PRACTICAL INTRODUCTION TO DDD & CQRS

GITHUB: @SKYNYRD, TWITTER, MEDIUM: @SURMELIANIL

DDD

- THE FIELD FOR WHICH A SYSTEM IS BUILT.
  - E.G. ORDER MANAGEMENT, ACCOUNT MANAGEMENT
- IT'S NOT UNUSUAL FOR AN APPLICATION TO SPAN SEVERAL DIFFERENT DOMAINS.

MODEL

A USEFUL APPROXIMATION TO THE PROBLEM AT HAND.
- GERRY SUSSMAN

DOMAIN MODEL OR DOMAIN OBJECT IS A MODEL FOR DOMAIN.

DOMAIN DRIVEN DESIGN

- YOU HAVE TO KNOW WHAT THAT SOFTWARE IS ALL ABOUT.
- YOU CANNOT CREATE A COMPLEX ECOMMERCE SOFTWARE SYSTEM UNLESS YOU HAVE A GOOD UNDERSTANDING OF WHAT ECOMMERCE IS ALL ABOUT
- ONE MUST UNDERSTAND THE **DOMAIN OF ECOMMERCE**.
- DOMAIN MENTIONED HERE CAN BE SUBDOMAIN (PROBABLY).

UBIQUITOUS LANGUAGE

- A SET OF TERMS USED BY ALL PEOPLE INVOLVED IN THE DOMAIN, DOMAIN MODEL, IMPLEMENTATION, AND BACKENDS. THE IDEA IS TO AVOID TRANSLATION

TRANSLATION BLUNTS COMMUNICATION AND MAKES KNOWLEDGE CRUNCHING ANEMIC.

– ERIC EVANS

- "OH, YOU'RE USING 'USER' IN THESE CASES WHERE I'M USING 'ACCOUNT'"
- WE LOSE A DIRECT ABILITY TO THINK CLEARLY ABOUT THE THING WE ARE BUILDING AND TO LET NEW KNOWLEDGE FLOW BACK AND FORTH BETWEEN DOMAIN AND IMPLEMENTATION.

BOUNDED CONTEXT

- A DIVISION OF A LARGER SYSTEM THAT HAS ITS OWN UBIQUITOUS LANGUAGE AND DOMAIN MODEL.
  - ORDER MANAGEMENT, CLAIM MANAGEMENT, SHIPPING MANAGEMENT, MERCHANT MANAGEMENT

DDD ❤

- **CLEAN CODE** (E.G. NAMING CONVENTION)
- **OOP PATTERNS** (E.G. COMMAND INVOKER)
- **TDD**
- **CQRS**
- **EVENT DRIVEN ARCHITECTURE**

RULE:

DOMAIN MODELS NEED TO BE

CLEAR

CUSTOMER CLAIM

ID (GUID)

NUMBER (STRING)

LFINR (INT)

PKTSTATUS (INT)

SAPNUMBER (STRING)

OMSNUMBER (INT)

CUSTOMER CLAIM

ID (IDENTIFIER)

TYPE (CLAIMTYPE)

APPROVAL (CLAIMAPPROVAL)

STATUS (CLAIMSTATUS)

REJECTION (CLAIMREJECTION)

QUANTITY (QUANTITY)

VALUE OBJECTS

WHICH VARIABLE TYPE WOULD YOU USE FOR REPRESENTING THE AGE OF A PERSON?

[ ] INTEGER

[ ] BOOLEAN

[ ] STRING

OF COURSE, AGE 😌

HASSLE-FREE SHARING

- IMMUTABLE
- DRAMATICALLY LOWERS ACCIDENTAL COMPLEXITY AND COGNITIVE LOAD REQUIRED TO AVOID INTRODUCING ANY BUG.
- MULTI THREADED ENVIRONMENT

HASSLE-FREE SHARING

```
final class Name {
    private String value;

    public Name(String value) {
        this.value = value;
    }
}
```

IMPROVED SEMANTICS

```
public struct Point
{
    private readonly int x;
    private readonly int y;

    public Point(int x, int y)
    {
        this.x = x;
        this.y = y;
    }

    public override bool Equals(object obj)
    {
        if (ReferenceEquals(null, obj)) return false;
        return obj is Quantity && Equals((Quantity)obj);
    }

    public override string ToString()
    {
        return $"x: {x}, y: {y}";
    }

    public static bool operator ==(Point first, Point second)
    {
        return first.Equals(second);
    }

    public static bool operator !=(Point first, Point second)
    {
        return !(first == second);
    }
}
```

SELF VALIDATION

```
final class Rank {
    private int value;

    public Rank(int value) {
        if (value < 1 || value > 13) {
            throw new InvalidRankValue(value); // Or isValid=False;
        }

        this.value = value;
    }
}
```

RULE:
DOMAIN MODELS ARE FIRST CLASS CITIZENS!

```
public class Claim
 {
     public string Id { get; set; }
     public DateTime CreatedAt { get; set; }
     public DateTime UpdatedAt { get; set; }
     public string LineItemId { get; set; }
     public string Status { get; set; }
     public bool Rejection { get; set; }
     public bool Approval { get; set; }
     public string Type { get; set; }
     public int Quantity { get; set; }
 }
```

S{O}LID

OPEN-CLOSED PRINCIPLE

SOFTWARE ENTITIES (CLASSES, MODULES, FUNCTIONS, ETC.) SHOULD BE OPEN FOR EXTENSION, BUT CLOSED FOR MODIFICATION.

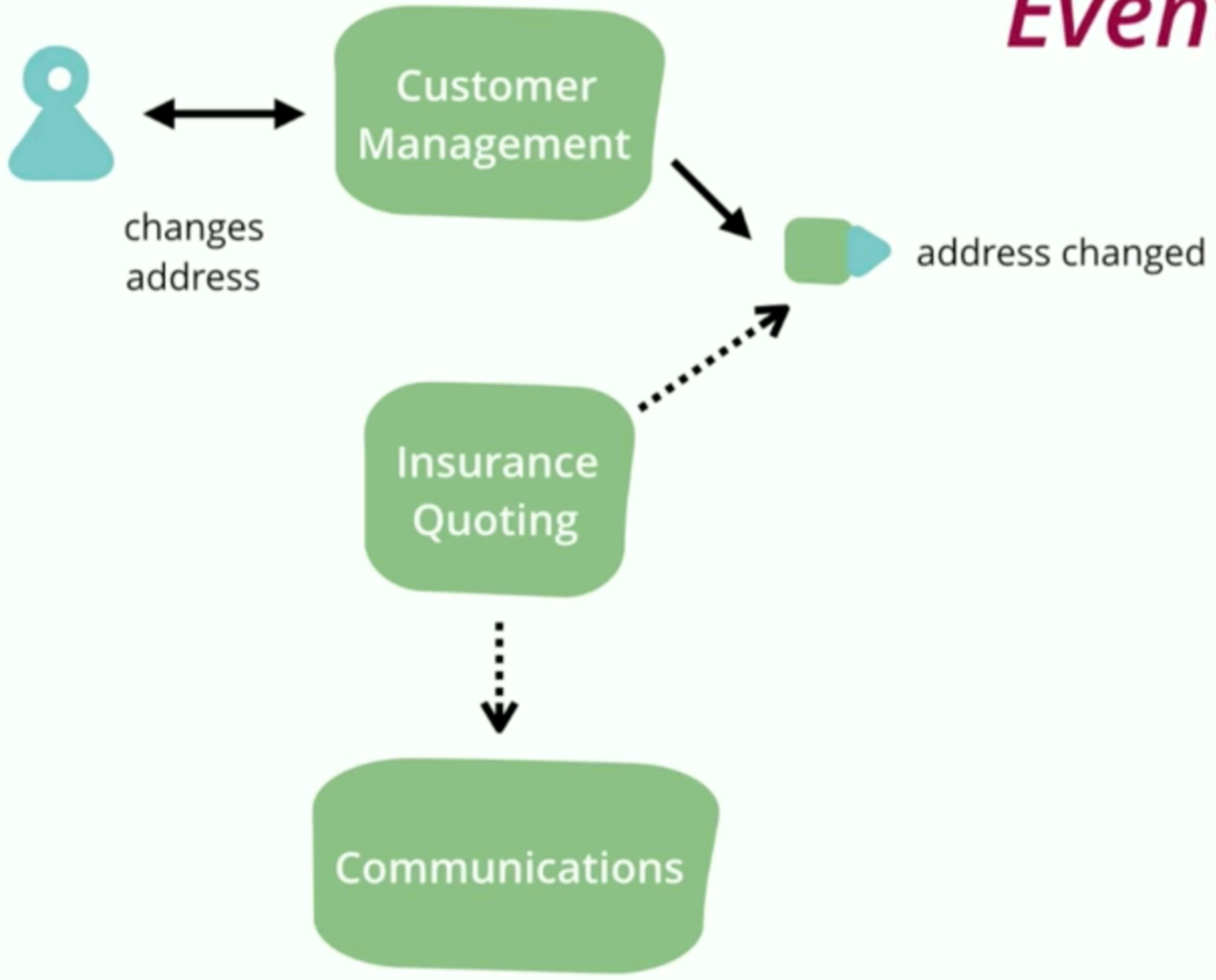
```
public class Claim : AggregateRoot
{
    public Identifier LineItemId { get; private set;}
    public ClaimStatus Status { get; private set; }
    public ClaimType Type { get; private set; }
    public ClaimRejection Rejection { get; private set; }
    public ClaimApproval Approval { get; private set; }
    public Quantity Quantity { get; private set; }

    public Claim(Identifier lineItemId, ClaimType type, Quantity quantity)
    {
        Id = new Identifier(Guid.NewGuid().ToString());
        CreatedAt = DateTime.Now;
        UpdatedAt = CreatedAt;
        LineItemId = lineItemId;
        Type = type;
        Quantity = quantity;

        AddEvent(new ClaimCreatedEvent());
    }

    // Will continue on the other page
}
```

```
public void MarkAsApproved(DateTime approvalDate, AdminComment adminComment)
{
    Approval = new ClaimApproval(approvalDate, adminComment);
    AddEvent(new ClaimIsApprovedEvent(Id, LineItemId, approvalDate));
}

public void MarkAsRejected(DateTime rejectionDate, RejectionReason reason, AdminComment adminComment)
{
    Rejection = new ClaimRejection(rejectionDate, reason, adminComment);
    AddEvent(new ClaimIsRejectedEvent(Id, LineItemId, rejectionDate));
}

public override void Load(ClaimDocument document)
{
    // SOME MAPPING HERE
}

public override ClaimDocument ToWriteModel()
{
    // SOME MAPPING HERE
    return claimDocument;
}
```

- THEY SHOULD NOT CALL ANY SERVICE.
- CLEAR DOMAIN LOGIC.
- EASY TO UNDERSTAND FOR JUNIORS/NEW DEVS.
- AGGREGATE ROOT WILL BE HELD IN DEMO.

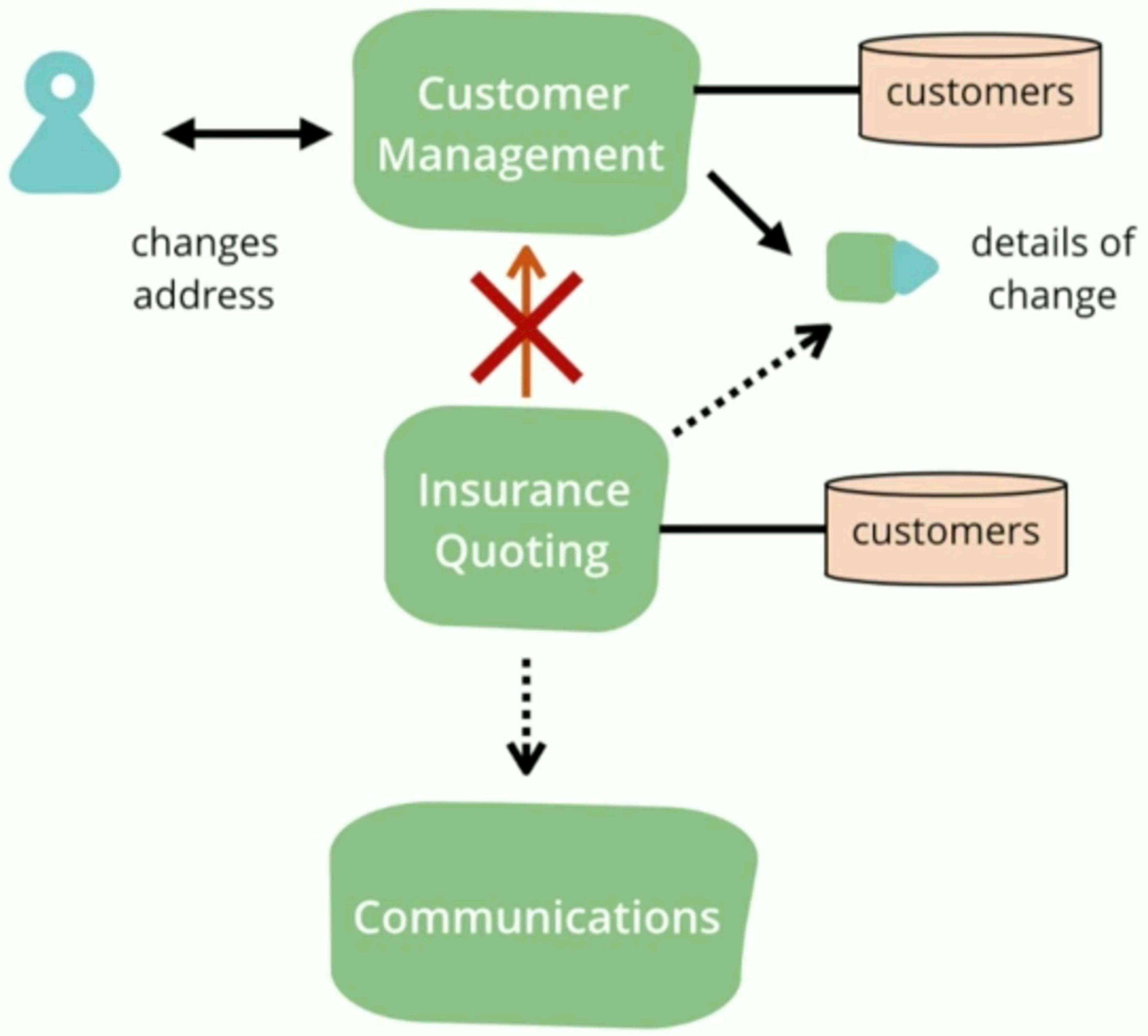
EVENT DRIVEN ARCHITECTURE

Event Notification



EVENT NOTIFICATION

- RECEIVER IS DECOUPLED FROM SENDER. 😎
- THERE IS NO STATEMENT OF OVERALL BEHAVIOUR. 👀
  - YOU CAN'T SEE THE BIG PICTURE. DEBUGGING CAN BE TRICKY.

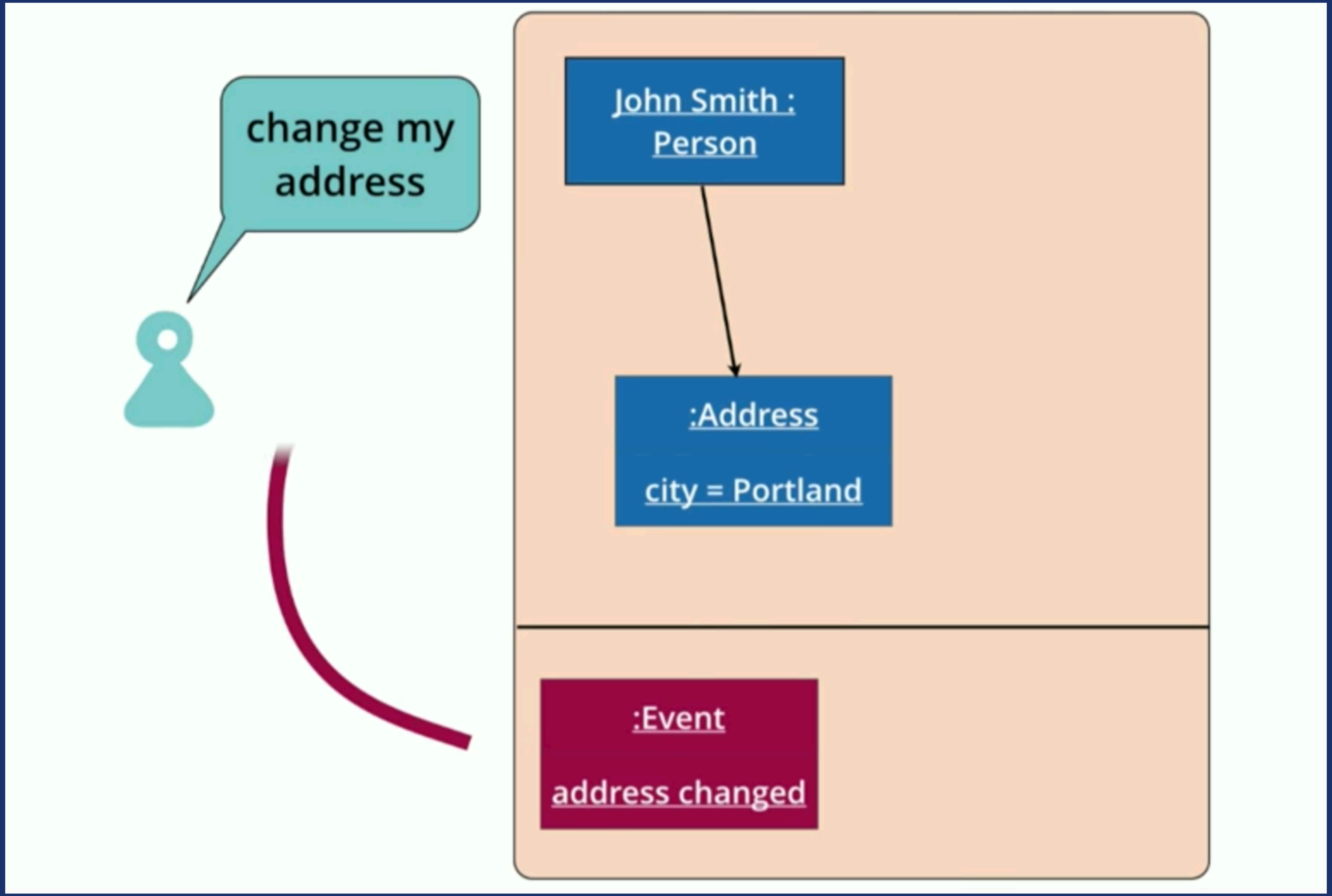


Event-carried State Transfer

- SOMETIMES CLIENT NEEDS TO ASK ADDITIONAL QUESTIONS TO THE EVENT SENDER.
  - ADDRESS CHANGED, I UNDERSTAND. BUT IS CHANGED ADDRESS THE DEFAULT ONE FOR THE CUSTOMER OR NOT?
- CLIENT KEEPS KINDA PROJECTION OF CUSTOMER DB AND LOOKS AT THAT INSTEAD OF ASKING TO THE CUSTOMER SERVICE. THE PROJECTION IS THE SAME WITH THE SOURCE DB.

- 😎
  - DECOUPLING. IF SENDER SERVICE IS DOWN, NOT A PROBLEM.
  - REDUCED LOAD ON SENDER SERVICE.
- 👀
  - REPLICATED DATA => EVENTUALLY CONSISTENT.

EVENT SOURCING



- NATURALLY EVENT SOURCED SYSTEMS: DOCTORS, LAWYERS (ADDENDUM)
- REFACTOR DOMAIN MODEL => CREATE MIGRATION SCRIPT
- PERSIST EVENTS AND FORM THE MODEL WRT EVENT SET.
- YOUR DOMAIN MODEL CONTINUOUSLY CHANGES, BUT YOUR FACTS (EVENTS) DOESN'T.

- SECURE AUDIT LOG IS IMPORTANT FOR SPECIALLY REGULATED INDUSTRIES. CURRENT STATE HAS NO MEANING FOR US.
  - E.G. CAN BALANCE EQUAL TO A COLUMN OF A SQL TABLE ? OR SUMMATION OF THE TRANSACTIONS? IT IS **FIRST LEVEL DERIVATIVE OF THE FACTS ON YOUR ACCOUNT.**

- COMPANY CAN NEED TO GET ANY VALUABLE INFORMATION THAT YOU CAN'T PRESENT.
  - I WANNA KNOW HOW MANY PEOPLE ARE REMOVING ITEMS WITHIN 5 MIN BEFORE THEY CHECKOUT.
  - YOU CAN EXTRACT A REPORT BY PROJECTIONS OF AN EVENT STREAM ANY TIME

ROLLING SNAPSHOT

- REPLAYING TONS OF EVENTS CAN NOT BE EASY.
- TAKE A SNAPSHOT FOR THE STATE IN THE EVENT STREAM.
- LOOK AT THE SNAPSHOT IF IT IS WHAT YOU WANT
- PLAY FROM THERE.
- LIKE A BREAKPOINT.
- VERY HARD TO IMPLEMENT ON FAST CHANGING SYSTEMS.

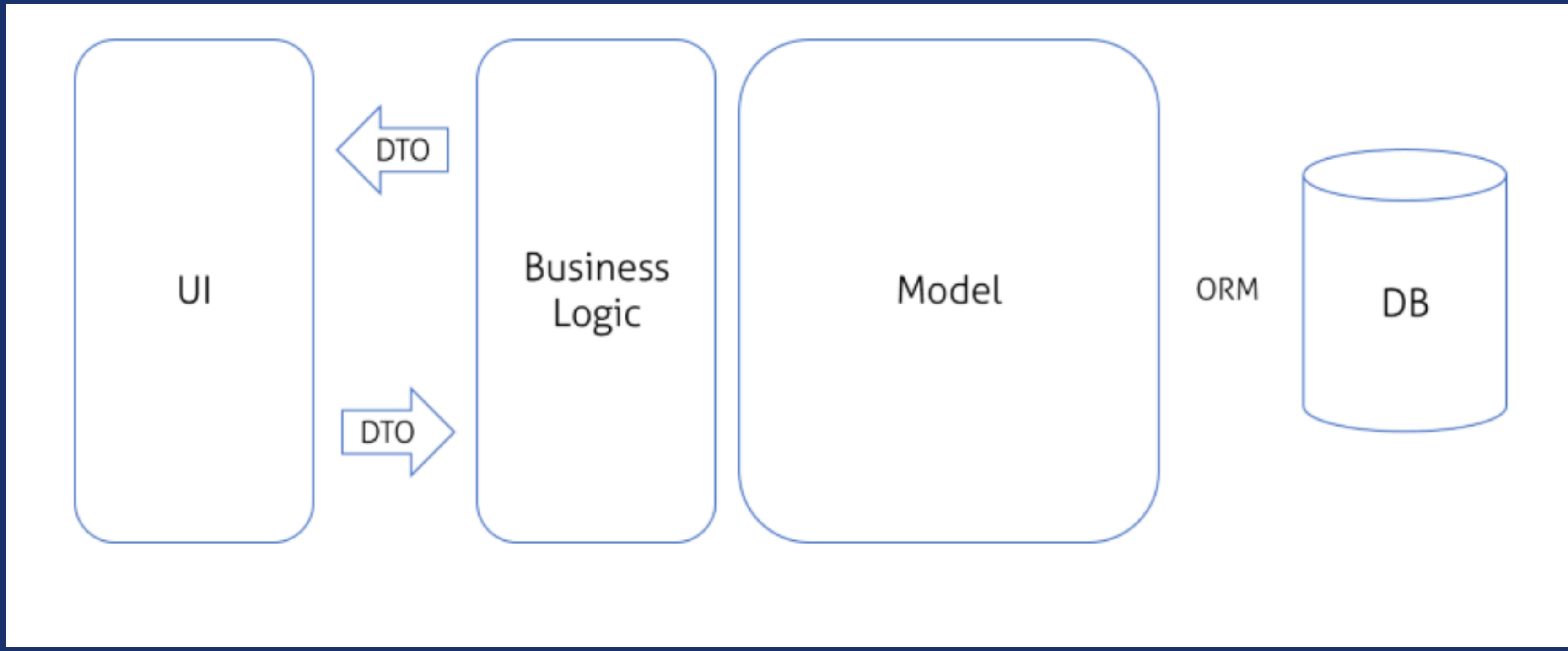
EVENT SOURCING

😎	👀
AUDIT	UNFAMILIAR
DEBUGGING	EXTERNAL SYSTEMS
HISTORIC STATE	EVENTUAL CONSISTENCY
MEMORY IMAGE	DIFFICULT LEARNING CURVE

CQRS

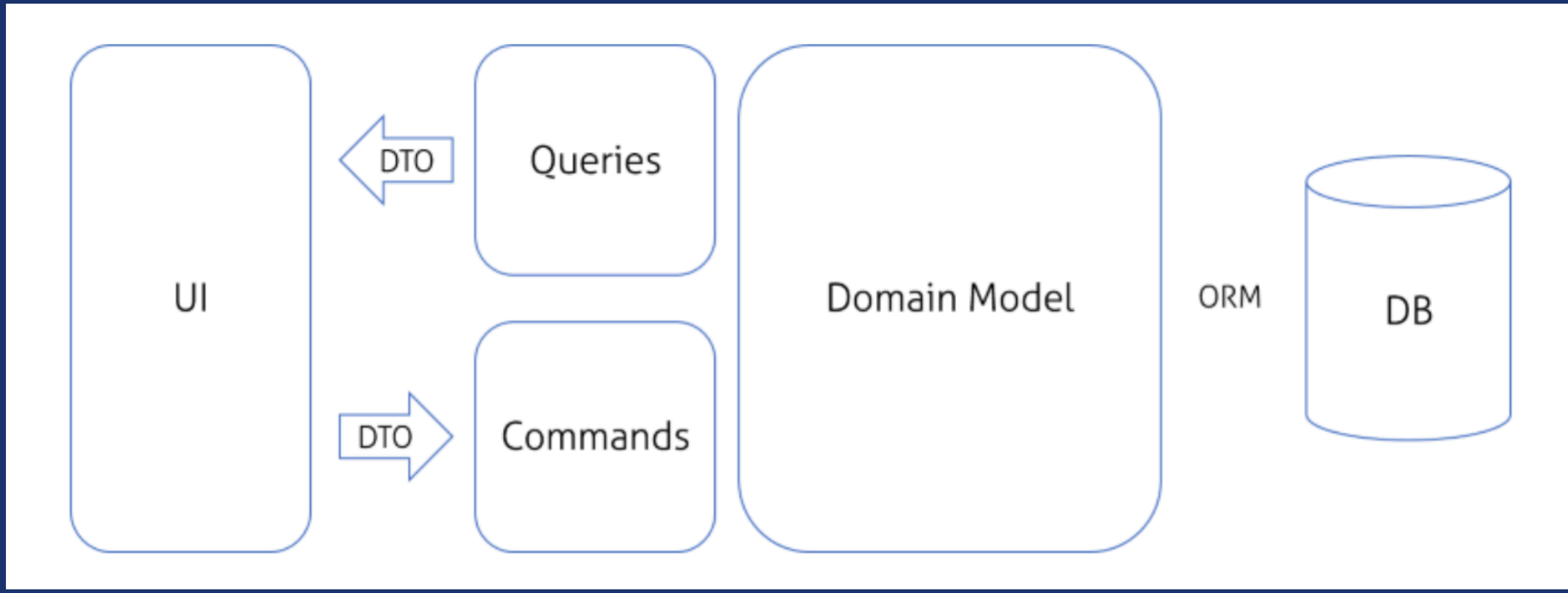
N LAYERED ARCHITECTURE

- EASY TO IMPLEMENT 😎
- BEST FOR SMALL APIS/NOT COMPLEX DOMAIN 😎
- YOU CAN ASK ANYONE 😎
- VERY COMPLEX SERVICE CLASSES. 👀
- TEND TO HAVE DEEPLY COUPLED COMPONENTS 👀
- EASY TO CREATE MESS (HELPERS?) 👀



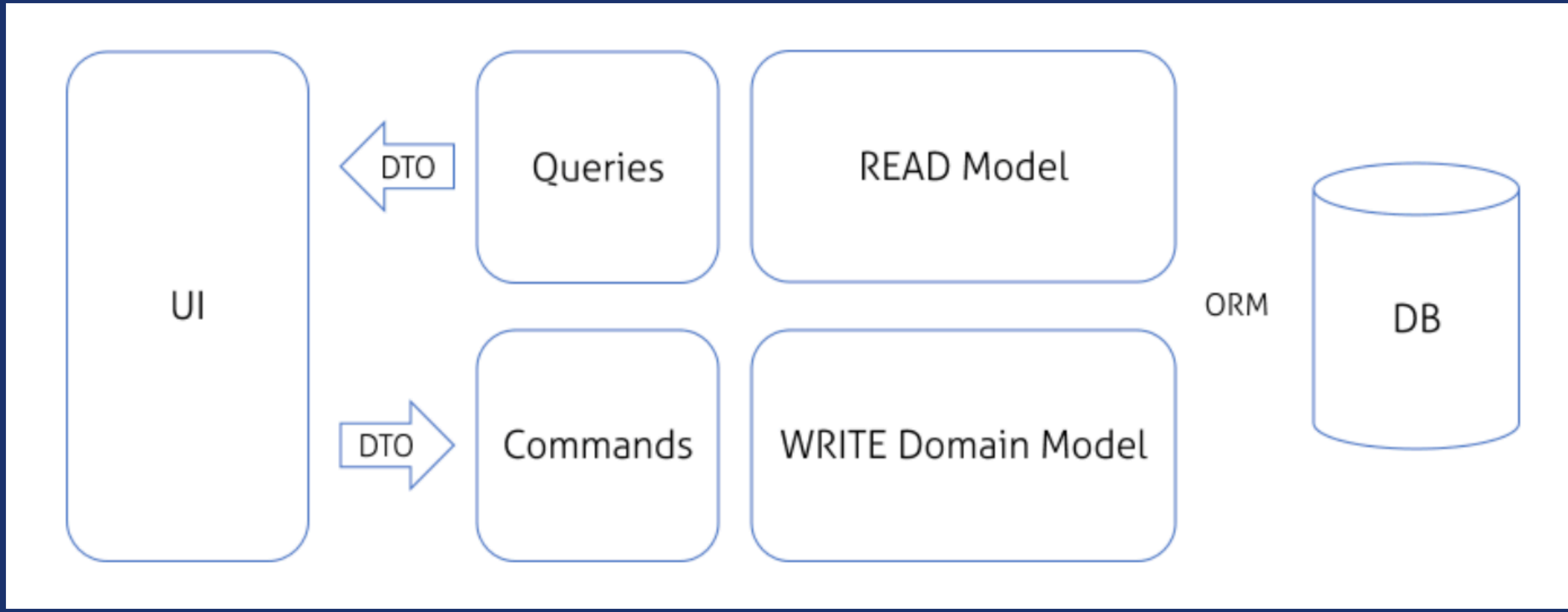
COMMAND INVOKER PATTERN

- ENHANCEMENT ON DECOUPLING
- COMMAND => ALLOWED TO MUTATE STATE. VOID RETURN TYPE. NOT IDEMPOTENT
- QUERY => NOT ALLOWED TO MUTATE STATE. NON VOID RETURN TYPE. IDEMPOTENT
- MORE EASY TO UNDERSTAND
- DESTRUCTION OF HUGE SERVICE CLASSES
- MY DEFAULT CHOICE FOR SMALL APPS.

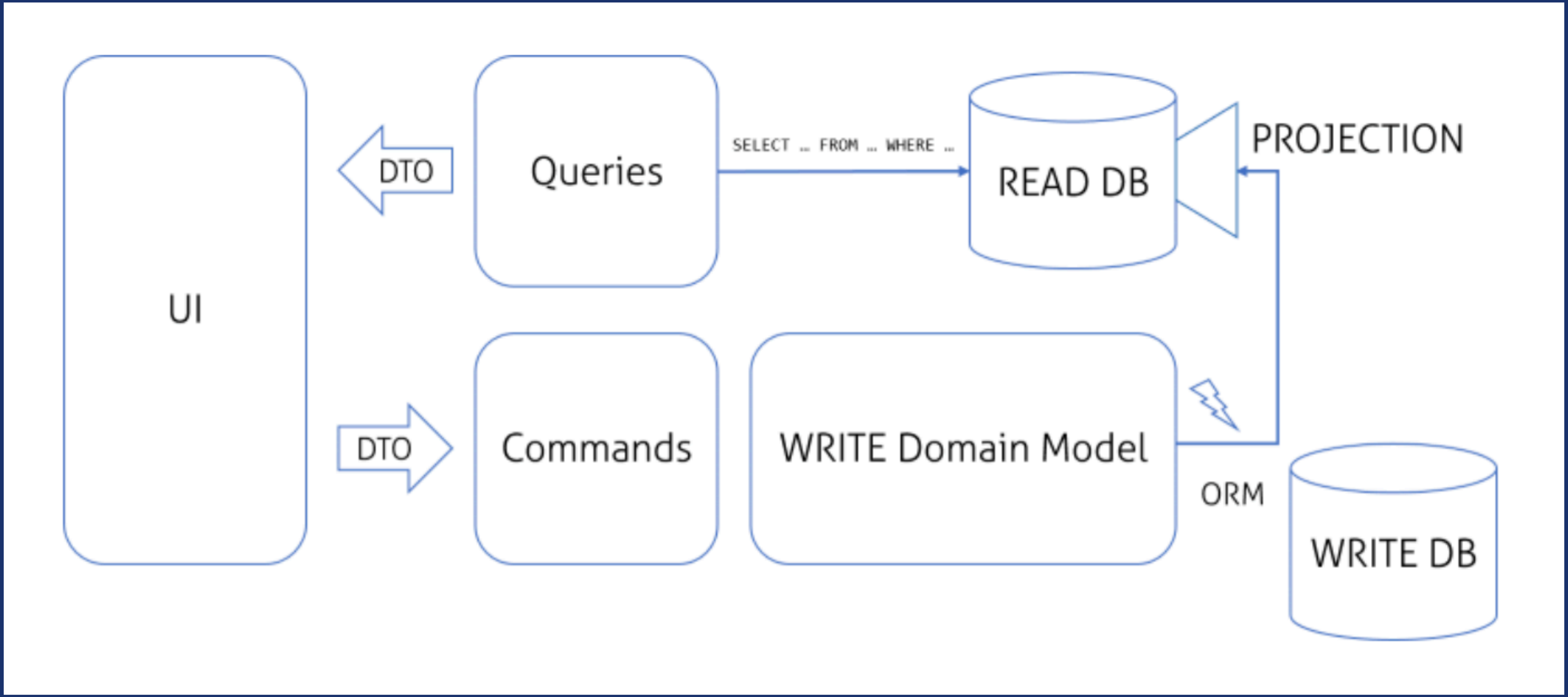


CQS (COMMAND QUERY SEPERATION)

- CLEARLY DECOUPLED STRUCTURE FOR READ AND WRITE.
- QUICK PROJECTIONS WITH LIGHTWEIGHT READ MODELS
- DEDICATED MODELS FOR A PURPOSE



CQRS


UI
DTO
Queries
SELECT … FROM … WHERE …
READ DB
PROJECTION
DTO
Commands
WRITE Domain Model
ORM
WRITE DB

CQRS (Command Query Responsibility Segregation)

- Generally, you need to scale up/out the query part of the system.
- 99% of your work is reading and you optimize for the write performance. Cool.
- Scaling query is much more easier than scaling command, because most queries can operate with relaxed consistency.

DEMO

References

```
* Eric Evans Talks and Articles
* Martin Fowler Articles
* https://herbertograca.com/2017/11/16/explicit-architecture-01-ddd-hexagonal-onion-clean-cqrs-how-i-put-it-all-together/
* https://speakerdeck.com/bobbycalderwood/commander-better-distributed-applications-through-cqrs-event-sourcing-and-immutable-logs?slide=2
* https://stackoverflow.com/questions/9495985/cqrs-event-sourcing-validate-username-uniqueness?noredirect=1&lq=1
* https://hackernoon.com/value-objects-like-a-pro-f1bfc1548c72
* https://www.future-processing.pl/blog/cqrs-simple-architecture/
* https://www.yegor256.com/2014/09/16/getters-and-setters-are-evil.html
```