Erwin de Gier - Trifork Amsterdam

Event-Driven Architecture with Spring WebFlux

@erwindeg

github.com/erwindeg

edegier.nl

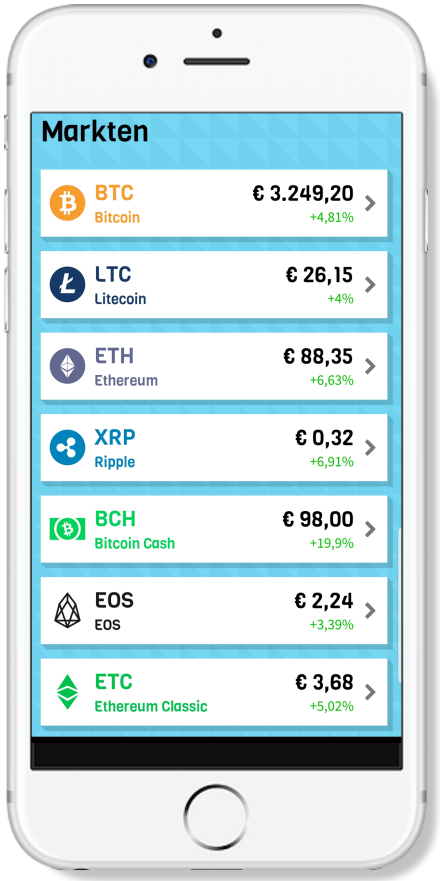
TRIFORK.
...think software

We are BLOX.

Crypto currency trading made easy.
Sign up, verify, deposit, trade in 60 seconds

REST services?

TRIFORK.
...think software



Ledger

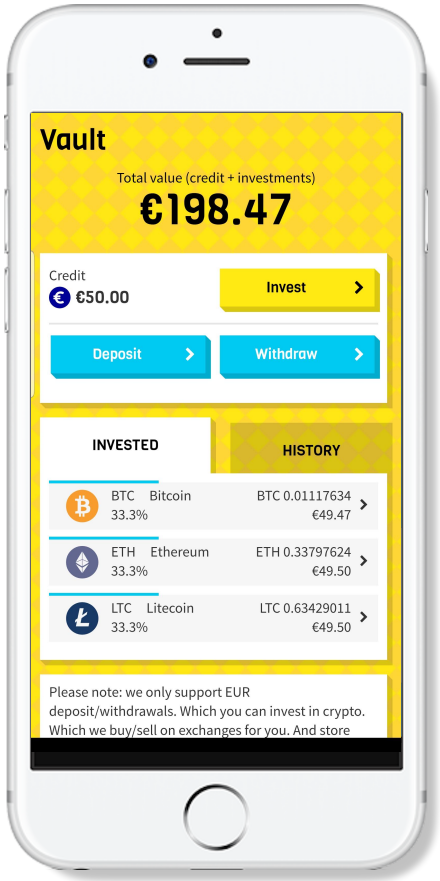
Market Data

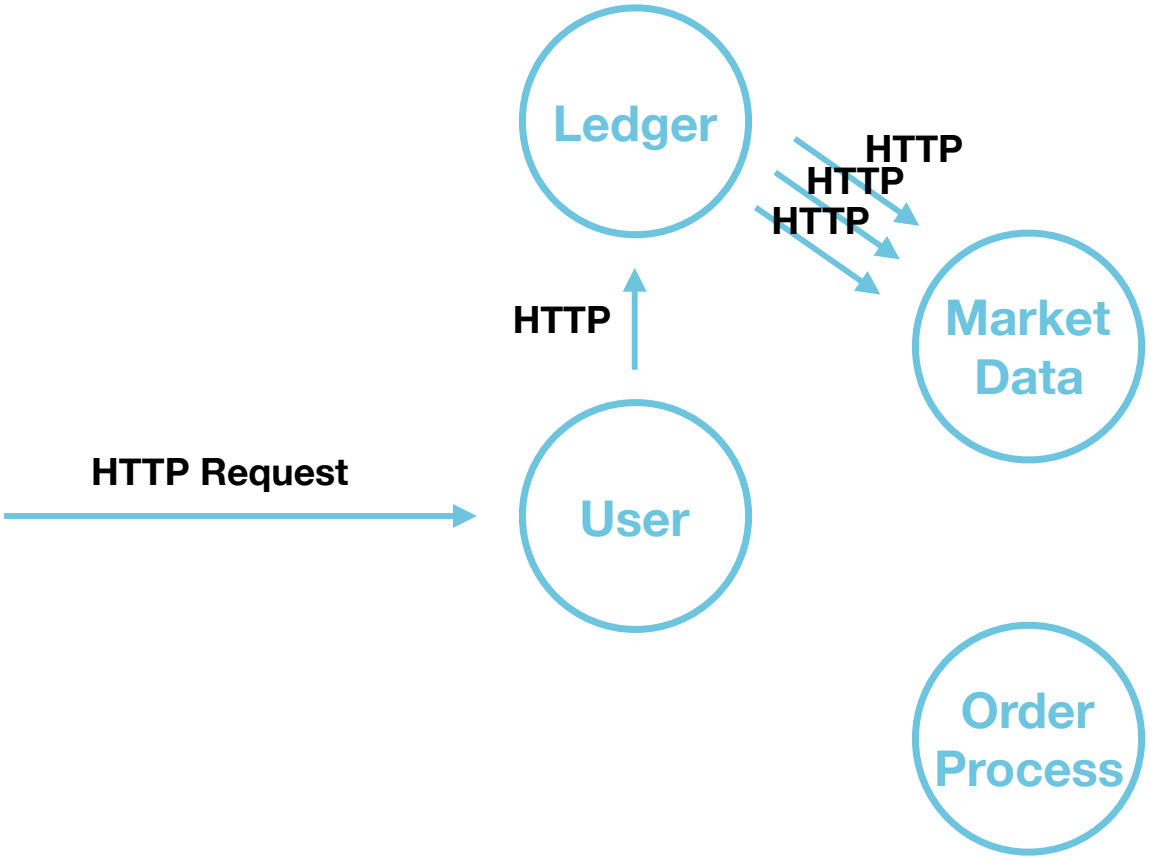
User

Order Process

Reading data

TRIFORK.
...think software





Inner join over HTTP

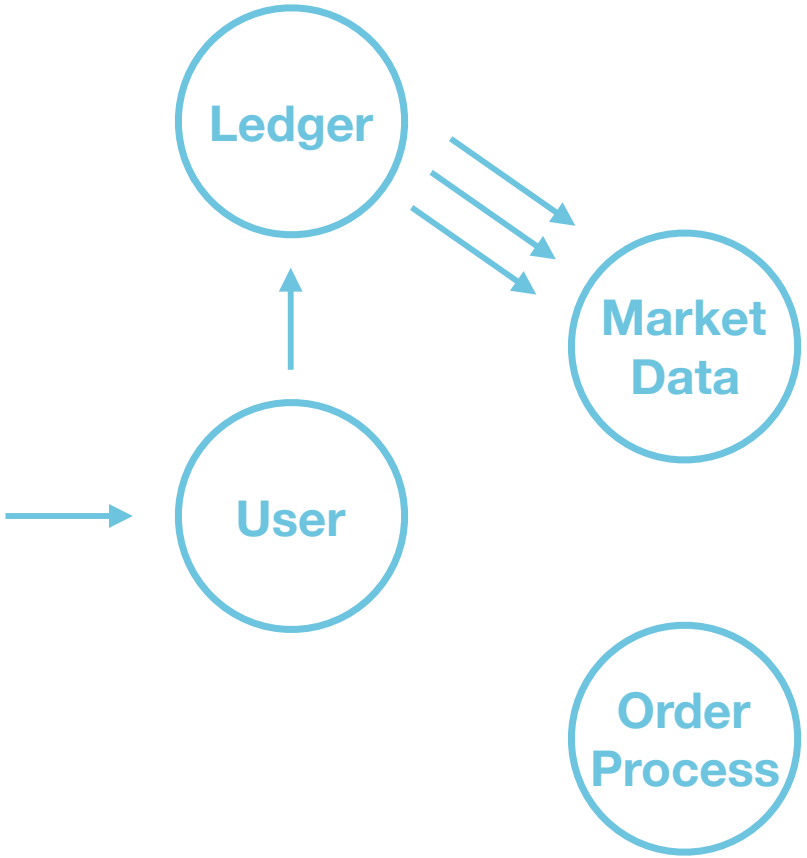
TRIFORK.
...think software

Latency

Making 5 HTTP calls of 20 ms. per call
will in total be 100 ms just for the HTTP calls

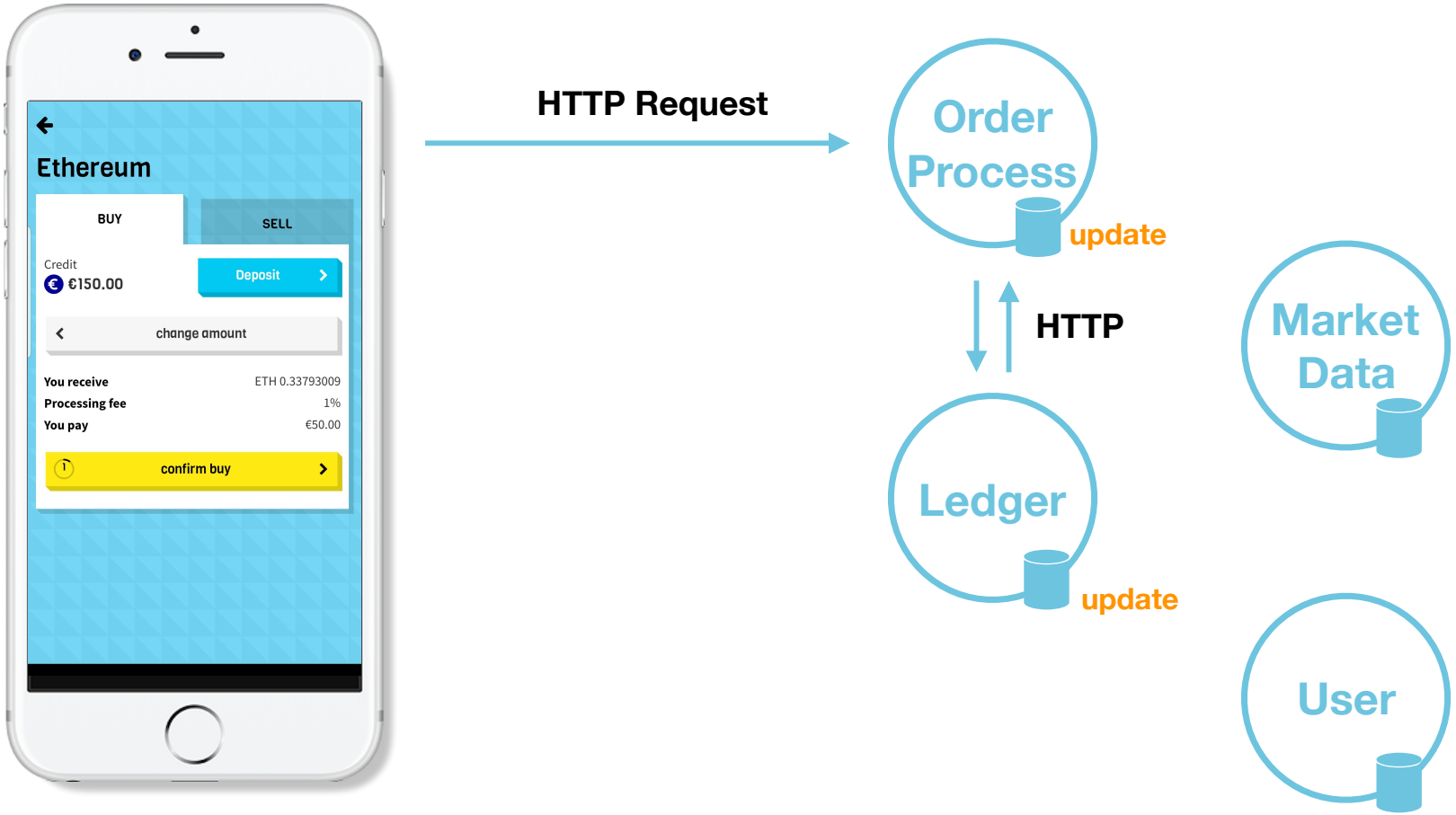
Availability

5 calls to 99,999% available services
will result in a total availability of $99{,}999^5 \approx 99{,}99\%$



Distributed transaction

TRIFORK.
...think software



https://dzone.com/articles/event-driven-data-management-for-microservices-1

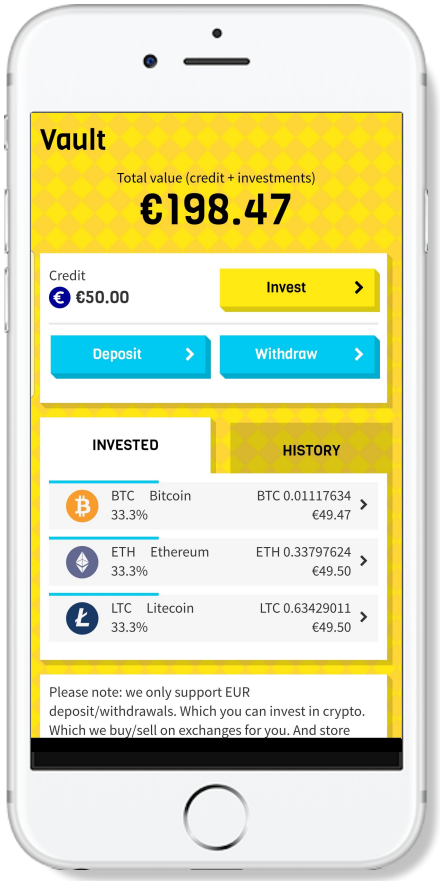
TRIFORK.
...think software

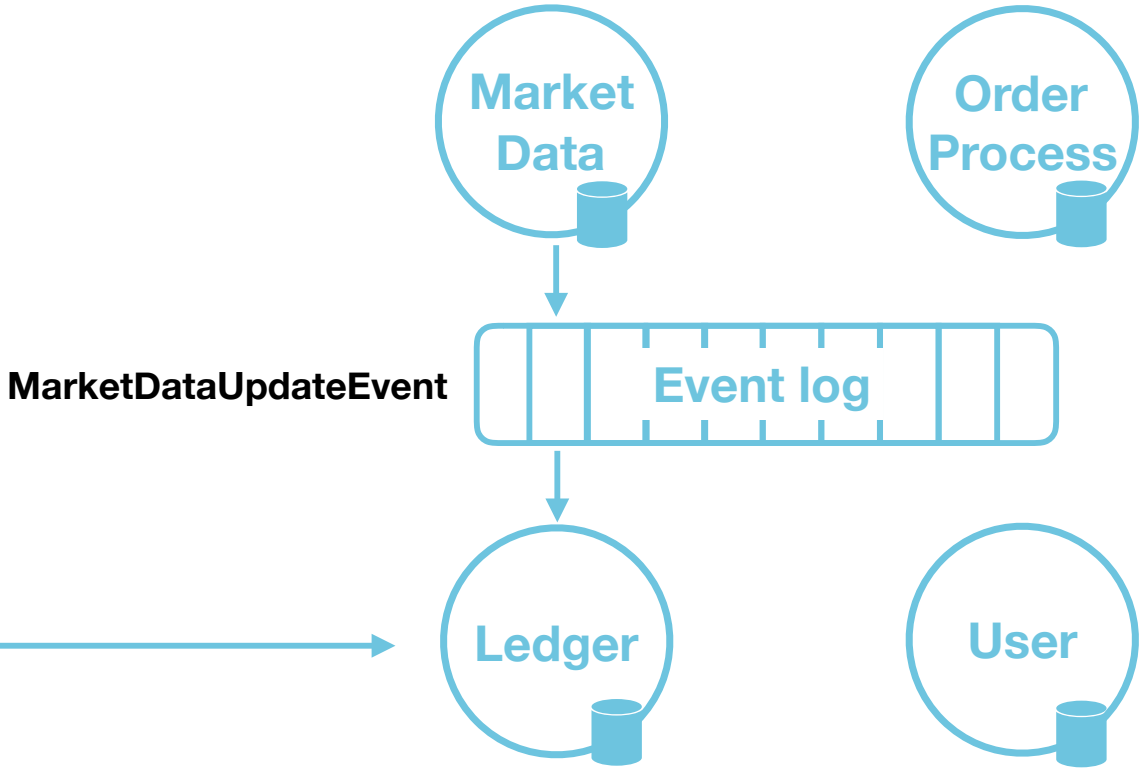
Events

https://www.reactivemanifesto.org

Event Driven Communication

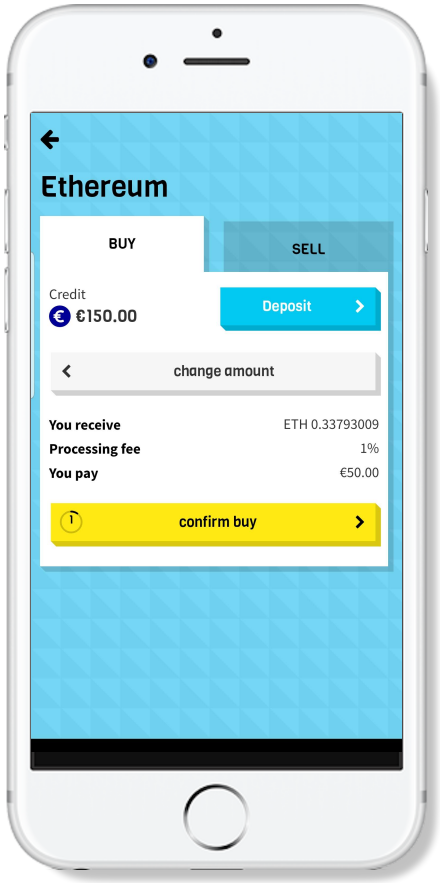
TRIFORK.
...think software

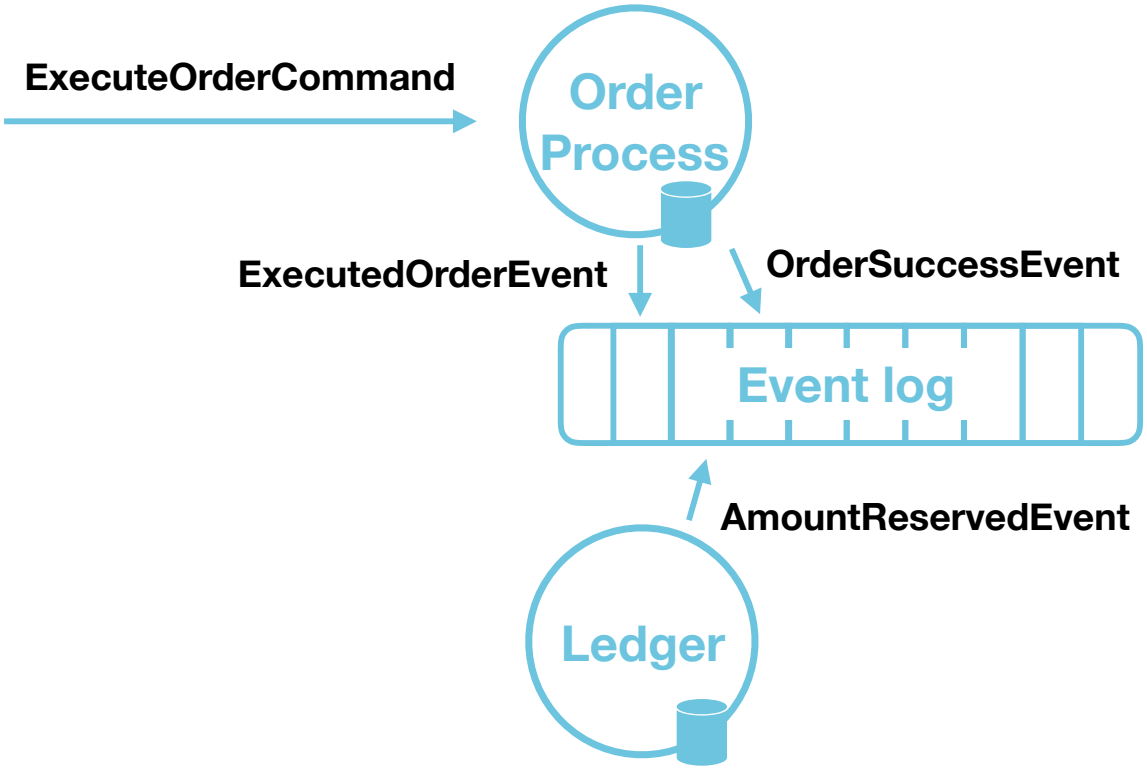




https://martinfowler.com/articles/201701-event-driven.html

TRIFORK.
...think software



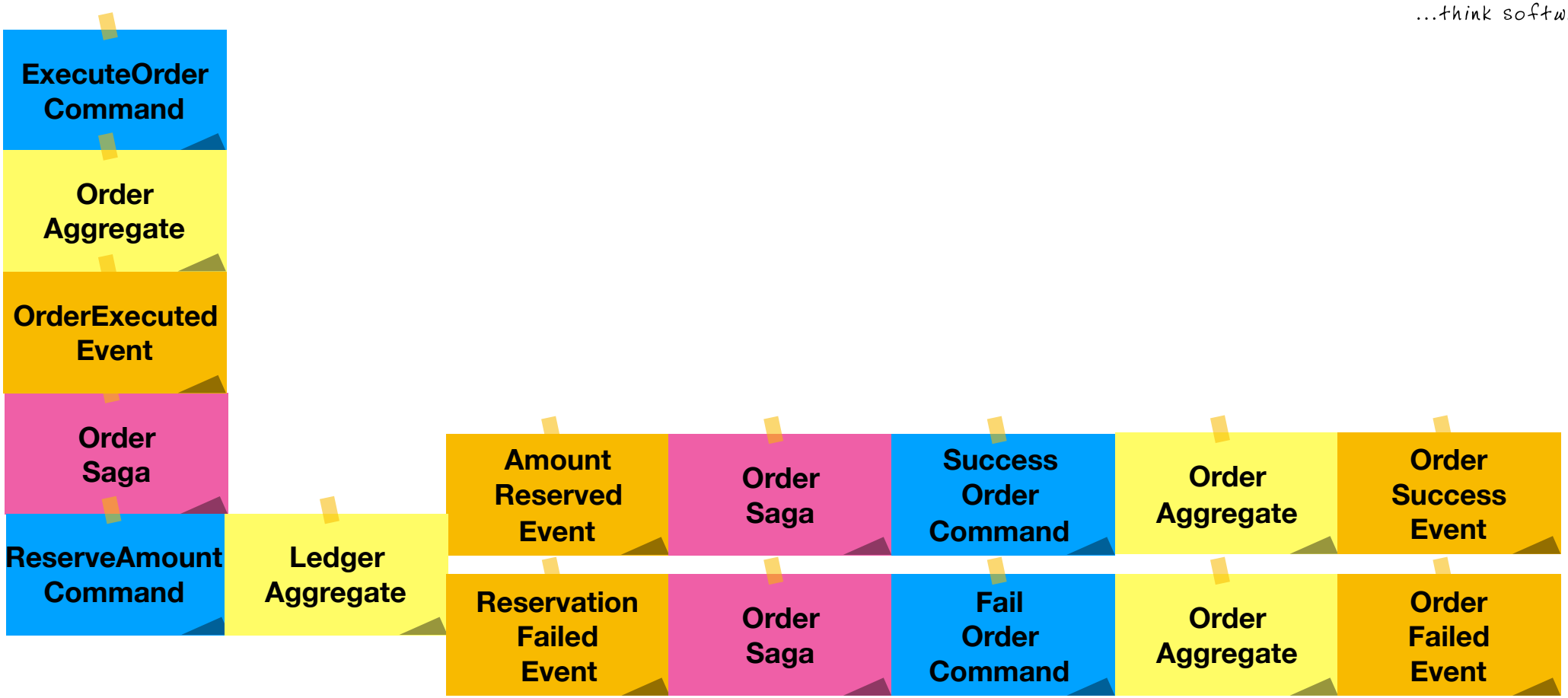


Event Sourcing

TRIFORK.
...think software

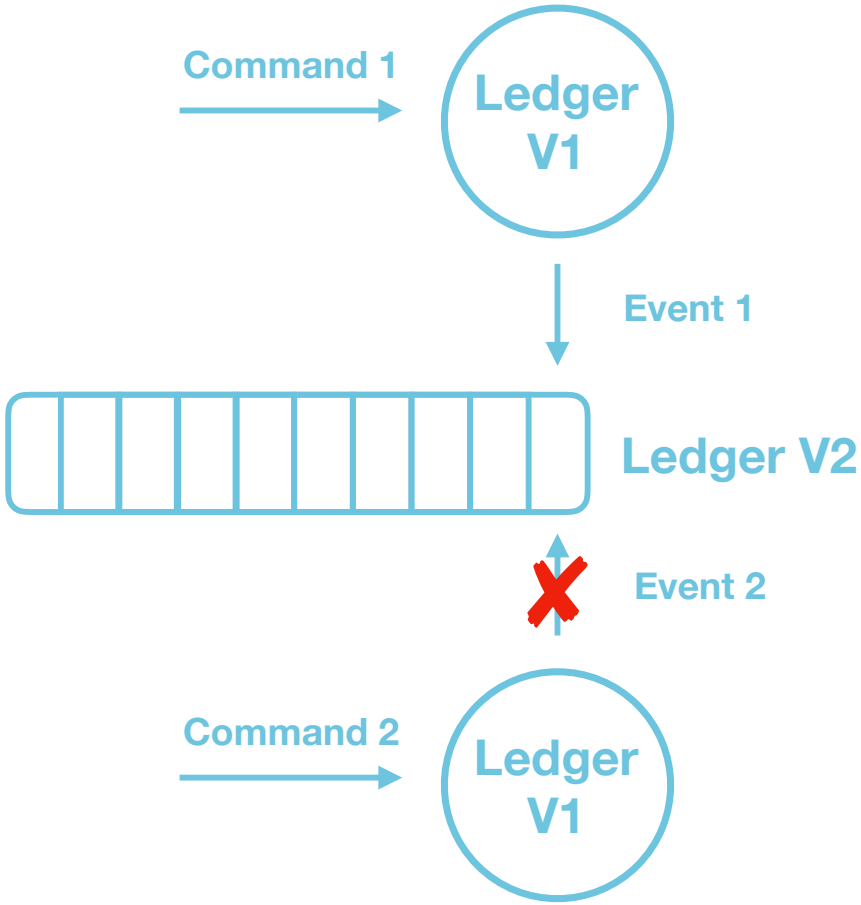
https://www.martinfowler.com/eaaDev/EventSourcing.html

TRIFORK.
...think software



https://www.eventstorming.com

TRIFORK.
...think software



TRIFORK.
...think software

Not Only Events

Not Only Events

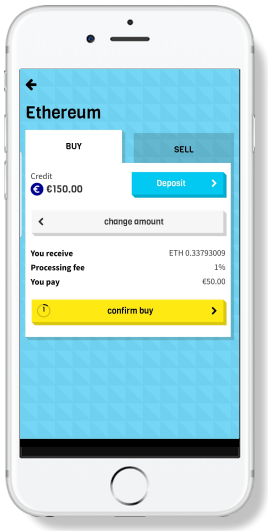
TRIFORK.
...think software

I want the system to do something

Something has happened

I want to know something

TRIFORK.
...think software



Request

Response

REST Facade

Market Data

Order Process

Command

Event

Query

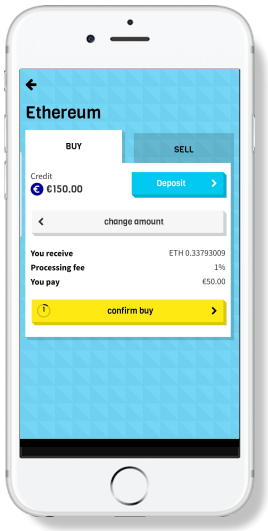
NoEventStore

User

Ledger

Blox Rest Events CQRS/ES Reactive

TRIFORK.
...think software



Request

Response

Spring Webflux

Axon

Axon

Command

Event

Query

AxonServer

Axon

Axon

Blox Rest Events CQRS/ES Reactive

Axon Framework with AxonServer

TRIFORK.
...think software

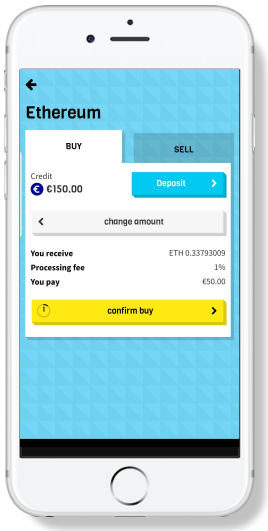
- CQRS + Event Sourcing
- Aggregate & Saga
- Handlers: events, commands, queries
- Message Routing
- NoEventstore
- Location transparency

Distributed systems

TRIFORK.
...think software

- Async communication
- Time is important!

TRIFORK.
...think software



Request

Response

REST Facade

Command

Query

Order Process

Ledger

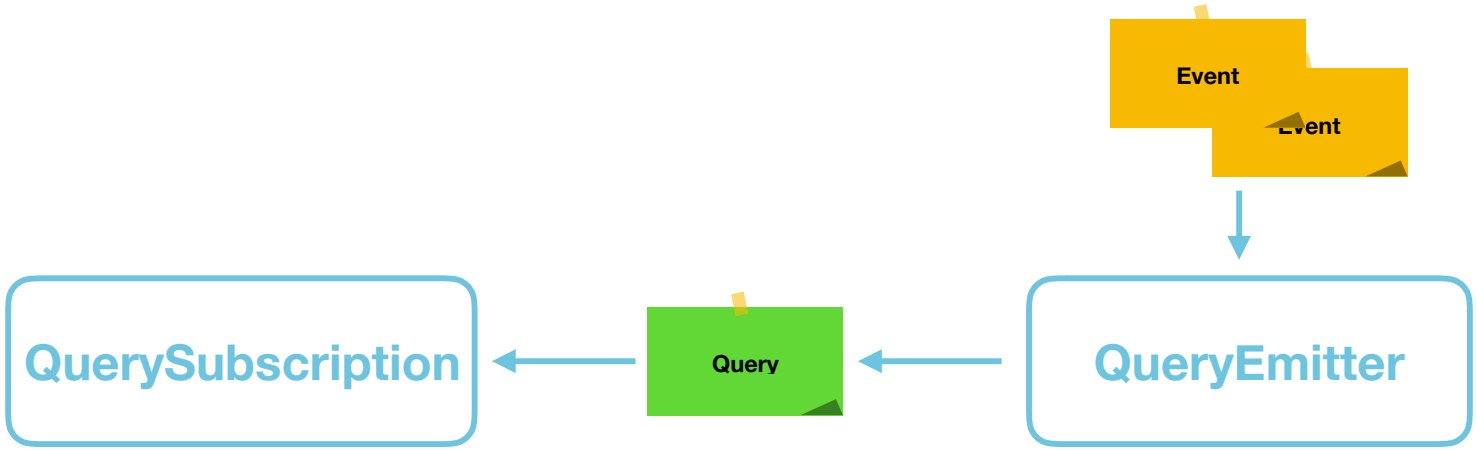
Event

Event

Blox Rest Events CQRS/ES Reactive

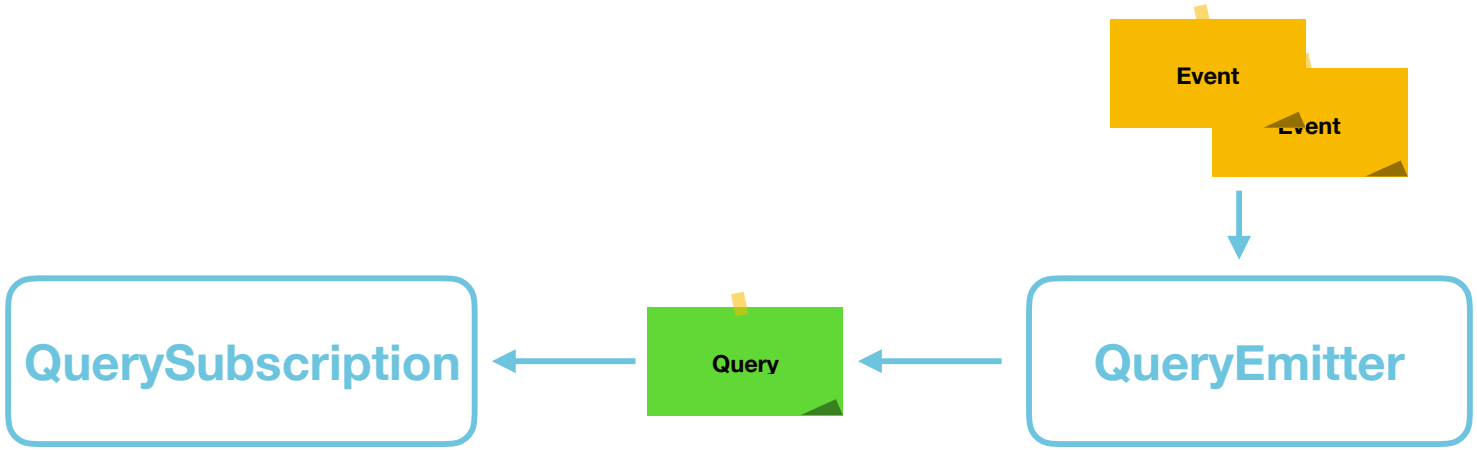
Subscription Queries

TRIFORK.
...think software



Subscription Queries

TRIFORK.
...think software




```
Flux<Order> orders =
    queryGateway.subscriptionQuery(
    new GetOrderQuery(orderId), OrderDto.class,
    OrderDto.class)
        .updates();
```

```
queryUpdateEmitter.emit(GetOrderQuery.class,
    query -> query.getId().equals(order.getId()), order);
```

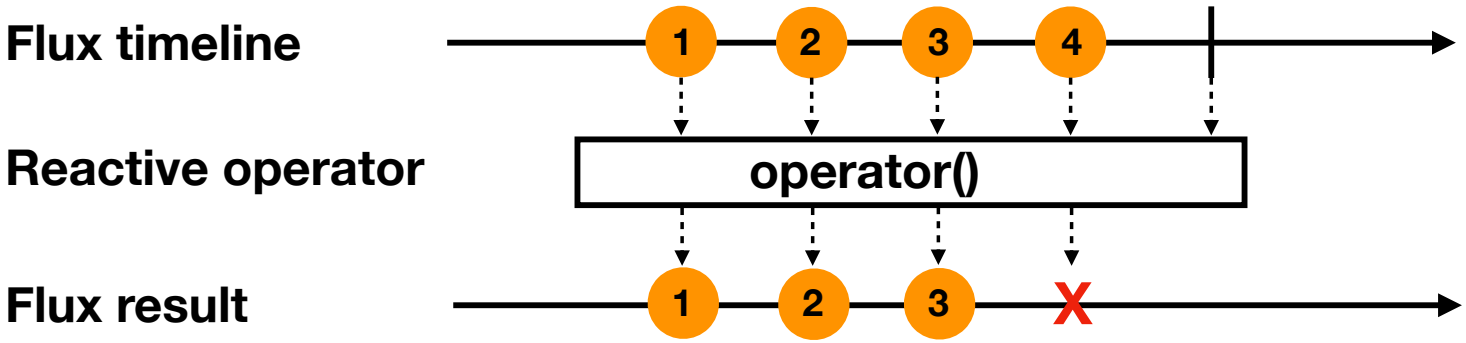
https://axoniq.io/blog-overview/introducing-subscription-queries

TRIFORK.
...think software

Reactive Programming

Reactive programming

TRIFORK.
...think software



Reactive programming

TRIFORK.
...think software

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

Reactive programming

TRIFORK.
...think software

```
Flux.range(0,10)                                    //Create a flux
        .map(i -> i)                                //operator
        .subscribe(v -> System.out.print(v)); //subscriber
```

will print:

0123456789

Reactive operators

TRIFORK.
...think software

Mono	**Source of 0 or 1 items**
Flux	**Source of 0 or N items**
subscribe()	**Activate reactive chain en create a subscription**
map()	**Transform using a synchronous function**
flatMap()	**Transform using an asynchronous function**
timeout()	**Throw error if item doesn't arrive in time**
filter()	**Only allows items which match predicate to pass**
onErrorReturn()	**Return a value on Exception**

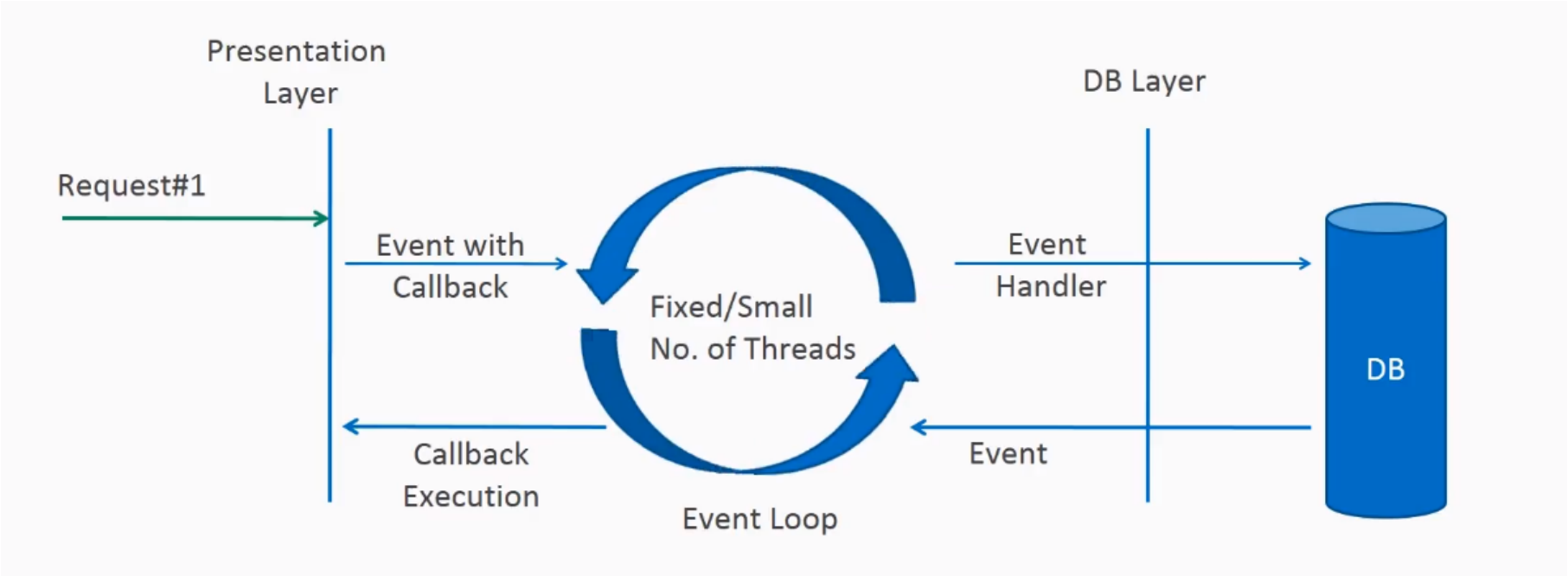
Spring WebFlux

TRIFORK.
...think software

- Annotation mapping
- Functional mapping
- Reactor pattern
- Single thread programming model
- Project Reactor
- Reactor Netty

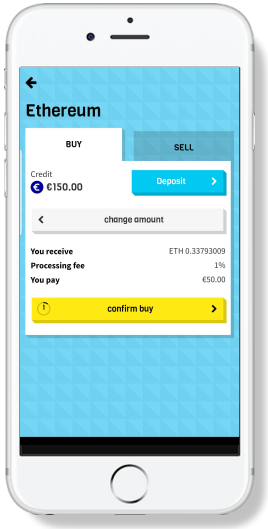
Spring WebFlux

TRIFORK.
...think software



Example

TRIFORK.
...think software



Request

Response

Order Controller

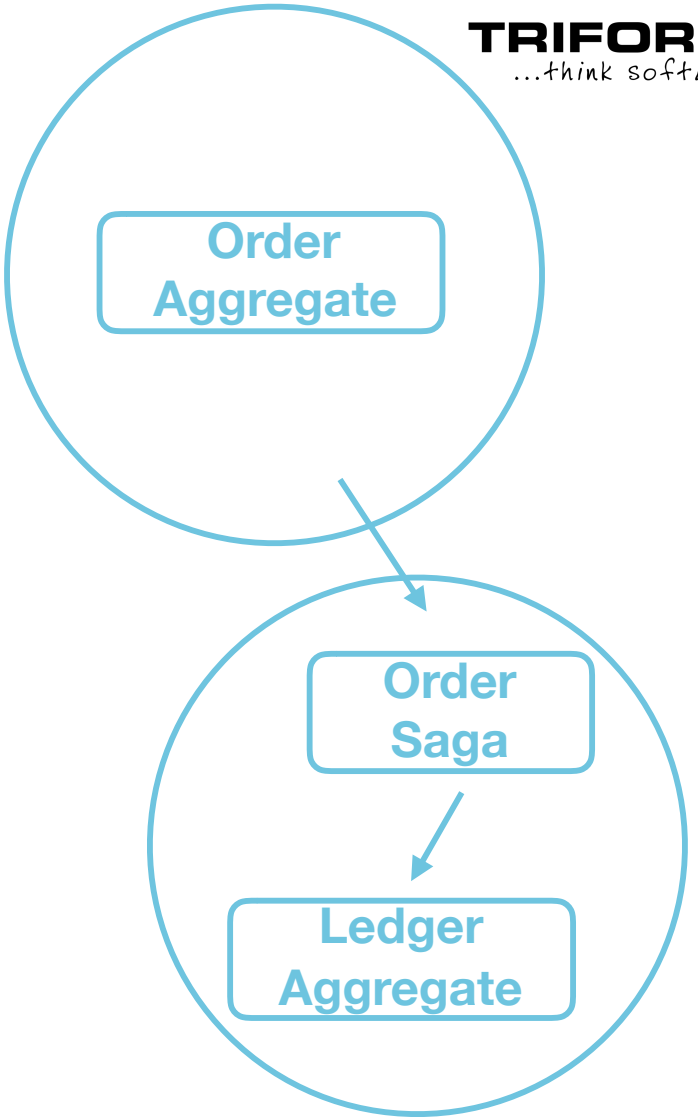
Execute Order Command

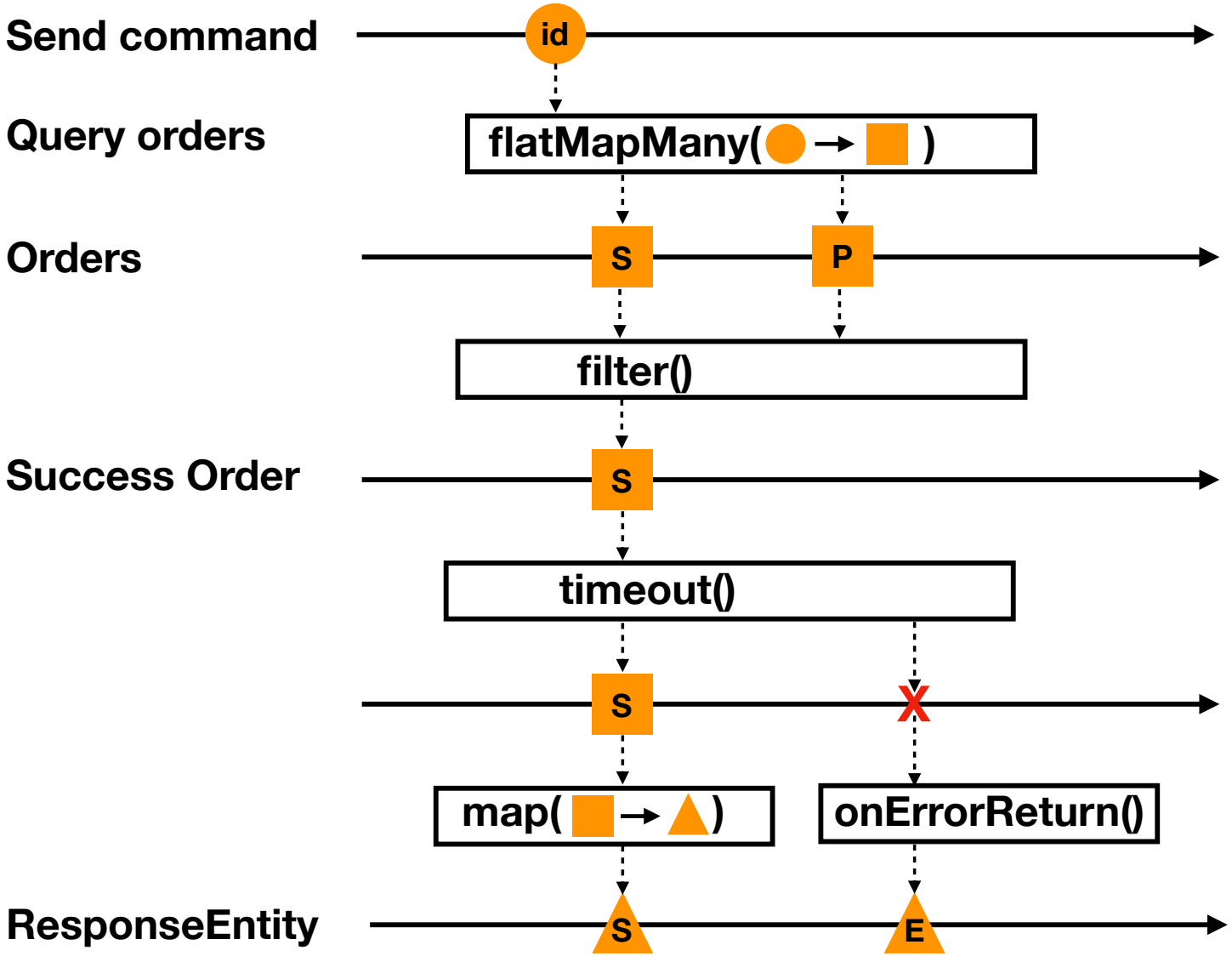
Query

Order Created

Order Pending

Order Success





TRIFORK.
...think software

Example “subscription query”: unit test

TRIFORK. ...think software

```
when(queryResultMock.updates())
        .thenReturn(
                Flux.just(
                        new OrderDto("orderId", EUR, BTC, ONE, TEN, PENDING),
                        new OrderDto("orderId", EUR, BTC, ONE, TEN, COMPLETED)
                ));

ResponseEntity<OrderDto> orderResponse = this.orderController.executeOrder(
    new OrderRequestDto("userId", "quoteId")).block(ofSeconds(1));

assertEquals(orderResponse.getStatusCode(), HttpStatus.OK);
assertEquals(COMPLETED, orderResponse.getBody().getStatus());
```

Example “subscription query”: implementation

TRIFORK. ...think software

```
Flux<OrderDto> orderResponseFlux = this.queryGateway.subscriptionQuery(
    new GetOrderQuery(orderId), OrderDto.class, OrderDto.class)
    .updates();

return sendCommand(new ExecuteOrderCommand(orderId, orderRequest.getUserId()))
        .flatMapMany(id -> orderResponseFlux)
        .filter(order -> !order.getStatus().equals(OrderStatus.PENDING))
        .timeout(ofSeconds(2))
        .next()
        .map(ResponseEntity::ok)
        .onErrorReturn(this::clientException, ResponseEntity.badRequest().build())
        .onErrorReturn(this::serverException, status(SERVICE_UNAVAILABLE).build());
```

Example “subscription query”: unit test timeout

TRIFORK. ...think software

```
when(queryResultMock.updates())
        .thenReturn(Flux.never());

ResponseEntity<OrderDto> orderResponse = this.orderControllerController.executeOrder(
    new OrderRequestDto("userId", "quoteId")).block();

assertEquals(SERVICE_UNAVAILABLE, orderResponse.getStatusCode());
```

Testing

TRIFORK.
...think software

- Blocking subscribe with timeout
- Flux.never() / Flux.error()
- Mockito.verify (mock, times(n))
- Mockito.verify (mock, timeout(ms))
- Reactor StepVerifier
- WebTestClient / @WebFluxTest

```
StepVerifier.create(Flux.just("foo", "bar"))
        .expectNext("foo")
        .expectNext("bar")
        .expectComplete()
        .verify();
```

Debugging

TRIFORK.
...think software

- Hooks.onOperatorDebug() (not for prod)
- ReactorDebugAgent (reactor-tools)
- Lambda breakpoints (IDE)
- Flux.log() / doOnNext()
- BlockHound

https://spring.io/blog/2019/03/28/reactor-debugging-experience

Blocking code

TRIFORK.
...think software

```
java.lang.Error: Blocking call! java.lang.Thread.yield
    at reactor.blockhound.BlockHound$Builder.lambda$new$0(BlockHound.java:154) ~[blockhound-1.0.0.M3.jar:na]
    at reactor.blockhound.BlockHound$Builder.lambda$install$8(BlockHound.java:254) ~[blockhound-1.0.0.M3.jar:na]
    at reactor.blockhound.BlockHoundRuntime.checkBlocking(BlockHoundRuntime.java:43) ~[blockhound-1.0.0.M3.jar:na]
    at java.lang.Thread.yield(Thread.java) ~[na:1.8.0_172]
    at java.util.concurrent.ConcurrentHashMap.initTable(ConcurrentHashMap.java:2227) ~[na:1.8.0_172]
    at java.util.concurrent.ConcurrentHashMap.putVal(ConcurrentHashMap.java:1017) ~[na:1.8.0_172]
    at java.util.concurrent.ConcurrentHashMap.put(ConcurrentHashMap.java:1006) ~[na:1.8.0_172]
```

```
Caused by: java.lang.Error: Blocking call! java.io.FileInputStream#readBytes
    at reactor.blockhound.BlockHound$Builder.lambda$new$0(BlockHound.java:154)
    at reactor.blockhound.BlockHound$Builder.lambda$install$8(BlockHound.java:254)
    at reactor.blockhound.BlockHoundRuntime.checkBlocking(BlockHoundRuntime.java:43)
    at java.io.FileInputStream.readBytes(FileInputStream.java)
    at java.io.FileInputStream.read(FileInputStream.java:255)
    at sun.security.provider.NativePRNG$RandomIO.readFully(NativePRNG.java:424)
    at sun.security.provider.NativePRNG$RandomIO.ensureBufferValid(NativePRNG.java:525)
    at sun.security.provider.NativePRNG$RandomIO.implNextBytes(NativePRNG.java:544)
    at sun.security.provider.NativePRNG$RandomIO.access$400(NativePRNG.java:331)
    at sun.security.provider.NativePRNG.engineNextBytes(NativePRNG.java:220)
    at java.security.SecureRandom.nextBytes(SecureRandom.java:468)
    at java.util.UUID.randomUUID(UUID.java:145)
```

TRIFORK.
...think software

Conclusion

Conclusion

TRIFORK.
...think software

- Events are 1 trait of Reactive Systems (Manifesto)
- Events are Async
- Reactive provides a concise async programming model
- Functional operators
- Testing is not that hard…
- … but debugging is.
- DIY / Workarounds

https://github.com/erwindeg/eventdriven-examples

Please

Remember to rate this session

Thank you!

goto; amsterdam

@GOTOamst

gotoams.nl