

# Object Orientation, Domain Driven Design, and the Honour of the Programmers' Guild

---

Dan Bergh Johnsson  
Partner and Spokesperson  
Omegapoint AB, Sweden





# <soapbox>

---

Ingredients: thoughts(60%), ideas(25%),  
code(16%), hand-waving(10%),  
morals(4%), history(3%)

Warning: this presentation may contain  
traces of philosophy



# Abstract / Conclusions / Agenda

---

What has Happened to Object Orientation?  
Object Orientation Requires Refactoring

Is Domain Driven Design any Good?  
Domain Driven Design is Good

Do we have Moral Responsibility?  
We Have Moral Responsibility

Is there Hope?  
There is Hope



# Abstract / Conclusions / Agenda

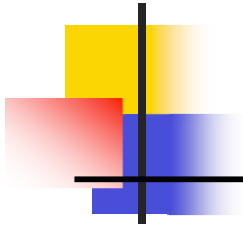
---

## What has Happened to Object Orientation?

Is Domain Driven Design any Good?

Do we have Moral Responsibility?

Is there Hope?



# Programs as Simulations



# All programs are simulations

---

- Simulate reality
  - Aerodynamic flow
  - Diffusion processes
  - Heat conduction
- Simulate what would have happened
  - System is big black box
  - Input/output as in reality
  - Model of reality living in the box



# Standard example: Bank

---

- Yore:
  - Clerk wrote in leather-bound ledger
- Now:
  - In/out registered in banking system

# Simulation – Maid Milking Cow

---

```
class Maid {  
    void milk(Cow c, Bucket b) {  
        c.decreaseMilk(2);  
        b.fill(2);  
        c.feelAtEase();  
    }  
}
```

```
class Cow {  
    void decreaseMilk(...) {  
        ...  
    }  
    void feelAtEase() {  
        ...  
    }  
}
```



# Cows Milk Themselves!

---

```
class Maid {  
    void milk(Cow c, Bucket b) {  
        c.milk(2);  
        b.fill(2);  
    }  
}
```

```
class Cow {  
    void milk(int amt) {  
        decreaseMilk(amt);  
        feelAtEase();  
    }  
    void decreaseMilk(...) { ... }  
    void feelAtEase() { ... }  
}
```

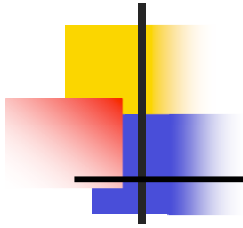
## Simulation?



# Simulations in Toon-World

---

- “Inanimate” objects
- Stoves that cook
- Tables that set themselves
- Shopping carts keeping track of sum



# Code & Meaning





# The Meaning of Code

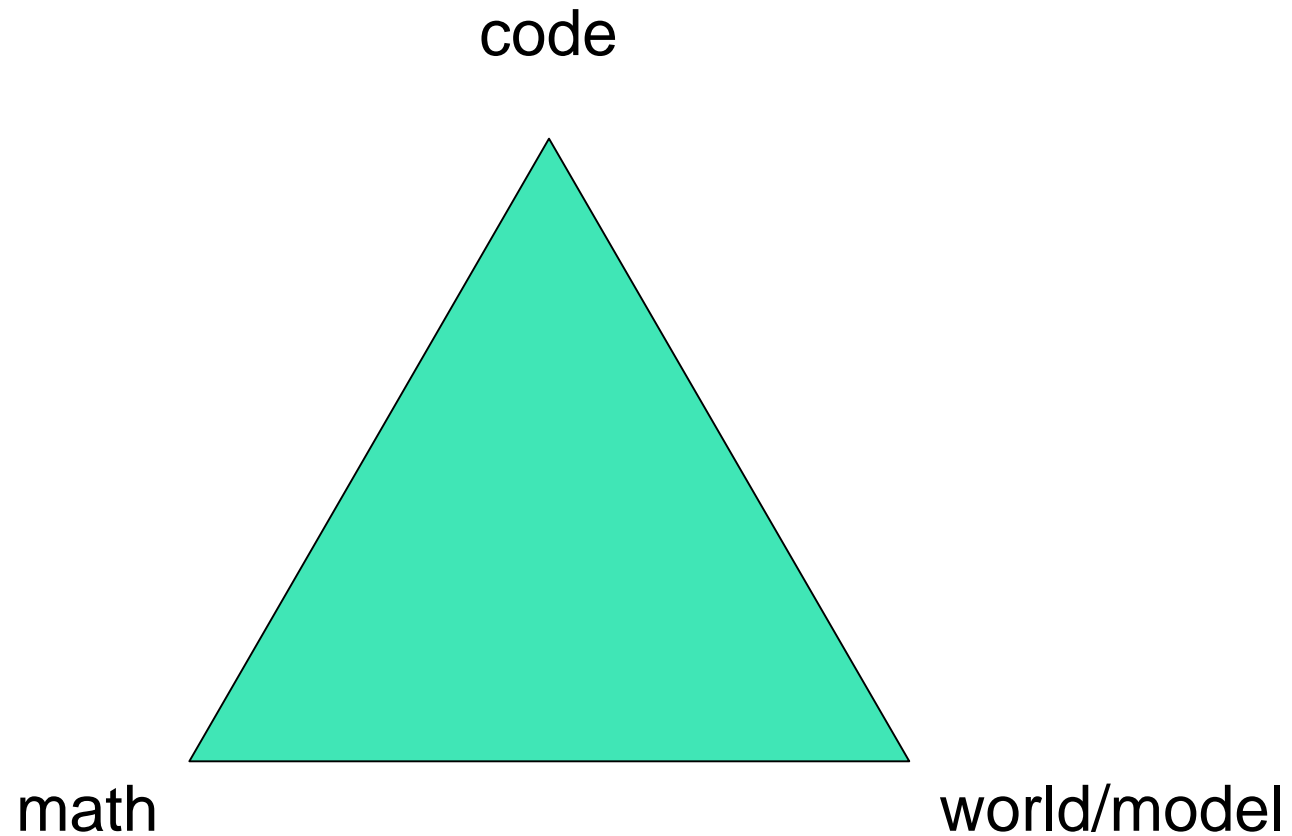
---

- `f(x) return x*x;`
- 0: `dload_1`  
1: `dload_1`  
2: `dmul`  
3: `dreturn`
- $f : (x) \rightarrow x^2$  in  $R \rightarrow R$
- amount of energy in two-kilo mass at given speed



# Triangle of Meaning

---





# Abstraction = simple

---

- “Abstrakt” part of organ
- Abstraction
  - Select relevant properties
  - Discard irrelevant properties
  - Simplification
- Mathematical model of complicated reality
  - How to chose our abstractions?



# Code that: Do vs Mean

---

- Code that *do* something

```
List<Trans> translist
translist.add( ... )
for ( ... : translist)
    sum+=
```

- Code that *mean* something

```
Ledger transledger
transledger.add(...)
transledger.getTotal()
```

```
class Ledger
    List translist
    ...
```



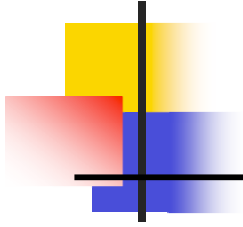
# Limited value of Collection<E>

---

```
interface AccountingService {  
    List<Transaction> monthlyCompilation(Month month);  
}  
for(Transaction t : accServ.monthlyCompilation(JUNE)) {  
    ... // compute balance-compensated risk level  
}
```

## VS

```
interface AccountingService {  
    Ledger monthlyCompilation(Month month);  
}  
Ledger juneLedger = accServ.monthlyCompilation(JUNE);  
... = juneLedger.balanceCompensatedRiskLevel();
```



# OO Failure Debrief



# Object Orientation Promise

---

- REUSE
- Fulfilled in everyday life?
- Why this promise?

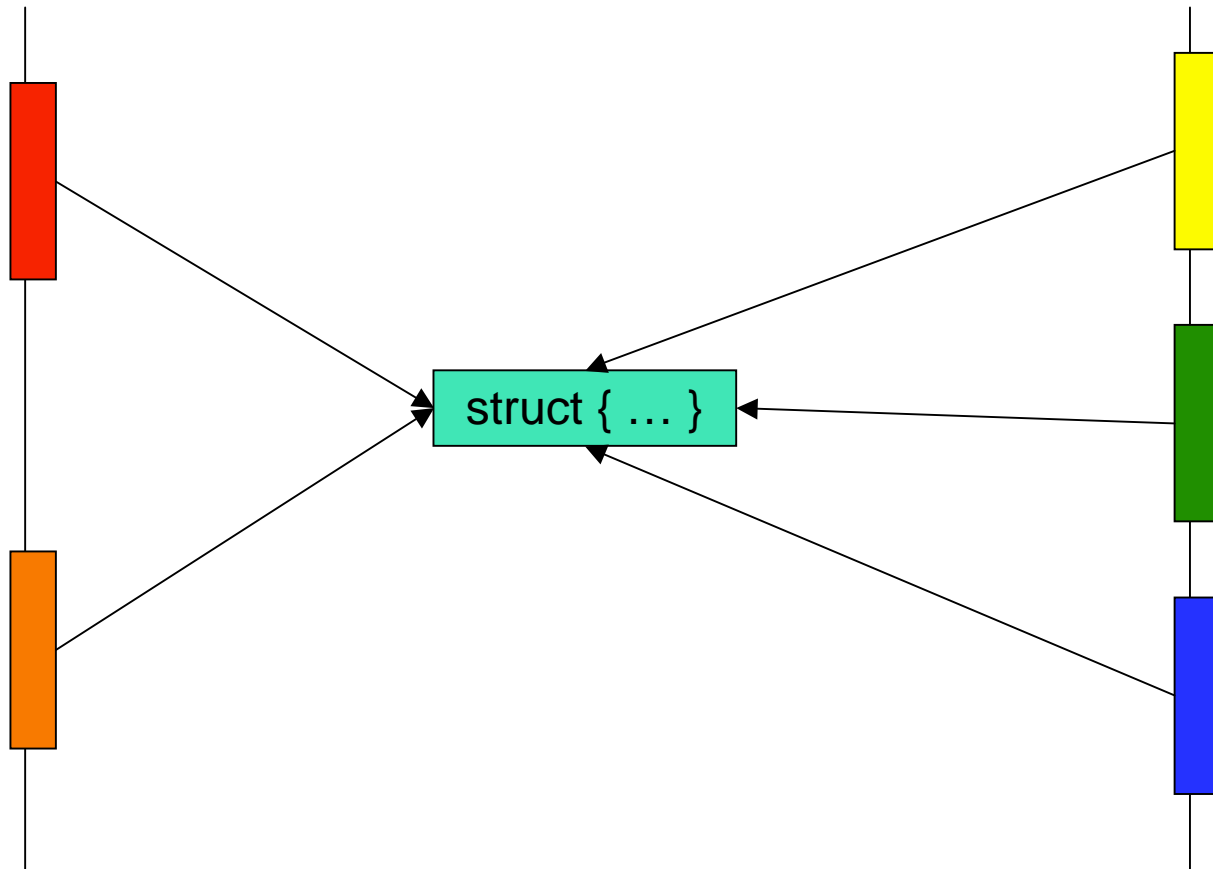


# Is OO New?

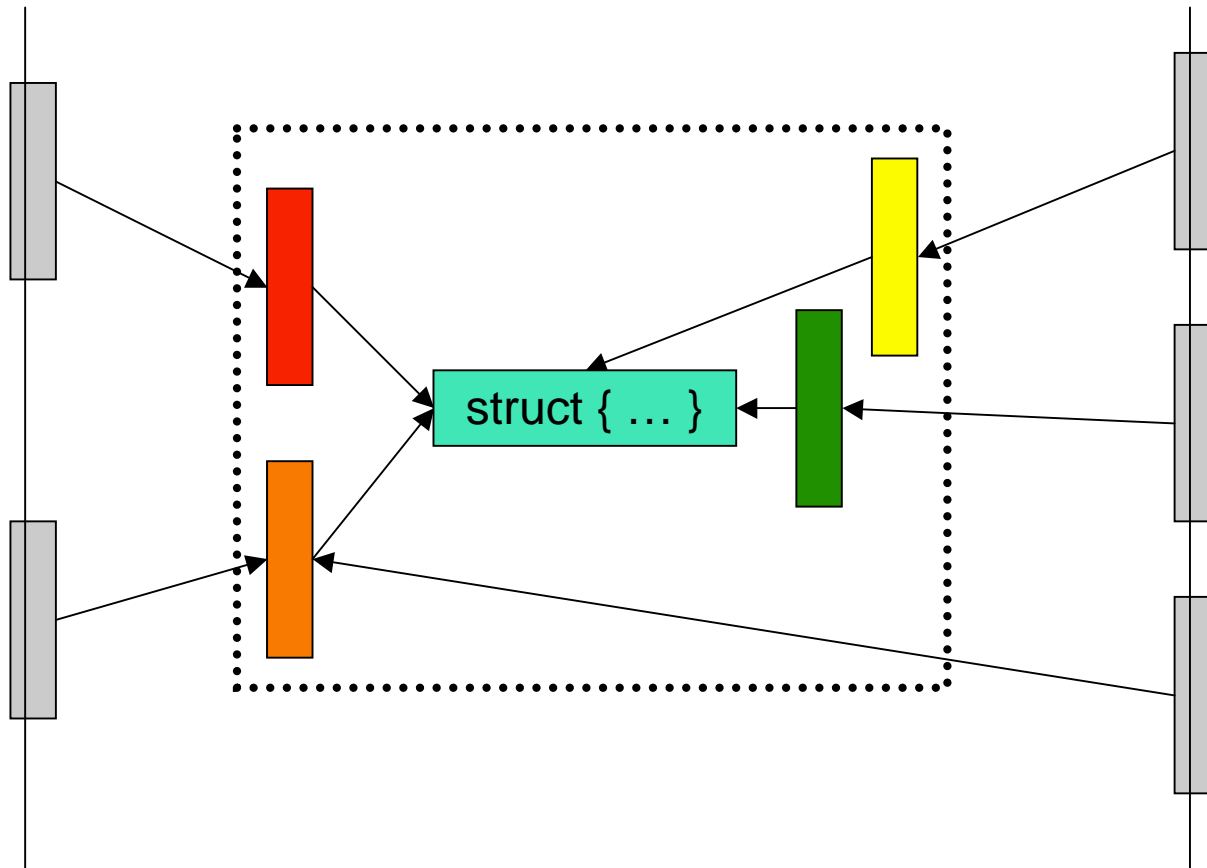
---

- 30 years
- Yet older ideas
- Should be well understood
- Should be well practiced
- Still not fulfilled promise

# OO is Old Stuff



# OO is Old Stuff





# OO is Completely New

---

- Thomas of Aquino
- Reformed theology
- God – the puppet master
  - God guides the fox on hunt of the rabbit
- God – the clockmaker
  - God embeds the will of hunting into the fox
  
- Object Orientation is new way of thinking



# Java is not Object Oriented

---

- Object Orientation property of Application
  
- How close is behaviour to data?



# Encapsulate Interpretation

---

You reuse what you  
encapsulate

Encapsulating data is  
uninteresting

Encapsulating  
interpretation is  
crucial

Get/set considered  
harmful

```
acc.setBal(acc.getBal() - amt)
```

```
acc.bal = acc.bal - amt
```

```
void decBal(amt)  
    bal = bal - amt
```

```
void decBal(amt)  
    check(amt <= bal)  
    bal = bal - amt
```



# Encapsulate data vs interpretation

---

```
class Order {  
    private String ordernr;  
    String getOrdernr(){  
        return ordernr;  
    }  
    void setOrdernr(String ordernr) {  
        this.ordernr = ordernr;  
    }  
}
```

```
Order order = ...  
order.setOrdernr(fromGui.getText());
```

```
class Order {  
    public Ordernr ordernr;  
}
```

```
class Ordernr {  
    private String nr;  
    Ordernr(String nr) {  
        check(nr);  
        this.nr = nr;  
    }  
}
```

```
Order order = ...  
order.ordernr = new  
    Ordernu(fromGui.getText());
```



# Encapsulation

---

- Encapsulated interpretation enforce consistency
- Consistency wrt set of interpretation rules
- Model = defined abstractions = interpretation rules



# How to get to Toon-World?

---

- Perfect design before coding
- No code added unless design revised
- Perfect knowledge of the code base
- ...
- Yeah!



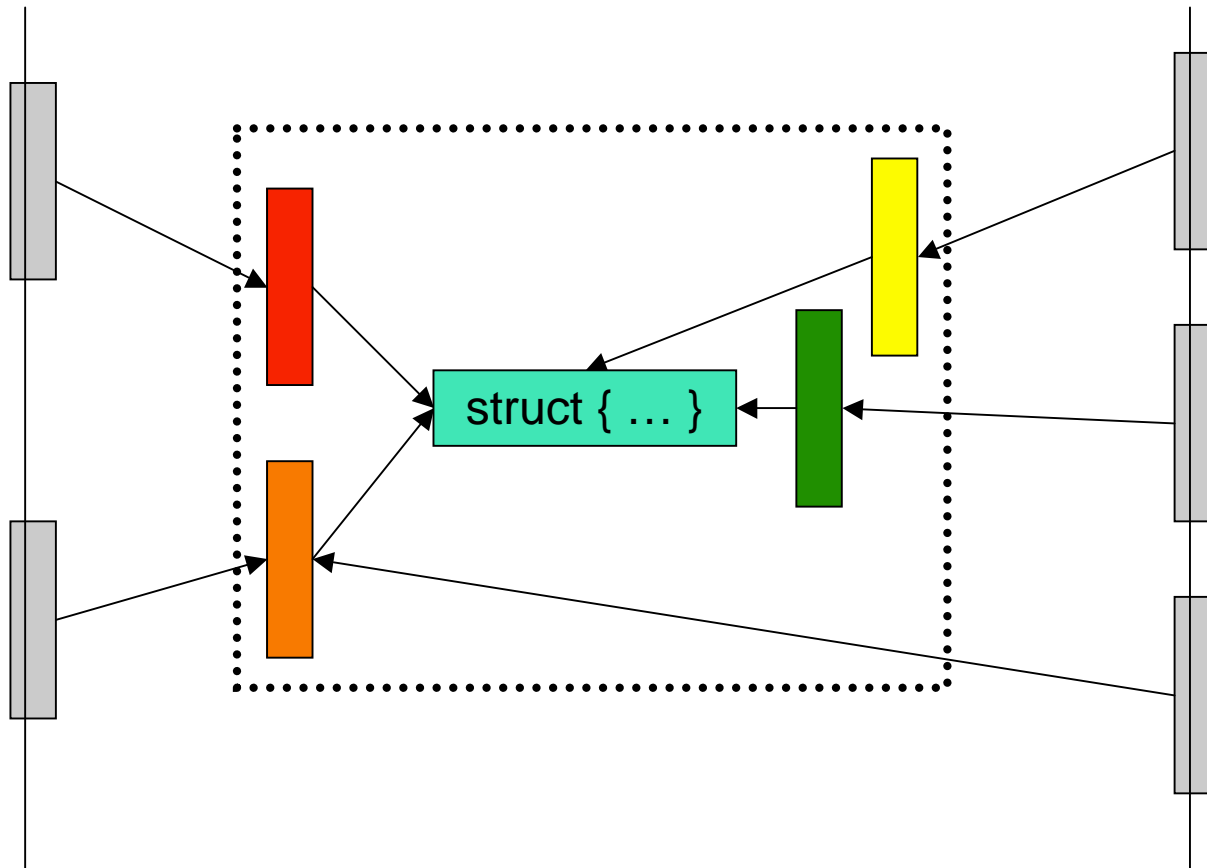
# Time Factor

---

- All the code not written at same occasion
- Incremental development
- Development / maintenance
- Large code
- New code in right place?
- Cannot trust all the people all the time

# Data is Gravity for Behaviour

## – Design “Falls” into Place





# Two Waves of Development

---

- Development wave front
- Consolidation wave front
  
- Too far apart – technical debt
- Too close – analysis paralysis



# Role of Refactoring

---

- Refactoring essential to Object Orientation
- Essential =
  - Necessary condition
  - Cannot do without
  - Meaningless if missing
  - ...



# Abstract / Conclusions / Agenda

---

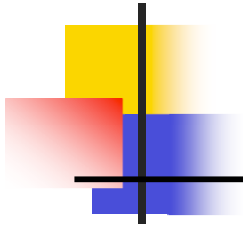
What has Happened to Object Orientation?

*Object Orientation Requires Refactoring*

**Is Domain Driven Design any Good?**

Do we have Moral Responsibility?

Is there Hope?



# Domain Driven Design



# Domain Driven Design

---

- OO tells us to **use objects**
- DDD tells us **which objects to use**
  
- Basic idea: structure the code after the domain
  - Domain examples: patient journals, packet transportation
  - **Solution expressed in the terminology of the problem**
- Realisation: classes for all concepts



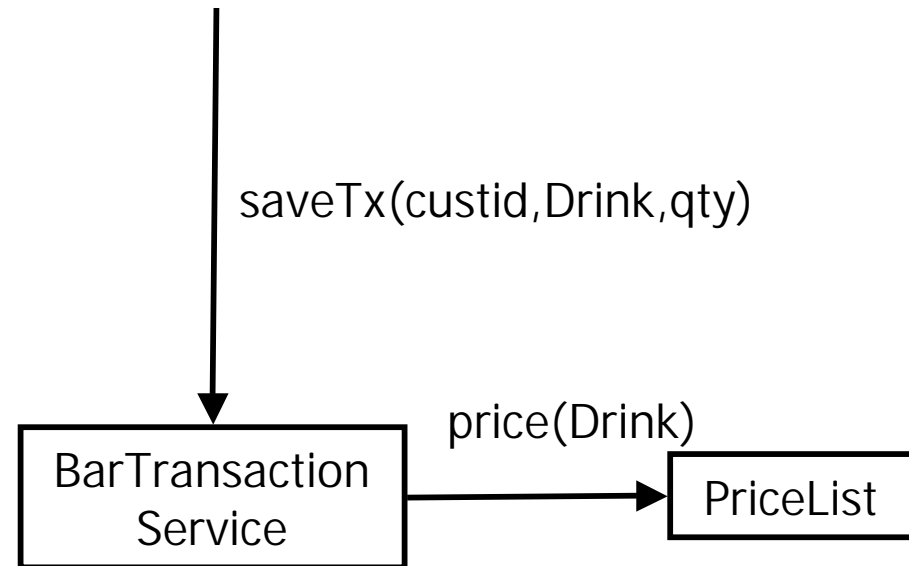
## Example: Night at a Bar, functions

---

- open a tab
- order drinks
- drink price calculation
- credit / credit check
- paying tab
  
- one tab at a time

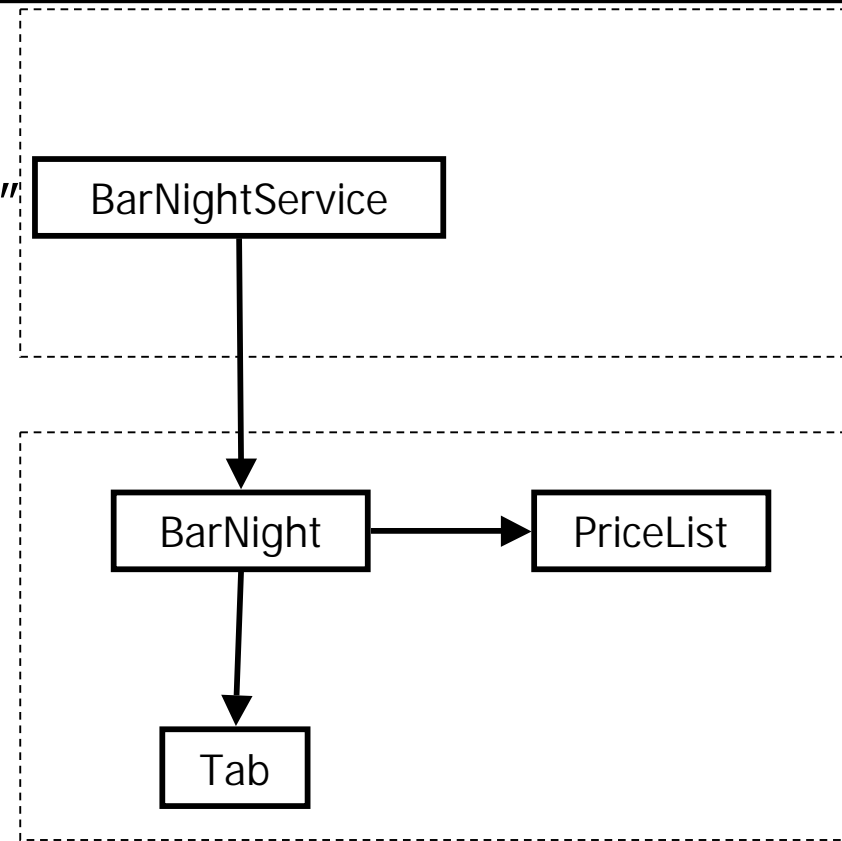
# Non-DDD solution

- Component for saving “bar transactions”
  1. Loads record (db/cache)
  2. Make computations
  3. Checks conditions
  4. Updates record
  5. Saves record
- Domain insight
- *Implicit* encoding



# DDD solution

- Component representing “bar night”
  - Holds object graph
  - Delegated state
  - Delegated computation/checks
- Domain insight
- **Explicit** encoding





# DDD advantages

---

- Explicit and shared understanding
  - “Glossary on steroids”
  - Ubiquitous Language
  - Problem Domain API
  - Solution formulated in problem terminology
- High Testability
- Easy to Extend



# DDD – when and where

---

- Focus on the (enterprise) problem to solve
- Good idea **if**:

Understanding the problem is the **critical**  
complexity

- Counterexamples:
  - Network router (I/O performance)
  - Mobile phone / embedded sensor (memory footprint, power consumption)



# Exercise: DDD good idea?

---

- User registration at web site
- Transport company collects/delivers packages
- Optimising compiler



# DDD is just OO?

---

- Modelling beyond “vanilla OO”
- Model as central communication tool
- Models judged by usability



# Deep Modelling

---

- LOTS of focus on investigating domain
- Constant remodelling and refactorings unveils “underlying contours”



## Deep Modelling and “Business Rules”

---

- V-deals
- H-deals
- “Good” Month
- D-deals
- H-deals
- New def of “good”
- New legislation
  - V must be matched with H
- Behaviour explained *within* model
- Business Rule



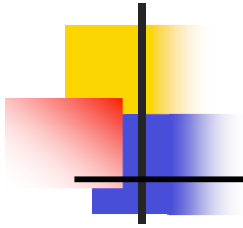
# What is DDD about?

---

- Tips and tricks about how to structure a program

or

- How we think when programming



# Design as Negotiation

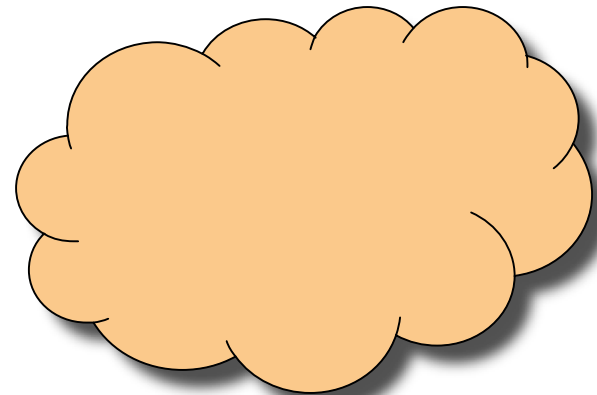
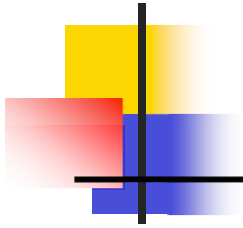


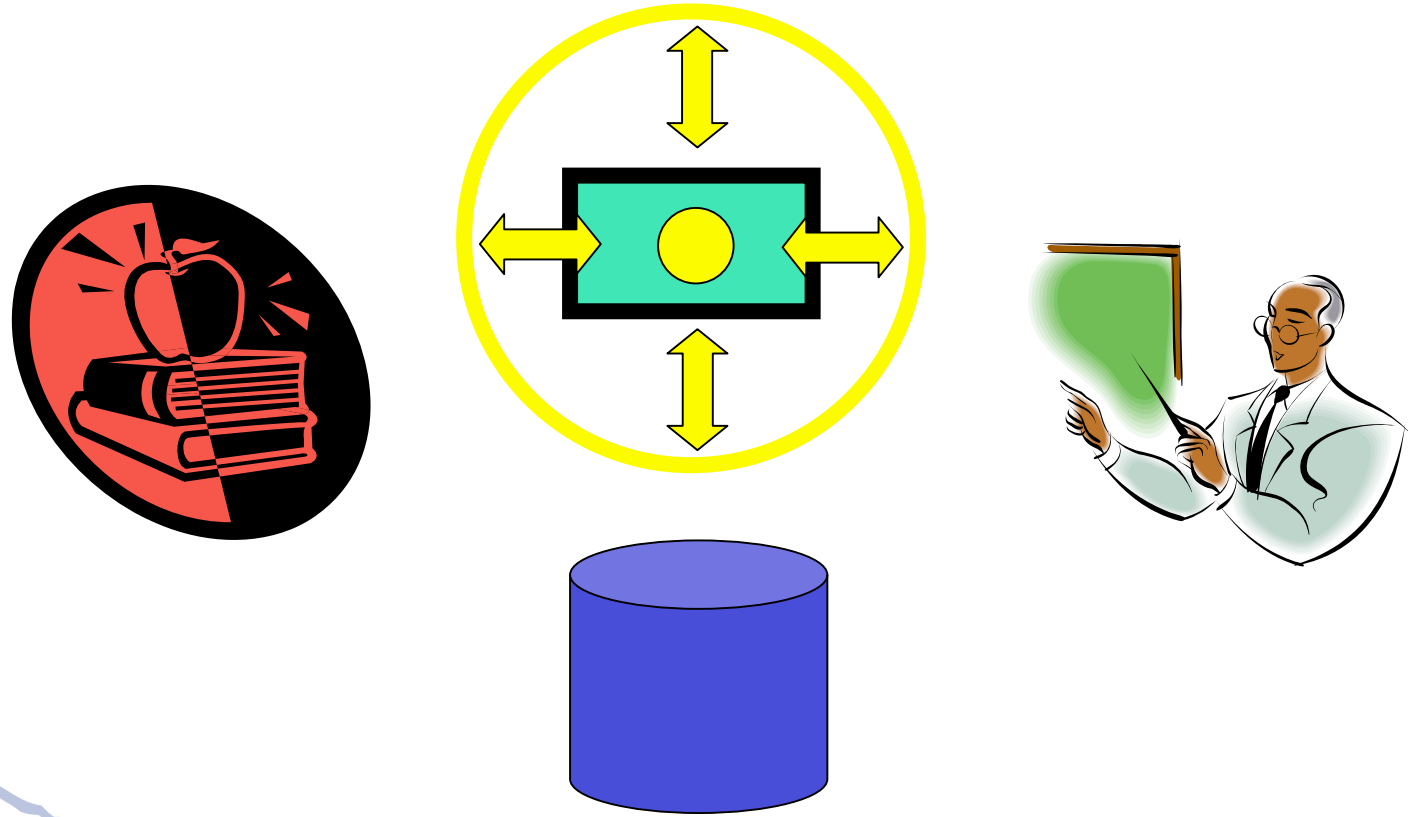
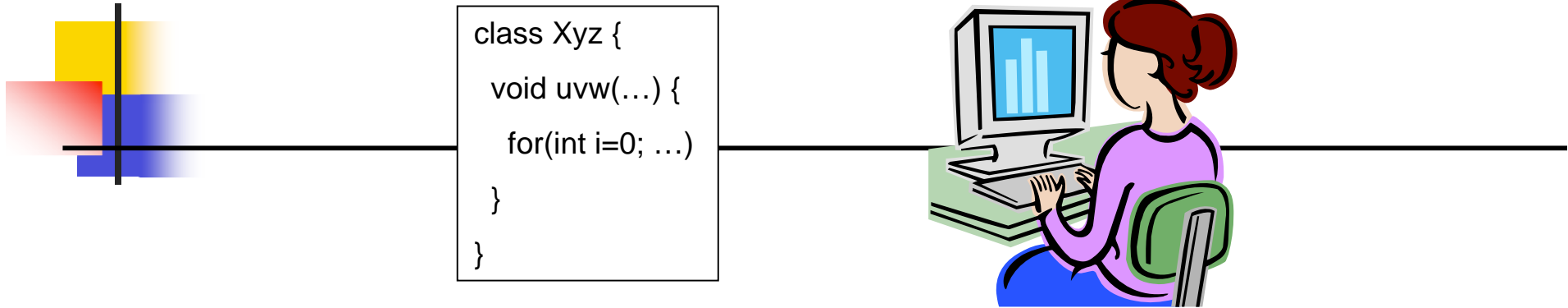
# Customer

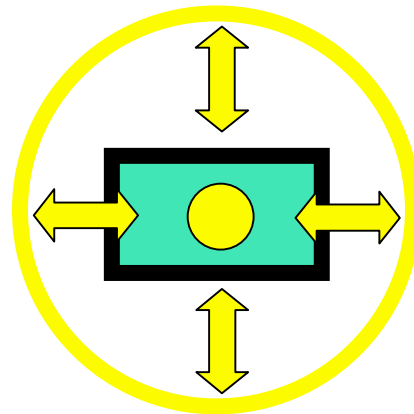
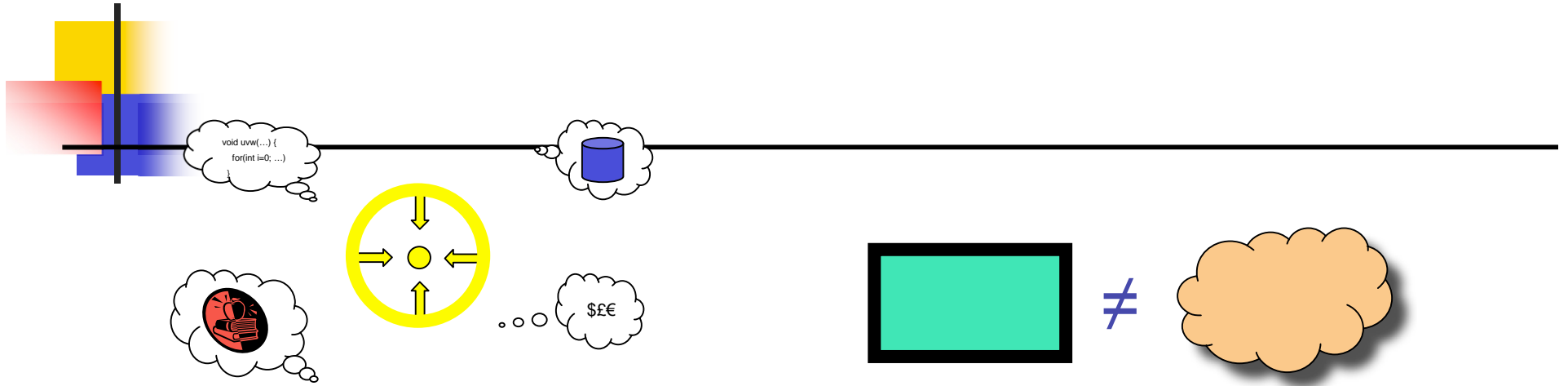
---

- Customer is always right
- Give the customer what they want











# Negotiation Crash Course

---

- Harvard Negotiation Project
- Position based negotiations
  - *aka* "Haggling"
- Interest based negotiations
  - Identify Interests
  - Invent Options for Mutual Gain
  - Use Objective Criteria



# Model is Negotiation Result

---

- Joint Project
- Purpose: address all interests
- Common terminology/language



# DDD Literature

---

- “Domain Driven Design”, Eric Evans
- “Applying DDD and Patterns”, Jimmy Nilsson
- “Domain Driven Design Quickly”, InfoQ
  
- [domaindrivendesign.org](http://domaindrivendesign.org)



# Abstract / Conclusions / Agenda

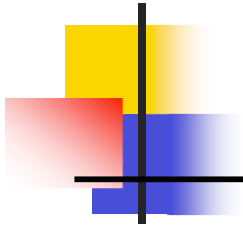
---

What has Happened to Object Orientation?  
Object Orientation Requires Refactoring

Is Domain Driven Design any Good?  
*Domain Driven Design is Good*

**Do we have Moral Responsibility?**

Is there Hope?



# Programming and Programmers in the Great Line of History





# Programmer = sorcerer

---

- Tools make us stronger
  - Rock on stick = strong arm
- Mechanising dull work
  - Machines reduce physical routine work
- Sorcerer's dream
  - Reduce intellectual routine work
  - Conjure with magical spells
- Power was in silicon stones



# Programming is Purely Intellectual

---

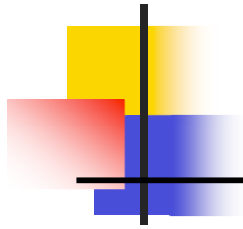
- when we have understood we are finished – only encoding remains
- “understanding the business”
- formulating our ideas unambiguous
- programming as a pedagogical trait



# Software Ideas and Code Units

---

- Valid format of “order number” is ...
- Book-keeping transactions must be balanced
- Cows feel at ease when milked
- class Uvw
- voix xyz(...)
- if (...)
- for(...)
- ... = ...
- Unit test document the capture of a software idea in a code unit



# Programmer = craftsman

---

- Craftsmanship, with stains of science and art
- We build machines
  - mechanising intellectual routine work



# Tricks of the Craft

---

Large level – slow heartbeat

- Architecture
- Processes

Small level – fast heartbeat

- Test Driven Development
  - red/green/refactor
  - psychology of testing
- Knowing Next Commit
- Speculative and Productive Programming

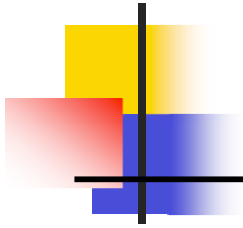


# Craftsmanship Honour

---

## Craftsmen

- proud, honourable chaps?
- cheating wranglers?



# Moral Responsibility



# My PL won't let me Refactor

---

- Argue
  - Subvert
  - Quit
- 
- ABAQUS (HKS Inc) was created when Dave Hibbit was not allowed to refactor MARC



# It is not my fault

---

- Do we believe in what we say?
- Do we do what we say we believe in
  
- “I just deliver what has been ordered”
- “I must do what I am told”
- “I can get fired”
  
- Nürnberg on soldiers and officers



# Abstract / Conclusions / Agenda

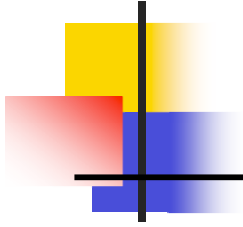
---

What has Happened to Object Orientation?  
Object Orientation Requires Refactoring

Is Domain Driven Design any Good?  
Domain Driven Design is Good

Do we have Moral Responsibility?  
*We Have Moral Responsibility*

**Is there Hope?**



# Craftsmanship

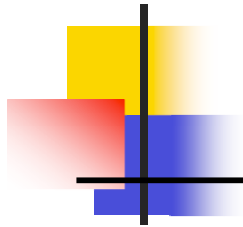




# Craftsmanship Honour

---

Do we laugh at Uppsala Cathedral?  
Will future laugh at our systems?



# Stone cutters' guild

---

We, who cut mere stone,  
must always envision cathedrals



# Abstract / Conclusions / Agenda

---

What has Happened to Object Orientation?  
Object Orientation Requires Refactoring

Is Domain Driven Design any Good?  
Domain Driven Design is Good

Do we have Moral Responsibility?  
We Have Moral Responsibility

Is there Hope?  
*There is Hope*





</ Object Orientation,  
Domain Driven Design,  
and the Honour of the Programmers' Guild >

---

Thanks for your attention  
afterthoughts:

- [dan.bergh.johnsson@omegapoint.se](mailto:dan.bergh.johnsson@omegapoint.se)
- [dearjunior.blogspot.com](http://dearjunior.blogspot.com)
- [www.omegapoint.se](http://www.omegapoint.se)

