model driven software engineering

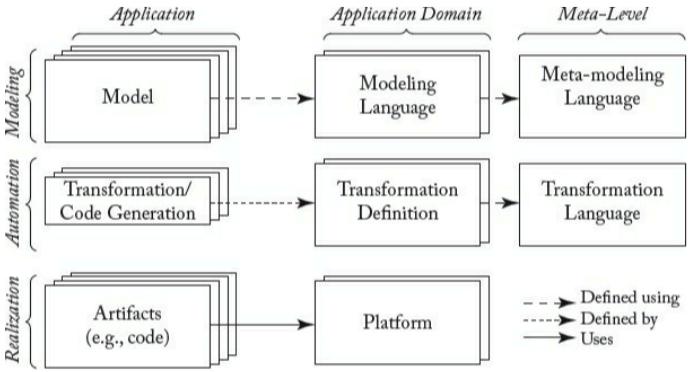
mdse in practice

4DV651

mdse - framework

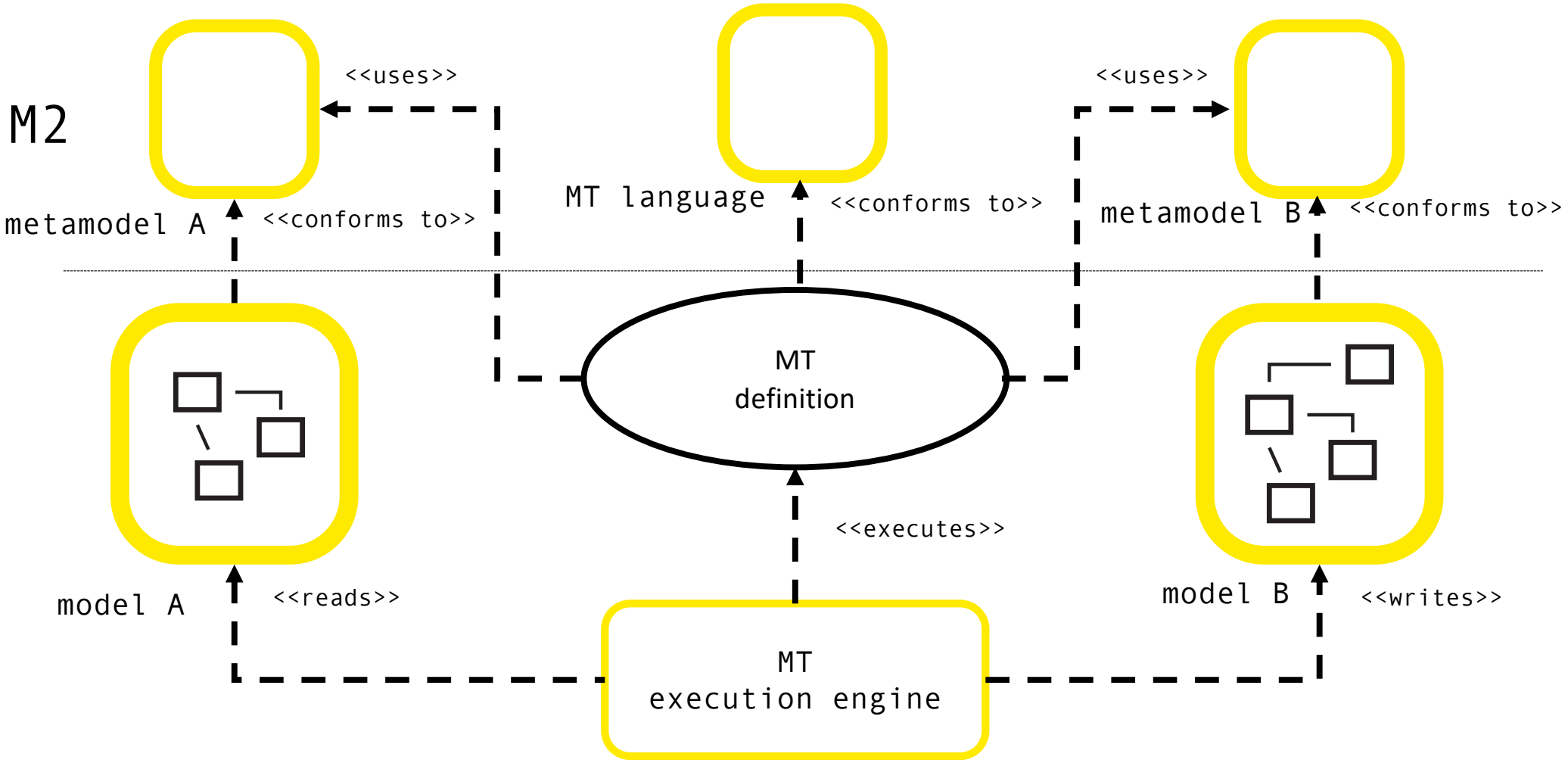
conceptualization

implementation



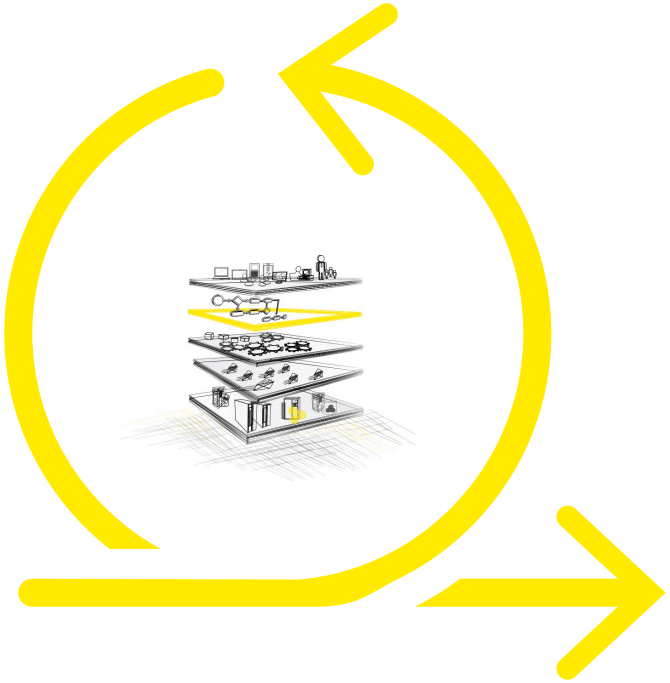
Model-Driven Software Engineering in Practice, Second Edition

transformations



model driven software engineering?

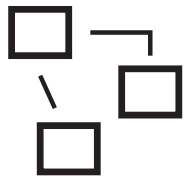
SOFTWARE is MODELS



spaces - problem & solution

spaces - problem & solution

problem space



domain model

solution

design

implementation

execution

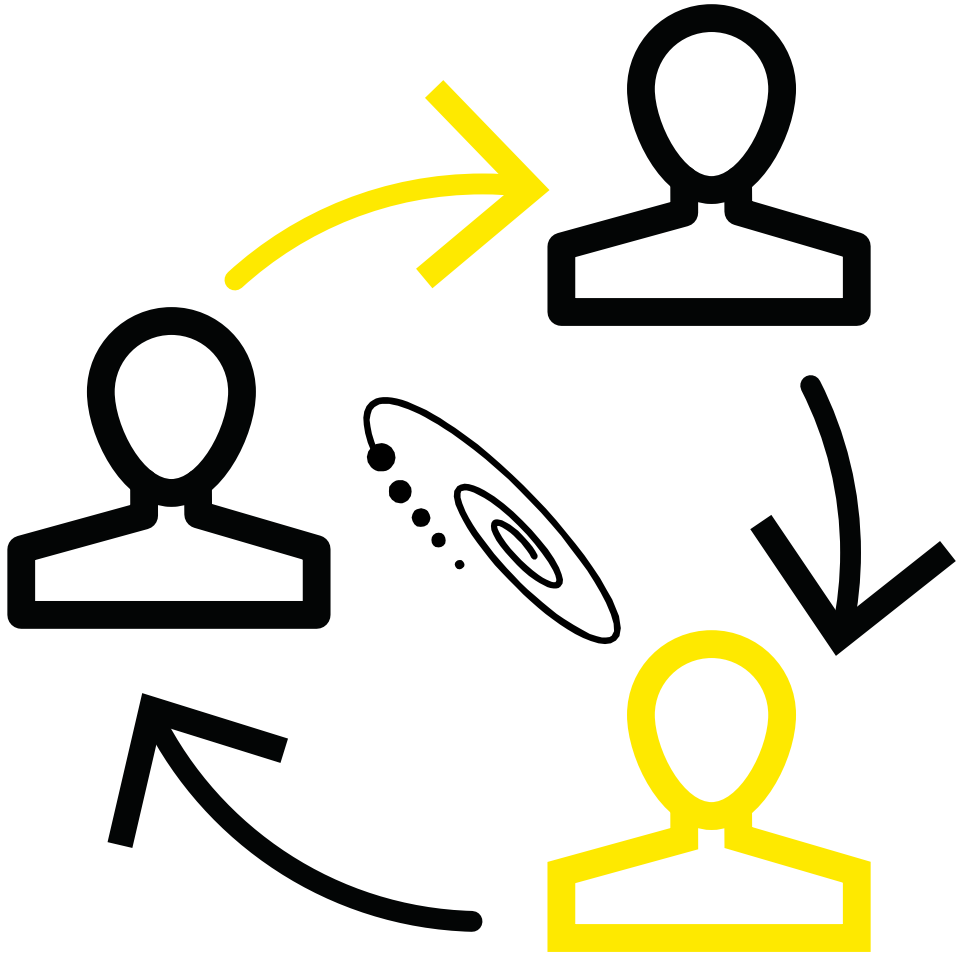
rapid application development

a development life cycle designed to give much faster development and higher quality systems than the traditional life cycle.

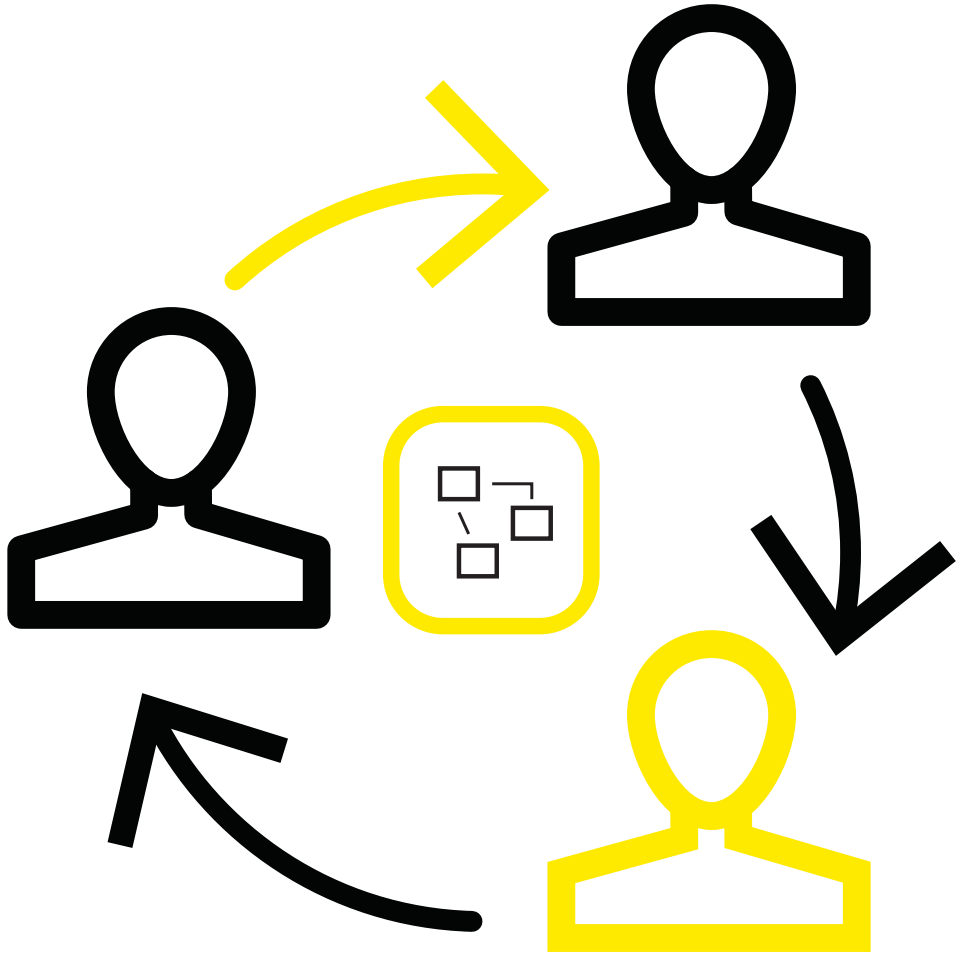
designed to take advantage of powerful development software like CASE tools, prototyping tools and code generators.

Key objectives – High Speed, High Quality and Low Cost

people centered

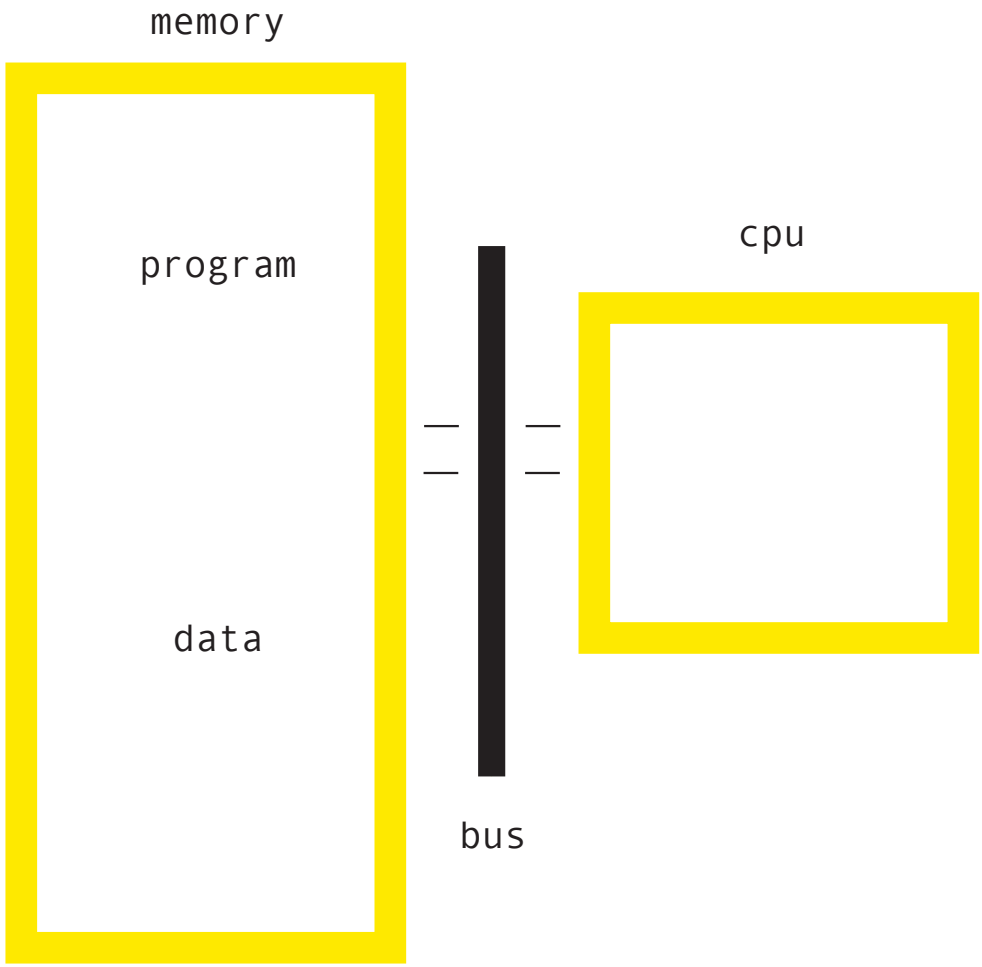


model centered



1st model

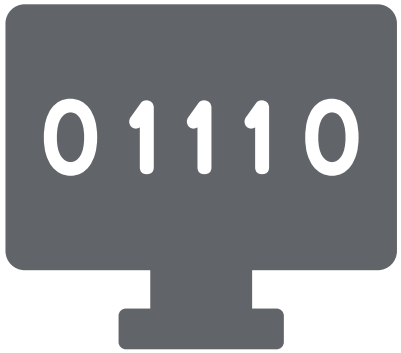
von Neumann architecture



program and data are stored in memory during execution.

program instructions are executed sequentially

instruction set



Instruction Set	Instruction
0001	Move
0010	Compare
0011	Bit test
0100	Bit clear
0101	Bit set
0110	Add
0111	See group 10
1000	See groups 11, 13, 14
1001	Move byte

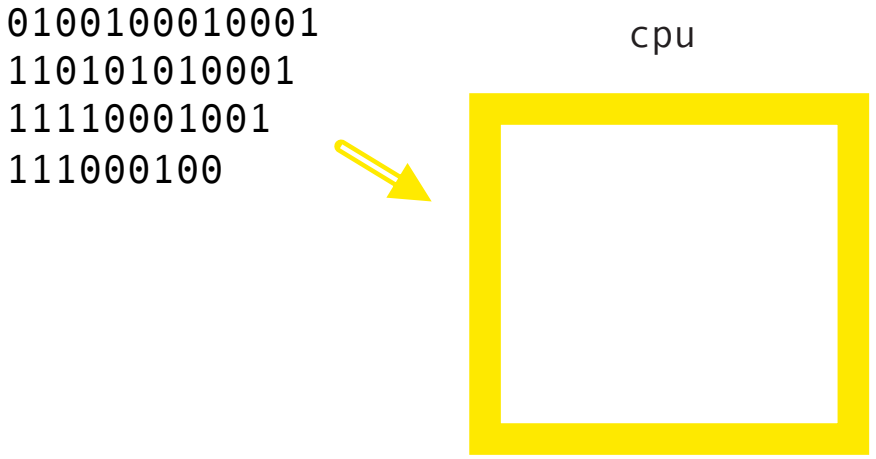
a vocabulary - list of instructions which can be executed by the CPU

the **only** instructions the CPU understand

1st generation languages

programming computers using the CPU's instruction set

aka **machine language**



1st generation

advantages of 1st gen.
software programs execute (run) relatively very quickly

software programs are relatively small in size (Insignificant advantages today)

disadvantages of 1st gen.
difficult to write, very detailed and takes a long time

difficult to read

difficult to debug

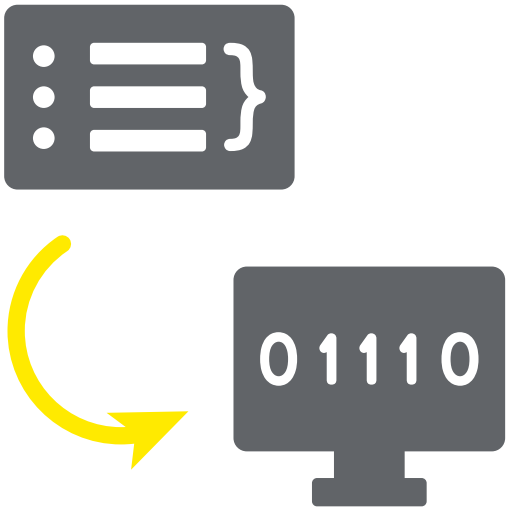
suitability for RAD?

2nd generation languages

```
mov RAX, 0
mov RSI, str
mov RDI, fmt_in
call scanf

mov RAX, 0
mov RBX, 0
mov RCX, 0

LOWER_CASE_CONVERT:
cmp qword[str + RCX], 0
jz FINISH_CONVERT
mov RAX, 0
mov AL, [str + RCX]
```



assembly Language – English-like mnemonics that are equivalent to the CPU's instruction set

data definitions (type system)

requires a translator

2nd generation

advantages of 2nd Gen.

easier to read than 1st gen.

easier to write than 1st gen.

easier to debug than 1st gen.

disadvantages of 2nd Gen.

still very difficult to write programs

suitability for RAD?

3rd generation languages

languages which are somewhere between machine language and the human language

provide constructs for data abstraction

program abstraction

3rd generation

advantages of 3rd Gen.

easier to read, write and debug

faster creation of programs

simplifies reuse

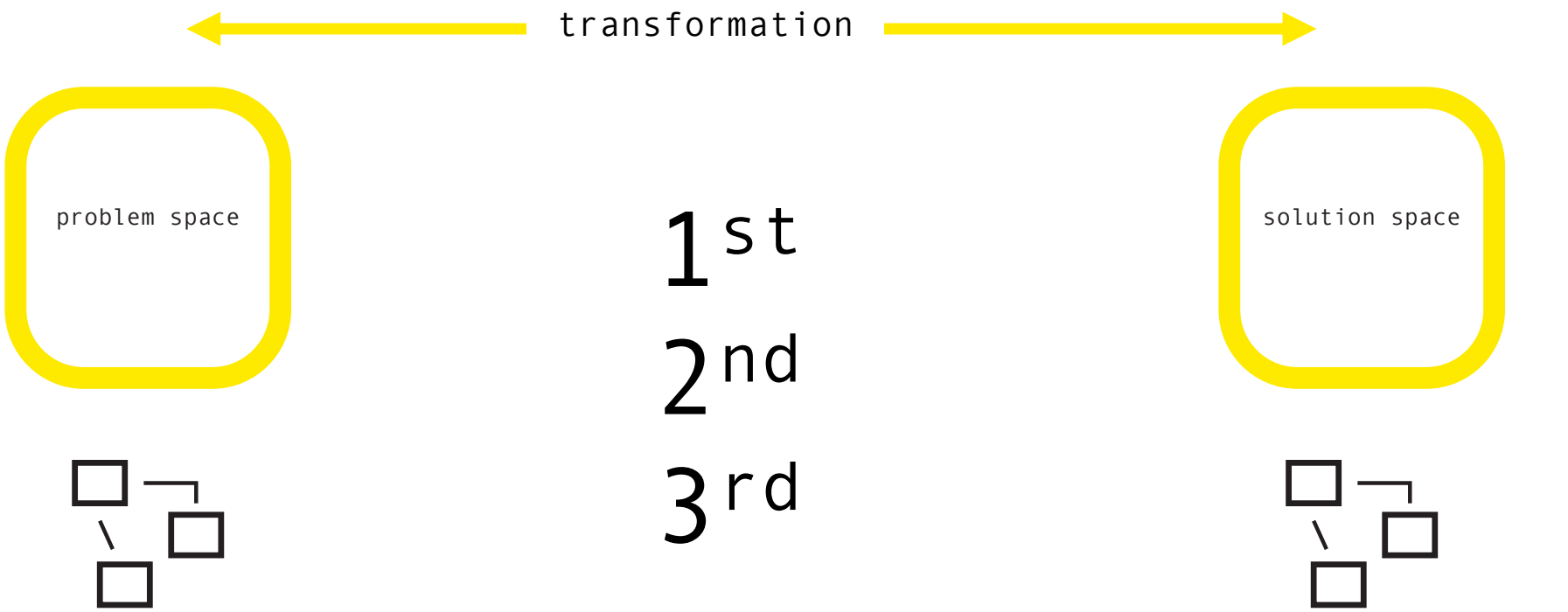
disadvantages of 3rd Gen.

still not a tool for the average user to create software programs

requires very good knowledge of programming and of the language

suitability for RAD?

distance – problem & solution

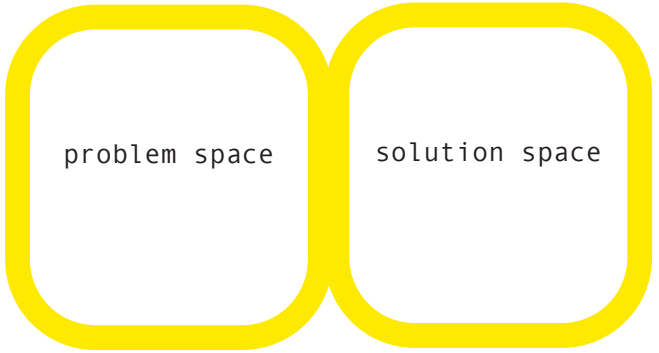


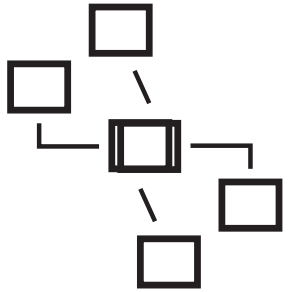
4th generation languages

languages which are closer to the problem space

closer to natural languages

uses English phrase structure



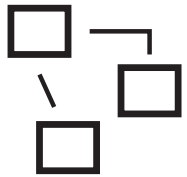


```
SELECT CustomerName, City FROM Customers;
```

domain + implementation model

model and language – SQL

data & operation



relational model


```
SELECT CustomerName, City FROM Customers;
```

interpreter