Is Domain-Driven Design more than Entities and Repositories?

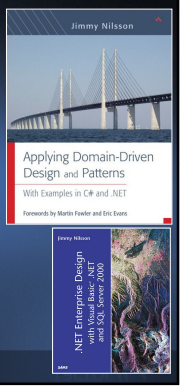
by Jimmy Nilsson

QCon

factor10

About Jimmy Nilsson

- Primarily a developer, but also a trainer and author
- Blogs at JimmyNilsson.com/blog/
- Author of *"Applying Domain-Driven Design and Patterns"* and *".NET Enterprise Design"*
- Co-founder and CEO of factor10



The situation for many developers and architects

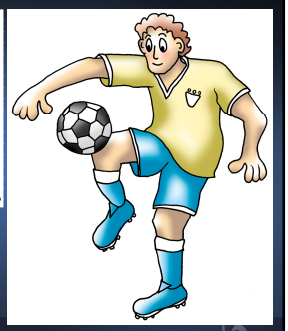
DB-driven and
Transaction script focused

factor10

First look: Same game, new name?

factor10

Closer look: Mess vs elegance!



First look: Nothing more than men running after a ball?

factor10

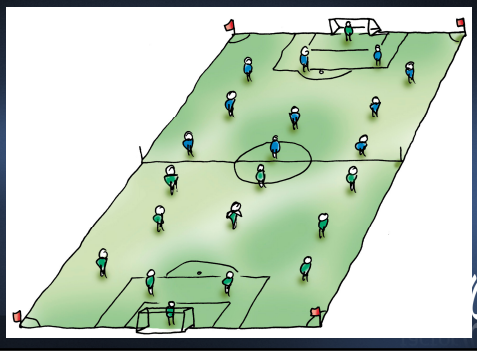
Closer look: Philosophy!

factor10

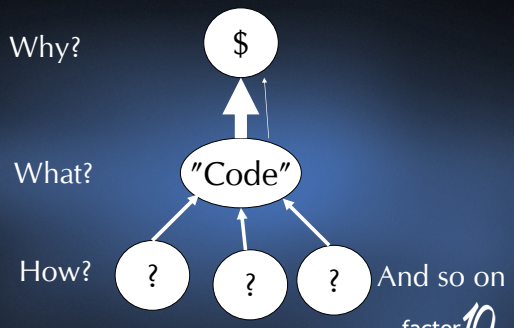
Closer look: Many more elements!

factor10

Closer look: Strategy!

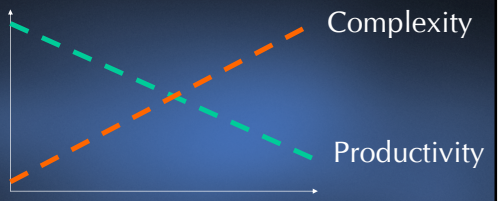


Let's take a step back...



factor10

Software economics



Question is, do we have essential or accidental complexity?

factor10

Why DDD?

Accidental complexity is bad

"Programming is understanding" (Kristen Nygaard)

"DDD = = OO done right"

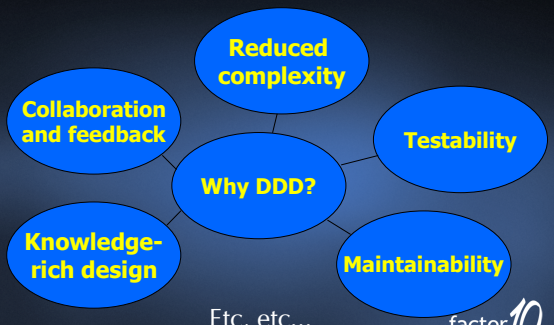
Context is king

Semantics over technology

...

factor10

Common effects of DDD?



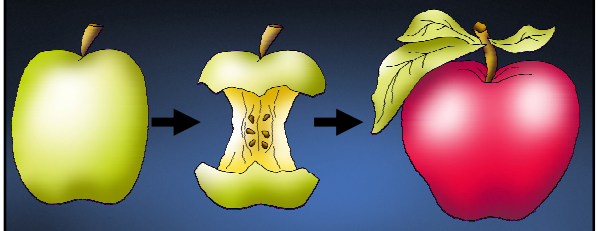
Etc, etc...

factor10

Closer look: Philosophy!

factor10

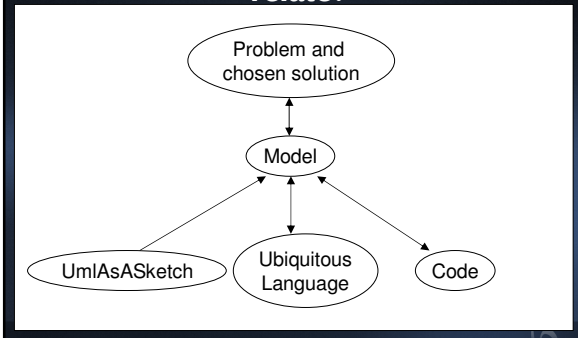
Main Focus of DDD?



- The core is the main focus!
- Forget distractions!

factor10

Model, code, etc... How do they relate?



Let's take simple example

"Customer has rentals"

factor10

Some typical T-SQL code

```
...
IF (SELECT COUNT(*) FROM RentalLines
INNER JOIN Products ON LineItems.ProductId =
Products.Id WHERE RentalId = inserted.Id AND
RequiredAge > @ageOfCustomer) > 0 BEGIN
    ROLLBACK
    RAISERROR...
    RETURN
END

UPDATE Rentals SET Status = 4 WHERE Id = inserted.Id
...
```

factor10

Another approach

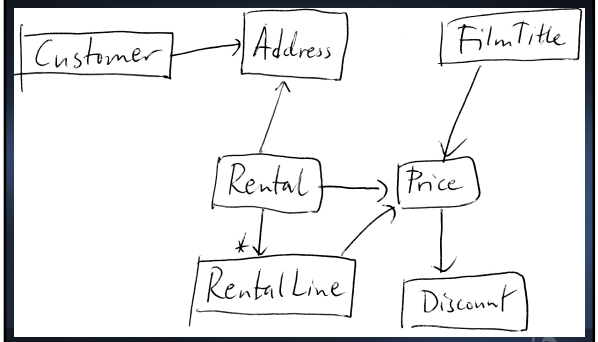
```
public bool CanCheckOut()
{
    if (_status != RentalStatus.New)
        return false;

    if (Customer.Age < _HighestRequiredAgeOfAnyOfTheChosenFilms())
        return false;

    return true;
}

public void CheckOut()
{
    if (! CanCheckOut())
        throw new...

    ...

    _status = RentalStatus.CheckedOut;
}
```

factor10

Closer look: Many more elements!

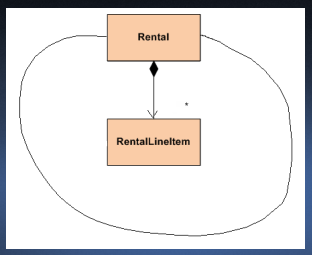
factor10

Value Objects



factor10

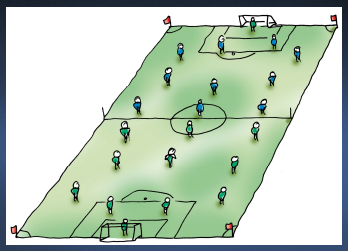
Some aggregate design thoughts



Has business meaning
Controlling dependencies coming in
Writes! Reads?
A *real* boundary
Controlling dependencies going out

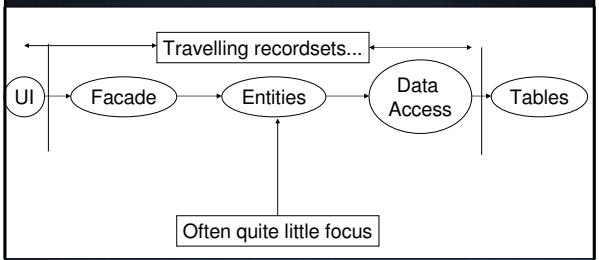
factor10

Closer look: Strategy!



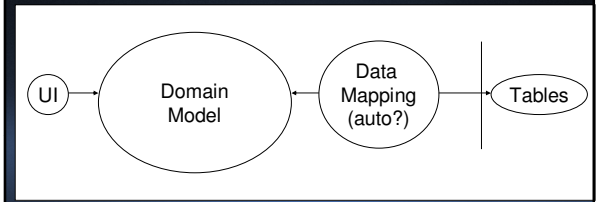
factor10

Classic layering (n-tier)



factor10

DDD basic layering



Strategic design [DDD]

A design language for design in the large

Bounded context

Set a boundary around a sub-model
Be explicit regarding the context

Example:
- Video rental
- Purchase

A style I often find good

- Avoid Enterprise Domain Model
- Avoid Integration DB
- Avoid huge teams
- Avoid unbalanced focus on loose coupling
- Avoid enforcing one ”internal” style

Nothing really new, more like an observation
A name? CCC? :-)

Isn’t it time to swap mess for elegance?

I think so.

References

[DDD] Eric Evans: Domain-Driven Design

[ADDDP] Jimmy Nilsson: Applying Domain-Driven Design and Patterns