Alexey Zimarev

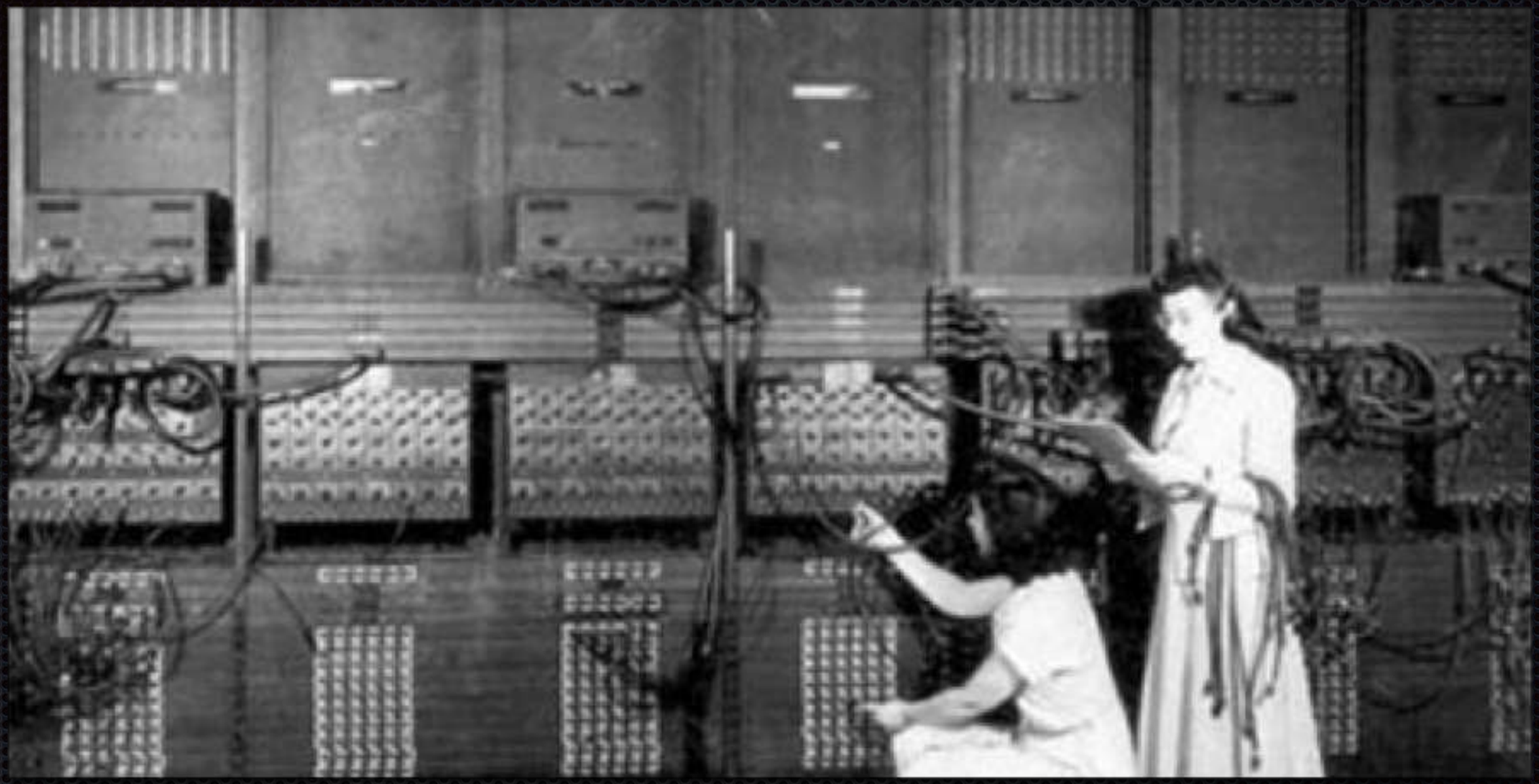
Software Architect at ABAX AS, Norway

DDD Norway Meetup Organiser

@Zimareff
alex@zimarev.com
https://www.linkedin.com/in/alexeyzimarev
http://zimarev.com

Domain-Driven Design

Why, What and How



History of Programming

FORTRAN

1957
Science
Math
Engineering

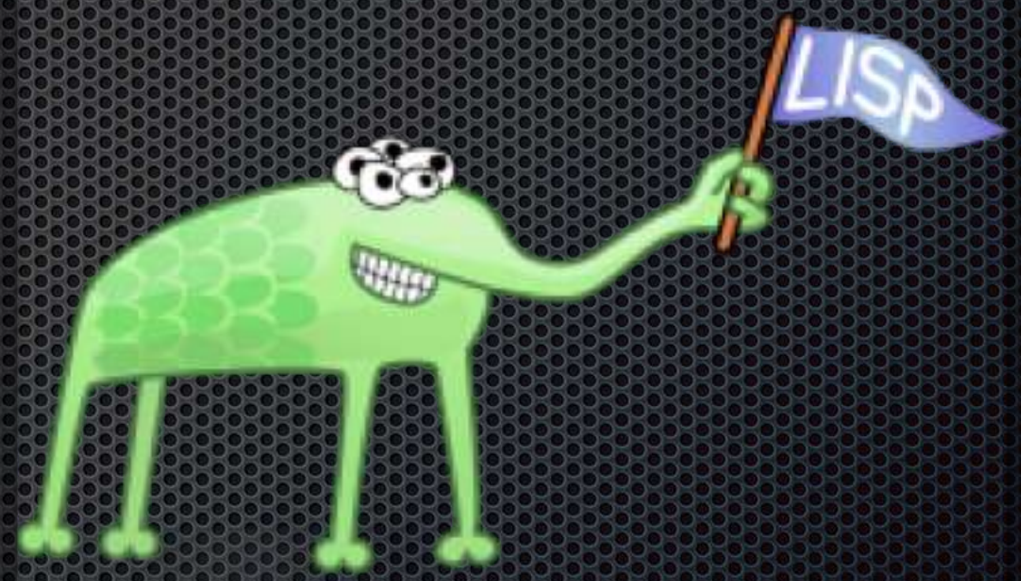
```
*L
C       CALCULATE STATISTICS O
        SUM=0
        SUMSQ=0
        TYPE 100
100     FORMAT("ENTER THE NUMB
        ACCEPT 10,N
10      FORMAT(I)
        DO 200 I=1,N
        READ 1,110,V
110     FORMAT(E)
        SUM=SUM + V
        SUMSQ=SUMSQ + V*V
        TYPE 120,I,V
120     FORMAT("VALUE",I,"IS",
200     CONTINUE
        SAMP=N
        AVRG=SUM/SAMP
        STD=SQTF(SUMSQ/SAMP -
        TYPE 300,N,AVRG,STD
300     FORMAT("NUMBER OF VALU
        END

*
.R FORT
?
```

LISP

1958
Lots of parenthesis

LISP

1958
Math
AI
Lambda calculus



COBOL
Common Business-Oriented Language

1959
Business
Finance
Administration

ALGOL

1958
Computer science

Found no adoption

```
                                        'PROCEDU
'BEGIN''INTEGER'L,U.
L%=L1.U%=U1.
PART%
P%=L.Q%=U.X%=A(P).Z%=A(Q).
'IF'X'GT'Z'THEN'
'BEGIN'Y%=X.A(P)%=X%=Z.
A(Q)%=Z%=Y 'END'.
'IF'U-L'GT'1'THEN'
'BEGIN'XX%=X.IX%=P.ZZ%=Z.IZ%=Q.
LEFT%
'FOR'P%=P+1'WHILE'P'LT'Q'DO'
'BEGIN'X%=A(P).
'IF'X'GE'XX'THEN''GOTO' RIGHT '
P%=Q-1.'GOTO' OUT.
RIGHT%
'FOR'Q%=Q-1'WHILE'Q'GT'P'DO'
'BEGIN'Z%=A(Q).'IF'Z'LE'ZZ'THEN
Q%=P.P%=P-1.Z%=X.X%=A(P).
DIST%
'IF'X'GT'Z'THEN''BEGIN'
Y%=X.A(P)%=X%=Z.A(Q)%=Z%=Y'END'
'IF'X'GT'XX'THEN''BEGIN'XX%=X.I
'IF'Z'LT'ZZ'THEN''BEGIN'ZZ%=Z.I
'GOTO' LEFT.
```

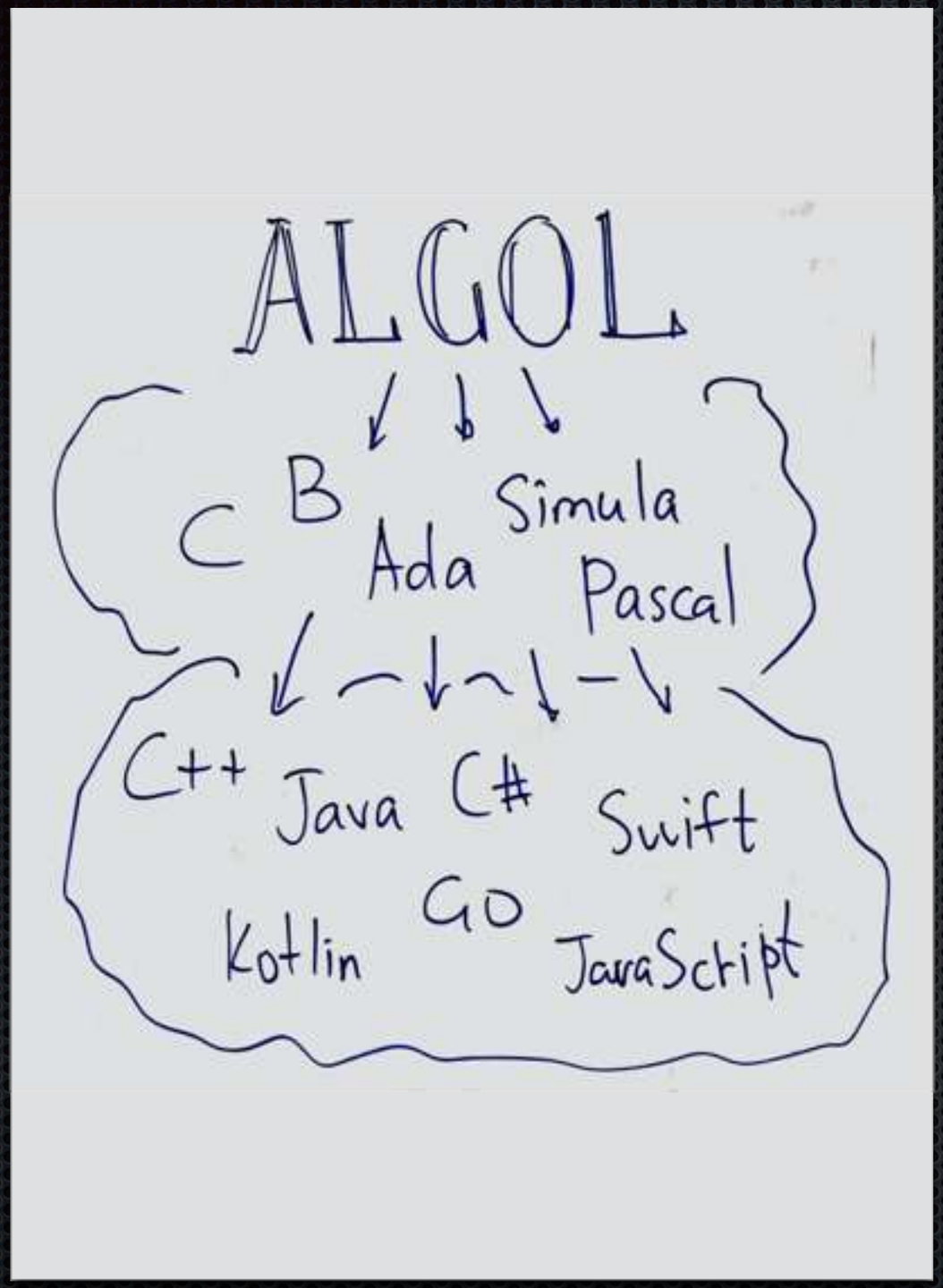
Early days programmers

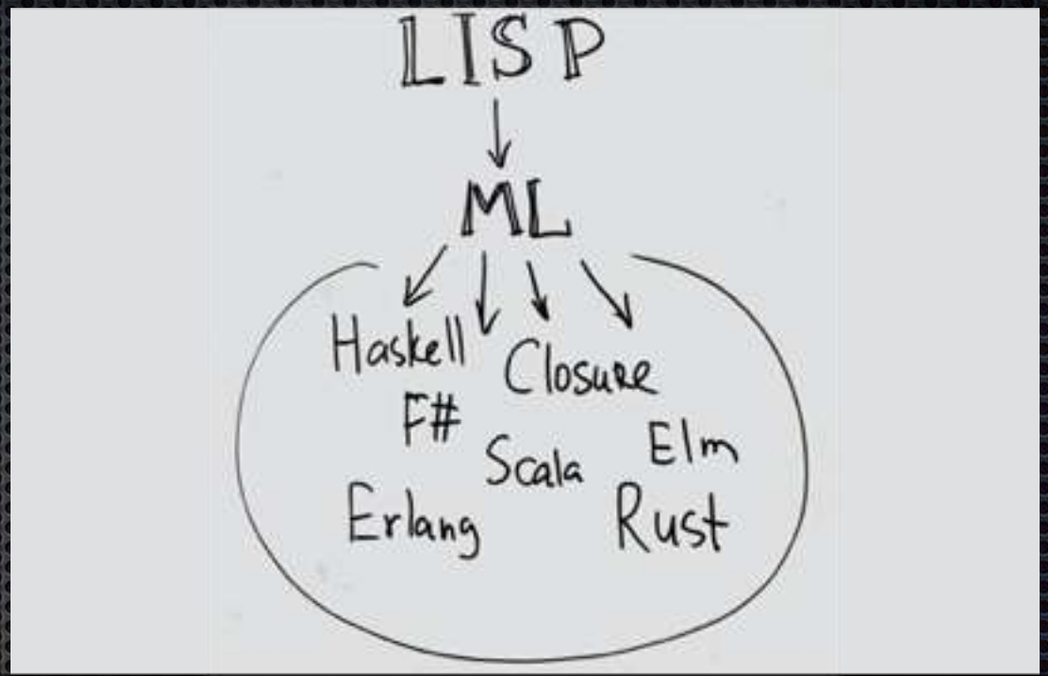
- Mathematicians
- Scientists
- Engineers
- Business people

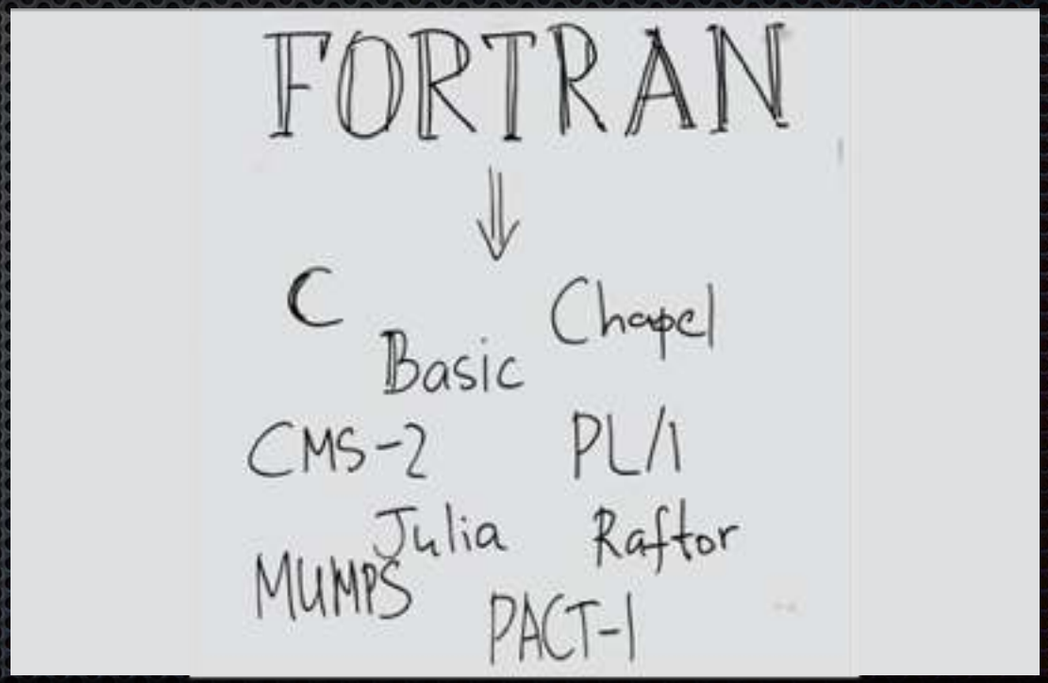
```
---+----1----+----2----+----3----+----4----+----5----+----6----+----7--
      PROCEDURE DIVISION.
          SET ACCOUNTANT TO TRUE
          SET INEXPERIENCED TO TRUE
          SET WANTS-TO-STAYPUT TO TRUE

          EVALUATE TRUE        ALSO TRUE          ALSO TRUE
             WHEN  ACCOUNTANT ALSO EXPERIENCED  ALSO WILLING-TO-TRAVEL
                MOVE 2000 TO SALARY
             WHEN  ACCOUNTANT ALSO INEXPERIENCED ALSO WANTS-TO-STAYPUT
                MOVE 1000 TO SALARY
             WHEN COMPUTER-SCIENTIST ALSO ANY    ALSO ANY
                MOVE 3000 TO SALARY
             WHEN SYSTEMS-ANALYST ALSO EXPERIENCED ALSO ANY
                MOVE 4000 TO SALARY
             WHEN ANY                ALSO EXPERIENCED ALSO ANY
                MOVE 500 TO SALARY
          END-EVALUATE

          DISPLAY 'SALARY = ' SALARY
          STOP RUN.
```

COBOL was almost like plain English

WE NEED COBOL PROGRAMMERS FOR OUR MAINFRAME MILLENIUM PROBLEM.
IF YOU SEE ANYONE WHO LOOKS LIKE A COBOL PROGRAMMER, LET ME KNOW.
TURN AROUND.
ARE YOU A COBOL PROGRAMMER?
NO, BUT I'M OFTEN TOLD I LOOK LIKE ONE.
YOU'RE HIRED.
www.unitedmedia.com
© 1997 United Feature Syndicate, Inc.


ALGOL
C
B
Ada
Simula
Pascal
C++
Java
C#
Swift
Kotlin
Go
JavaScript

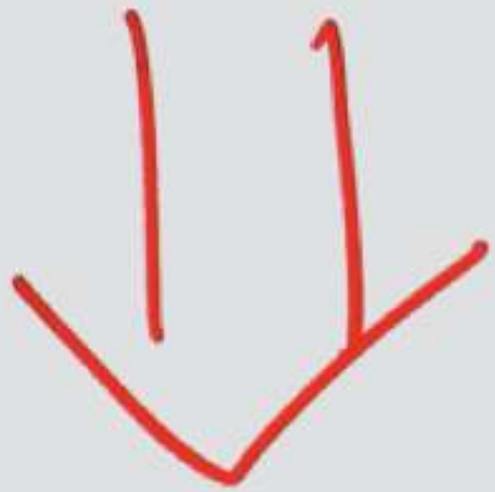

LISP
ML
Haskell
Closure
F#
Scala
Elm
Erlang
Rust


FORTRAN
C
Basic
Chapel
CMS-2
PL/1
Julia
Raftor
MUMPS
PACT-1

Computer people took the language with least adoption among domain experts

and started using it everywhere

COBOL



COBOLSCRIPT

Cobol only evolved to another Cobol

HackerRank

COBOL Skill Gap

55

average age of COBOL programmer

70%

of universities surveyed say they don't include COBOL in curriculum

18

COBO's rank in Tiobi Index of popular programming languages

70%

of COBOL used in all business transactions

220 Bn

lines of code in COBOL

1.5 Bn

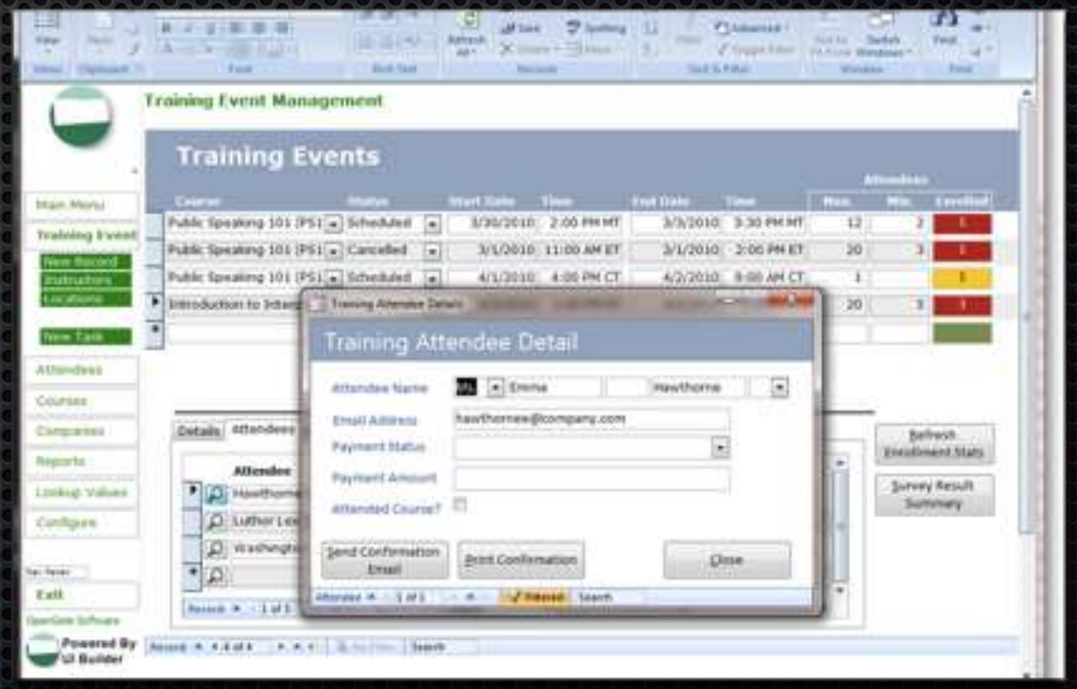
new lines of code in COBOL per year

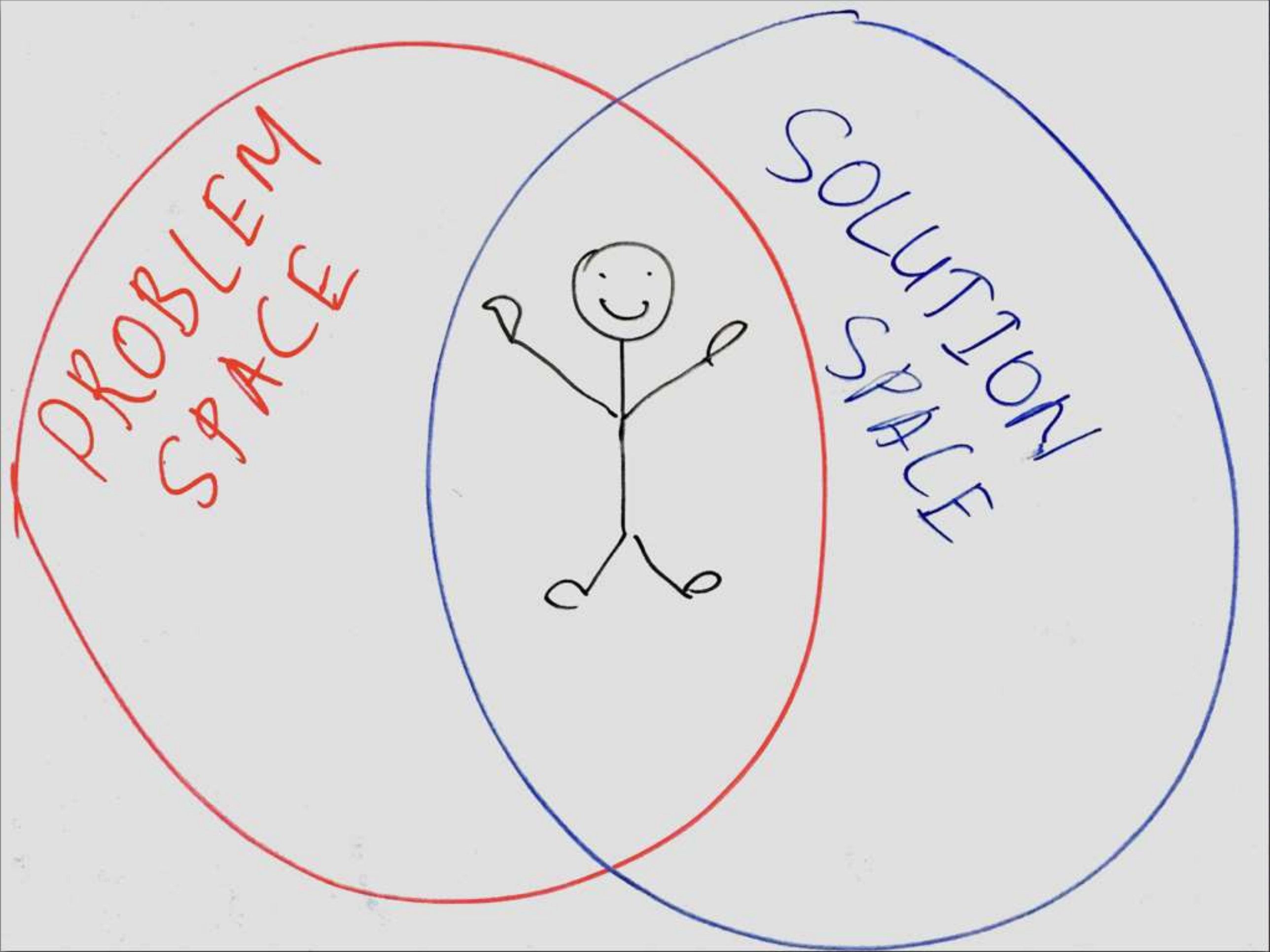
Source: Microfocus, Gartner, laserfiche

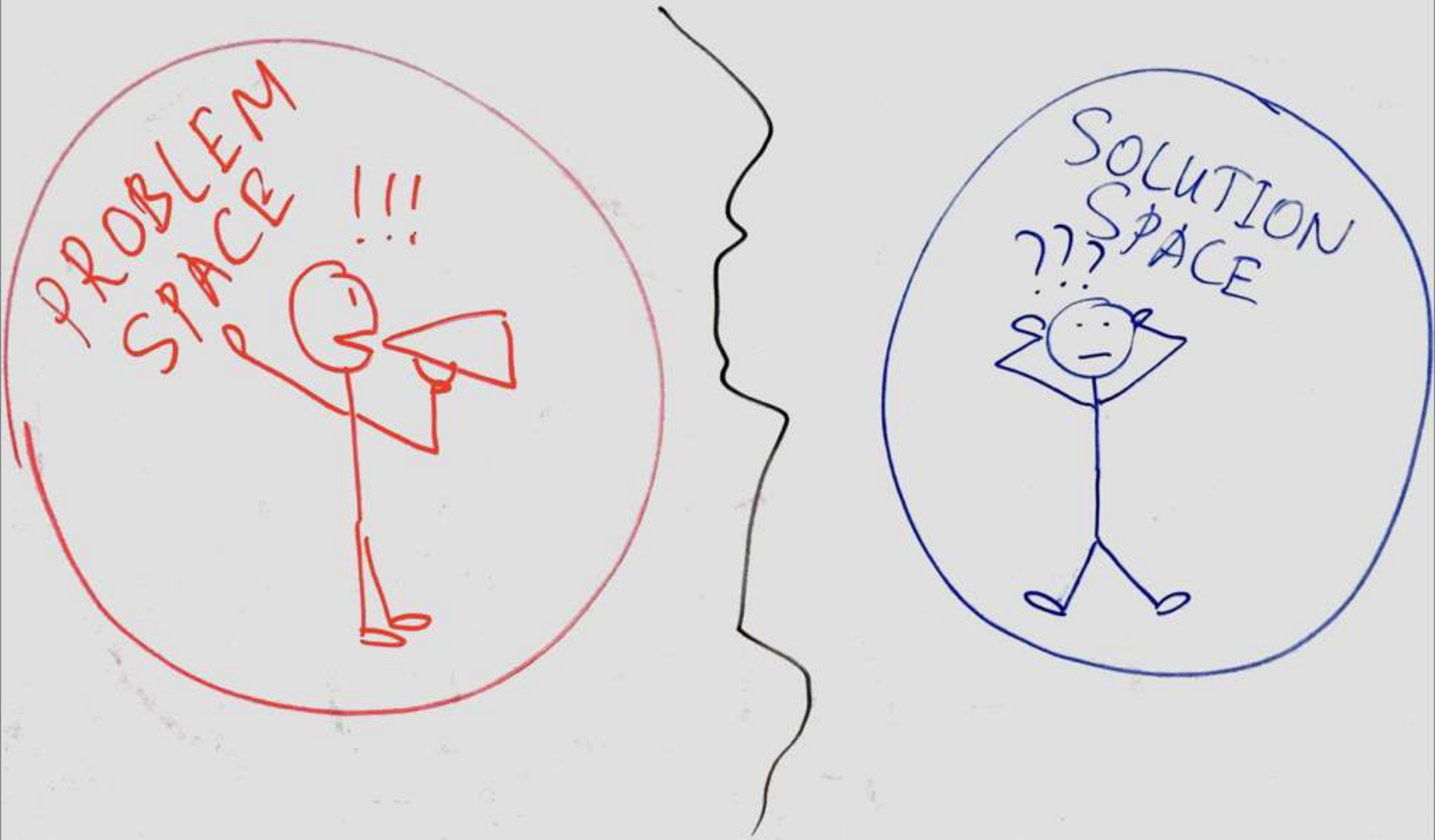
Business

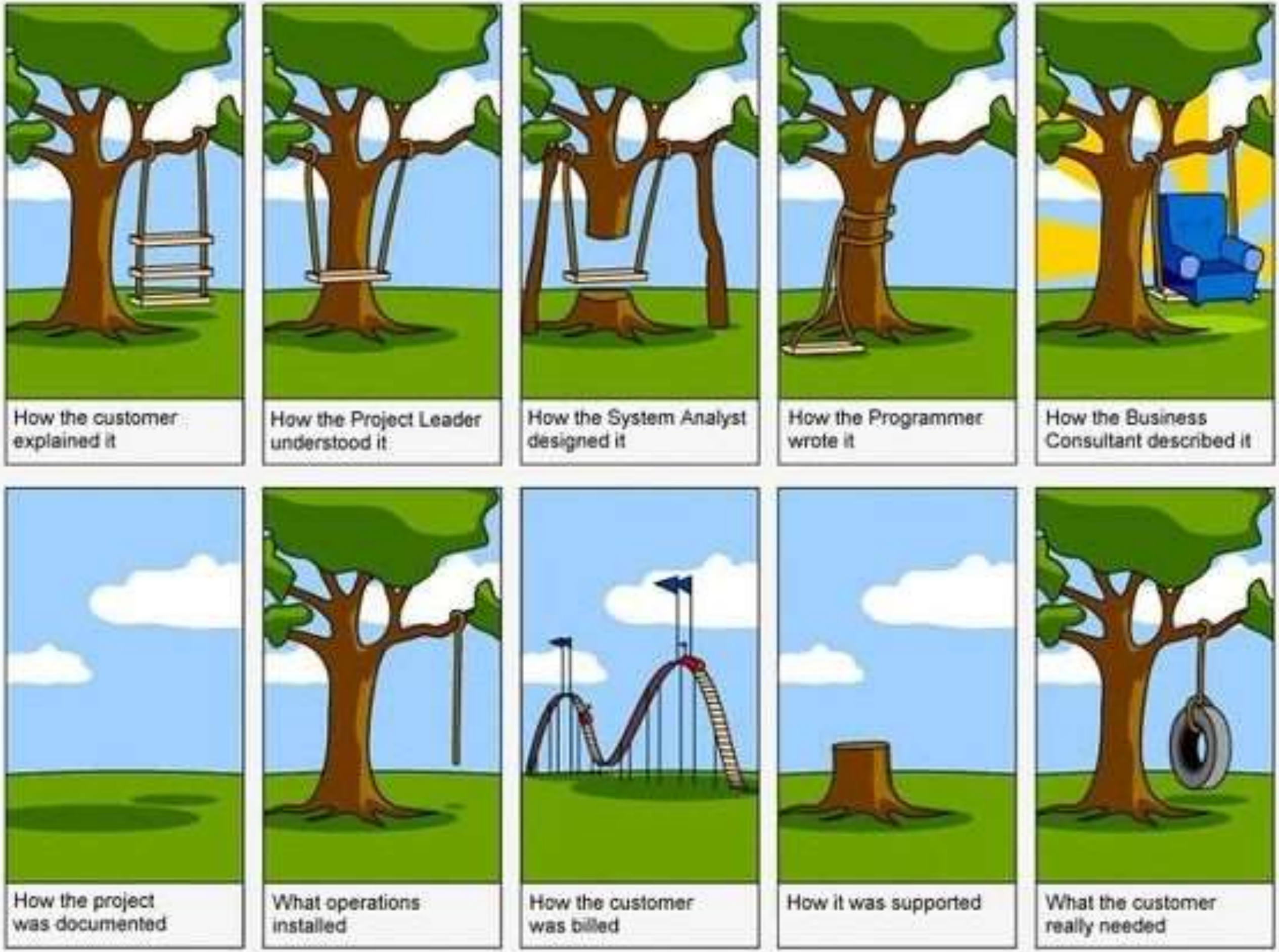
Still does programming

A	B	C	D	E	F
gmtpmt& mnpmt					Formu
	OTBT Rzpsmr	OTBT N/B	Tsct OTBT NB		
	11434	64037	200000		
zt vglmOzs mnpmt					
	Lsst 12 Ognths sslzs snd rzpsmr zxpgsmrz pzrmgd cslcmlstmgn				
	Tgtsl bssz	svzrsgz rmntmOz	BzTgrz 12 O	R/Onth -1	
bmmlt:	1412	11.97	1400	0	
rzpsmr bmmlt:	306	4.36	0	2	
mmlt mnpmt	272	4.01	0	1	
gr 2 bmmlt mnpmt	0	#DIV/0!	0	0	
gr 3 bmmlt mnpmt	34	7.16	0	1	
mR mNAUT					
		Rzpsmr mssgz		Rzpsmr pgpmlstmgn	
	Tgtsl Rzpsmr	Rzpsmrzd DOs	Rzpsmrzd Wsrr	NgrOsl Rzpsmr	Lgc
	306	2	40	264	
nct mnTgrOstmgn				% cmrrznt mnstslizd bssz sTtzr T	
	Argdmct NsOz	ytylz NsOz	Rzvmzw Dstz	Ognths sctmvz	svzr Tmtmrz
	DzhmOmdmTmzr	sB6065903CT	25/06/1998	76.8	
NmObzr mnTgrOstmgn			Ognth/dsy/yzsr		
	sDy Asrt NmObzr	yzt-mp Dstz	ytsndsrd cgst	Ognthly mssgz	12 Ognt
	43588136-002	1/02/1992	$675.92	9.7	
		Ognth/dsy/yzsr			



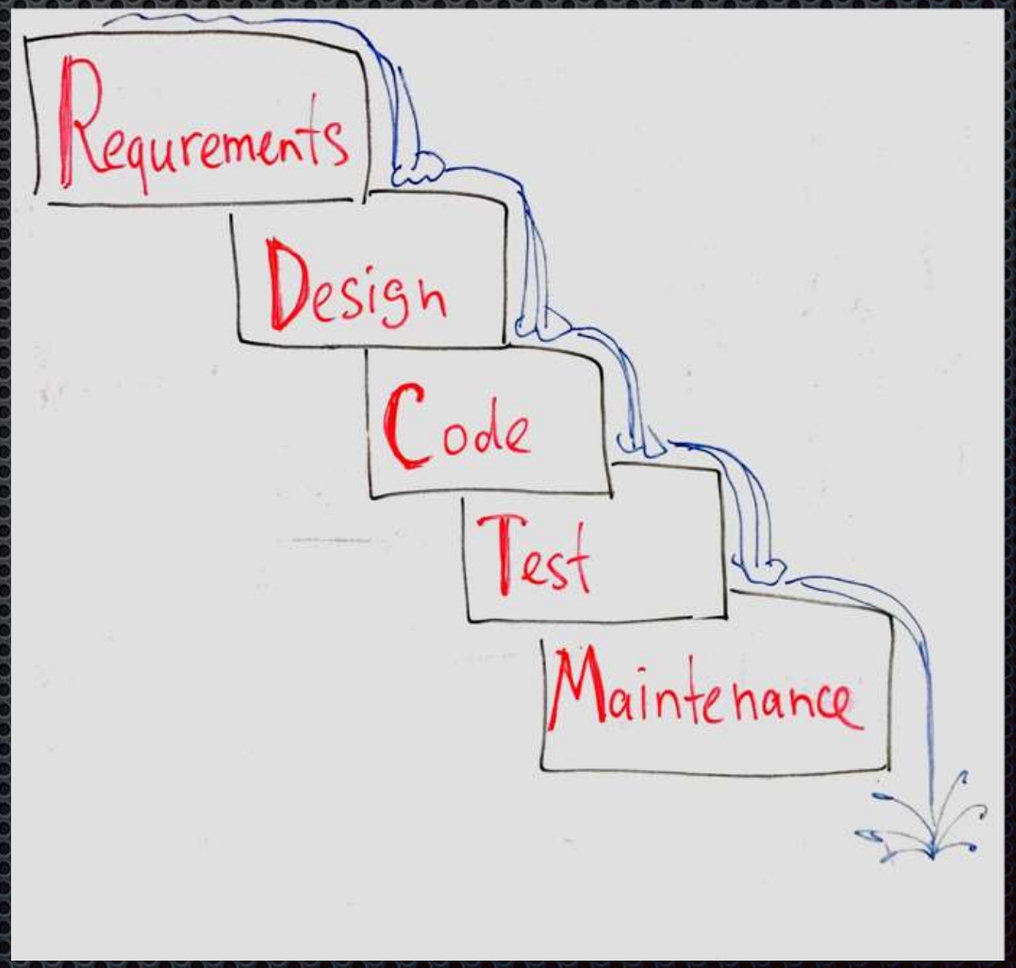

PROBLEM SPACE
SOLUTION SPACE


PROBLEM SPACE
!!!
SOLUTION SPACE
???

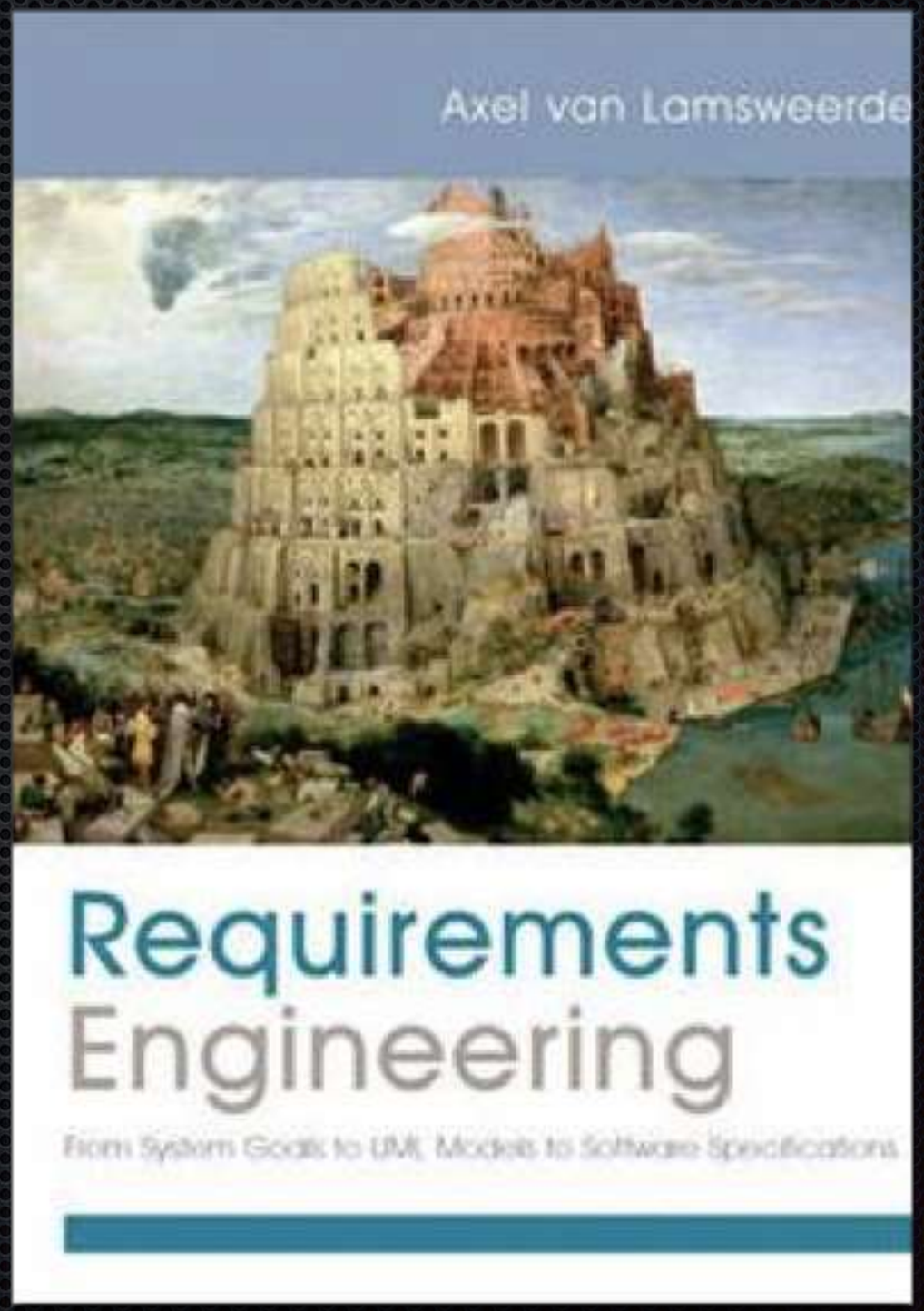

How the customer explained it
How the Project Leader understood it
How the System Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Waterfall

- Collect requirements
- Design
- Code
- Test
- Maintain



Book on Requirements

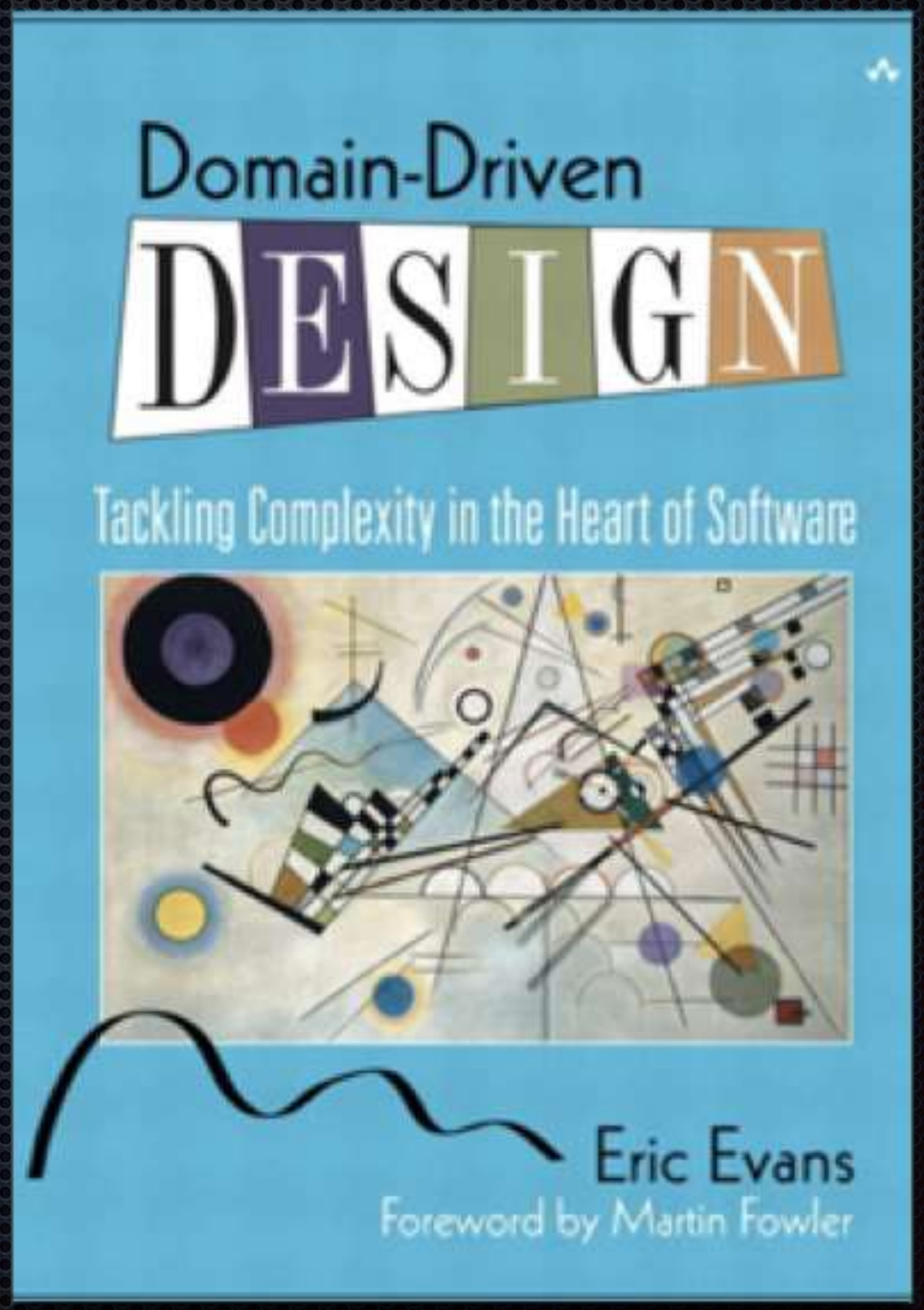


Customer

Gone missing

Domain-Driven Design

2003
Eric Evans

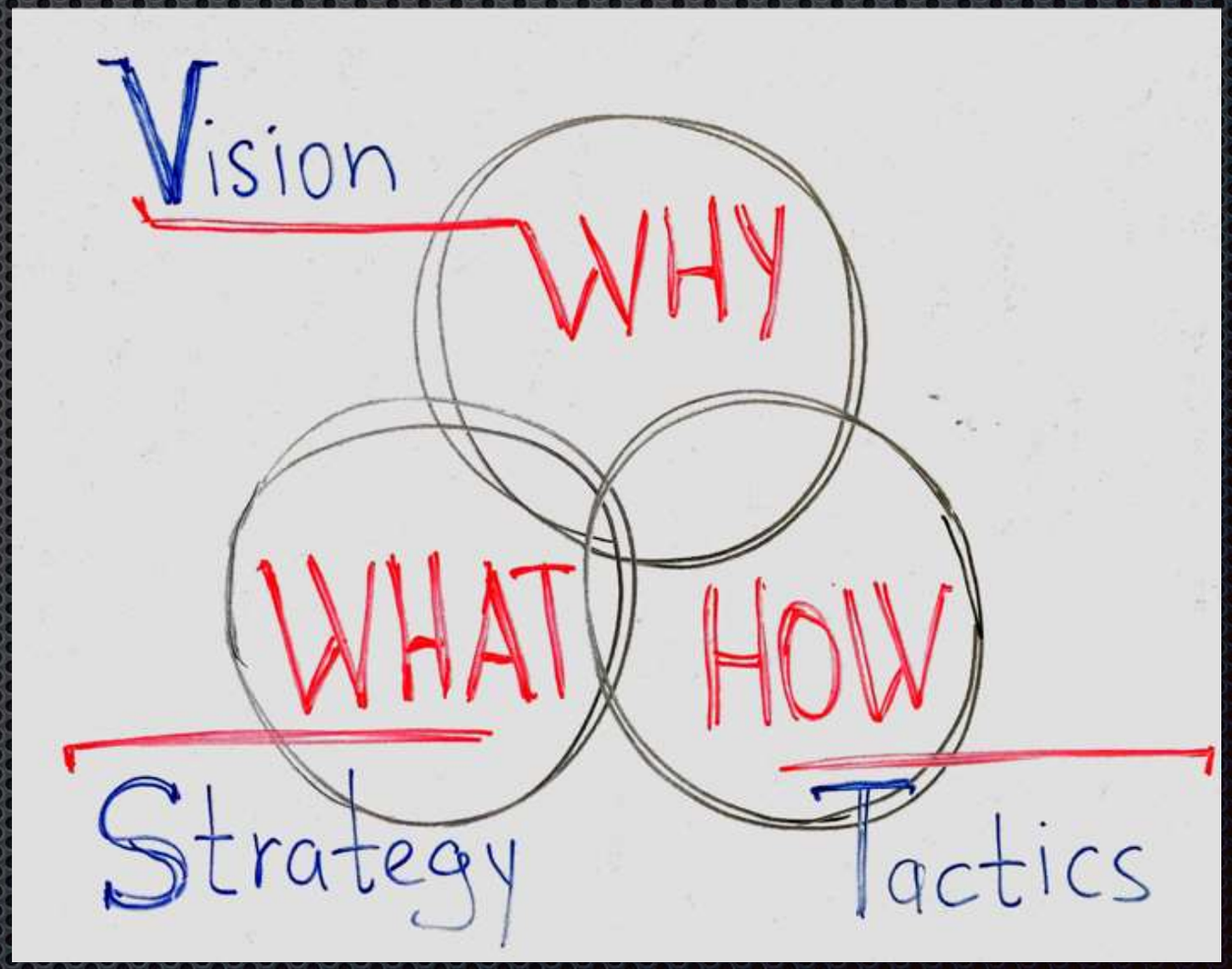


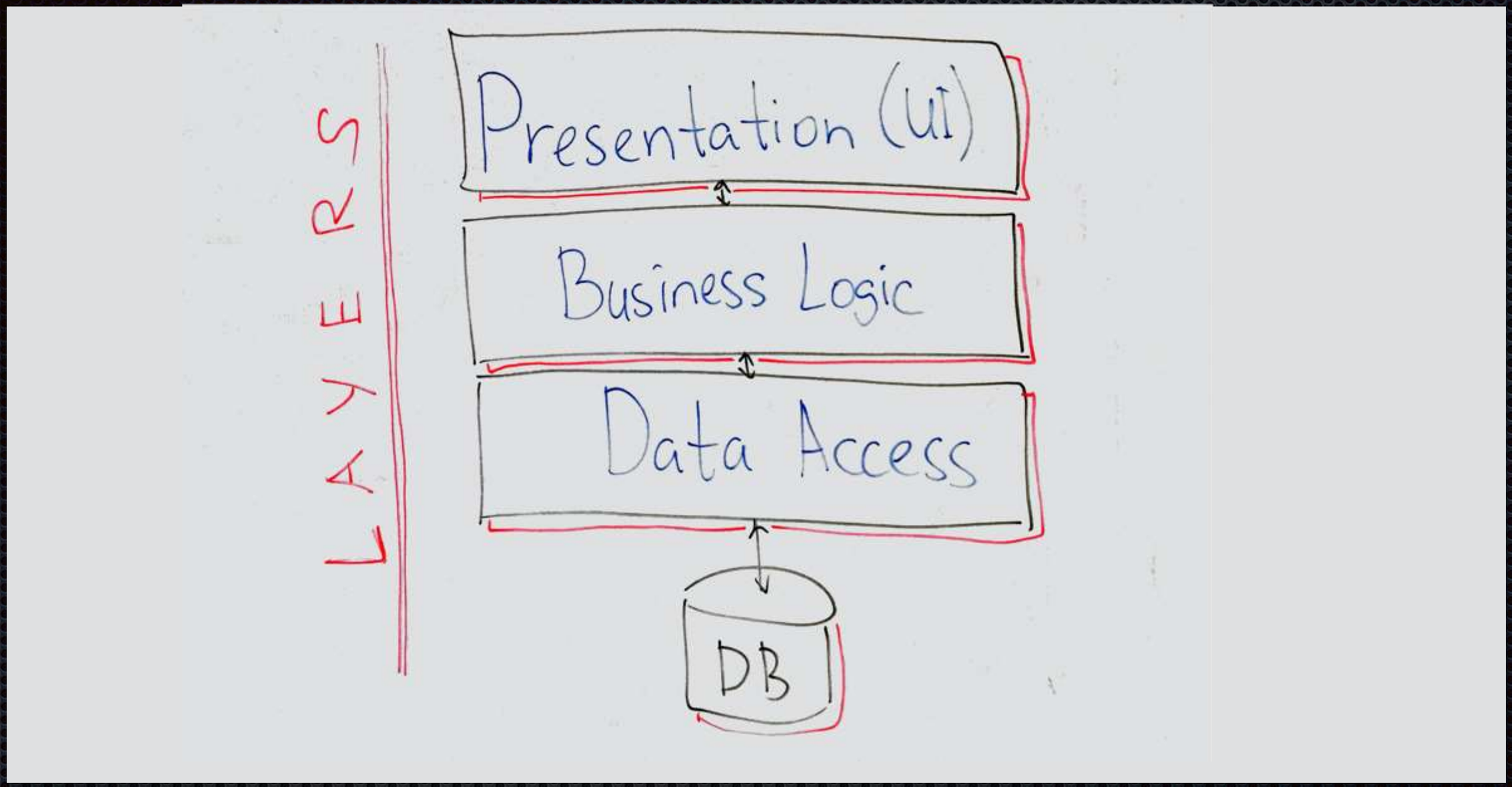
The Blue Book

- Knowledge crunching
- Ubiquitous language
- Model-driven design
- Implementation patterns (entity, aggregate, aggregate root, value object, strategy, domain service, domain event, repository, etc)
- Supple design, Refactoring towards deeper insight
- Strategic design (bounded context, context map, core domain, subdomains)

What went wrong?

- Strategy is hard
- Tactics is easier
- People prefer easier tasks

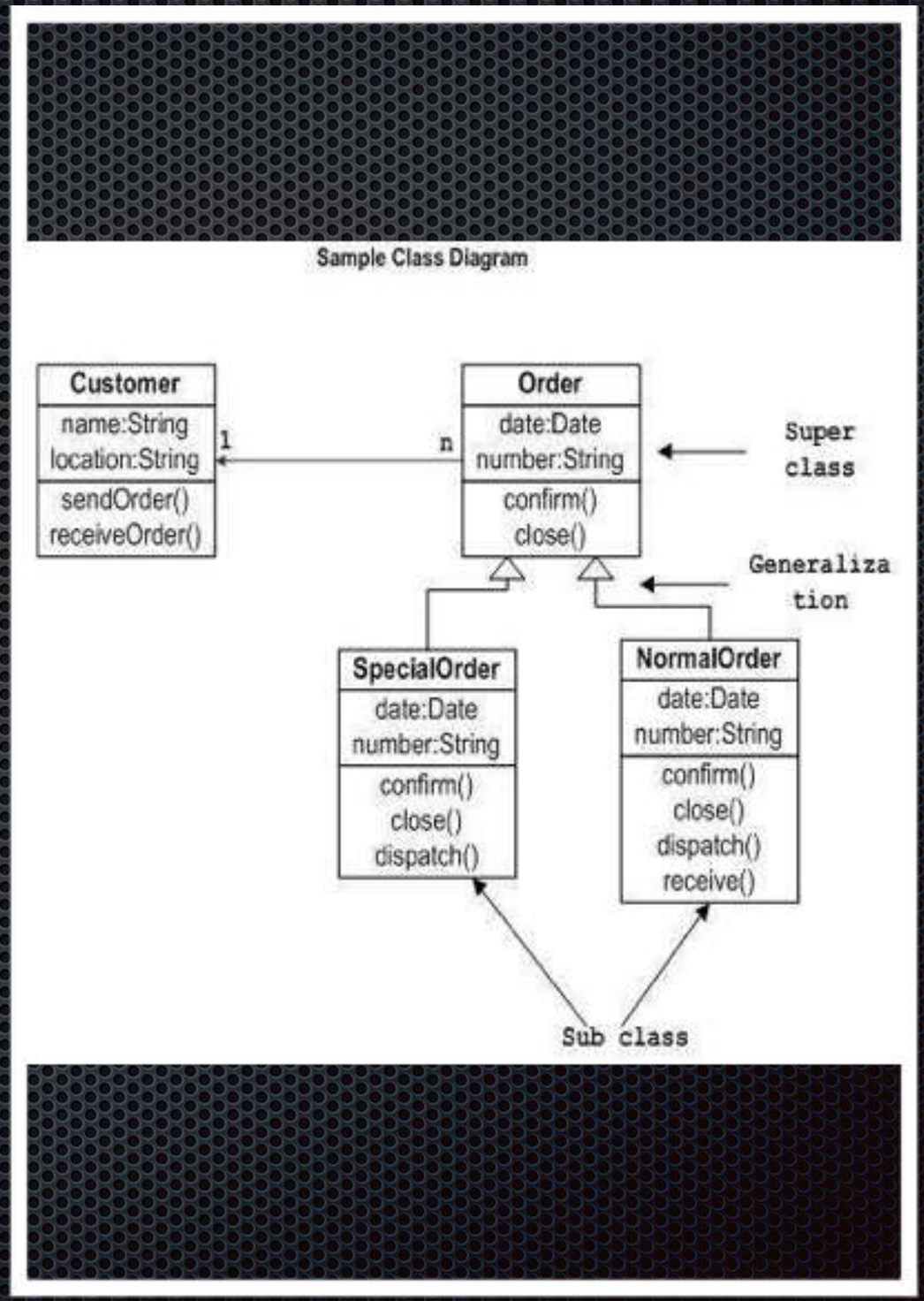




Layered Architecture

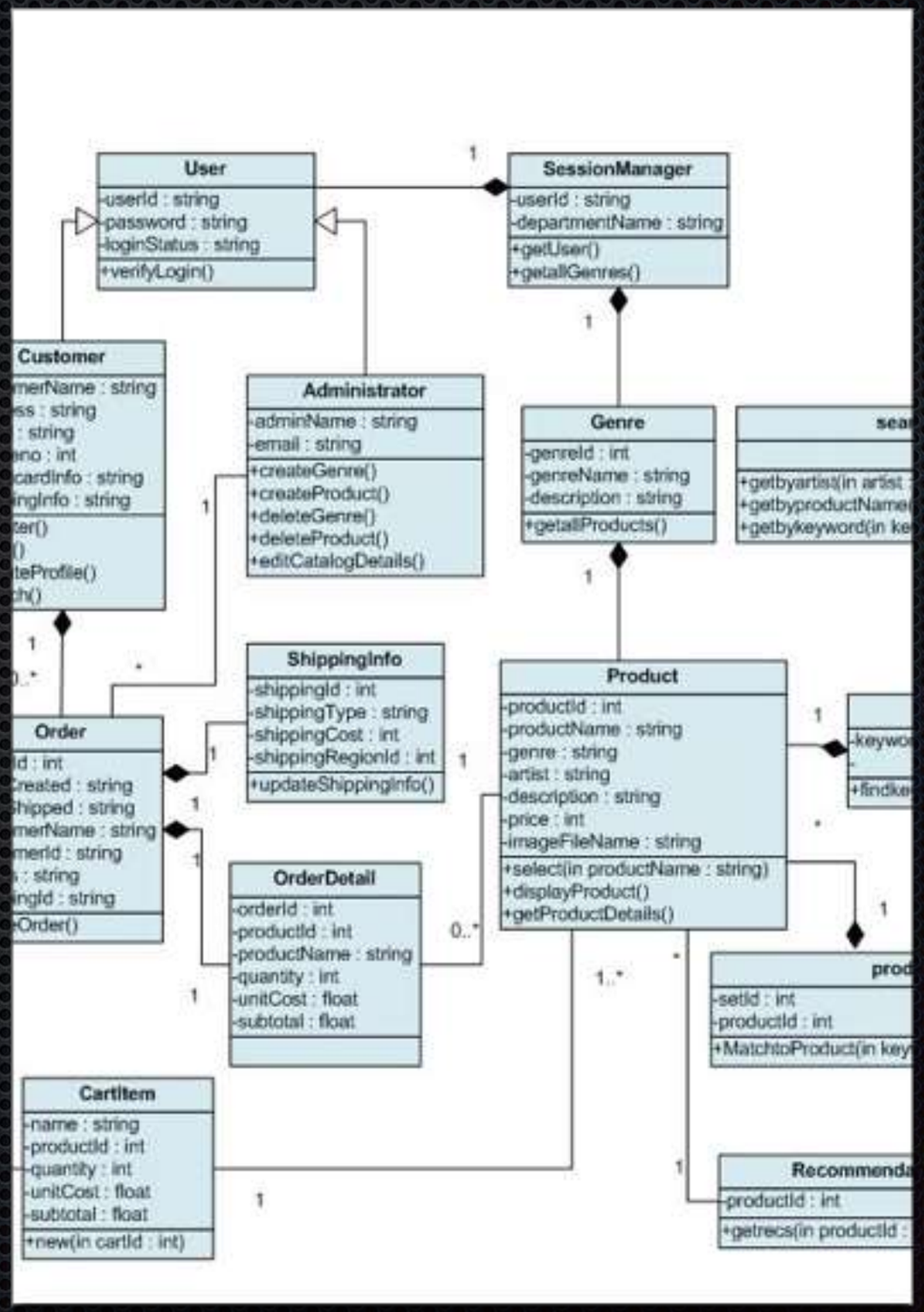
Domain Model

Starts simple and clean



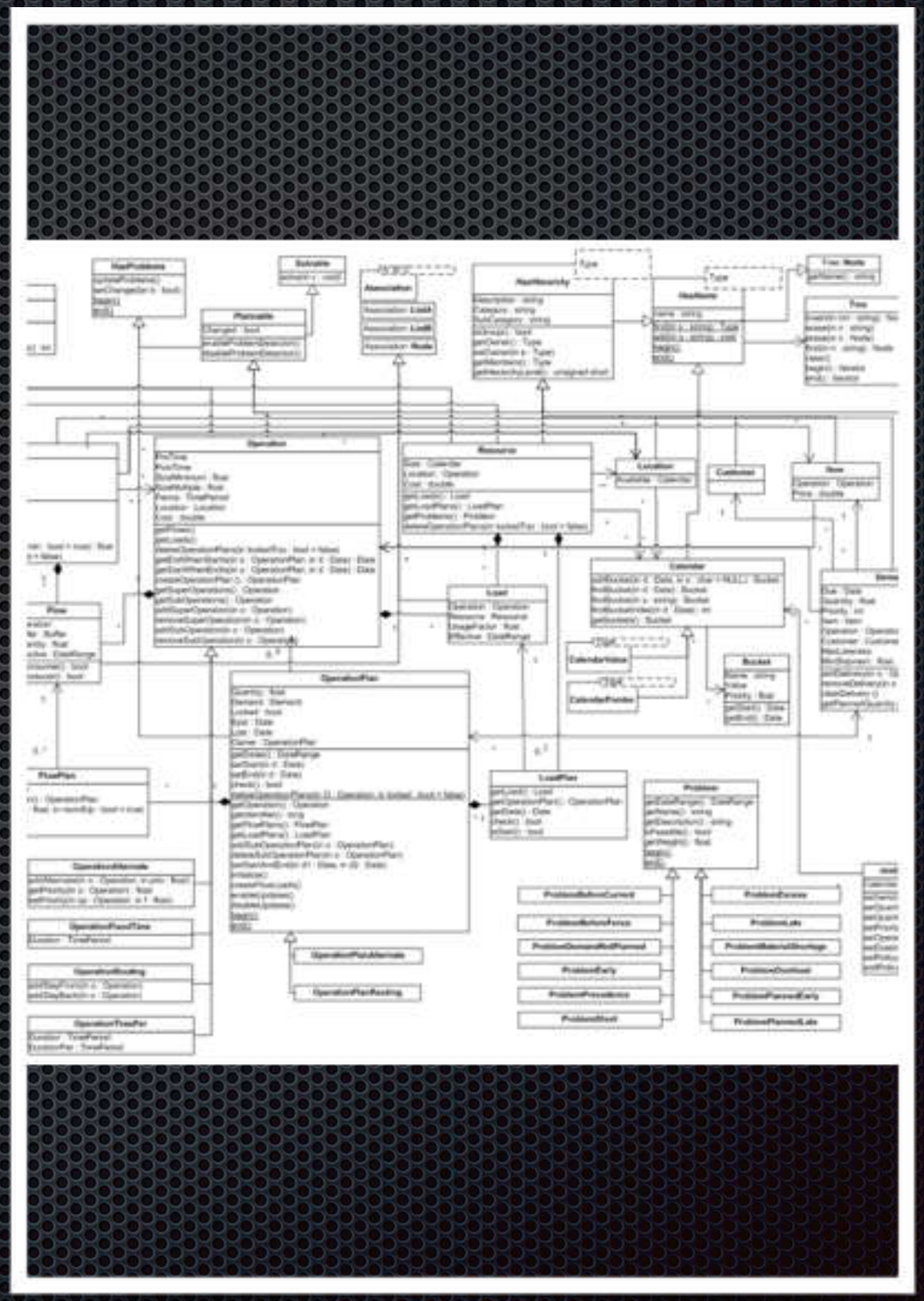
Domain Model

Growth a little



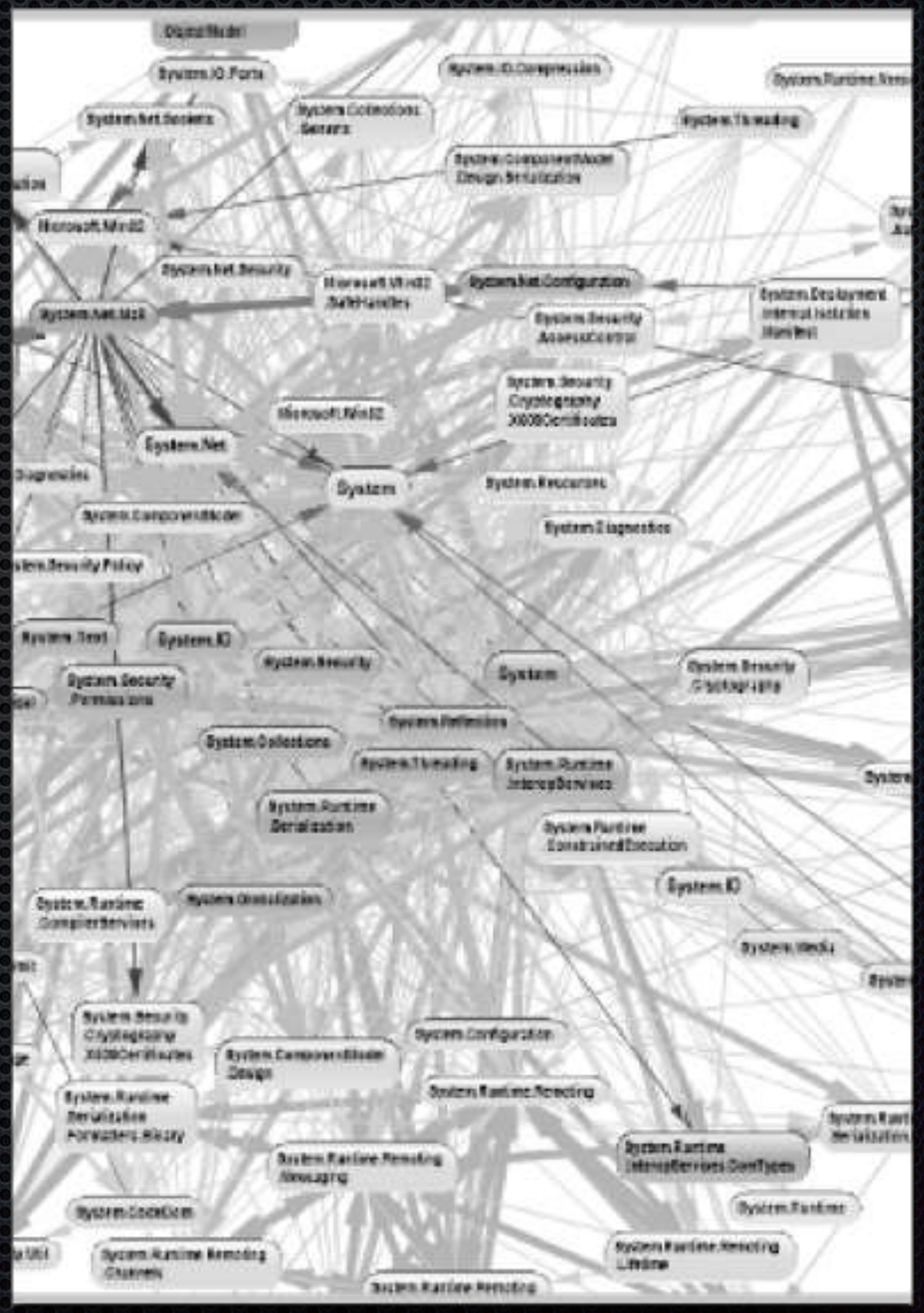
Domain Model

A little more



Domain Model

Eventually....



Big Ball of Mud

“A Big Ball of Mud is a haphazardly structured, sprawling, sloppy, duct-tape-and-baling-wire, spaghetti-code jungle. These systems show unmistakable signs of unregulated growth, and repeated, expedient repair. Information is shared promiscuously among distant elements of the system, often to the point where nearly all the important information becomes global or duplicated. The overall structure of the system may never have been well defined. If it was, it may have eroded beyond recognition”

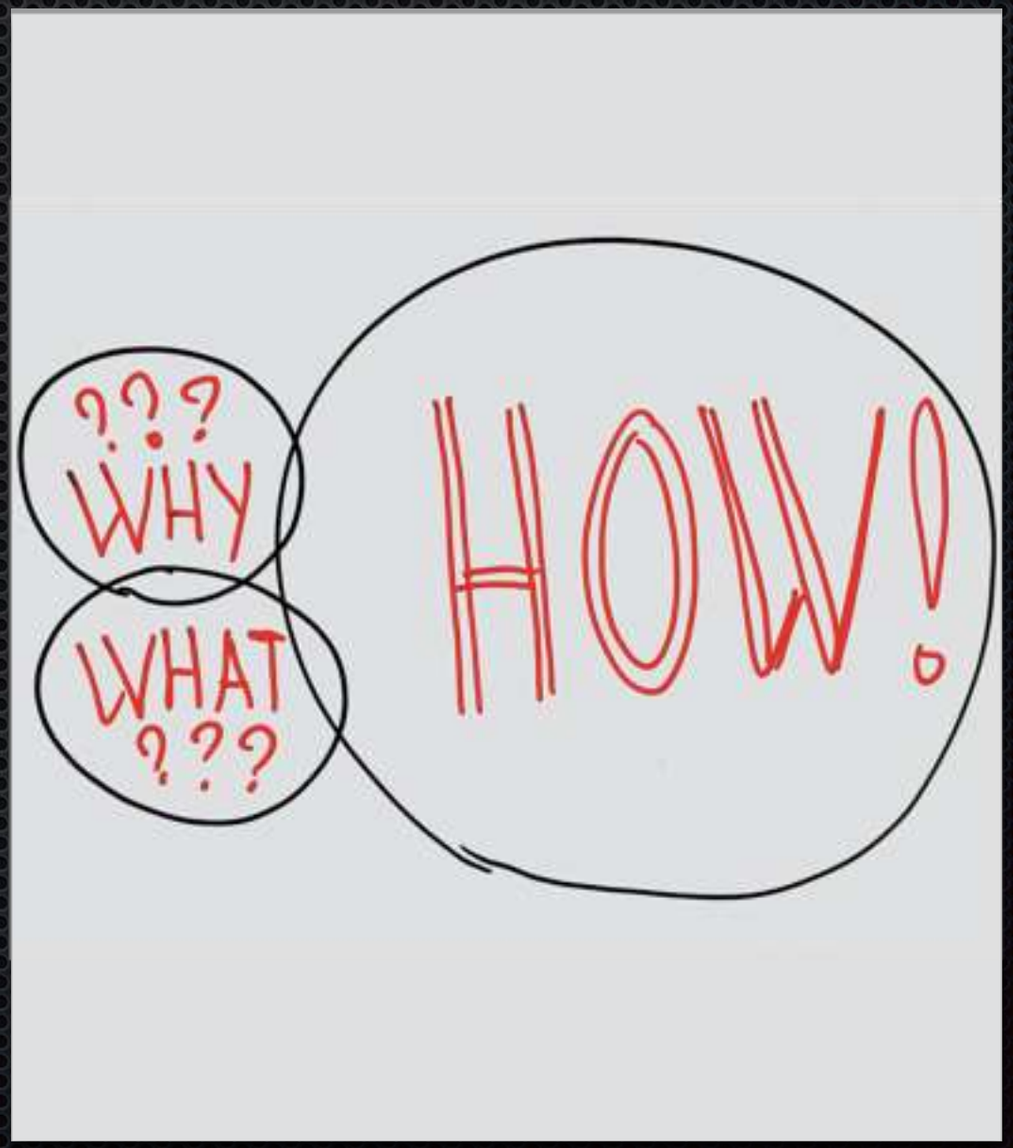
–Brian Foote and Joseph Yoder's - Big Ball of Mud - 1997

Big Ball of Mud

- Most popular architecture style
- Does something useful
- Without telling anyone how
- Hard to impossible to change

It started so well!

- Look, I have entities
- Wow, here is my aggregate root
- Check this, I got bunch of repositories



“DDD Light”

- Look, I have entities
- Wow, here is my aggregate root
- Check this, I got bunch of repositories

That is not DDD

developing

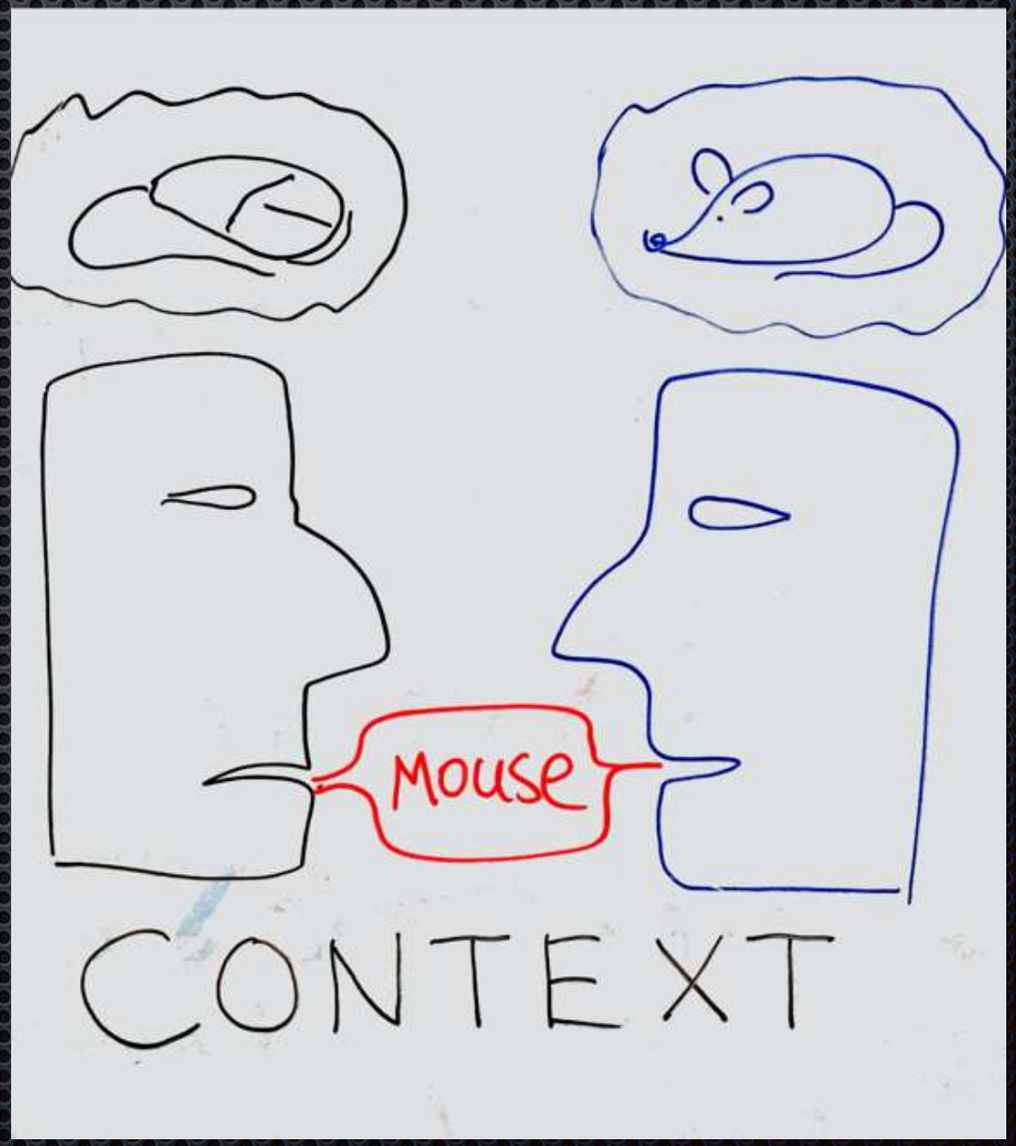
Ubiquitous Language

within

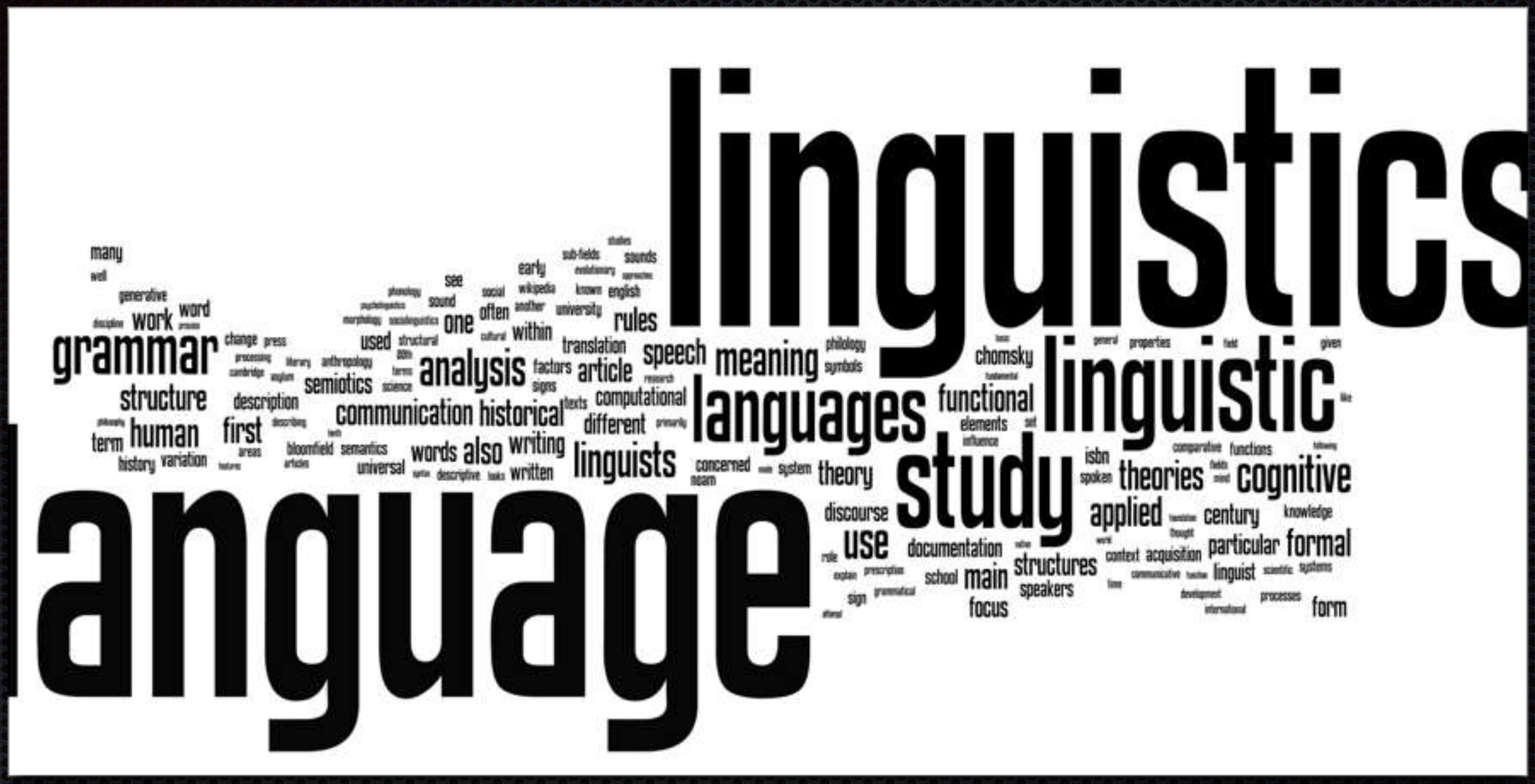
Bounded Context

Context is the thing

- The bandage was **wound** around the **wound**.
- We must **polish** the **Polish**.
- He could **lead** if he would get the **lead**.
- The soldier decided to **desert** his **dessert** in the **desert**.
- Since there is no time like the **present**, he thought it was time to **present** the **present**.



http://www.mtmlinguasoft.com/context-is-everything-especially-in-translation/



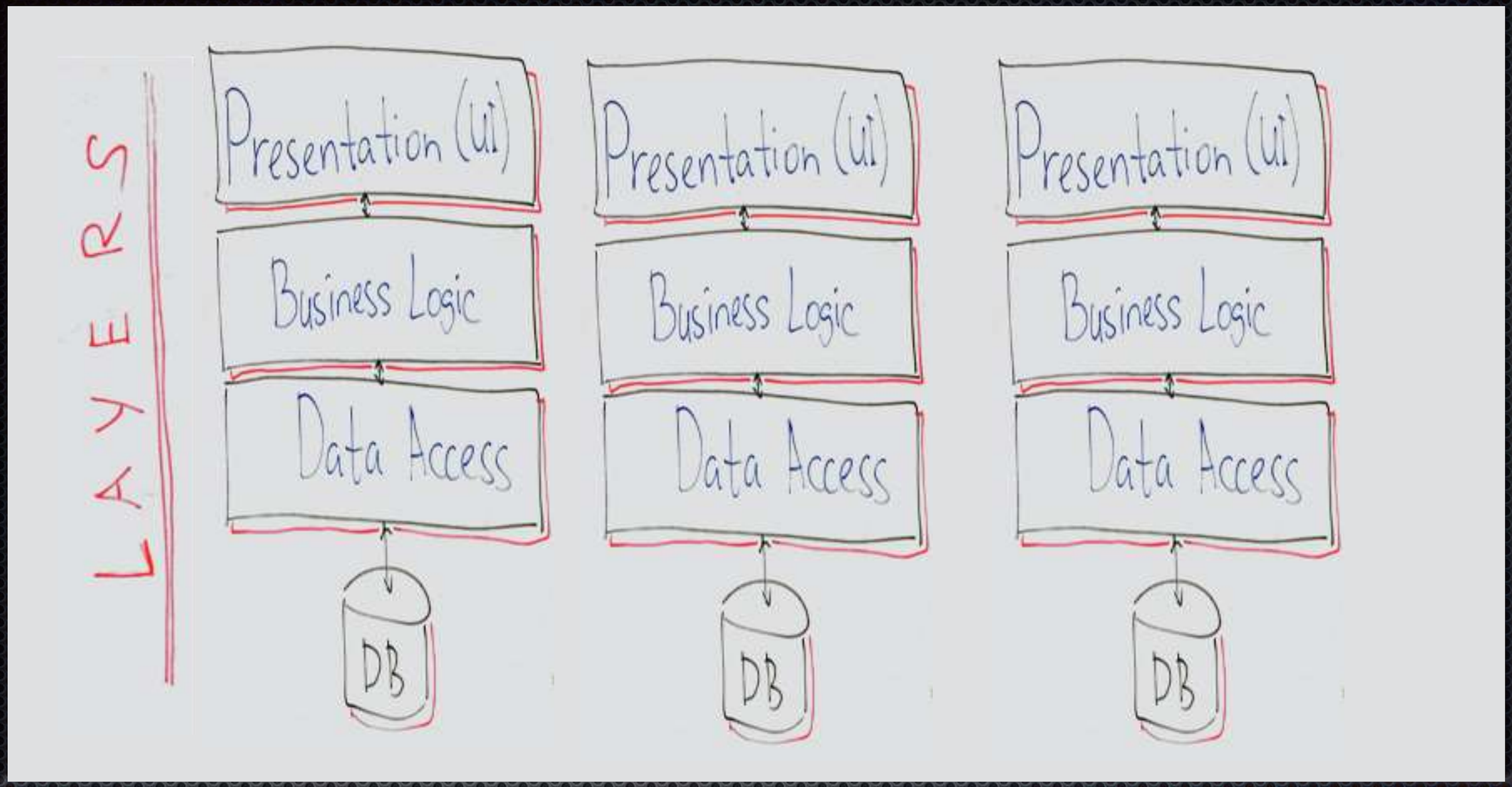
We are linguists

Talk to domain experts

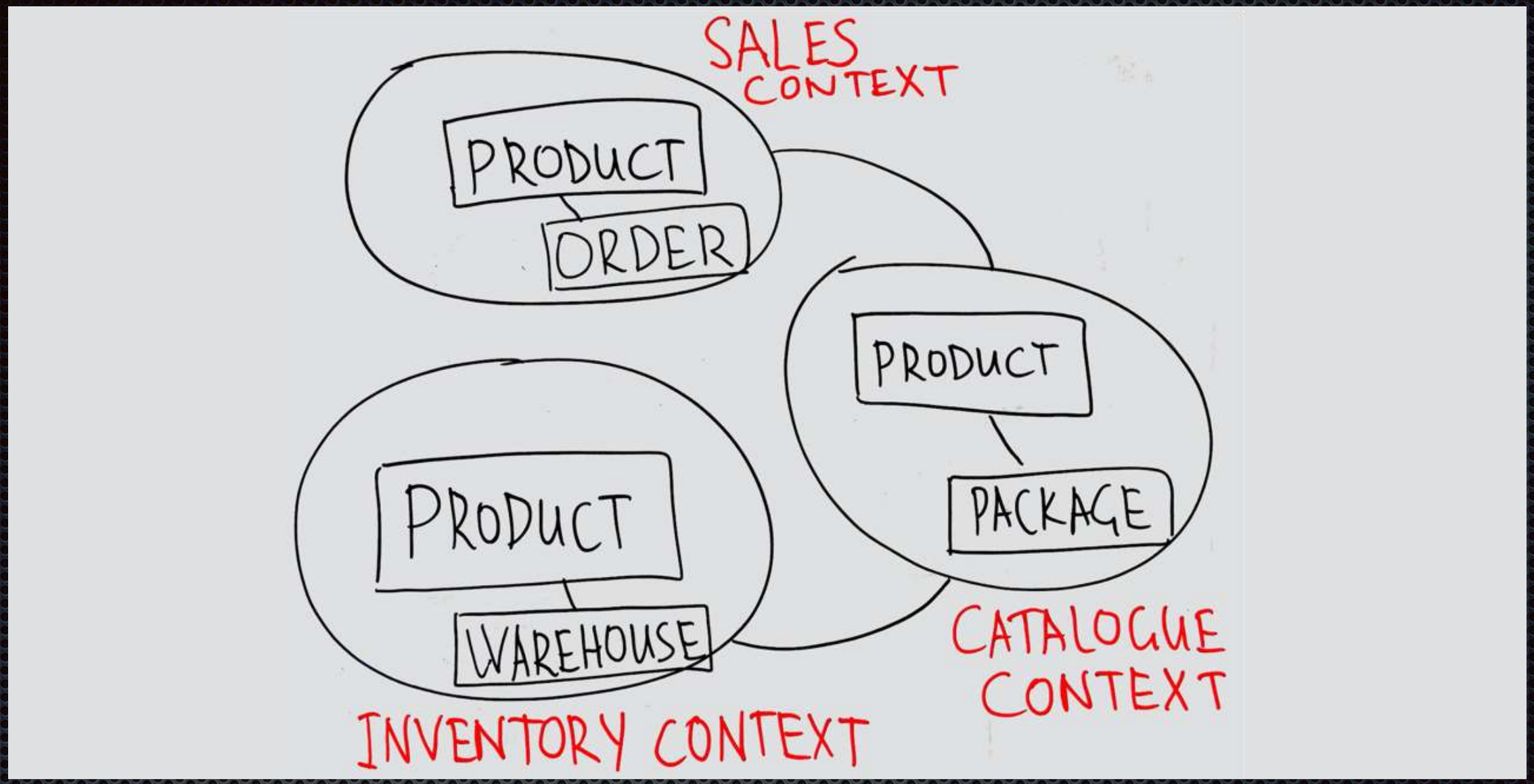
- Whiteboard
- Event Storming
- Observations
- Business Model Canvas
- BDD/Visual Language (Given-When-Then)

Bounded Context

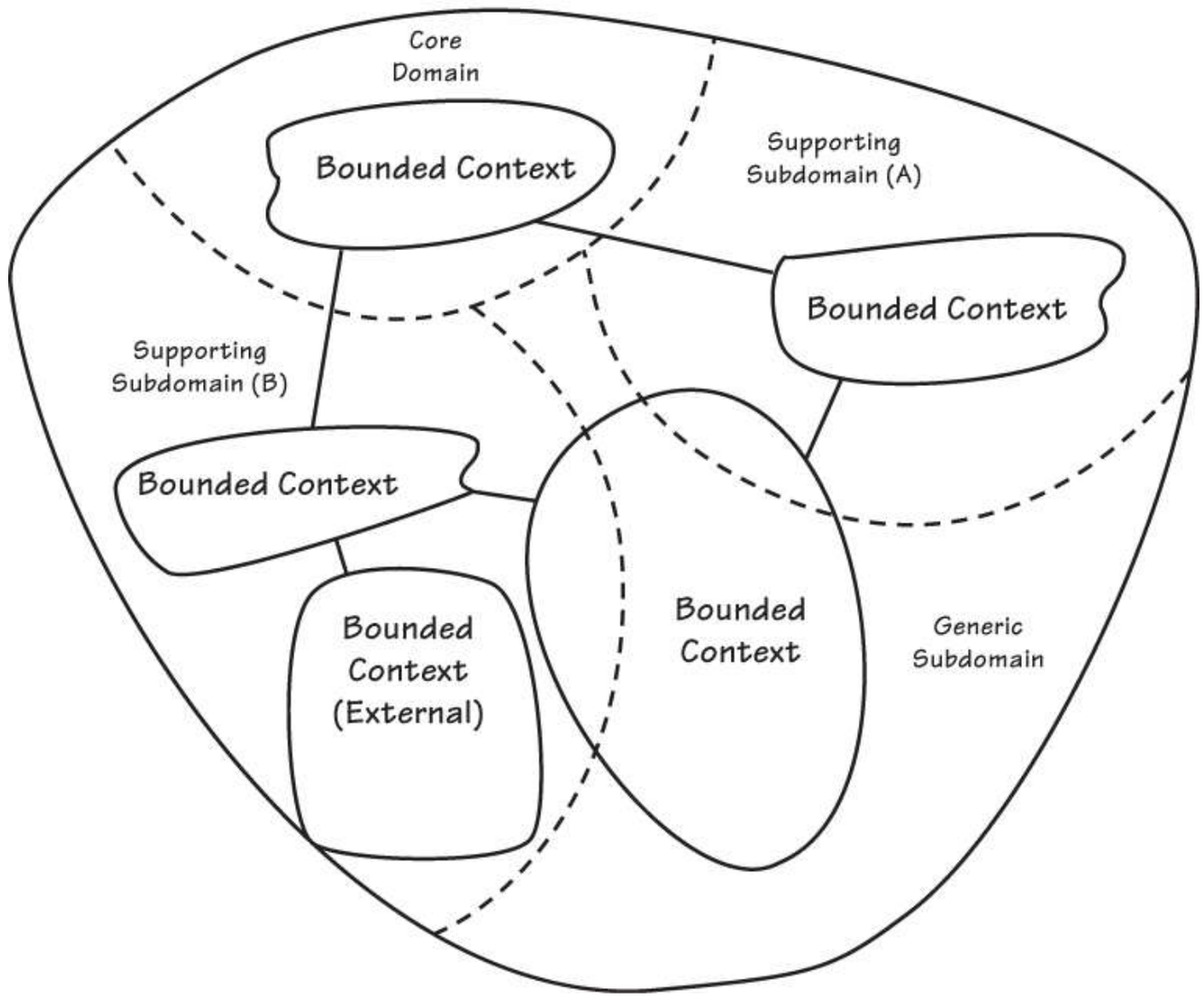
- The place where language lives
- Same word might have different meaning in another context
- High cohesion - hard to split
- Speed of change - things change together



Layers - per context


SALES CONTEXT
PRODUCT
ORDER
PRODUCT
PACKAGE
PRODUCT
WAREHOUSE
CATALOGUE CONTEXT
INVENTORY CONTEXT

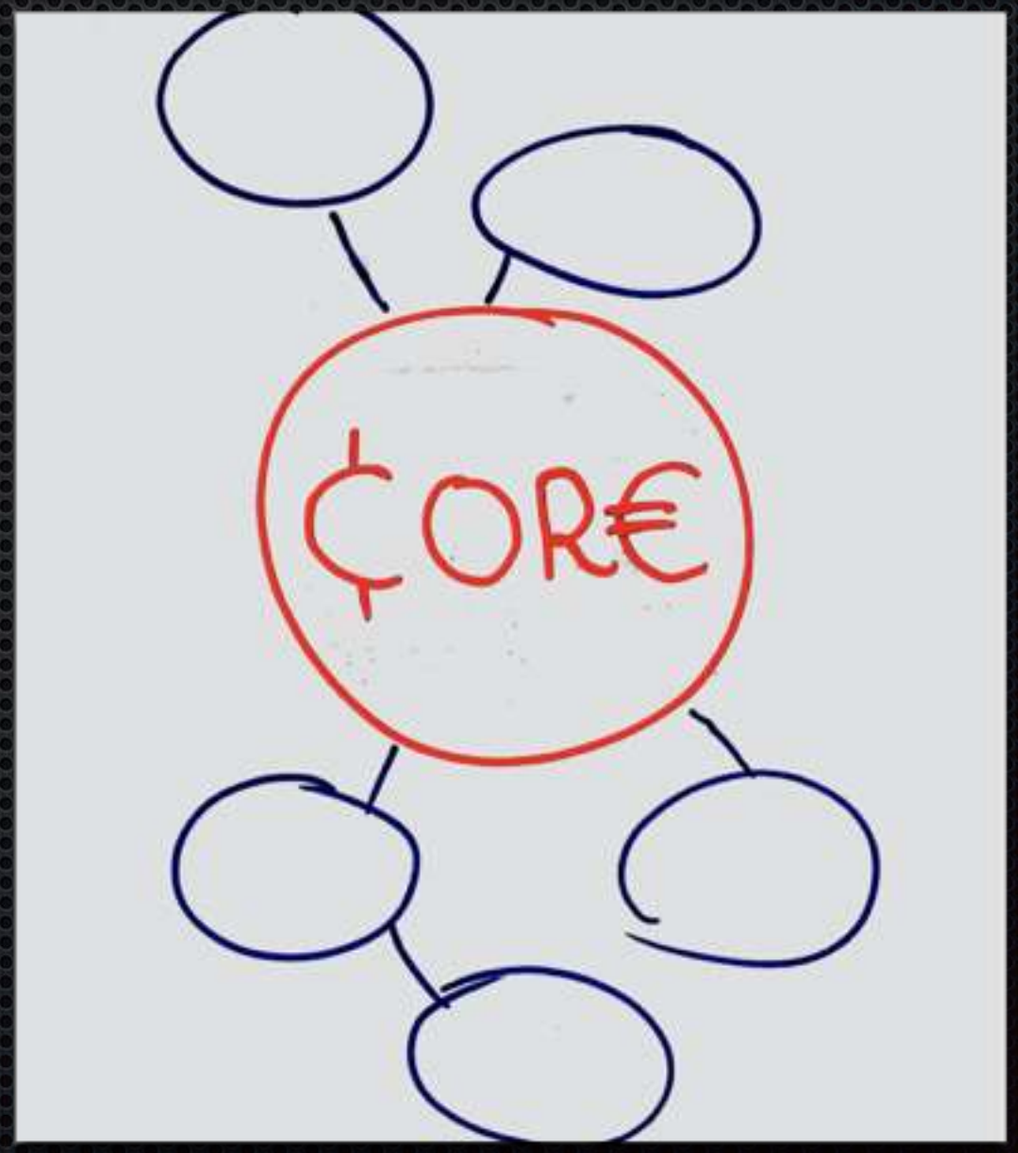
Context Map


Domain
Core
Domain
Bounded Context
Supporting
Subdomain (A)
Bounded Context
Supporting
Subdomain (B)
Bounded Context
Bounded
Context
(External)
Bounded
Context
Generic
Subdomain

Core Domain

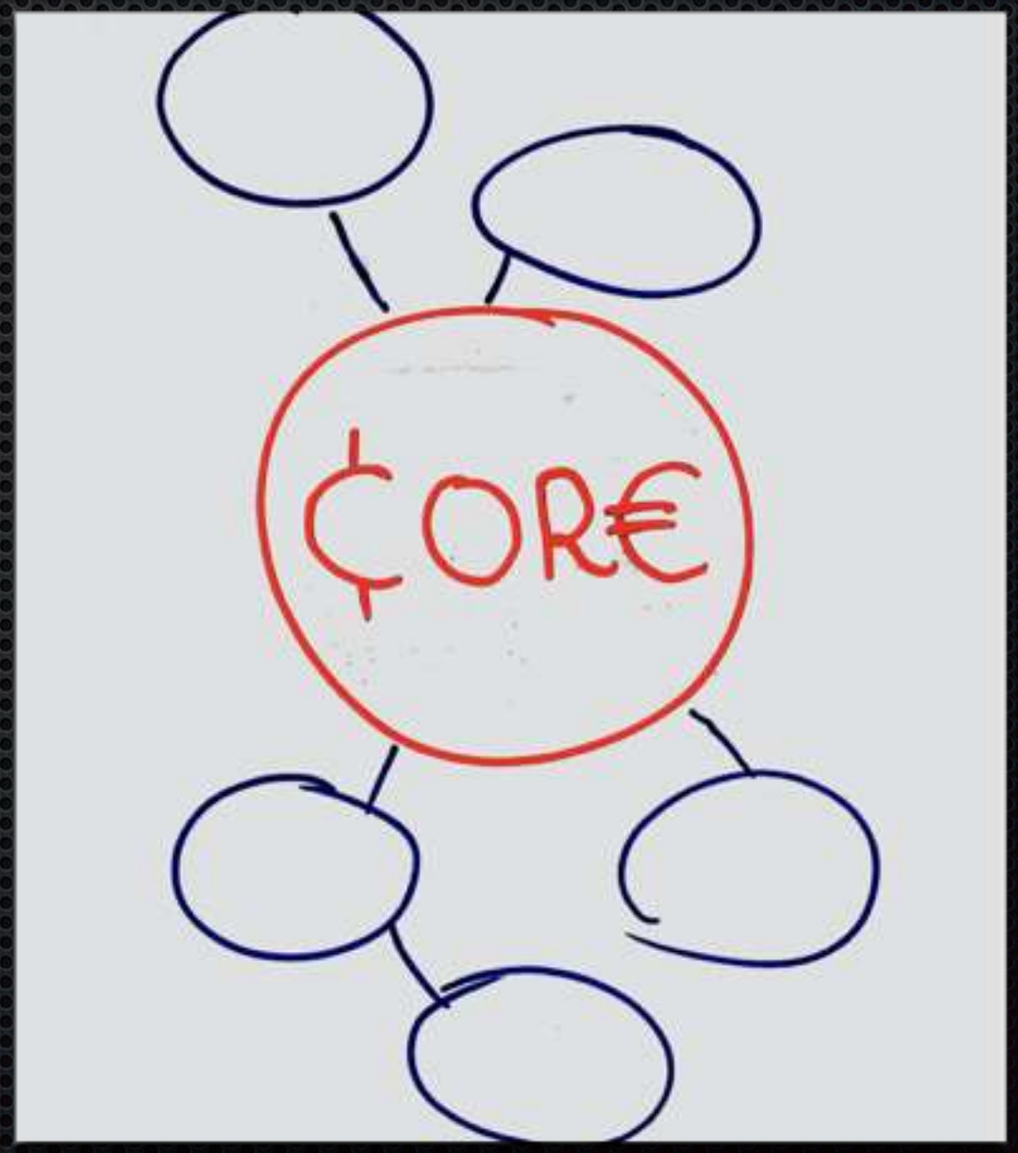
Core Domain

- Competitive advantage
- Absolutely critical part
- Distinction from competitors
- Problem space complexity



Core Domain

- Best resources
- Strong focus
- Good design

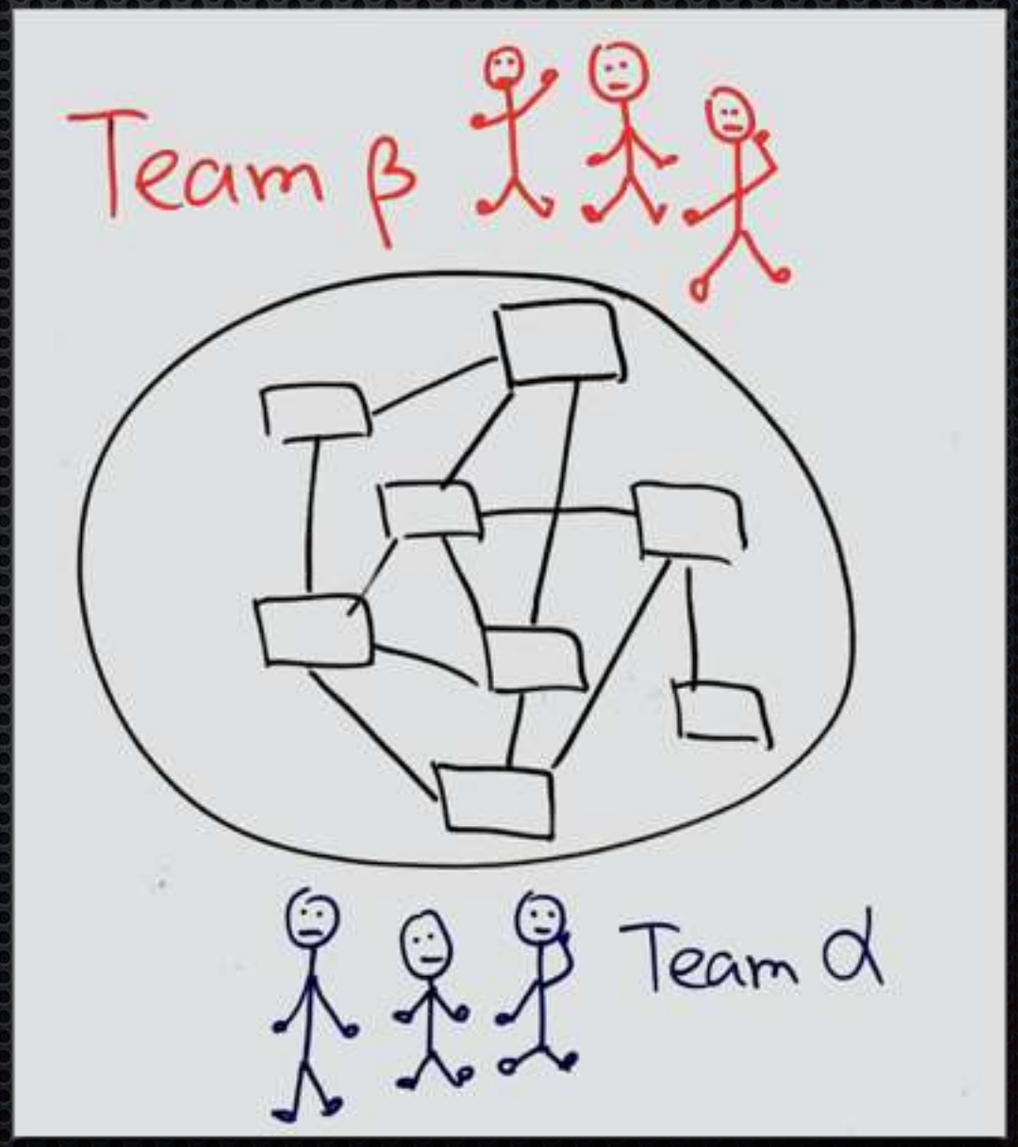


Supportive subdomain

- Candidate for off-the-shelf software
- Prefer buying versus building
- Candidate for CRUD
- Can be outsourced

Single model

- Wobbly
- Too big - hard to reason about
- Tightly coupled
- Hard to test
- Requires coordination

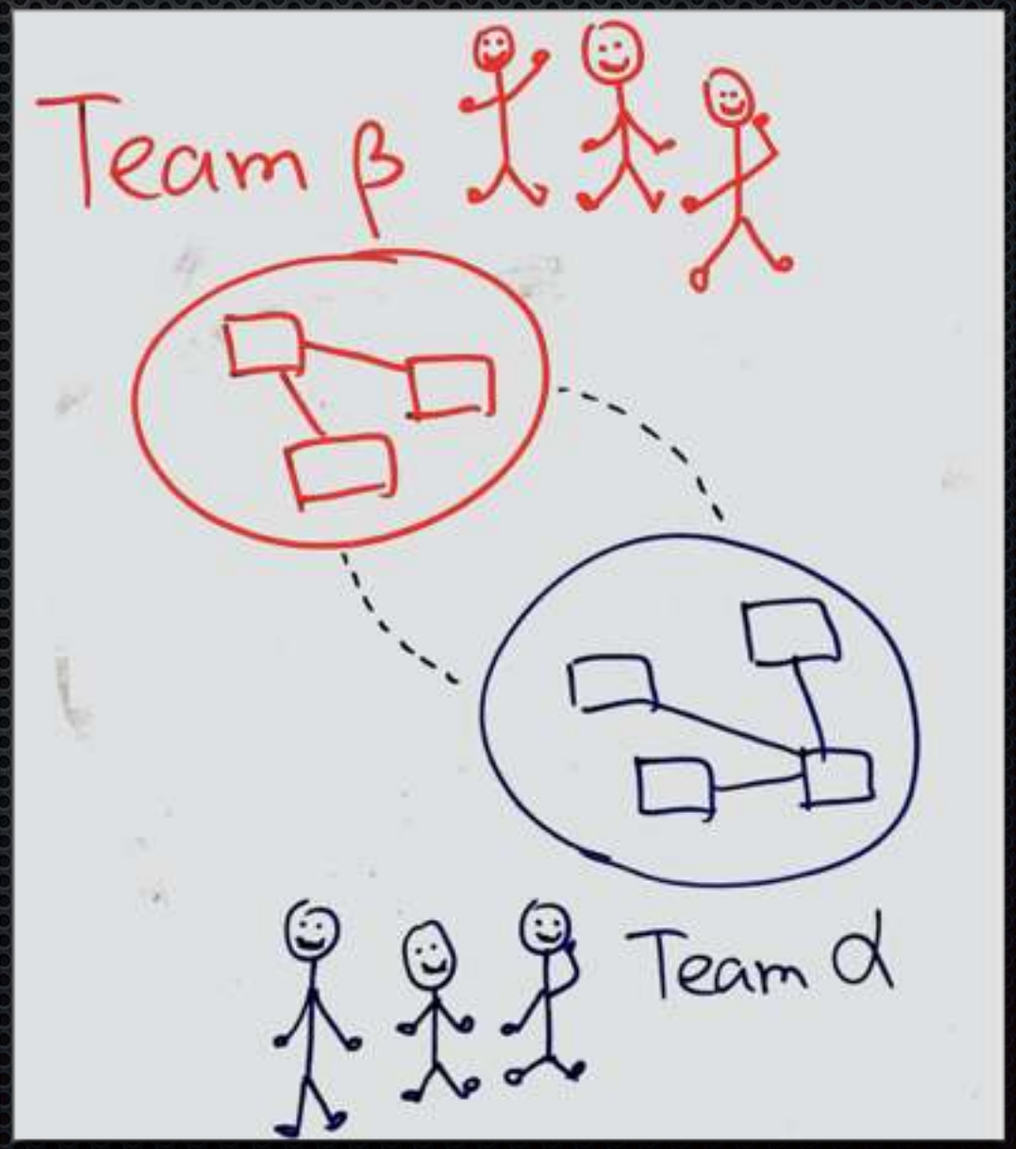


Coordination

Reality is different

Contextual models

- Focused
- Small - easier to reason about
- Decoupled
- Easy to change
- Enables **autonomy**



Autonomy

- Bounded context creates autonomy
- Allows teams working in parallel
- Little or no coordination
- Independent release and feedback cycle

“DRY is inside the bounded context,
NOT the whole system.”

–Eric Evans

DO NOT

- Don’t concentrate on technicalities
- Don’t expect the ideal model, expect many iterations
- Don’t over-engineer
- Protect context boundaries, don’t allow shortcuts and leaking abstractions from other contexts
- Respect and don’t ubiquitous language

"Good design is imperfect design."

–Eric Evans

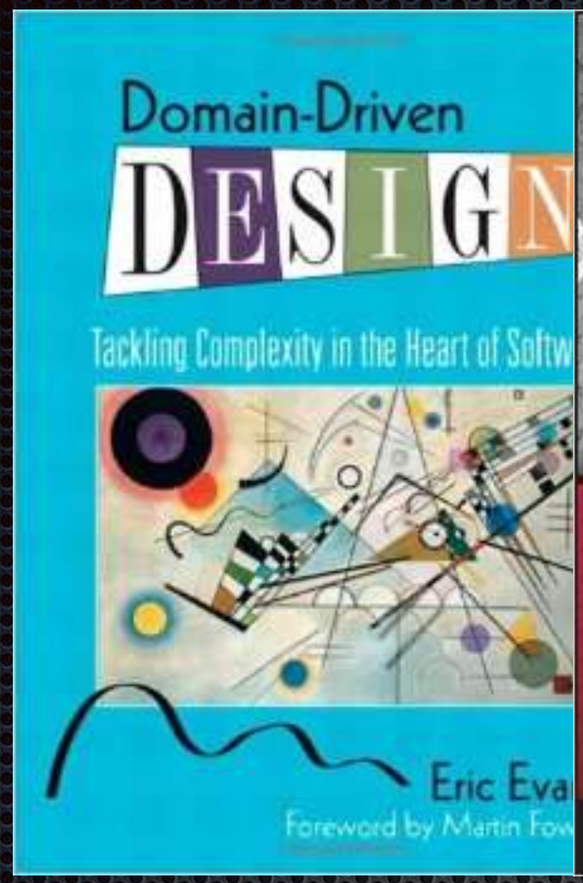
DO

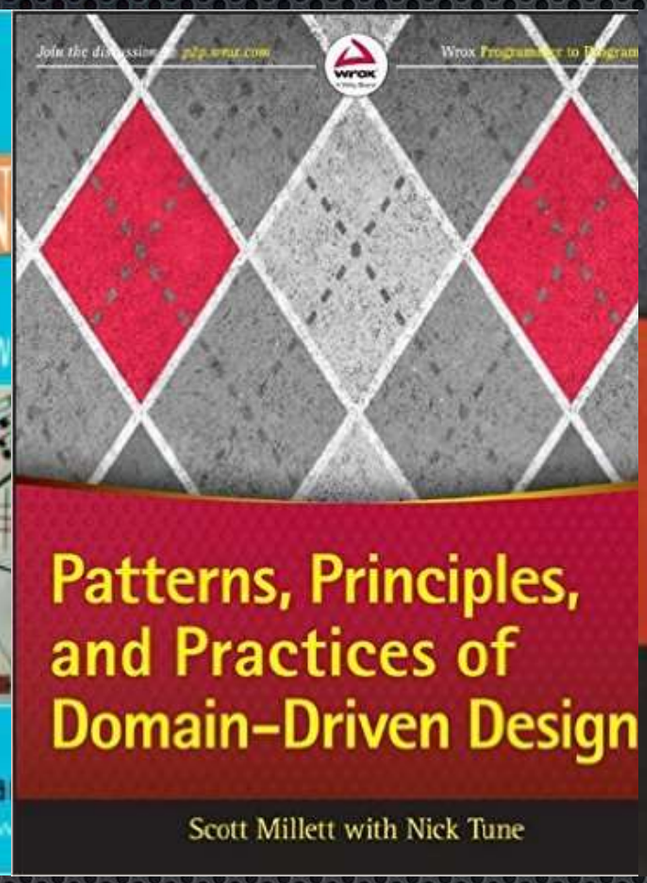
- Speak to domain experts (using their language)
- Crunch knowledge
- Find core domain
- Refactor towards deeper insight

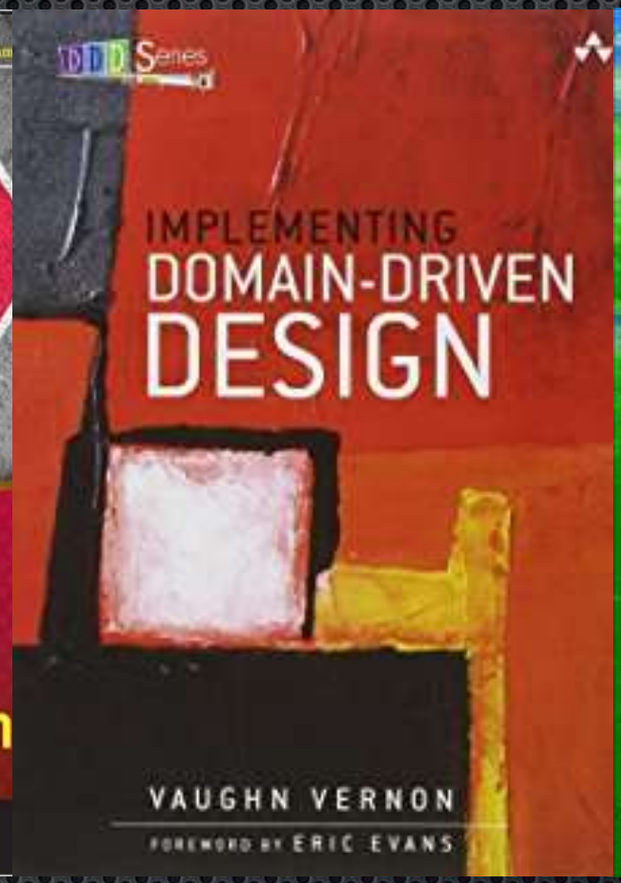
“Software development is a learning process.
Working software is a side effect”

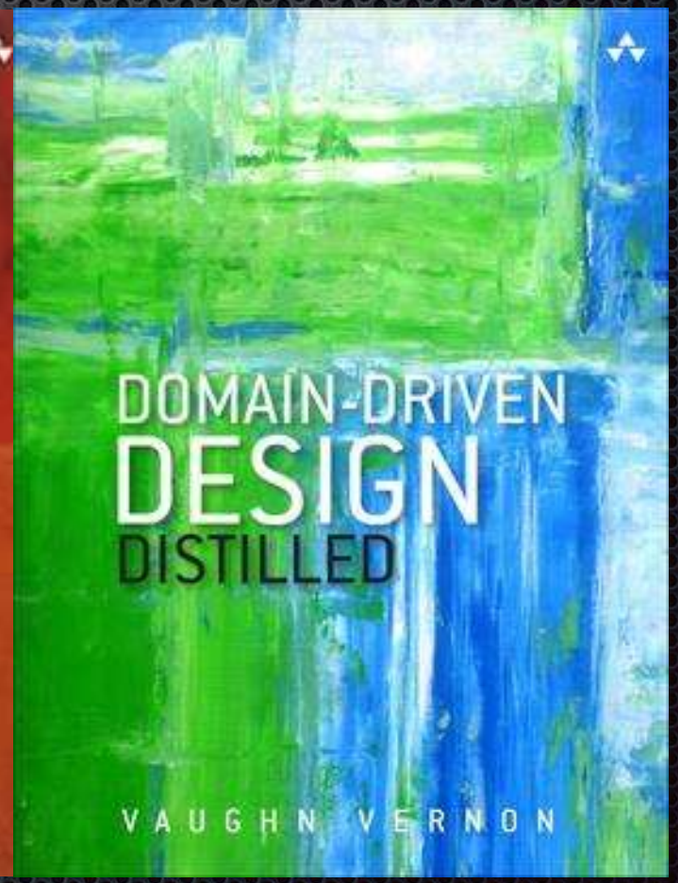
–Alberto Brandolini

Reading


Domain-Driven
DESIGN
Tackling Complexity in the Heart of Softw
Eric Eva
Foreword by Martin Fow


Patterns, Principles, and Practices of Domain-Driven Design
Scott Millett with Nick Tune


IMPLEMENTING
DOMAIN-DRIVEN
DESIGN
VAUGHN VERNON
FOREWORD BY ERIC EVANS


DOMAIN-DRIVEN
DESIGN
DISTILLED
VAUGHN VERNON