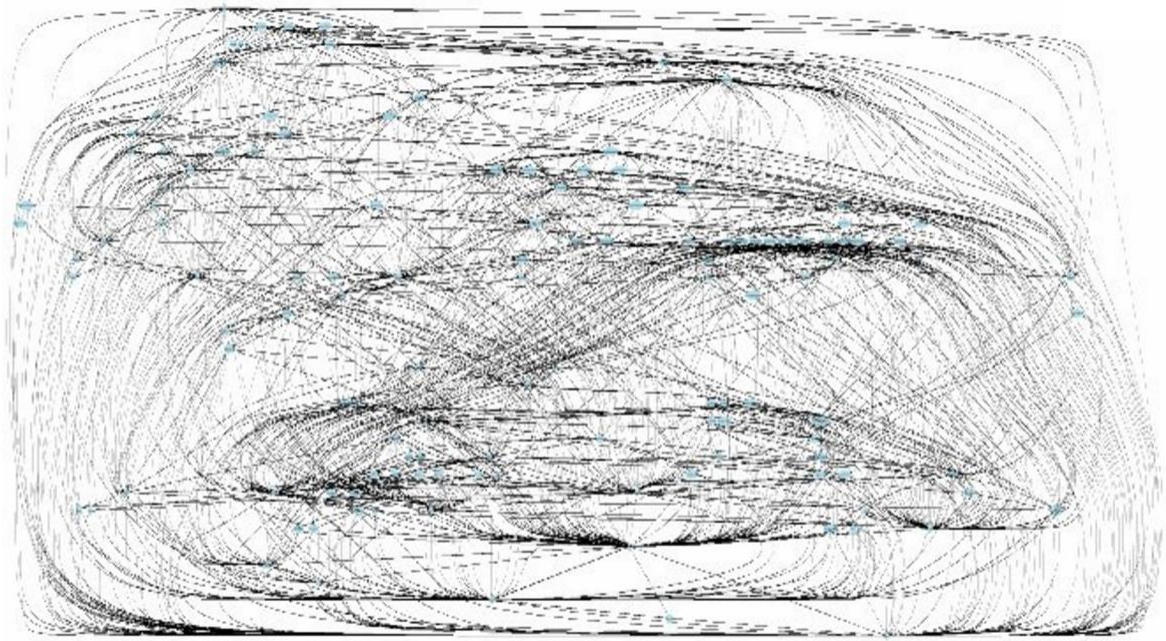
CODE COMPLEXITY

Gaetano Mondelli
mondelli.gaetano@gmail.com



Courtesy The Daily WTF

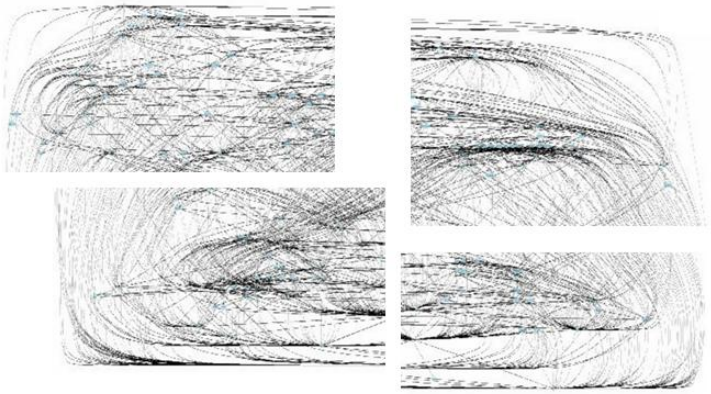
WIRED

A 2002 study by America's National Institute of Standards (NIST), a government research body, found that **software errors cost the American economy $59.5 billion annually**. Worldwide, it would be safe to multiply this figure by a factor of two. So who is to blame for such systematic incompetence?

THE ECONOMIST, Nov 25th 2004

BREAK DOWN THE COMPLEXITY

Splitting the problem into
SMALLER PARTS



Reduce the number of
EDGES

APPROACH THE COMPLEXITY - CQS

"Asking a question should not change the answer"

Bertrand Meyer, 1980

COMMAND QUERY SEPARATION

Separation of
functions that **WRITE**
and functions that **READ**

Functions that write are called COMMAND and must not return a value

Functions that read are called QUERY and must not change the state of the system (no side-effects)

CQS – BREAK DOWN COMPLEXITY

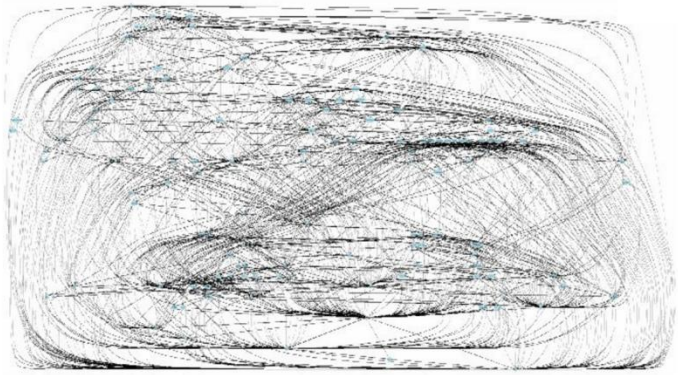
```
int globalValue = 0;

int rq(int x)
{
  globalValue++;
  return x + globalValue
}

int rt(int x)
{
  return x + 1;
}
```

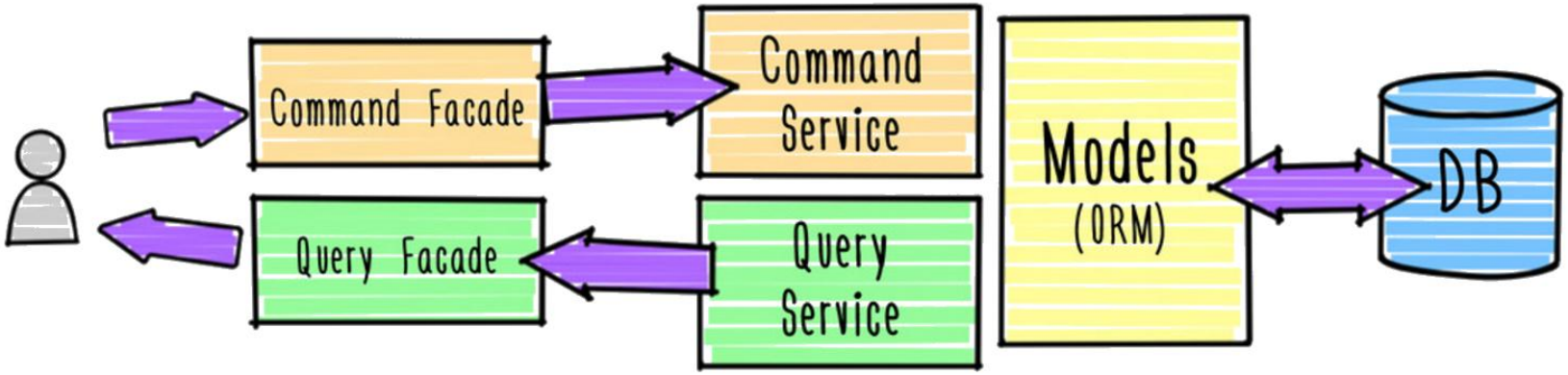
Referential transparency

$$rt(x) = rt(y),\ x == y$$



Courtesy The Daily WTF

CQRS – BREAK DOWN COMPLEXITY ON A DIFFERENT LEVEL

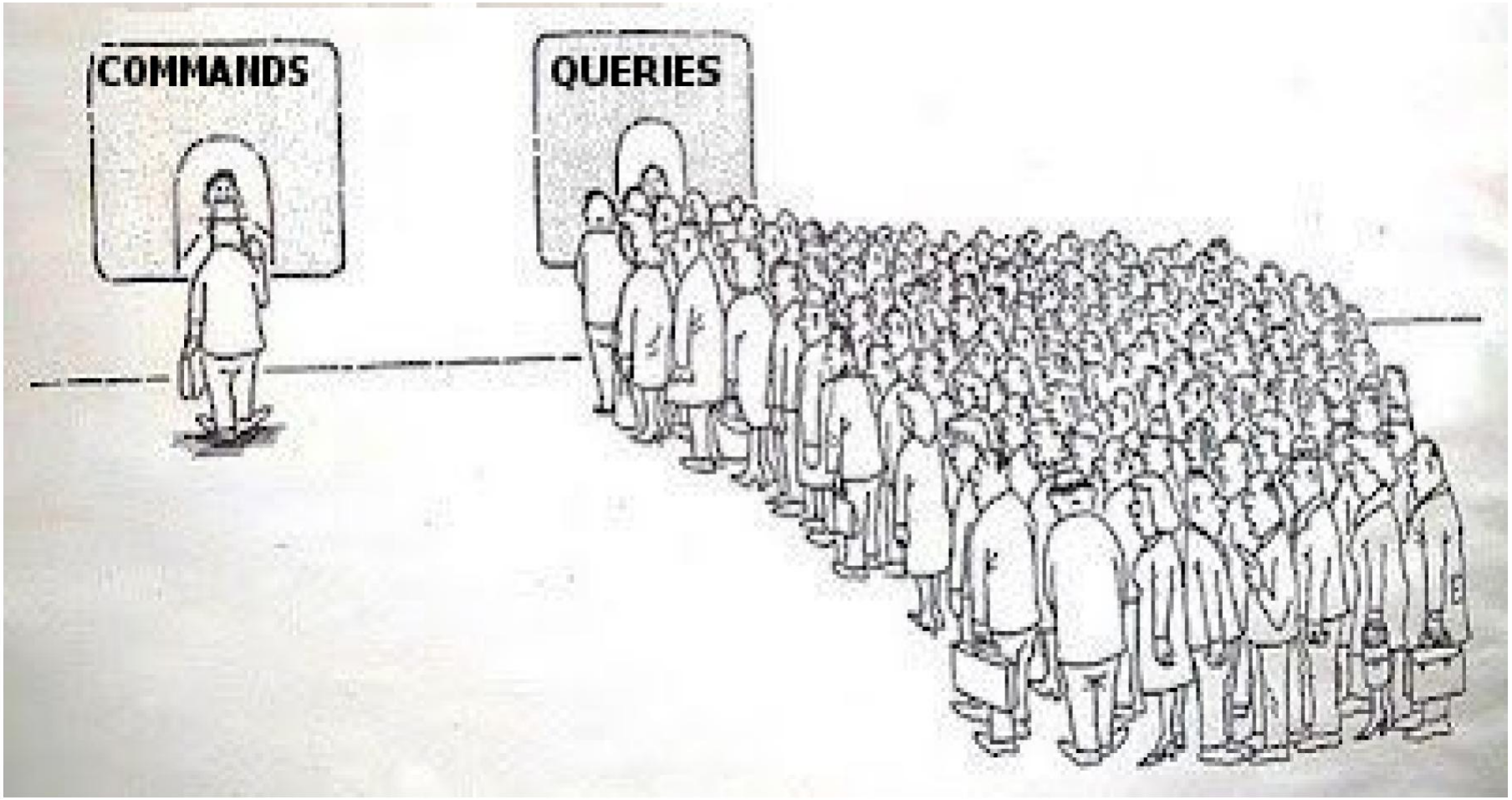


CQRS – COMMAND QUERY RESPONSIBILITY SEGREGRATION

“ CQRS is simply the creation of **two objects** where previously there was only one ”

Greg Young

CQRS



CQRS – BREAK DOWN COMPLEXITY ON A DIFFERENT LEVEL

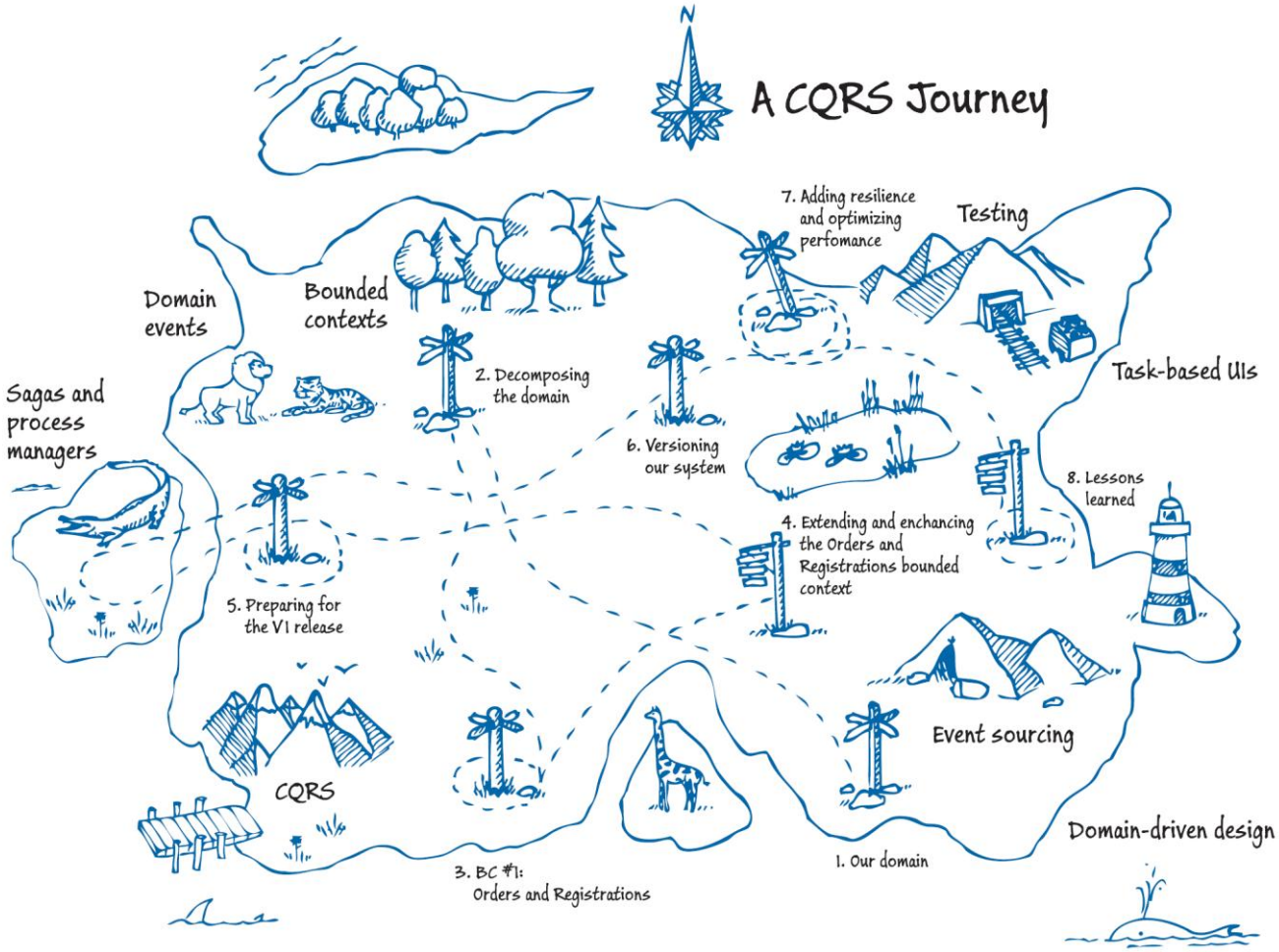


FIGURE 1
A CQRS journey

CQRS – BREAK DOWN COMPLEXITY ON A DIFFERENT LEVEL

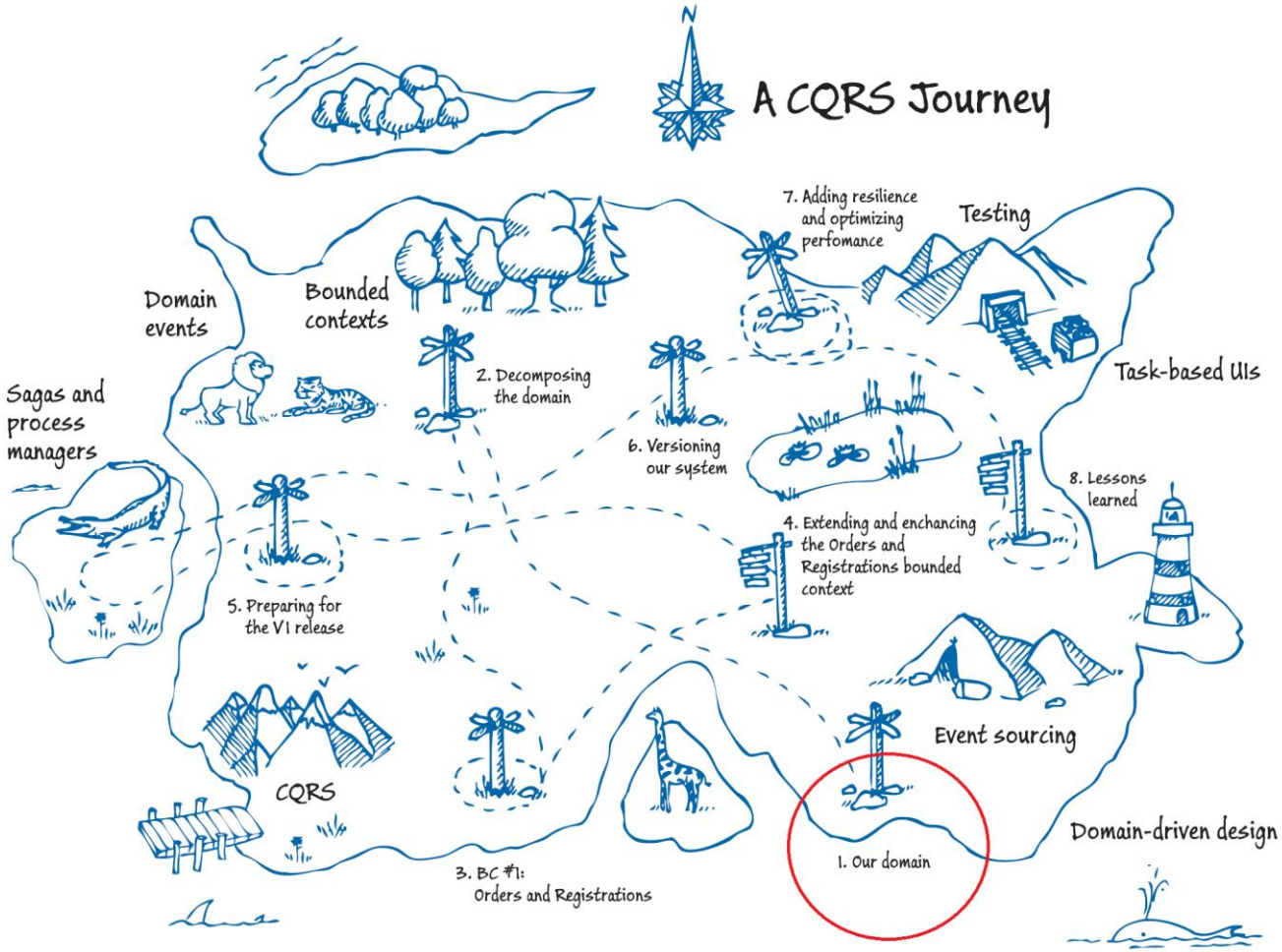


FIGURE 1
A CQRS journey

CQRS – BREAK DOWN COMPLEXITY ON A DIFFERENT LEVEL

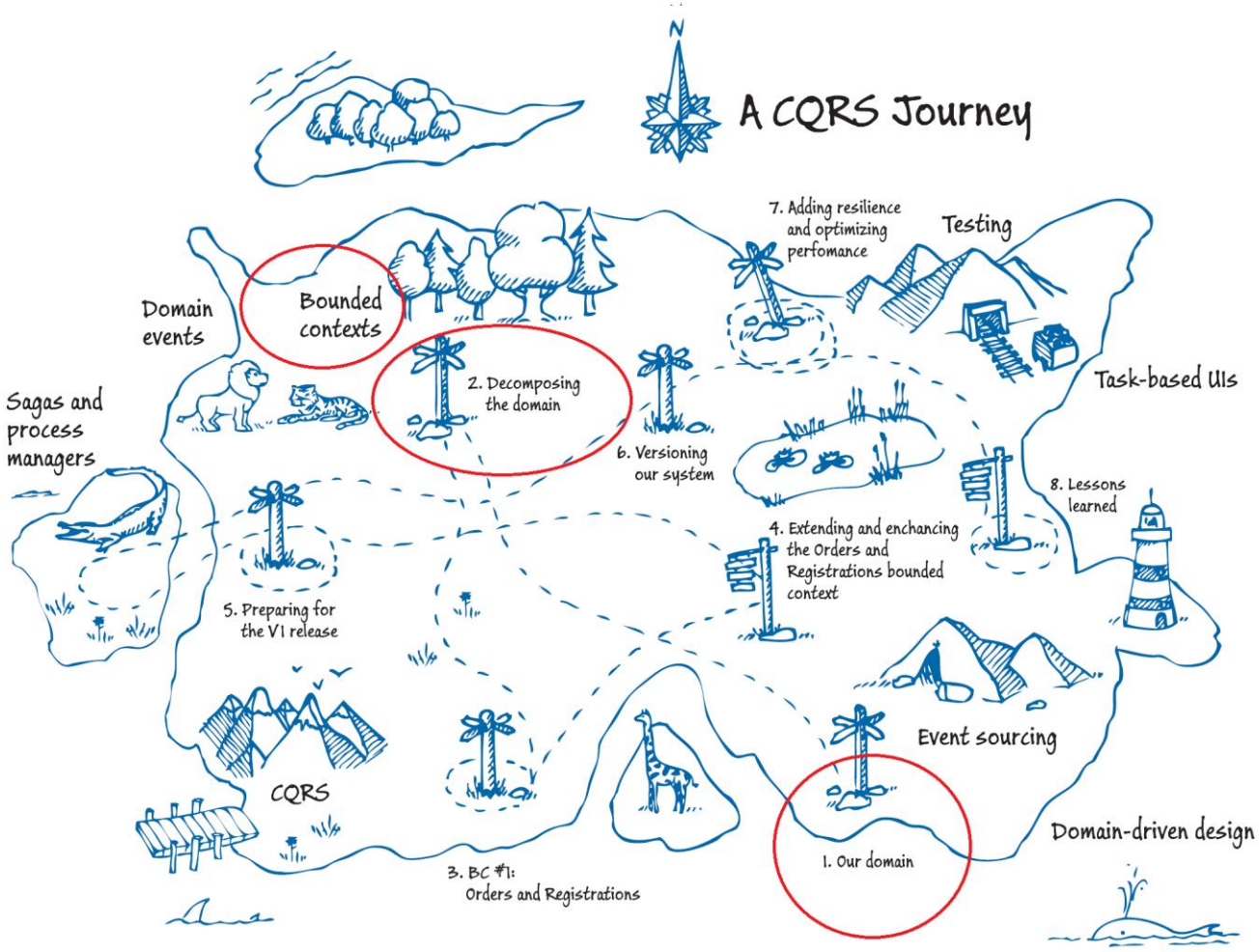


FIGURE 1
A CQRS journey

CQRS – BREAK DOWN COMPLEXITY ON A DIFFERENT LEVEL

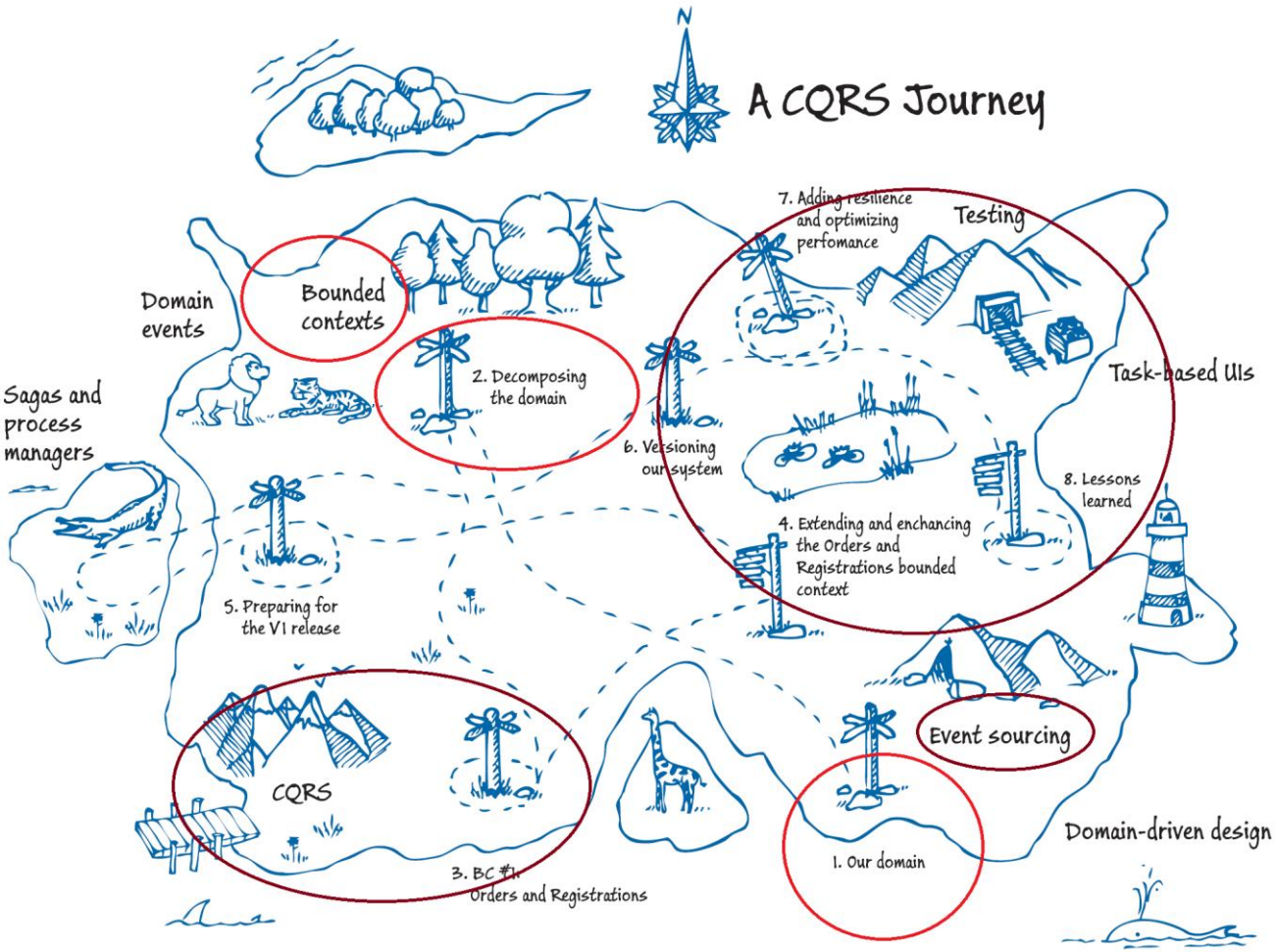


FIGURE 1
A CQRS journey

CQRS – BREAK DOWN COMPLEXITY ON A DIFFERENT LEVEL

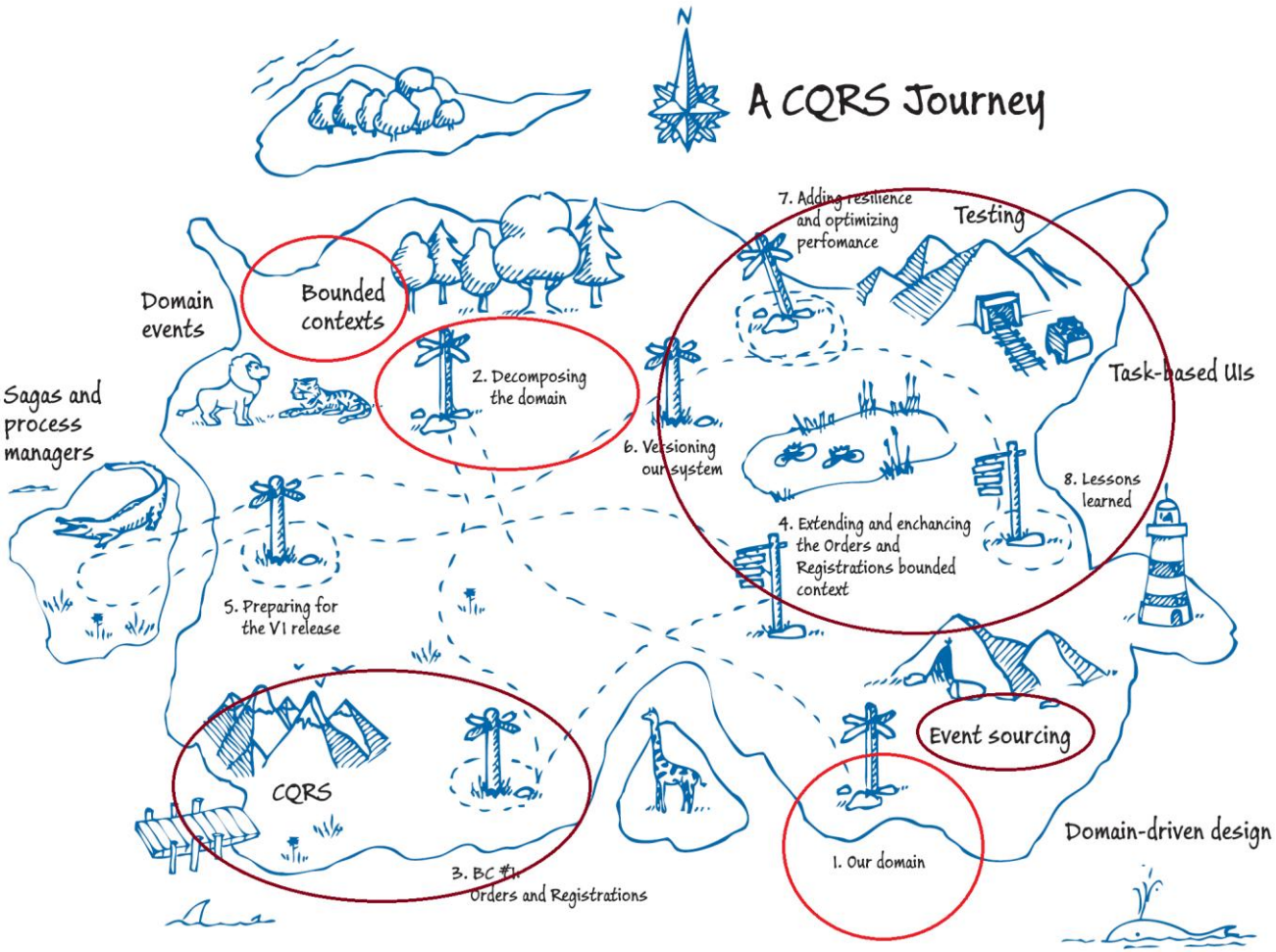
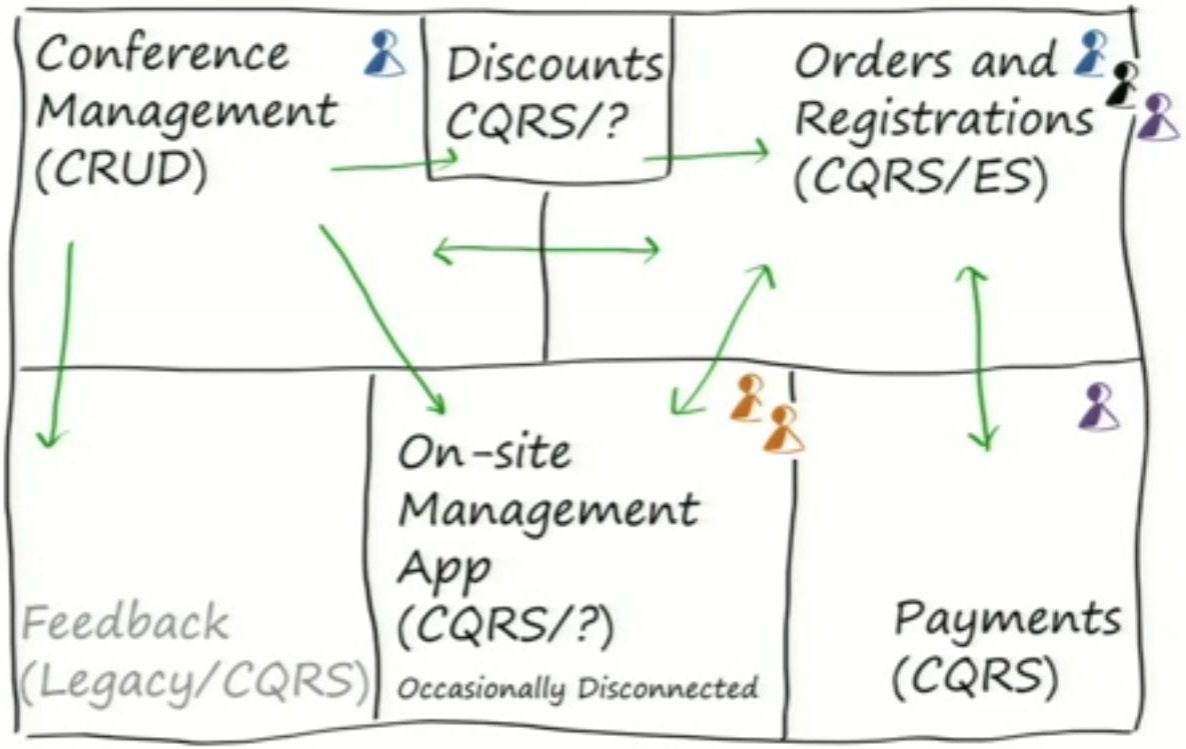
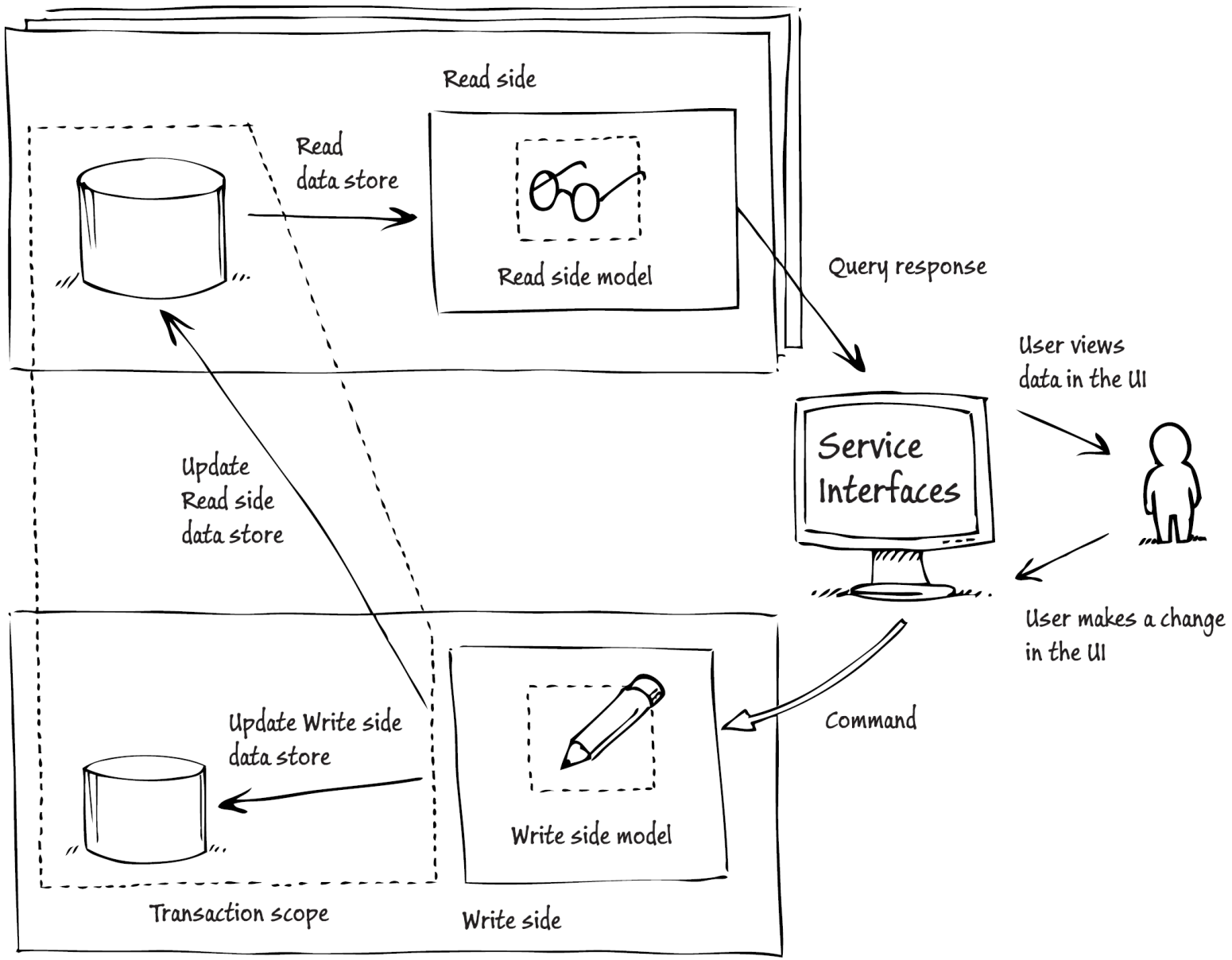


FIGURE 1
A CQRS journey

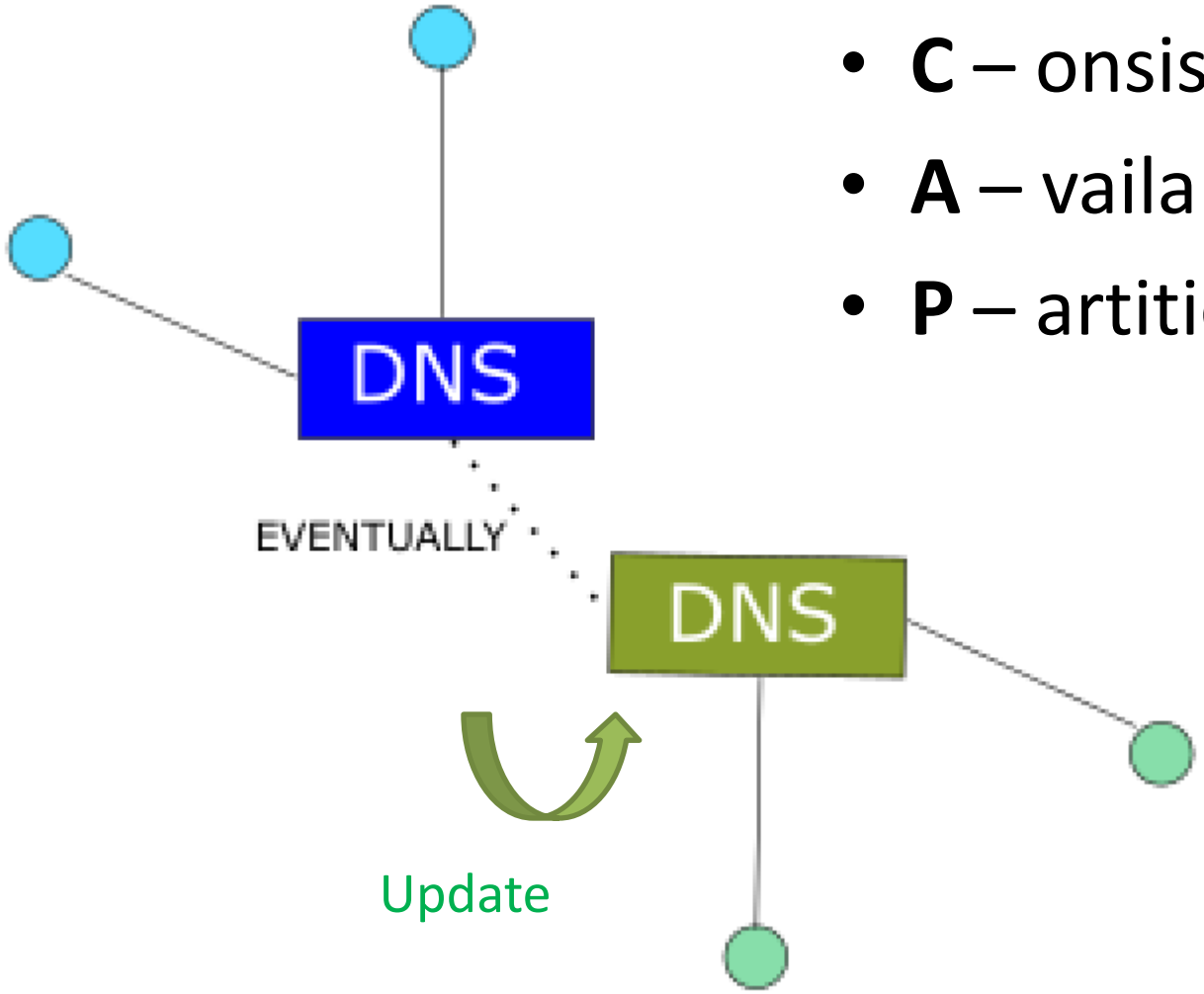
Decomposing the Domain



CQRS - TRANSACTIONAL MODEL

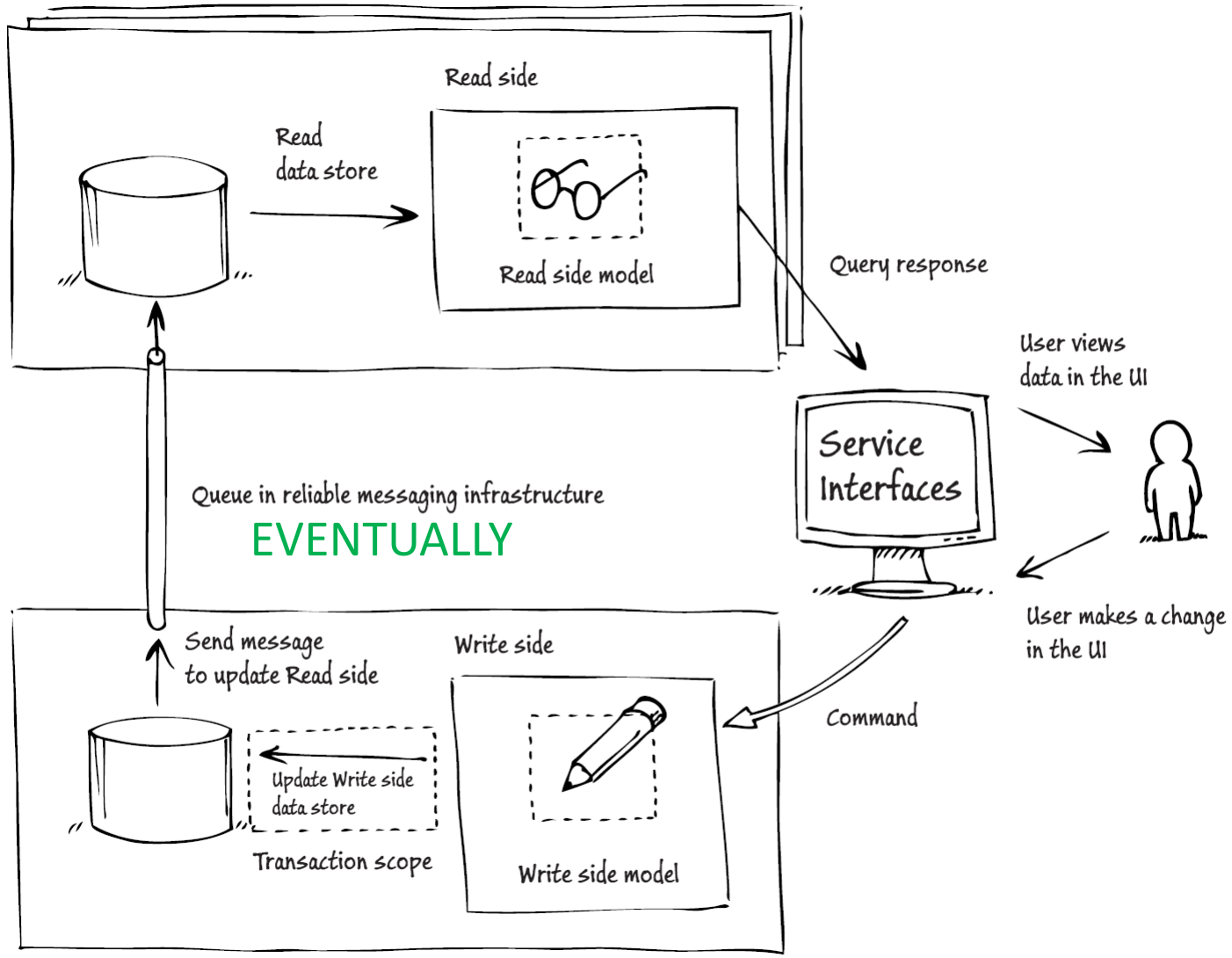


EVENTUAL CONSISTENCY



- **C** – onsistency
- **A** – vailability
- **P** – artition Tollerance

EVENTUAL CONSISTENCY ONLY WITH WRITE SIDE- CQRS -



CQRS – ORM SOLUTION

```
public class OrderCommandHandler :
      ICommandHandler<RegisterToConference>,
      ICommandHandler<MarkSeatsAsReserved>,
      ICommandHandler<RejectOrder>,
      ICommandHandler<AssignRegistrantDetails>,
      ICommandHandler<ConfirmOrder>
{
      private readonly IEventSourcedRepository<Order> repository;

      public OrderCommandHandler(IEventSourcedRepository<Order> repository)
      {
              this.repository = repository;
      }

      public void Handle(RegisterToConference command)
      {
              var items = command.Seats.Select(t => new OrderItem(t.SeatType,
               t.Quantity)).ToList();
              var order = repository.Find(command.OrderId);
              if (order == null)
              {
                      order = new Order(command.OrderId, command.ConferenceId, items);
              }
```

```
                else
                {
                        order.UpdateSeats(items);
                }

                repository.Save(order, command.Id.ToString());
        }

        public void Handle(ConfirmOrder command)
        {
                var order = repository.Get(command.OrderId);
                order.Confirm();
                repository.Save(order, command.Id.ToString());
        }

        public void Handle(AssignRegistrantDetails command)
        {
                ...
        }

        public void Handle(MarkSeatsAsReserved command)
        {
                ...
        }

        public void Handle(RejectOrder command)
        {
                ...
        }
}
```

EVE NT SOURCING

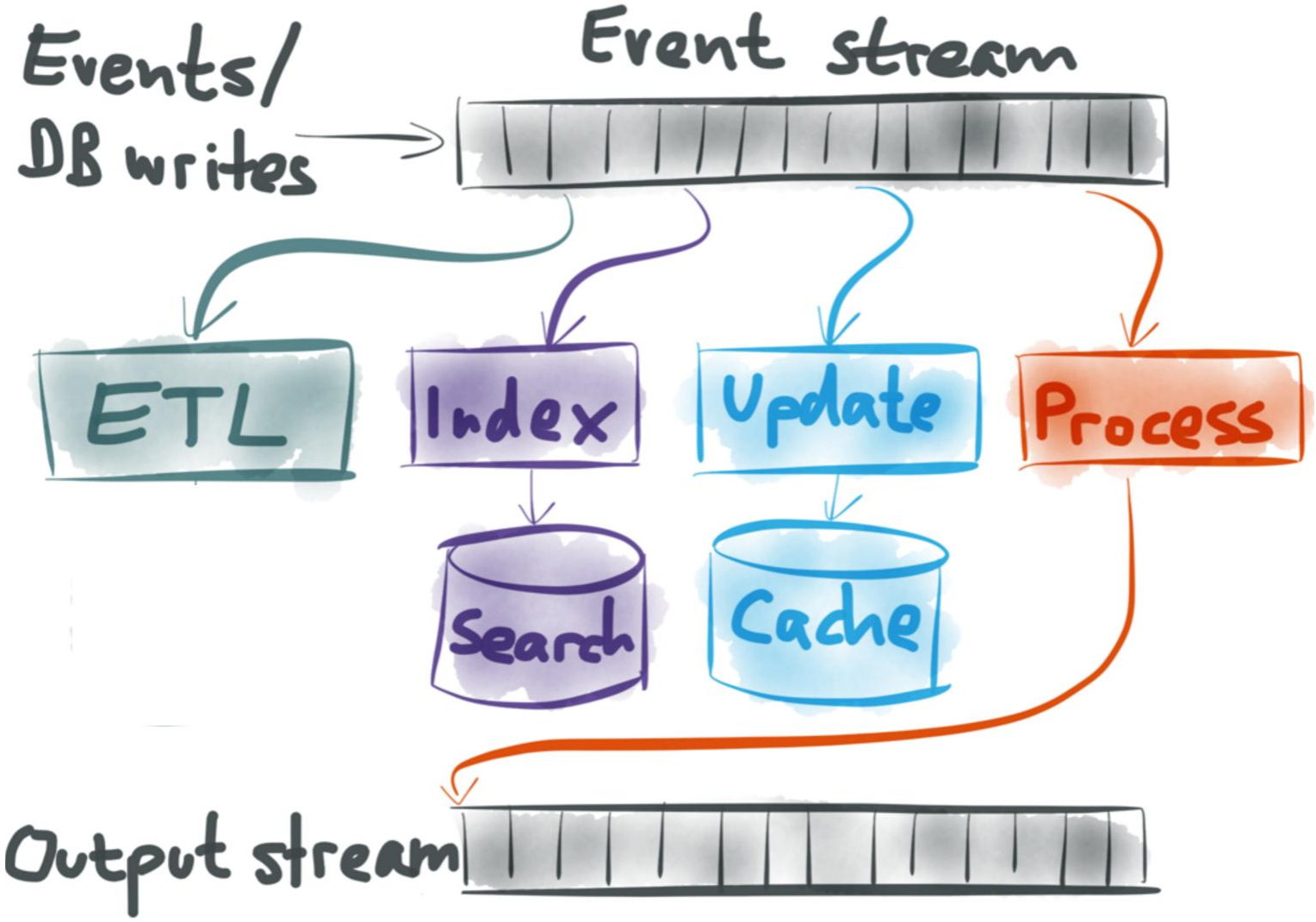
```
public interface IEventSourced
{
    Guid Id { get; }

    int Version { get; }

    IEnumerable<IVersionedEvent> Events { get; }
}
...
public abstract class EventSourced : IEventSourced
{
    private readonly Dictionary<Type, Action<IVersionedEvent>> handlers =
        new Dictionary<Type, Action<IVersionedEvent>>();
    private readonly List<IVersionedEvent> pendingEvents =
        new List<IVersionedEvent>();

    private readonly Guid id;
    private int version = -1;
```

EVE NT SOURCING



EVE NT SOURCING

```
    protected void Handles<TEvent>(Action<TEvent> handler)
    {
        this.handlers.Add(typeof(TEvent), @event => handler((TEvent)@event));
    }

    protected void LoadFrom(IEnumerable<IVersionedEvent> pastEvents)
    {
        foreach (var e in pastEvents)
        {
            this.handlers[e.GetType()].Invoke(e);
            this.version = e.Version;
        }
    }

    protected void Update(VersionedEvent e)
    {
        e.SourceId = this.Id;
        e.Version = this.version + 1;
        this.handlers[e.GetType()].Invoke(e);
        this.version = e.Version;
        this.pendingEvents.Add(e);
    }
}
```

EVE NT SOURCING

```
public class Order : EventSourced
{
    private List<SeatQuantity> seats;

    protected Order(Guid id) : base(id)
    {
        base.Handles<OrderUpdated>(this.OnOrderUpdated);
        ...
    }

    public Order(Guid id, IEnumerable<IVersionedEvent> history) : this(id)
    {
        this.LoadFrom(history);
    }

    public void UpdateSeats(IEnumerable<OrderItem> seats)
    {
        this.Update(new OrderUpdated { Seats = ConvertItems(seats) });
    }
```

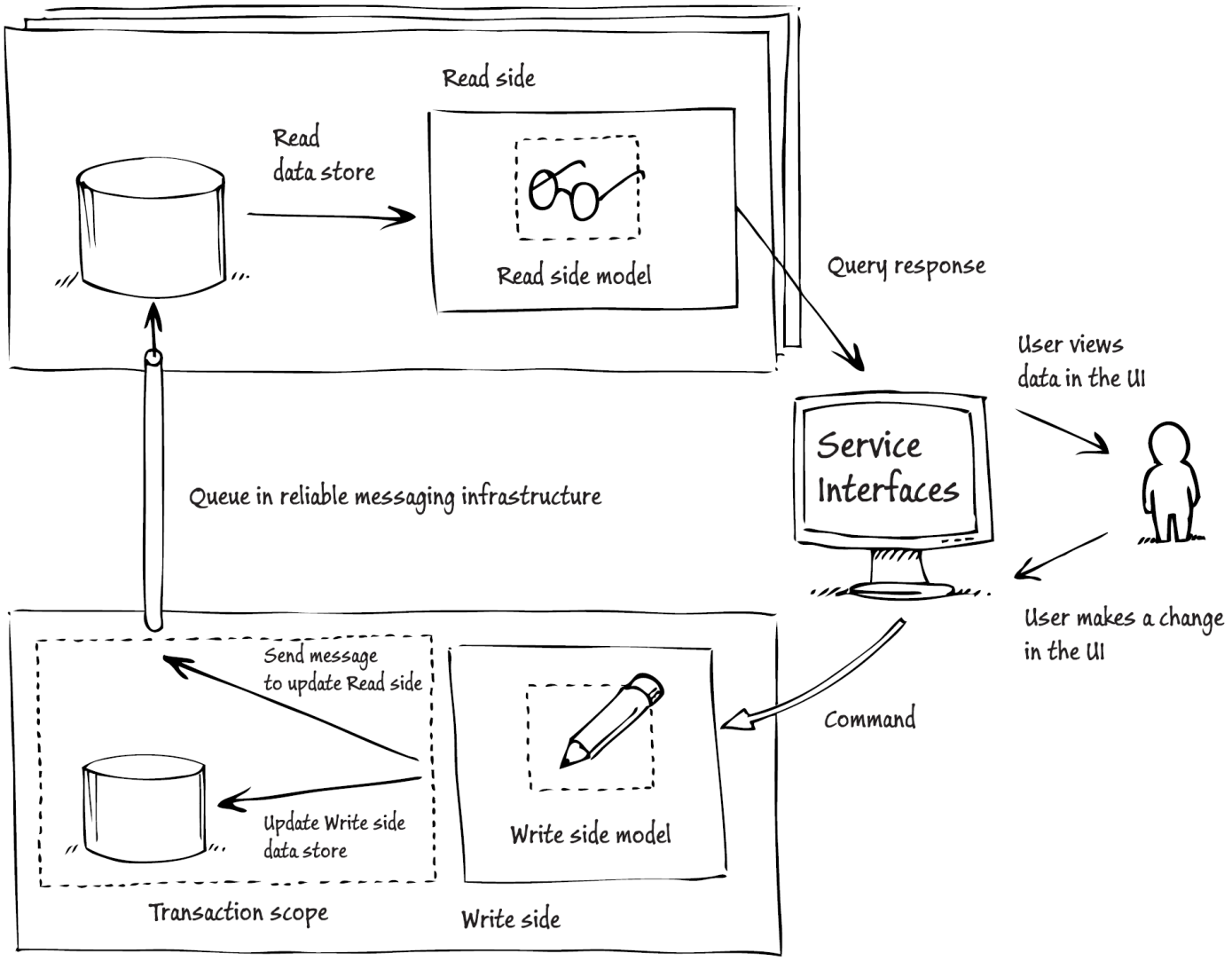
CQRS- ES

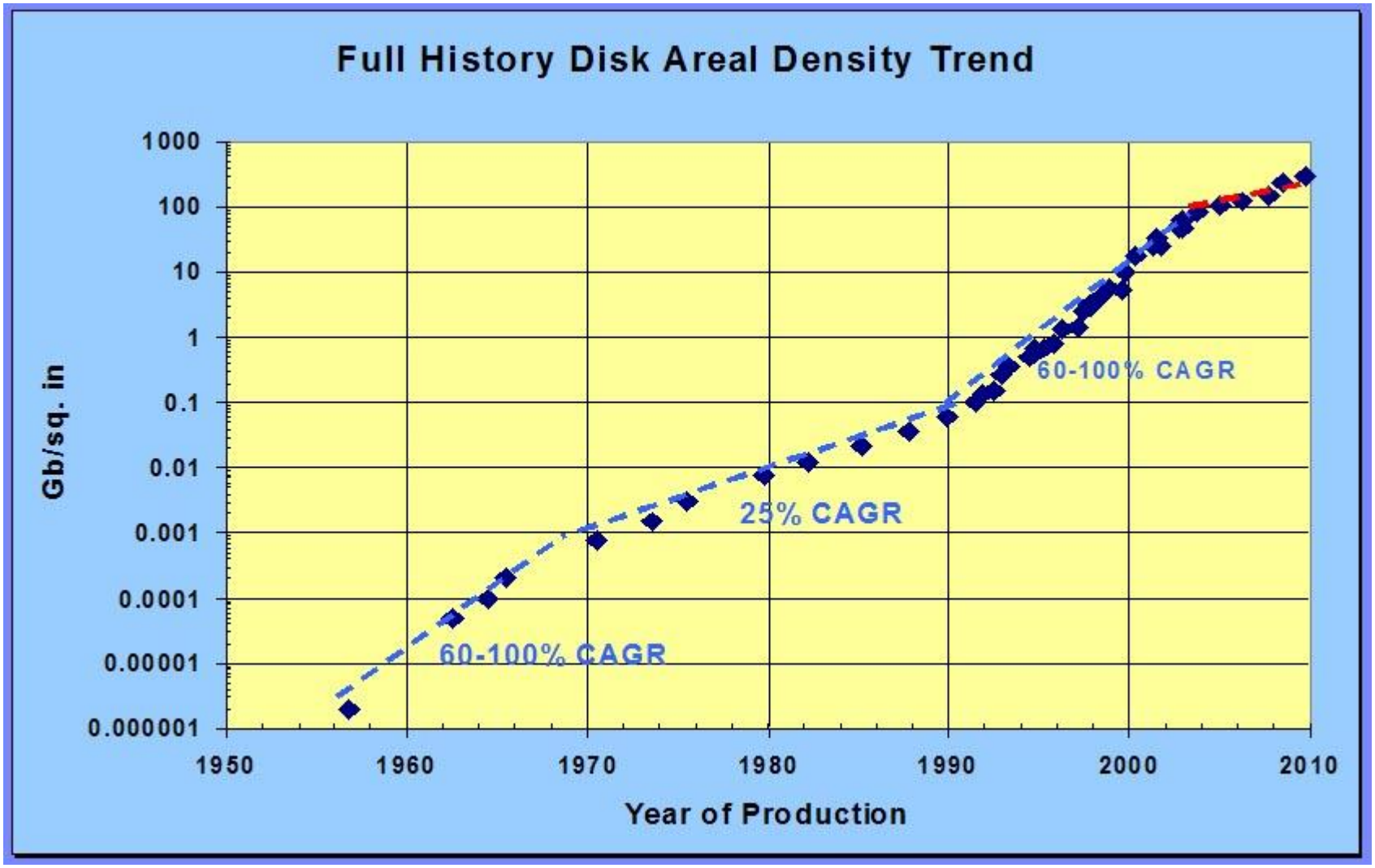
- MODELLING EVENTS FORCES BEHAVIOURAL FOCUS
- MODELLING EVENTS FORCES TEMPORAL FOCUS (DDD **EVENT STORMING**)
- **ES IS A NATURAL FUNCTIONAL MODEL**
- INPUTS != OUTPUT
- (CQRS) COMMANDS CAN RETURN

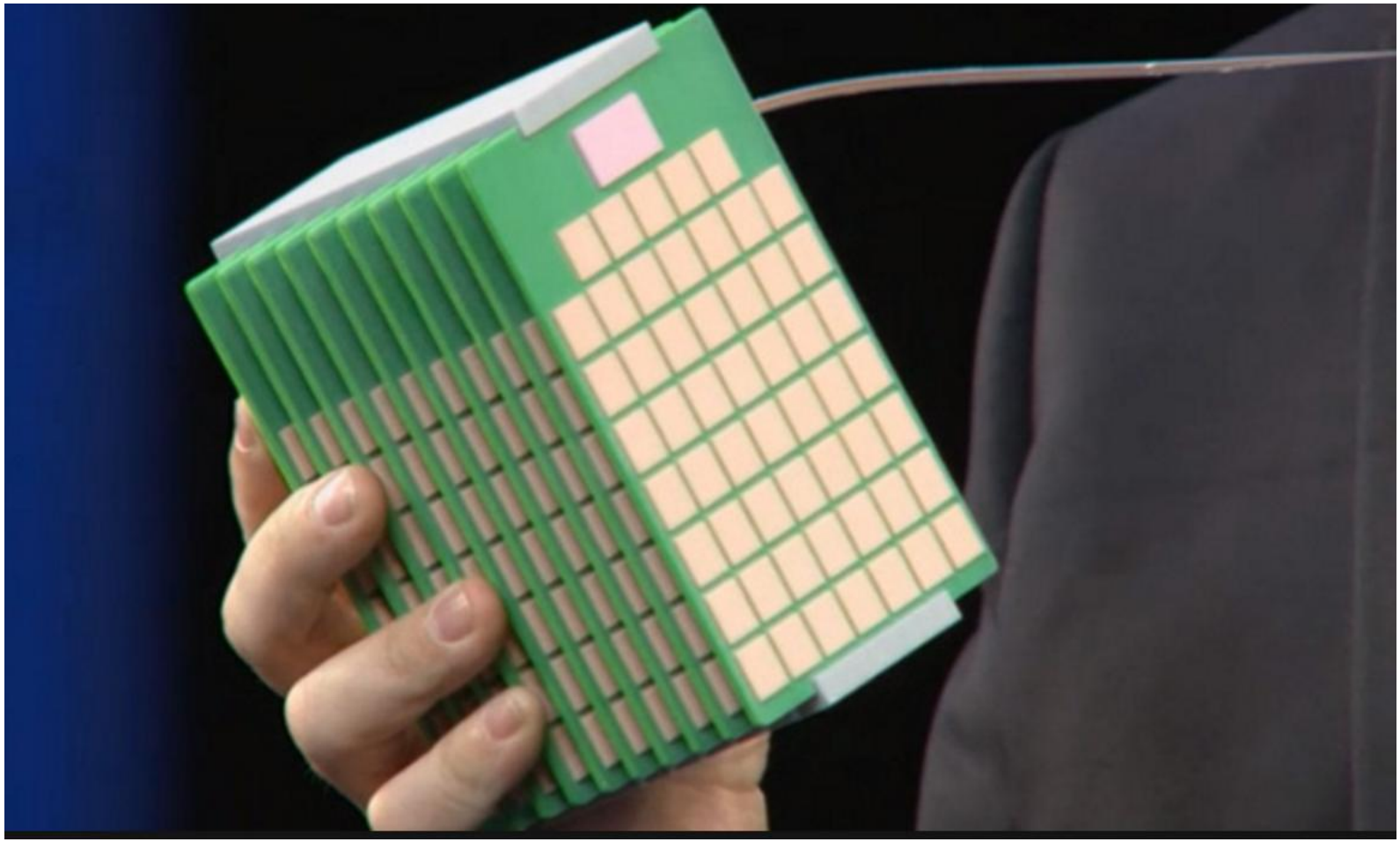
THANK YOU

- https://channel9.msdn.com/events/patterns-practices-symposium-online/patterns-practices-symposium-online-2012/a-journey-into-cqrs1

EVENTUAL CONSISTENCY ONLY WITH WRITE SIDE- CQRS -




Full History Disk Areal Density Trend
Gb/sq. in
1000
100
10
1
0.1
0.01
0.001
0.0001
0.00001
0.000001
1950
1960
1970
1980
1990
2000
2010
Year of Production
60-100% CAGR
25% CAGR
60-100% CAGR



Element	Type	Sends	Receives
ConferenceController	MVC Controller	N/A	ConferenceDetails
OrderController	MVC Controller	AssignSeat UnassignSeat	DraftOrder OrderSeats PricedOrder
RegistrationController	MVC Controller	RegisterToConference AssignRegistrantDetails InitiateThirdParty-ProcessorPayment	DraftOrder PricedOrder SeatType
PaymentController	MVC Controller	CompleteThirdParty-ProcessorPayment CancelThirdParty-ProcessorPayment	ThirdPartyProcessor-PaymentDetails
Conference Management	CRUD Bounded Context	ConferenceCreated ConferenceUpdated ConferencePublished ConferenceUnpublished SeatCreated SeatUpdated	OrderPlaced OrderRegistrantAssigned OrderTotalsCalculated OrderPaymentConfirmed SeatAssigned SeatAssignmentUpdated SeatUnassigned
Order	Aggregate	OrderPlaced *OrderExpired *OrderUpdated *OrderPartiallyReserved *OrderReservation-Completed *OrderPaymentConfirmed *OrderRegistrantAssigned	RegisterToConference MarkSeatsAsReserved RejectOrder AssignRegistrantDetails ConfirmOrderPayment
SeatsAvailability	Aggregate	SeatsReserved *AvailableSeatsChanged *SeatsReservation-Committed *SeatsReservationCancelled	MakeSeatReservation CancelSeatReservation CommitSeatReservation AddSeats RemoveSeats