

# “Beyond” DDD: uno sguardo a CQRS ed event sourcing

Alessandro Melchiori

alessandro@codiceplastico.com

<http://blogs.ugidotnet.org/amelchiori>

<http://twitter.com/amelchiori>

# Agenda

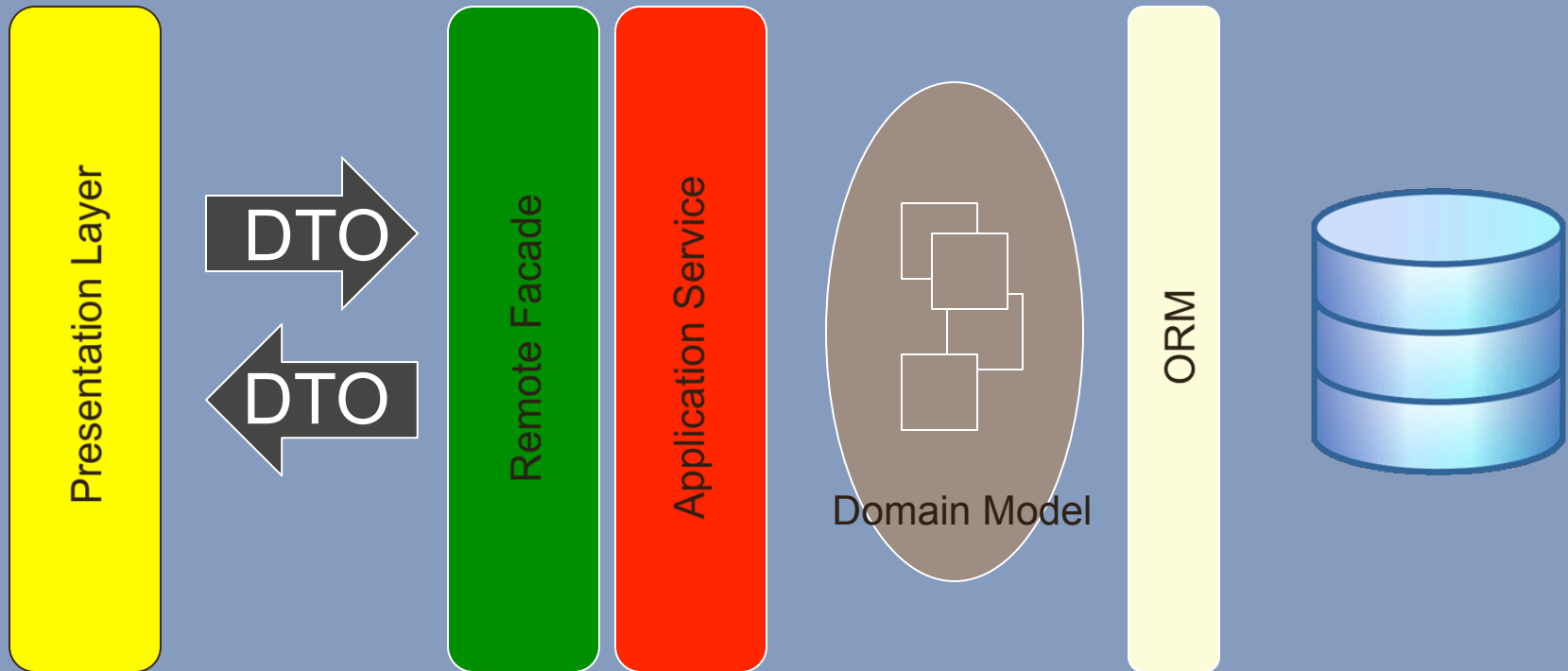
- Un'ora di stress-test...
- ...e poi parlate voi 😊

# Domain Model

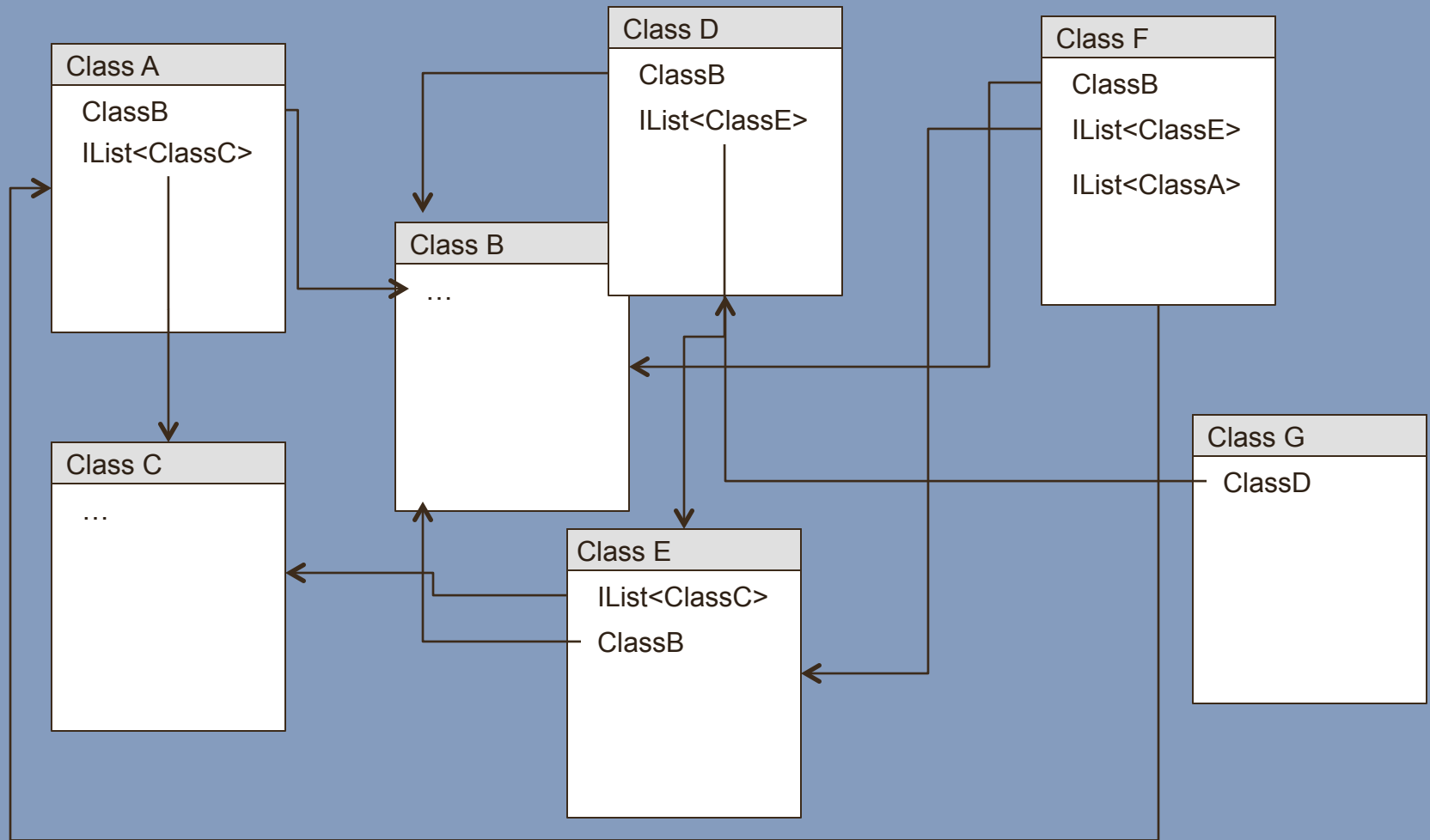
- An object model of the domain that incorporates both behavior and data

[Martin Fowler - <http://martinfowler.com/eaCatalog/domainModel.html>]

# Un'architettura per tutti i gusti...



# Domain Model

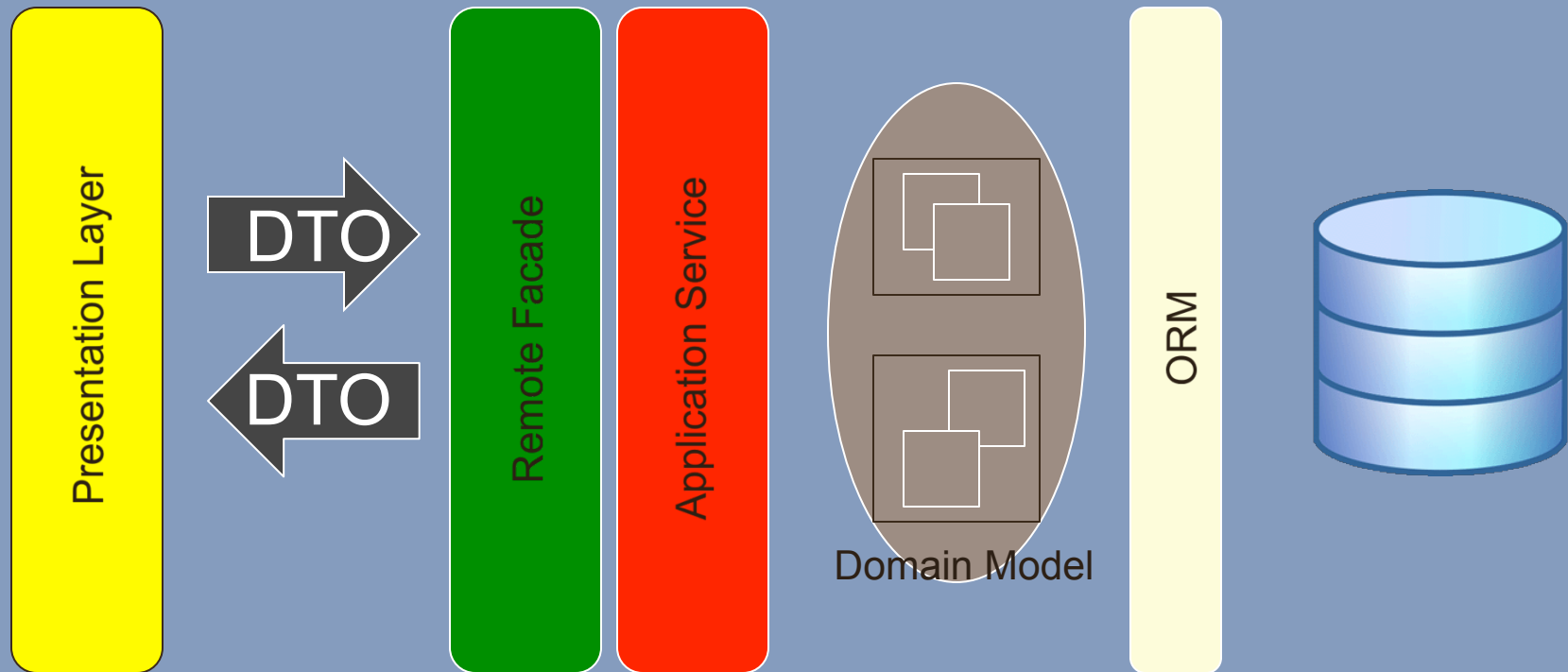


# Domain Driven Design

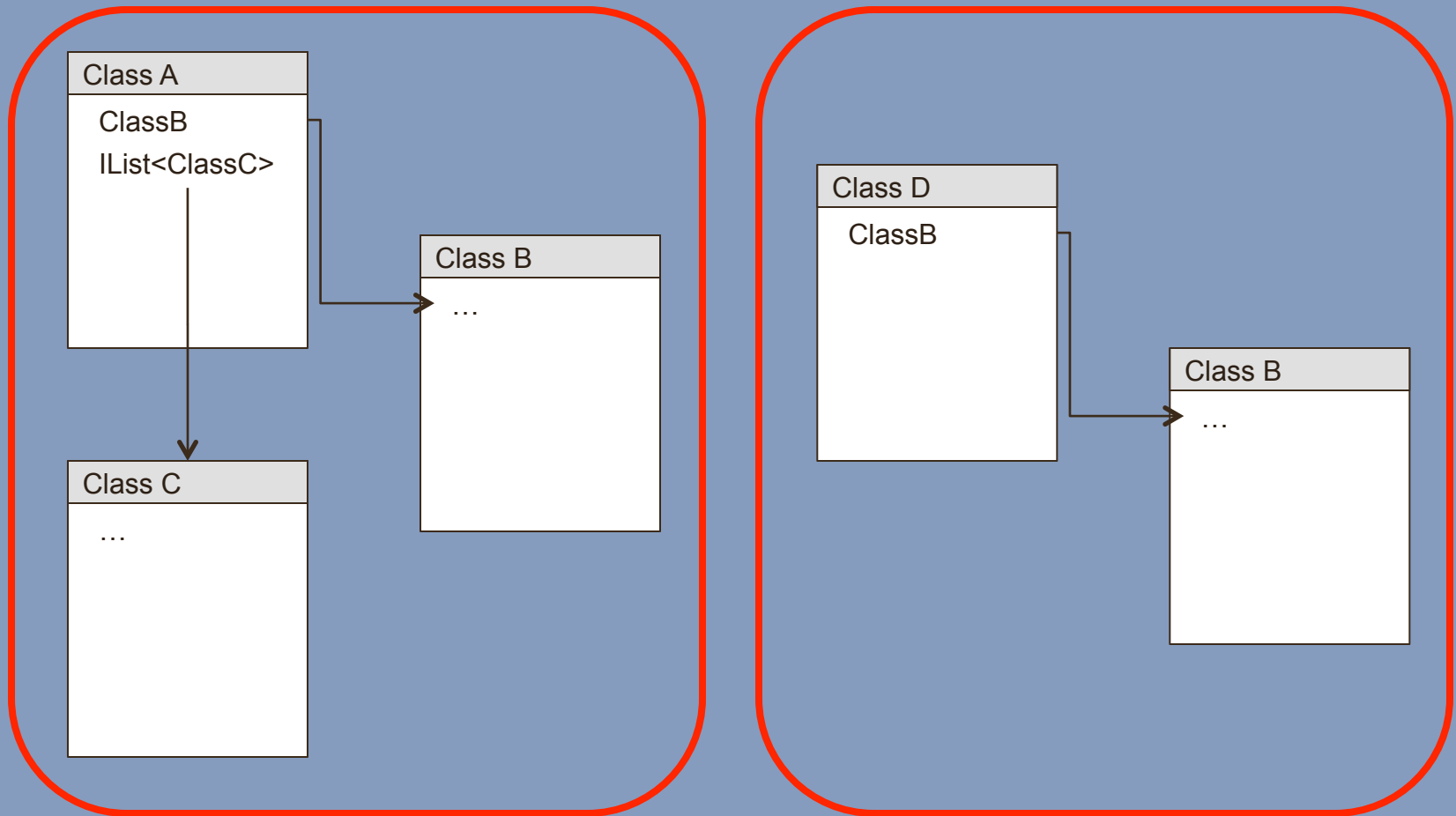
- Use AGGREGATES as unit of consistency across your domain model
- Protect your model with clearly defined BOUNDED CONTEXT

Il mio amico Eric Evans ☺

# Un'architettura per tutti i gusti...



# Aggregate e BoundedContext



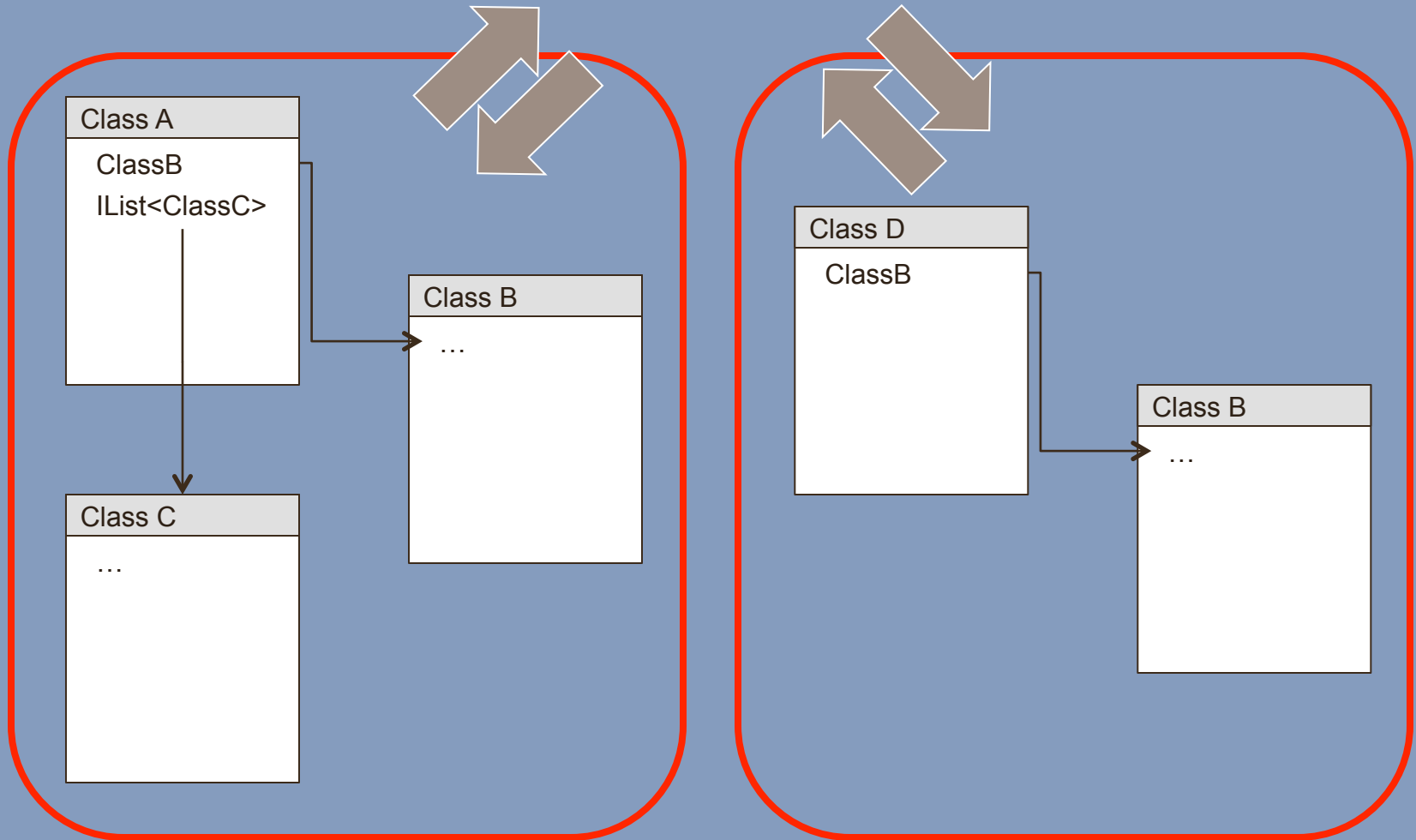


# Domain Events

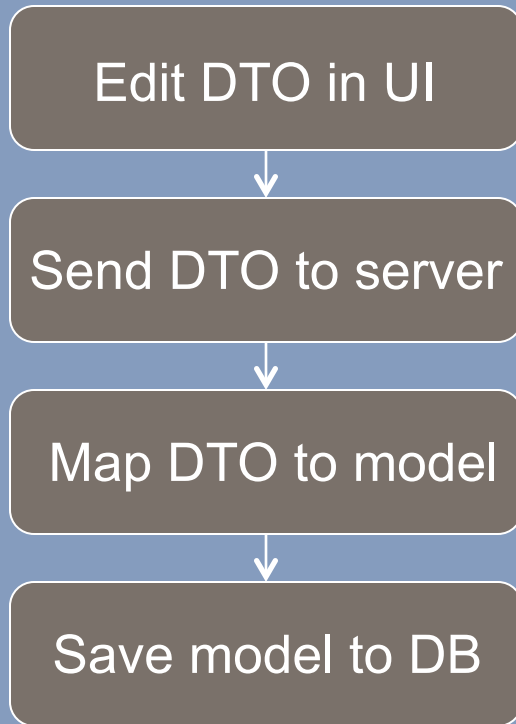
- It's really become clear to me in the last couple of years that we need a new building block and that is the Domain Events

Sempre il mio amico Eric Evans ☺

# Domain Events



# Qual è il problema?



- Aggiunta di un campo sulla UI
- Aggiunta di una property sul DTO
- Aggiunta della property sul DM
- Mapping tra DTO e DM
- Mapping ORM
- Aggiunta campo sul DB

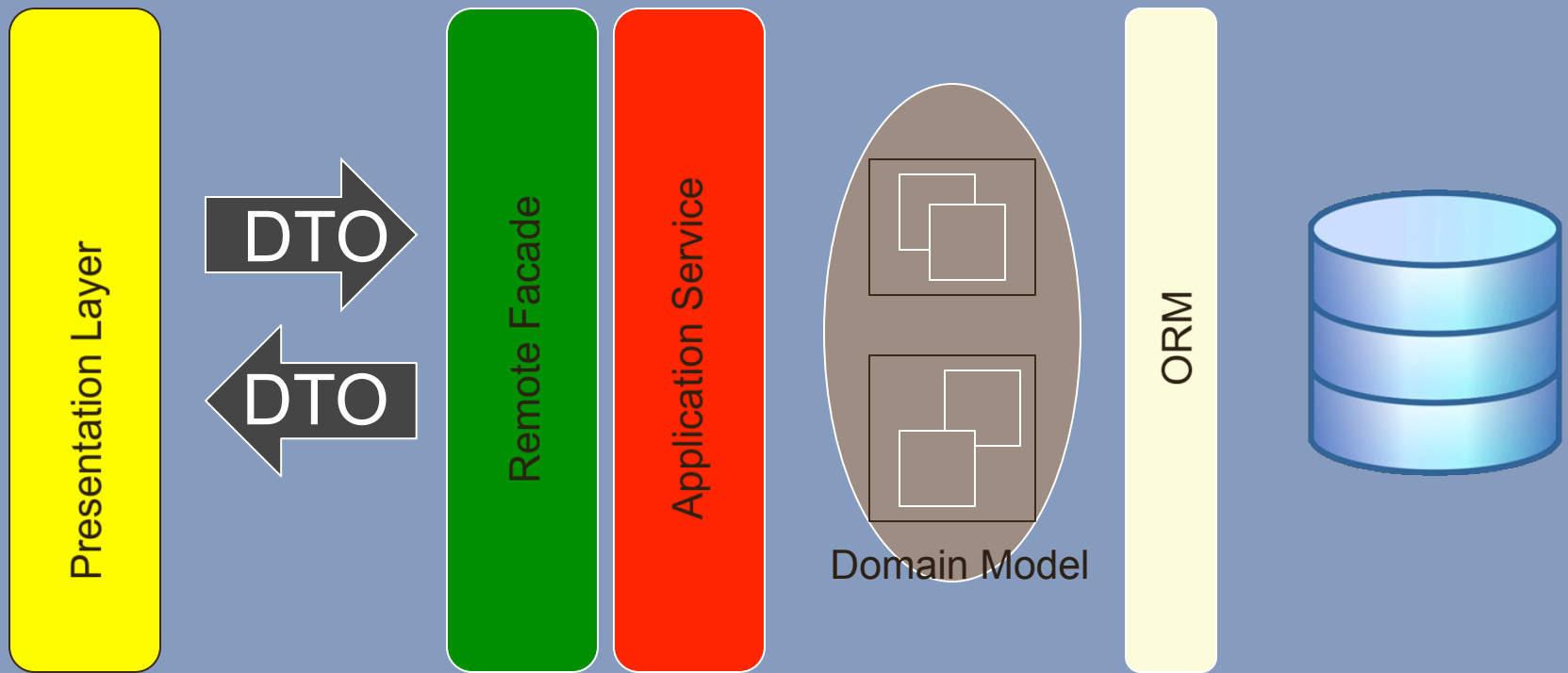
**BORING!!!!**

# Qual è il problema?

A single model cannot be appropriate for reporting, searching and transactional behavior

Greg Young, 2008

# ...ripartiamo da qui...

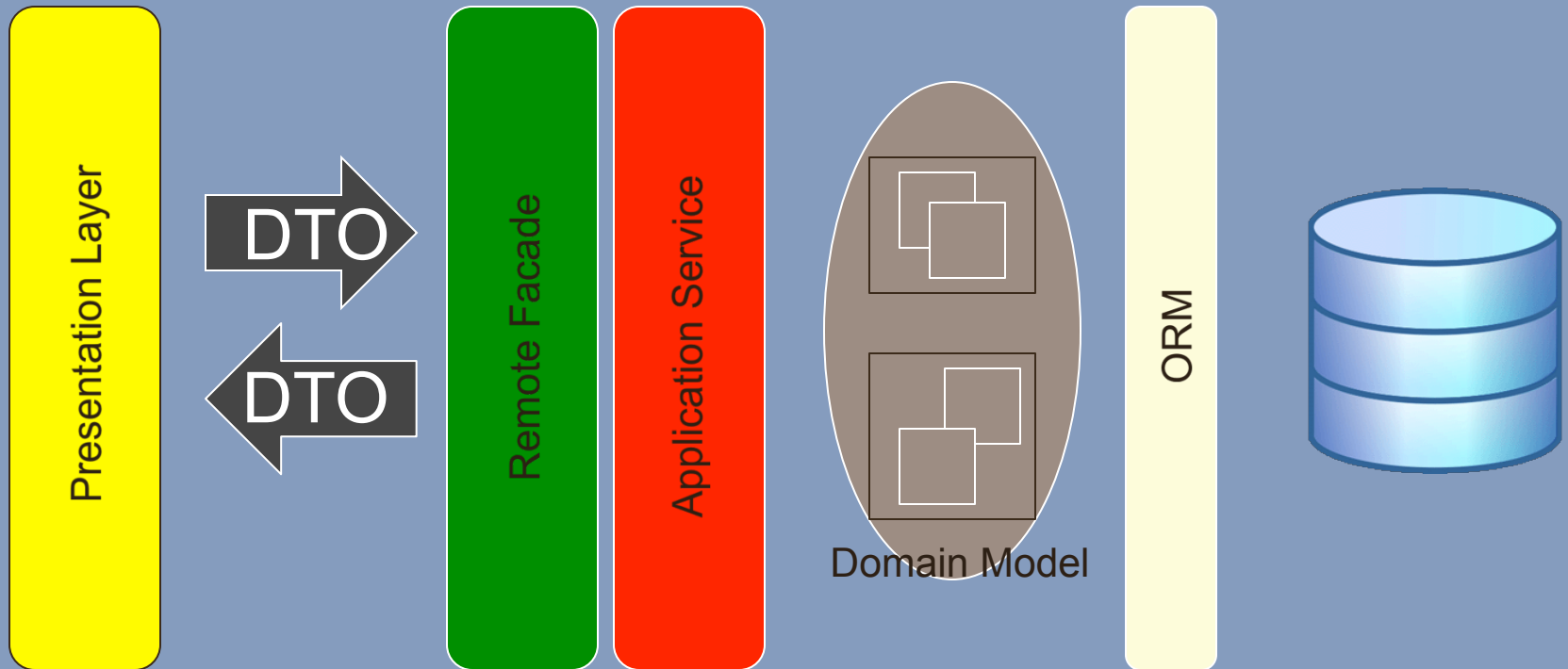


# ...ripartiamo da qui...

Presentation Layer

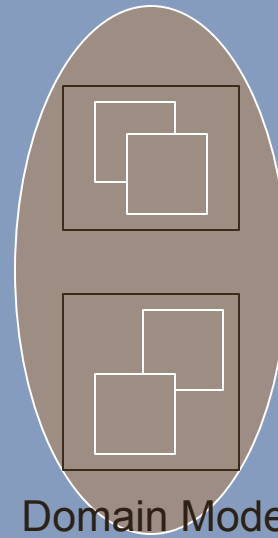
- C'è comportamento?
- Abbiamo necessariamente bisogno di un “modello”?
- Abbiamo bisogno di integrità referenziale e normalizzazione?
- Possiamo accontentarci di dati “quasi” istantaneamente aggiornati?

# ...ripartiamo da qui...



# ...ripartiamo da qui...

- Il domain model non nasce per “collezionare” o mostrare dati
- Il domain model deve sempre essere in uno stato consistente



Domain Model

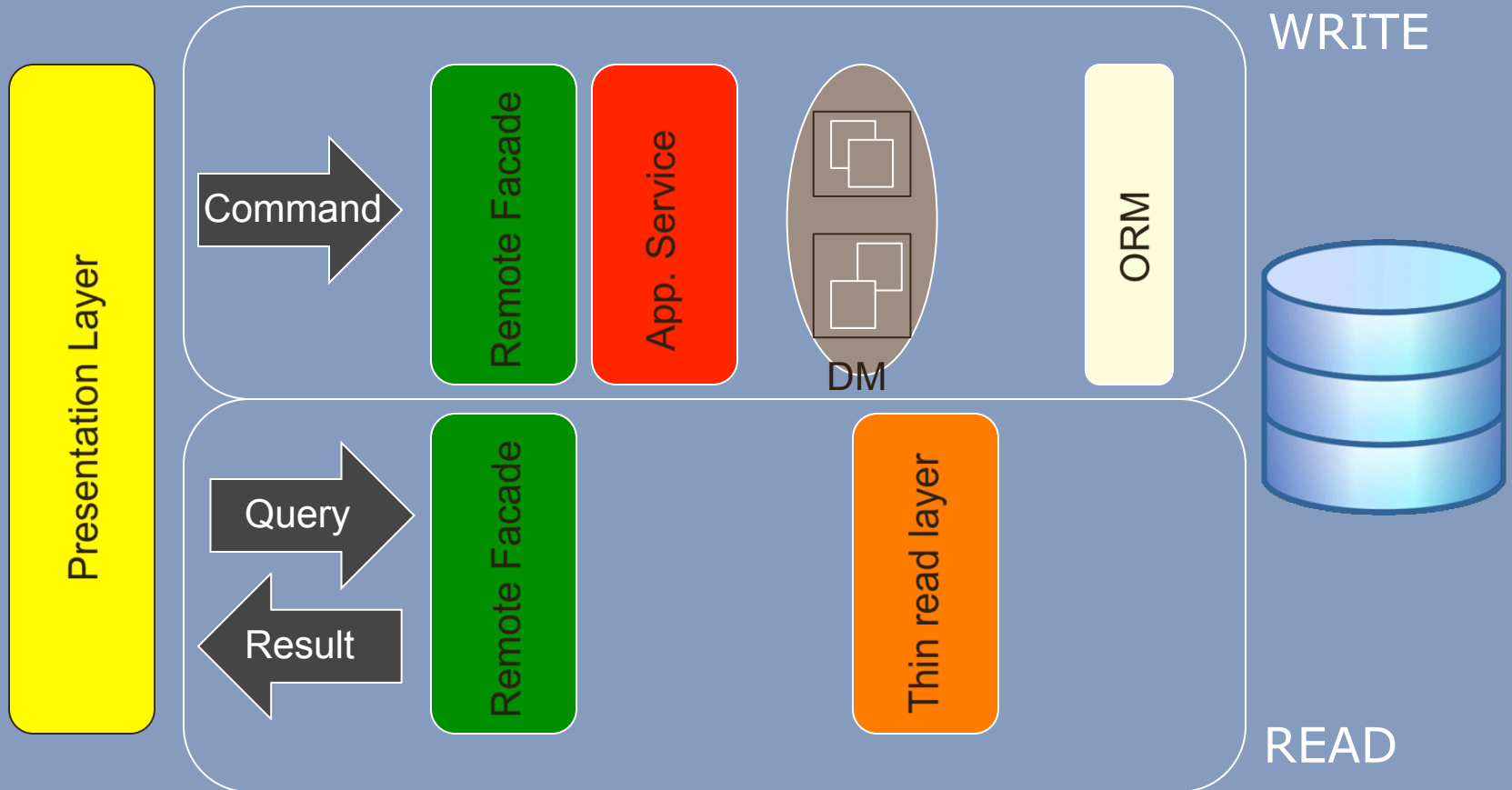


# Command query separation

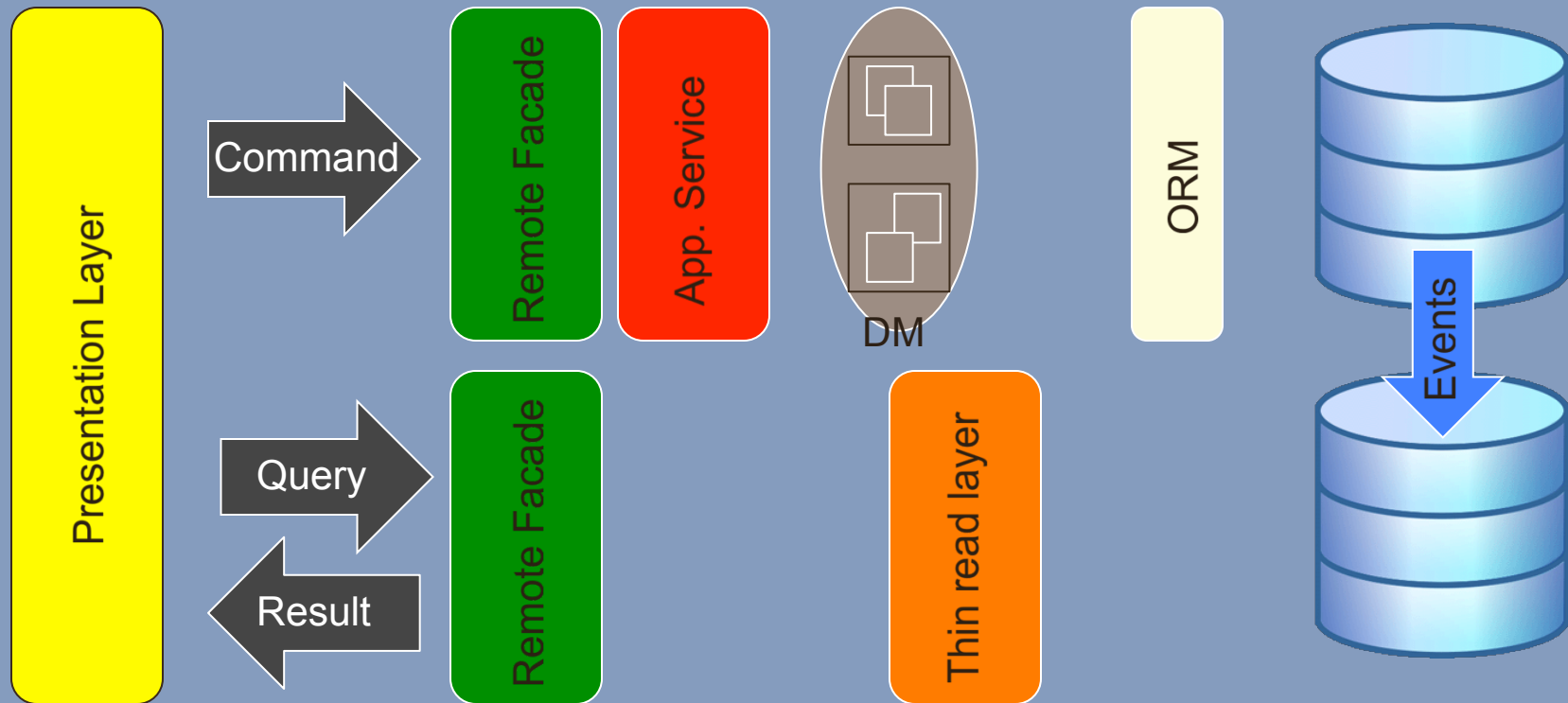
Every method should either be a command that performs an action, or a query that returns data to the caller, but NOT BOTH.

Bertrand Meyer

# CQRS



# CQRS



# CQRS

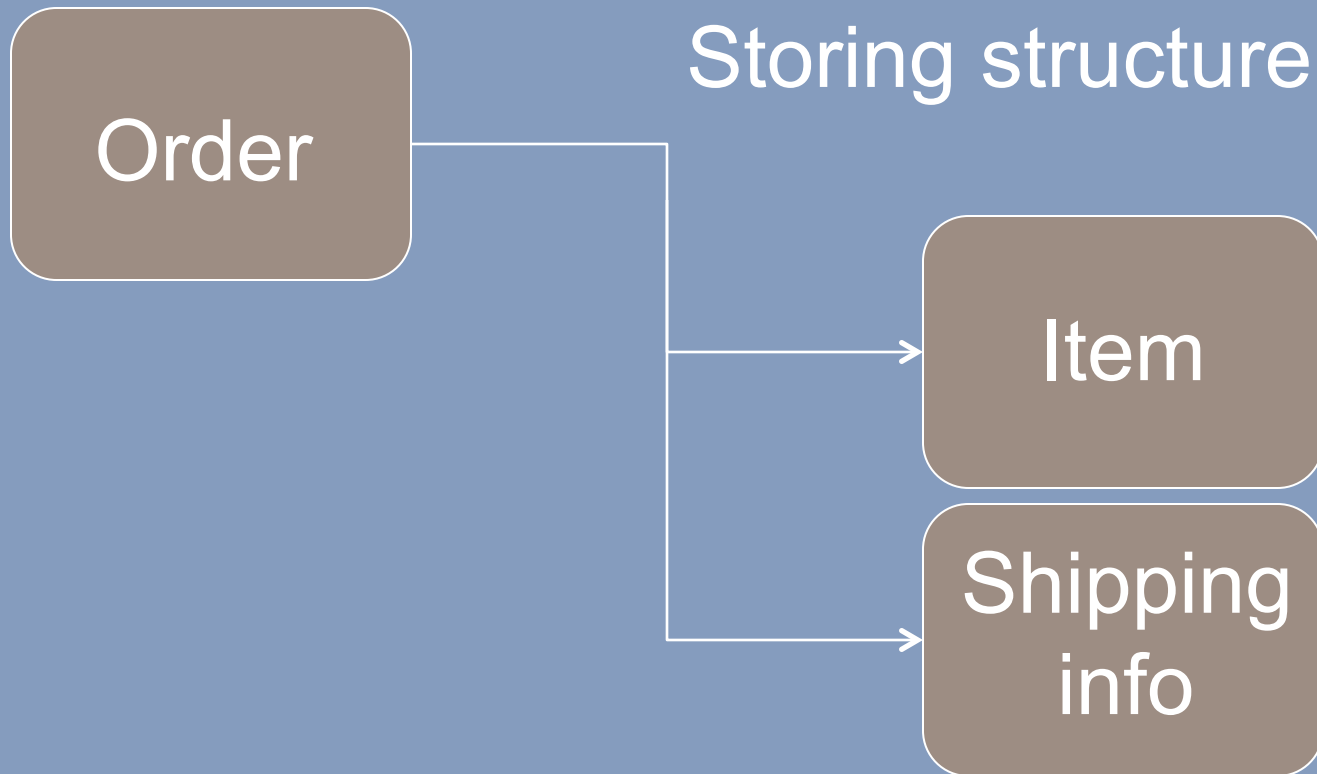
- Gli aggregate root ricevono Command e pubblicano eventi
- L'aggiornamento della base dati in lettura avviene tramite la gestione degli eventi
- Tutte le query impattano su una base dati “dedicata” e non coinvolgono il domain model
- Separazione delle competenze

# Event sourcing

State transition are an important part of our problem space and should be modeled within our domain

Greg Young, 2008

# Event sourcing



# Event sourcing

## Storing deltas



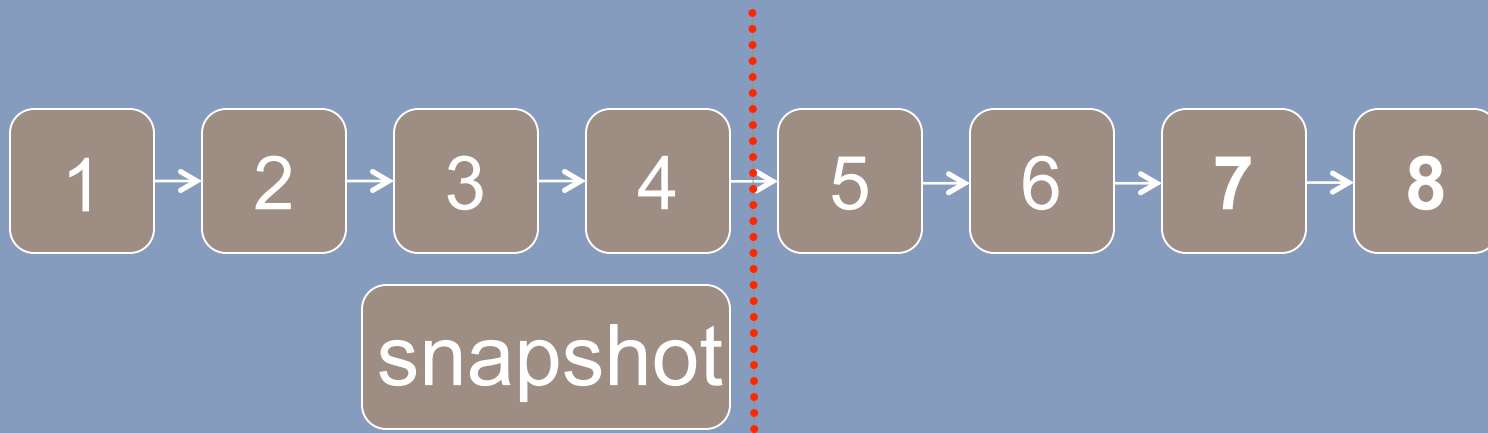
# Event sourcing

- Ogni cambiamento di stato è rappresentato da un evento
- Ogni evento è memorizzato in un EventLog / EventQueue
- E' possibile aggiungere "listener" in corso d'opera



# Event sourcing

- Lo stato corrente è “costruibile” (ri) eseguendo gli eventi
- I dati non sono persistiti in “strutture”, ma come serie di transazioni di stato



# Benefits

- No object-relational impedence mismatch
- Sistema “nativo” di auditing e tracking
- Possibilità di “rieseguire” la pipeline degli eventi
  - bug fix – ricostruzione scenari produzione
  - si possono aggiungere feature “retroattive”

# Quando CQRS fa per me?

## Chiedilo al “tuo” UBIQUITOUS LANGUAGE

# Quando CQRS fa per me?

## Tackling Complexity in the Heart of Software

# Contatti

[alessandro@codiceplastico.com](mailto:alessandro@codiceplastico.com)

<http://blogs.ugidotnet.org/amelchiori>

<http://twitter.com/amelchiori>

# Slides e materiale

Nei prossimi giorni su

<http://www.communitydays.it/>