CQRS & EVENT SOURCING

IN THE WILD

Michiel Rook – @michieltcs

- Developer, consultant, trainer, speaker
- @michieltcs

MAKE.IO FourScouts D·W·A Dutch Web Alliance

YOU

RAISE YOUR HAND

IF YOU HAVE

RAISE YOUR HAND

IF YOU HAVE

read CQRS / Event Sourcing theory

RAISE YOUR HAND
IF YOU HAVE

read CQRS / Event Sourcing theory

followed a tutorial, built a hobby project

RAISE YOUR HAND
IF YOU HAVE

read CQRS / Event Sourcing theory

followed a tutorial, built a hobby project

used it in production

Axon Framework

QUICK RECAP

Event Sourcing ensures that all changes to application state are stored as a sequence of events.

-Martin Fowler

ACTIVE RECORD VS. EVENT SOURCING

Account Id	Account number	Balance
1234	12345678	€50.00
...	...	...

Account Opened	
Account Id	1234
Account number	12345678

Money Deposited	
Account Id	1234
Amount	€100.00

Money Withdrawn	
Account Id	1234
Amount	€50.00

COMMANDS TO EVENTS

Deposit Money	
Account Id	1234
Amount	€100.00

```
class DepositMoney {
    @TargetAggregateIdentifier
    public String accountId;
    public BigDecimal amount;
}
```

COMMANDS TO EVENTS

Deposit Money	
Account Id	1234
Amount	€100.00


```
@CommandHandler
public void depositMoney(DepositMoney command) {
    apply(new MoneyDeposited(
        command.getAccountId(),
        command.getAmount(),
        ZonedDateTime.now()));
}
```

COMMANDS TO EVENTS

Deposit Money	
Account Id	1234
Amount	€100.00

Money Deposited	
Account Id	1234
Amount	€100.00

```
class MoneyDeposited {
    public String accountId;
    public BigDecimal amount;
    public ZonedDateTime timestamp;
}
```

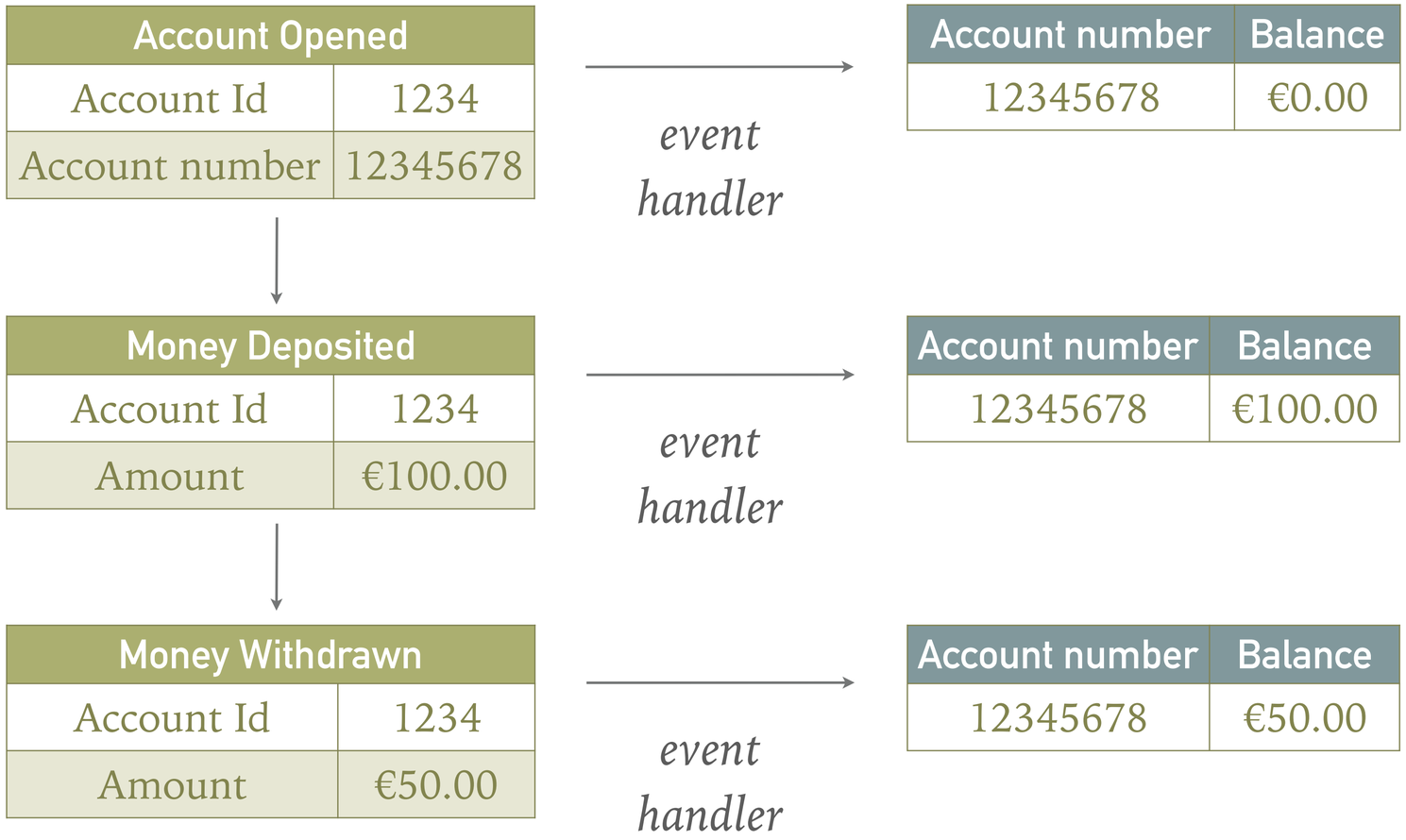
AGGREGATES

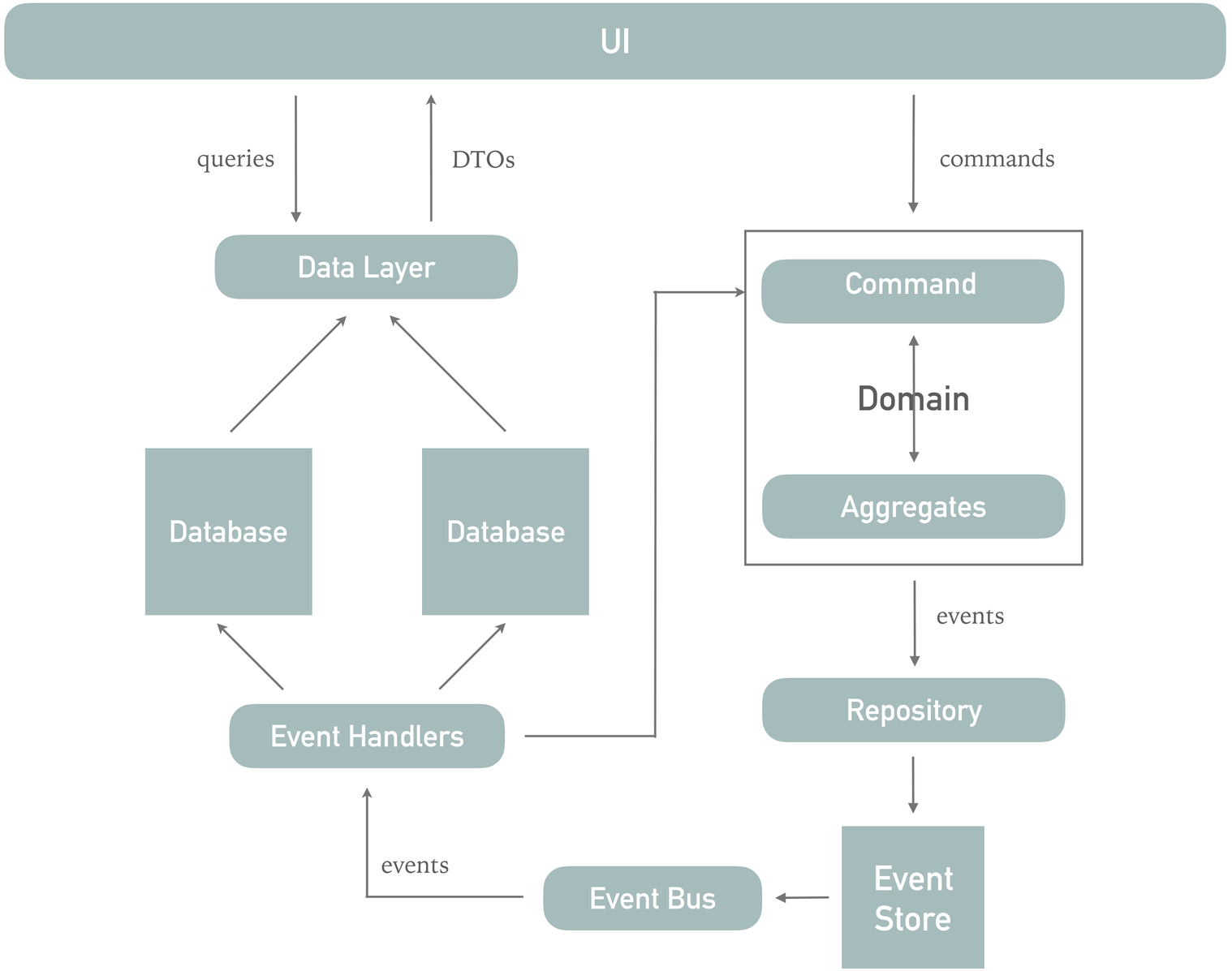
```
class BankAccount {
    @AggregateIdentifier
    public String accountId;
    public String accountNumber;
    public BigDecimal balance;

    // ...
    @EventHandler
    public void accountOpened(AccountOpened event) {
        this.accountId = event.getAccountId();
        this.accountNumber = event.getAccountNumber();
        this.balance = BigDecimal.valueOf(0);
    }

    @EventHandler
    public void moneyDeposited(MoneyDeposited event) {
        this.balance = this.balance.add(event.getAmount());
    }
}
```

AGGREGATE STATE



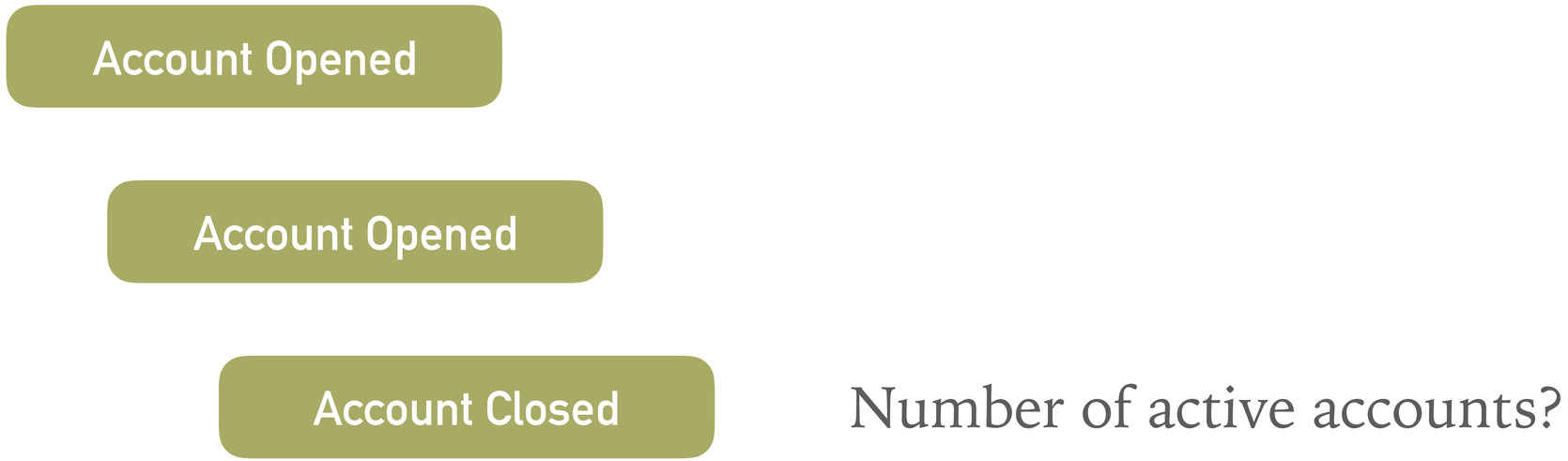

UI
queries
DTOs
commands
Data Layer
Command
Domain
Aggregates
Database
Database
events
Repository
Event Handlers
events
Event Bus
Event Store

REPLAYS AND REBUILDS

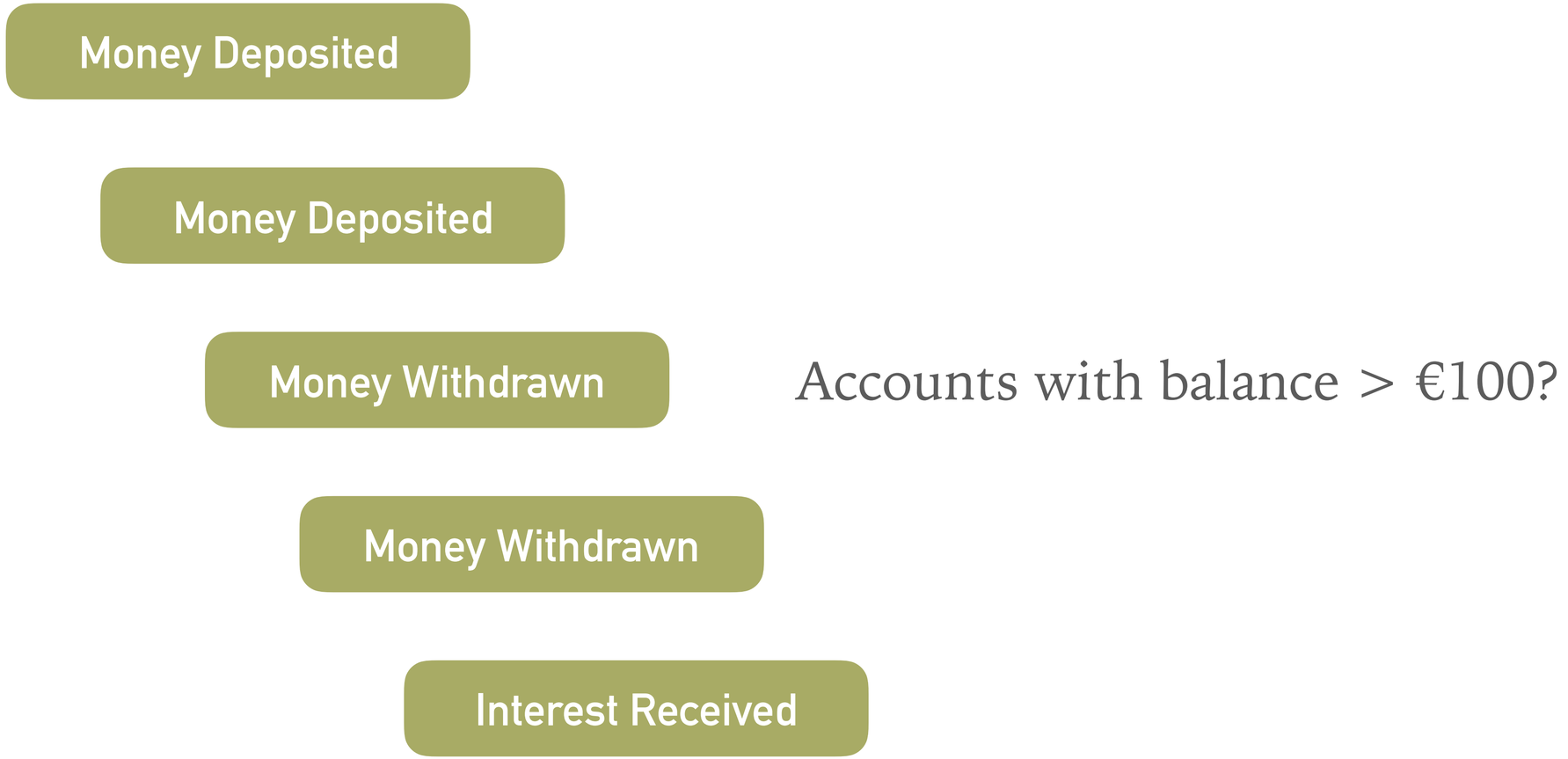
ANSWERING QUERIES

BASED ON EVENTS

QUERIES

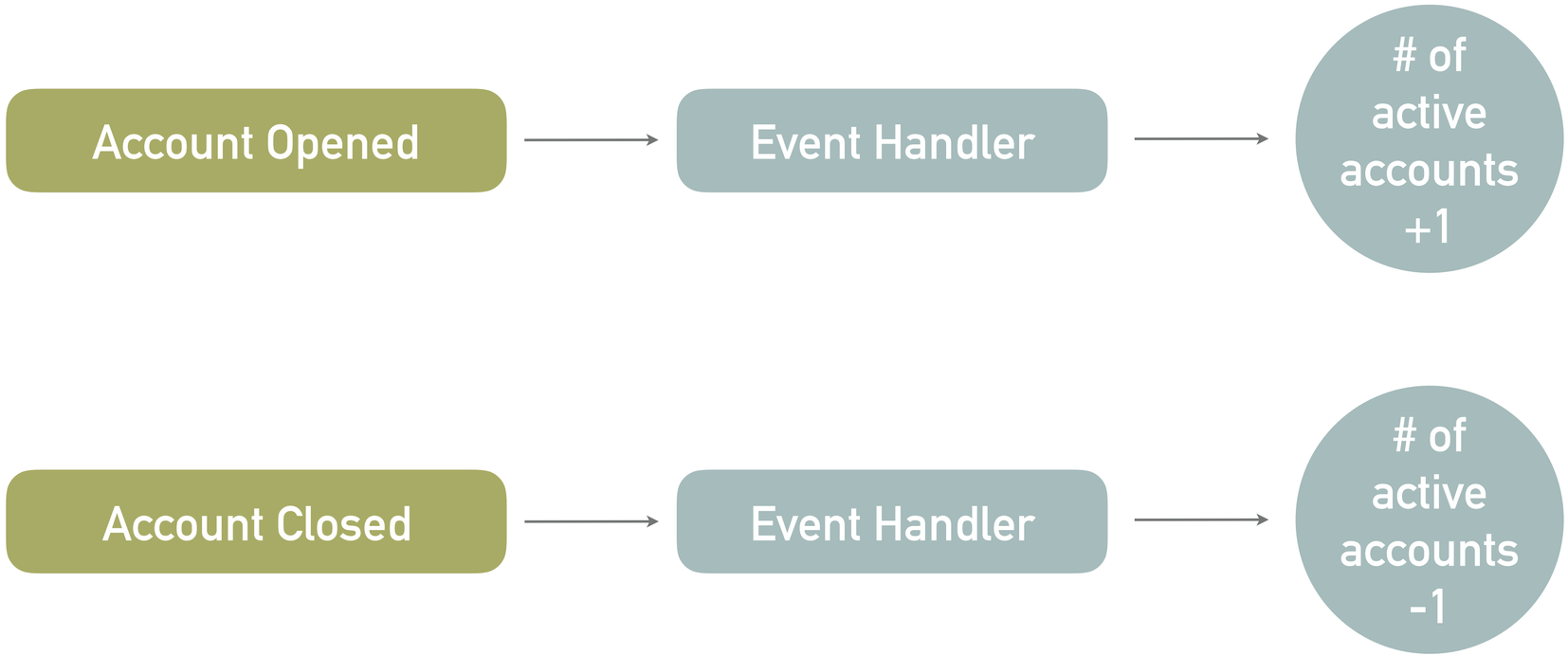

Account Opened
Account Opened
Account Closed
Number of active accounts?

QUERIES



Accounts with balance > €100?

PROJECTION


Account Opened
Event Handler
# of active accounts +1
Account Closed
Event Handler
# of active accounts -1

PROJECTION

NEW PROJECTION

NEW STRUCTURE

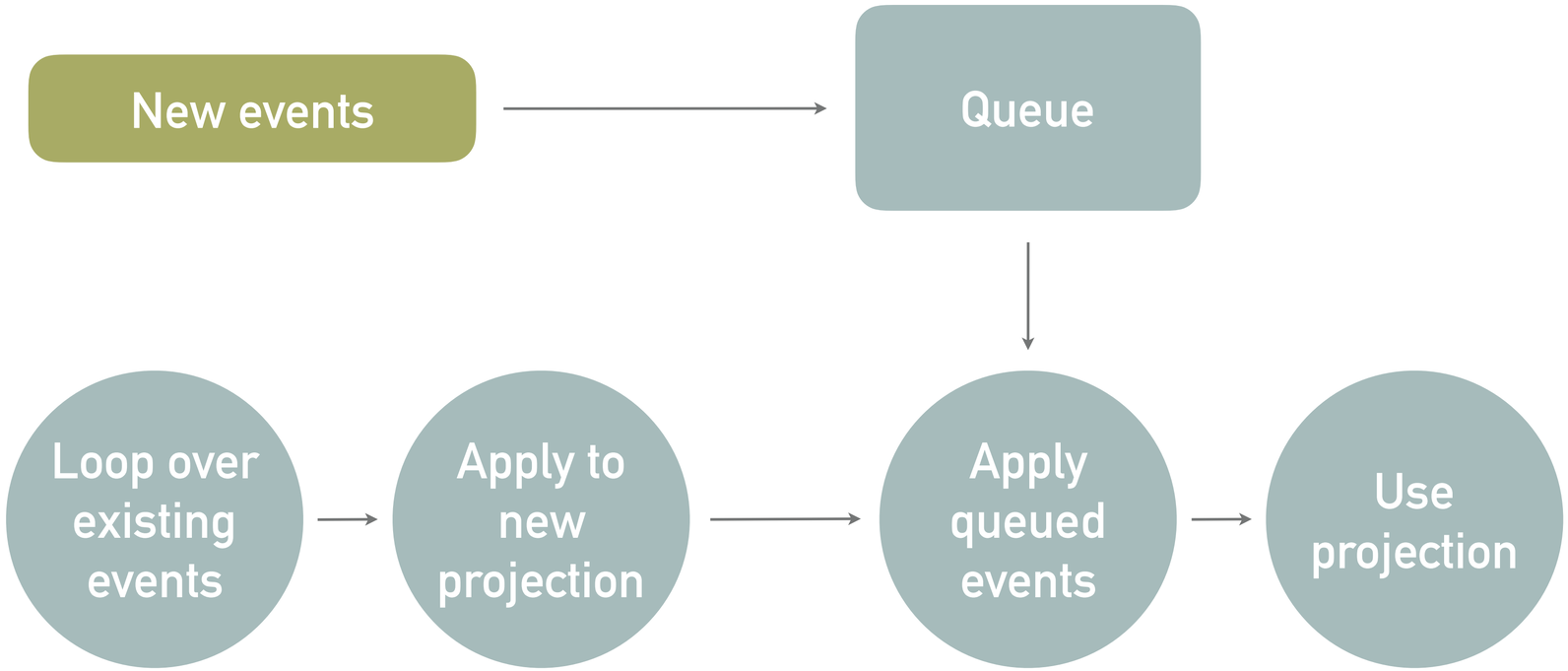
BASED ON EXISTING EVENTS

REBUILDING

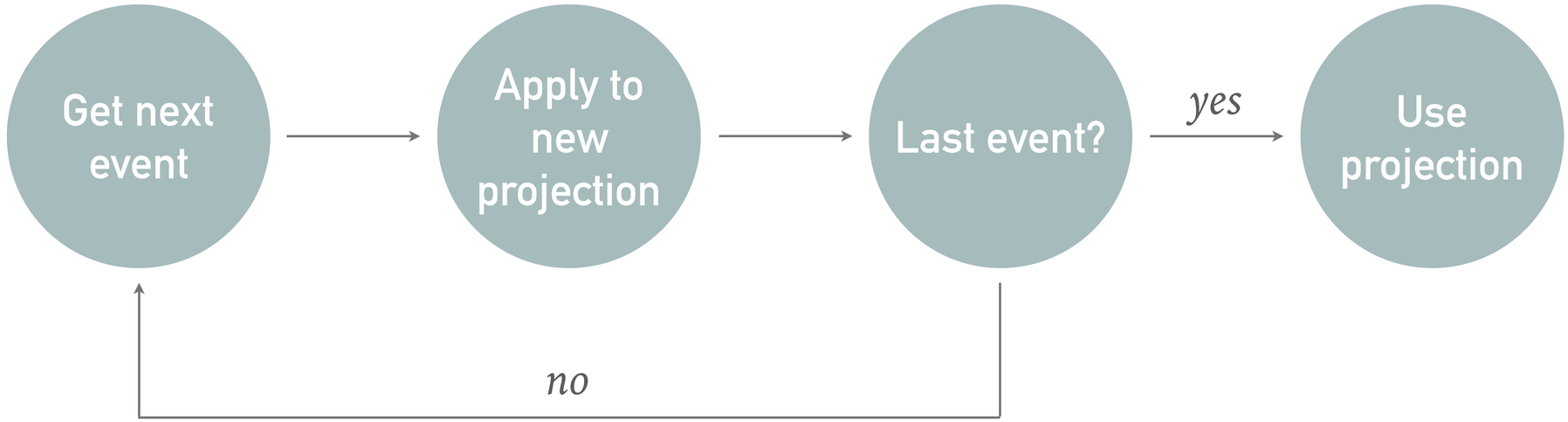

Stop app
Loop over events
Apply to projection
Start app
Cleanup

ZERO DOWNTIME

ZERO DOWNTIME



ZERO DOWNTIME



LONG RUNNING REBUILDS?

IN MEMORY

DISTRIBUTED

PARTIAL

BACKGROUND TASK

TRACKING EVENT PROCESSOR

```
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
public @interface TrackedProjection {
}
```

TRACKING EVENT PROCESSOR

```
@Configuration
public class ProjectionsConfiguration {
    @Autowired
    private EventHandlingConfiguration eventHandlingConfiguration;

    @PostConstruct
    public void startTrackingProjections() throws ClassNotFoundException {
        // ...
    }
}
```

TRACKING EVENT PROCESSOR

```
ClassPathScanningCandidateComponentProvider scanner =
    new ClassPathScanningCandidateComponentProvider(false);

scanner.addIncludeFilter(new AnnotationTypeFilter(TrackedProjection.class));

for (BeanDefinition bd : scanner.findCandidateComponents("org.demo")) {
    Class<?> aClass = Class.forName(bd.getBeanClassName());
    ProcessingGroup processingGroup = aClass.getAnnotation(ProcessingGroup.class);

    String name = Optional.ofNullable(processingGroup)
        .map(ProcessingGroup::value)
        .orElse(aClass.getPackage().getName());

    eventHandlingConfiguration.registerTrackingProcessor(name);
}
```

EVENT VERSIONING

NEW BUSINESS REQUIREMENTS

CHANGING VIEW ON EVENTS

IRRELEVANT

DIFFERENT FIELDS

WRONG NAME

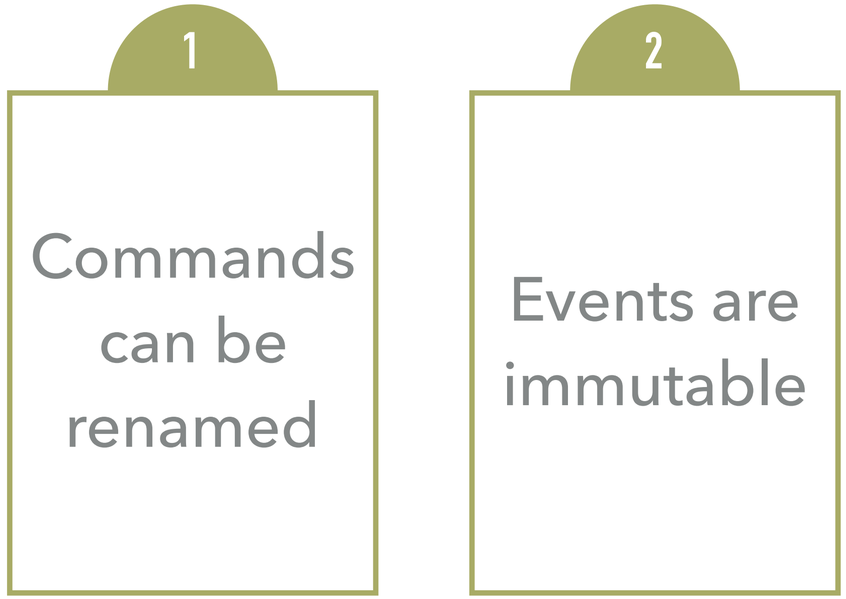
TOO COARSE

TOO FINE

SUPPORT YOUR LEGACY?

1

Commands can be renamed


1
Commands can be renamed
2
Events are immutable

1

Commands can be renamed

2

Events are immutable

3

Correct old events with new events

COMPENSATING ACTIONS

```
class MoneyWithdrawn {
    String accountId;
    BigDecimal amount;
}
```

Typo: too much withdrawn!

```
class WithdrawalRolledBack {
    String accountId;
    BigDecimal amount;
}
```

COMPENSATING ACTIONS

```
class AccountOpened {
    String accountId;
    String accountNumber;
}
```

Duplicate account number!

```
class DuplicateAccountClosed {
    String accountId;
}
```

UPCASTING

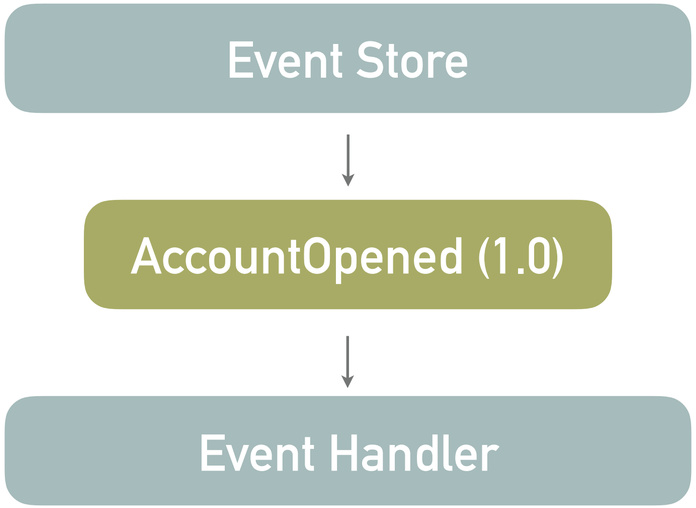
UPCASTING

Event Store

UPCASTING

```
@Revision("1.0")
@Value
class AccountOpened {
    String accountId;
    String accountNumber;
}
```

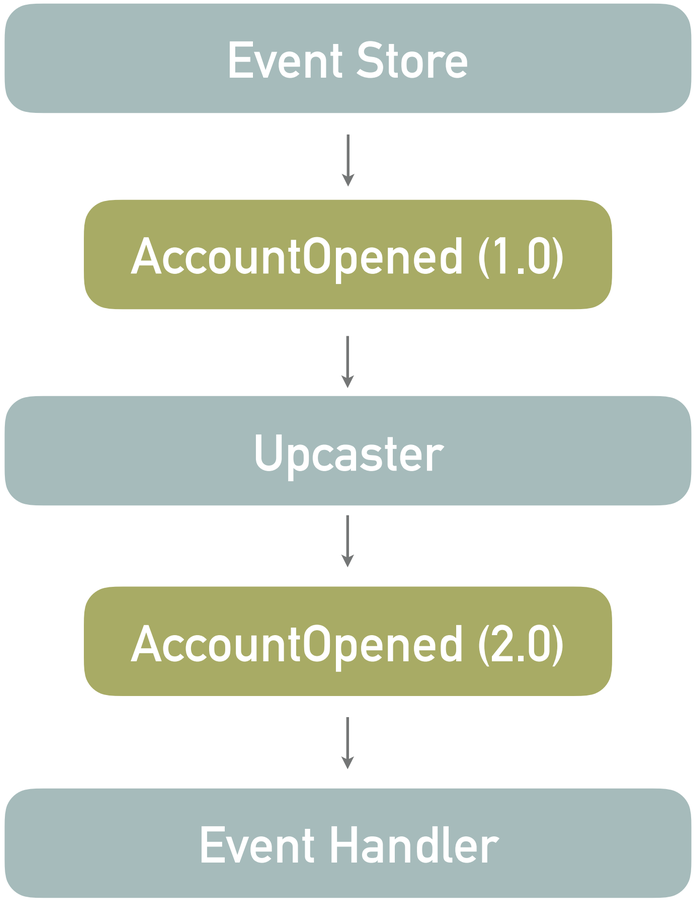
UPCASTING



UPCASTING

```
@Revision("2.0")
@Value
class AccountOpened {
    String accountId;
    String accountNumberIban;
}
```

UPCASTING


Event Store
AccountOpened (1.0)
Upcaster
AccountOpened (2.0)
Event Handler

UPCASTING

```
private static SimpleSerializedType targetType =
    new SimpleSerializedType(AccountOpened.class.getTypeName(), "1.0");

private static SimpleSerializedType outputType =
    new SimpleSerializedType(AccountOpened.class.getTypeName(), "2.0");
```

UPCASTING

```
public Stream<IntermediateEventRepresentation> upcast(
    Stream<IntermediateEventRepresentation> intermediateRepresentations) {
    return intermediateRepresentations.map(evt -> {
        if (!evt.getType().equals(targetType)) {
            return evt;
        }

        return evt.upcastPayload(outputType, Document.class, document -> {
            Element rootElement = document.getRootElement();
            Element accountNumberElement = rootElement.element("accountNumber");
            rootElement.remove(accountNumberElement);
            rootElement.addElement("accountNumberIban")
                .setText(toIban(accountNumberElement.getText()));
            return document;
        });
    });
}
```

VERSIONED EVENT STORE

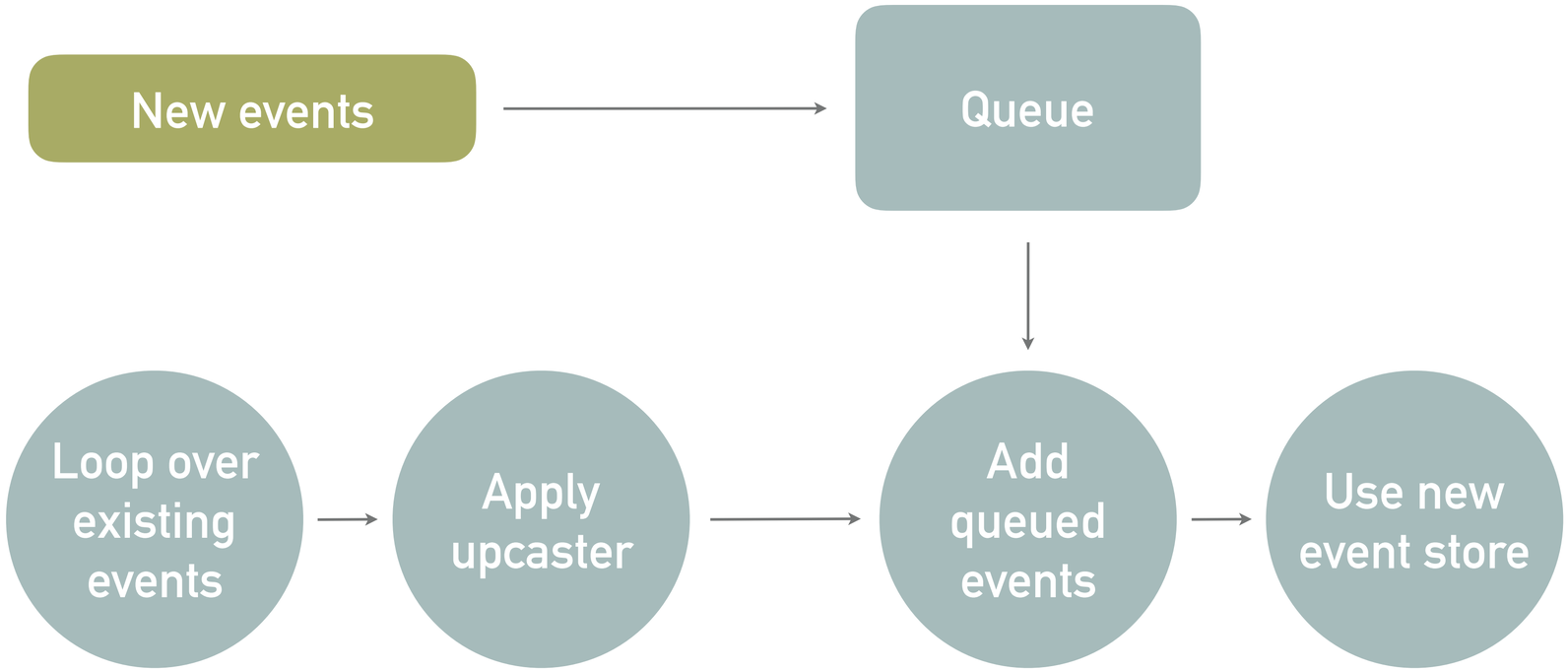
VERSIONED EVENT STORE

events_v1

```
[
  {
    "id": "12345678",
    "type": "AccountOpened",
    "aggregateType": "Account",
    "aggregateIdentifier": "1234",
    "sequenceNumber": 0,
    "payloadRevision": "1.0",
    "payload": { ... },
    "timestamp": ...
    ...
  },
  ...
]
```

COPY & REPLACE

VERSIONED EVENT STORE



VERSIONED EVENT STORE

events_v2

```
[
  {
    "id": "12345678",
    "type": "AccountOpened",
    "aggregateType": "Account",
    "aggregateIdentifier": "1234",
    "sequenceNumber": 0,
    "payloadRevision": "2.0",
    "payload": { ... },
    "timestamp": ...
    ...
  },
  ...
]
```

GDPR

"RIGHT TO ERASURE"

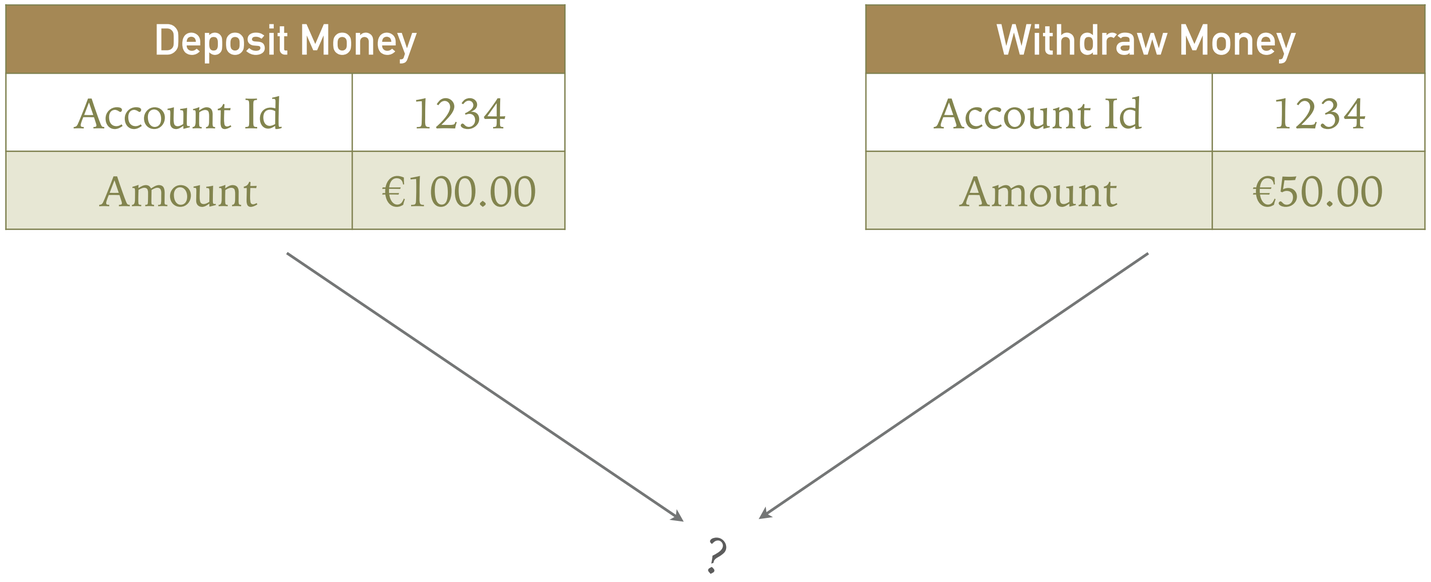
PERSONALLY IDENTIFIABLE INFORMATION

ANONYMIZED OR REMOVED

IMMUTABLE EVENTS?

CONCURRENCY

CONCURRENT COMMANDS



PESSIMISTIC LOCKING

Deposit Money	
Account Id	1234
Amount	€100.00

lock →

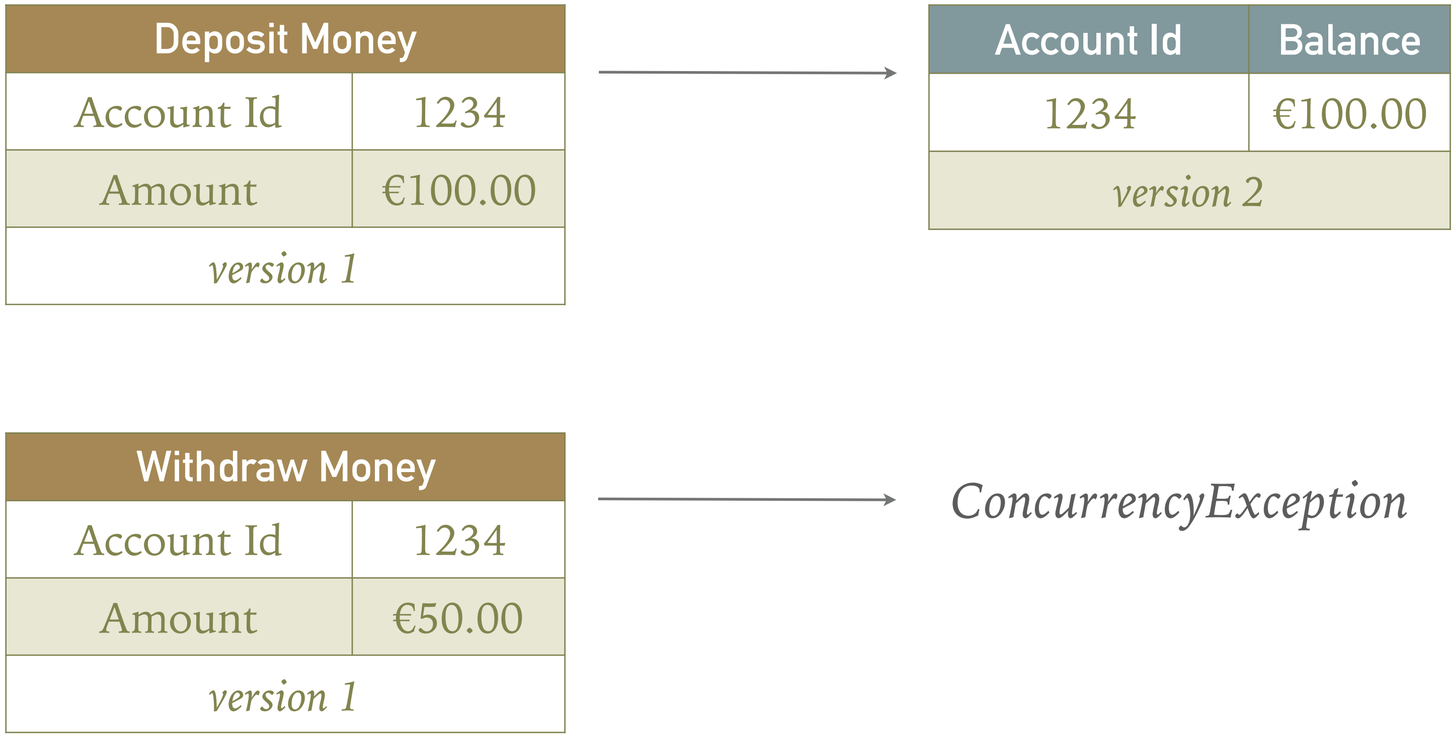
Account Id	Balance
1234	€100.00

Withdraw Money	
Account Id	1234
Amount	€50.00

wait for lock

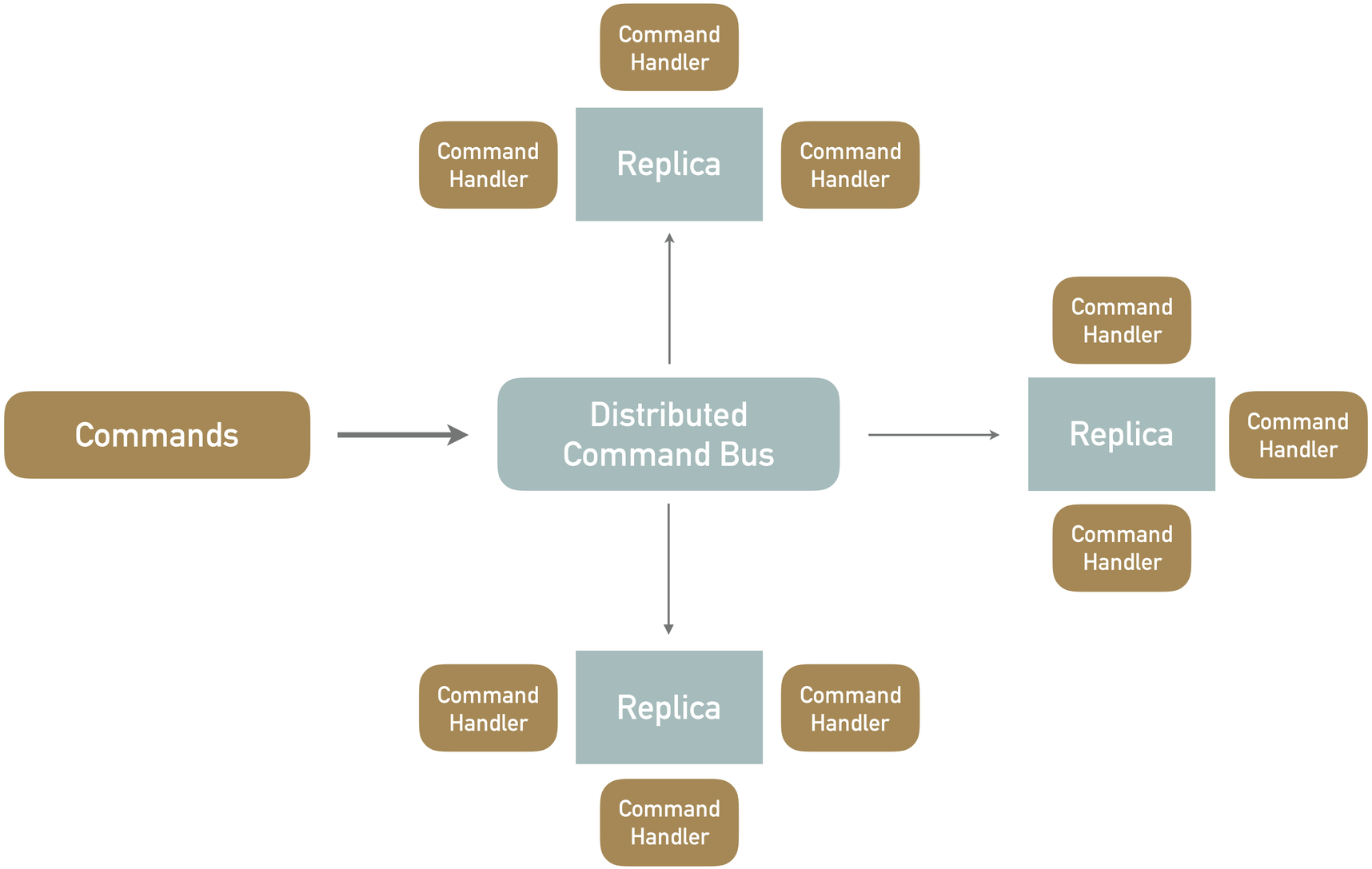
Account Id	Balance
1234	€50.00

OPTIMISTIC LOCKING



MULTIPLE INSTANCES

MULTIPLE INSTANCES



SCALE

PERFORMANCE

PERFORMANCE

Server

Database

PERFORMANCE

Server

Database

Framework

PERFORMANCE

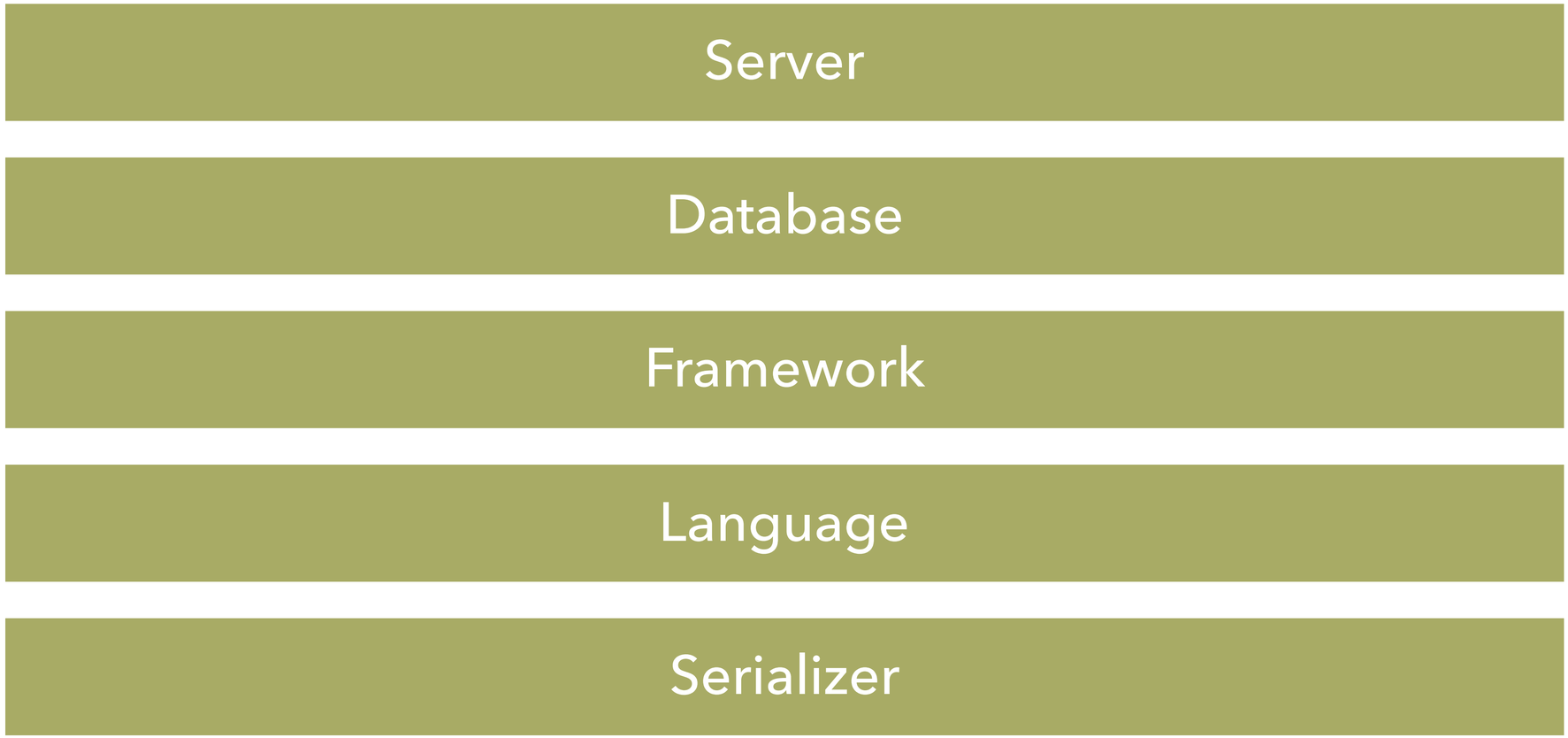
Server

Database

Framework

Language

PERFORMANCE



STORAGE

#events

STORAGE

#events

#aggregates

STORAGE

#events

#aggregates

#events per aggregate

STORAGE

#events

#aggregates

#events per aggregate

Serializer

STORAGE

#events

#aggregates

#events per aggregate

Serializer

Payload

SNAPSHOTS

Events	
...	...
997	Account Opened
998	Money Deposited
999	Money Withdrawn
1000	Money Deposited

SNAPSHOTS

Events	
...	...
997	Account Opened
998	Money Deposited
999	Money Withdrawn
1000	Money Deposited
1001	Money Withdrawn

SNAPSHOTS

Events	
...	...
997	Account Opened
998	Money Deposited
999	Money Withdrawn
1000	Money Deposited
1001	Money Withdrawn

Events	
...	...
~~997~~	~~Account Opened~~
~~998~~	~~Money Deposited~~
~~999~~	~~Money Withdrawn~~
~~1000~~	~~Money Deposited~~
SNAPSHOT	
1001	Money Withdrawn

CLOSING WORDS

CQRS + ES = AWESOME

NO SILVER BULLET

COMPLEXITY

INFRASTRUCTURE

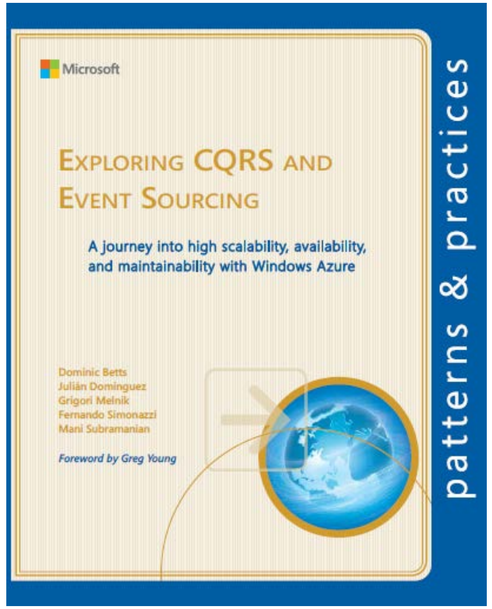
AUDIT TRAIL

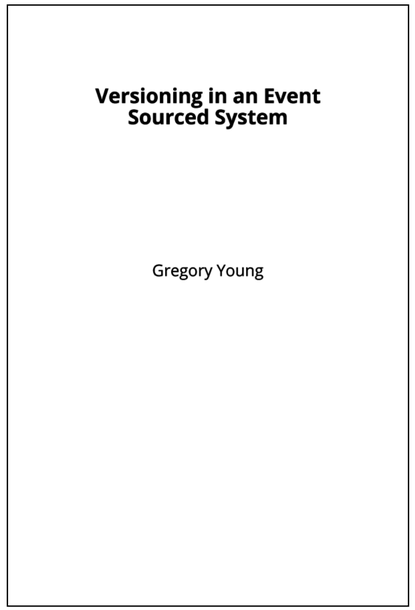
SCALABILITY

TESTING

DOMAIN FIT

LITERATURE


Microsoft
Exploring CQRS and Event Sourcing
A journey into high scalability, availability, and maintainability with Windows Azure
Dominic Betts
Julián Dominguez
Grigori Melnik
Fernando Simonazzi
Mani Subramanian
Foreword by Greg Young
patterns & practices


Versioning in an Event Sourced System
Gregory Young

THANK YOU!

@michieltcs / mrook@fourscouts.nl

www.michielrook.nl

MAKE.IO FourScouts