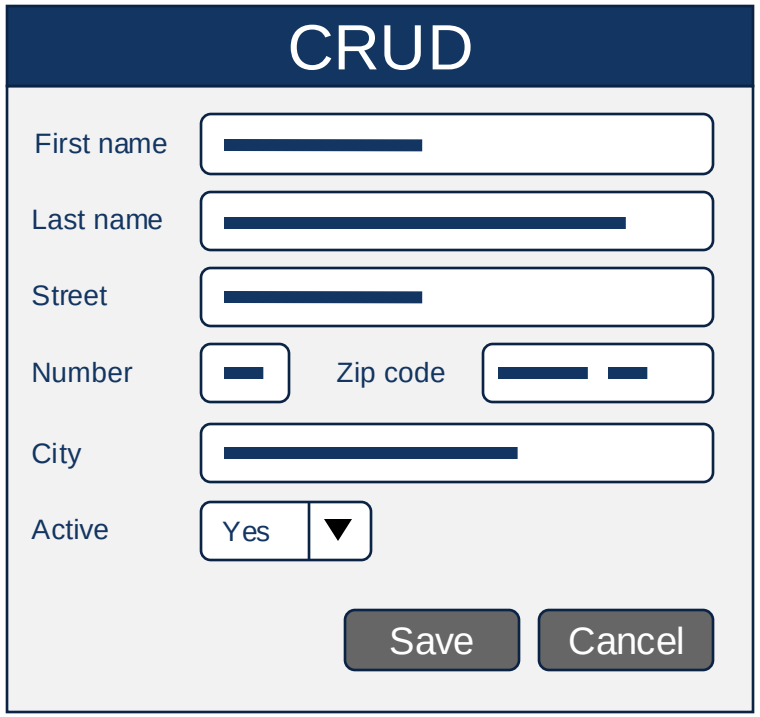
- Events are things that have happened
  - Always in the form <Noun><Verb (past tense)>
    - CustomerRegistered, ItemCheckedOut, AccountClosed, …


RESTful API
(HTTP / JSON)
RESTful API
(HTTP / JSON)
Microservice
Command Handler
Message Receiver
(Background worker)
API Controller
Command Handler
Aggregate(s)
(Business Logic)
Projector(s)
UOW
Repository
Repository
Event
Store
Query Handler
API Controller
Message Receiver
(Background worker)
Projector(s)
Repository
Read
model(s)
Message Broker

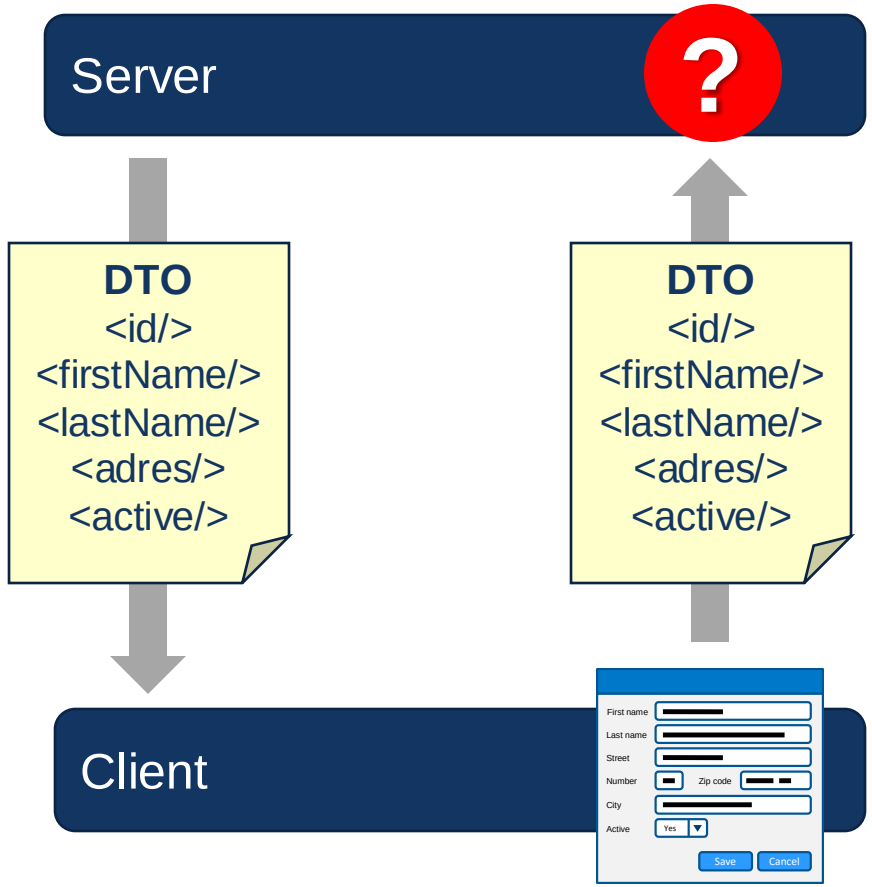
CQRS - under the covers



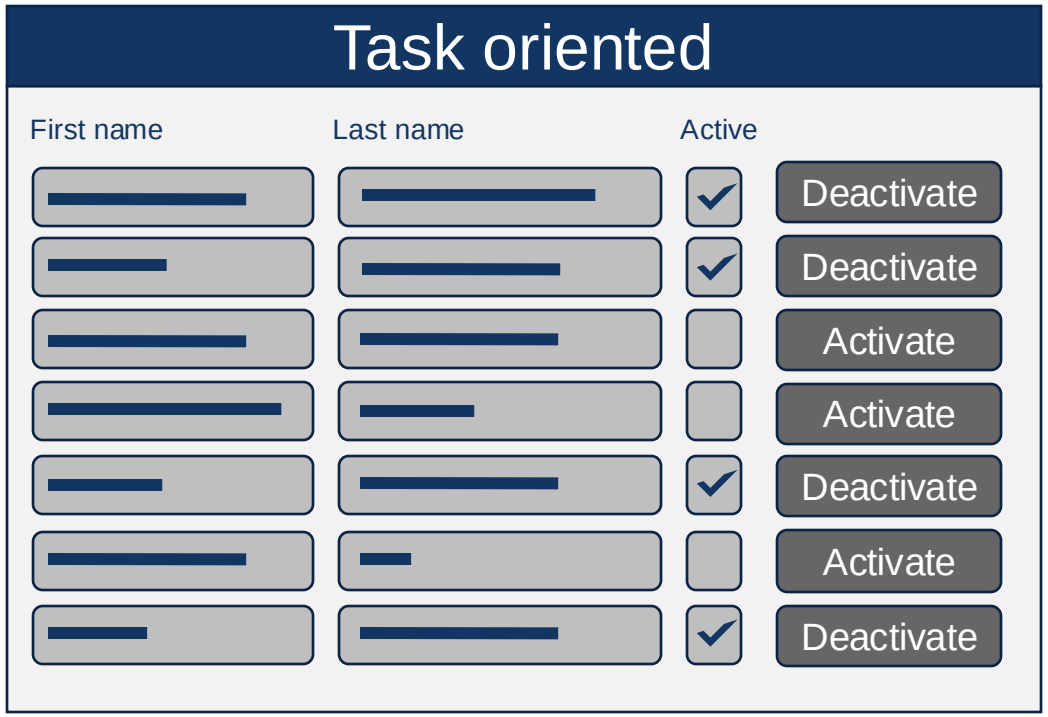
CQRS - CRUD vs. Task Based



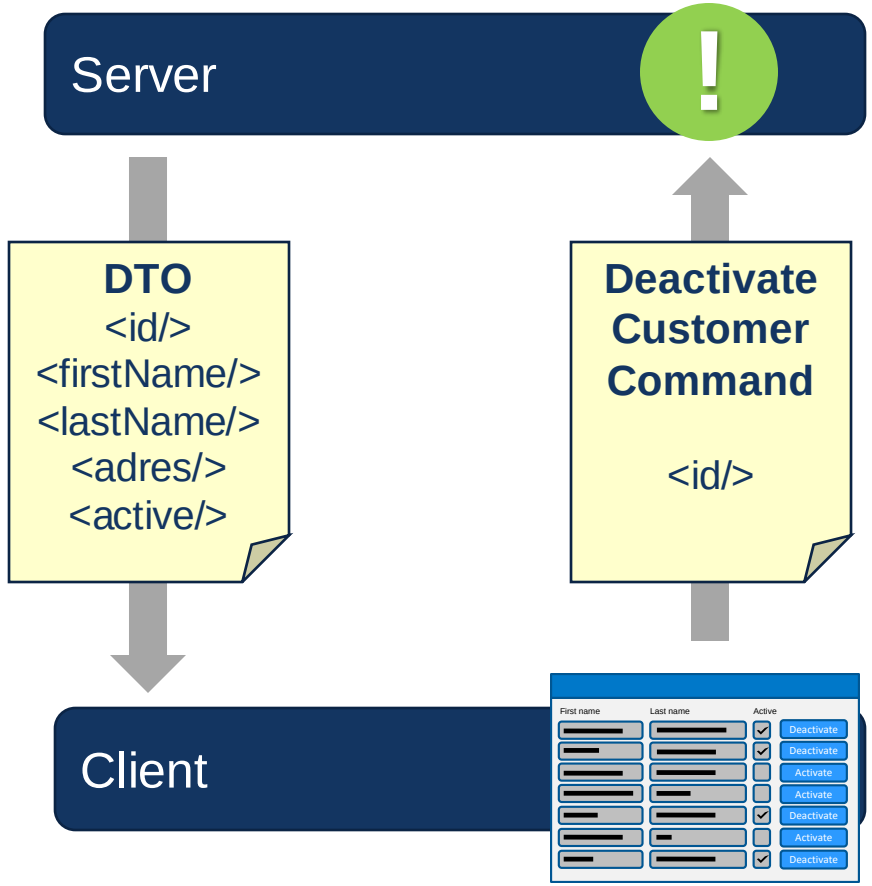
CQRS - CRUD vs. Task Based



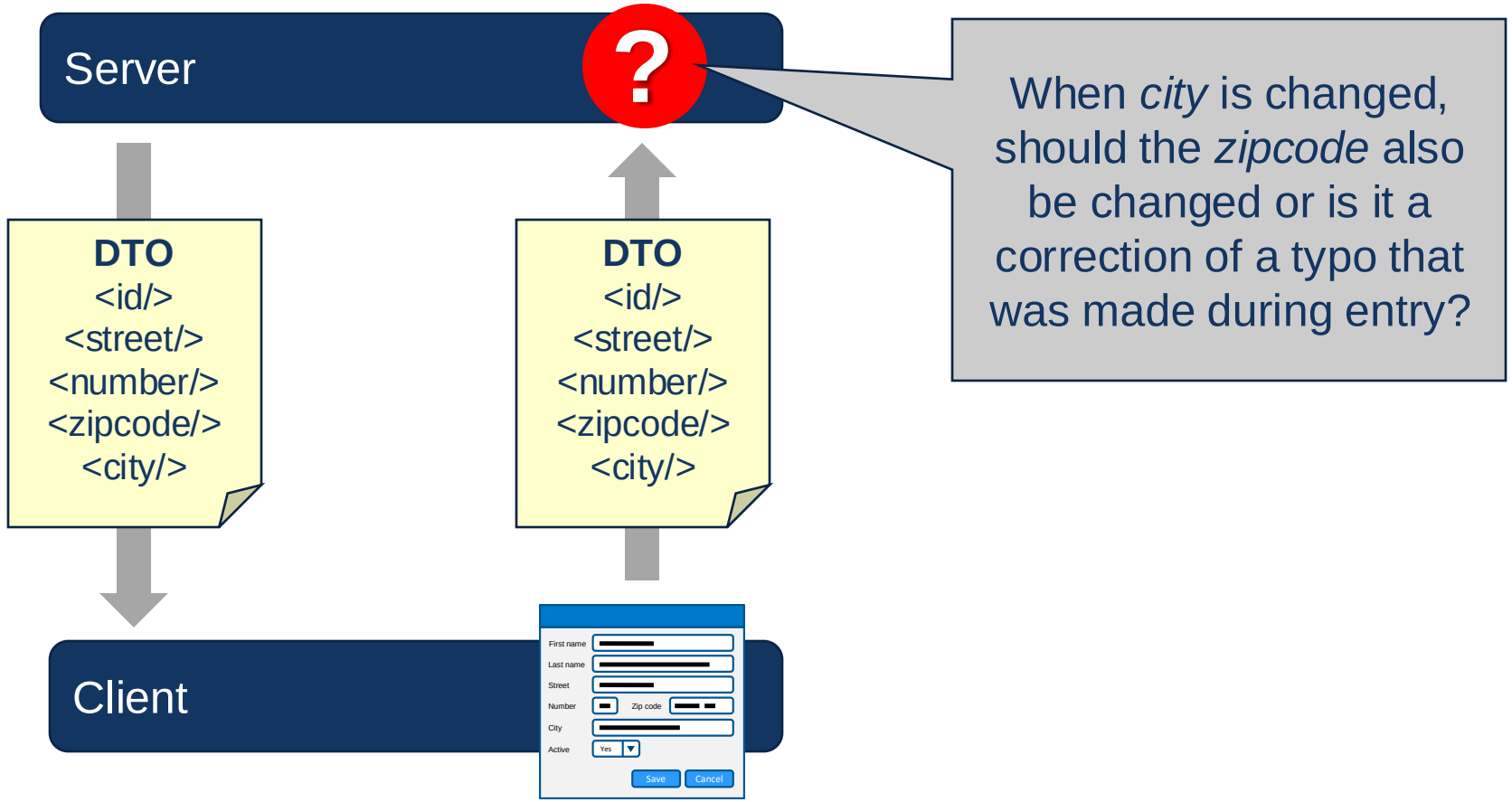
CQRS - CRUD vs. Task Based



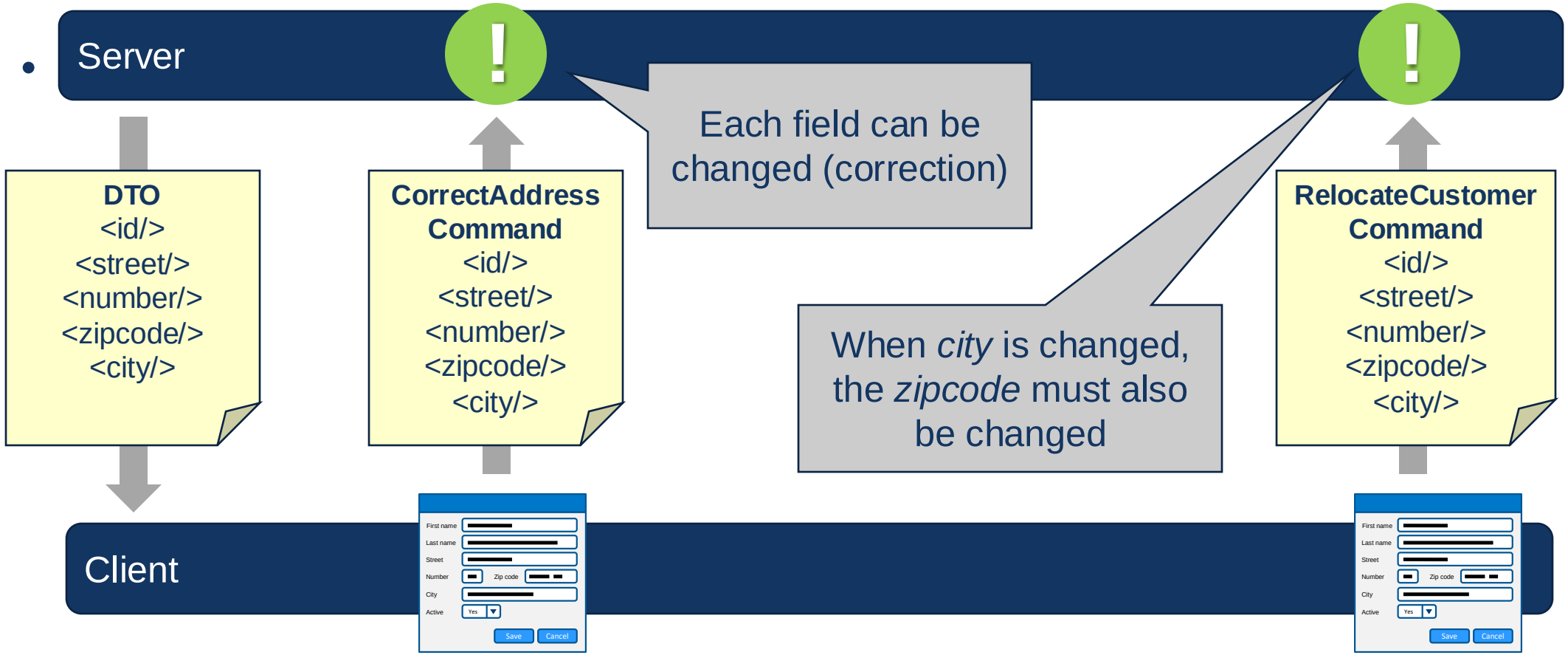
CQRS - CRUD vs. Task Based



CQRS - Business Intent



CQRS - Business Intent



Command-handling

- Handling a command is a 2 phase process:
  - Check phase
    - Check all invariants and business-rules to make sure the command can be executed
    - External resources or services can be called in this phase
  - Execution phase
    - Update the state of the domain
    - Events are published
- This separation paves the way for Event Sourcing

Event Sourcing

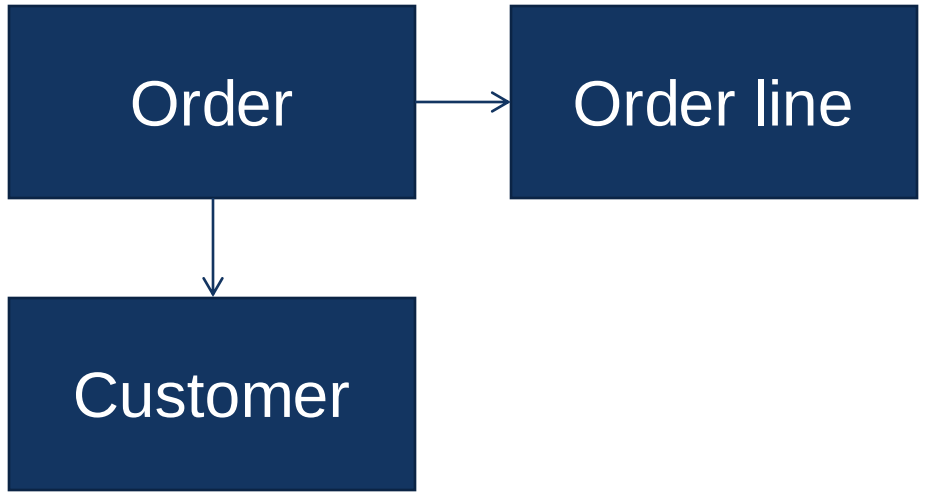
Solid Innovator

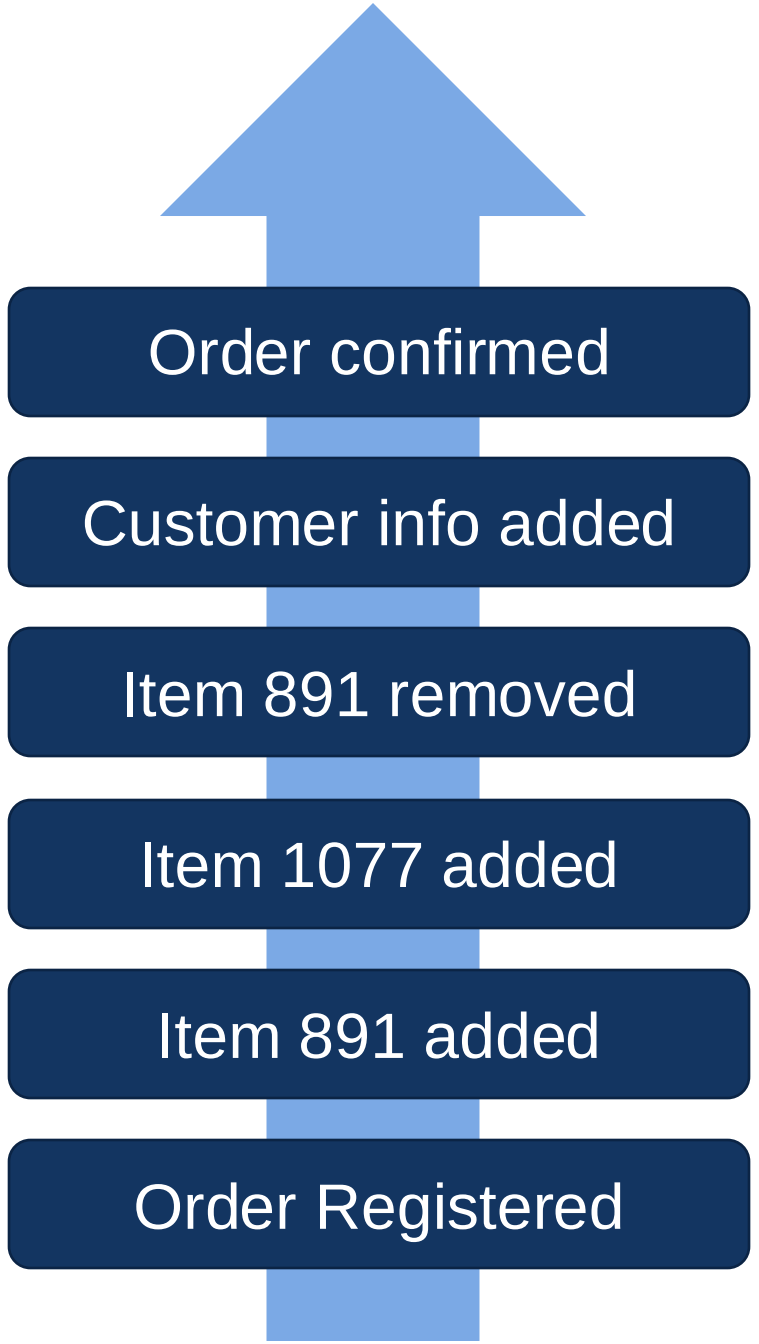
Event-sourcing

- Event-sourcing is an alternative way of persisting the state of your domain-objects

- Not normalized in an RDBMS, but as an immutable list of events that have occurred over time

Event-sourcing





Event-sourcing

- Events are immutable and new events only be appended (not be inserted in between)
  - Think accountant's ledger
  - Appending "Correction" events are allowed
- Snapshots can be used to boost performance
  - Only when absolutely necessary
  - Splitting up the domain can eliminate the need for snapshots

Why event-sourcing?

- Append only, so super fast (no locking etc.)
- Ability to completely rebuild the state based on event history
- Ability to analyze behavior that occurred in the past
  - Audit log for free
- State can be built-up by issuing events
  - Simplifies automated testing
- Ability to apply changes in retrospect

Changes in retrospect

Date	Event	Amount	Total (calculated)
28-08-2017	1% Interest payed out	€ 16,50	€ 1.666,17
01-08-2017	Premium received	€ 150,00	€ 1.649,67
28-07-2017	1% Interest payed out	€ 14,85	€ 1.499,67
01-07-2017	Premium received	€ 150,00	€ 1.484,82
28-06-2017	1% Interest payed out	€ 13,22	€ 1.334,82
01-06-2017	Premium received	€ 150,00	€ 1.321,60
28-05-2017	1% Interest payed out	€ 11,60	€ 1.171,60
01-05-2017	Premium received	€ 150,00	€ 1.160,00
28-04-2017	1% Interest payed out	€ 10,00	€ 1.010,00
01-04-2017	Payment received	€ 1.000,00	€ 1.000,00
01-04-2017	Account opened	€ -	€ -

- Premium is collected every 1st of the month
- Interest is payed out every 28th of the month

Date	Event	Amount	Total (calculated)
28-08-2017	1% Interest payed out	€ 16,50	€ 1.666,17
01-08-2017	Premium received	€ 150,00	€ 1.649,67
28-07-2017	1% Interest payed out	€ 14,85	€ 1.499,67
01-07-2017	Premium received	€ 150,00	€ 1.484,82
28-06-2017	1% Interest payed out	€ 13,22	€ 1.334,82
01-06-2017	Premium received	€ 150,00	€ 1.321,60
28-05-2017	1% Interest payed out	€ 11,60	€ 1.171,60
01-05-2017	Premium received	€ 150,00	€ 1.160,00
28-04-2017	1% Interest payed out	€ 10,00	€ 1.010,00
01-04-2017	Payment received	€ 1.000,00	€ 1.000,00
01-04-2017	Account opened	€ -	€ -

29-08-2017: Process an additional payment of € 500,- payed on June 11th 2017

29-08-2017: Process an additional payment of € 500,- payed on June 11th 2017

Date	Event	Amount	Total (calculated)
28-08-2017	1% Interest payed out	€ 16,50	€ 1.666,17
01-08-2017	Premium received	€ 150,00	€ 1.649,67
28-07-2017	1% Interest payed out	€ 14,85	€ 1.499,67
01-07-2017	Premium received	€ 150,00	€ 1.484,82
28-06-2017	1% Interest payed out	€ 13,22	€ 1.334,82
01-06-2017	Premium received	€ 150,00	€ 1.321,60
28-05-2017	1% Interest payed out	€ 11,60	€ 1.171,60
01-05-2017	Premium received	€ 150,00	€ 1.160,00
28-04-2017	1% Interest payed out	€ 10,00	€ 1.010,00
01-04-2017	Payment received	€ 1.000,00	€ 1.000,00
01-04-2017	Account opened	€ -	€ -

In memory

Date	Event	Amount	Total

29-08-2017: Process an additional payment of € 500,- payed on June 11th 2017

Date	Event	Amount	Total (calculated)
28-08-2017	1% Interest payed out	€ 16,50	€ 1.666,17
01-08-2017	Premium received	€ 150,00	€ 1.649,67
28-07-2017	1% Interest payed out	€ 14,85	€ 1.499,67
01-07-2017	Premium received	€ 150,00	€ 1.484,82
28-06-2017	1% Interest payed out	€ 13,22	€ 1.334,82
01-06-2017	Premium received	€ 150,00	€ 1.321,60
28-05-2017	1% Interest payed out	€ 11,60	€ 1.171,60
01-05-2017	Premium received	€ 150,00	€ 1.160,00
28-04-2017	1% Interest payed out	€ 10,00	€ 1.010,00
01-04-2017	Payment received	€ 1.000,00	€ 1.000,00
01-04-2017	Account opened	€ -	€ -

In memory

Date	Event	Amount	Total
01-06-2017	Premium received	€ 150,00	€ 1.321,60
28-05-2017	1% Interest payed out	€ 11,60	€ 1.171,60
01-05-2017	Premium received	€ 150,00	€ 1.160,00
28-04-2017	1% Interest payed out	€ 10,00	€ 1.010,00
01-04-2017	Payment received	€ 1.000,00	€ 1.000,00
01-04-2017	Account opened	€ -	€ -

29-08-2017: Process an additional payment of € 500,- payed on June 11th 2017

Date	Event	Amount	Total (calculated)
28-08-2017	1% Interest payed out	€ 16,50	€ 1.666,17
01-08-2017	Premium received	€ 150,00	€ 1.649,67
28-07-2017	1% Interest payed out	€ 14,85	€ 1.499,67
01-07-2017	Premium received	€ 150,00	€ 1.484,82
28-06-2017	1% Interest payed out	€ 13,22	€ 1.334,82
01-06-2017	Premium received	€ 150,00	€ 1.321,60
28-05-2017	1% Interest payed out	€ 11,60	€ 1.171,60
01-05-2017	Premium received	€ 150,00	€ 1.160,00
28-04-2017	1% Interest payed out	€ 10,00	€ 1.010,00
01-04-2017	Payment received	€ 1.000,00	€ 1.000,00
01-04-2017	Account opened	€ -	€ -

replay

In memory

Date	Event	Amount	Total
11-06-2017	Payment received	€ 500,00	€ 1.821,60
01-06-2017	Premium received	€ 150,00	€ 1.321,60
28-05-2017	1% Interest payed out	€ 11,60	€ 1.171,60
01-05-2017	Premium received	€ 150,00	€ 1.160,00
28-04-2017	1% Interest payed out	€ 10,00	€ 1.010,00
01-04-2017	Payment received	€ 1.000,00	€ 1.000,00
01-04-2017	Account opened	€ -	€ -

29-08-2017: Process an additional payment of € 500,- payed on June 11th 2017

Date	Event	Amount	Total (calculated)
28-08-2017	1% Interest payed out	€ 16,50	€ 1.666,17
01-08-2017	Premium received	€ 150,00	€ 1.649,67
28-07-2017	1% Interest payed out	€ 14,85	€ 1.499,67
01-07-2017	Premium received	€ 150,00	€ 1.484,82
28-06-2017	1% Interest payed out	€ 13,22	€ 1.334,82
01-06-2017	Premium received	€ 150,00	€ 1.321,60
28-05-2017	1% Interest payed out	€ 11,60	€ 1.171,60
01-05-2017	Premium received	€ 150,00	€ 1.160,00
28-04-2017	1% Interest payed out	€ 10,00	€ 1.010,00
01-04-2017	Payment received	€ 1.000,00	€ 1.000,00
01-04-2017	Account opened	€ -	€ -

In memory

Date	Event	Amount	Total
28-08-2017	1% Interest payed out	€ 21,60	€ 2.181,32
01-08-2017	Premium received	€ 150,00	€ 2.159,72
28-07-2017	1% Interest payed out	€ 19,90	€ 2.009,72
01-07-2017	Premium received	€ 150,00	€ 1.989,82
28-06-2017	1% Interest payed out	€ 18,22	€ 1.839,82
11-06-2017	Payment received	€ 500,00	€ 1.821,60
01-06-2017	Premium received	€ 150,00	€ 1.321,60
28-05-2017	1% Interest payed out	€ 11,60	€ 1.171,60
01-05-2017	Premium received	€ 150,00	€ 1.160,00
28-04-2017	1% Interest payed out	€ 10,00	€ 1.010,00
01-04-2017	Payment received	€ 1.000,00	€ 1.000,00
01-04-2017	Account opened	€ -	€ -

29-08-2017: Process an additional payment of € 500,- payed on June 11th 2017

Difference: € 515,15

(€ 500,- payment + € 15,15 interest)

Date	Event	Amount	Total (calculated)
28-08-2017	1% Interest payed out	€ 16,50	€ 1.666,17
01-08-2017	Premium received	€ 150,00	€ 1.649,67
28-07-2017	1% Interest payed out	€ 14,85	€ 1.499,67
01-07-2017	Premium received	€ 150,00	€ 1.484,82
28-06-2017	1% Interest payed out	€ 13,22	€ 1.334,82
01-06-2017	Premium received	€ 150,00	€ 1.321,60
28-05-2017	1% Interest payed out	€ 11,60	€ 1.171,60
01-05-2017	Premium received	€ 150,00	€ 1.160,00
28-04-2017	1% Interest payed out	€ 10,00	€ 1.010,00
01-04-2017	Payment received	€ 1.000,00	€ 1.000,00
01-04-2017	Account opened	€ -	€ -

replay

In memory

Date	Event	Amount	Total
28-08-2017	1% Interest payed out	€ 21,60	€ 2.181,32
01-08-2017	Premium received	€ 150,00	€ 2.159,72
28-07-2017	1% Interest payed out	€ 19,90	€ 2.009,72
01-07-2017	Premium received	€ 150,00	€ 1.989,82
28-06-2017	1% Interest payed out	€ 18,22	€ 1.839,82
11-06-2017	Payment received	€ 500,00	€ 1.821,60
01-06-2017	Premium received	€ 150,00	€ 1.321,60
28-05-2017	1% Interest payed out	€ 11,60	€ 1.171,60
01-05-2017	Premium received	€ 150,00	€ 1.160,00
28-04-2017	1% Interest payed out	€ 10,00	€ 1.010,00
01-04-2017	Payment received	€ 1.000,00	€ 1.000,00
01-04-2017	Account opened	€ -	€ -

29-08-2017: Process an additional payment of € 500,- payed on June 11th 2017

Date	Event	Amount	Total (calculated)
28-08-2017	1% Interest payed out	€ 16,50	€ 1.666,17
01-08-2017	Premium received	€ 150,00	€ 1.649,67
28-07-2017	1% Interest payed out	€ 14,85	€ 1.499,67
01-07-2017	Premium received	€ 150,00	€ 1.484,82
28-06-2017	1% Interest payed out	€ 13,22	€ 1.334,82
01-06-2017	Premium received	€ 150,00	€ 1.321,60
28-05-2017	1% Interest payed out	€ 11,60	€ 1.171,60
01-05-2017	Premium received	€ 150,00	€ 1.160,00
28-04-2017	1% Interest payed out	€ 10,00	€ 1.010,00
01-04-2017	Payment received	€ 1.000,00	€ 1.000,00
01-04-2017	Account opened	€ -	€ -

29-08-2017: Process an additional payment of € 500,- payed on June 11th 2017

Date	Event	Amount	Total (calculated)
29-08-2017	Retro Payment 11-06	€ 500,00	€ 2.166,17
28-08-2017	1% Interest payed out	€ 16,50	€ 1.666,17
01-08-2017	Premium received	€ 150,00	€ 1.649,67
28-07-2017	1% Interest payed out	€ 14,85	€ 1.499,67
01-07-2017	Premium received	€ 150,00	€ 1.484,82
28-06-2017	1% Interest payed out	€ 13,22	€ 1.334,82
01-06-2017	Premium received	€ 150,00	€ 1.321,60
28-05-2017	1% Interest payed out	€ 11,60	€ 1.171,60
01-05-2017	Premium received	€ 150,00	€ 1.160,00
28-04-2017	1% Interest payed out	€ 10,00	€ 1.010,00
01-04-2017	Payment received	€ 1.000,00	€ 1.000,00
01-04-2017	Account opened	€ -	€ -

29-08-2017: Process an additional payment of € 500,- payed on June 11th 2017

Date	Event	Amount	Total (calculated)
29-08-2017	Retro Correction 11-06	€ 15,15	€ 2.181,32
29-08-2017	Retro Payment 11-06	€ 500,00	€ 2.166,17
28-08-2017	1% Interest payed out	€ 16,50	€ 1.666,17
01-08-2017	Premium received	€ 150,00	€ 1.649,67
28-07-2017	1% Interest payed out	€ 14,85	€ 1.499,67
01-07-2017	Premium received	€ 150,00	€ 1.484,82
28-06-2017	1% Interest payed out	€ 13,22	€ 1.334,82
01-06-2017	Premium received	€ 150,00	€ 1.321,60
28-05-2017	1% Interest payed out	€ 11,60	€ 1.171,60
01-05-2017	Premium received	€ 150,00	€ 1.160,00
28-04-2017	1% Interest payed out	€ 10,00	€ 1.010,00
01-04-2017	Payment received	€ 1.000,00	€ 1.000,00
01-04-2017	Account opened	€ -	€ -

29-08-2017: Process an additional payment of € 500,- payed on June 11th 2017

Date	Event	Amount	Total (calculated)
01-09-2017	Premium received	€ 150,00	€ 2.331,32
29-08-2017	Retro Correction 11-06	€ 15,15	€ 2.181,32
29-08-2017	Retro Payment 11-06	€ 500,00	€ 2.166,17
28-08-2017	1% Interest payed out	€ 16,50	€ 1.666,17
01-08-2017	Premium received	€ 150,00	€ 1.649,67
28-07-2017	1% Interest payed out	€ 14,85	€ 1.499,67
01-07-2017	Premium received	€ 150,00	€ 1.484,82
28-06-2017	1% Interest payed out	€ 13,22	€ 1.334,82
01-06-2017	Premium received	€ 150,00	€ 1.321,60
28-05-2017	1% Interest payed out	€ 11,60	€ 1.171,60
01-05-2017	Premium received	€ 150,00	€ 1.160,00
28-04-2017	1% Interest payed out	€ 10,00	€ 1.010,00
01-04-2017	Payment received	€ 1.000,00	€ 1.000,00
01-04-2017	Account opened	€ -	€ -

29-08-2017: Process an additional payment of € 500,- payed on June 11th 2017

Date	Event	Amount	Total (calculated)
28-09-2017	1% Interest payed out	€ 23,31	€ 2.354,63
01-09-2017	Premium received	€ 150,00	€ 2.331,32
29-08-2017	Retro Correction 11-06	€ 15,15	€ 2.181,32
29-08-2017	Retro Payment 11-06	€ 500,00	€ 2.166,17
28-08-2017	1% Interest payed out	€ 16,50	€ 1.666,17
01-08-2017	Premium received	€ 150,00	€ 1.649,67
28-07-2017	1% Interest payed out	€ 14,85	€ 1.499,67
01-07-2017	Premium received	€ 150,00	€ 1.484,82
28-06-2017	1% Interest payed out	€ 13,22	€ 1.334,82
01-06-2017	Premium received	€ 150,00	€ 1.321,60
28-05-2017	1% Interest payed out	€ 11,60	€ 1.171,60
01-05-2017	Premium received	€ 150,00	€ 1.160,00
28-04-2017	1% Interest payed out	€ 10,00	€ 1.010,00
01-04-2017	Payment received	€ 1.000,00	€ 1.000,00
01-04-2017	Account opened	€ -	€ -

Event-sourcing

- Several ES products are available
- Evaluate your needs before adopting a product
  - When only persistence of events is needed, custom built is fine
  - When you need projections / aggregations / complex event processing on events, a product can be more cost effective

- Optimizing code for Event Sourcing
- Database
- Optimistic Concurrency

Optimizing code for Event Sourcing

- **Performing an operation** by **Replaying an event**

```
public void PlanMaintenanceJob(PlanMaintenanceJob command)
{
    // check business rules
    this.NumberOfParallelMaintenanceJobsMustNotExceedAvailableWorkStations(command);
    this.NumberOfParallelMaintenanceJobsOnAVehicleMustNotExceedOne(command);

    // perform operation by 'replaying' an event
    MaintenanceJobPlanned e = command.MapToMaintenanceJobPlanned();
    RaiseEvent(e);
}
```

Adds the event to list of events and
executes the replay-functionality for that event

Two tables with generic layout for each aggregate type

AggregateID	Version

For optimistic concurrency

The order in which the events should be replayed

The type of the serialized Event

The Serialized Event (in JSON)

Id	AggregateID	Order	EventType	EventData

Optimistic Concurrency

- For each aggregate keep track of original version and current version
  - Each time an event is applied, increase the current version
- After executing a command,
  - Only add the new event(s) that occurred to the database …
  - … when the original version of the aggregate is equal to the current version in the database

- Options when concurrency errors occur
  1. Redo the ‘Save’ action (often not desirable)
  2. Restart the complete handling of the command again (if possible)
  3. Raise error

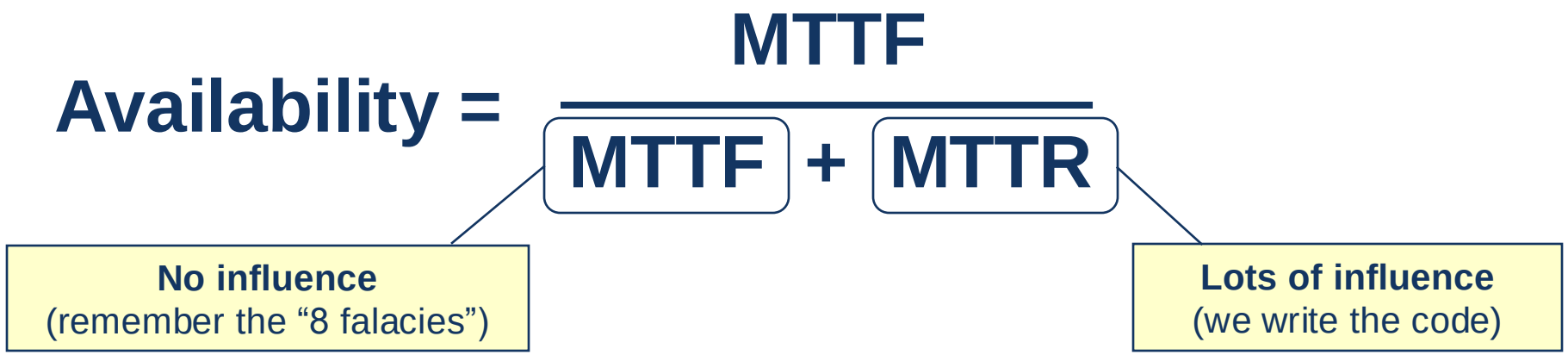
Design for failure

Design for failure

- When building a distributed system **things will be off-line**
- Make sure you **handle failure gracefully**
- Beware of the **"8 fallacies of distributed computing"** (*L. Peter Deutsch*)
- Use (oss) **libraries / frameworks** to help you with this
- Support **versioning** in your contracts and end-points
- Be as **idempotent** as possible
- Be **redundant** when it counts
- Think about **point-in-time restore** when designing the system

Design for failure

- Errors WILL occur - make sure you can recover fast!

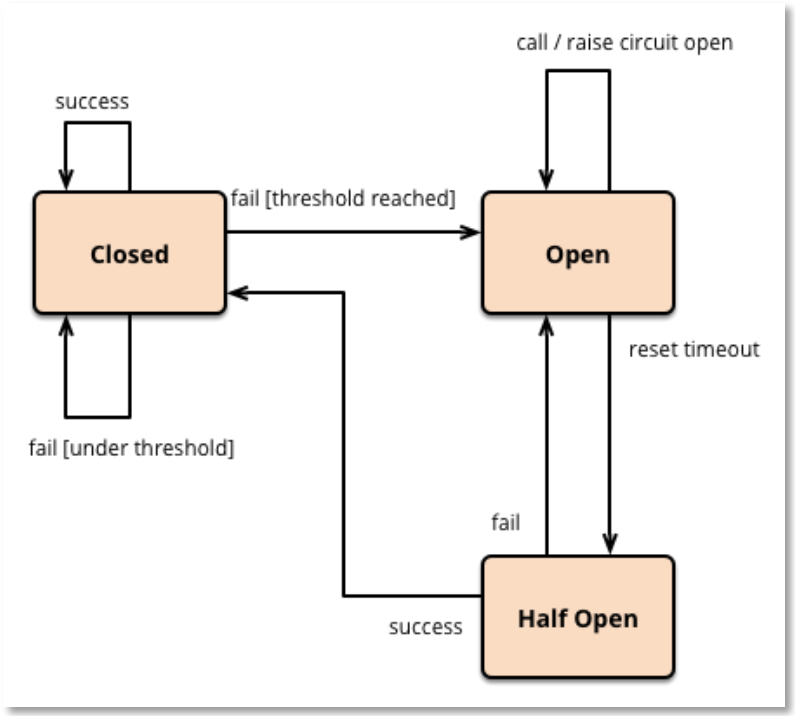


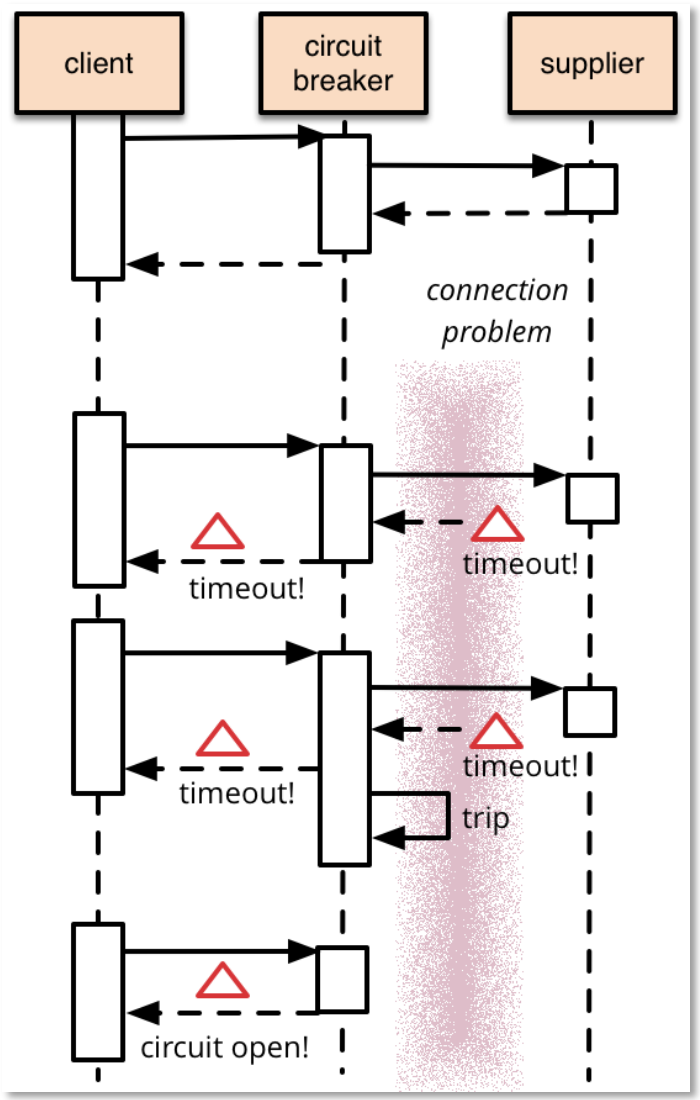
MTTF: Mean Time To Failure
MTTR: Mean Time To Recovery

Isolation - Design for failure

- Introduce fault domains in your system
  - **Bulkhead** pattern (nautical term)
  - Make sure if something breaks, the system only breaks partially
- Built-in retries (with back-off) where possible
- “Fail fast”
  - **Circuit-breaker** pattern
  - Make sure you don’t keep waiting on time-outs from an unhealthy service
    - This bogs down performance and starves thread pools

Circuit breaker





http://martinfowler.com/bliki/CircuitBreaker.html