This is how developers might punish your database

a.k.a.

# Twisted SQL Server use cases

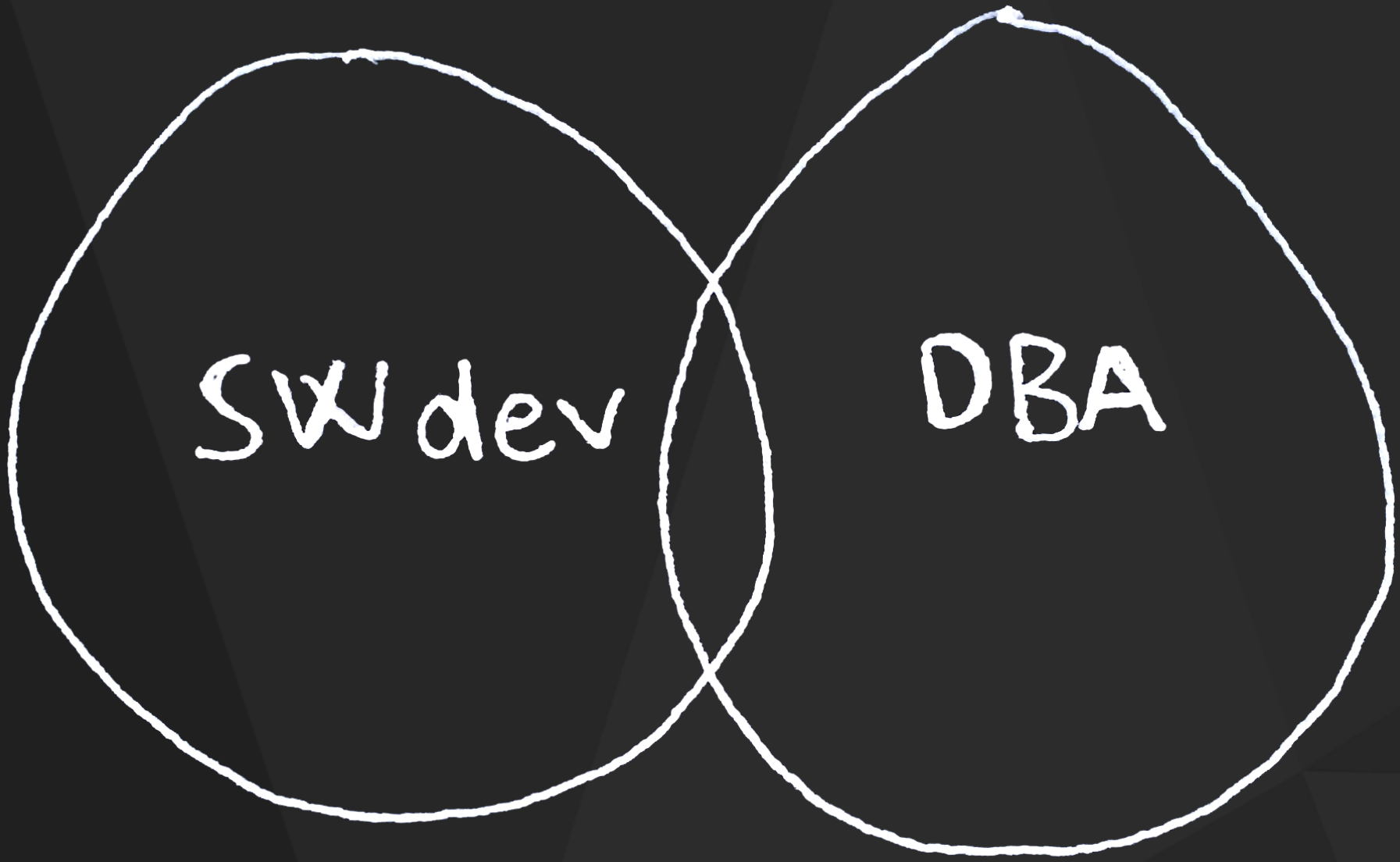
a.k.a.

# SQL Server All The Things


Mogens Heller Grabe

Technical Evangelist @ d60

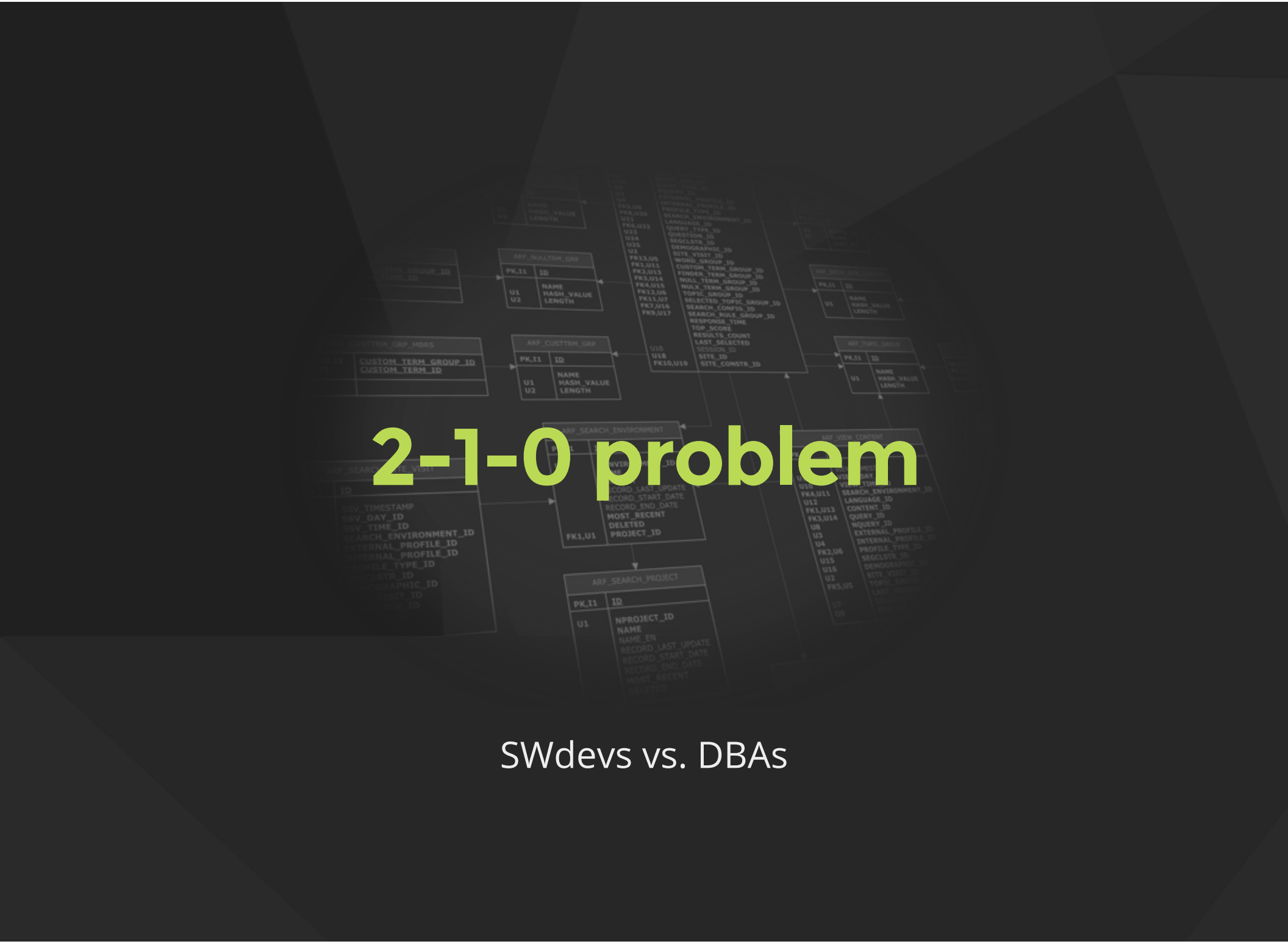
mhg@d60.dk
@mookid8000
http://mookid.dk/oncode/


SW dev
DBA

# Agenda

- 2 worlds - 1 database - 0 problem?
- Object-oriented models
- Messaging
- Event-centric models
- Conclusion

...there will be C#...


2-1-0 problem
SWdevs vs. DBAs

# SWdevs

*tend to value* stuff around their applications, e.g.

- technology that fits the varied types of tasks they're given
- new ways of doing things

so that they can solve all kinds of problems...

# DBAs

*tend to value* stuff around their database, e.g.

- operational simplicity
- moderation in data size and access patterns
- predictability

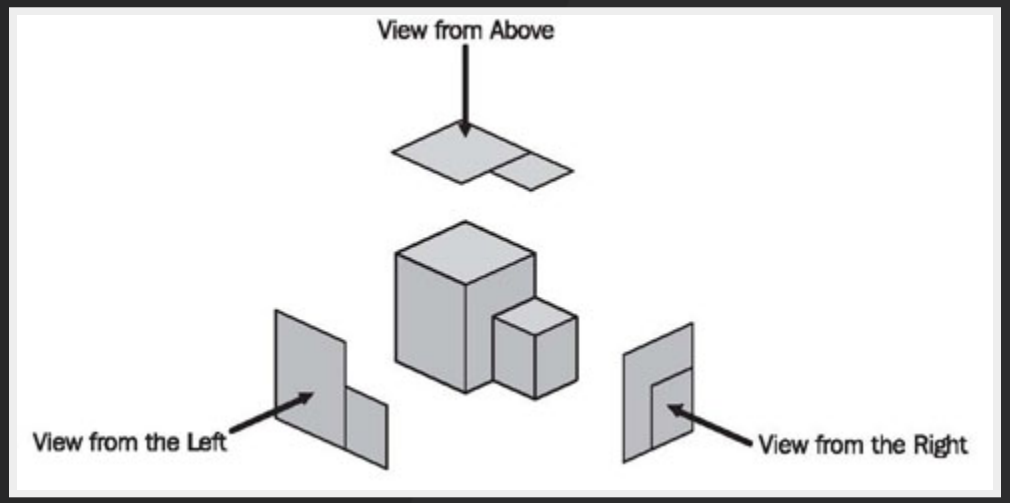
so that they can keep the database in good form...

# Companies

*tend to value*

- SQL Server

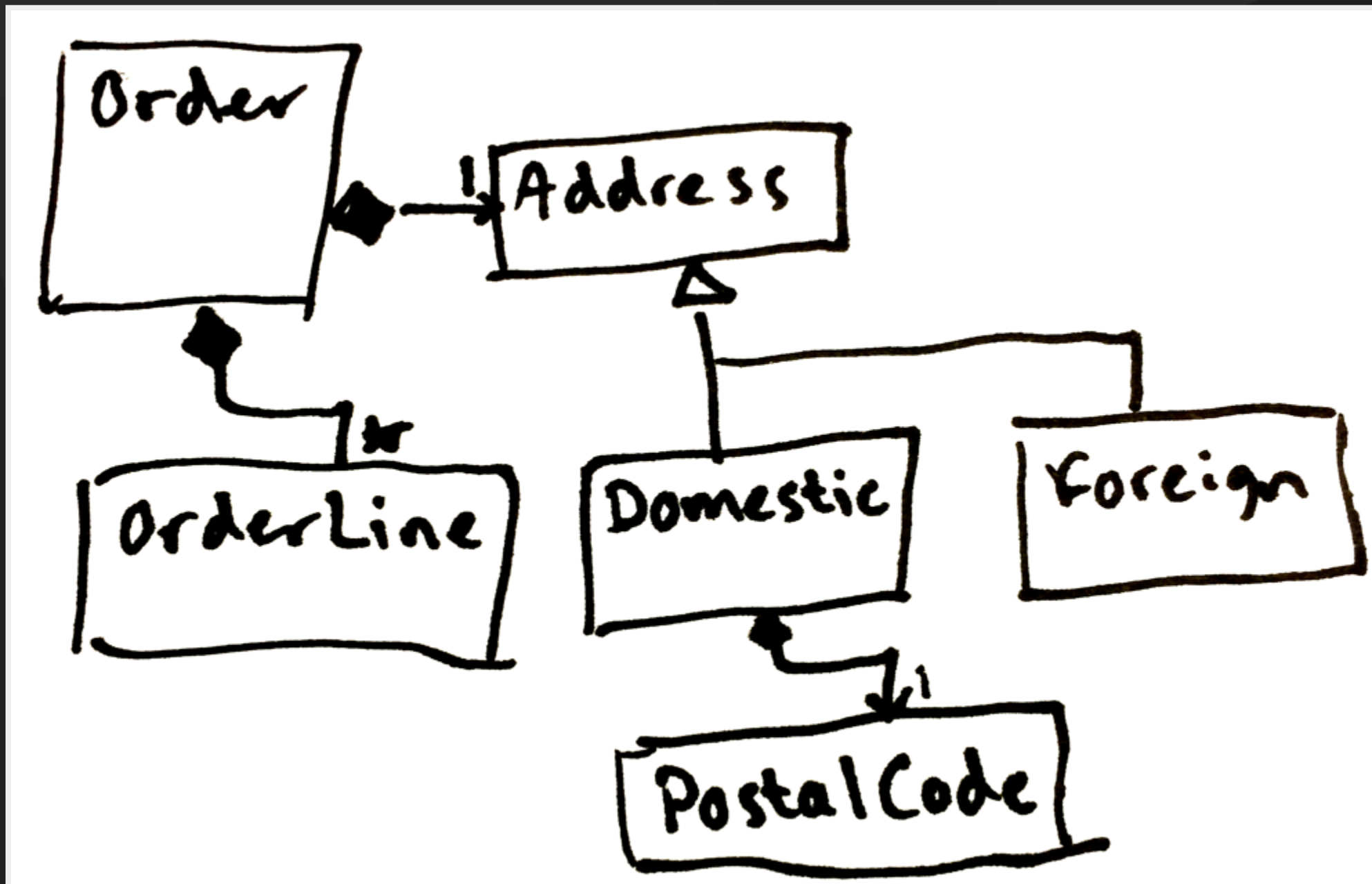
so that they can have their reports...


View from Above
View from the Left
View from the Right

# Object-oriented models

Problem: High granularity + impedance mismatch

Example: An order...




How can we persist such an atrocity?

# Document database

```
{
        what: "a document!"
}
```

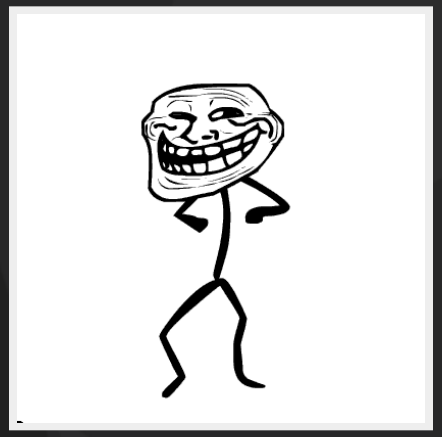
```
{
        what: "a document!",
        with: {
                what: "an embedded document"
        },
        and: [
                {text: "an array"},
                {text: "of documents"}
        ]
}
```

# Document database

```
{
        OrderLines: [
                { Product: "Beer", Quantity: 6 },
                { Product: "Nuts", Quantity: 2 }
        ],
        DeliveryAddress: {
                Street: "Torsmark",
                HouseNumber: "4",
                PostalCode: "8700",
                City: "Horsens"
        }
}
```

# Document database

- MongoDB
- Azure Document DB
- RavenDB
- CouchDB
- ...
- **SQL Server**



# Demo 1

SQL Server as a document-oriented database with HybridDB

Demo code is here:
https://github.com/d60/SqlServerAllTheThings/tree/master/HybridDbDemo

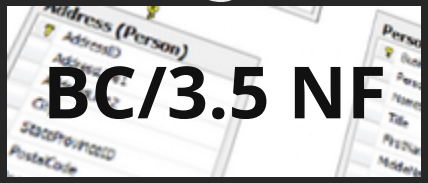
# What was that?

- HybridDB
- Linq provider
- JSON-serialization
- Indexing
- Open source, MIT
- Other options:
  - SisoDB
  - Hand-rolled JSON-serialization for the occasion

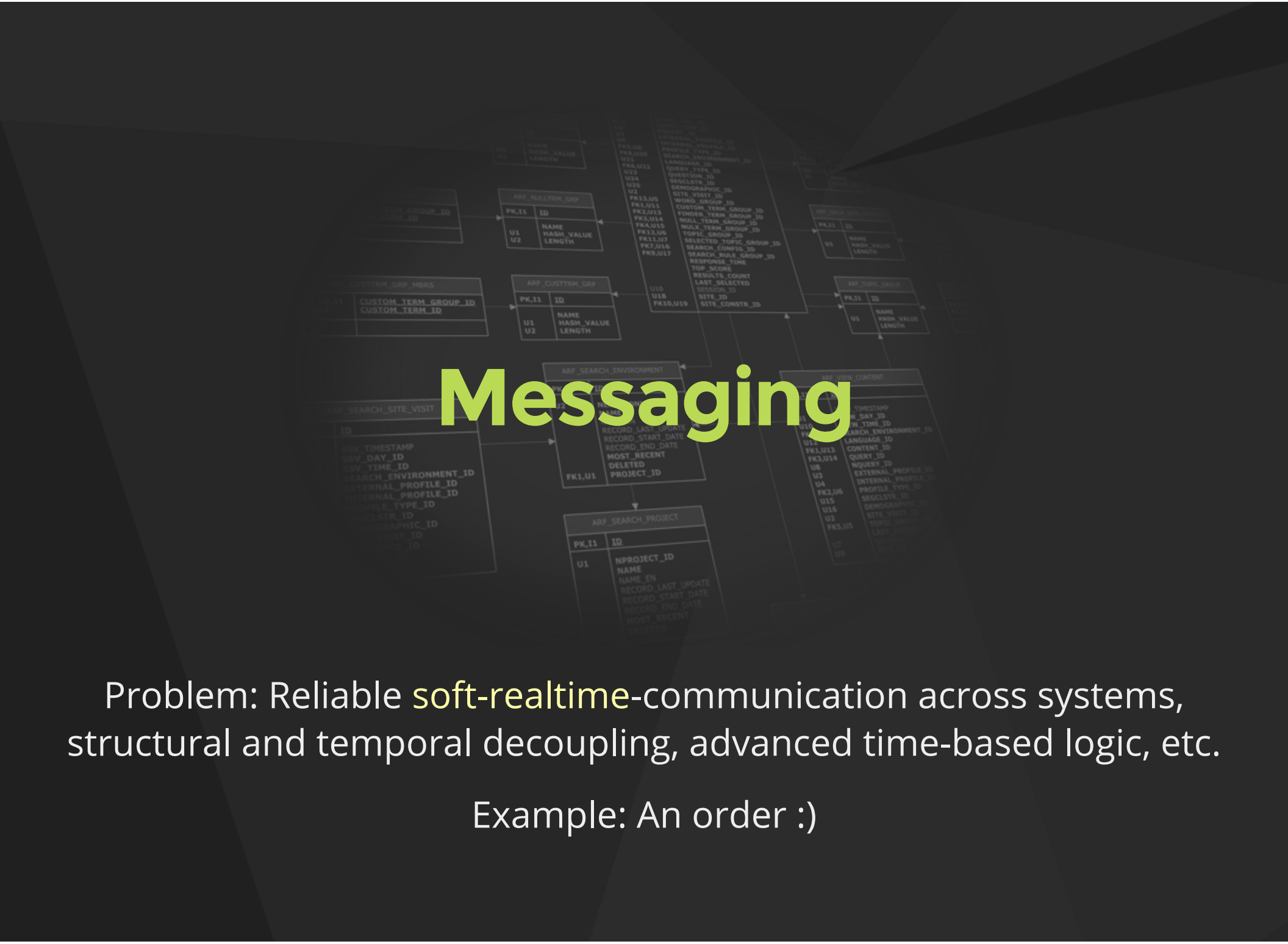
# Advantages

- "Dynamic" model
- Agile and fast
- Trivial to model with proper value types like e.g. `Currency`, `Money`, `PostalCode`, `Date`, `Week`, `Range<T>`, `Ssn`, `AccountNumber` etc etc.
- High granularity does not impact performance nearly as much

# Disadvantages

- Not exactly 

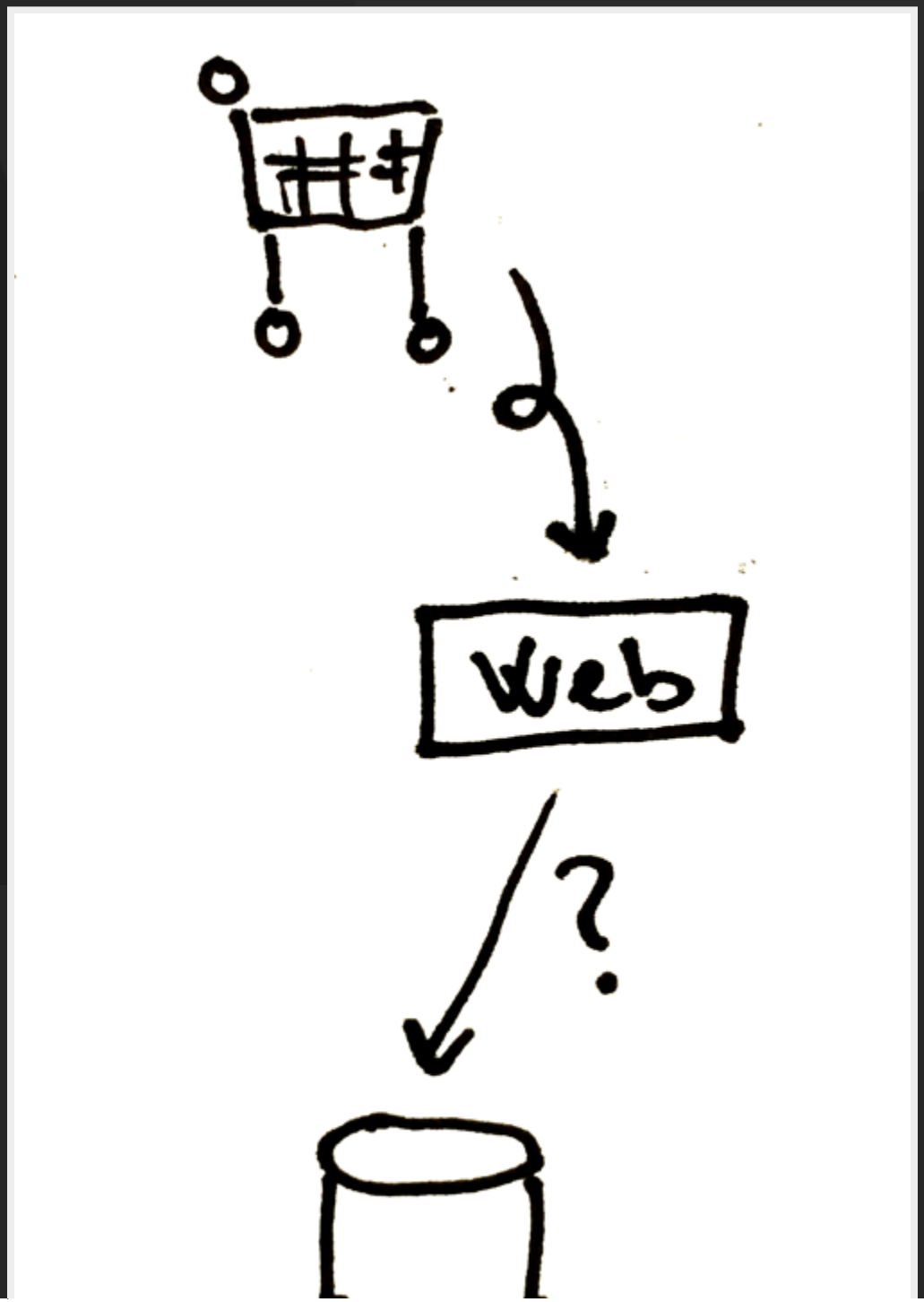
- Black Box



# Messaging

Problem: Reliable soft-realtime-communication across systems, structural and temporal decoupling, advanced time-based logic, etc.

Example: An order :)


Web
?

Error 1205 : Transaction (Process ID) was deadlocked on resources with another process and has been chosen as the deadlock victim. Rerun the transaction

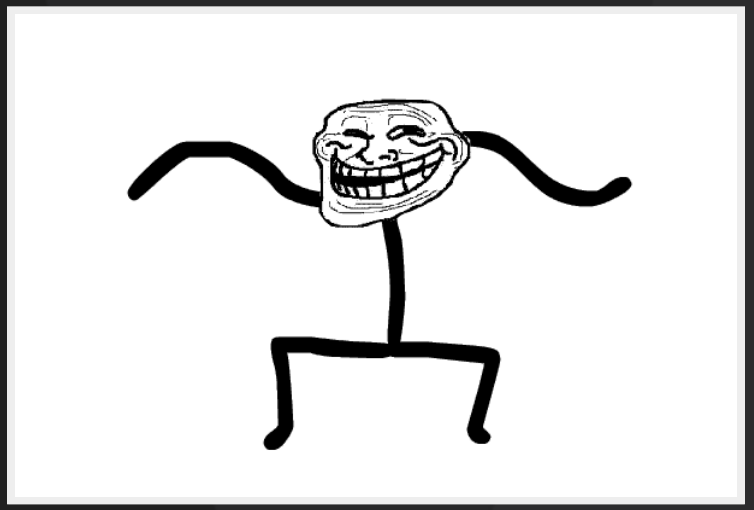
# To place an order

- Reliable
- Scalable
- Integration with
  - inventory
  - shipping
  - finance
- "Buyer's remorse"

# Message queues

- MSMQ
- RabbitMQ
- Azure Service Bus
- Apache ActiveMQ
- ...
- SQL Server Service Broker ~~SQL Azure~~ 

- **SQL Server**

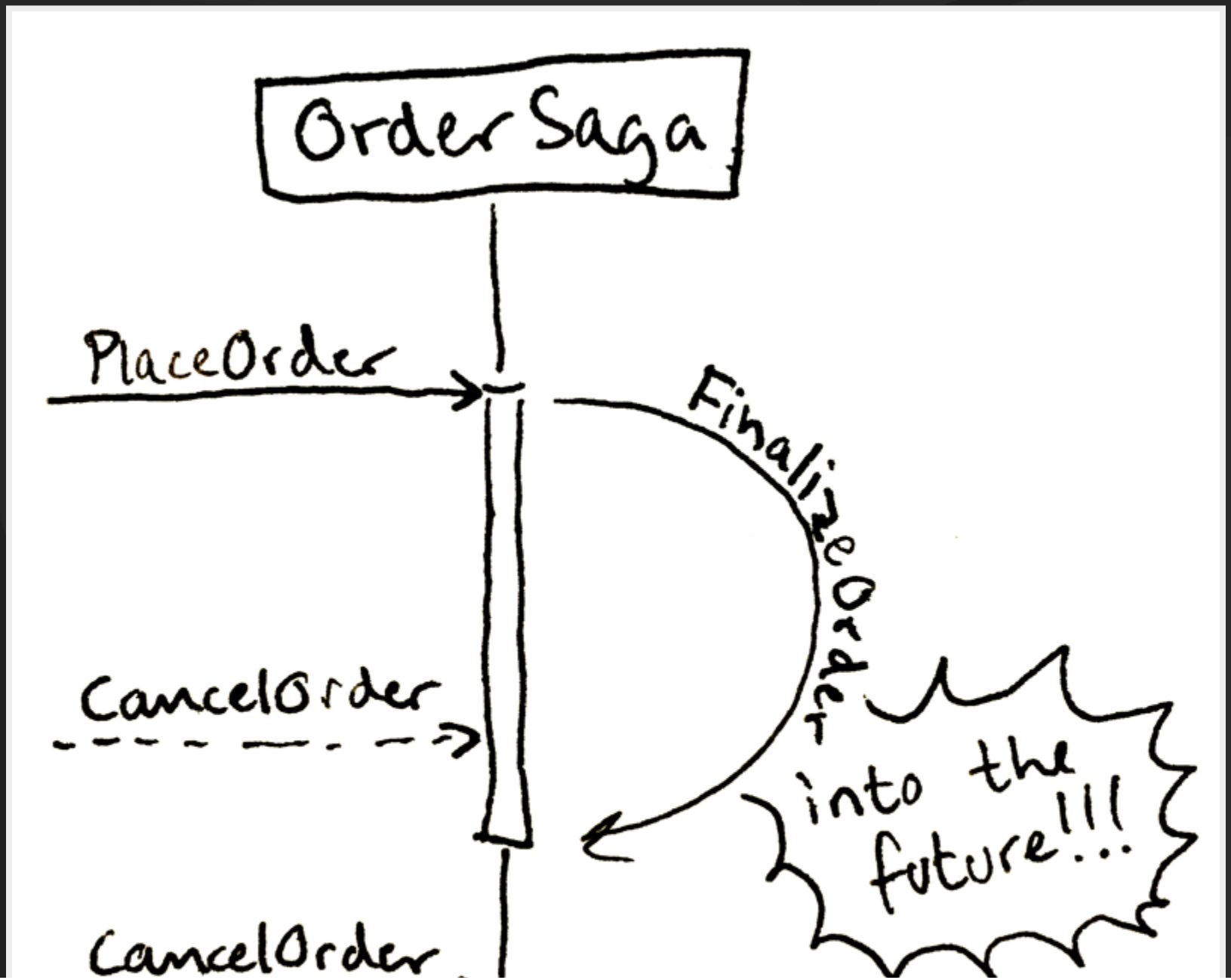


# Demo 2

SQL Server as a message queue with Rebus

Demo code is here:
https://github.com/d60/SqlServerAllTheThings/tree/master/RebusDemo

# What was that?



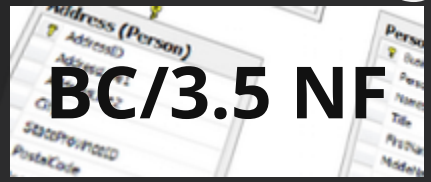
# What was that?

- Rebus
- Open source, MIT
- Reliable messaging
- Carefully crafted non-locking SQL
- Complex time-based asynchronous workflow

# Advantages

- Does not require extra infrastructure
- Has decent performance
- Easily portable to e.g. MSMQ, Azure Service Bus, etc.

# Disadvantages

- Not exactly 

- Black Box
- Centralized
- Does not scale like real message queues

# Event sourcing

Problem: Event-based domain model in a CQRS-architecture

"How did we get here?"

Example: An order :) :) :)

# Event sourcing

- All state is described with events.
- I mean everything.
- Everything(*).

(* seriously)

:)

# Event sourcing

```
{ _id: "order/8", _type: "OrderCreated"   }
```

```
{       _id: "order/8", _type: "ItemAdded",
        item: "Beers", quantity: 6            }
```

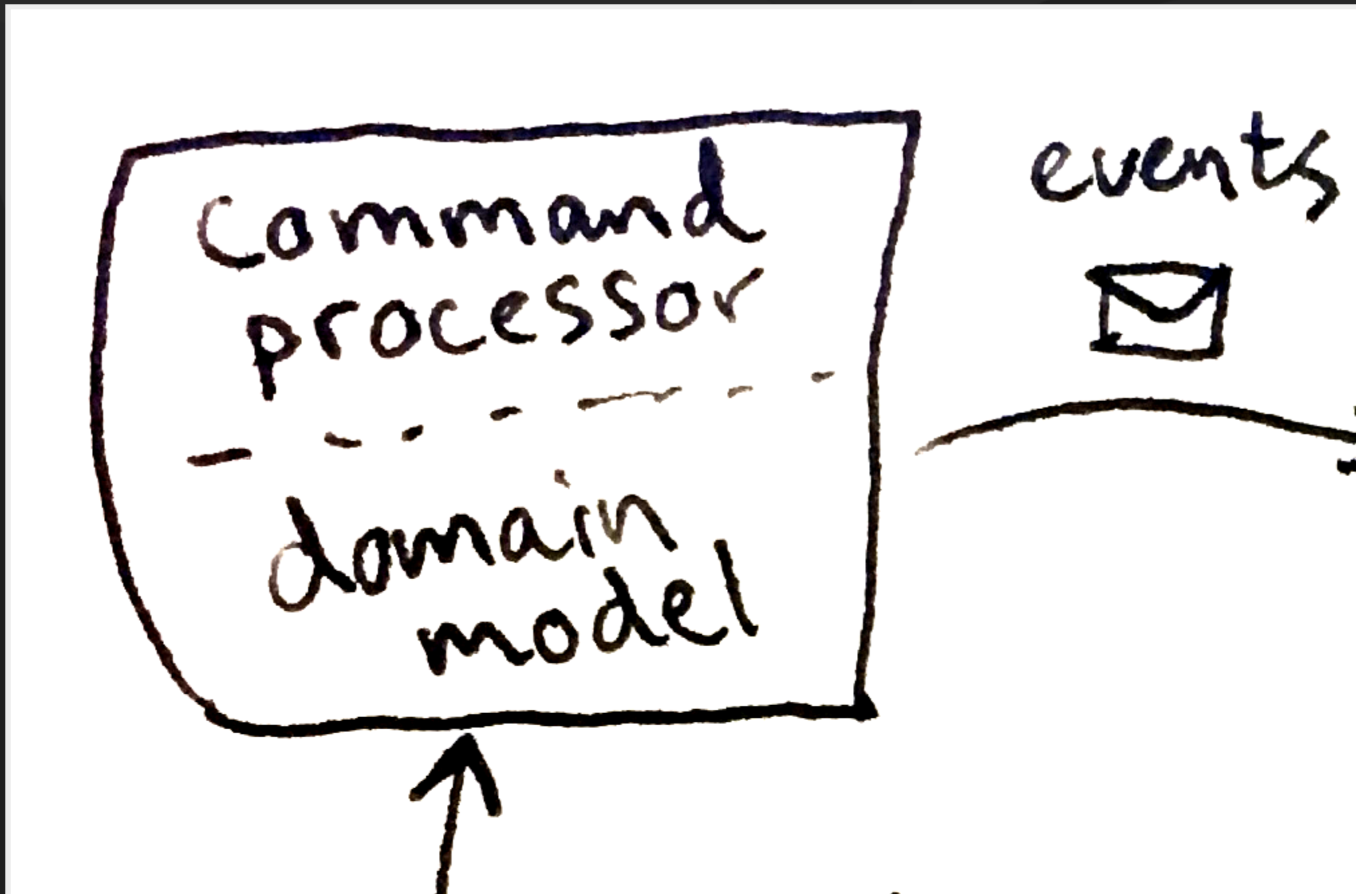
```
{       _id: "order/8", _type: "ItemAdded",
        item: "Nuts", quantity: 2             }
```

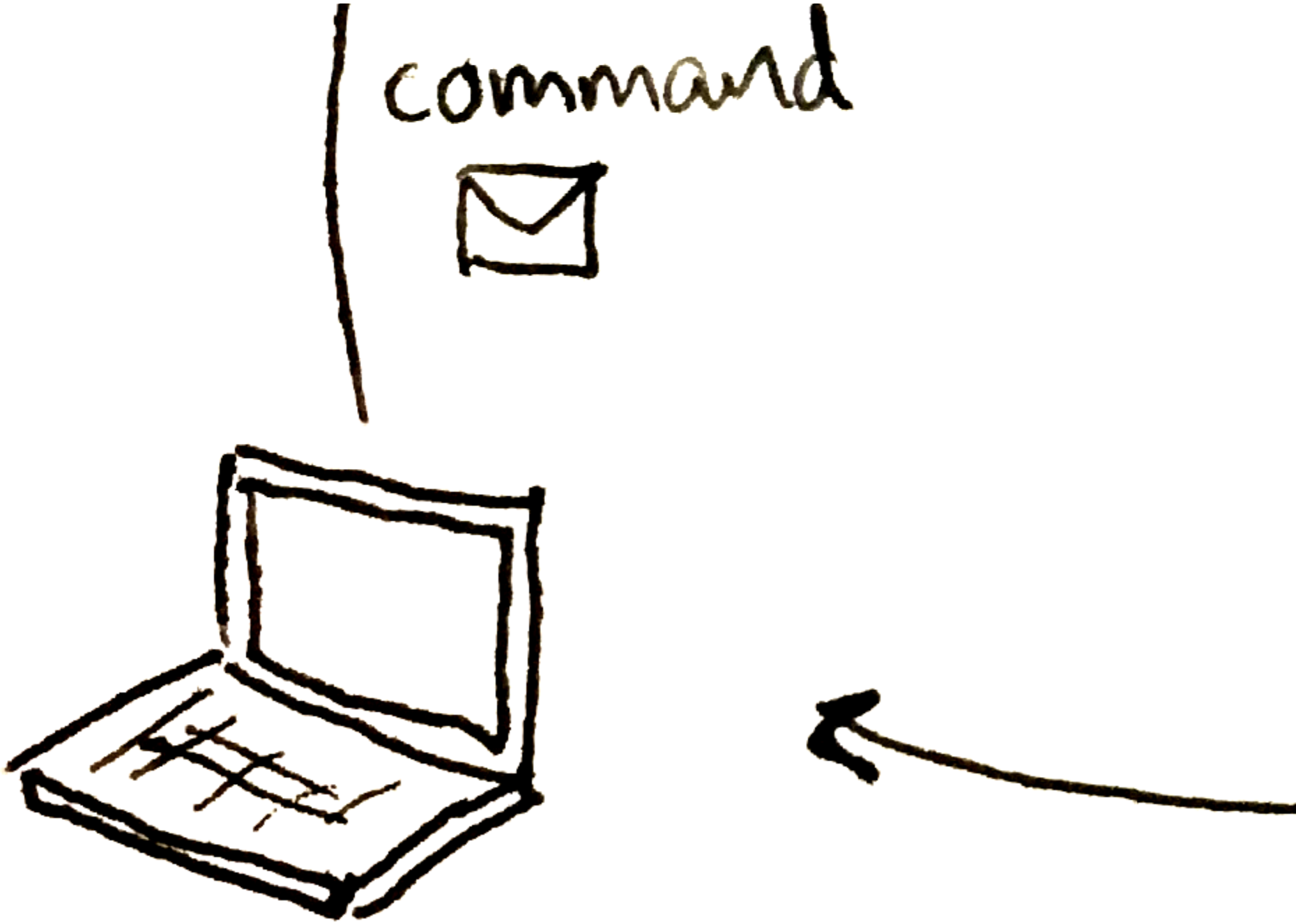
```
{       _id: "order/8", _type: "ItemAdded",
        item: "Big TV", quantity: 1           }
```

```
{       _id: "order/8", _type: "ItemRemoved",
        item: "Big TV", quantity: 1           }
```

```
{       _id: "order/8",
        _type: "ShipmentAddressSpecified",
        street: "Torsmark", houseNumber: "4",
        postalCode: "8700", city: "Horsens"   }
```

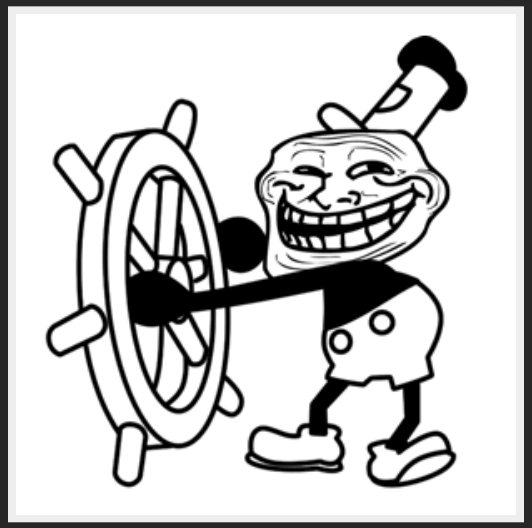
# Event sourcing + CQRS




command

# Event-centric databases

- Event Store
- Datomic
- ...
- **SQL Server**



# Demo 3

SQL Server as an event store with Cirqus

Demo code is here:
https://github.com/d60/SqlServerAllTheThings/tree/master/CirqusDemo

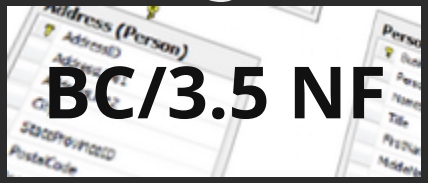
# What was that?

- Cirqus
- Event sourcing and CQRS framework
- Domain model built from events
- Open source, MIT
- Asynchronous projections/views
  - MongoDB
  - PostgreSQL
  - In-mem
  - SQL Server
  - HybridDB
- Other options:
  - NCQRS
  - CQRSLite
  - Agr.CQRS
  - Lokad CQRS
  - ...

# Advantages

- Does not require additional infrastructure
- Can perform very very well
- Full audit trail is built-in
- Extreme flexibility + scalability

# Disadvantages

- Not exactly 

- Black Box
- Paradigm shift

# Conclusion

# A couple of my own

- SQL Server as a key-value store with automatic secondary indexes... allright! :)
- SW developers: Be creative :) - your company's SQL Server locked-down policy need not prevent you from building great solutions.
- SW developers: Understand the operational consequences dimension.
- BI developers: You can still have your tables :) (you were not going to mess with the production DB anyway, were you?)

# Your comments?

# Thanks for listening!

...and a huge shout-out to Hakim because he made it possible to code presentations in HTML and JavaScript


Mogens Heller Grabe

mhg@d60.dk
@mookid8000
http://mookid.dk/oncode/

## ou to our Sponsors :

PLATINUM

Kroll Ontrack.

Profis

GOLD

PYRAMID ANALYTICS

d60

Rehfeld

redgate

SILVER

COZYROC™
Go to the next level

Say thank yo

BRONZE



devart

CatMan® Solution

# Stick around for RAFFLE and the

- All our volunteers and organise
  organizing this event – If you s
  - Give them a hug
  - Shake their hand
  - Say thank you
  - Spread the word
  - Get involved yourself

- Don’t forget to thank the spons
- Thank the speakers for donatin expenses