worldline

Techforum eXplore 2015

3rd June, Lille, France

MDE Applied

Model, Meta-Model & Code Generation on Industrial Cases.

Pedro J. Molina, PhD.

Chief Research Officer

pjmolina@Icinetic.com

@pmolinam

icinetic

whoami

- Chief Research Officer for **Icinetic** (MDE company)
- 6 years as Manager & Software Architect at **Capgemini**
- 5 years as Research & UI Lead Engineer at **CARE Technologies**
- **PhD** in Modelling UIs and Code Generation for Business Apps
- Working the last 20 years on MDD, Code Gen. Conferences
  - 2 Patents in the USPTO and another one on-process

@pmolinam

open questions

- Do you apply MDD techniques?
- Did MDD worked for you?

How much we can do?

Developers + Tooling = Apps

impossible equation

Developers

- Grows linearly

VERSUS

Demand for Applications

- Grows at exponential rate.

The impossible equation by Prof. Jean Bezivin (Univ. Nantes)

that means

More demand for Applications than people able to create them

➔ increased prices

unless…

unless

Create **better tools** for App Development to

- Make it easy
- Make it cheaper
- Make it ***usable by non-developers***

Rise of the
Citizen Developer

citizen developer

Term coined by Gartner

Definition:

- A user operating outside of the scope of enterprise IT and its governance who creates new business applications for consumption by others from scratch or by composition.

Prediction:

- By the end of 2014, 25% of new Business Apps will be created by Citizen Developers

Complexity in Software

- Essential Complexity
- Accidental Complexity

Terms from : “Fred Brooks, 1986, No Silver Bullet”

Essential Complexity

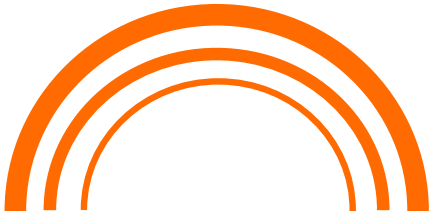
Complexity inherent to the system been designed.

> *"Everything should be made as simple as possible, but not simpler."* *(A. Einstein)*

Accidental Complexity

Any other **Extra** Complexity arisen from tools, methods, technologies, etc. used to build the system.

Programming languages, tools, frameworks… computers, devices introduce many, many ***Accidental Complexity****.*



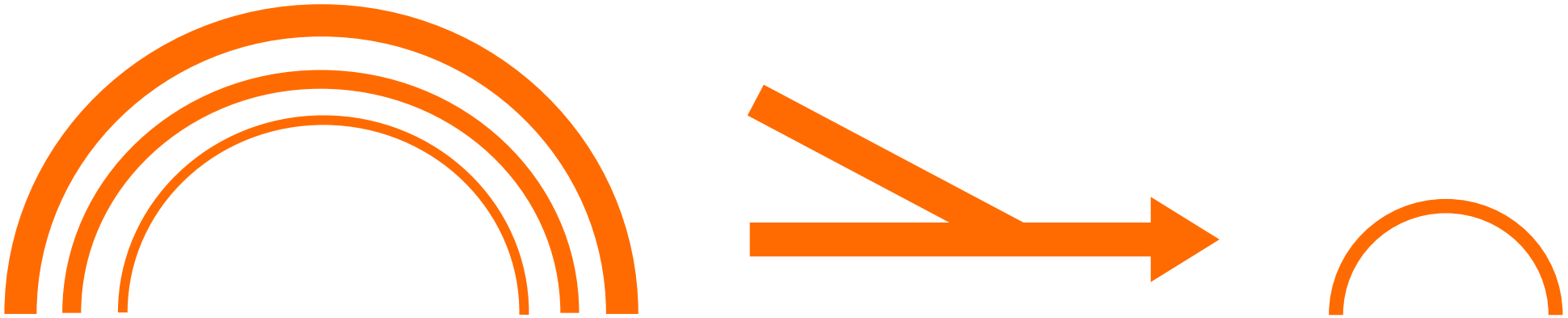
Accidental Complexity

From idea → usage

Idea → design → build → test → debug → provision → deploy → usage

Can we do something to speed up this critical delivery path?

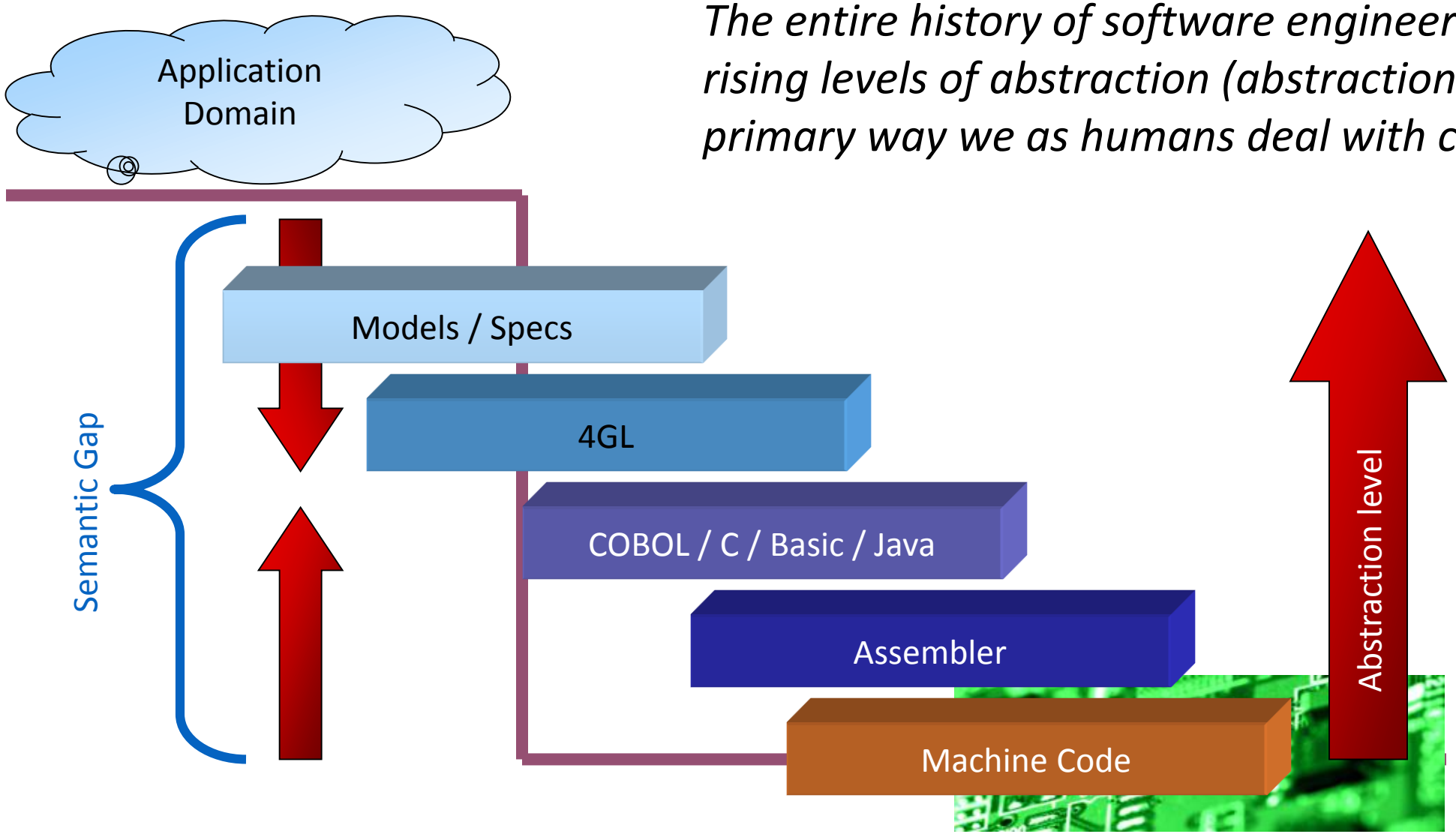
The ultimate “noble” Quest for MDE



MDE helps to reduce **Accidental Complexity**.

Makes our life easier!

Abstraction Levels



The entire history of software engineering is one of rising levels of abstraction (abstraction is the primary way we as humans deal with complexity).

Grady Booch

samples

D←mos

- Essential
- AppNow
- Radarc Online
- Windows Phone AppStudio

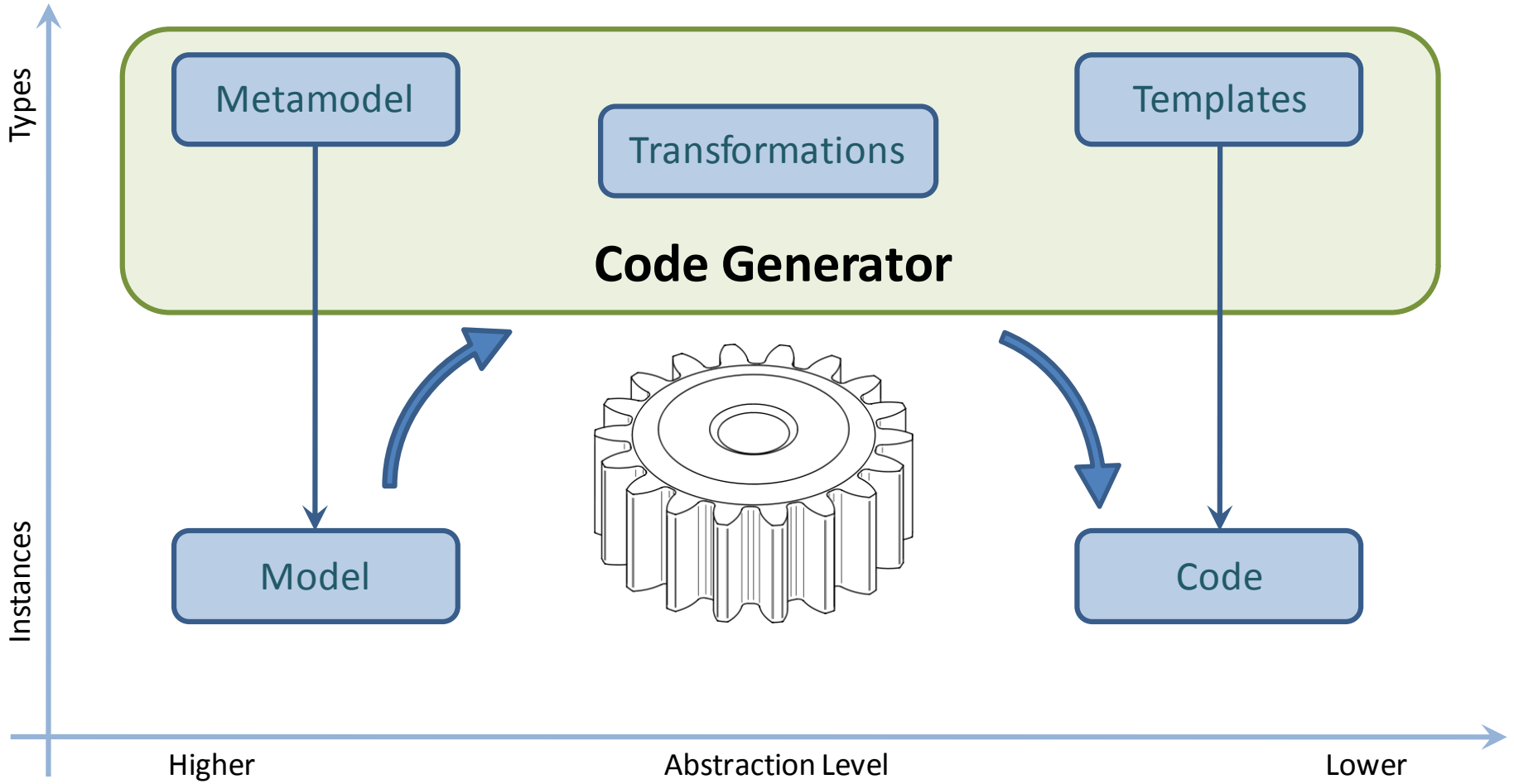
Essential

Project for **lean** Model, Meta-model and Code Generation

Base foundations used in MDD commercial products

http://pjmolina.com/essential

Essential Primitives



App Now

App Now

- Cloud-based **Microservices** in minutes

https://appnow.radarconline.com

Radarc Online

radarc online

Native Mobile Apps for Citizen Developers

https://www.radarconline.com

Windows Phone App Studio

Windows | App Studio

Microsoft Corporation

App Builder for Windows Phone

Windows App Studio

Some numbers in 2014:

- *350.000 users in 7 months*
- *300.000 projects*
- *20.000 apps published in the MS Store (25% of the total in such period)*

- Source: http://blogs.windows.com/buildingapps/2014/02/20/new-ui-and-capabilities-for-windows-phone-app-studio-beta-developers/

mdd

Model Driven Development

Definition:

*The usage of **M**odels as the main artefacts to **D**rive the software **D**evelopment.*

mdd

- Cost saving
- Increased quality

- Why UML tools failed?
- Why people tend to avoid formal methods?
- Why Code-Gen is a failed and forget technique for a lot of people?

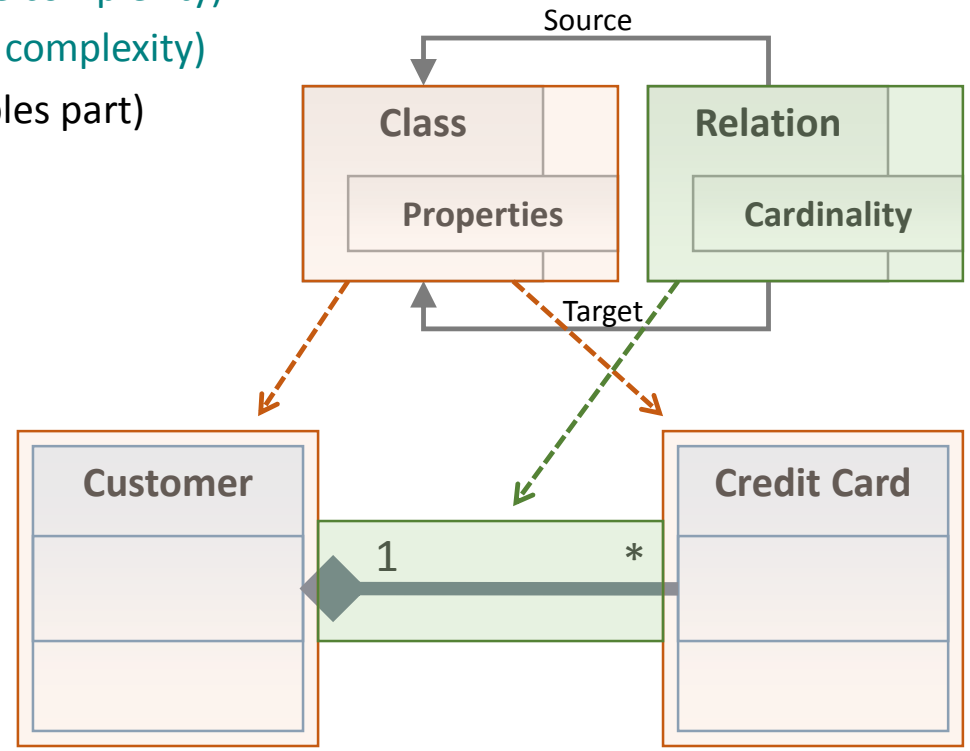
mdd principles reloaded

- KISS
- DRY
- Separation of Concerns
- Cost/Benefit approach to Software Development
- Executable Models
- Canonical
- Do not model nothing not (yet) runnable

what's a model?

- A model allows
  - the description of a family of problems for a domain
  - Having the abstraction level carefully selected to:
    - Discard irrelevant details (reduce complexity)
    - Discard constant details (reduce complexity)
    - Explicit important details (variables part)

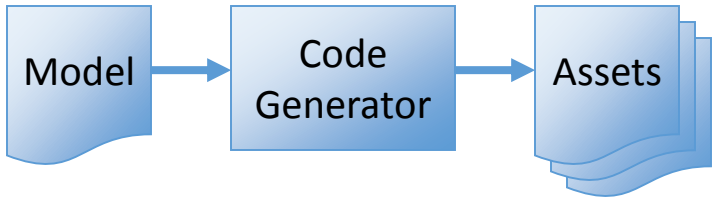
- What's a meta-model?
  - A model describing model.



meta-models

- MOF levels
- Minimal Meta-Model
  - Object, Property, Relation
- Building from Scratch
  - Essential

code generation



Definition:

The automated synthesis of SW assets like source code, documentation or models using models as input.

Novak's rule

"Automatic Programming is defined as the synthesis of a program from an specification.

If automatic programming is to be useful, the specification must be smaller and easier to write than the program would be if written in a conventional programming language."

G.S. Novak

Commonality / Variability

- Family of programs

  (D. Parnas)

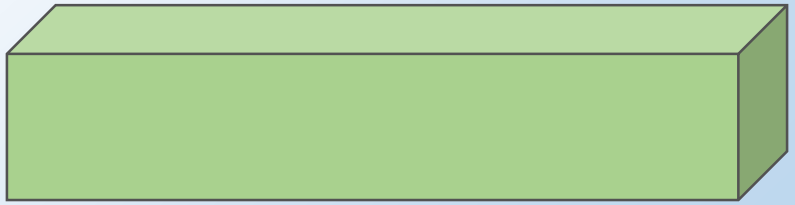
  - Common part
    - Standard, Fixed.
    - Implementable in common & shared base libraries
  - Variable part
    - Specify in the model
    - Generable

Separation of Concerns (SoC)

Know-How captured in two separated buckets:

What

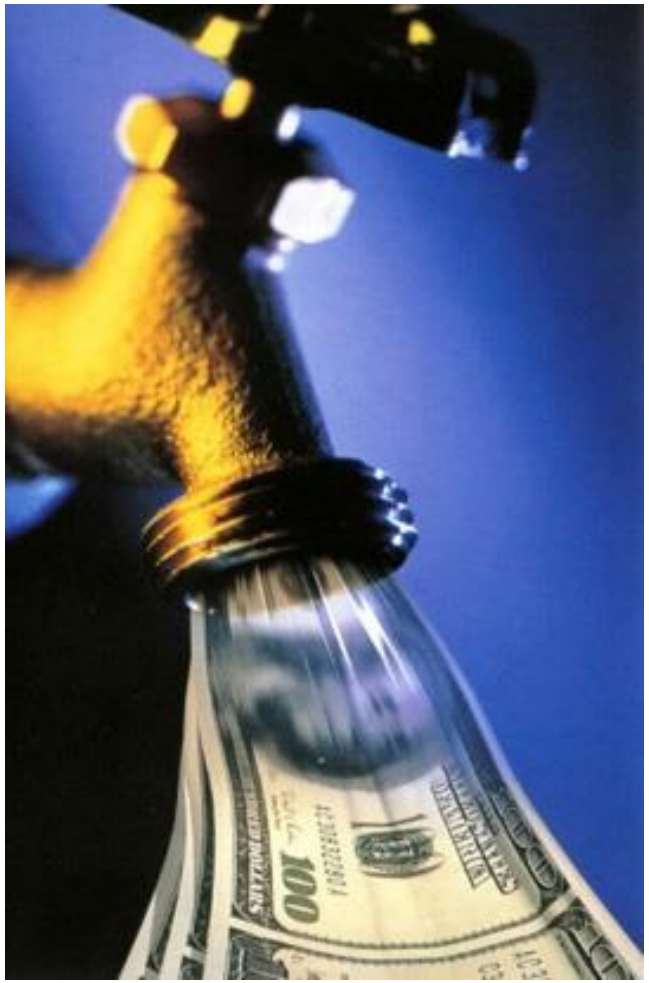
- **Business Know-How:** captured in form of models (**specifications**): isolated from technological issues



How

- **Technological Know-How:** captured & encapsulated in form of best practices, frameworks, templates & code patterns in code generators & interpreters.

ROI

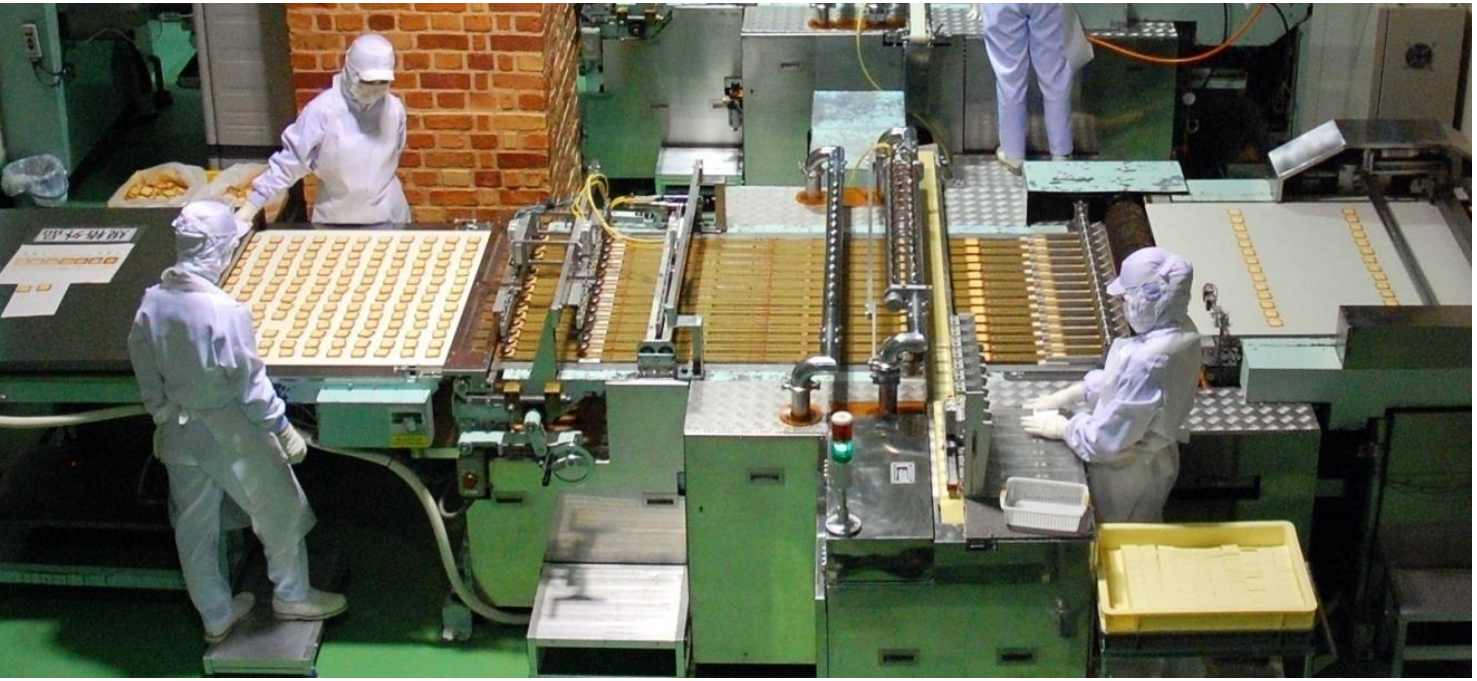


- Economies of Scale
- Economies of Scope
- Economics of MDSD

- Development Life Cycle Impact
- Quality

Economies of Scale

- **Economies of Scale**
  - *The condition where few inputs, as effort and time, are needed to produce big quantities of a unique output.* [Wit96]

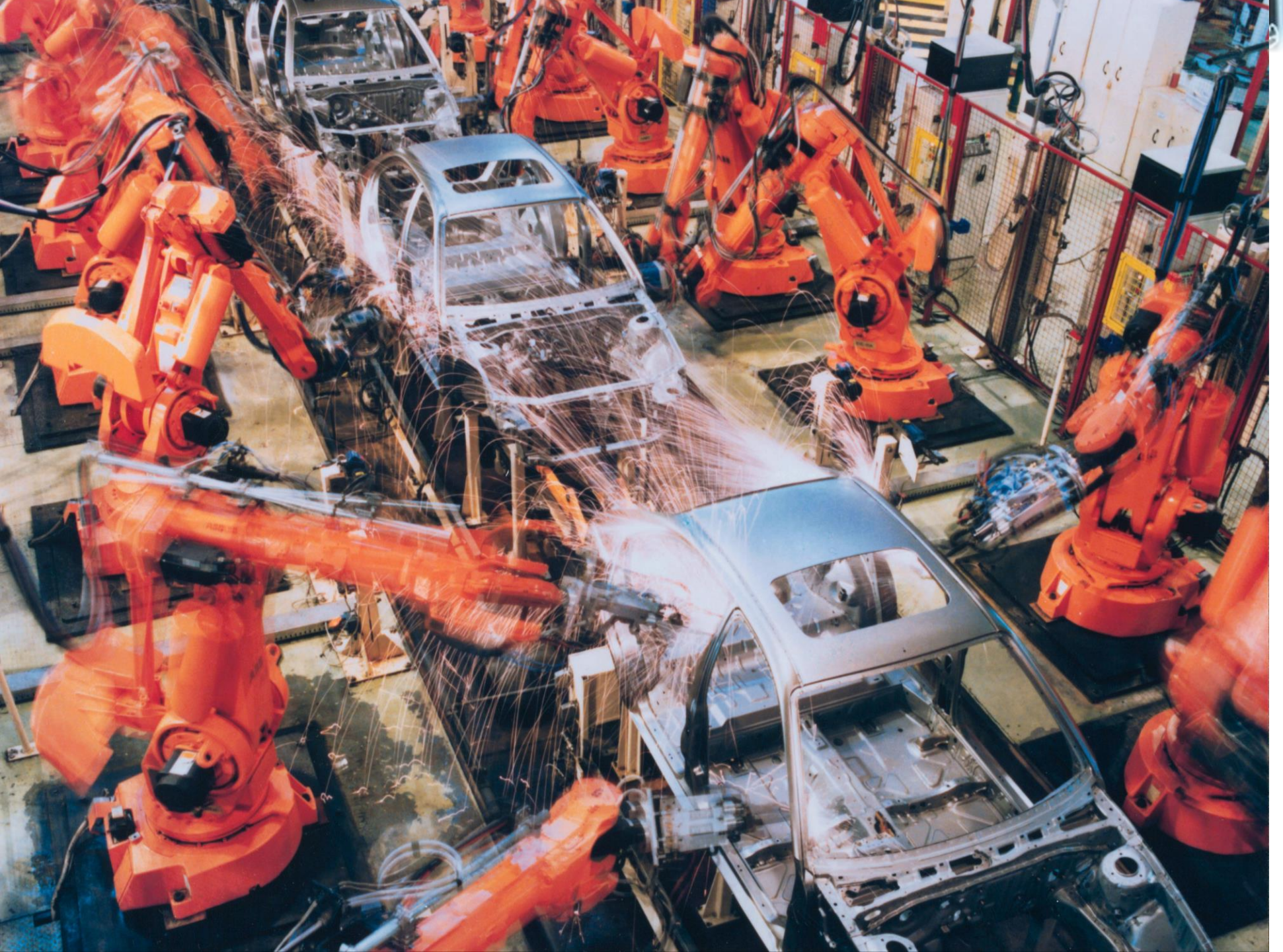


Japanese Cookie Factory. Production Line.

But: **Can't be applied to SW!**

Once the SW is produced

Copy cost is = 0 €!

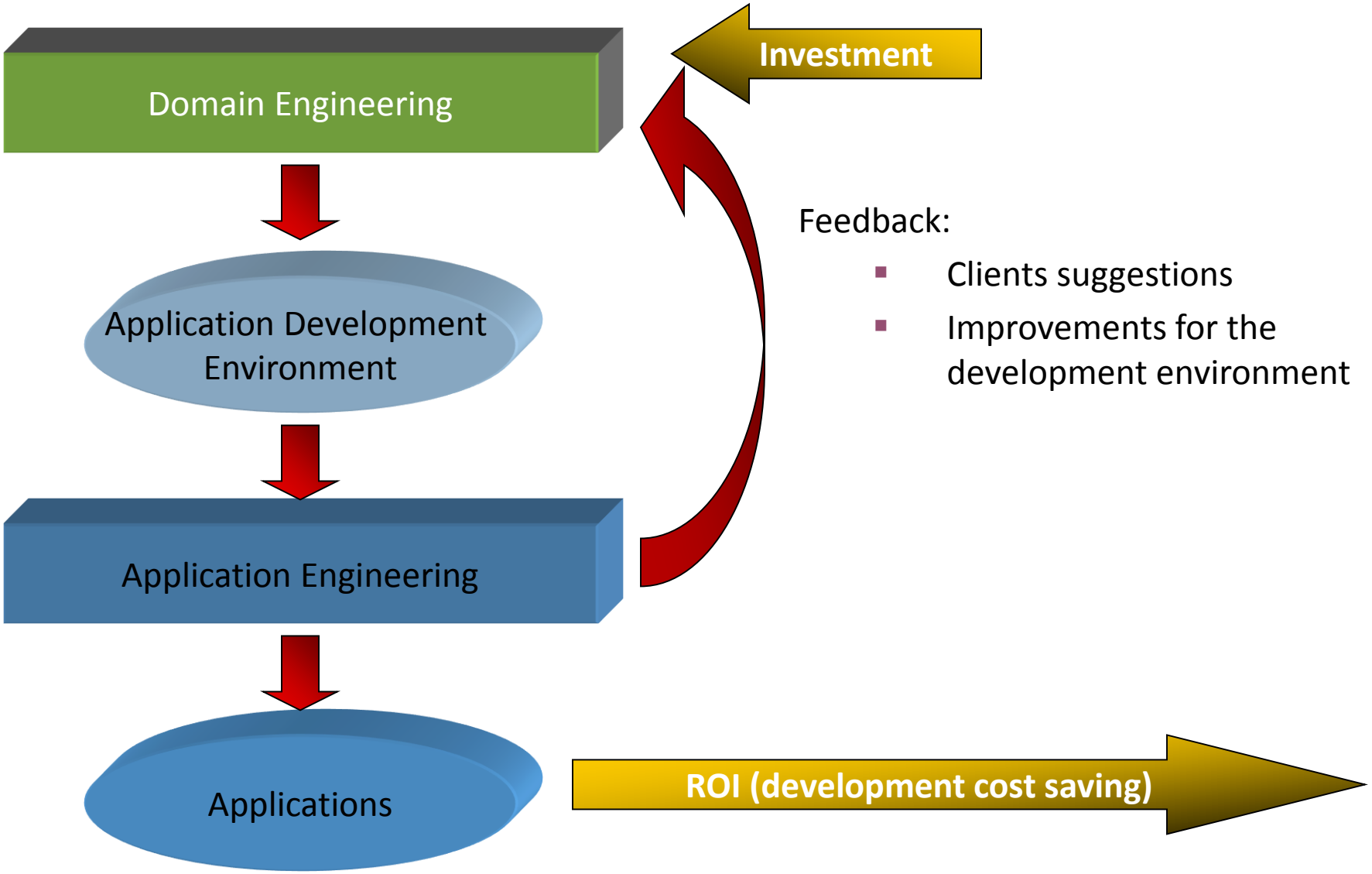


Economies of Scope

- **Economies of Scope**
  - *The condition where few inputs, as effort and time, are needed to produce a great variety of outputs. It is produced more added value producing in the same line different outputs. To produce each output independently creates an overcost in the common parts.*

  - Economy of Scope occurs when the cost of combining two or more products in a unique product line is lower than producing them independently. [Wit96]

MDSD: Economic Model



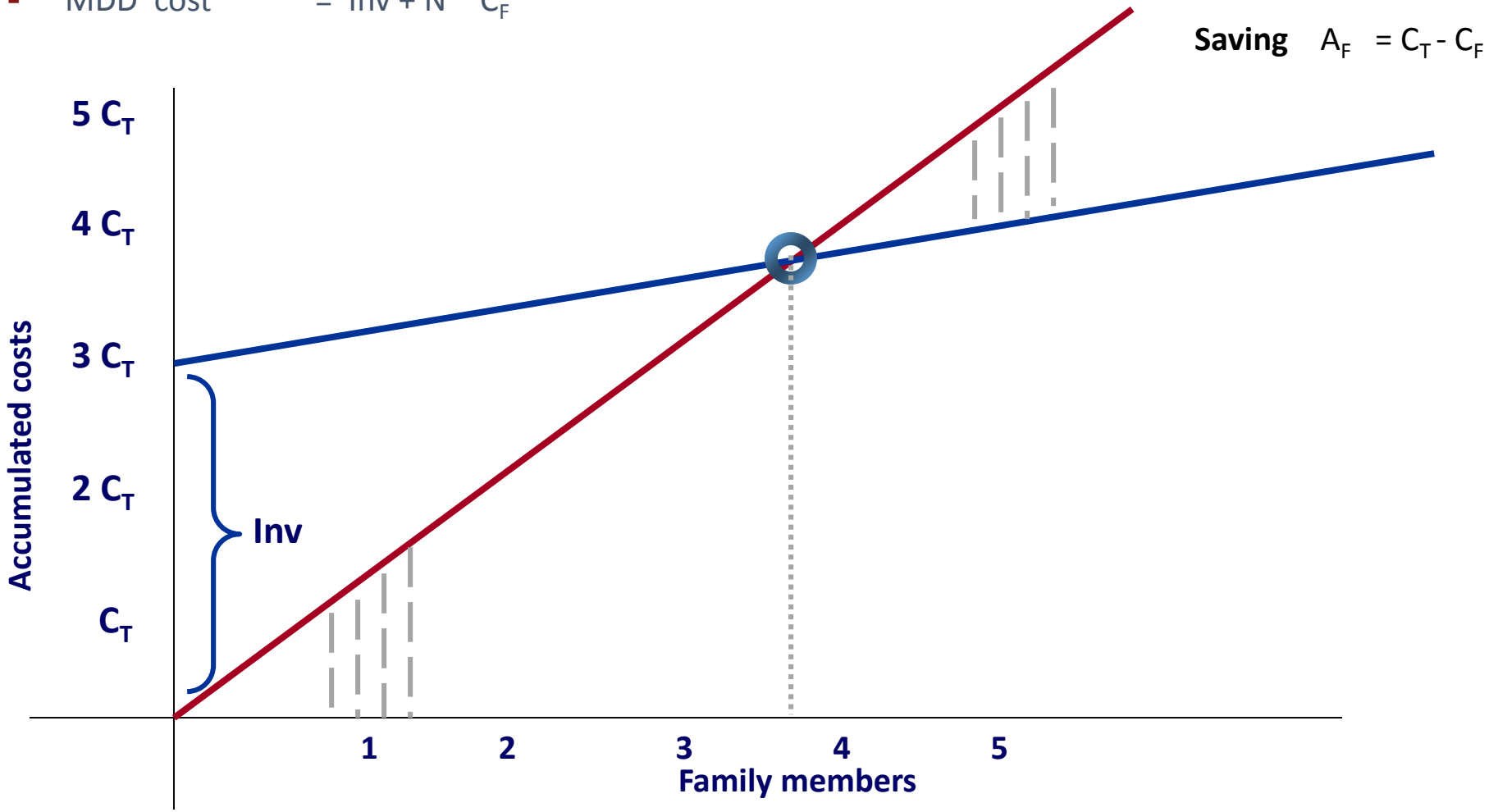
Feedback:

- Clients suggestions
- Improvements for the development environment

MDSD: Economic Model

- Traditional Cost = $N * C_T$
- MDD cost = $Inv + N * C_F$

Saving $A_F = C_T - C_F$



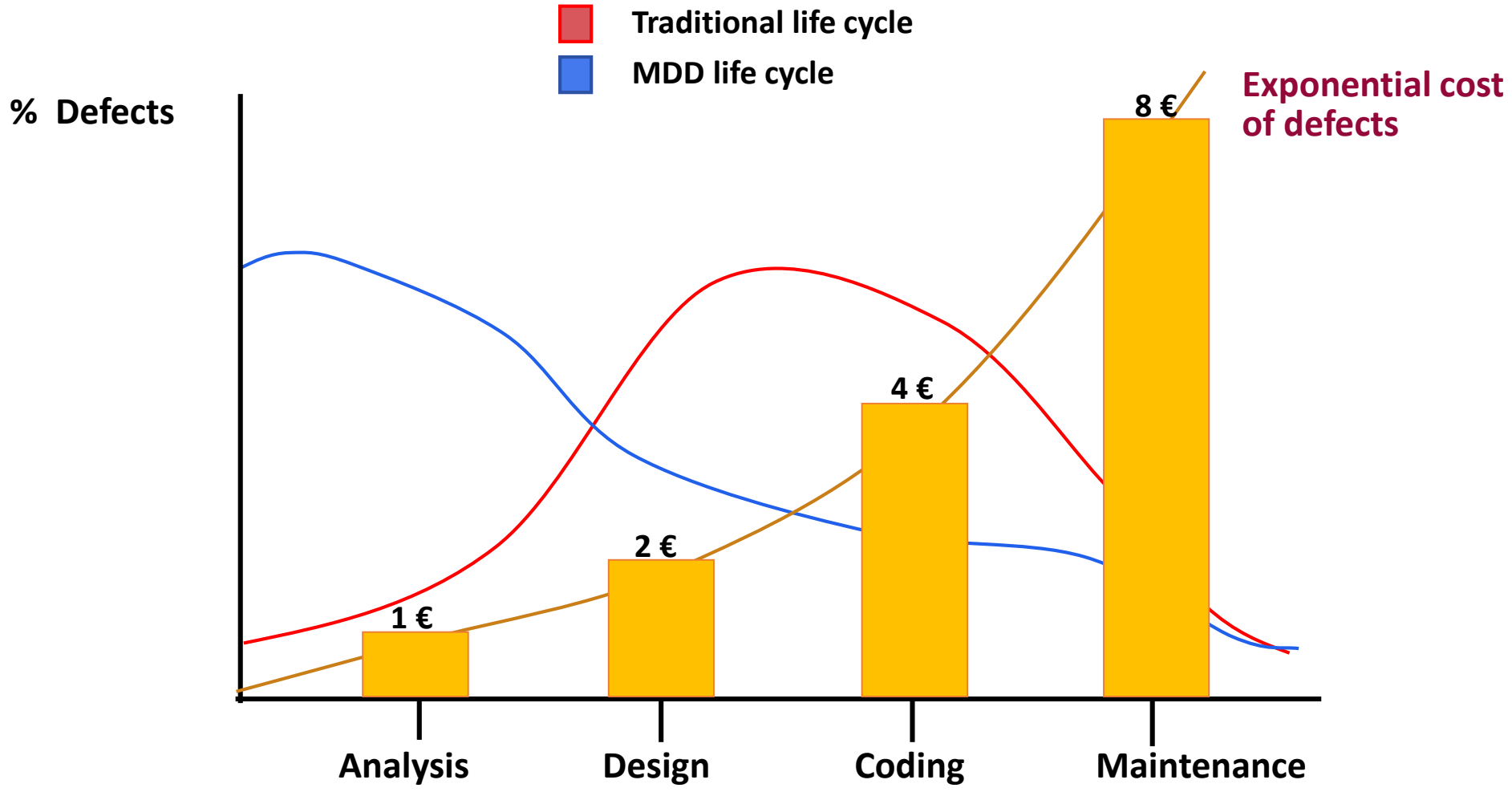
SW Life-cycle Impact

- More time in analysis and design tasks
- Less time in coding

- Less defect, more Quality
- Improved productivity
  - Order of magnitude

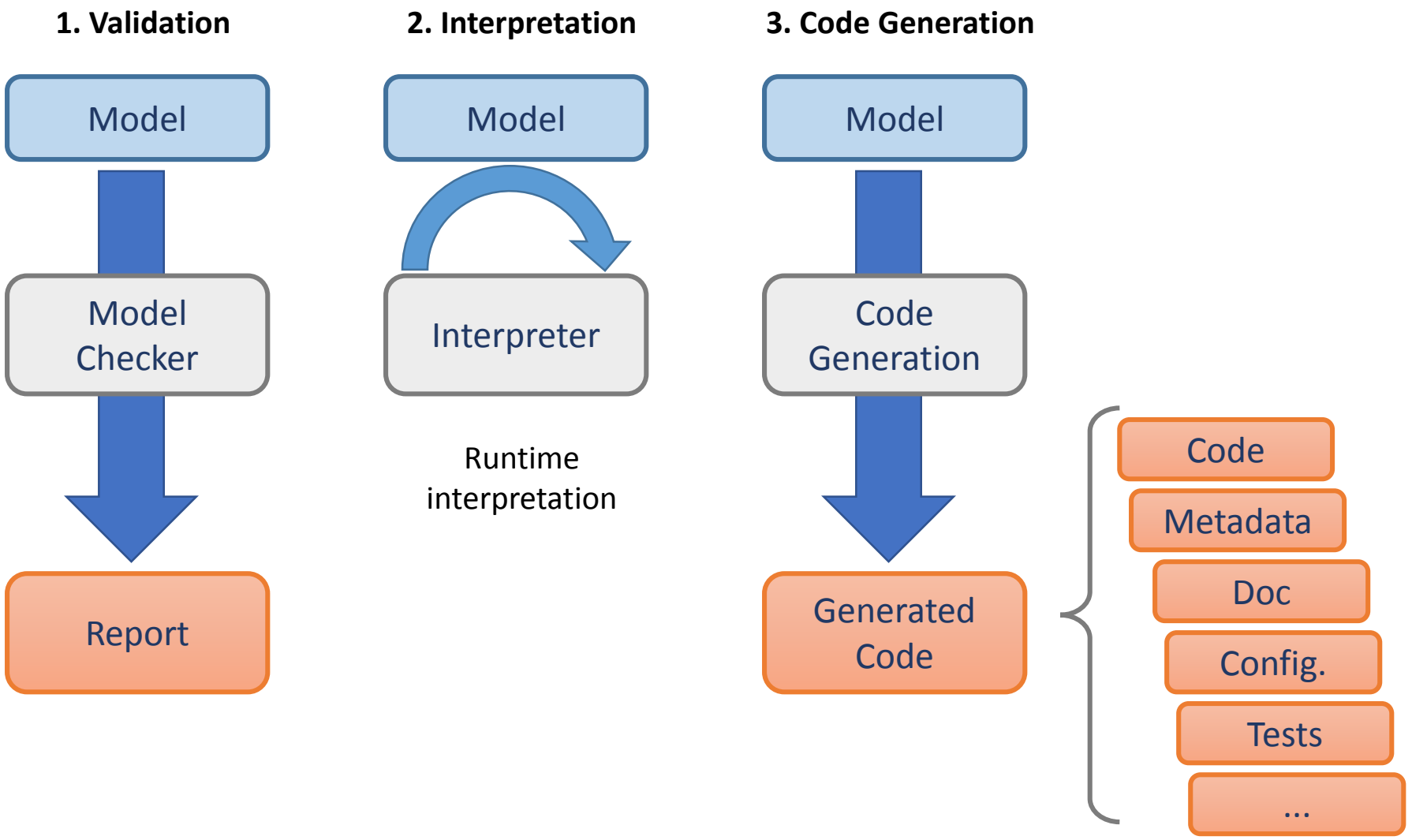
- Continuous Integration
- Agile development cycles
- Less cost

Defects Cost and Distribution

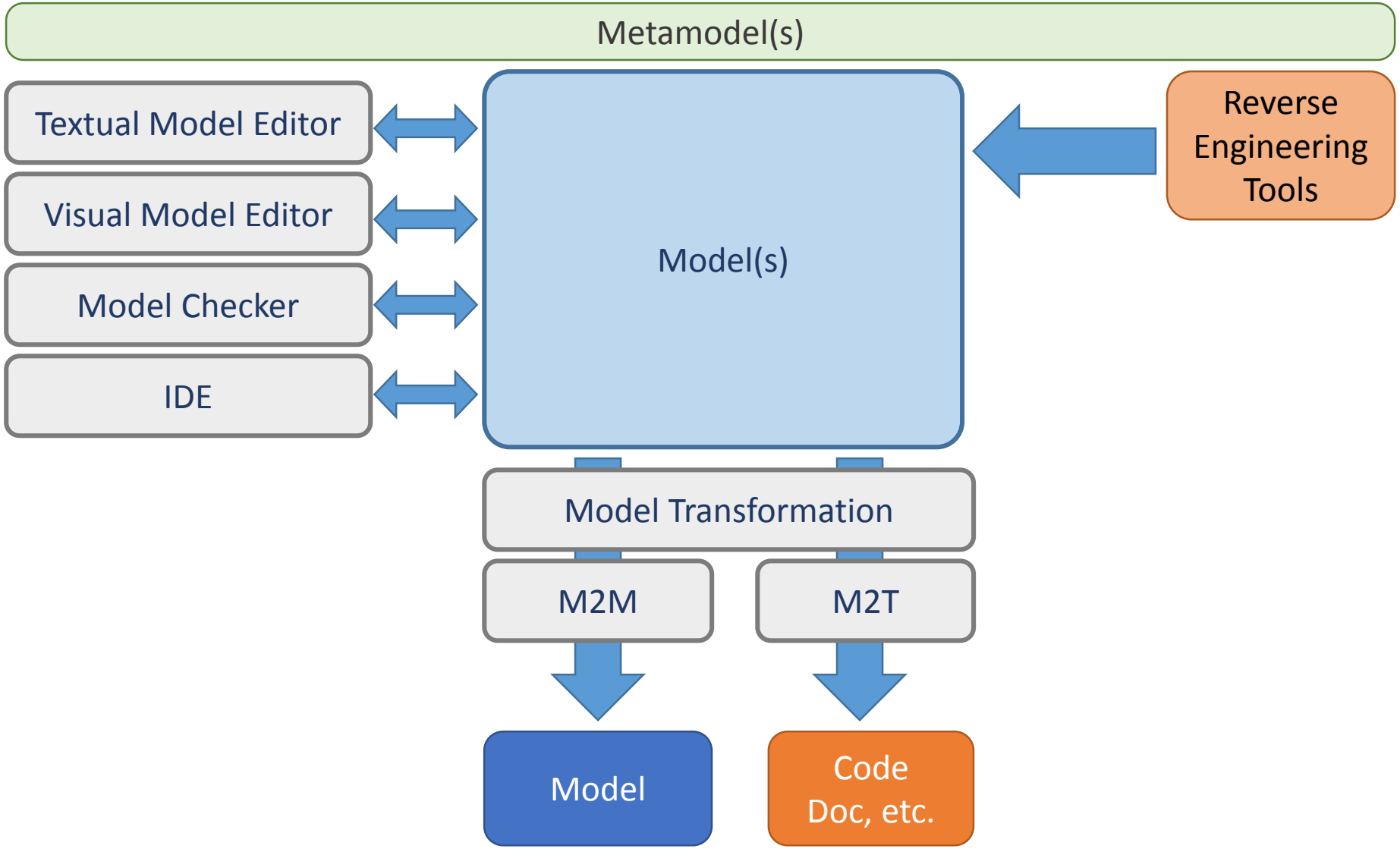


Snow ball effect

Models: some Cases of Use



Tools matters



The Tech Tale & UI Technology Trends

BTW: **That is why I love Technology Independence in MDD.**
Always ready to throw away my technology stack for a another one.

Generate everything?

Surely not!

There is **No Silver Bullet.**

Cost/Benefit approach following Novak's rule

- Generate the 80%
- Make the rest 20% easy to change and extend

All abstractions leaks

Yes but…

- Languages and compilers also do.
- And they are leaky indeed.

Therefore choosing the right level of abstraction is always a trade-off based on your requirements.

SoC is key to split concerns and hide complexity to the proper level.

Railroading: Good or not?

Railroading: Good or not?

Cons

- Lack of full design freedom
- Suboptimal solutions
- Constrained extensibility
- Architecture enforced

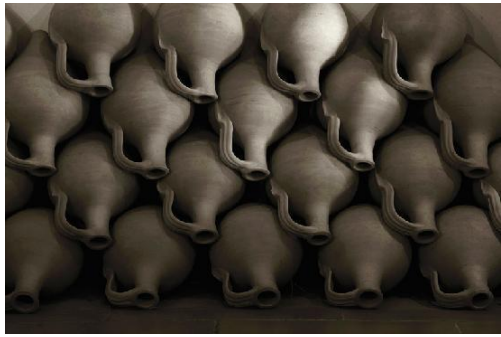
Pros

- Time to Market
- Proven Architectures
- Standardization
- Reliable/tested/compliant code
- Avoid tedious works
- Much cheaper

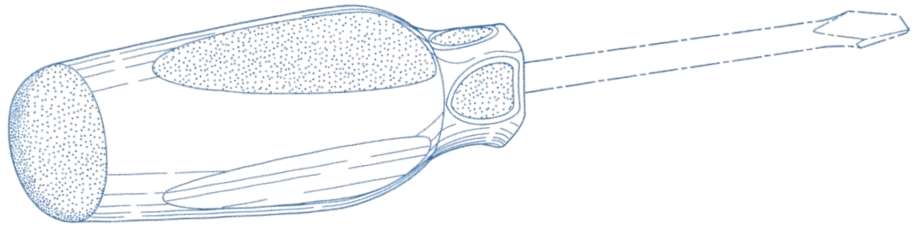
The role of MDE

- Developers: tendency to craftsmanship / artists / Not seen like an engineer…
- Citizen Developers: benefit directly from automation and complexity hiding

- We are not going to be enough
- MDE is a tool for **Lowering the Entry Barrier,** apply SoC & hide complexity
- Allowing non programmers to DIY to solve their day to day problems
- Mobility, Device Fragmentation, Cloud, Ubiquity, wearable IoT are here to stay



The world needs Apps…



Let's create better tools & build them!

Contact: @pmolinam