# CQRS Best Practices and Misconceptions

Vladimir Khorikov

@vkhorikov www.enterprisecraftsmanship.com

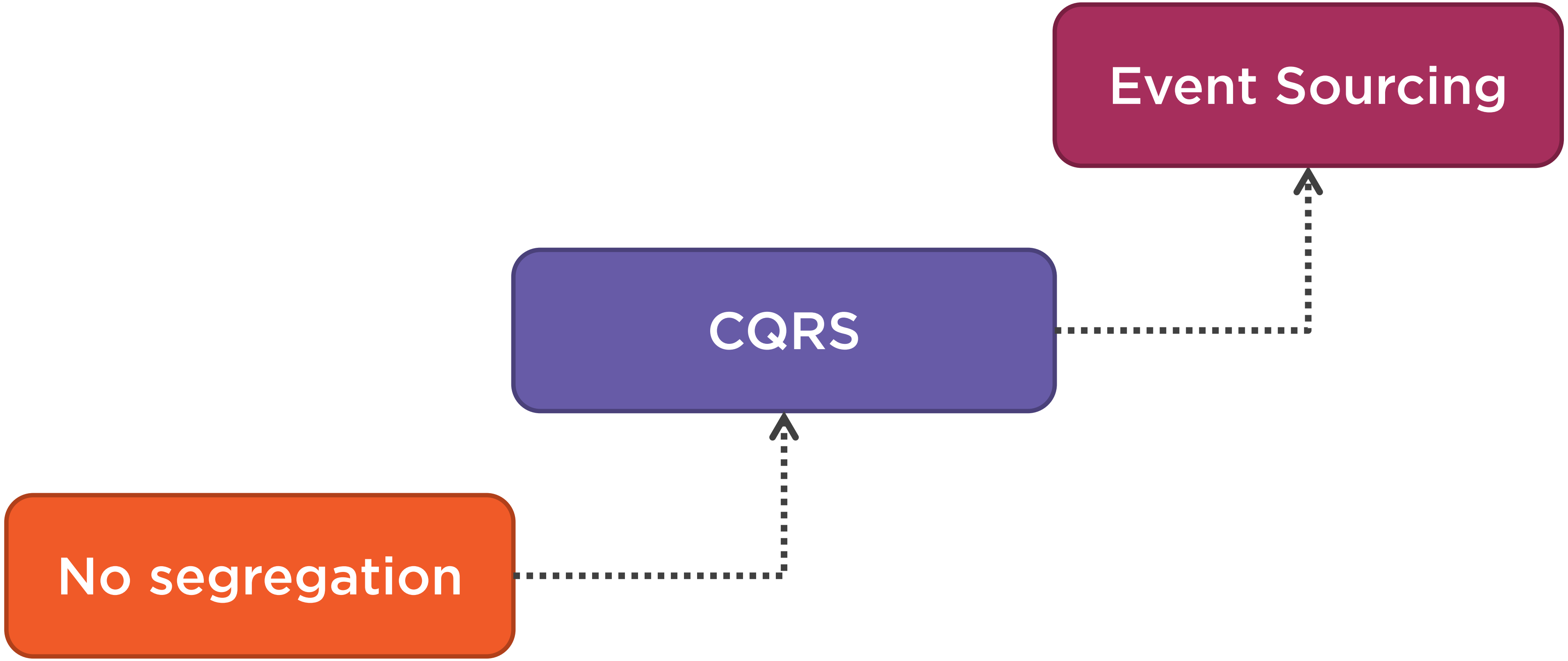
# CQRS and Event Sourcing

CQRS

?

Event Sourcing

# CQRS and Event Sourcing



CQRS can provide a lot of benefits without Event Sourcing.

# CQRS and Event Sourcing

The bar for Event Sourcing is much higher than it is for CQRS

There's still a lot of value in event sourced systems

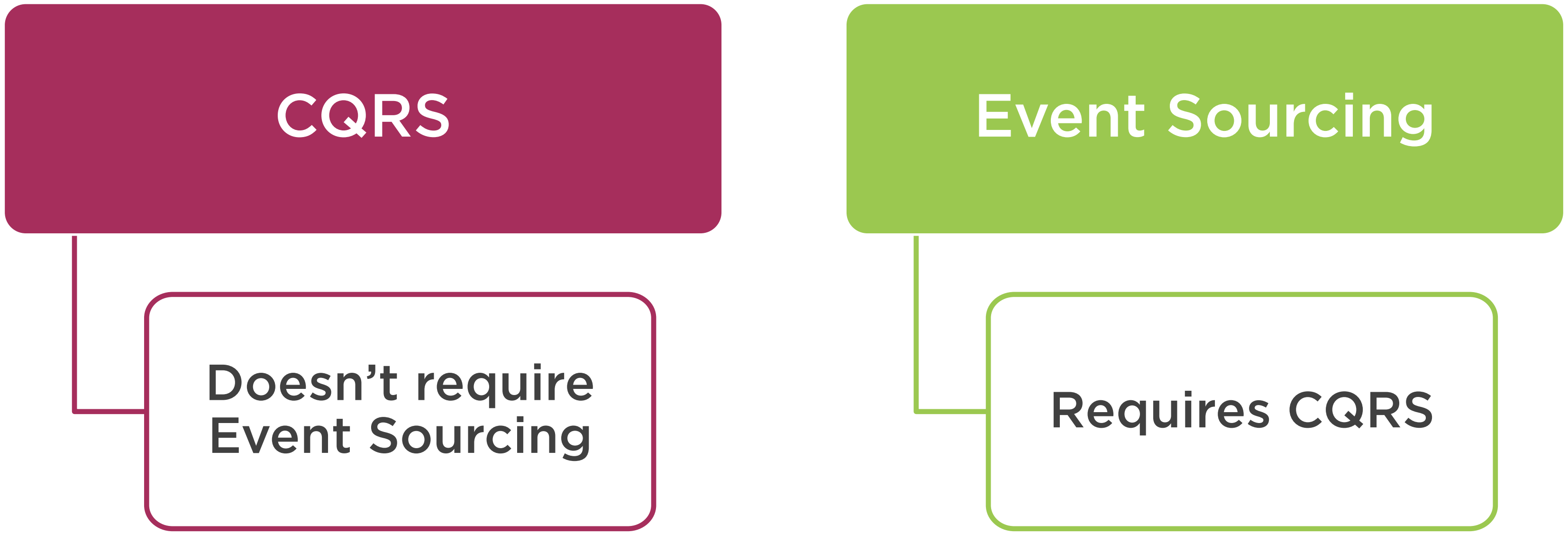
# CQRS and Event Sourcing

**Finance tech**

Has to keep track of all events

A trail of the financial transactions

# CQRS and Event Sourcing



CQRS

Doesn't require Event Sourcing

Event Sourcing

Requires CQRS

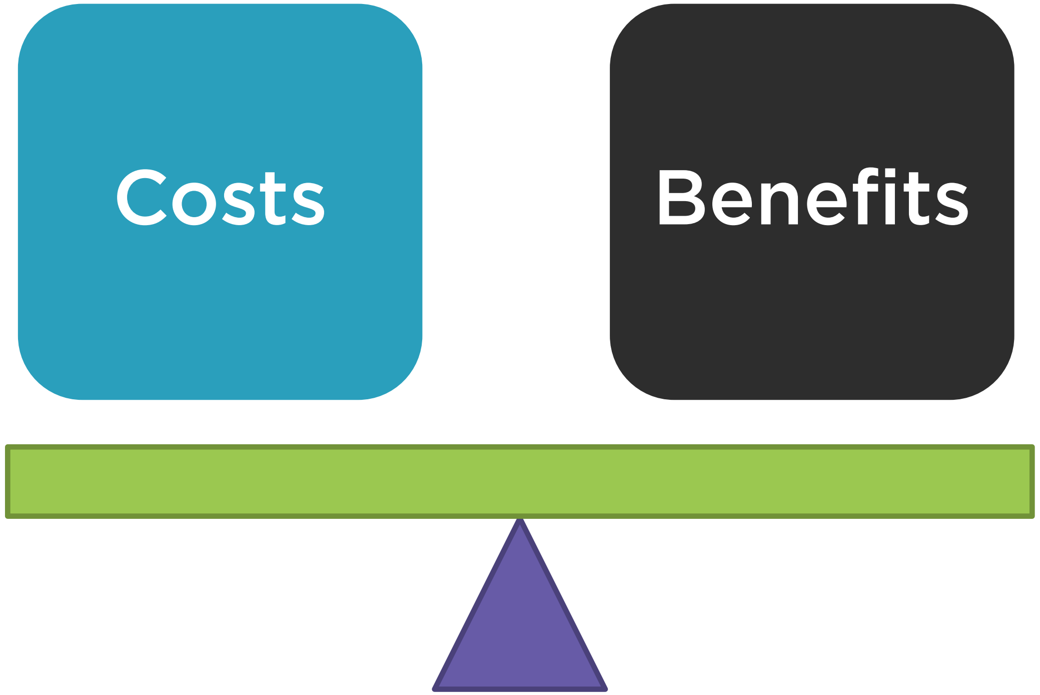
⚠ **Event Sourcing without CQRS is a less scalable solution**

# Evolutionary Design

You don't have to implement all the techniques from this course

Defer these decisions until you have proven their need

# Evolutionary Design

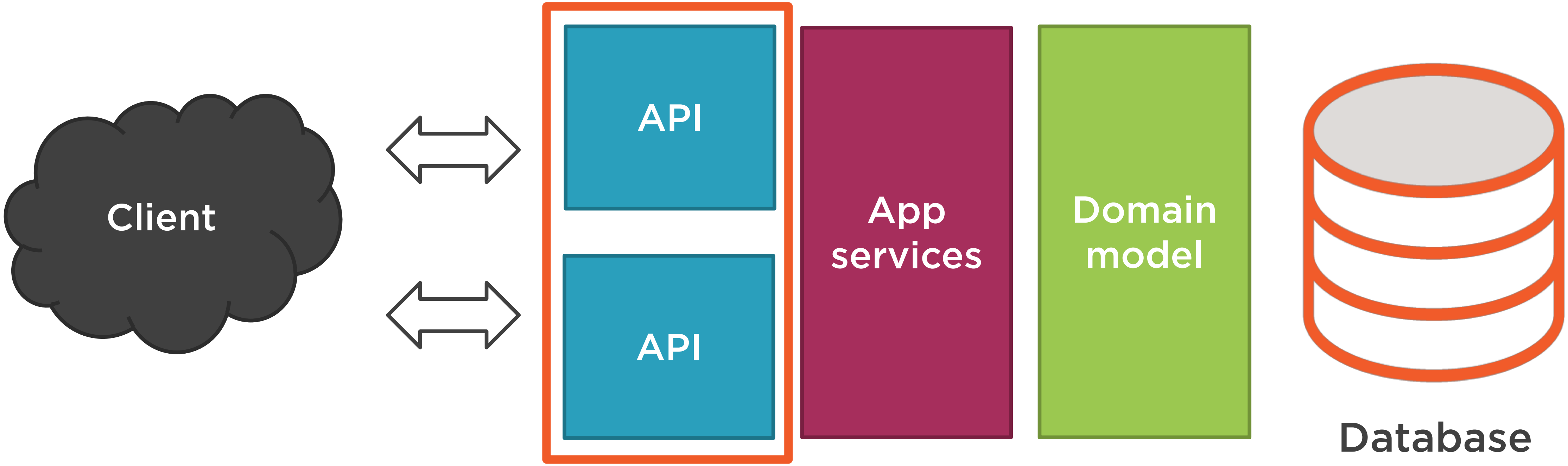


Costs

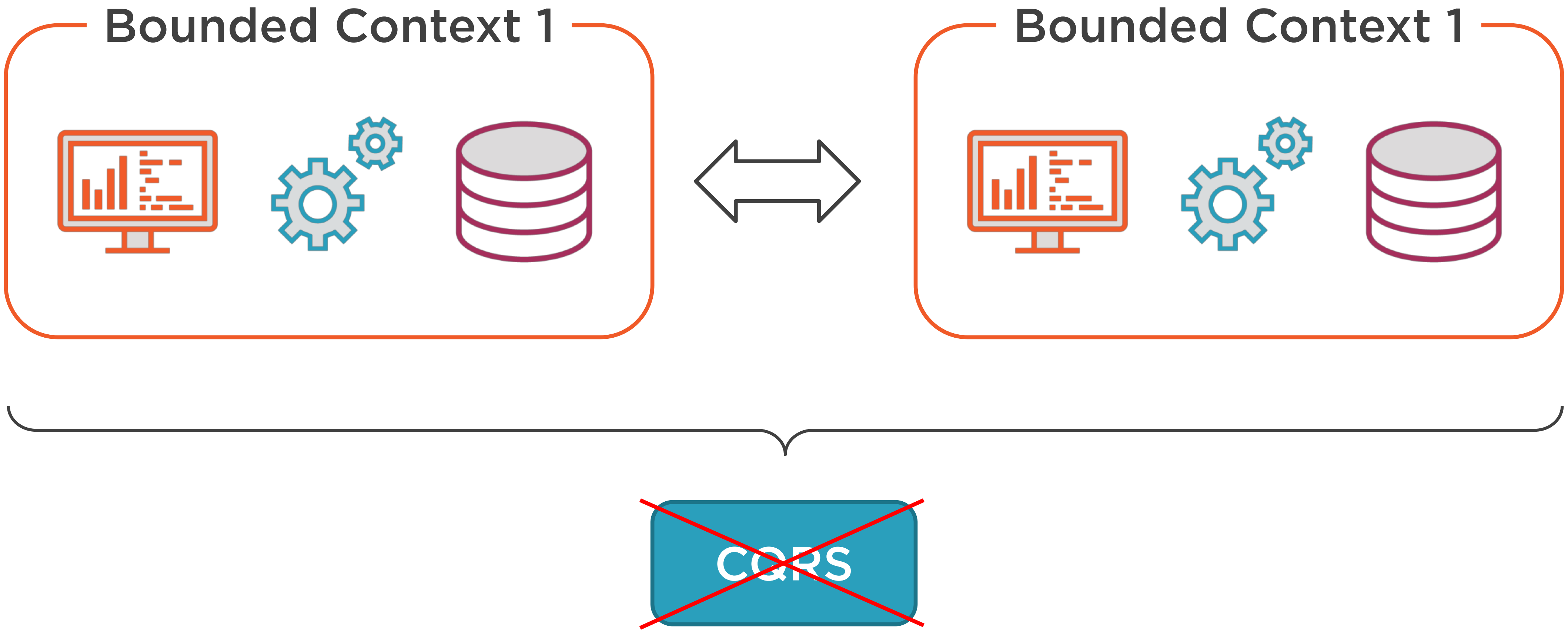
Benefits

✓ Ensure the benefits outweigh the costs before applying a pattern

# Evolutionary Design



# Evolutionary Design

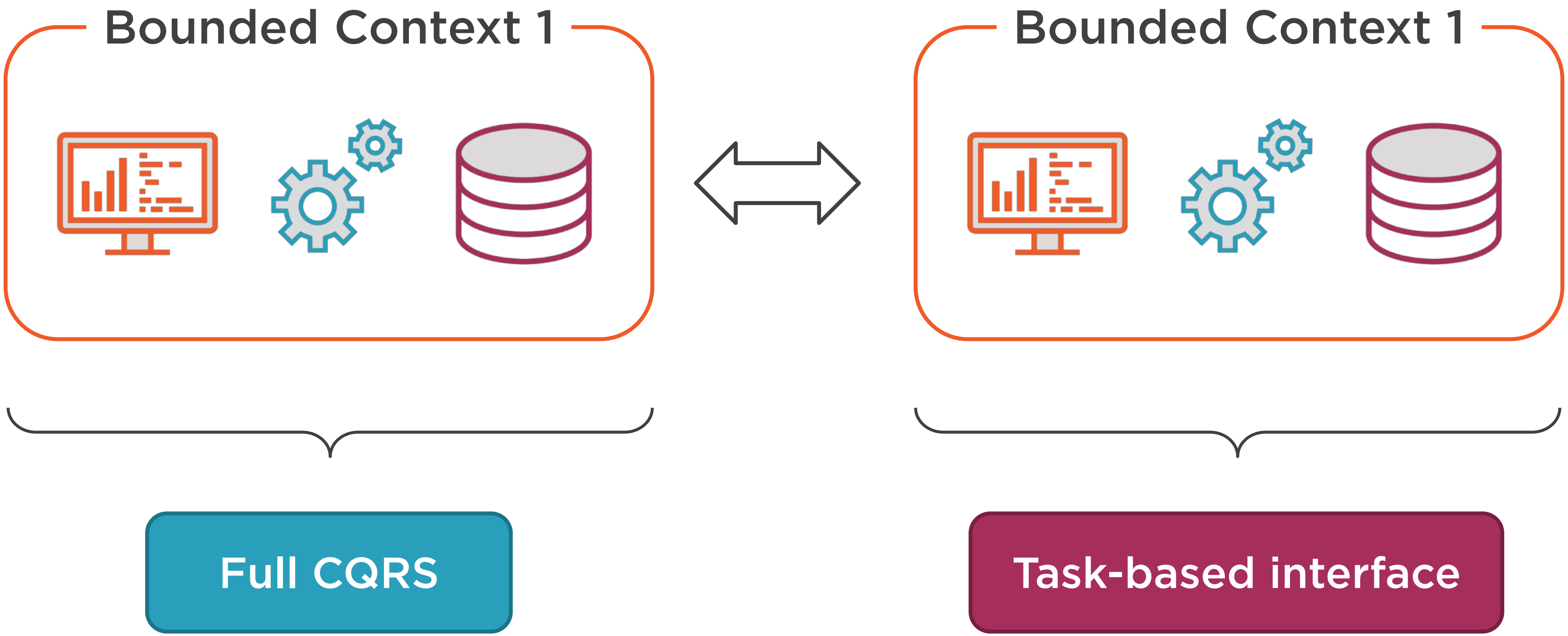


Bounded Context 1

Bounded Context 1

~~CQRS~~

# Evolutionary Design



Bounded Context 1

Bounded Context 1

Full CQRS

Task-based interface

✓ Separation down to the domain model level is the sweet spot

# Using Commands and Queries from Handlers

**Can you use other commands and queries from command and query handlers?**

# Using Commands and Queries from Handlers

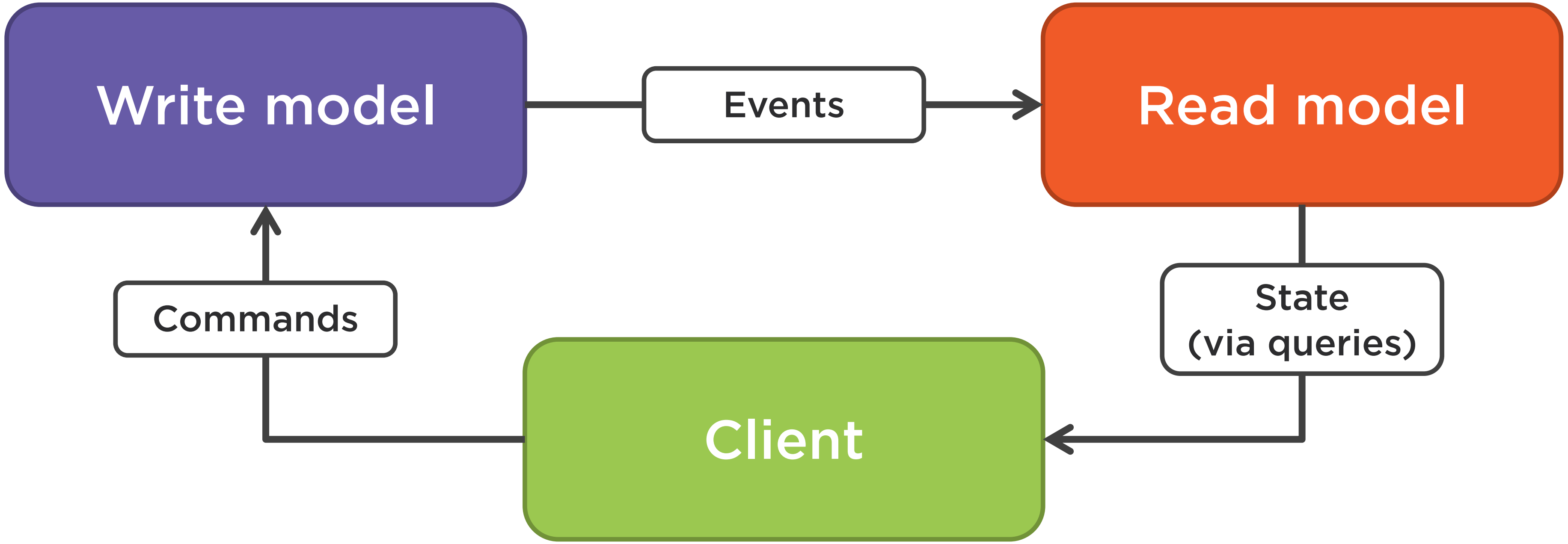
| | Command | Query |
|---|---|---|
| Command handler | ? | |
| Query handler | | |

# Using Commands and Queries from Handlers

```
public sealed class UnregisterCommandHandler : ICommandHandler<UnregisterCommand>
{
    public Result Handle(UnregisterCommand command)
    {
        var unitOfWork = new UnitOfWork(_sessionFactory);
        var repository = new StudentRepository(unitOfWork);
        Student student = repository.GetById(command.Id);
        if (student == null)
            return Result.Fail($"No student found for Id {command.Id}");

        _messages.Dispatch(new DisenrollCommand(student.Id, 0, "Unregistering"));
        _messages.Dispatch(new DisenrollCommand(student.Id, 1, "Unregistering"));

        repository.Delete(student);
        unitOfWork.Commit();

        return Result.Ok();
    }
}
```

✗ **Our application shouldn't trigger commands**

# Using Commands and Queries from Handlers



- ✓ Command: what can be done with application
- ✓ Event: what has happened to the application
- ✓ State: sum of all the events

# Using Commands and Queries from Handlers

| | Command | Query |
|---|---|---|
| Command handler | ✗ Nope | ? |
| Query handler | | |

# Using Commands and Queries from Handlers

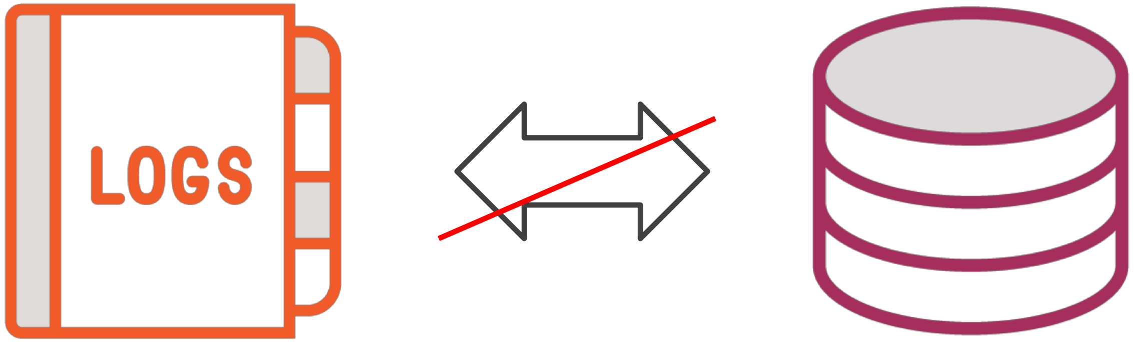
**Do you have a reliable way to request the current state in the command side?**

✓ **Query the command database directly**

✗ **Use a query from a command handler**

# Using Commands and Queries from Handlers

## Event Sourcing



Event log

State

✓ Have to use the read database

⚠ Whole set of issues related to consistency

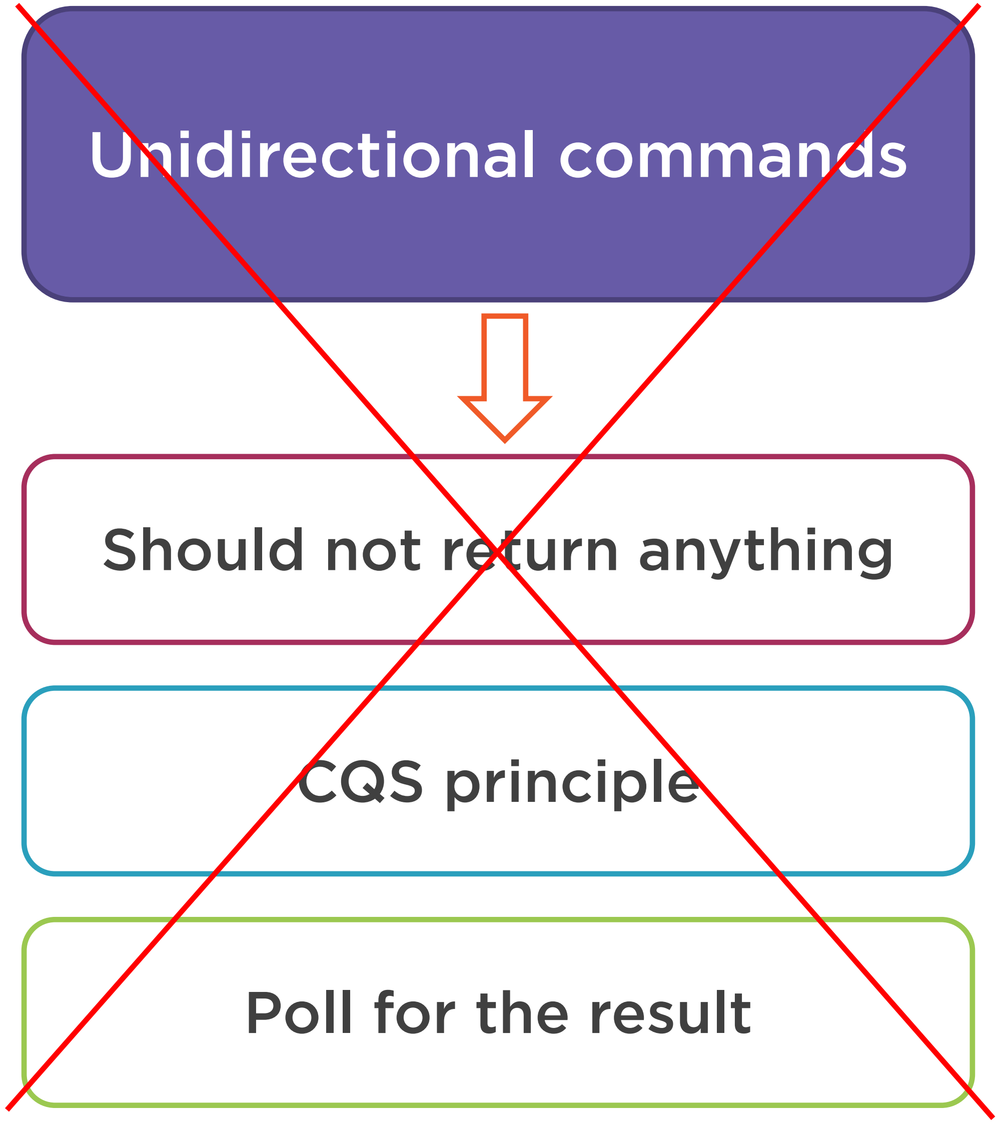
# Using Commands and Queries from Handlers

| | Command | Query |
|---|---|---|
| Command handler | ✗ Nope | ✗ Only with Event Sourcing |
| Query handler | ? | ? |

# Using Commands and Queries from Handlers

| | Command | Query |
|---|---|---|
| Command handler | ✗ Nope | ✗ Only with Event Sourcing |
| Query handler | ✗ ? Nope | ? |

# One-way Commands



~~Unidirectional commands~~

~~Should not return anything~~

~~CQS principle~~

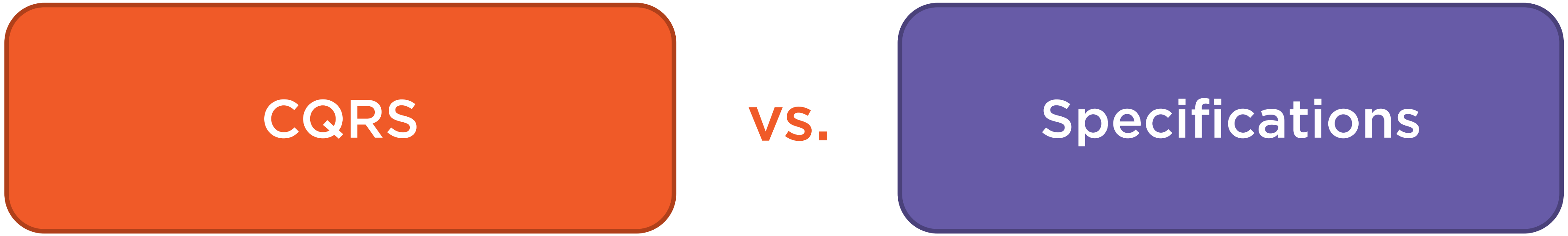
~~Poll for the result~~

# One-way Commands

Truly one-way commands are impossible

- ? Why return a locator?
- ✓ Complete the operation synchronously
- ✓ An Id is fine too

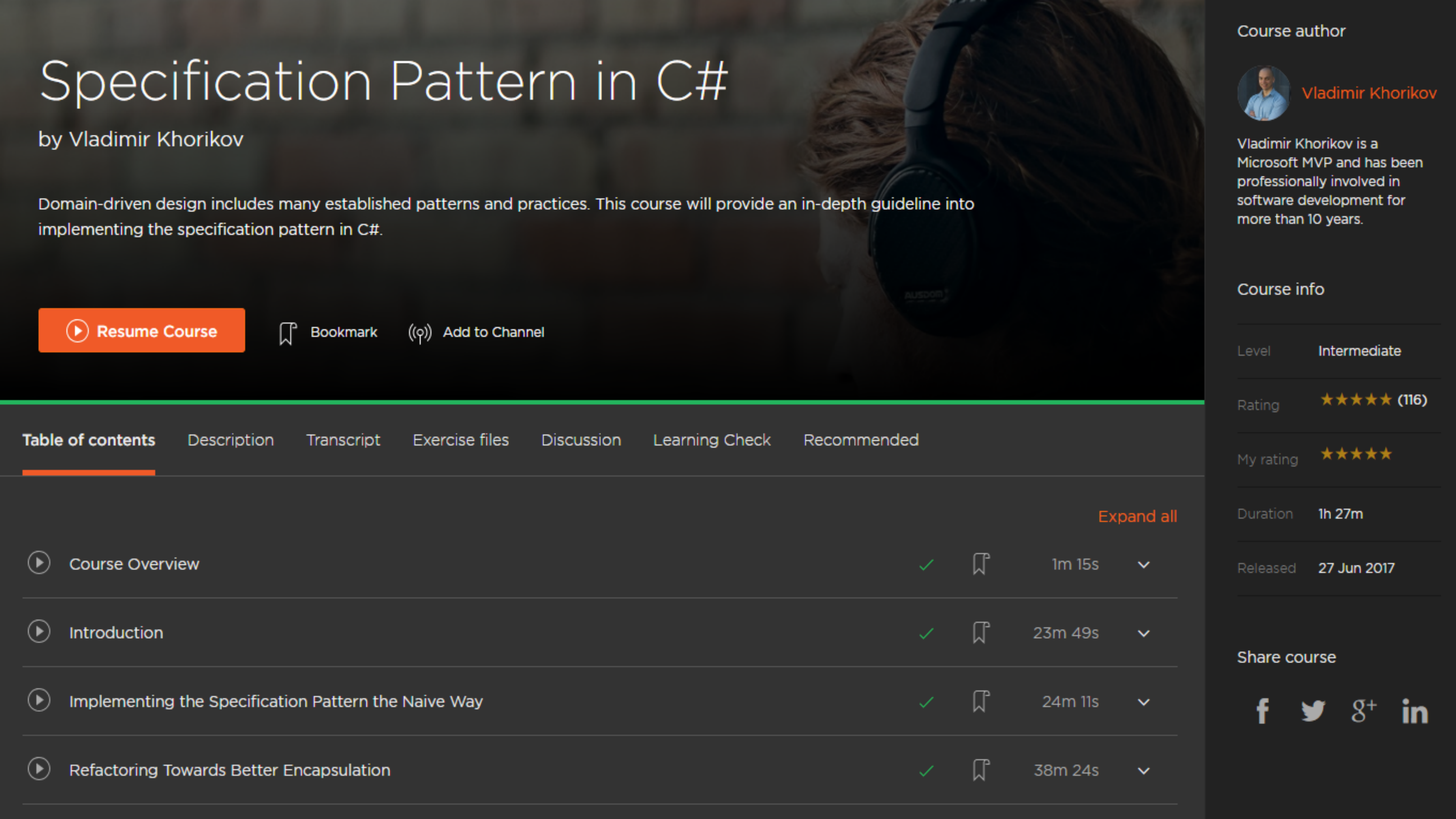
# CQRS vs. the Specification Pattern



CQRS

vs.

Specifications



# Specification Pattern in C#

by Vladimir Khorikov

Domain-driven design includes many established patterns and practices. This course will provide an in-depth guideline into implementing the specification pattern in C#.

| | | |
|---|---|---|
| Course Overview | ✓ | 1m 15s |
| Introduction | ✓ | 23m 49s |
| Implementing the Specification Pattern the Naive Way | ✓ | 24m 11s |
| Refactoring Towards Better Encapsulation | ✓ | 38m 24s |

Course author

Vladimir Khorikov

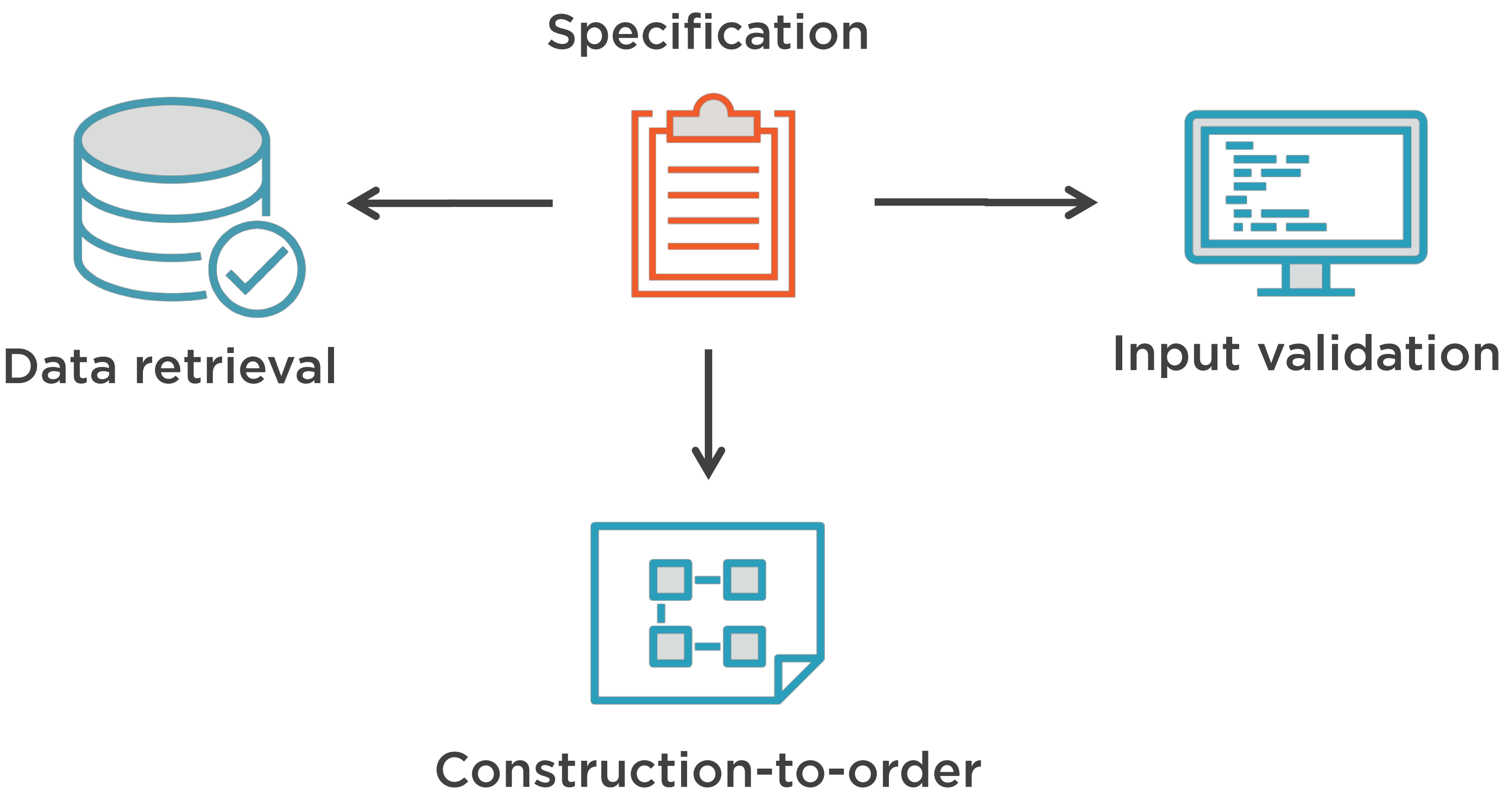
Vladimir Khorikov is a Microsoft MVP and has been professionally involved in software development for more than 10 years.

Course info

| | |
|---|---|
| Level | Intermediate |
| Rating | ★★★★★ (116) |
| My rating | ★★★★★ |
| Duration | 1h 27m |
| Released | 27 Jun 2017 |

Share course

# CQRS vs. the Specification Pattern



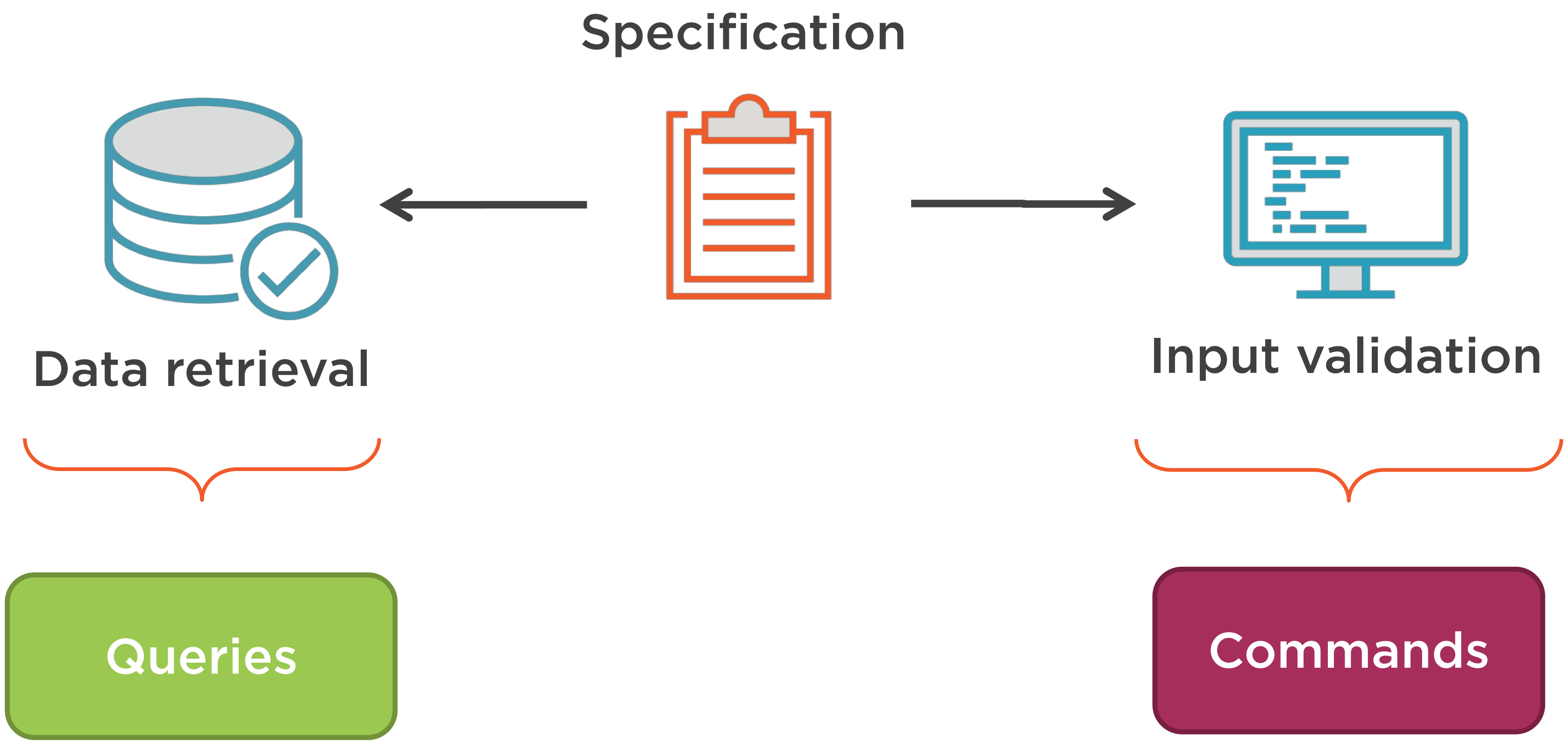
Specification

Data retrieval

Input validation

Construction-to-order

# CQRS vs. the Specification Pattern



Specification

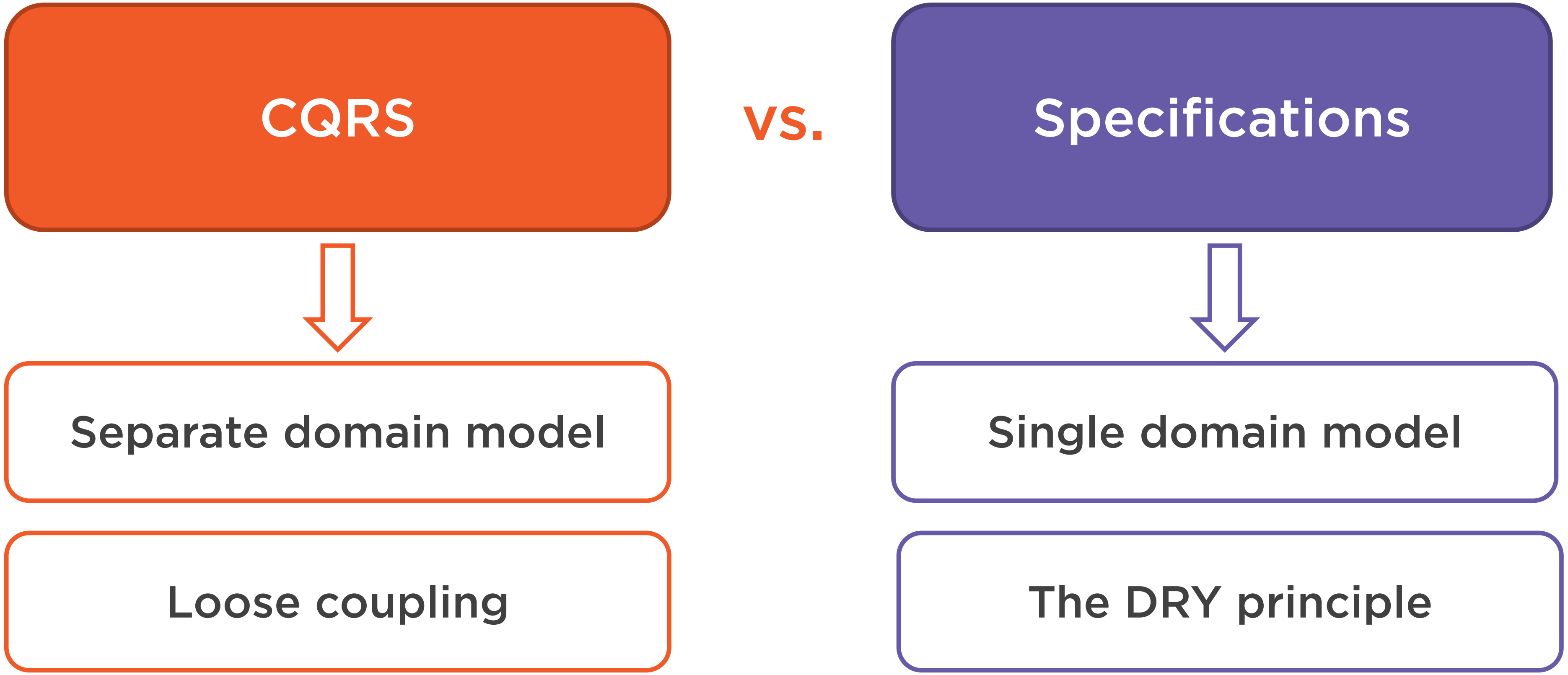
Data retrieval

Input validation

Queries

Commands

# CQRS vs. the Specification Pattern



CQRS vs. Specifications

CQRS → Separate domain model; Loose coupling

Specifications → Single domain model; The DRY principle

✓ Domain knowledge duplication is a lesser evil

✗ High coupling puts too many restrictions

# Resource List

| | |
|---|---|
| Source code | https://github.com/vkhorikov/CqrsInPractice |
| | http://bit.ly/cqrs-code |
| Domain-Driven Design in Practice | https://www.pluralsight.com/courses/domain-driven-design-in-practice |
| | http://bit.ly/ddd-ps |
| CQRS book by Greg Young | https://cqrs.files.wordpress.com/2010/11/cqrs_documents.pdf |
| Use of event listeners in NHibernate and the change tracker in Entity Framework | https://enterprisecraftsmanship.com/2018/06/13/ef-core-vs-nhibernate-ddd-perspective/ (Section 7) |
| | http://bit.ly/ef-vs-nh (Section 7) |
| Starbucks doesn't use two-phase commit | https://www.enterpriseintegrationpatterns.com/ramblings/18_starbucks.html |
| | http://bit.ly/starbucks-cons |

# Course Summary

**CQRS pattern: what it is and how to implement it in practice**

**Goals of CQRS:**

- Simplicity, performance, and scalability

**Gradually introduced the separation**

- API endpoints
- Explicit commands, queries, and handlers, decorators
- No domain model in reads
- Separate database

**Synchronization between the read and write databases**

**CQRS best practices and common misconceptions**

# Contacts

http://bit.ly/ddd-new

vladimir.khorikov@gmail.com

@vkhorikov

http://enterprisecraftsmanship.com/