# CQRS

## Command Query Responsibility Segregation

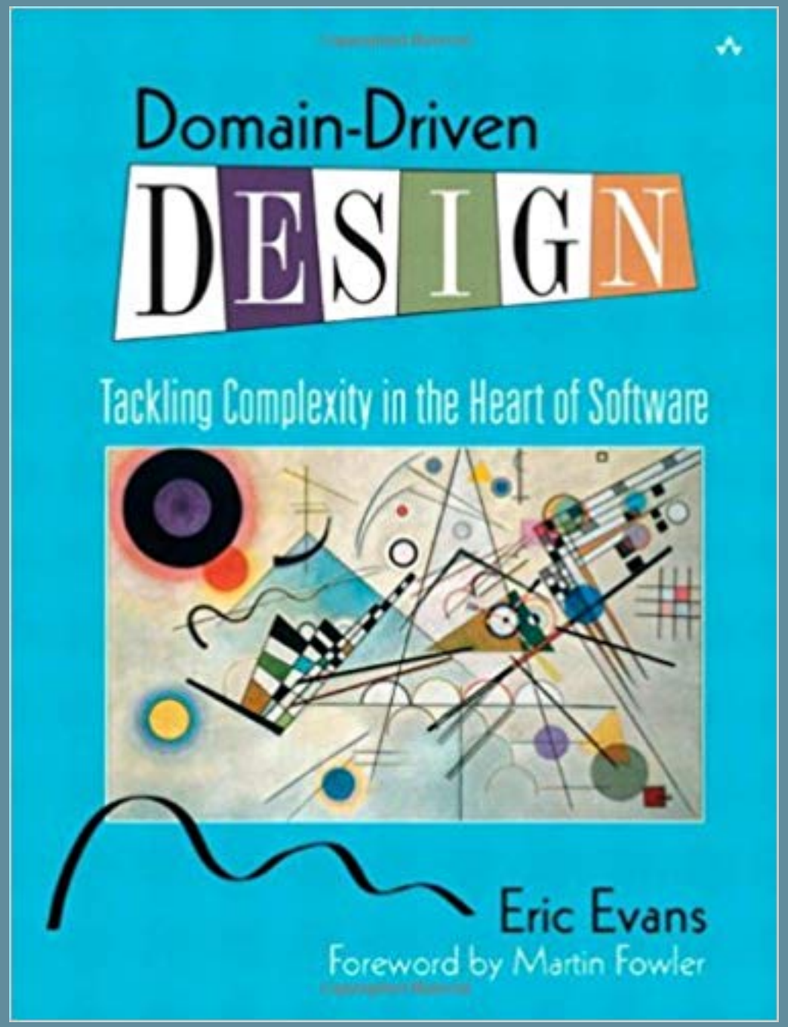
# About Me

Seth Kraut
Senior Software Engineer @ Upstart

Slides and more are available at http://tech.sethkraut.com

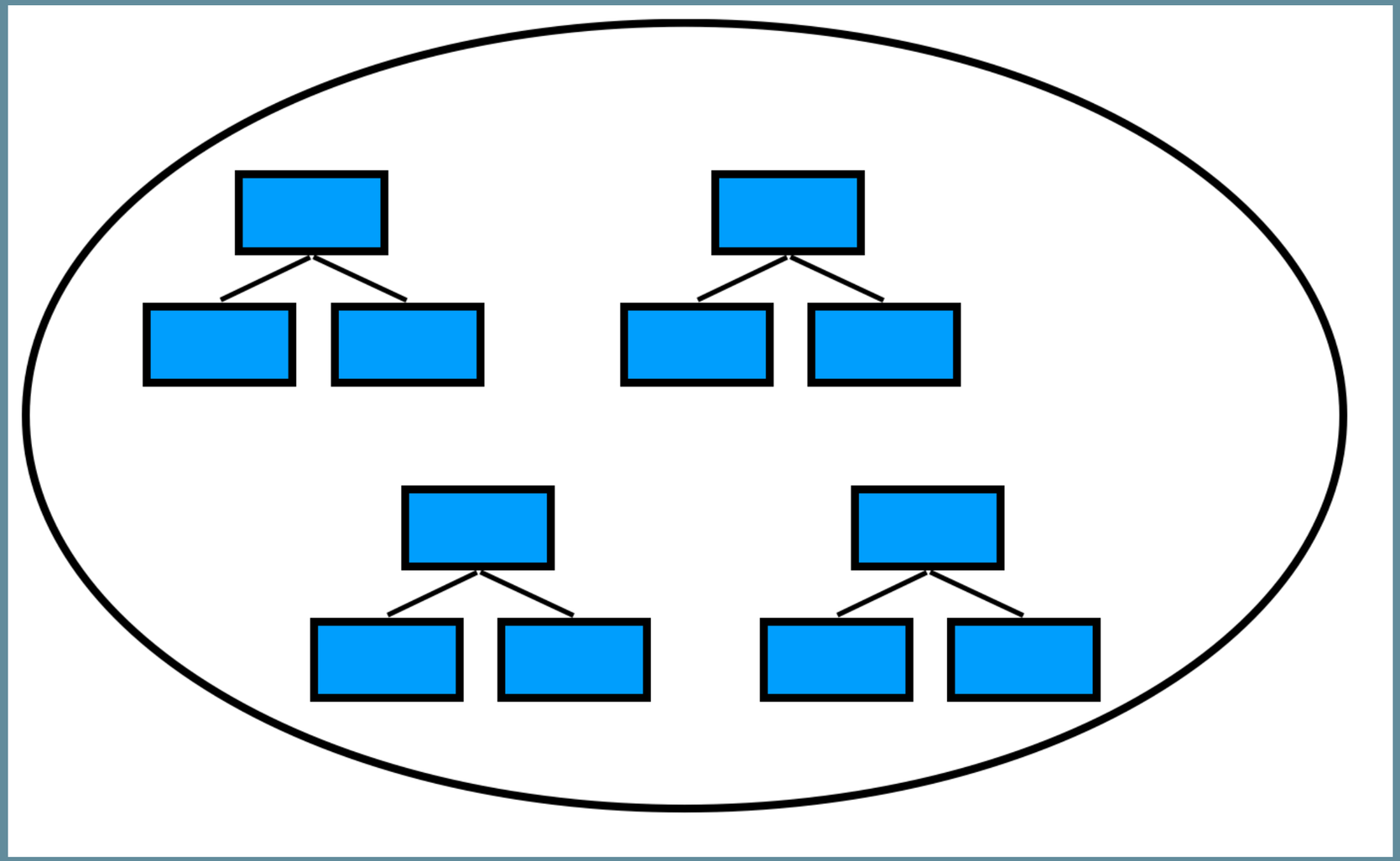

# Architecture in the 2000's

# Domain Driven Design

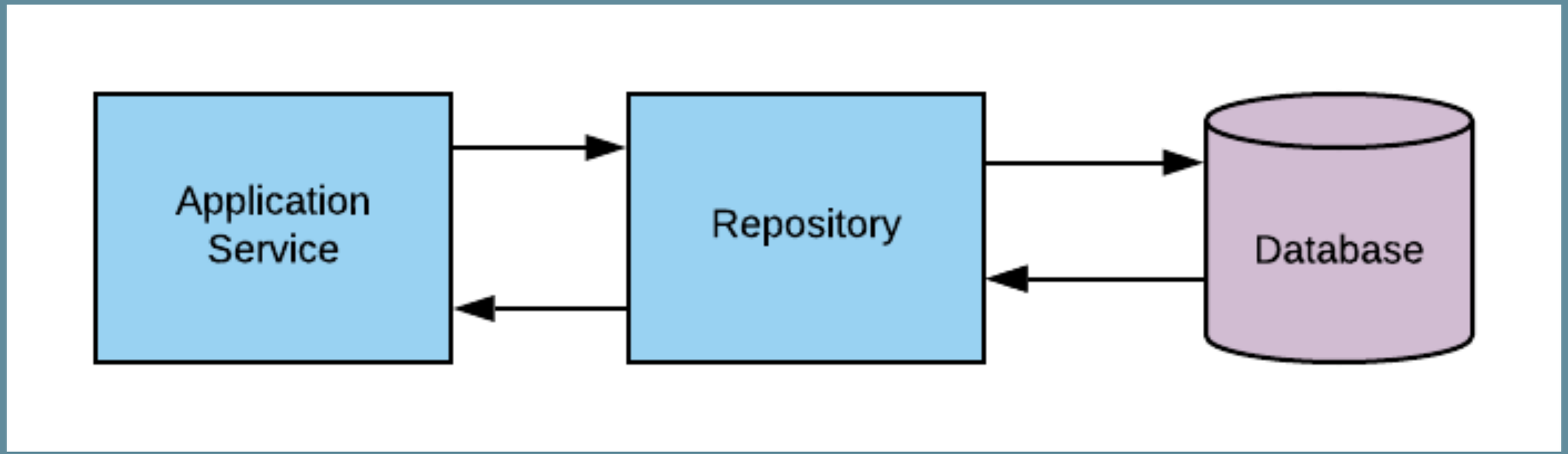


# Bounded Context

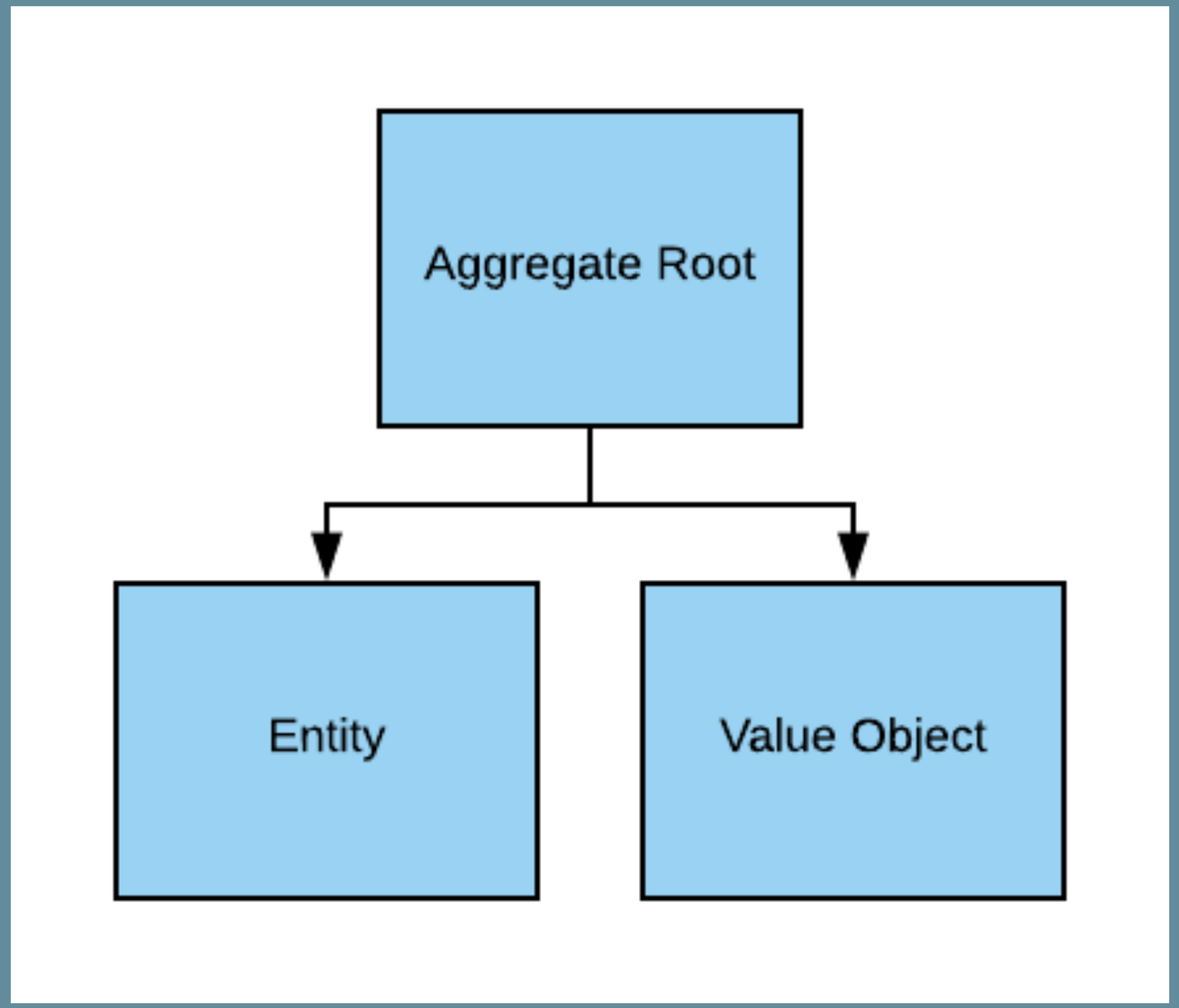
# Bounded Context



# Tactical DDD



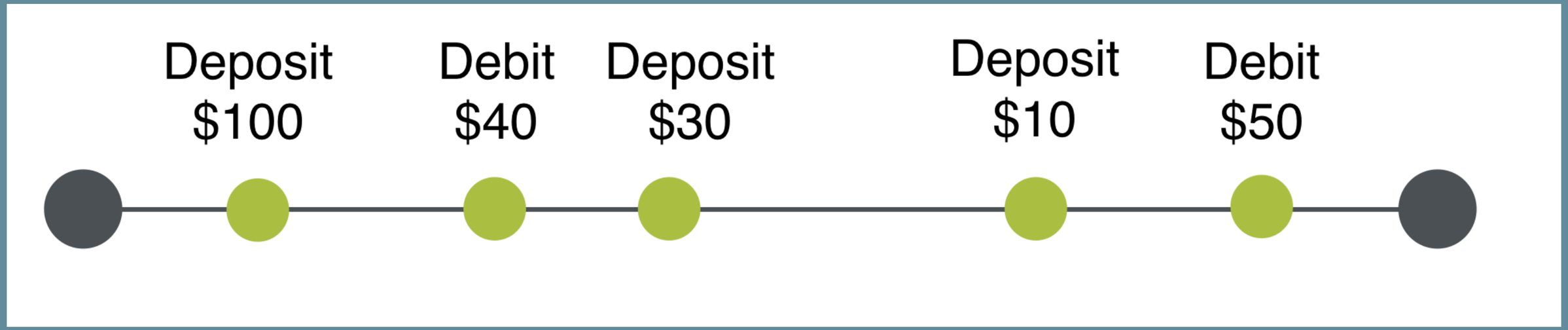
# Domain Objects



# What is CQRS?

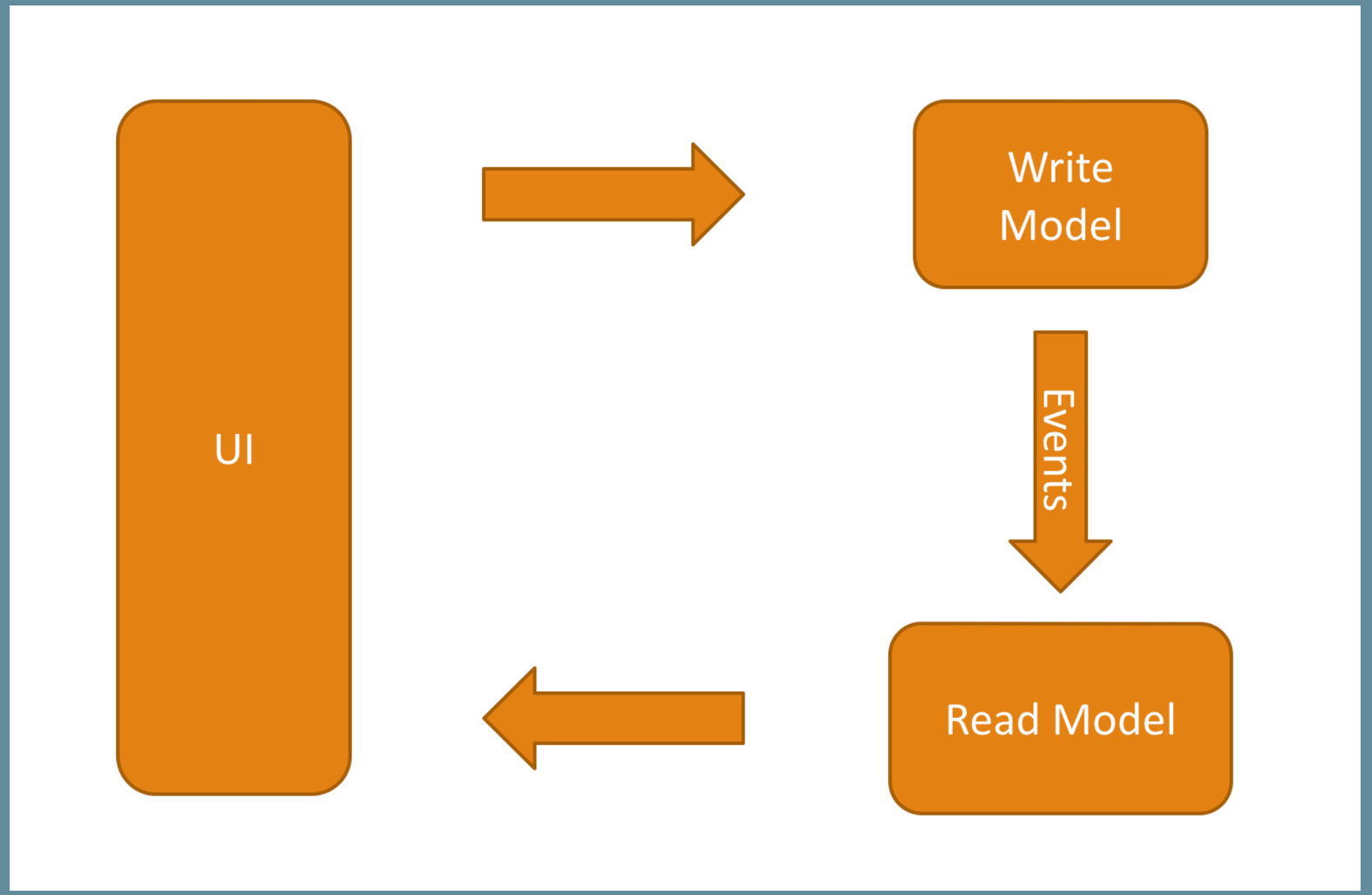
- Separate Read Write Models
- Event Oriented

# Event Sourced

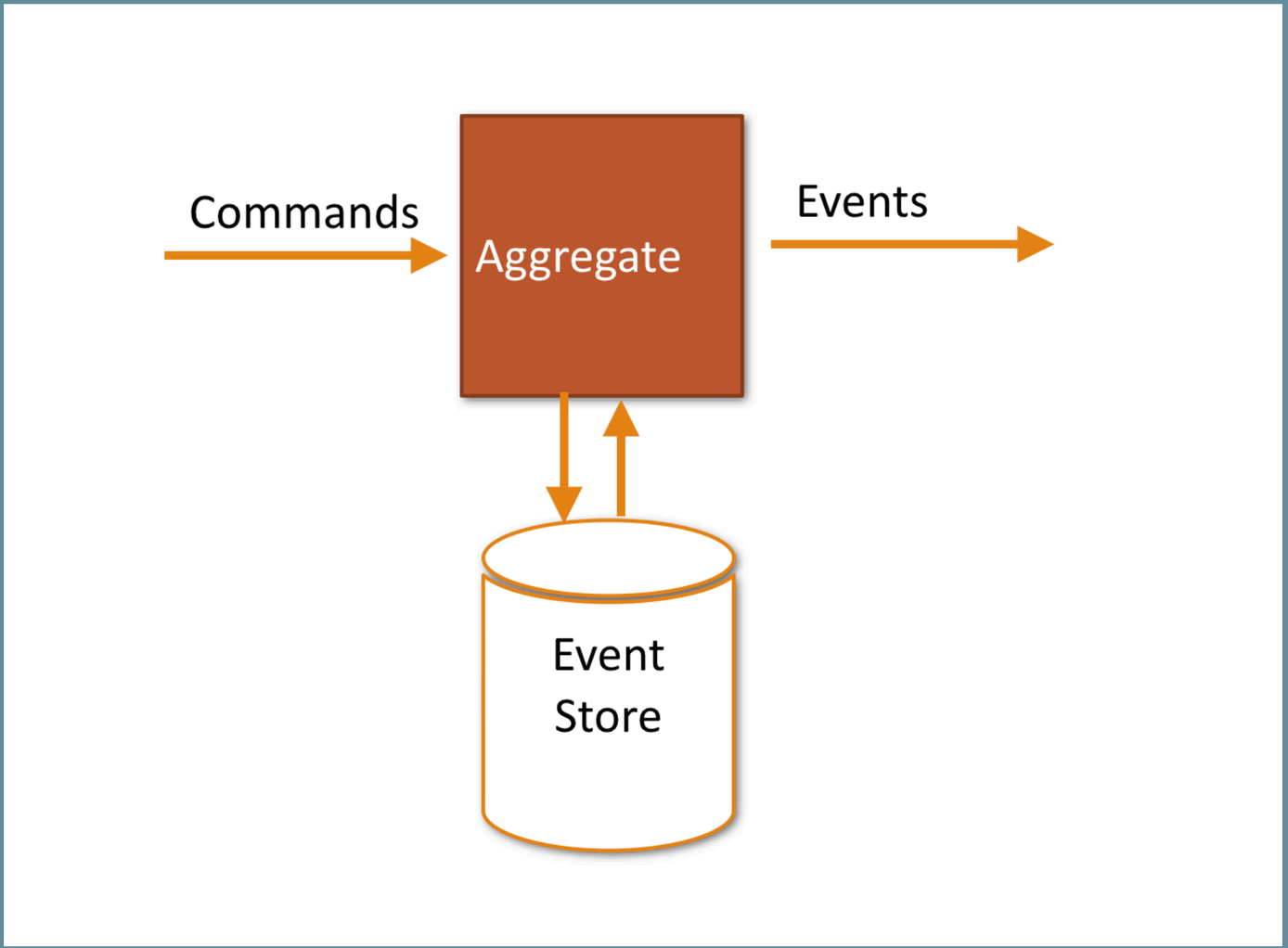


# Event Oriented DDD

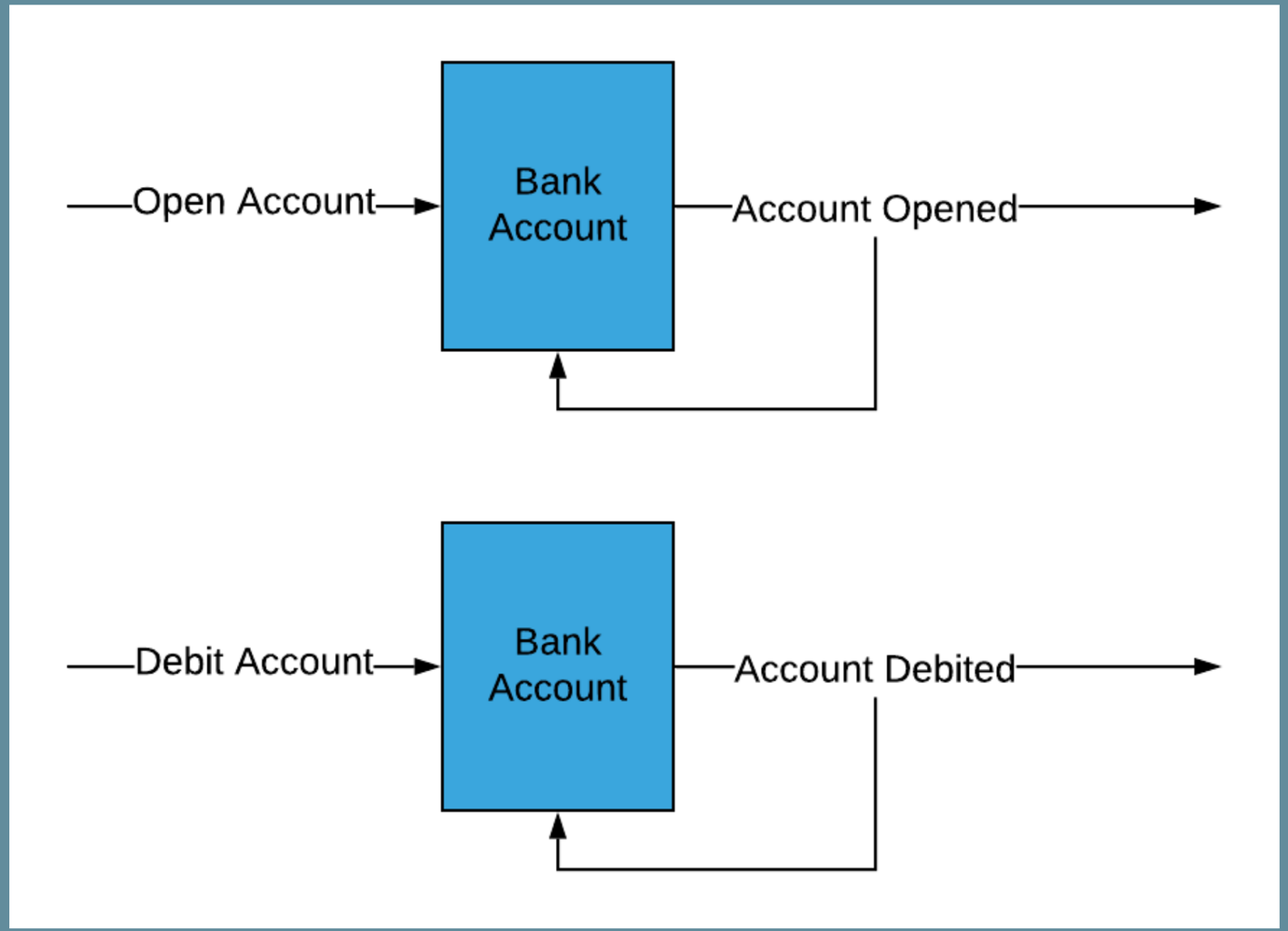
# Overall



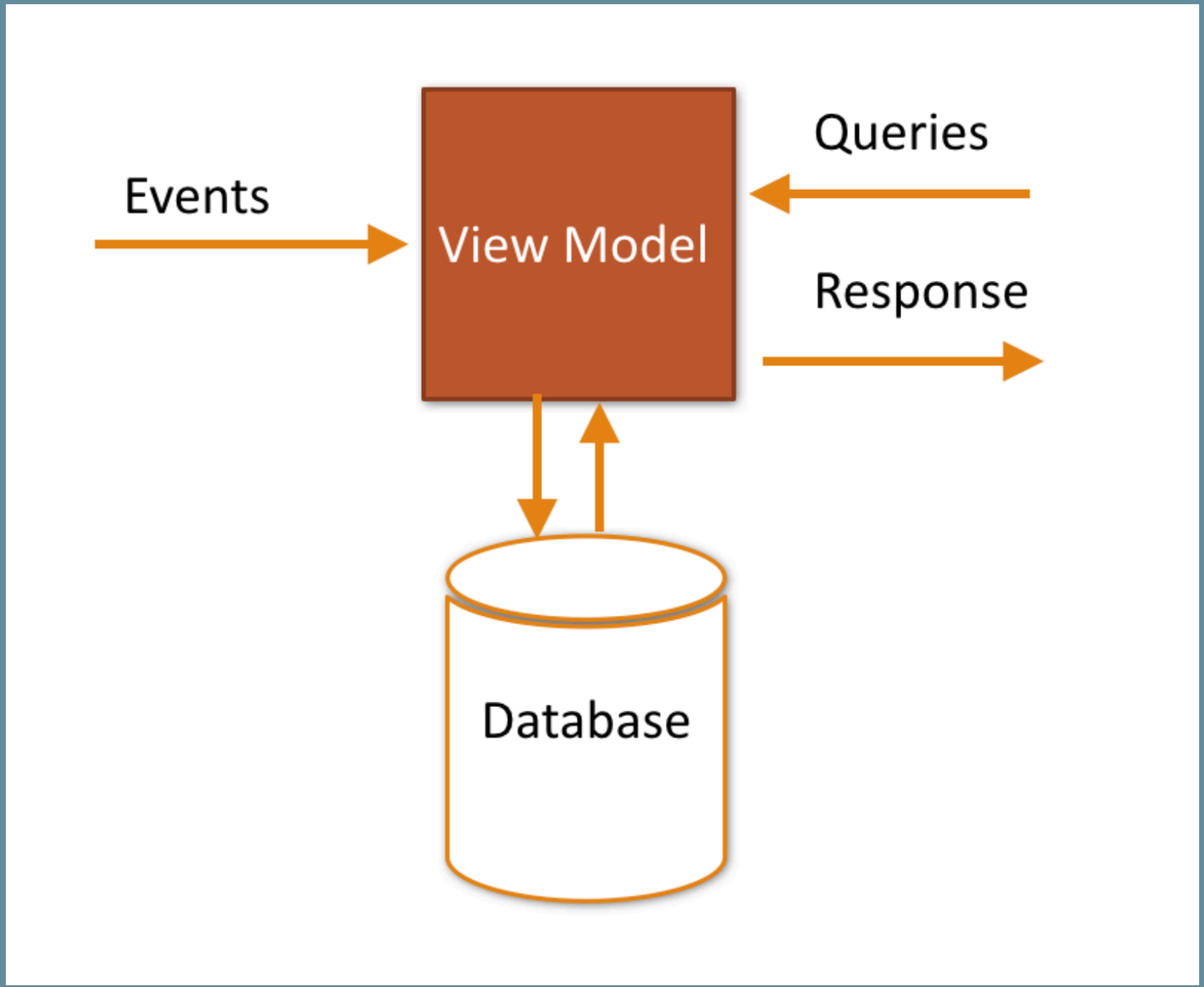
# Write Model



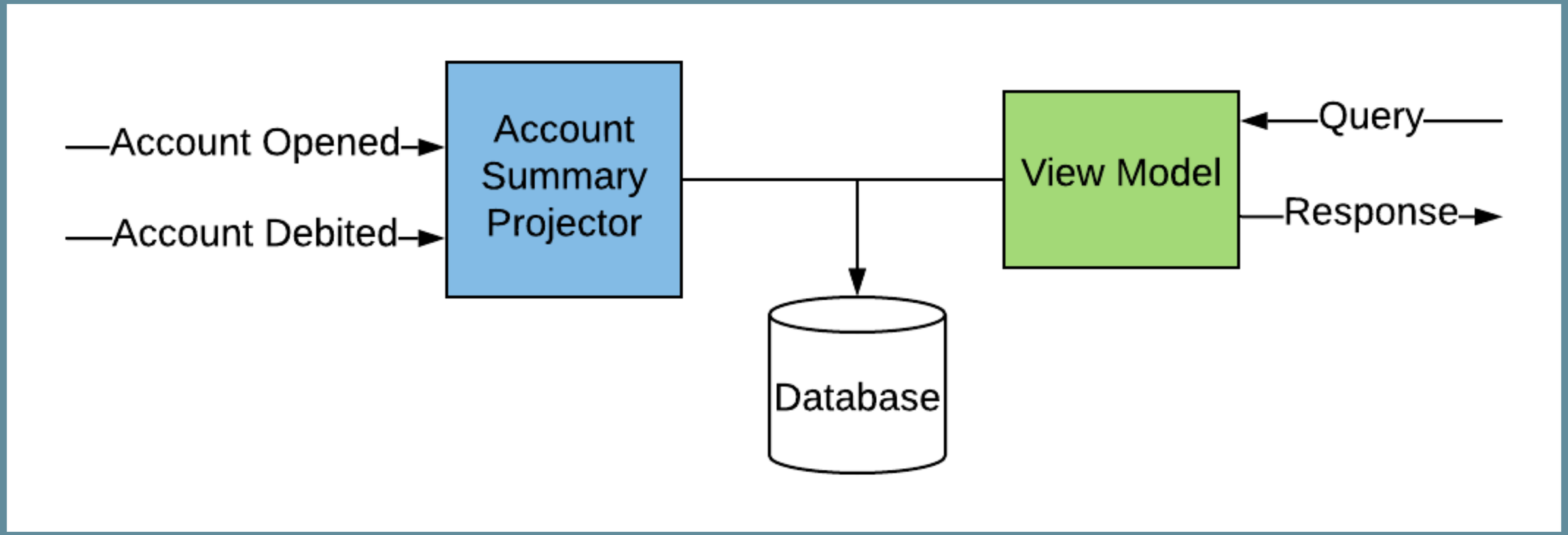
# Write Model Example



# Read Model

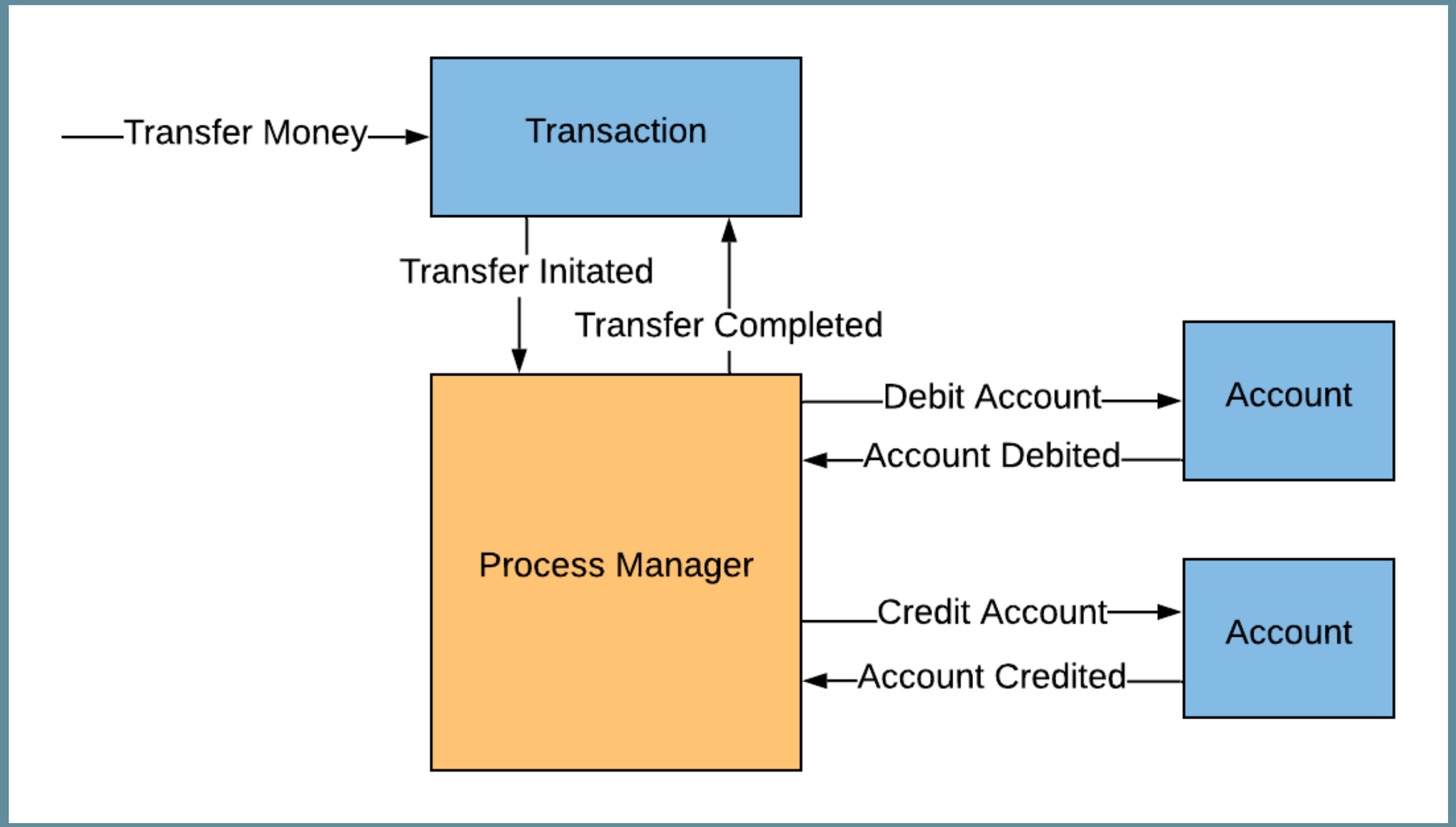


# Read Model Example



# Saga/Process Manager

# Banking Process Manager



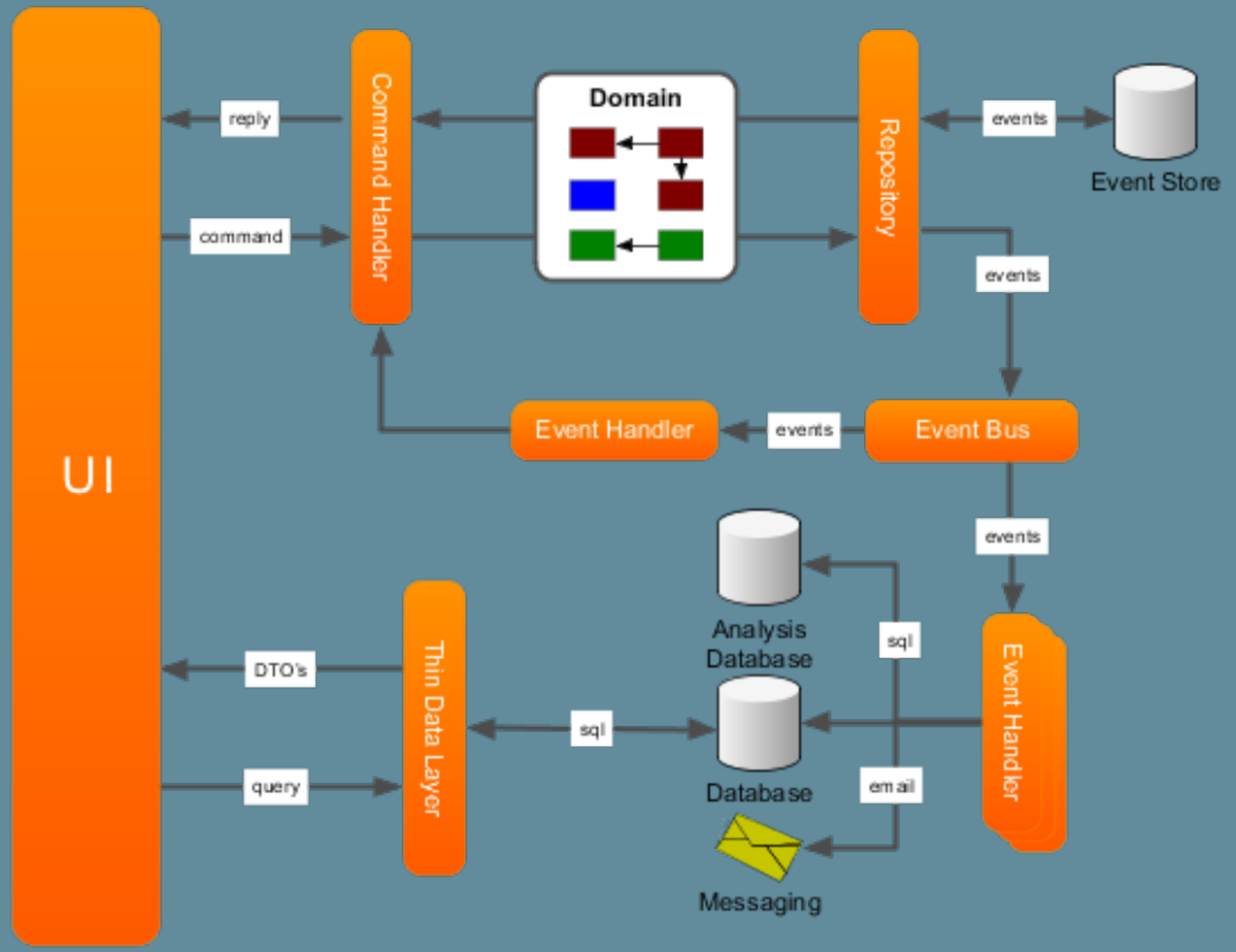
# Should you use CQRS?

# CQRS and Microservices

# Tools

- EventStore
- NServiceBus
- Axon
- Lagom/Akka

# Axon Framework



# Sample Write Model

```
@Aggregate
@NoArgsConstructor(access = AccessLevel.PRIVATE)
public class Account {
    @AggregateIdentifier
    private String id;
    private BigDecimal amount;

    @CommandHandler
    public Account(OpenAccount command) {
        AggregateLifecycle.apply(new AccountOpened(command.getId(), command.getAmount()));
    }

    @EventHandler
    public void on(AccountOpened event) {
        this.id = event.getAccountId();
        this.amount = event.getAmount();
    }
```

# Sample Write Model

```
@CommandHandler
public void handle(CreditAccount command) {
    AggregateLifecycle.apply(new AccountCredited(id, command.getAmount()));
}

@EventHandler
public void on(AccountCredited event) {
    this.amount = amount.add(event.getAmount());
}
```

# Command

```
@AllArgsConstructor
@Getter
public class OpenAccount {
    @RoutingKey
    private String accountId;
    private BigDecimal amount;
}
```

# Event

```
@AllArgsConstructor
@Getter
public class AccountOpened {
    @AggregateIdentifier
    private String accountId;
    private BigDecimal amount;
}
```

# Sample Write Model

```
@CommandHandler
public void handle(DebitAccount command) {
    if (hasAtLeast(command.getAmount())) {
        throw new IllegalArgumentException("Not enough money");
    }
    AggregateLifecycle.apply(new AccountDebited(id, command.getAmount()));
}

@EventHandler
public void on(AccountDebited event) {
    this.amount = amount.subtract(event.getAmount());
}
```

# Calling the Write Model

```
@RestController
@RequestMapping("account")
@AllArgsConstructor
public class AccountWriteController {
    private final CommandGateway commandGateway;

    @PostMapping
    public void createShowing(@RequestBody CreateAccount createAccount) {
        commandGateway.sendAndWait(new OpenAccount(newId(), createAccount.getAmount()));
    }

    @AllArgsConstructor
    @NoArgsConstructor(access = AccessLevel.PRIVATE)
    @Getter
    public static class CreateAccount {
        private BigDecimal amount;
    }
}
```

# Sample Read Model

```
@Component
@AllArgsConstructor
public class AccountSummaryProjector {
    private AccountSummaryRepository repository;
    @EventHandler
    public void on(AccountOpened event) {
        repository.save(new AccountSummaryView(event.getAccountId(), event.getAmount()));
    }
    @EventHandler
    public void on(AccountDebited event) {
        AccountSummaryView view = repository.find(event.getAccountId());
        view.setAmount(view.getAmount().subtract(event.getAmount()));
        repository.save(view);
    }
    @EventHandler
    public void on(AccountCredited event) {
        AccountSummaryView view = repository.find(event.getAccountId());
        view.setAmount(view.getAmount().add(event.getAmount()));
        repository.save(view);
    }
    @QueryHandler
    public List<AccountSummaryView> handle(AllAccountSummaryQuery query) {
        return repository.findAll();
    }
```

# Sample Read Model

```
@AllArgsConstructor
@NoArgsConstructor(access = AccessLevel.PRIVATE)
@Getter
public class AccountSummaryView {
    private String accountId;
    @Setter
    private BigDecimal amount;
}
```

# Calling the Read Model

```
@RestController
@RequestMapping("account")
@AllArgsConstructor
public class AccountReadController {
    private final QueryGateway queryGateway;

    @GetMapping("/{id}")
    public AccountSummaryView account(@PathVariable String id) {
        return queryGateway.query(new SingleAccountSummaryQuery(id), AccountSummaryView.class).get();
    }

    @GetMapping
    public List<AccountSummaryView> accounts() {
        return (List<AccountSummaryView>) queryGateway.query(
            new AllAccountSummaryQuery(),
            new MultipleInstancesResponseType(AccountSummaryView.class)).get();
    }
}
```

# Review

- Write Model (Aggregate)
  - Handles Commands
  - Sends Events
  - Receives Events
- Read Model (View Model/Projection)
  - Handles Queries
  - Sends Responses
  - Receives Events

# Resources

Udi Dahan
Greg Young
Axon

Microservices and CQRS
- Microservices.io
- Microservices Patterns by Chris Richardson

# Thank You

Seth Kraut
tech.sethkraut.com