# The Heart of Domain-Driven Design

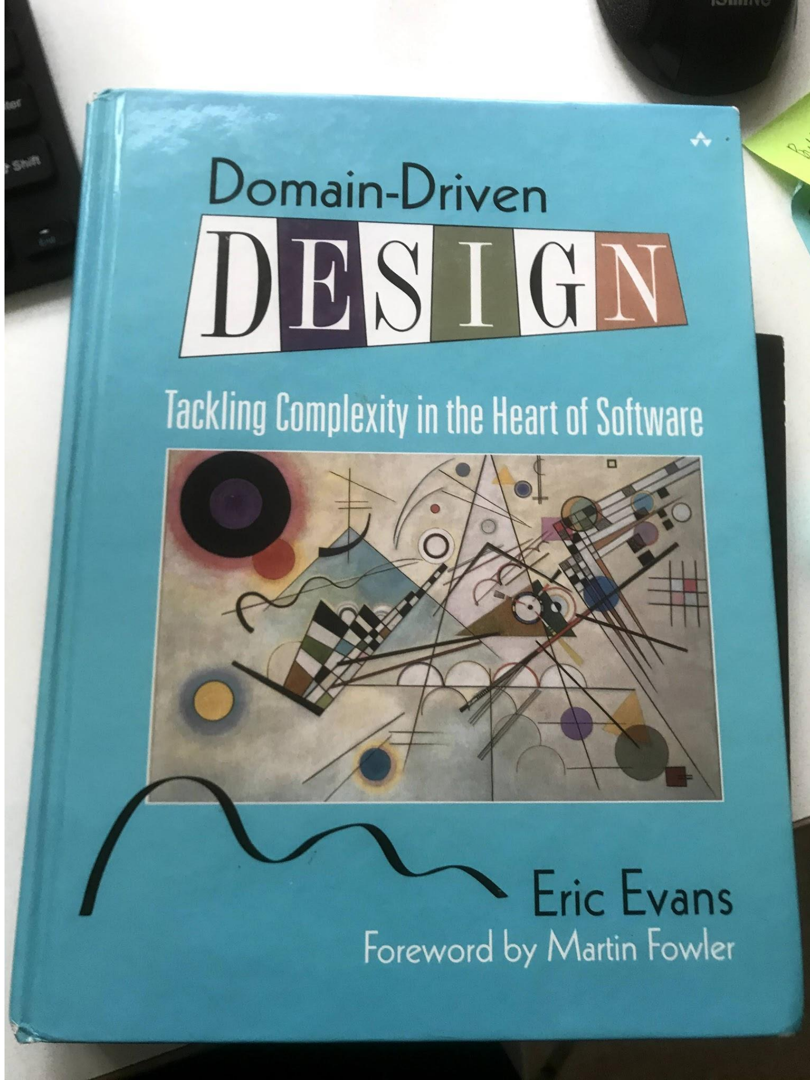
Andrew Harmel-Law, Tech Principal, ThoughtWorks

(Now with Microservices!)

@AL94781 #HeartOfDDD

## A little about me…

(check out my course;
"Domain-Driven Design: First Steps"
on O'Reilly)



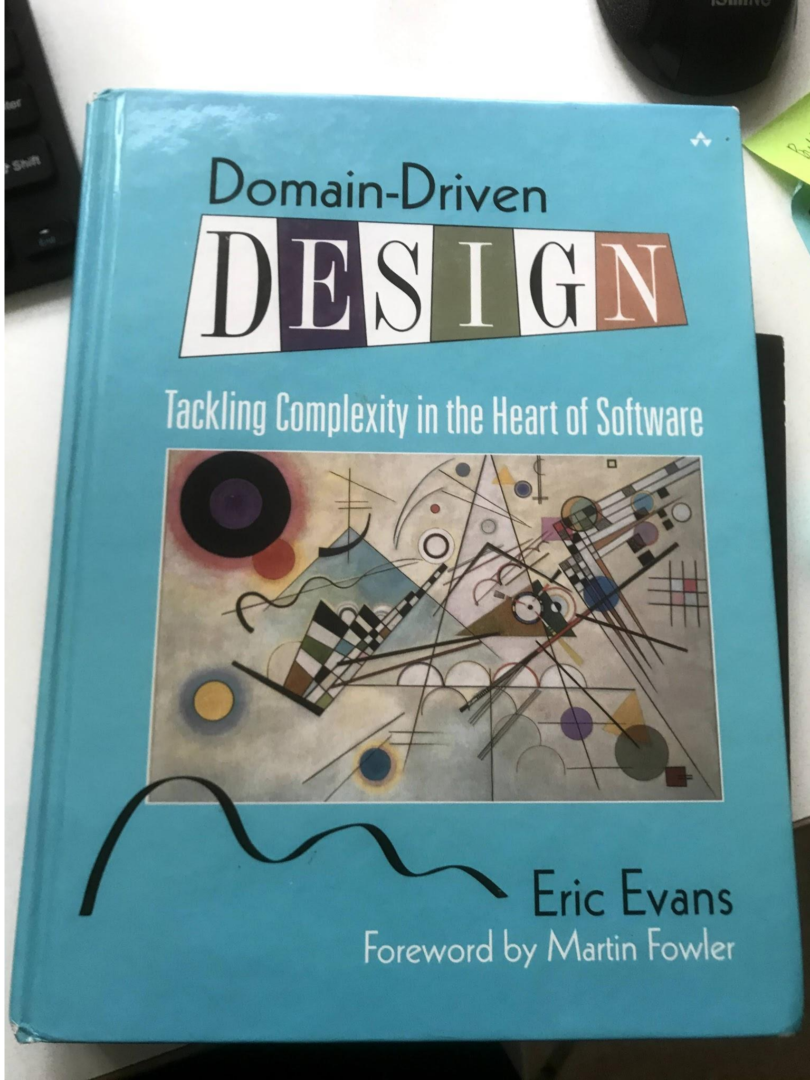
# Domain-Driven DESIGN

Tackling Complexity in the Heart of Software

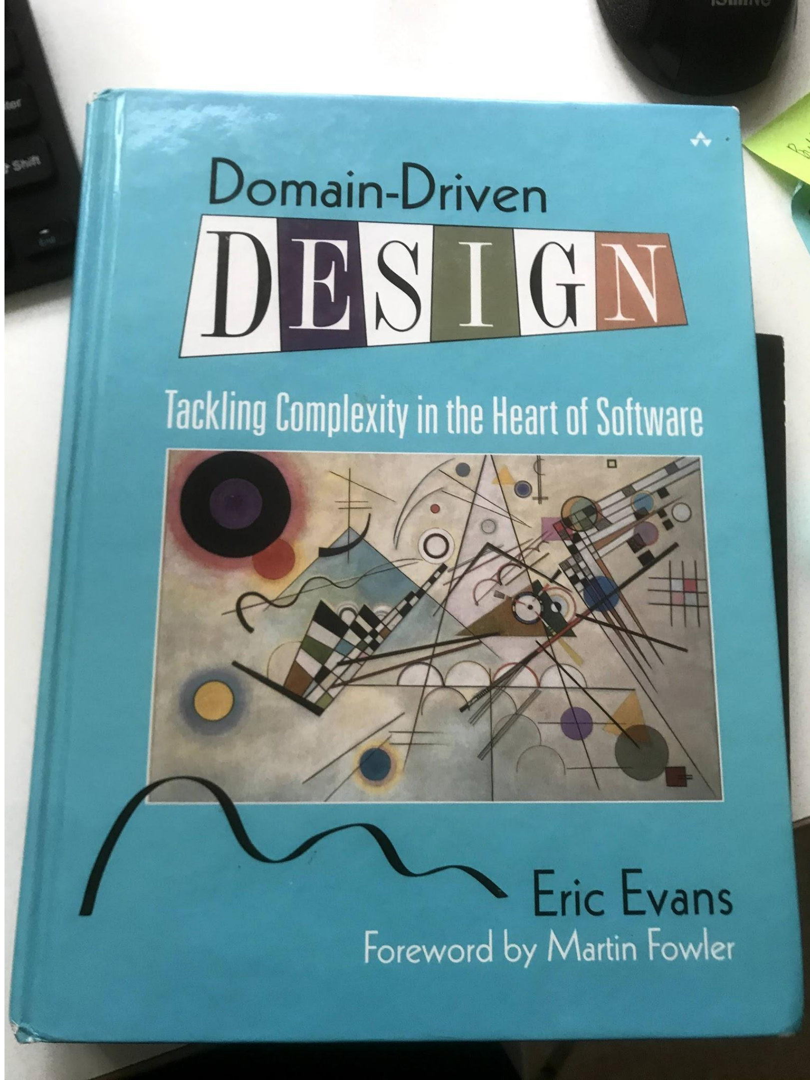
Eric Evans

Foreword by Martin Fowler

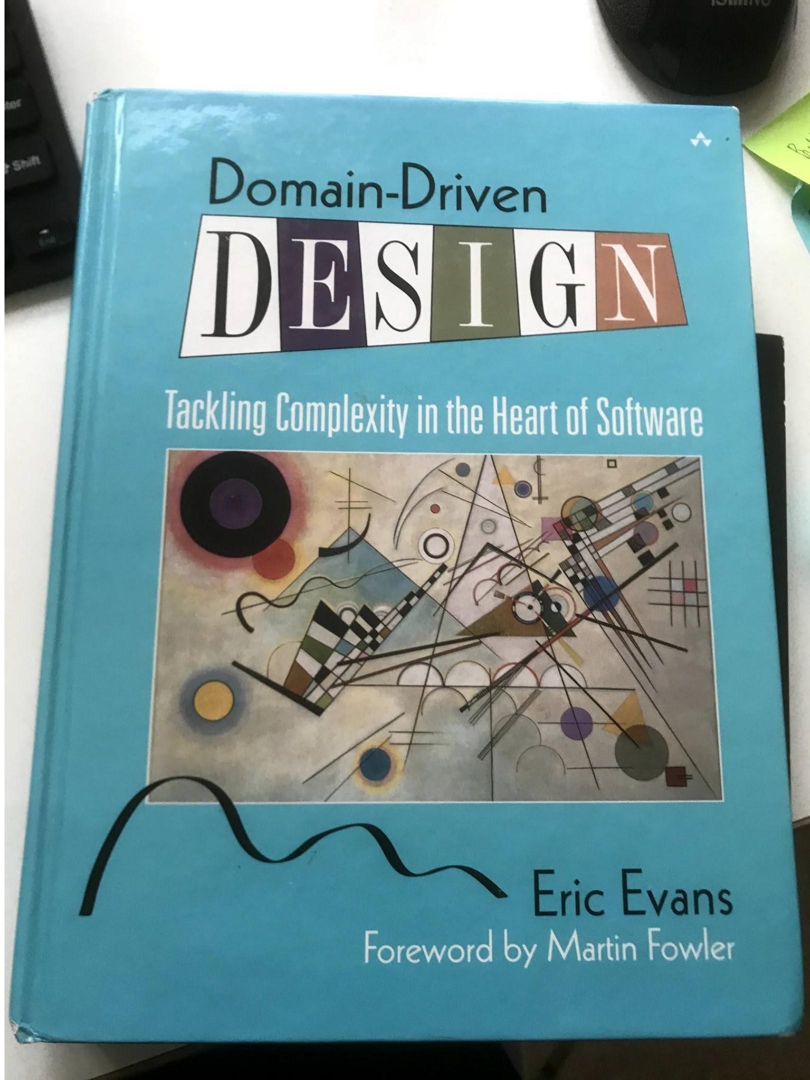
"Tackling complexity in the heart of software"

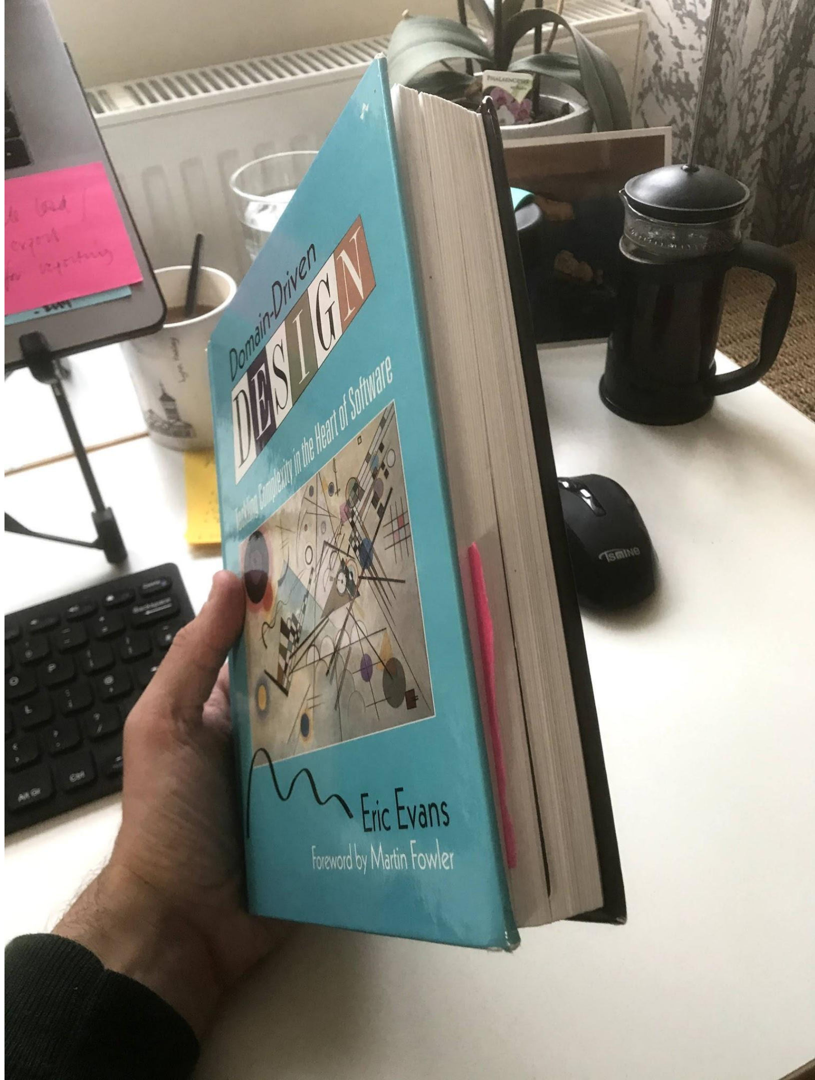

Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler

"Tackling complexity in the heart of software"


Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler

“Tackling complexity in the heart of software”


Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler


Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler

## Ubiquitous language

## Ubiquitous language

## Hands-on modelers

Ubiquitous language
Hands-on modelers
Multiple models

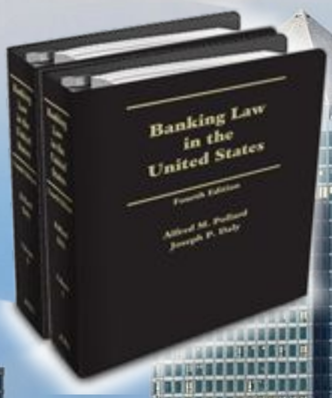
Ubiquitous language
Hands-on modelers
Multiple models

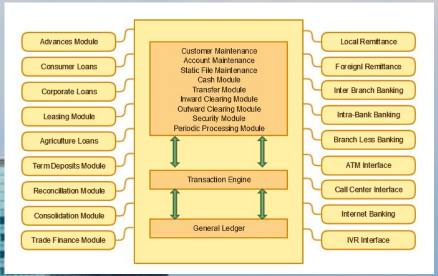
## Ubiquitous language

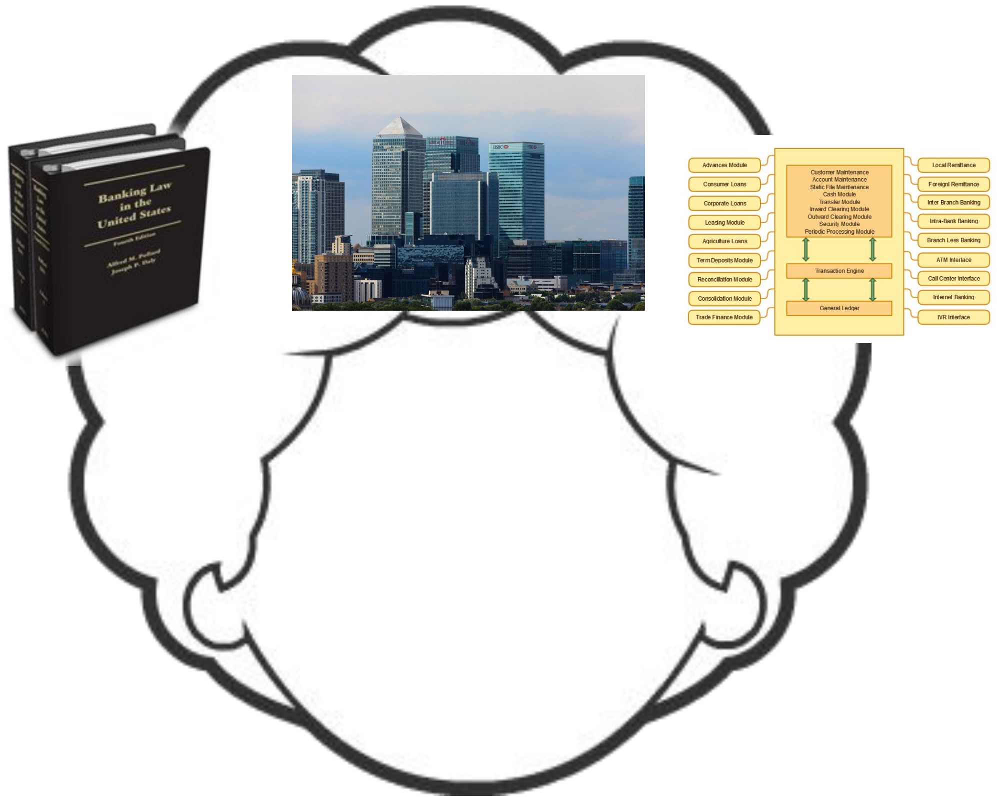
citi
citi
HSBC
HSBC

Banking Law in the United States
Fourth Edition
Alfred M. Pollard
Joseph P. Daly
citi
HSBC

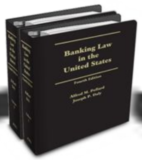
citi
citi
HSBC

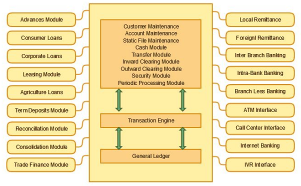

Banking Law in the United States
Fourth Edition
Alfred M. Pollard
Joseph P. Daly


Advances Module
Consumer Loans
Corporate Loans
Leasing Module
Agriculture Loans
Term Deposits Module
Reconciliation Module
Consolidation Module
Trade Finance Module
Customer Maintenance
Account Maintenance
Static File Maintenance
Cash Module
Transfer Module
Inward Clearing Module
Outward Clearing Module
Security Module
Periodic Processing Module
Transaction Engine
General Ledger
Local Remittance
Foreignl Remittance
Inter Branch Banking
Intra-Bank Banking
Branch Less Banking
ATM Interface
Call Center Interface
Internet Banking
IVR Interface



@AL94781 #HeartOfDDD


Banking Law
in the
United States
Fourth Edition
Alfred M. Pollard
Joseph P. Daly

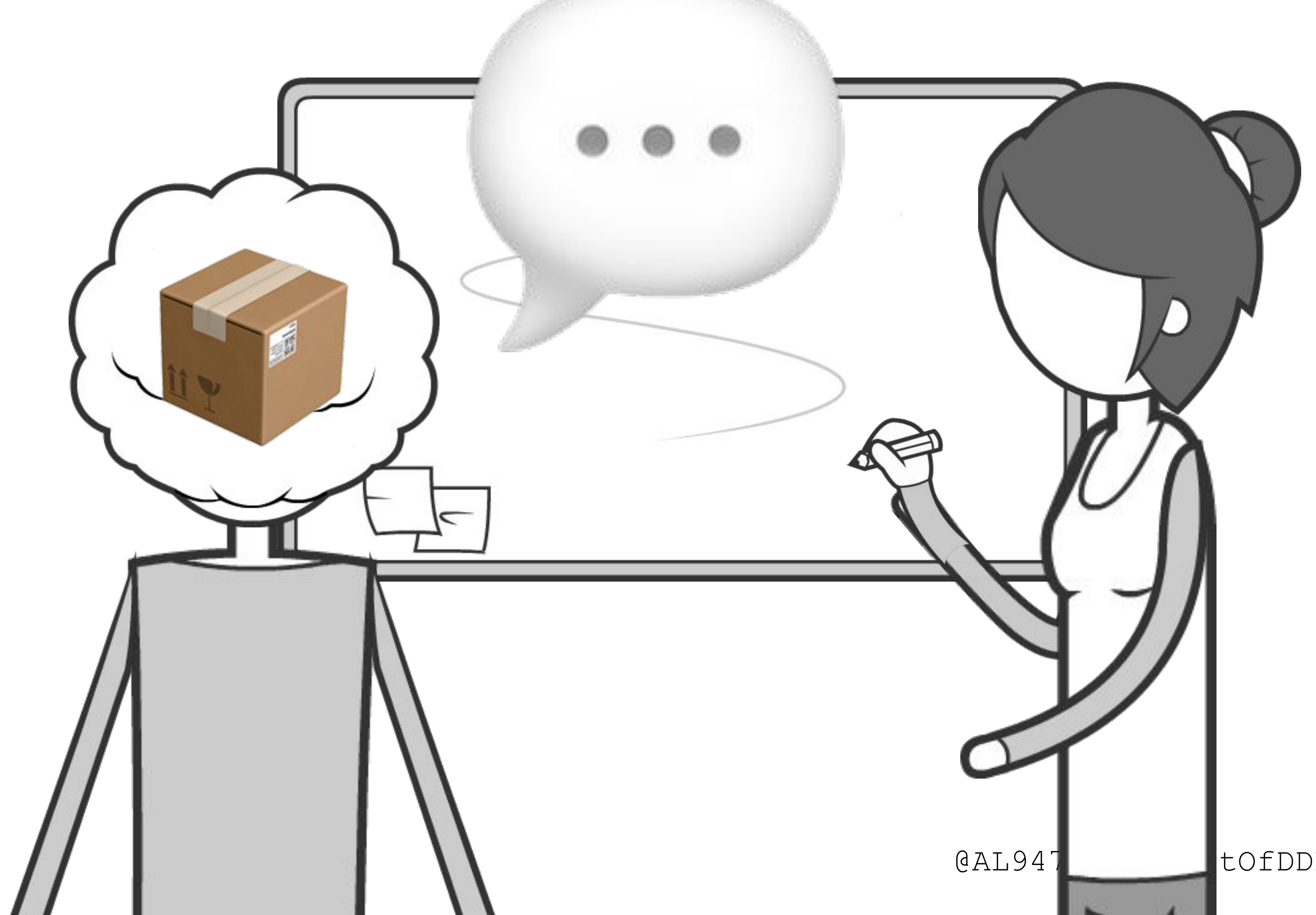

Advances Module
Consumer Loans
Corporate Loans
Leasing Module
Agriculture Loans
Term Deposits Module
Reconcillation Module
Consolidation Module
Trade Finance Module
Customer Maintenance
Account Maintenance
Static File Maintenance
Cash Module
Transfer Module
Inward Clearing Module
Outward Clearing Module
Security Module
Periodic Processing Module
Transaction Engine
General Ledger
Local Remittance
Foreignl Remittance
Inter Branch Banking
Intra-Bank Banking
Branch Less Banking
ATM Interface
Call Center Interface
Internet Banking
IVR Interface

FRAGILE

OfDDD

“A DOMAIN MODEL in software should be the distillation of [an] expert’s knowledge”

@AL947 tOfDDD



@AL947 tOfDDD

@AL947 tOfDDD

1011010110100001110101
11000110101100101 0
1111010100101011001
10010010101
10101000101000010101010

“Our domain models [are] ‘rigorously organised, selective abstractions’ of these mental equivalents”

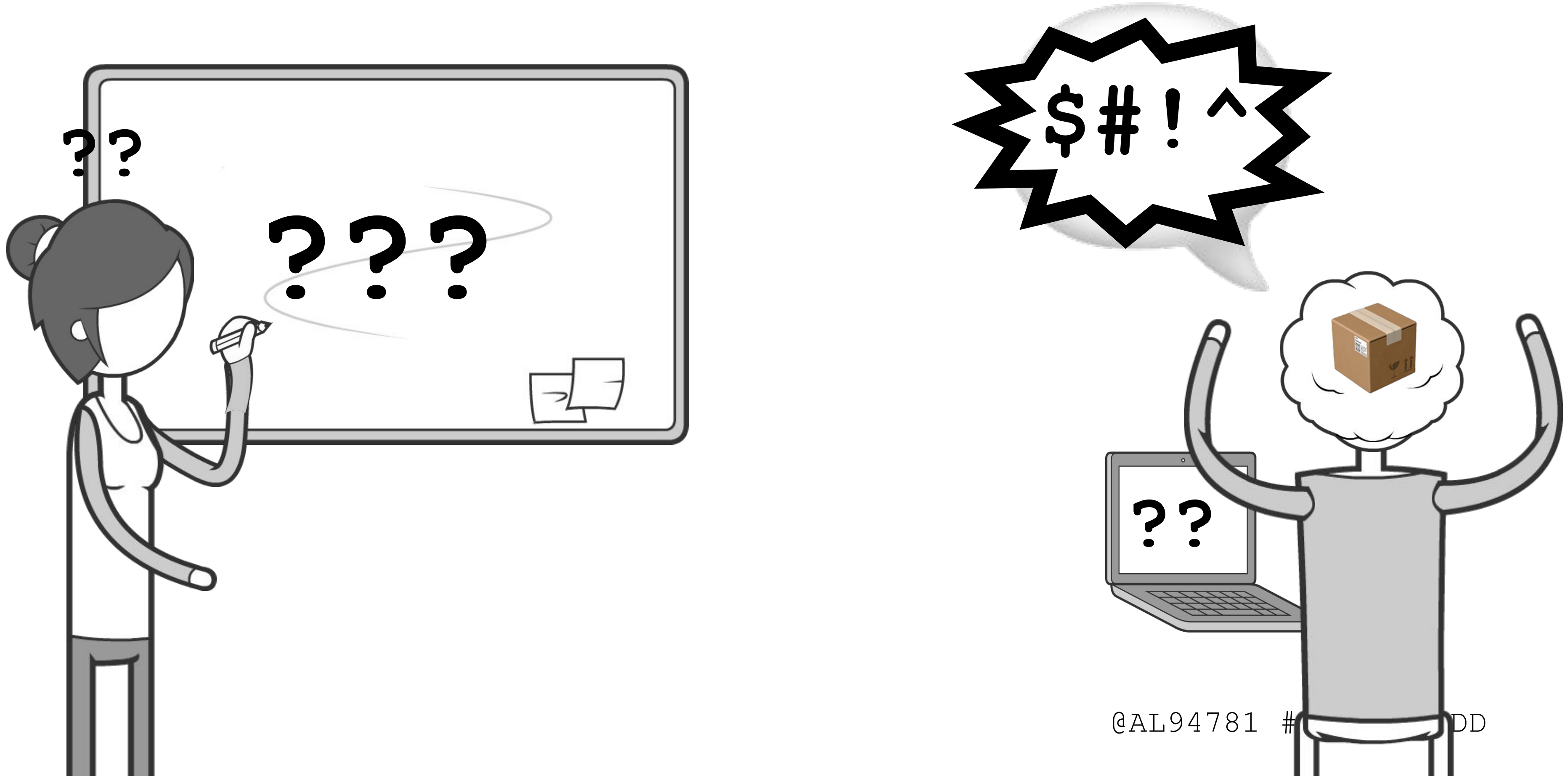
Domain

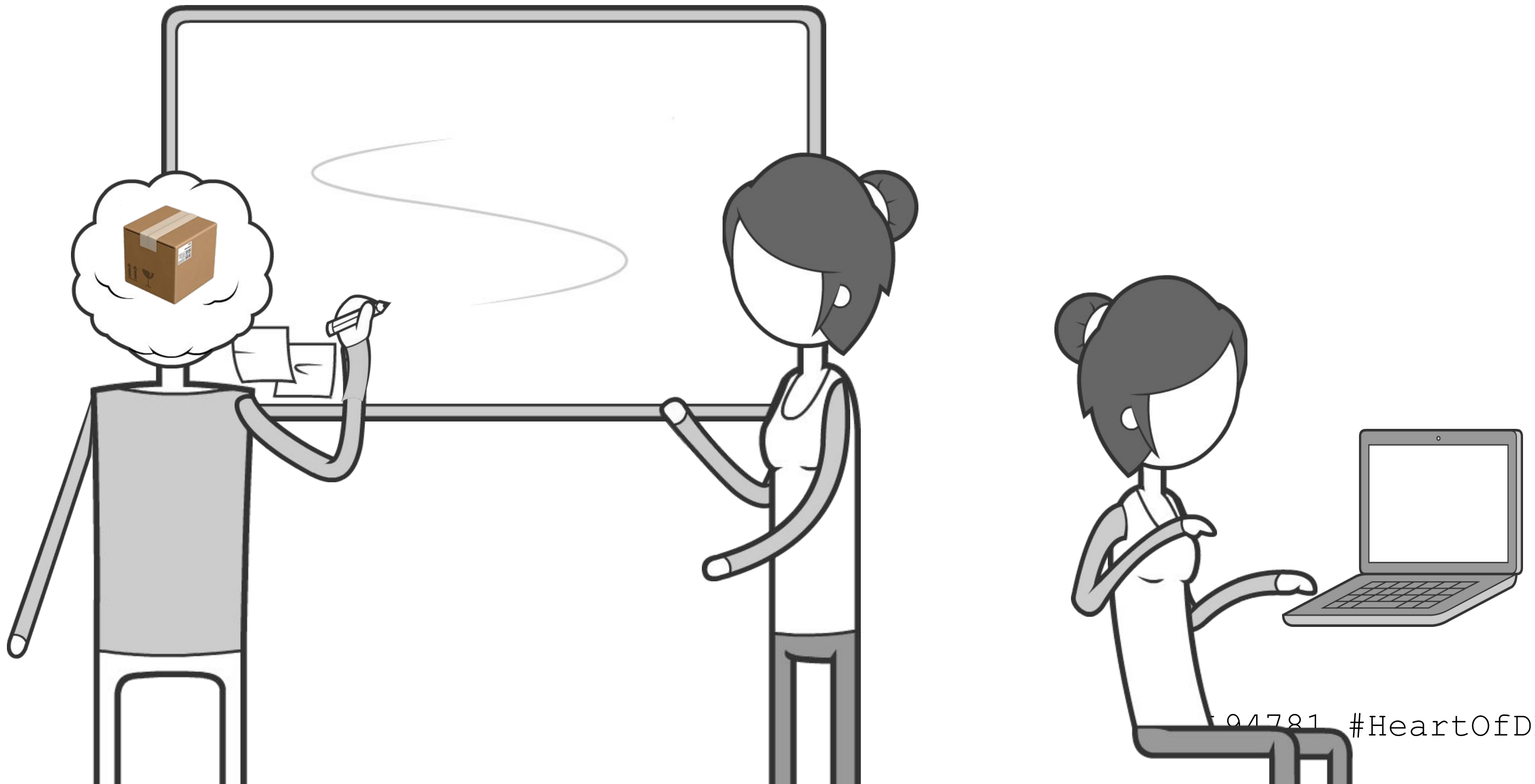
*language*

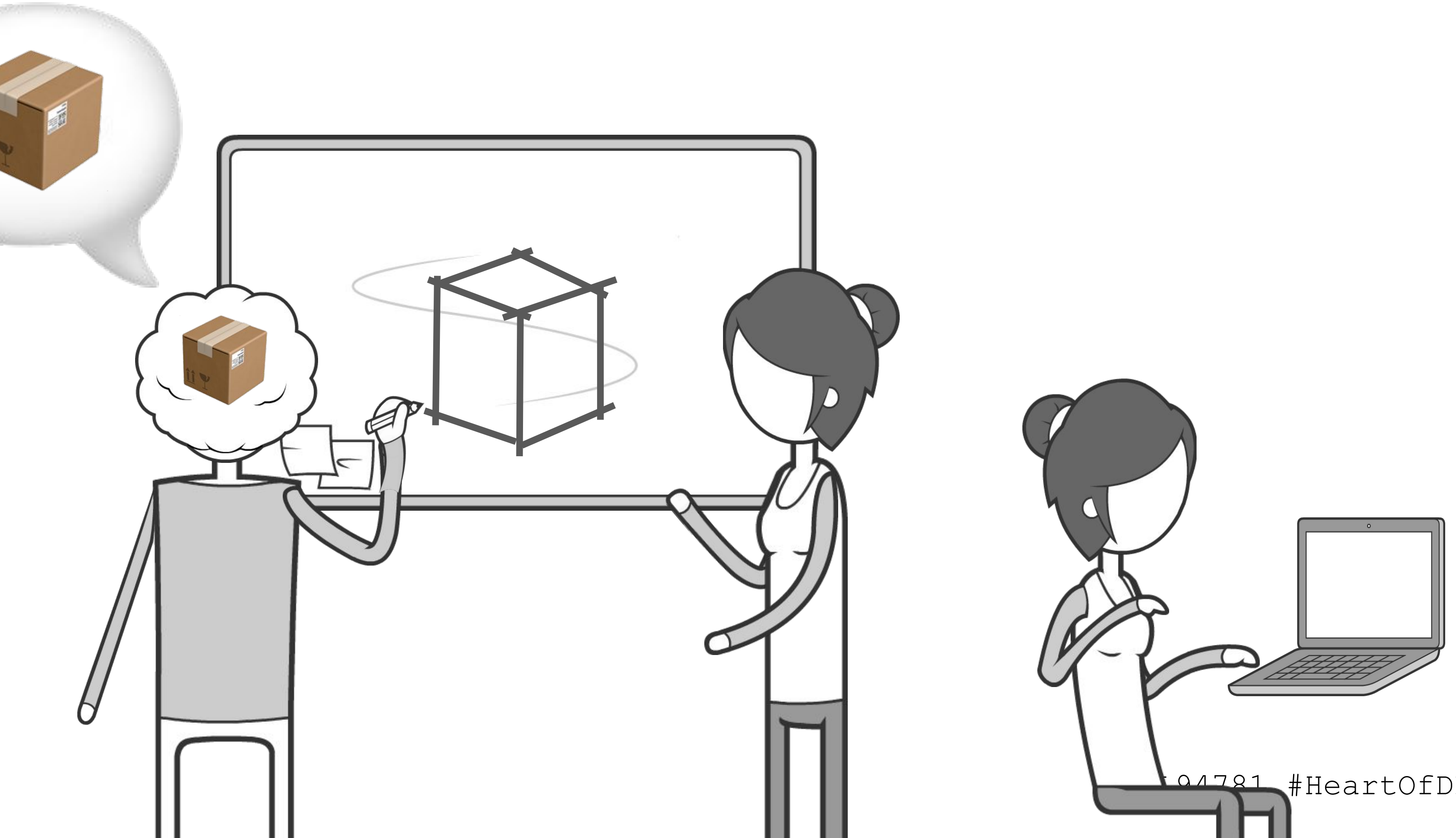
@AL94781 #HeartOfDDD

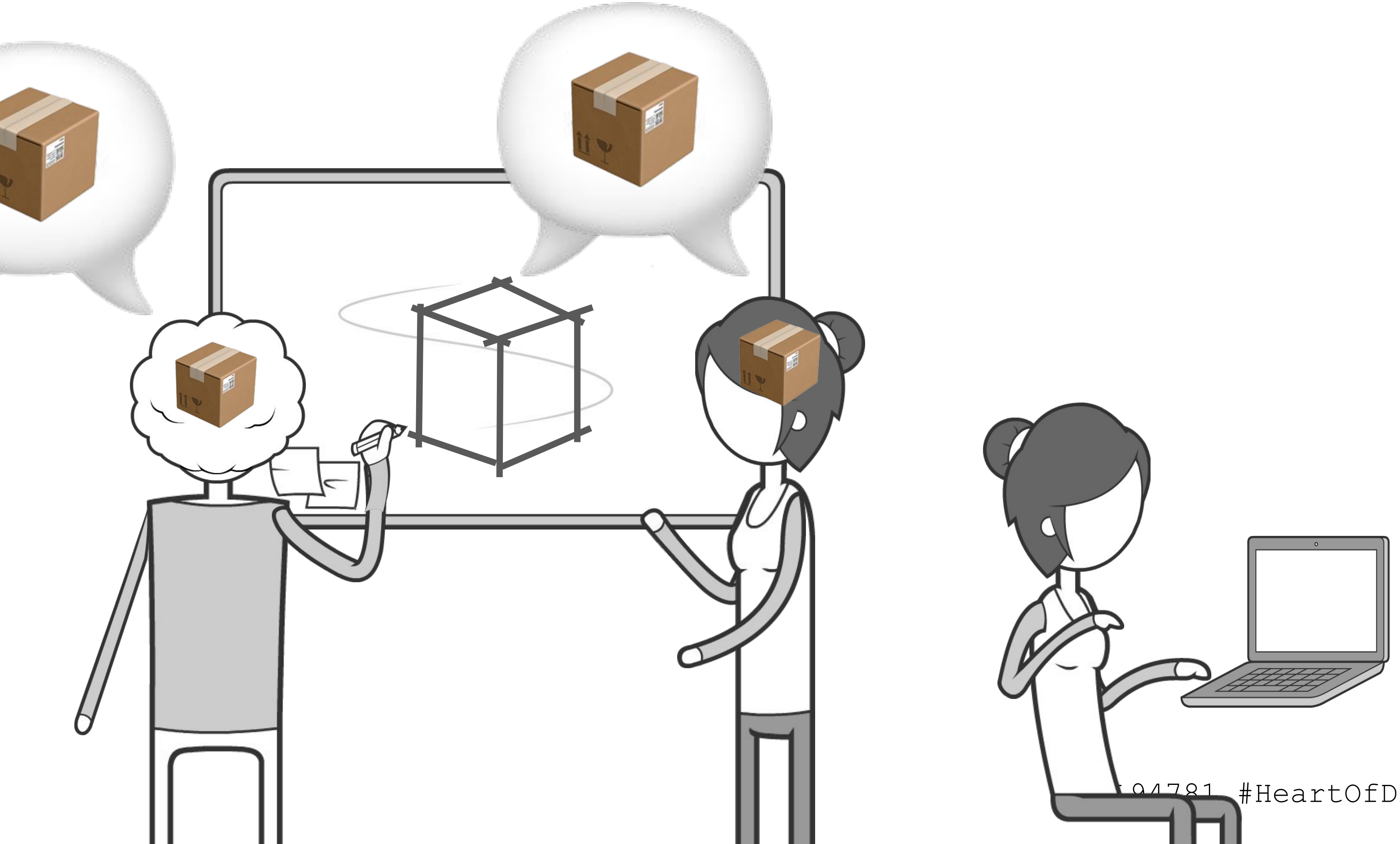
## Domain *language*

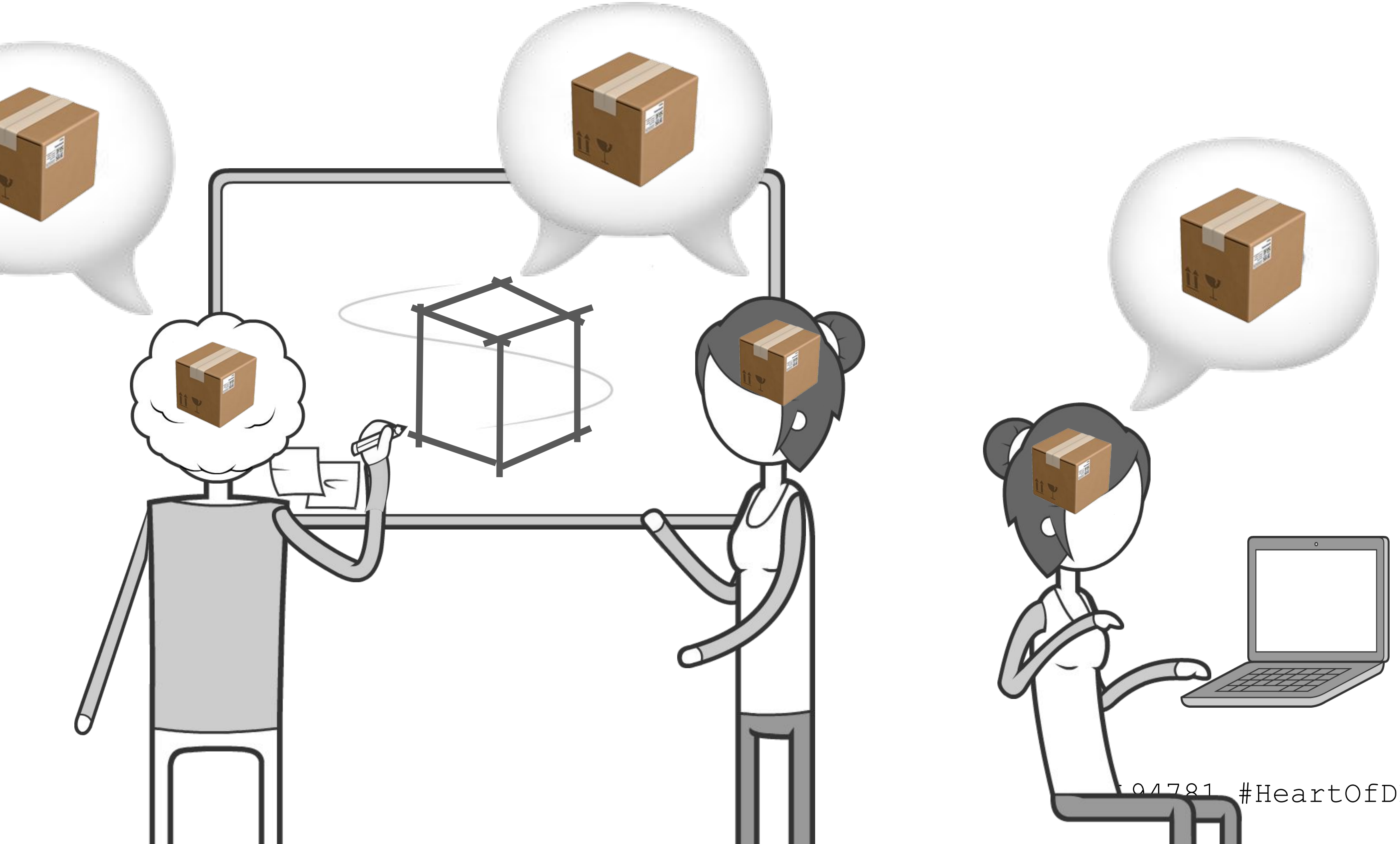
“As we talk to domain experts, we rapidly discover that their mental models go beyond ‘find the nouns’.”

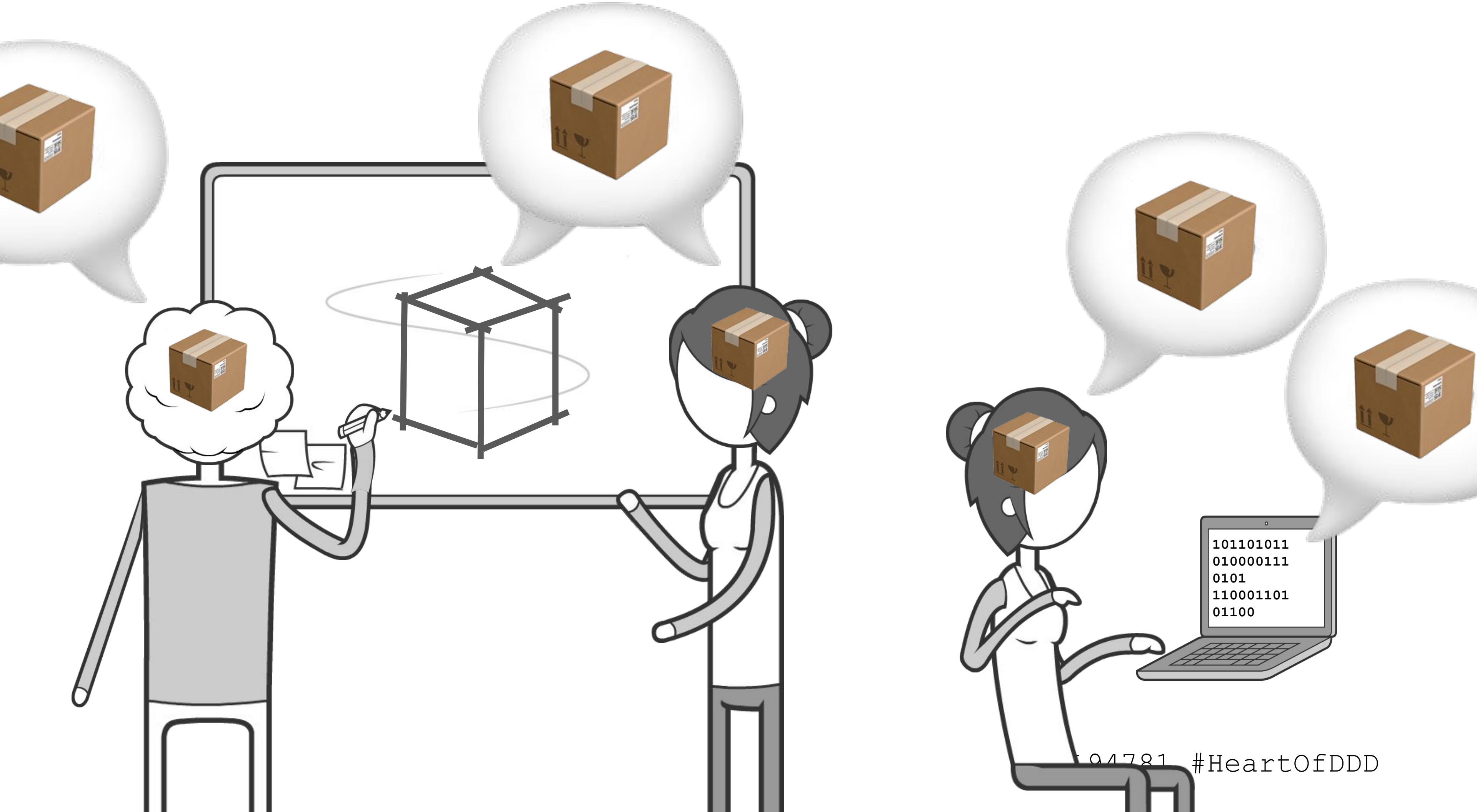


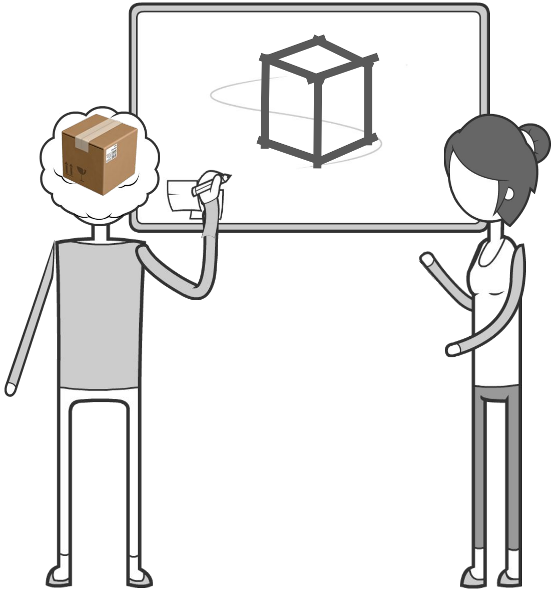


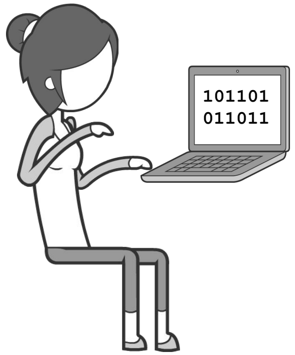


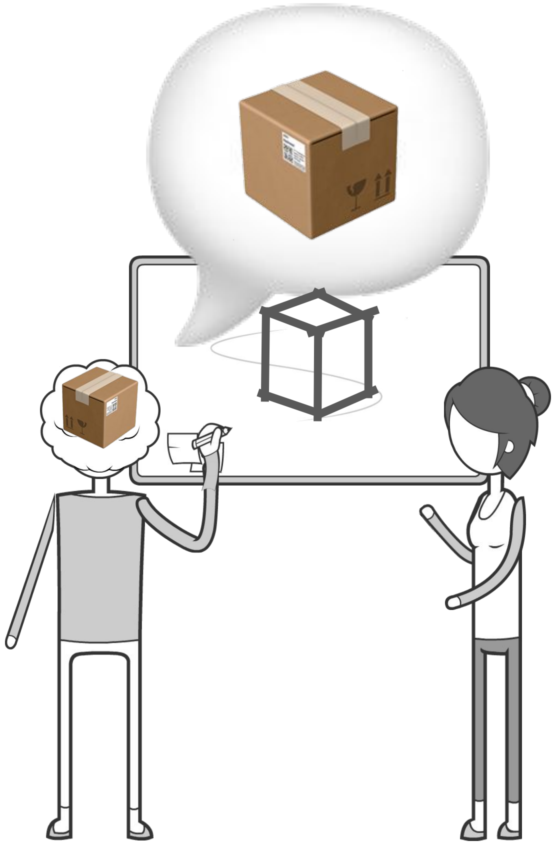


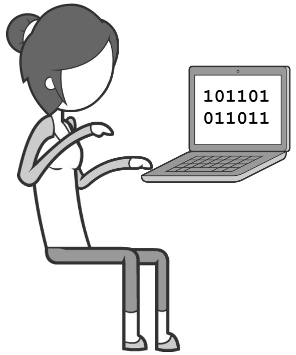



101101011
010000111
0101
110001101
01100

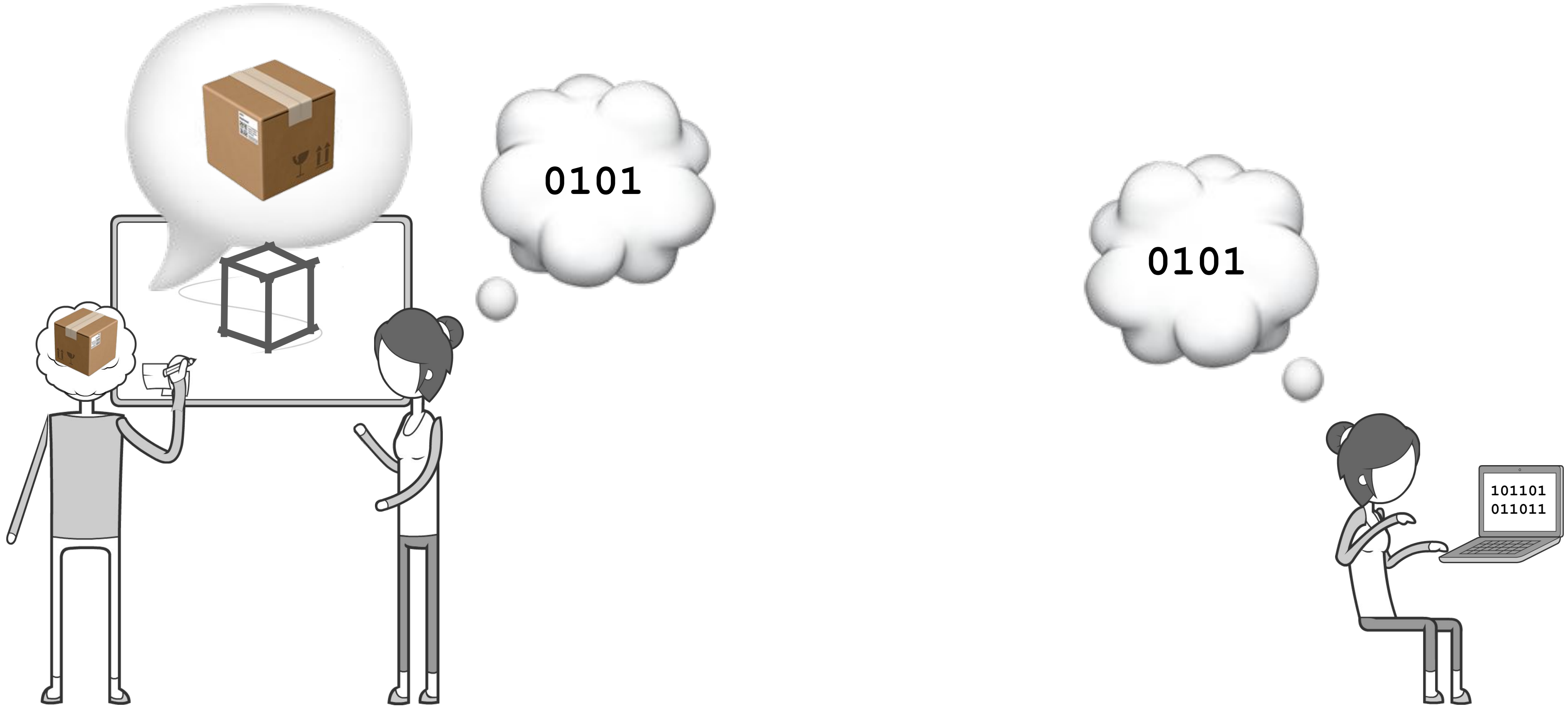



101101
011011

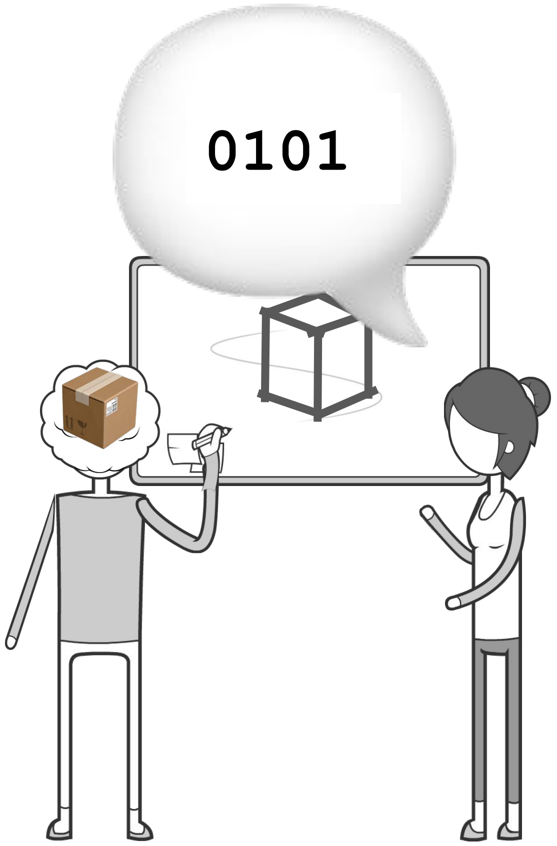


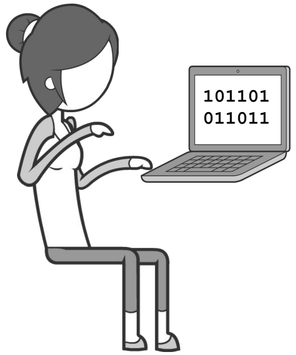


@AL94781 #HeartOfDDD

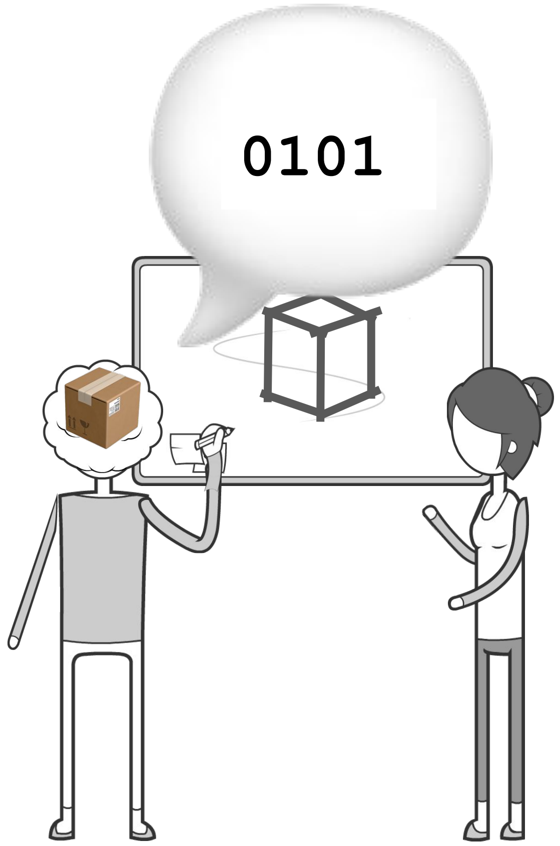


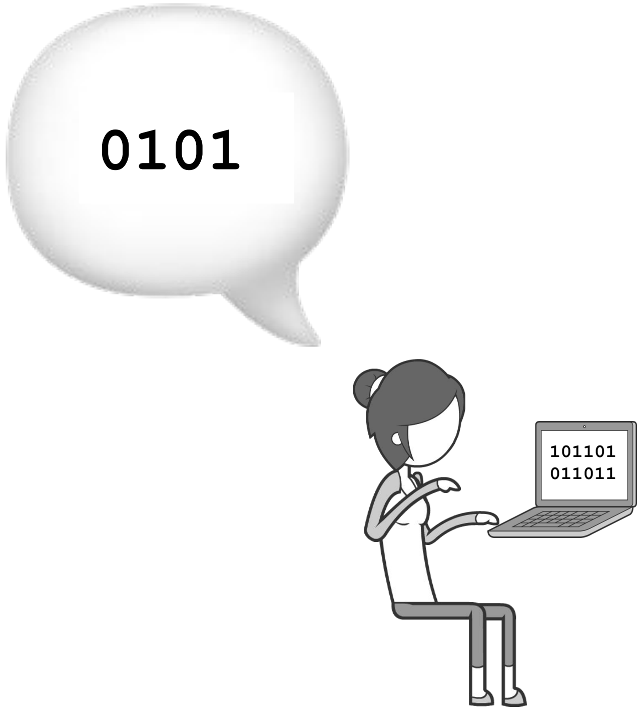
@AL94781 #HeartOfDDD





@AL94781 #HeartOfDDD





@AL94781 #HeartOfDDD

Credit an Account
1011010110
11

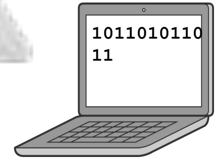
1011010110
11

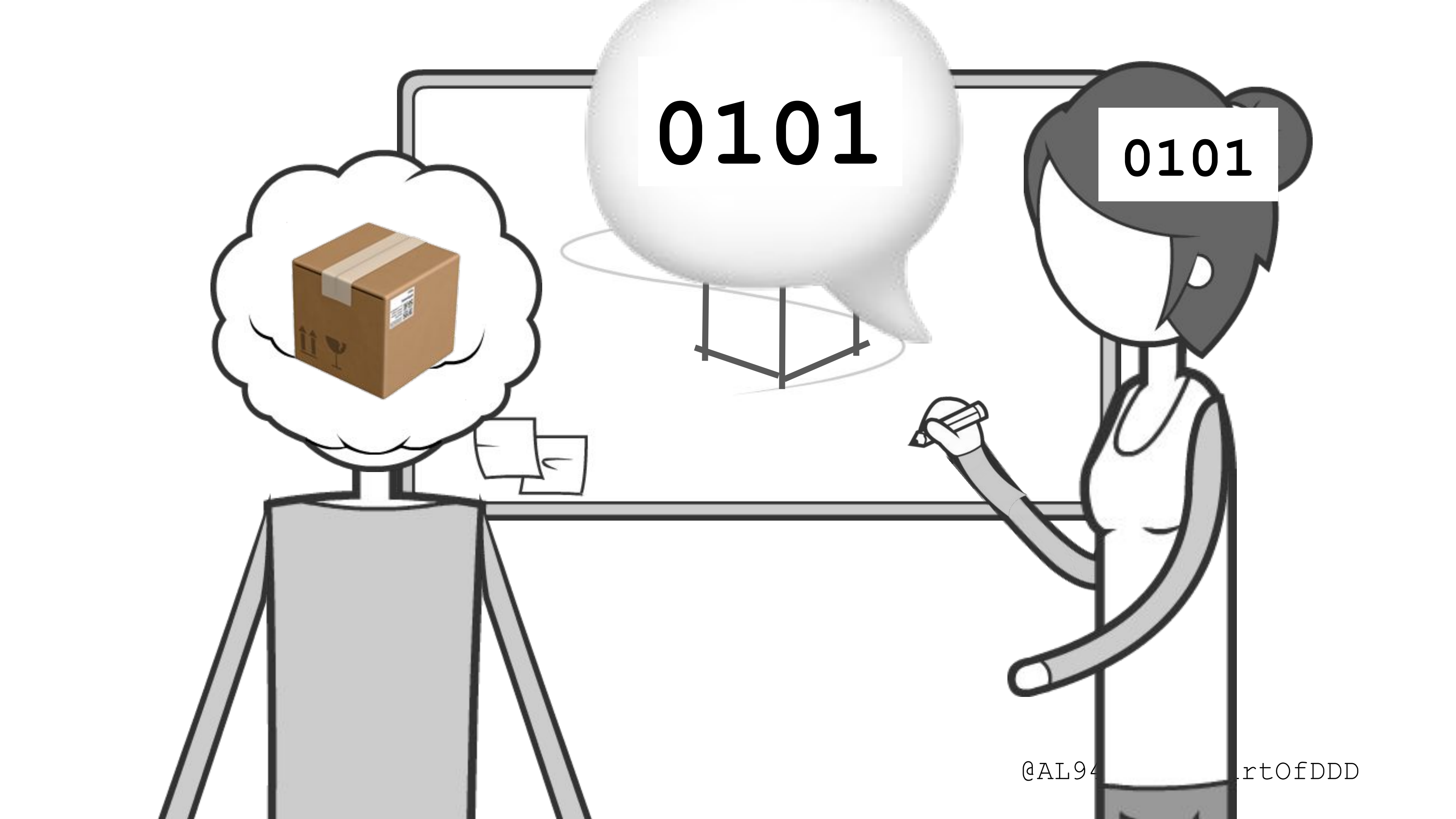
```
balance++
```

Open a new Account
1011010110
11

@AL94781 #HeartOfDDD

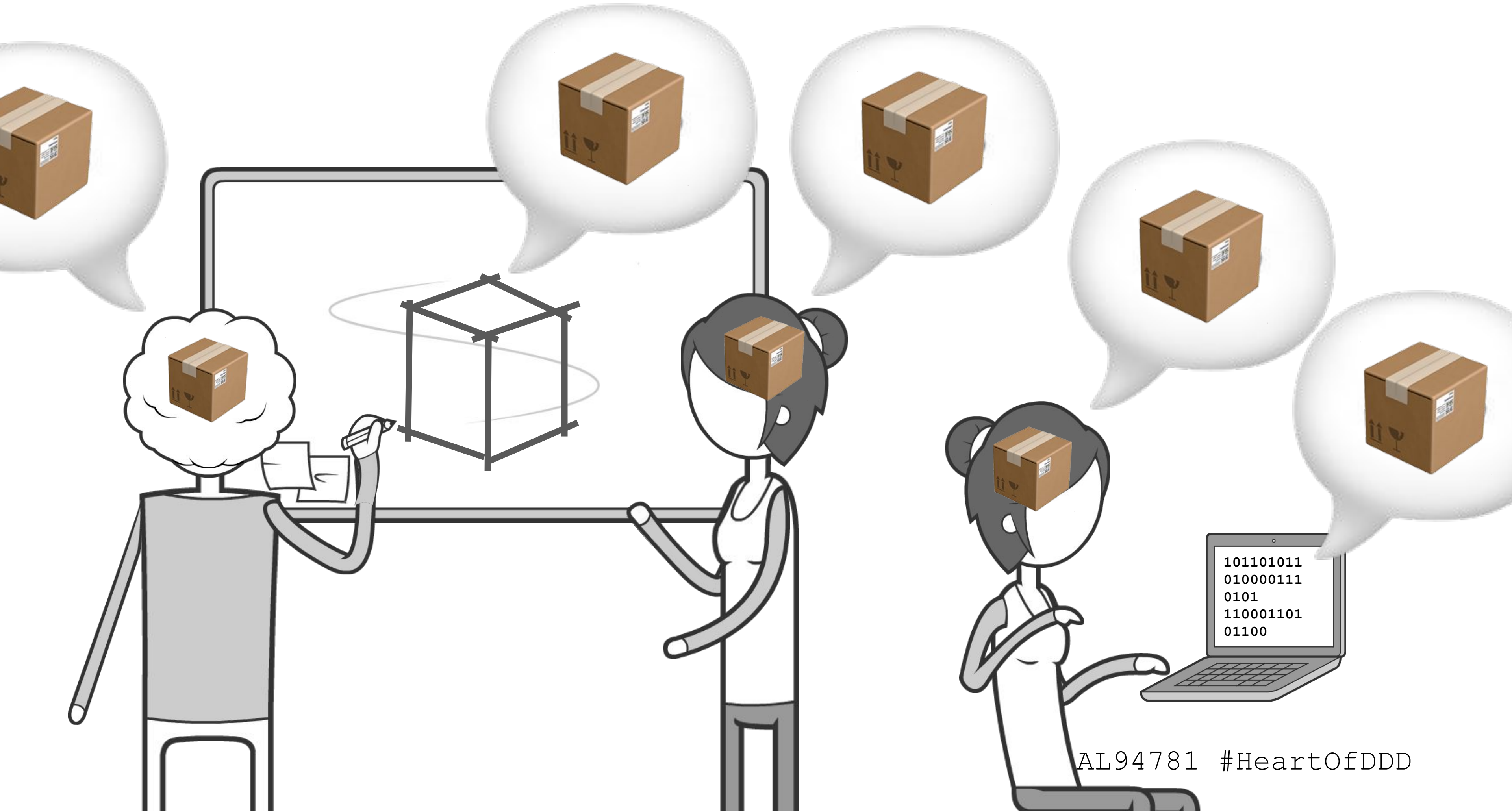
INSERT INTO ACCOUNTS





@AL94 rtOfDDD

@AL94 rtOfDDD



AL94781 #HeartOfDDD

## Ubiquitous language

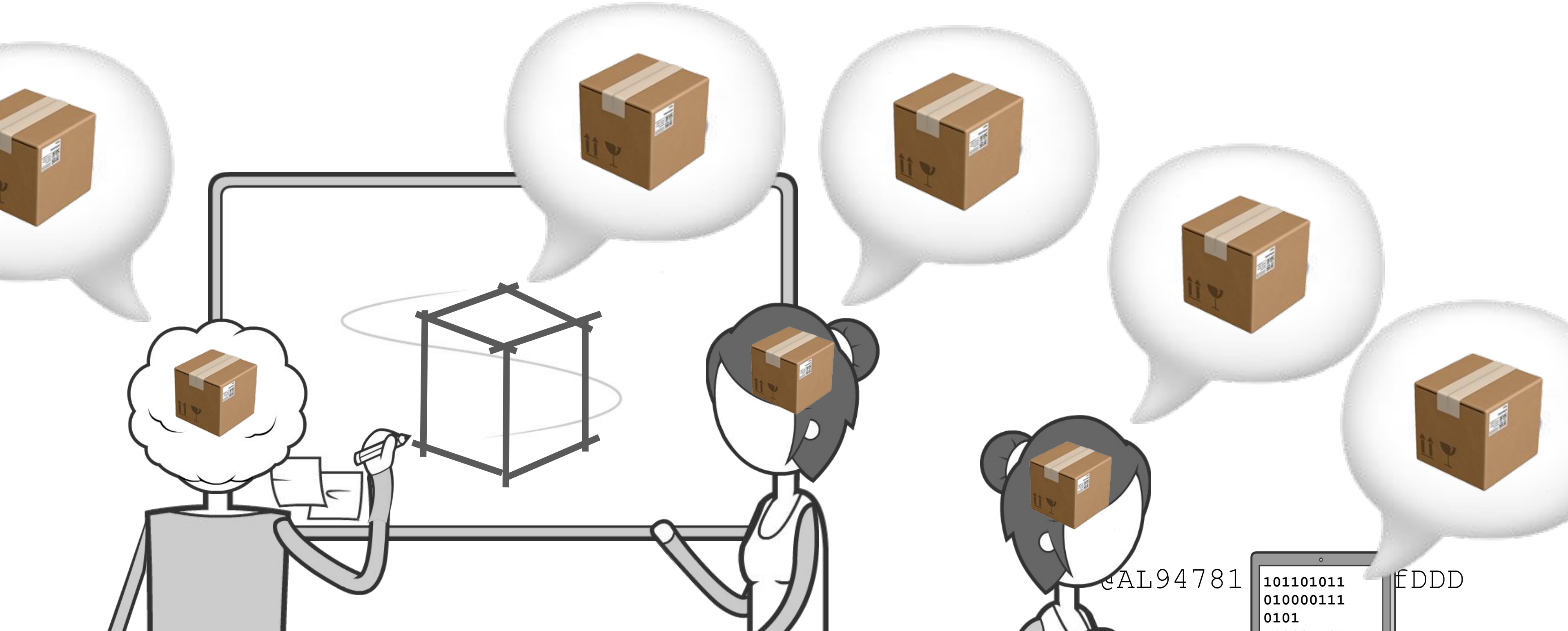
## Let's take a break…

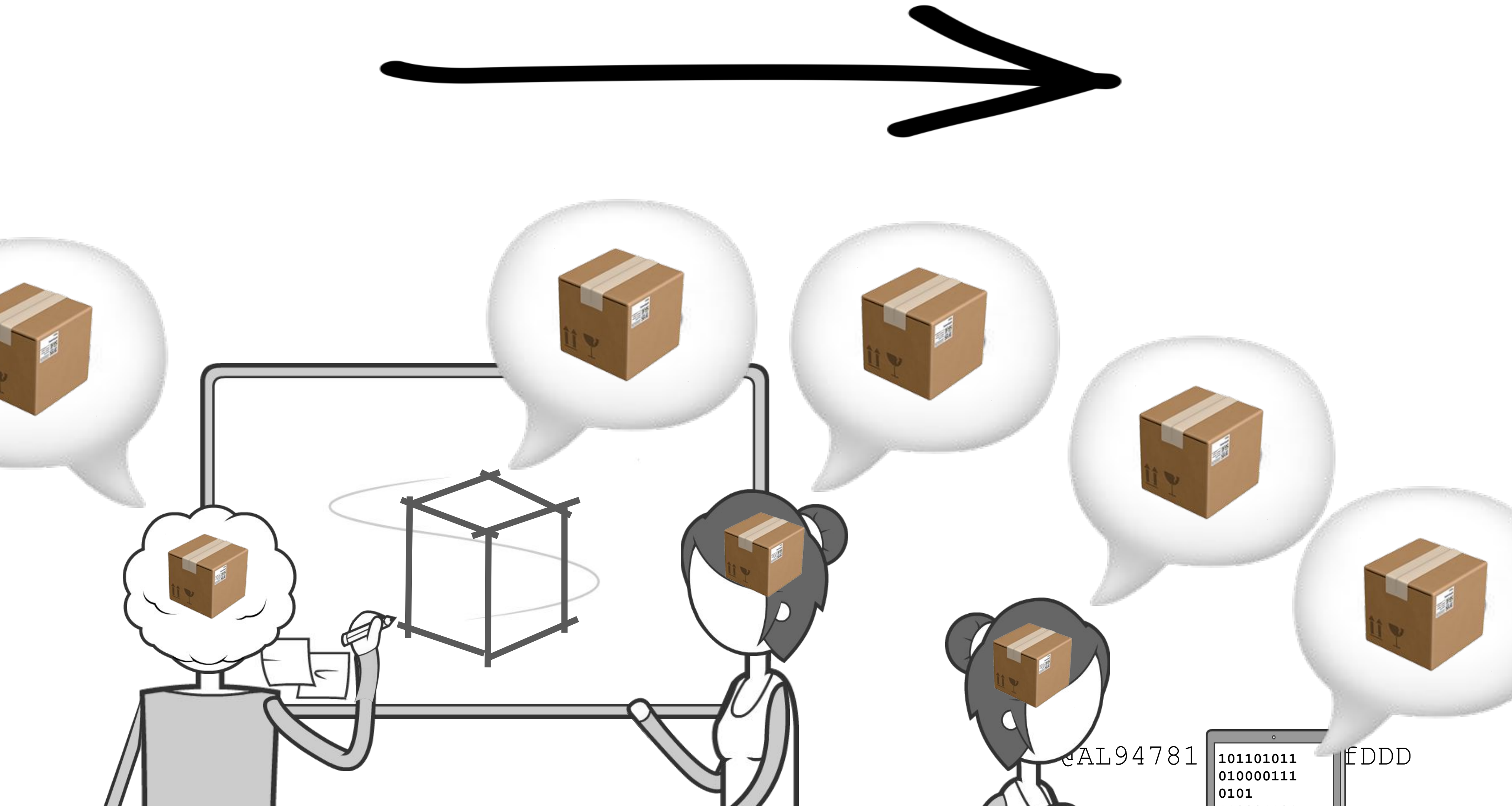
## Ubiquitous language

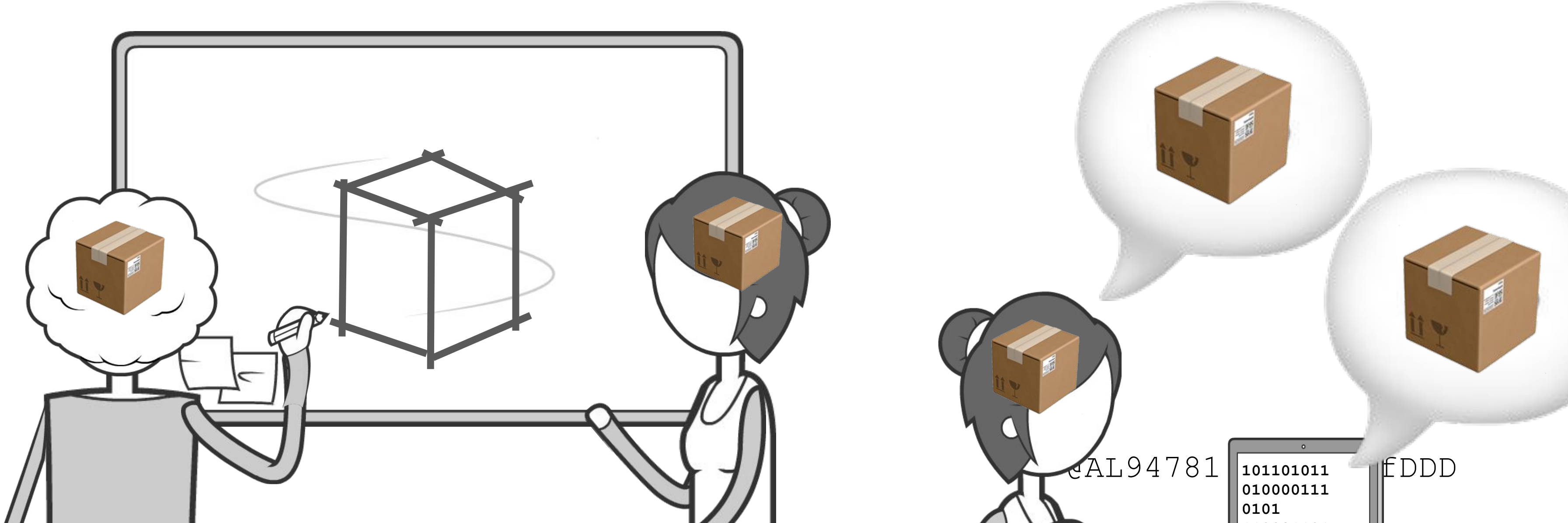
## Hands-on modelers

Ubiquitous language 💬

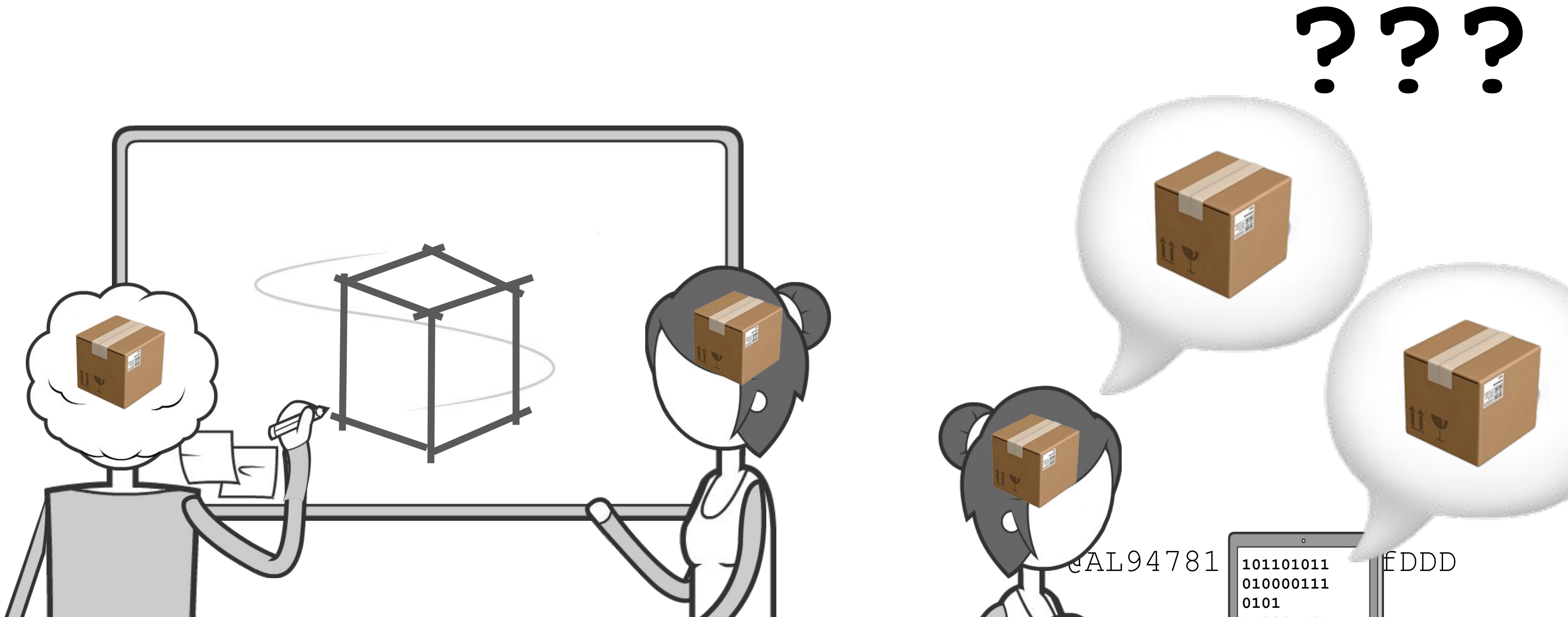
Hands-on modelers 🖐

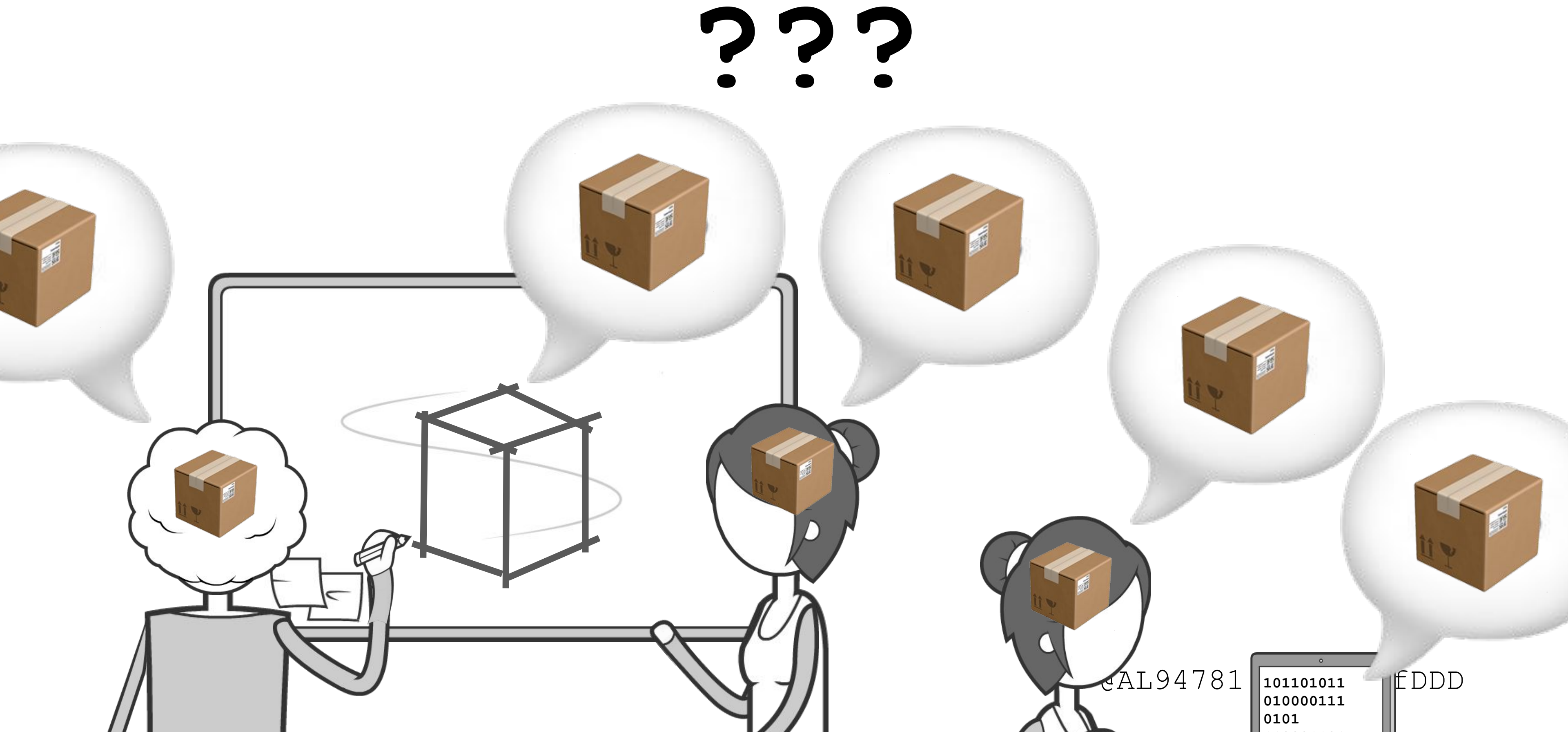

@AL94781
101101011
010000111
0101
fDDD

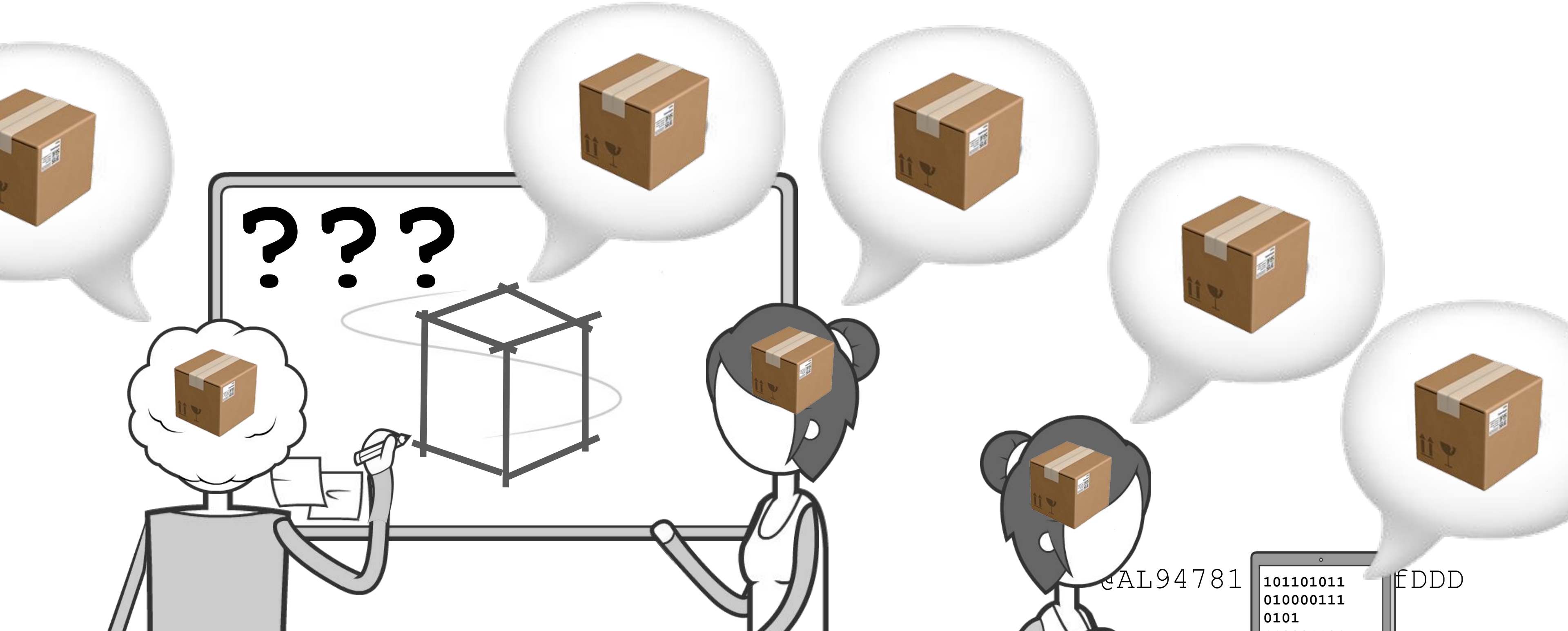

@AL94781
101101011
010000111
0101
fDDD



@AL94781 fDDD


???
@AL94781
101101011
010000111
0101
fDDD

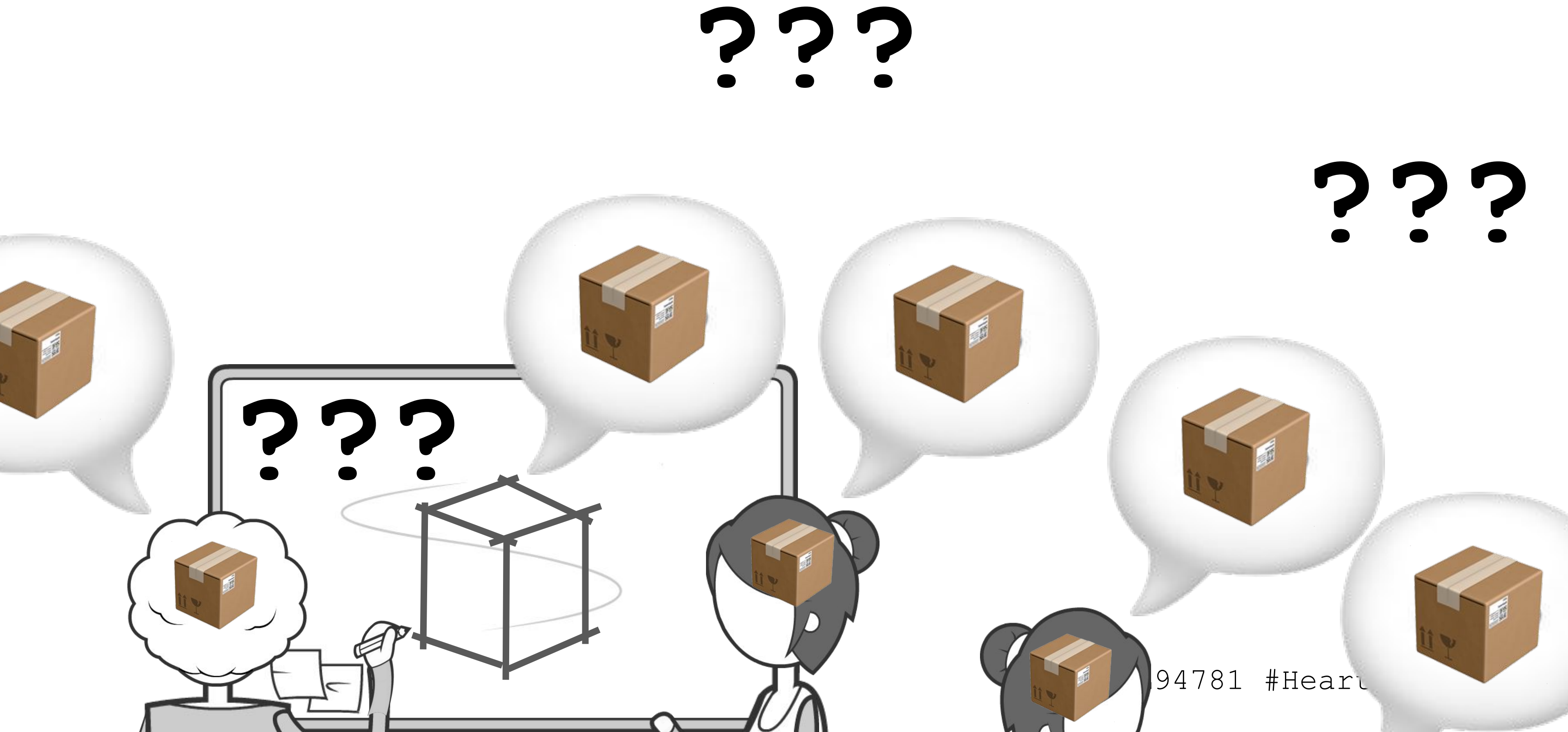

???
@AL94781
101101011
010000111
0101
fDDD

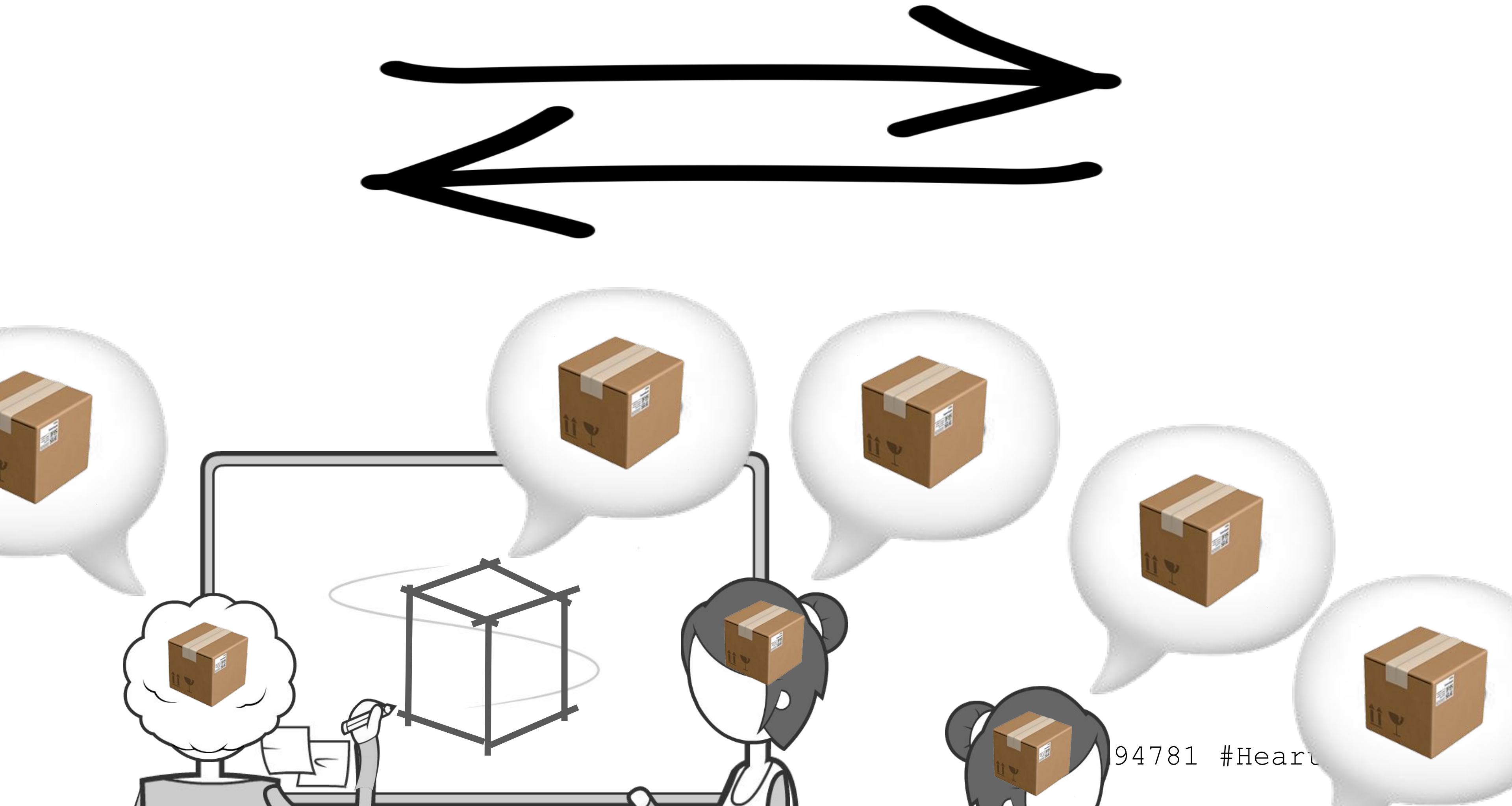


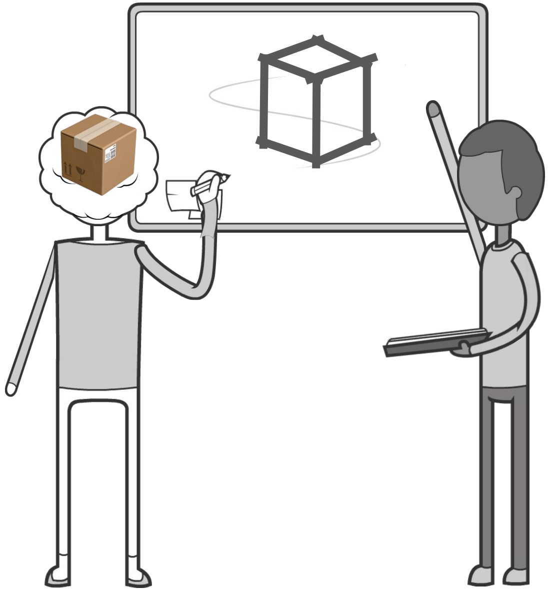
@AL94781 fDDD

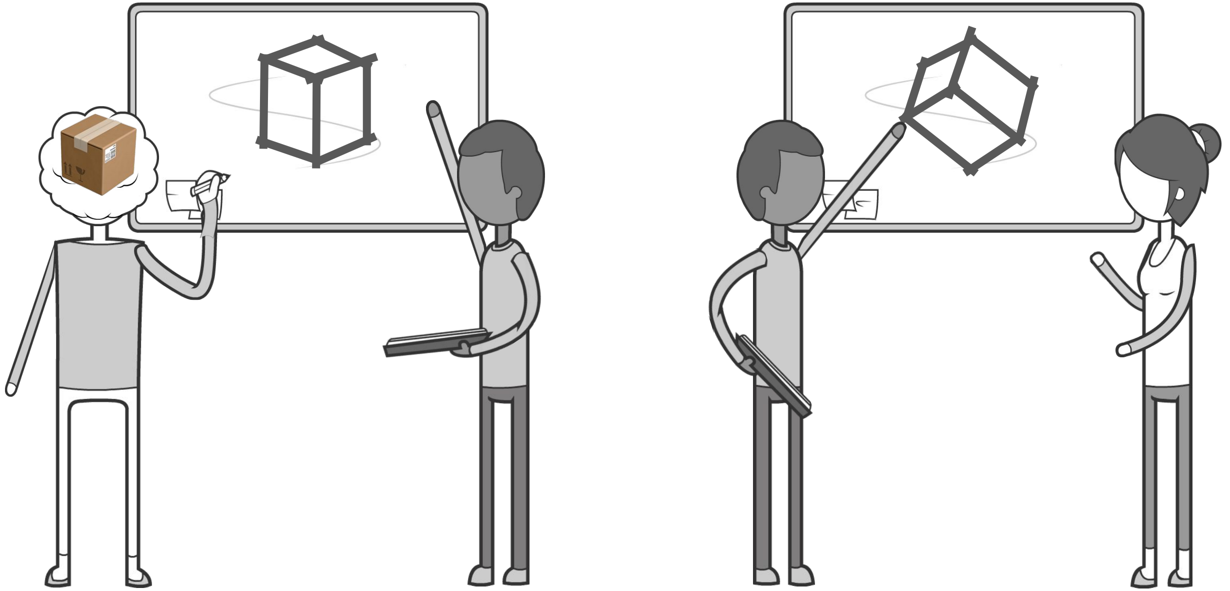
“In having these conversations you both go on a journey. Domain experts are usually not aware of how complex their mental processes are.”

@AL94781 #HeartOfDDD

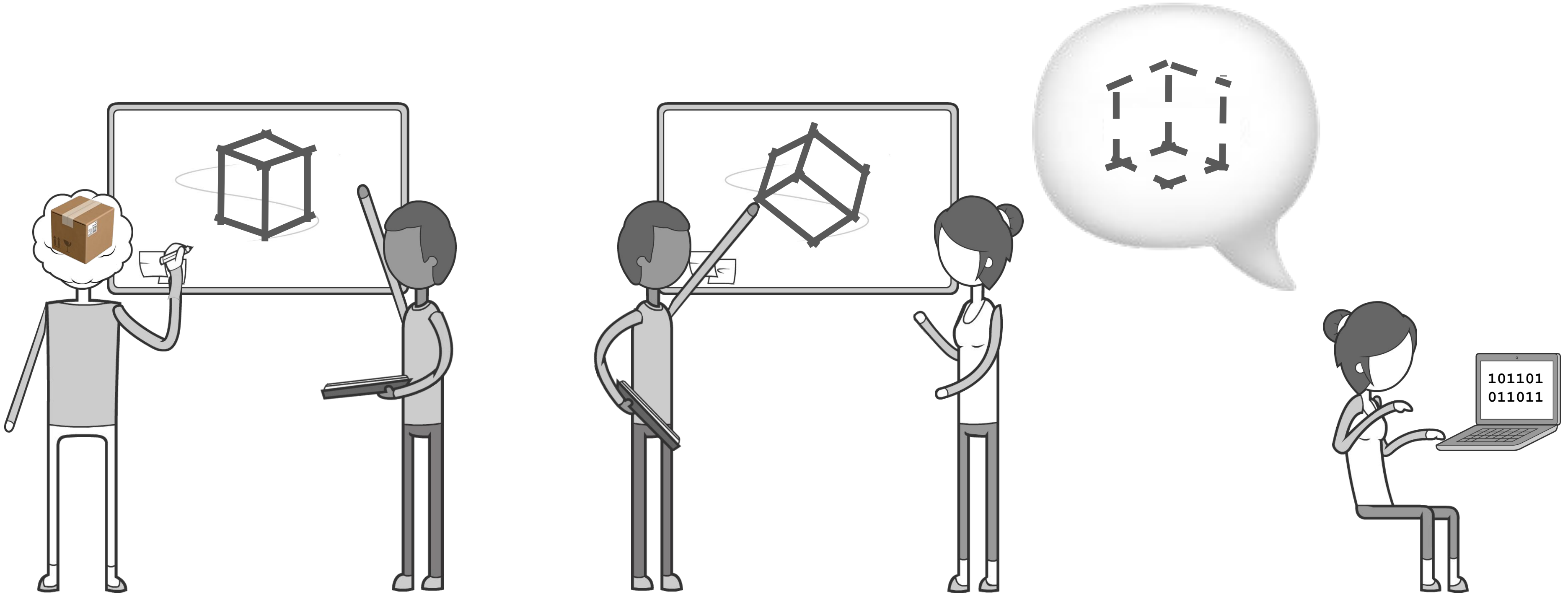

???
???
???
94781 #Hear


94781 #Hear



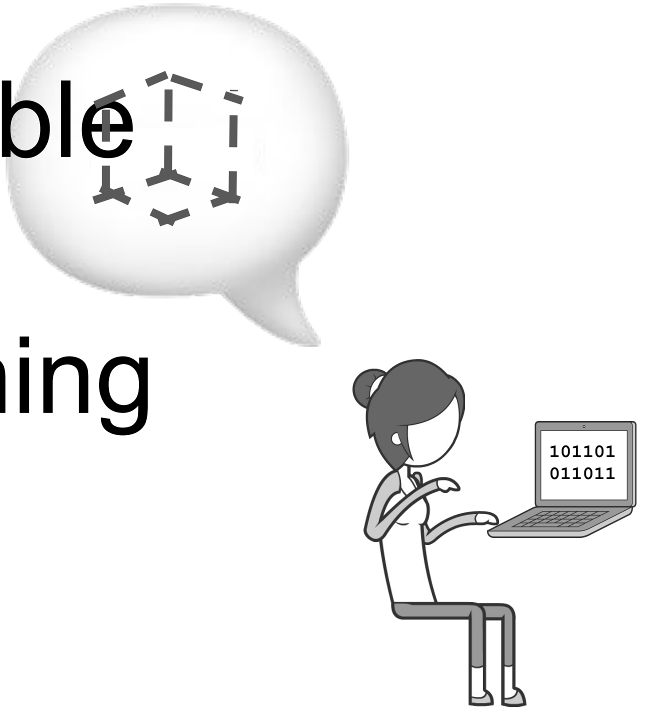


@AL94781 #HeartOfDDD



@AL94781 #HeartOfDDD

“If those who wrote the code don’t feel responsible for the model [...] then the model has nothing to do with the software.”

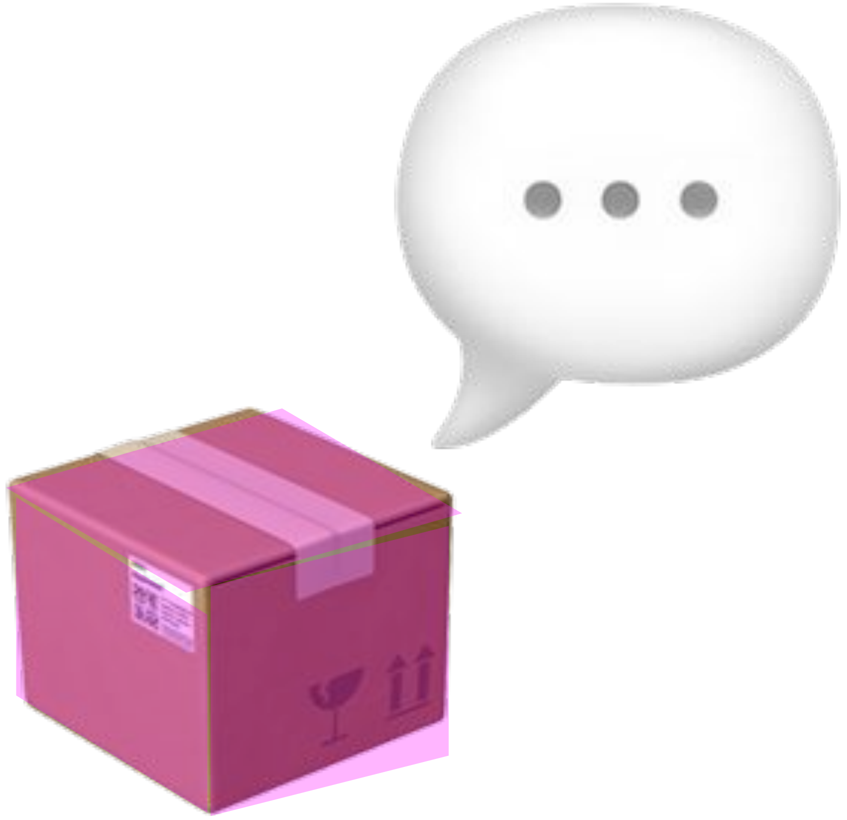


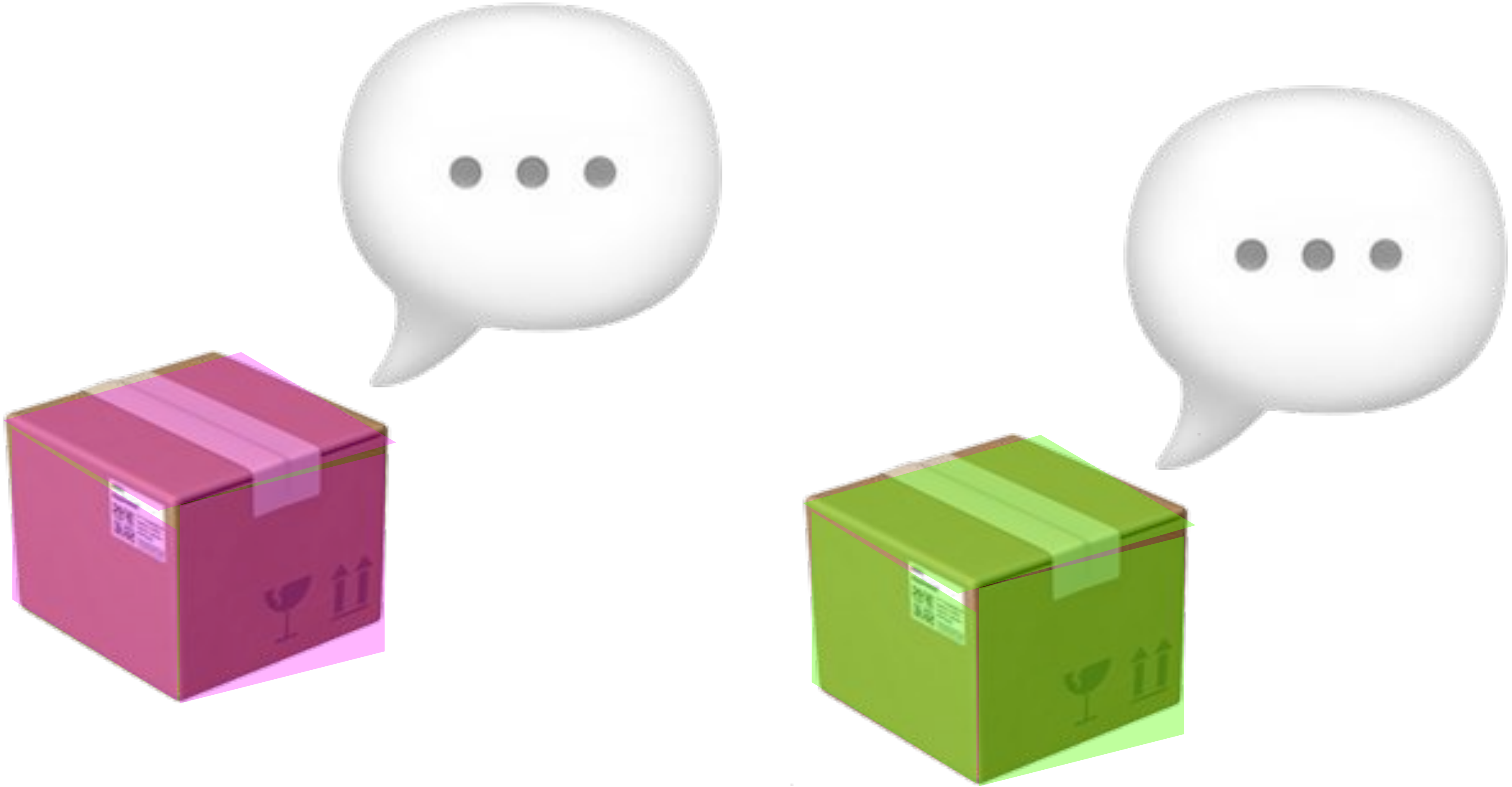
## Hands-on modelers

## Let's take another break...

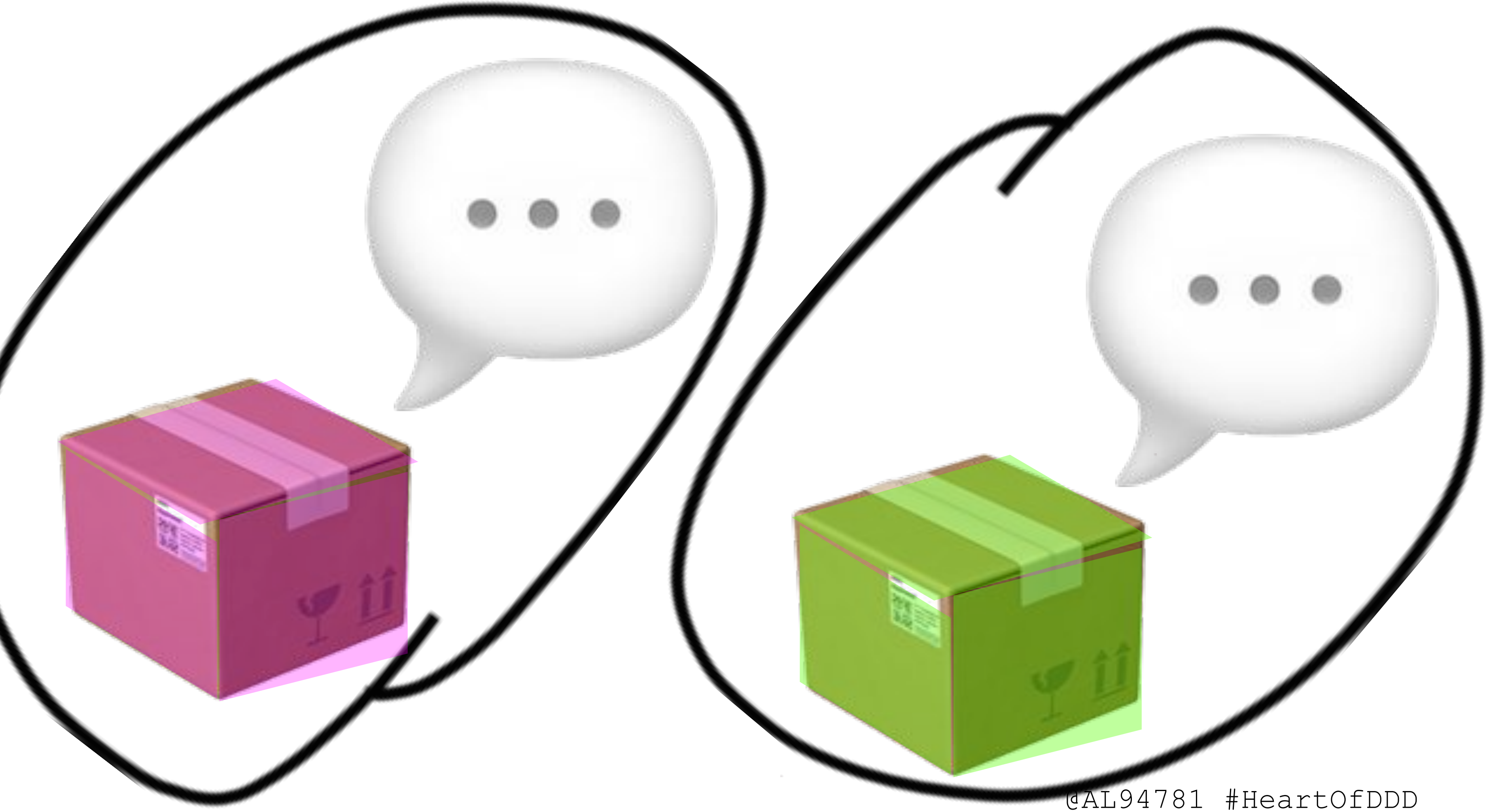
## Ubiquitous language
## Hands-on modelers

Ubiquitous language
Hands-on modelers
Multiple models

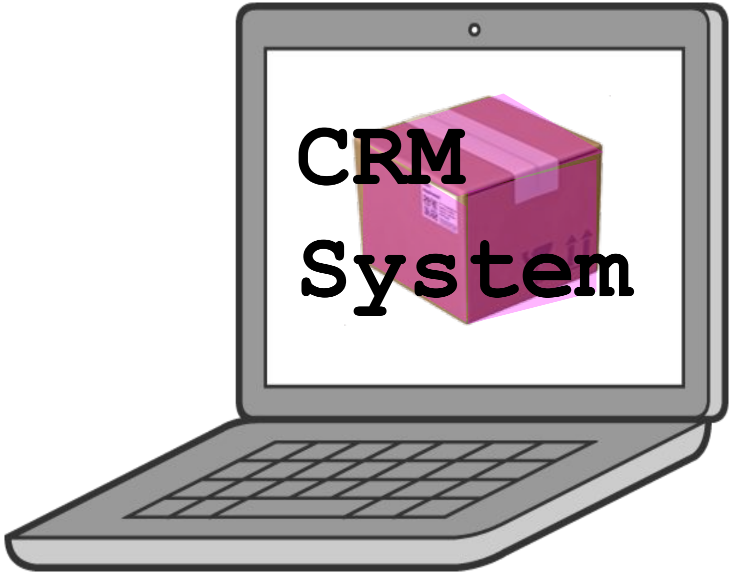




@AL94781 #HeartOfDDD

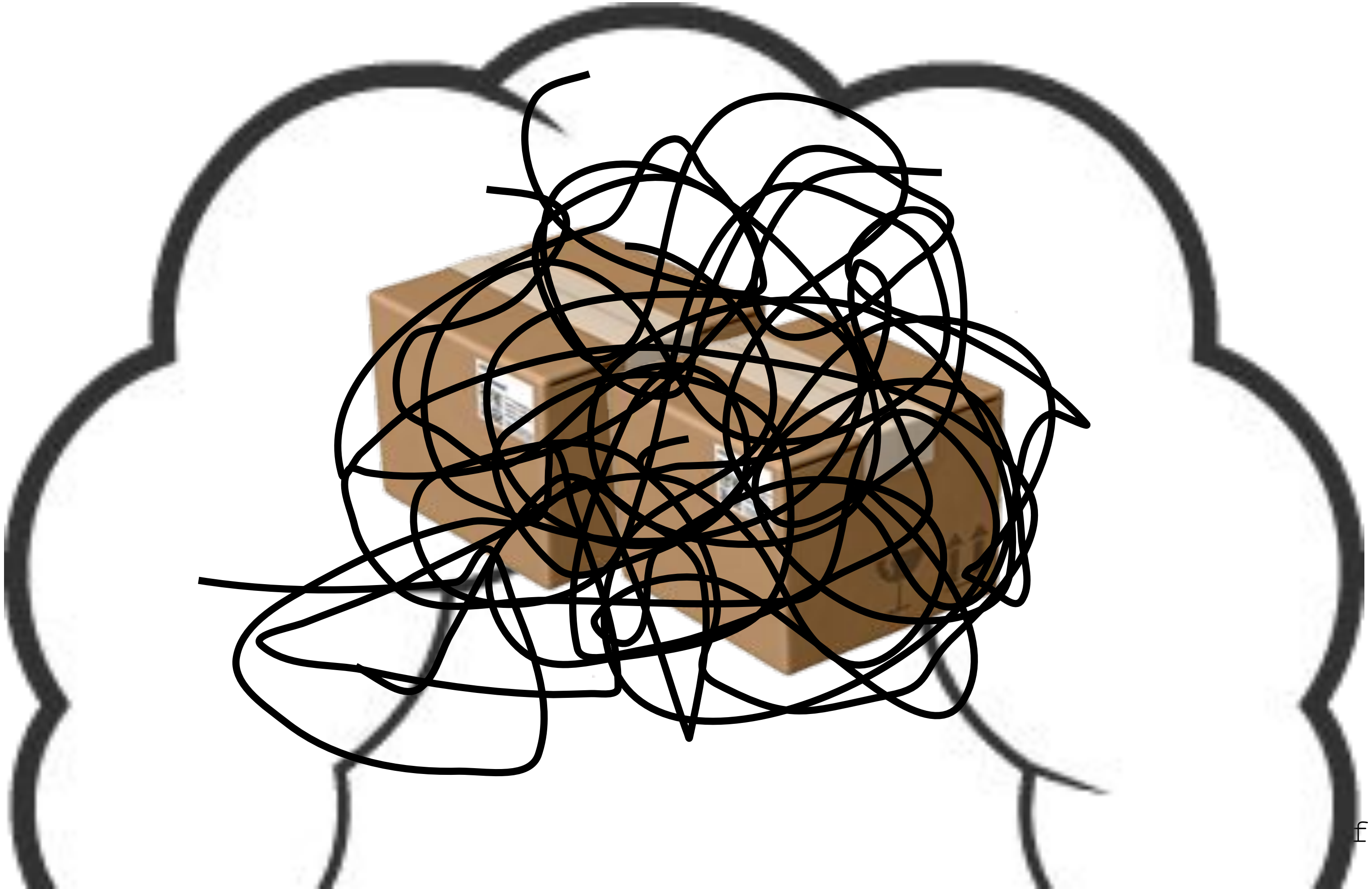


@AL94781 #HeartOfDDD

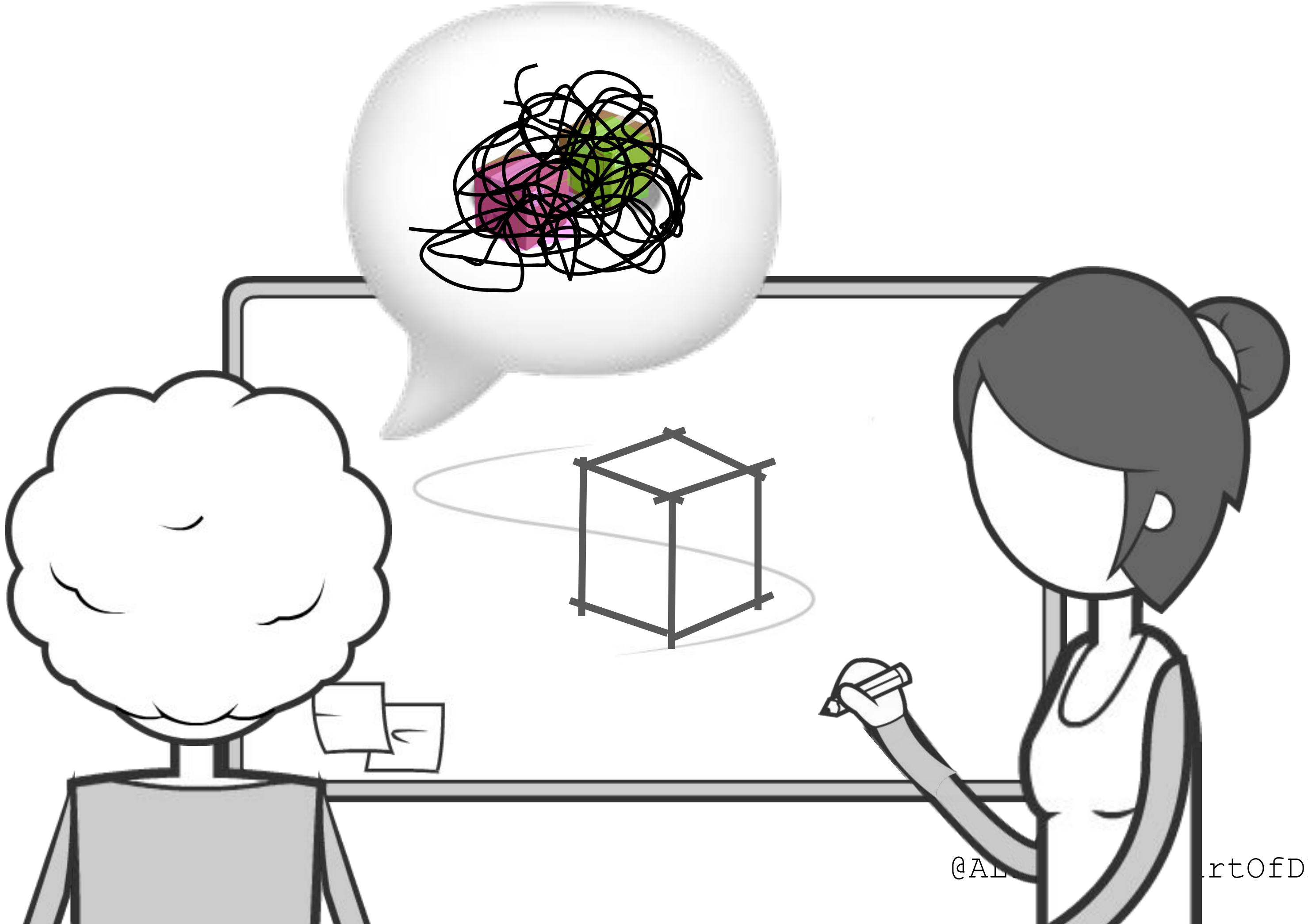

CRM
System

HR
System

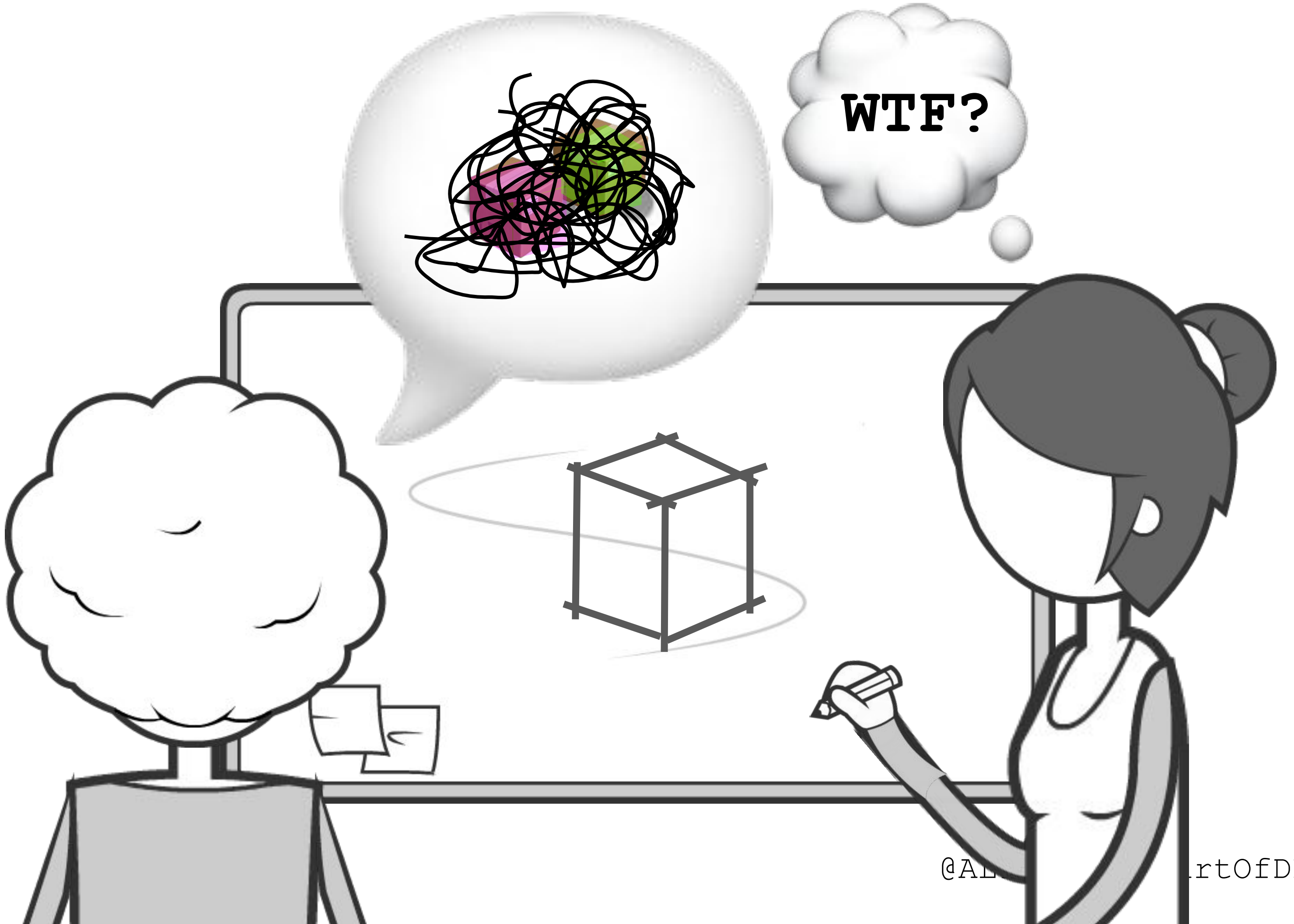
“When the discrepancy is not with an external system, but within the same code base,
it is less likely to be recognised.”



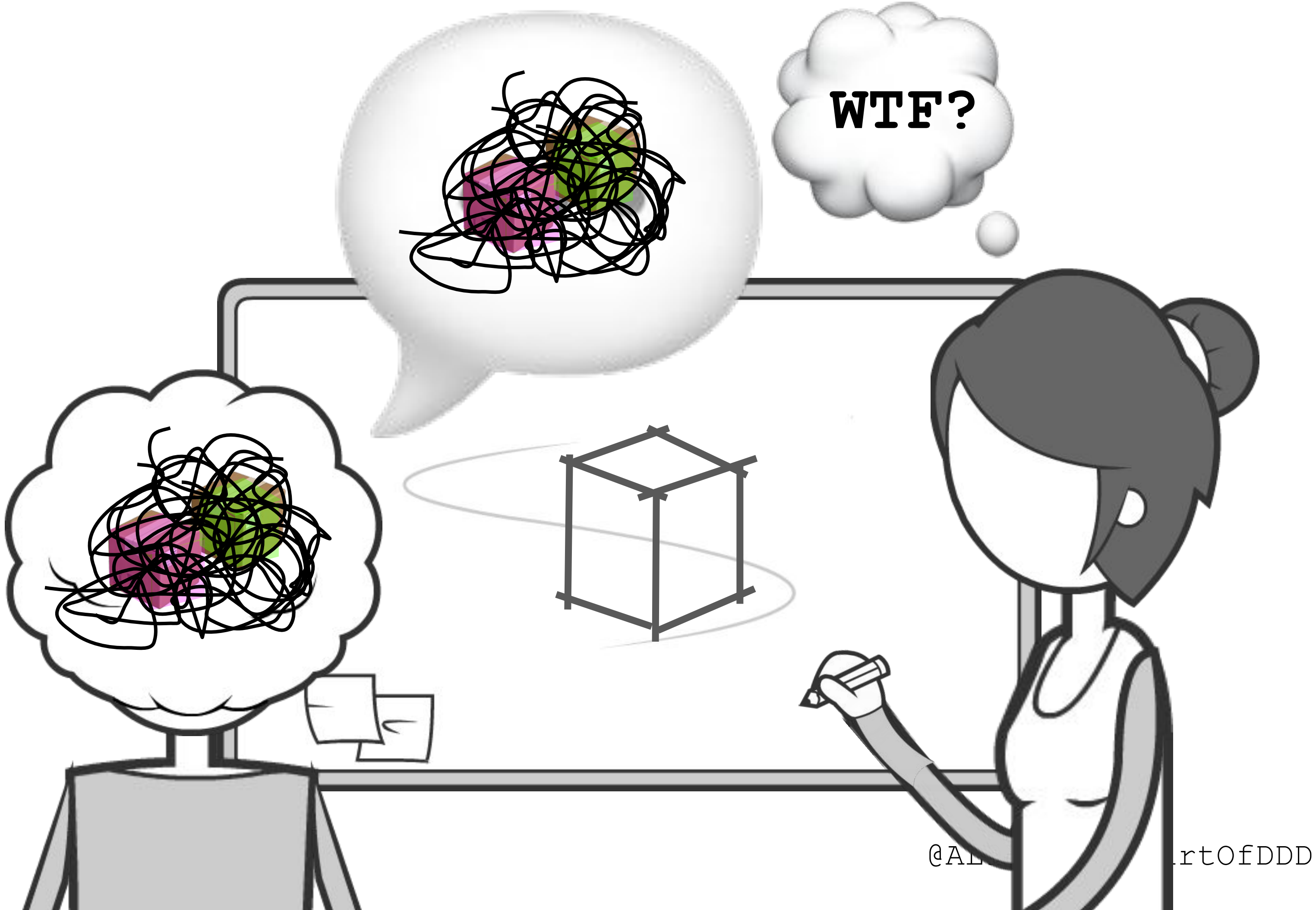
fDDD



@AL rtOfDDD



@Al rtOfDDD



@AL rtOfDDD

# Multiple *distinct* models

Ubiquitous language
Hands-on modelers
Multiple models

## Experiences, thoughts & puzzles...

💬 ✋ 📦📦

## Ubiquitous language

Ubiquitous all the way to the code

# Ubiquitous language

Ubiquitous all the way to the code

You really do need domain experts

## Ubiquitous language

Ubiquitous all the way to the code

You really do need domain experts

# Ubiquitous language

Where to find domain experts?

Ubiquitous all the way to the code

You really do need domain experts

## Ubiquitous language

Where to find domain experts?

Conflicting experts!!!

Hands-on modelers

## Hands-on modelers

What about BAs, QAs, UX?

# Hands-on modelers

What about BAs, QAs, UX?

DDD <3 TDD

# Hands-on modelers

OO & Functional? Yes.

What about BAs, QAs, UX?

DDD <3 TDD

OO & Functional? Yes.

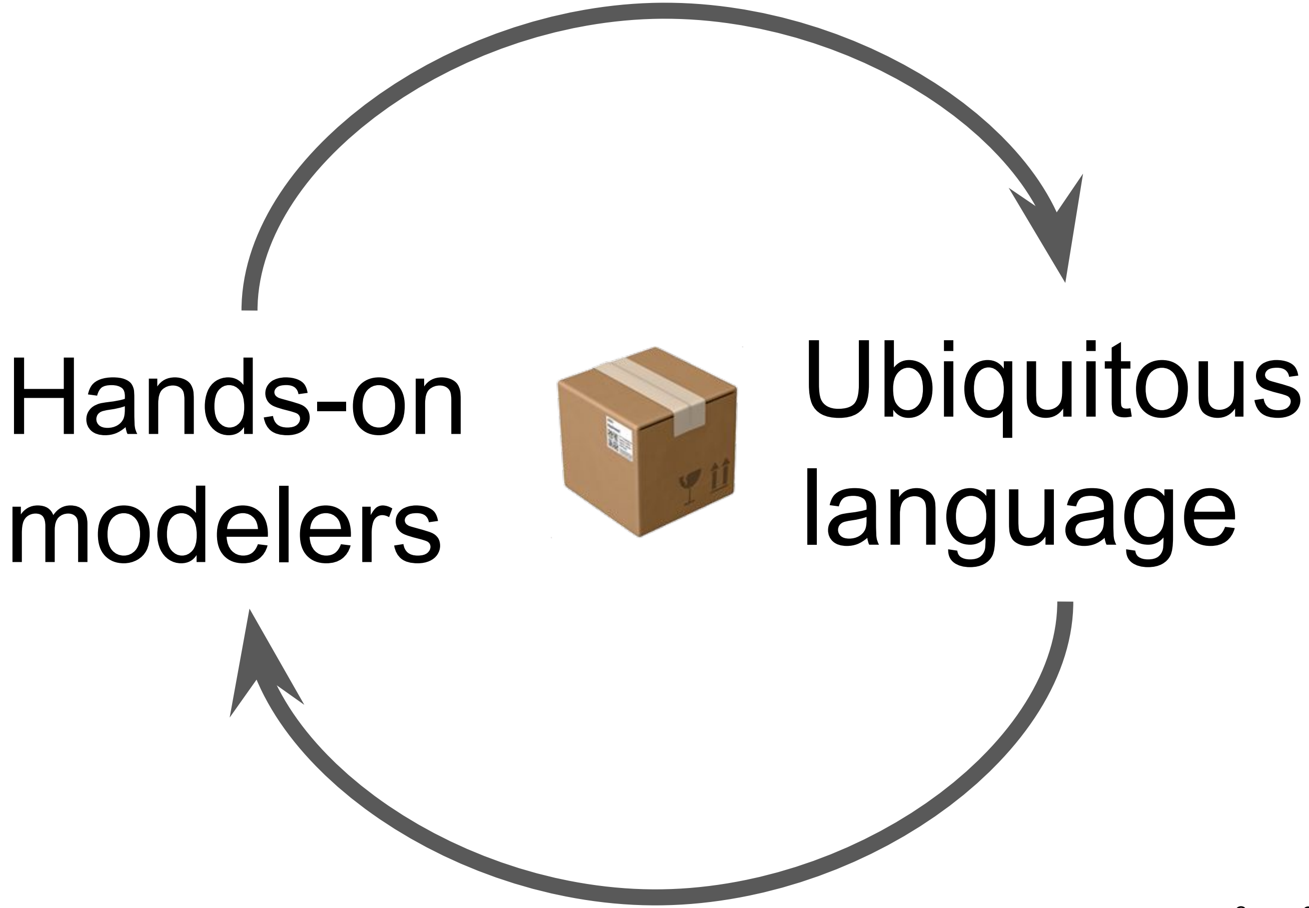
What about BAs, QAs, UX?

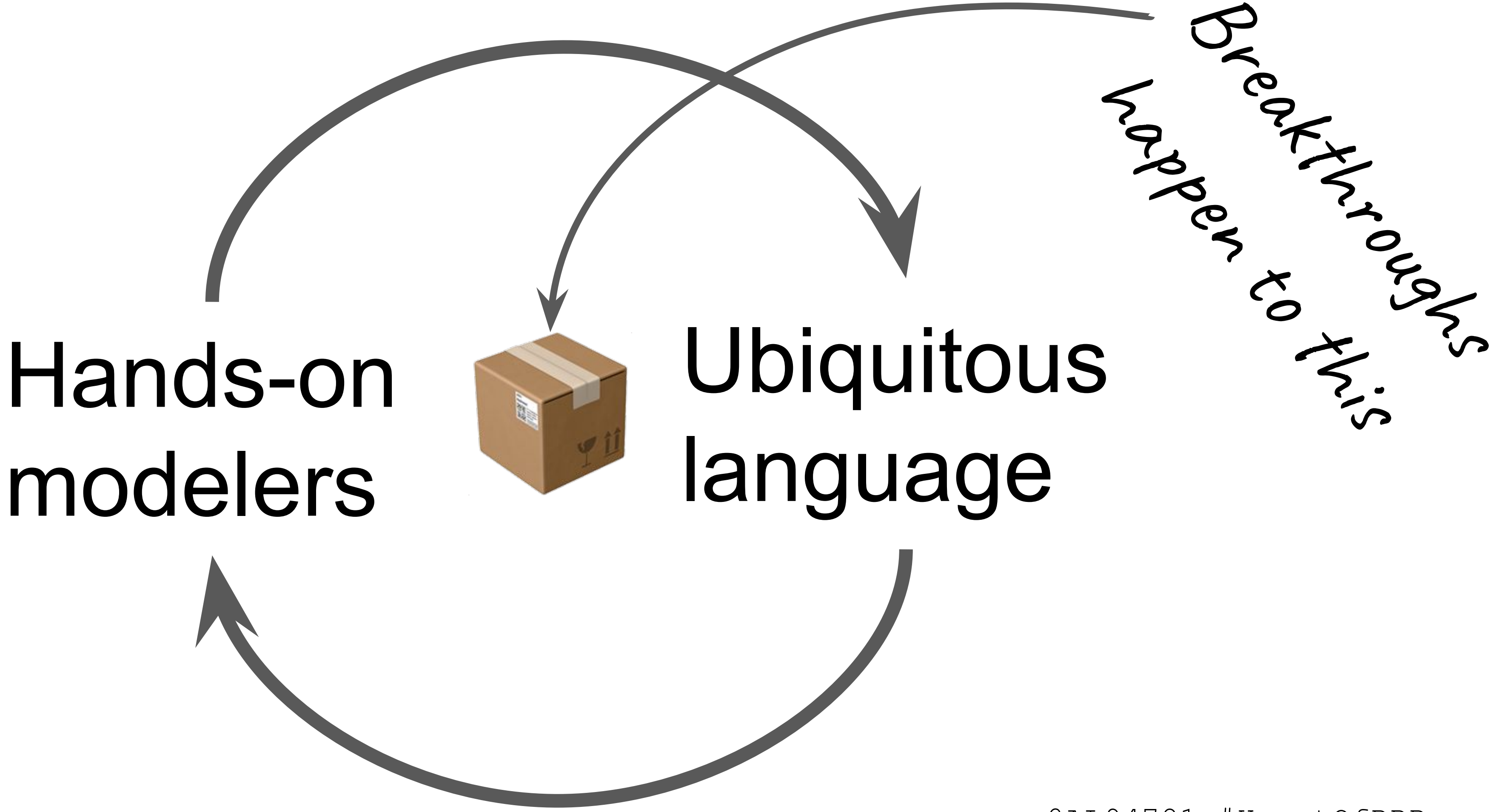
# Hands-on modelers

Procedural & data-centric? No.

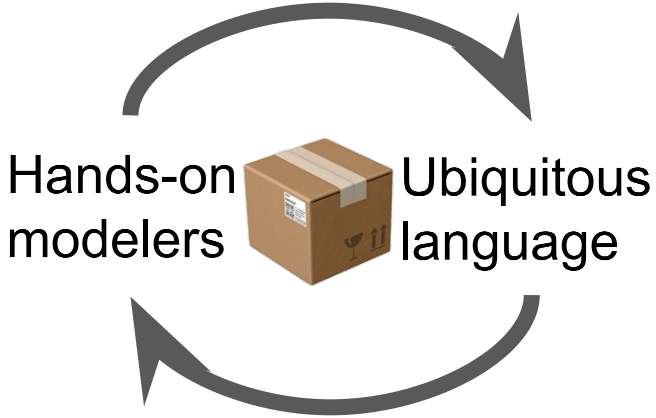
DDD <3 TDD

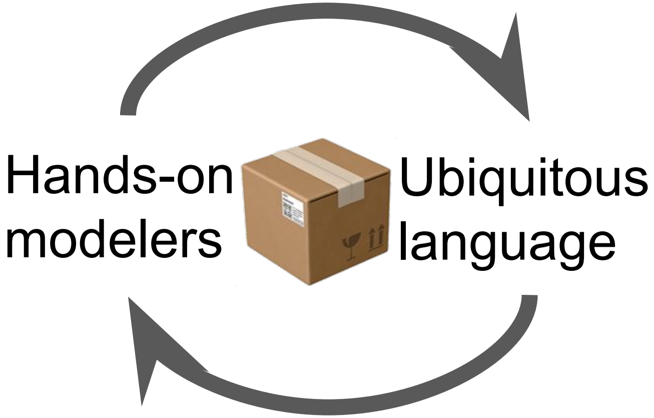
Hands-on modelers 📦 Ubiquitous language

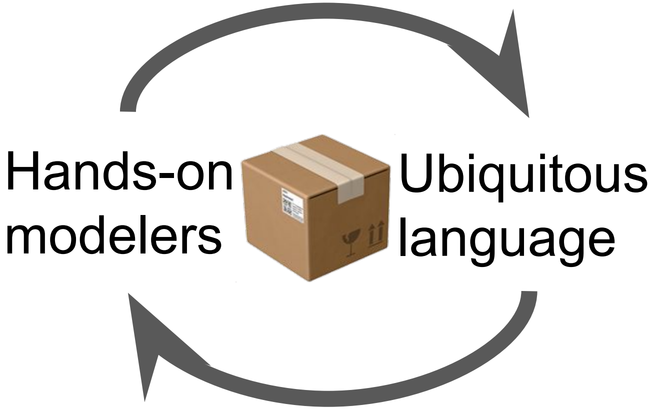

Hands-on modelers
Ubiquitous language



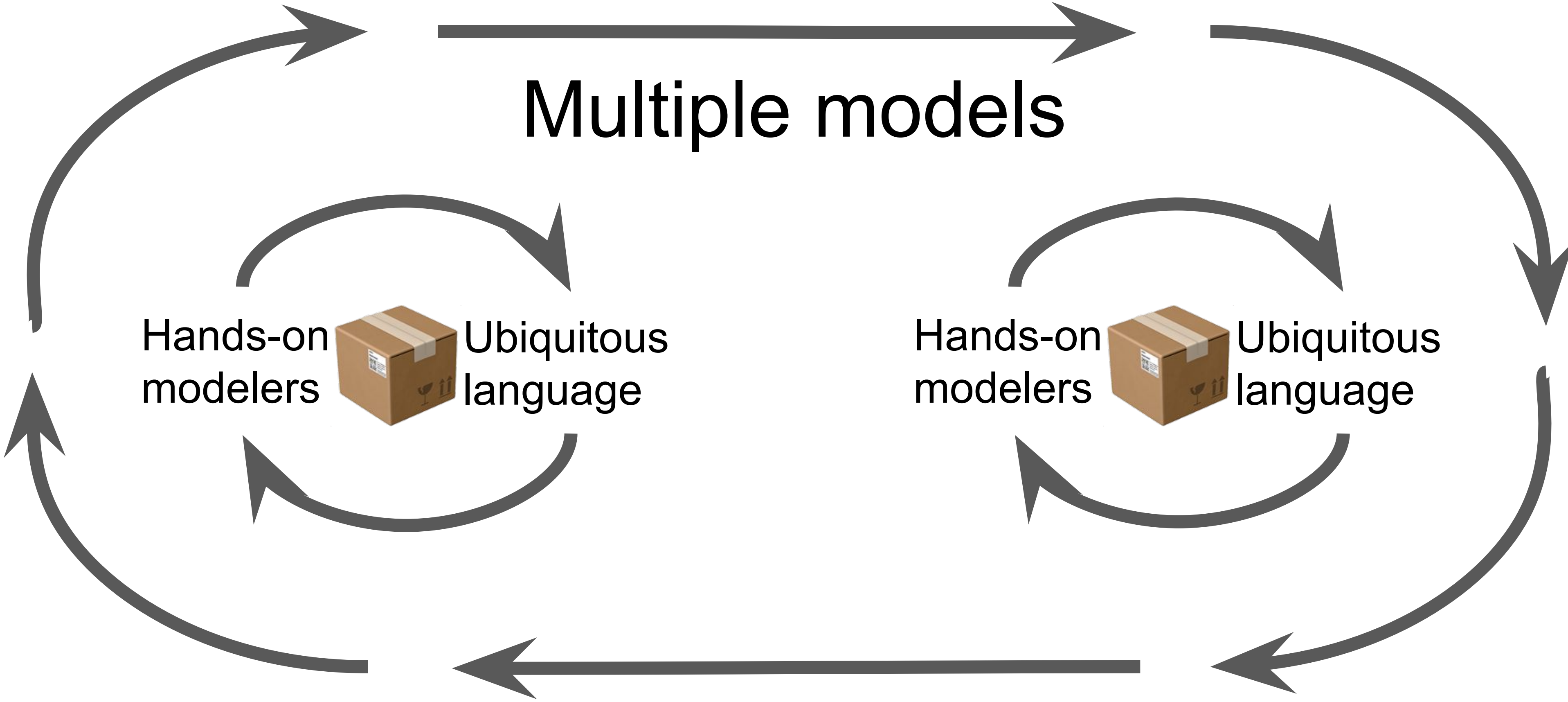
@AL94781 #HeartOfDDD


Hands-on modelers
Ubiquitous language


Hands-on modelers
Ubiquitous language


Hands-on modelers
Ubiquitous language

# Multiple models



Multiple
models

Unlocks Bounded Contexts

Multiple models

BUT what about DRY?

Unlocks Bounded Contexts

Multiple models

BUT what about DRY?

Unlocks Bounded Contexts

## Multiple models

Canonical Data Models? No.

## Multiple models

BUT what about DRY?

Unlocks Bounded Contexts

Nested Bounded Contexts don't exist

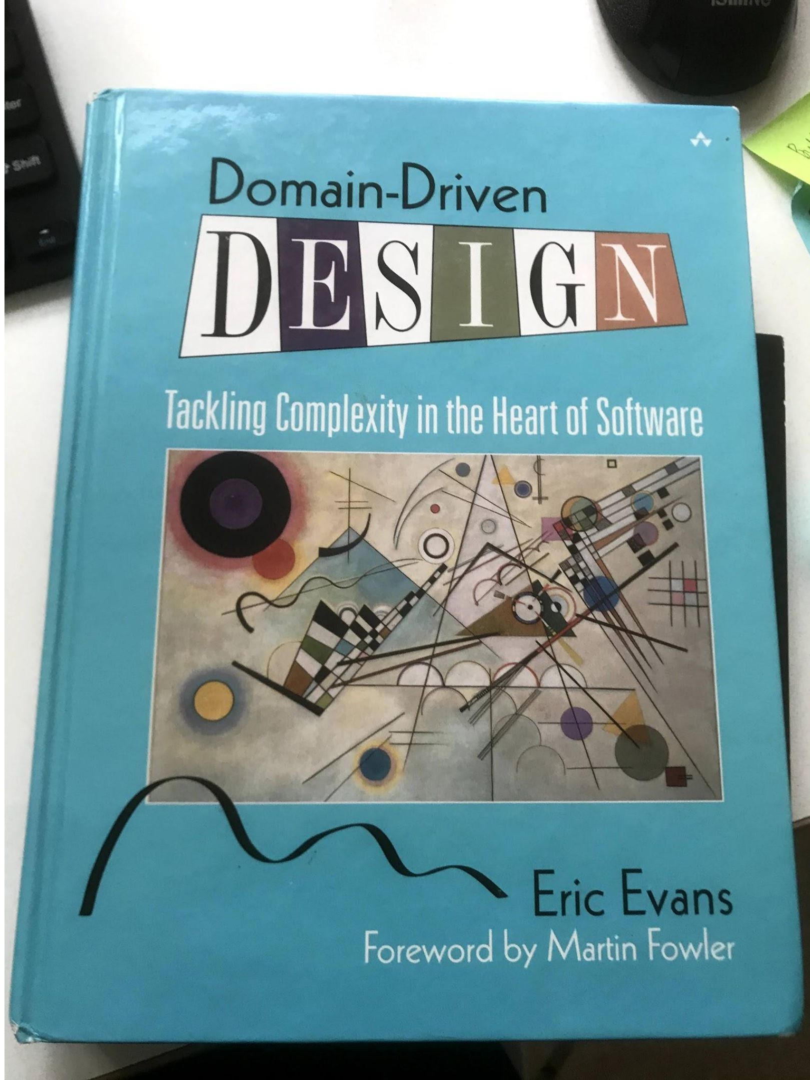
Canonical Data Models? No.

@AL94781 #HeartOfDDD

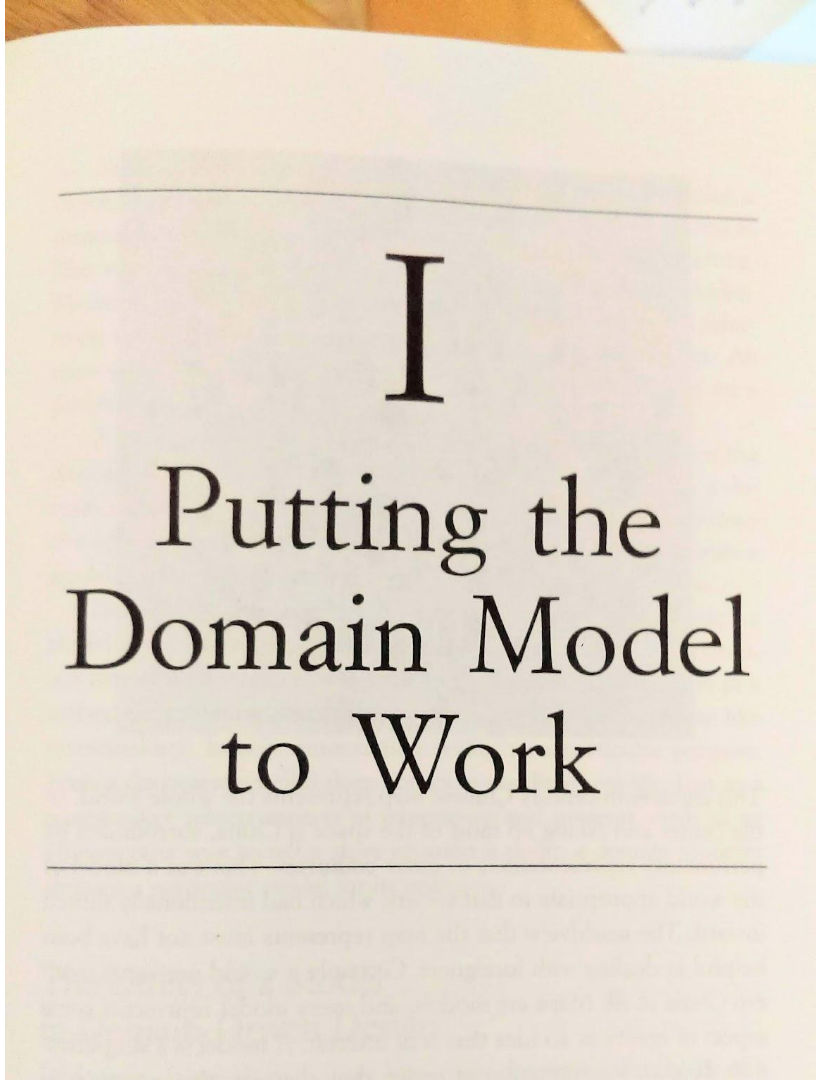
## The Heart of Domain-Driven Design?

## How do we get from here ...
## to Microservices?

Back to the book...


Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler

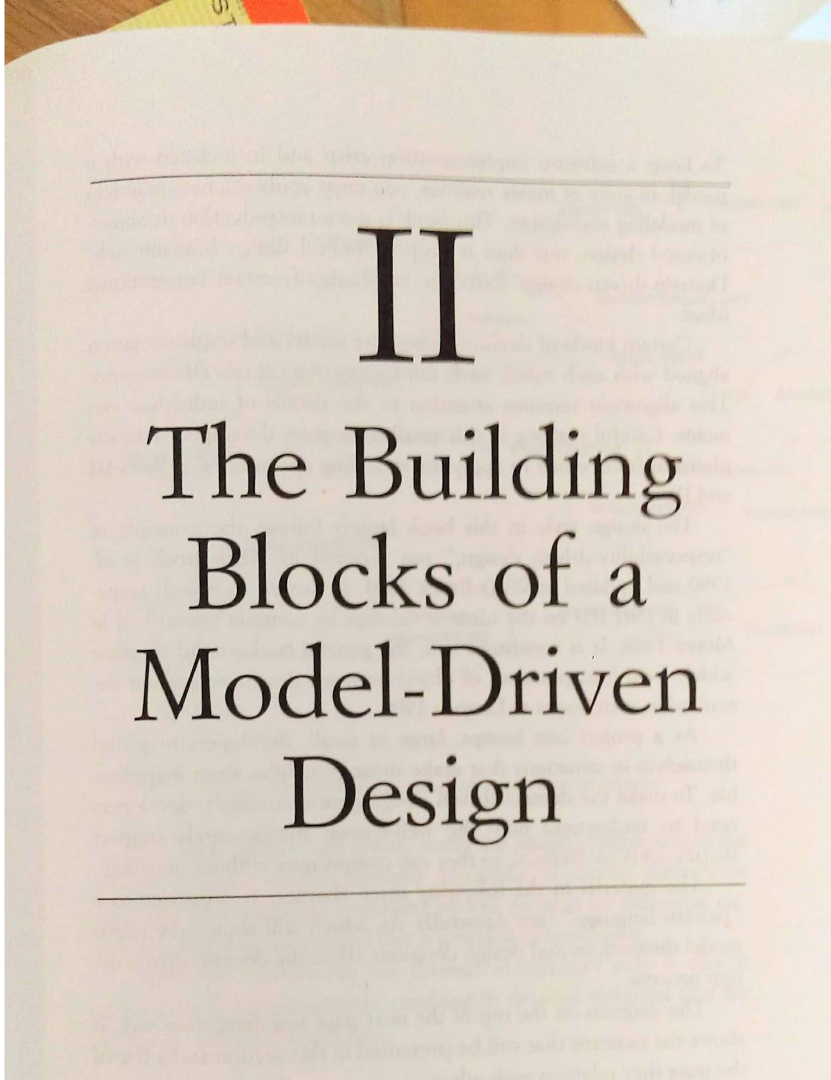
Back to the book...



I

Putting the Domain Model to Work

Back to the book...



II

The Building Blocks of a Model-Driven Design

## Entities

Warning:

DDD Entities

& Database Entities are different!

The goal of Entities is to focus on Identity

## Entities *and Identity*

## Entities and Identity:
## An example

## *Modeling* Entities

- *Make identity primary to the definition*
- *Keep class definition simple, focused on life-cycle*
- *Remove behaviour and attributes to other, associated objects*
- *Coordinate the operations of these owned objects*

Phew! That's Entities. Let's take another break...

“Beyond identity issues, Entities tend to fulfil their responsibilities by co-ordinating the operations of objects they own”

“Beyond identity issues, Entities tend to fulfil their responsibilities by co-ordinating the operations of objects they own”

## Value Objects

“Treat the Value Object as immutable. Don’t give it any identity.”

*Immutable, identity-free*

## Value Objects

## Using Value Objects

## Value Objects:
### *An Example*

## Entity or Value Object?

Phew!
Well done us, we're almost there...

## Aggregates

## Aggregates *are made up*

## Aggregates *vs* *the Real World*

“Minimalist design of associations helps simplify traversal and limit the explosion of relationships ...”

"... but most business domains are so interconnected that we still end up tracing long, deep paths through object references."

## Aggregates and the problem of changing connected things in our system

Aggregates *maintain invariants between groups of associated objects*

“The primary job of an AGGREGATE is to guarantee the consistency of changes to objects in a model with complex associations.”

## Aggregates vs Database transactions?

Aggregates are made up - they don't come directly from the real world

Aggregates *are a set of rules for implementing transactions*

## *Modeling* Aggregates

- *An object cluster, treated as a unit for data changes*

## *Modeling* Aggregates

- An object cluster, treated as a unit for data changes
- Each has a root and a boundary
- The root is a single, specific ENTITY
- The root is the only part that outside objects are allowed to hold references to

## Congratulations!

## Entities *model identity & own relationships*

Entities *model identity & own relationships,*
Value Objects *hold immutable attributes*

Entities *model identity & own relationships,* Value Objects *hold immutable attributes, &* Aggregates *maintain invariants across clusters*

## Aggregates are Microservices?

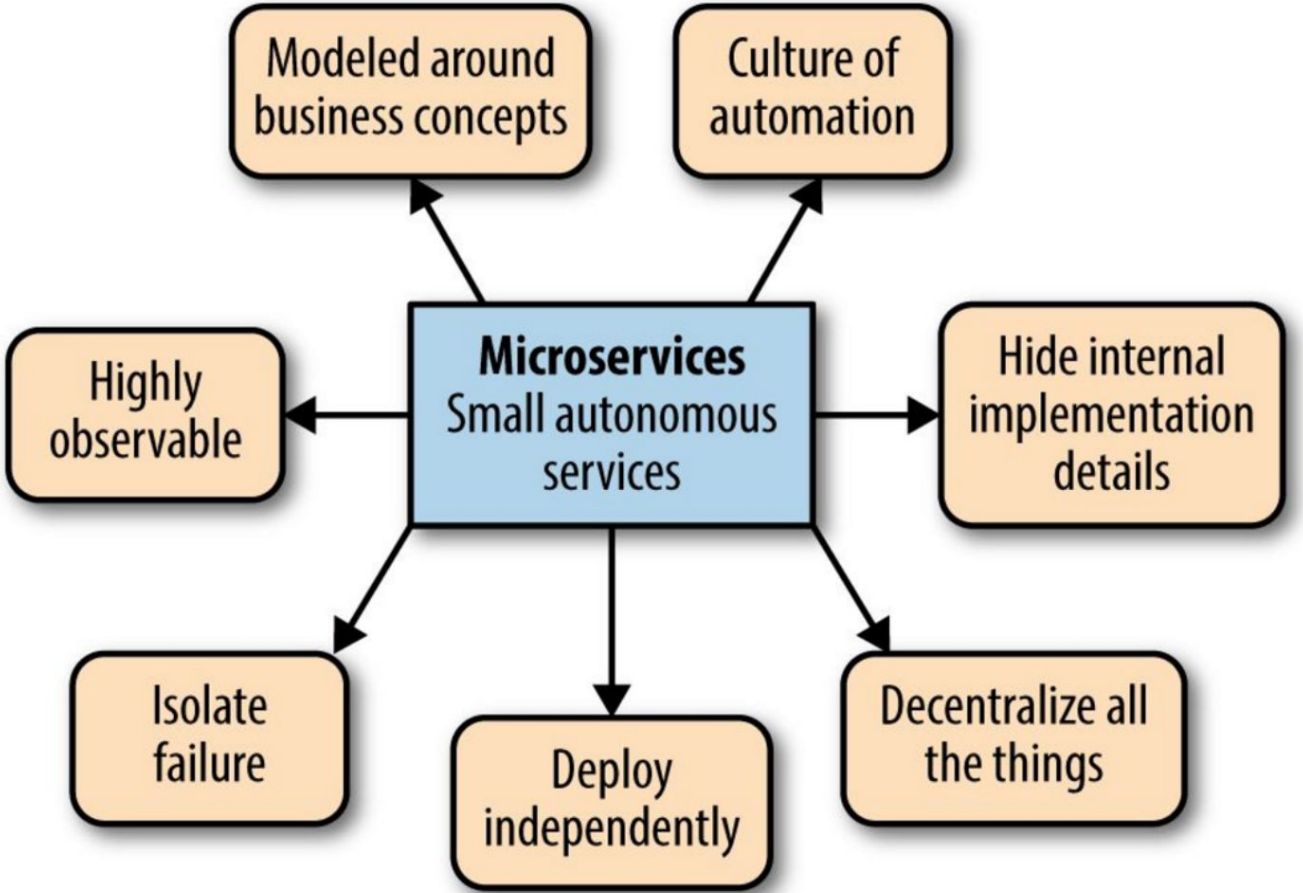
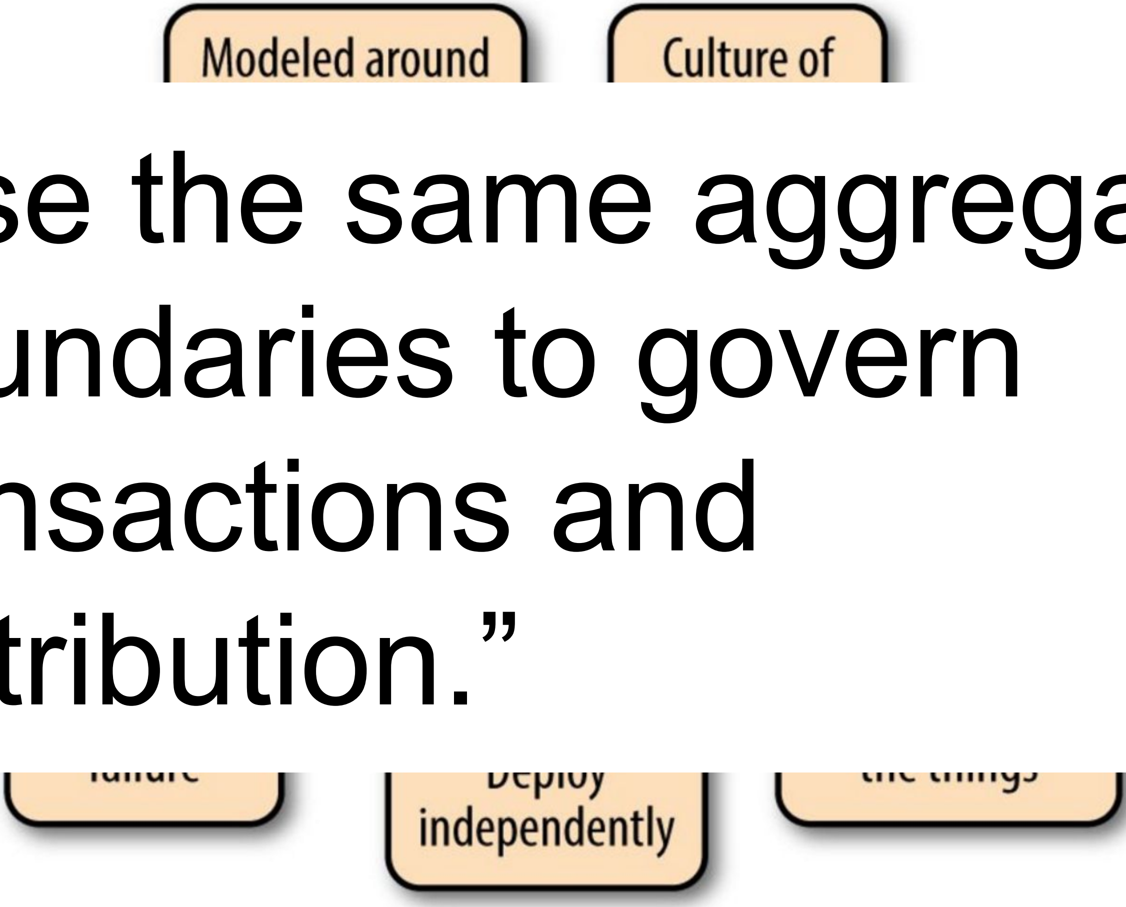

Modeled around business concepts
Culture of automation
Highly observable
Microservices
Small autonomous services
Hide internal implementation details
Isolate failure
Deploy independently
Decentralize all the things

*Figure 12-1. Principles of microservices*



“Use the same aggregate boundaries to govern transactions and distribution.”

*Figure 12-1. Principles of microservices*

## The best of both worlds: *business-focussed, & technically operable*

## What questions do you have?

(check out my course, "Domain-Driven Design: First Steps" on O'Reilly
and many thanks to @steve_cable's [UX Comics Pattern Library])