MDE 2.0 – Pragmatic formal model verification and other stories

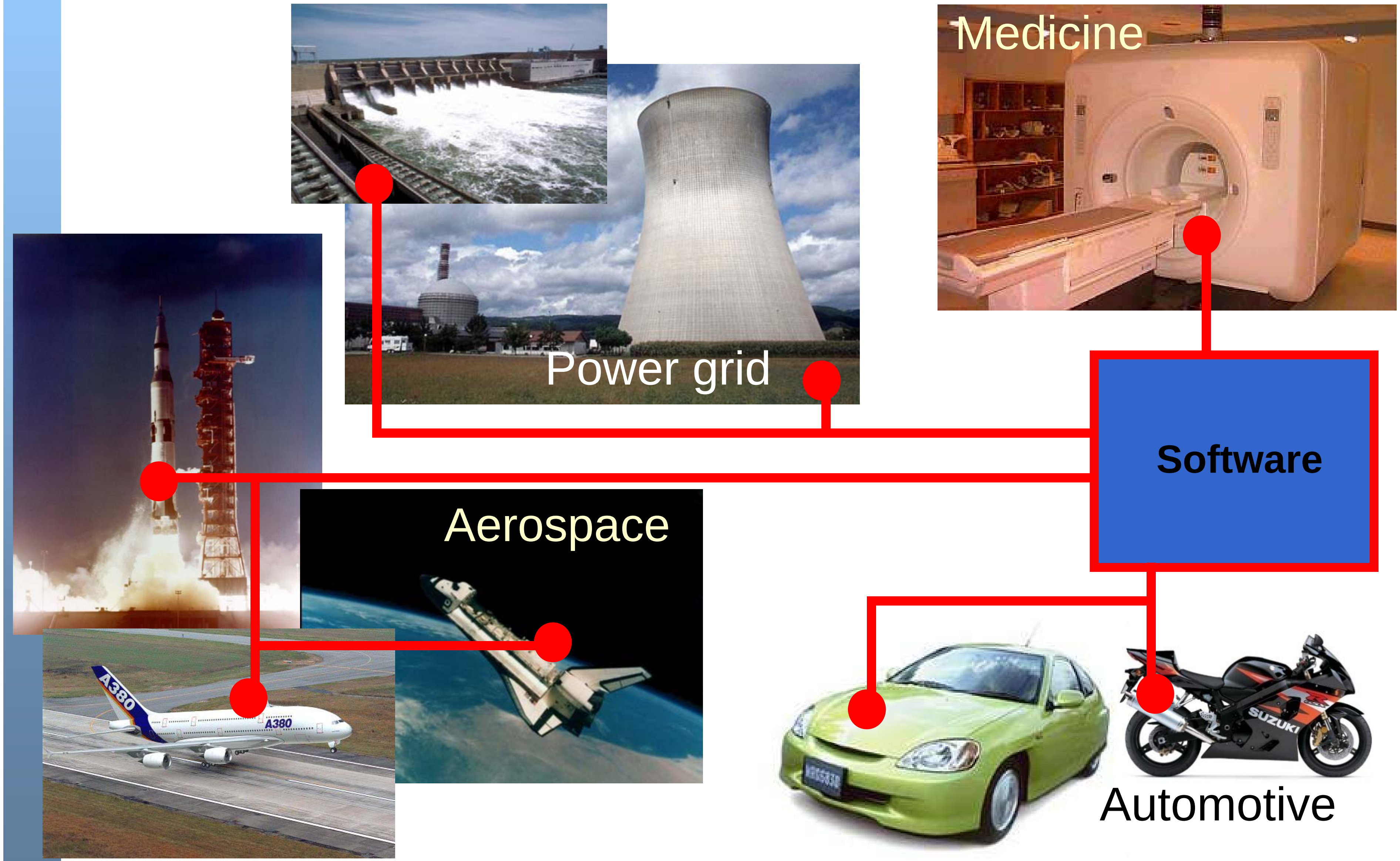
Jordi Cabot

HdR

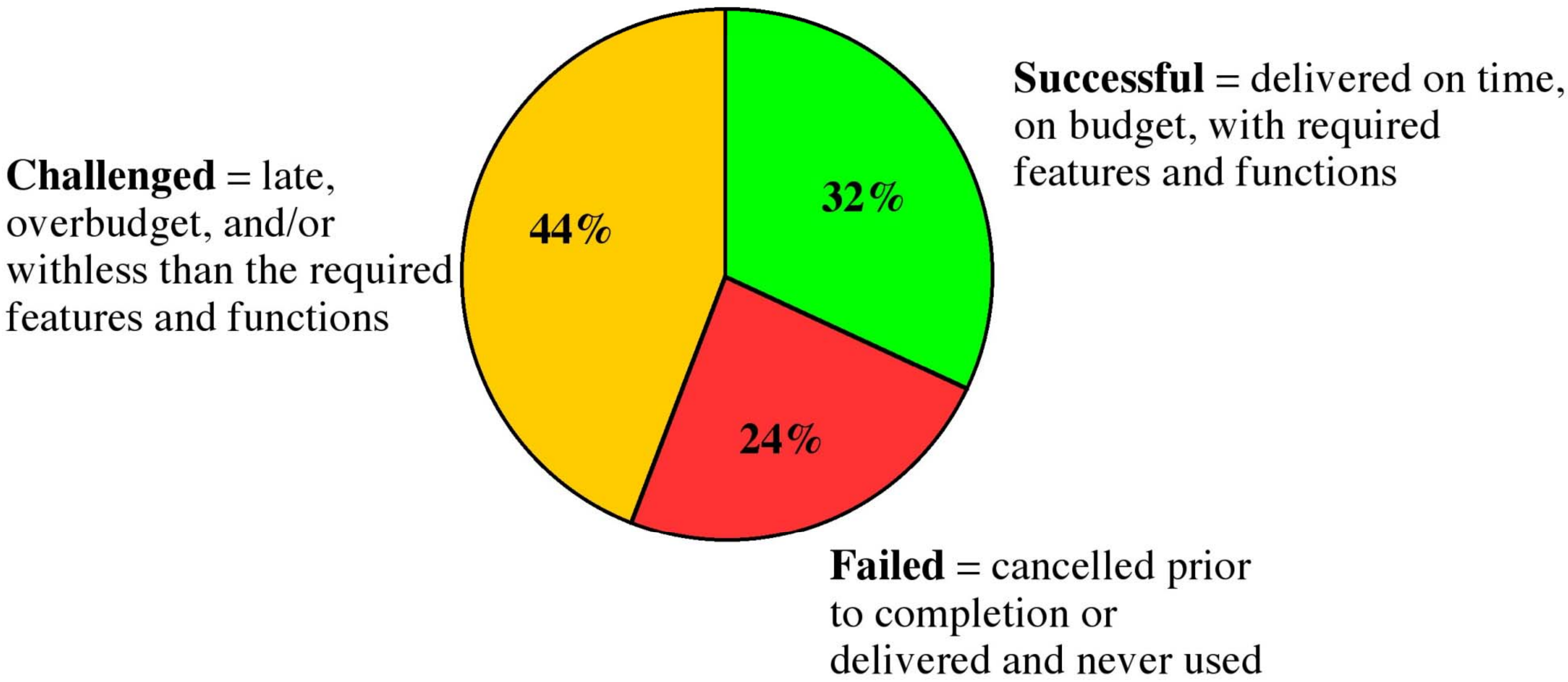
Index

- **Introducing MDE**
- Research in MDE
- A Research Agenda for MDE 2.0
  - Models & Quality
  - Legacy systems
  - Social aspects
  - Very Large models
- Dissemination and technology transfer
- Credits
- Conclusions

Software is everywhere

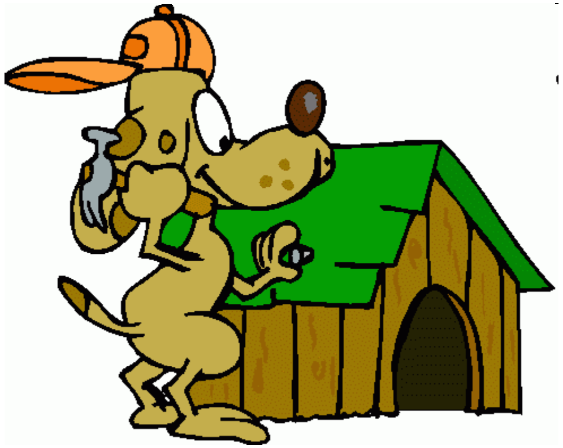


but software development is still a challenge

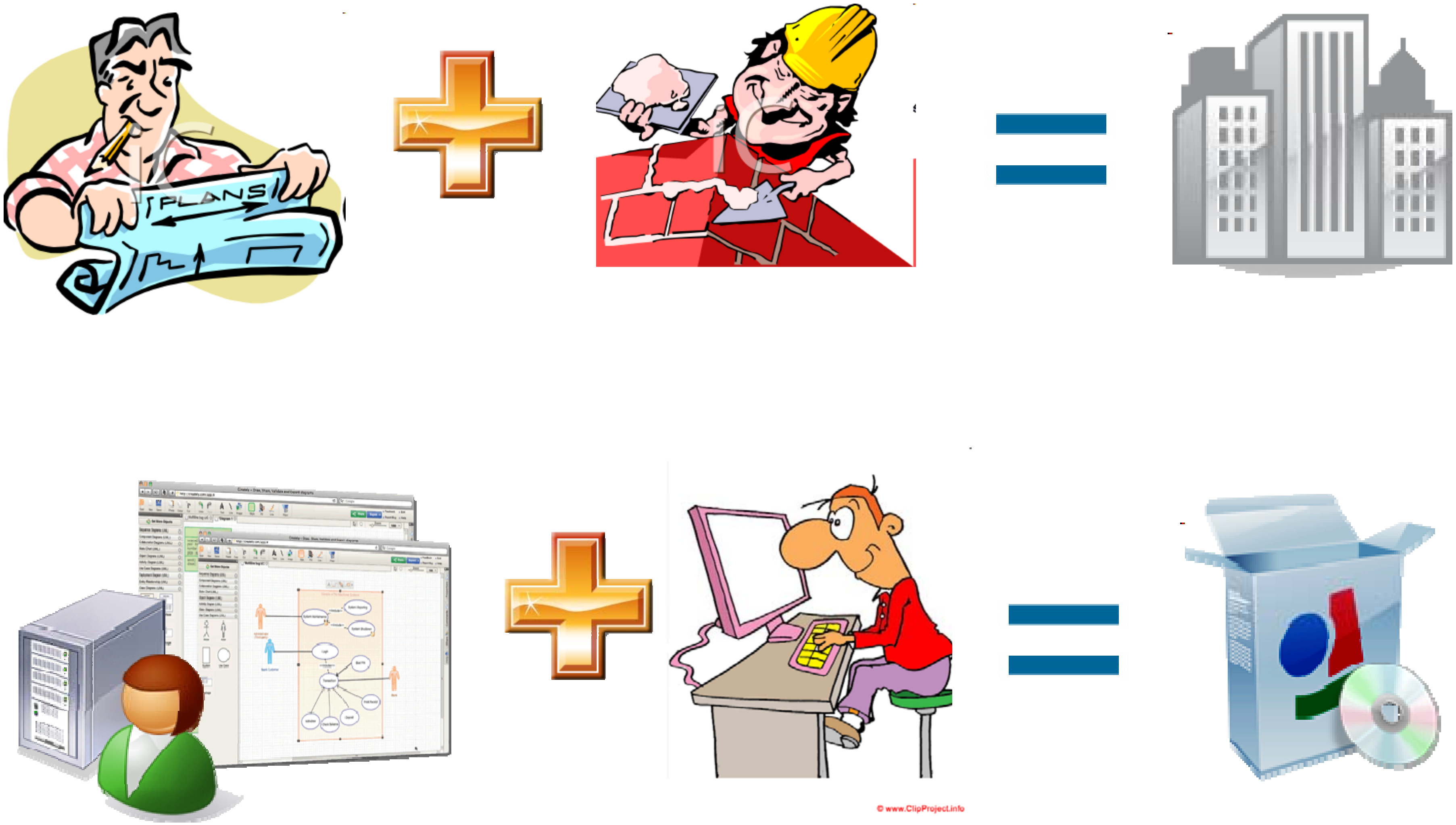


Model-driven Engineering

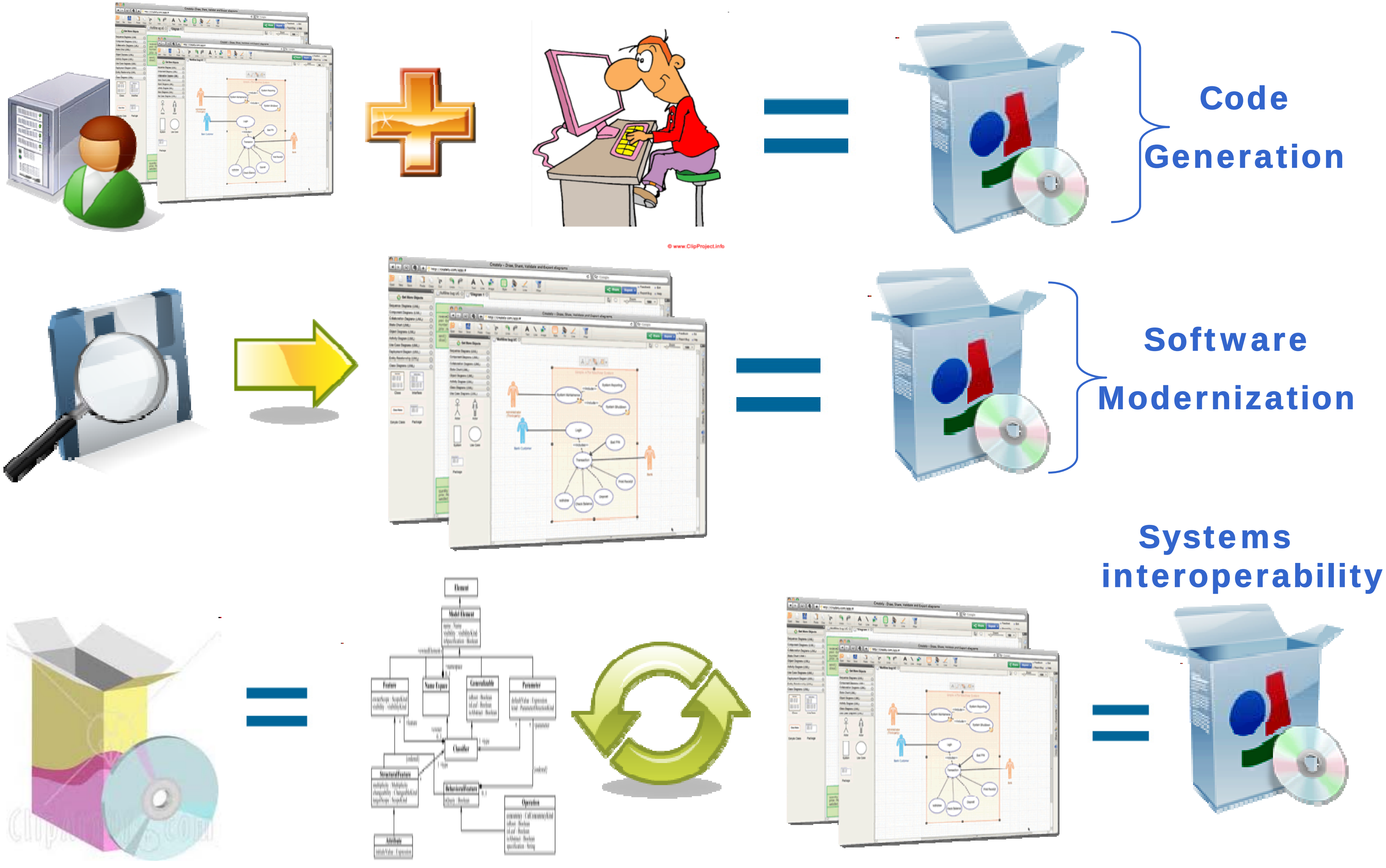
- MDE tries to improve this situation by promoting engineering principles in Software Engineering
- MDE advocates the rigorous use of software models as the main artifacts in all software engineering activities.
- This in fact is common practice in many other professions
- Does anybody imagine building a house without plans?



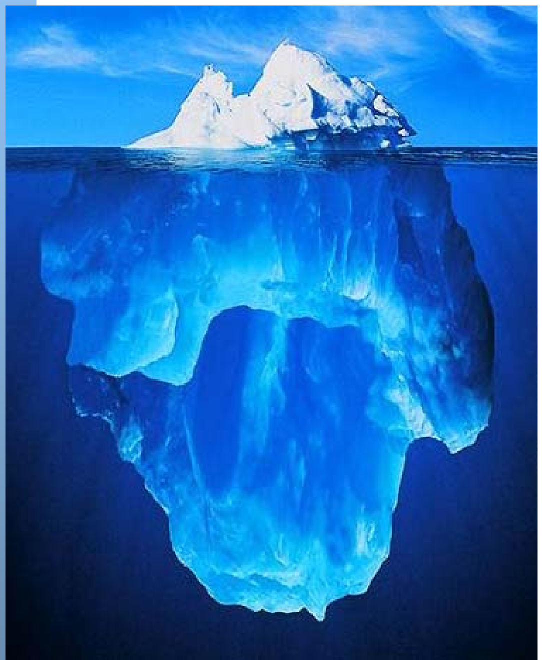
The MDEequation



Many MDE applications

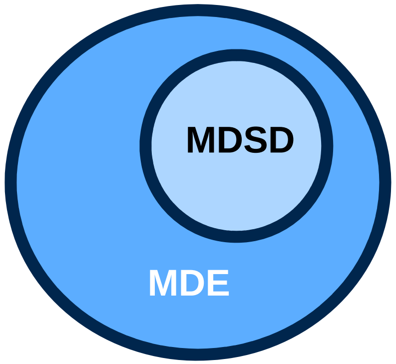


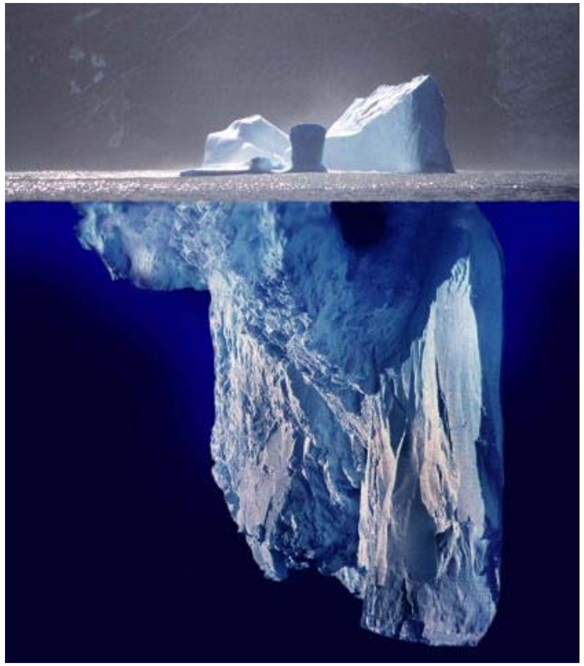
MDE = Model-Driven Everything



MDSD
(software
development)

MDE
(engineering)





What is MDE?

UML Profile OCL OMG

Ecore Code-Generation QVT

Marte TGG

Model Evolution Metamodel

MOF DSLs SBVR

EMF

Profiles Model-driven Rev. Eng

MDD

ATL

M2M Multi-modeling MDA

Graph

M2T

Model-Based Testing BPMN

T2M Model Quality

There's hope - Order in the Chaos

- Basic principle: **Everything is a model (J. Bézivin)**
- Models are designed/built/generated to be
  - –Observed
  - –Transformed
- The **MDE Equation**:

  Models + transformations = Software

where of course transformations can also be regarded as models

Models + transformation models = Software

Models + Models = Software

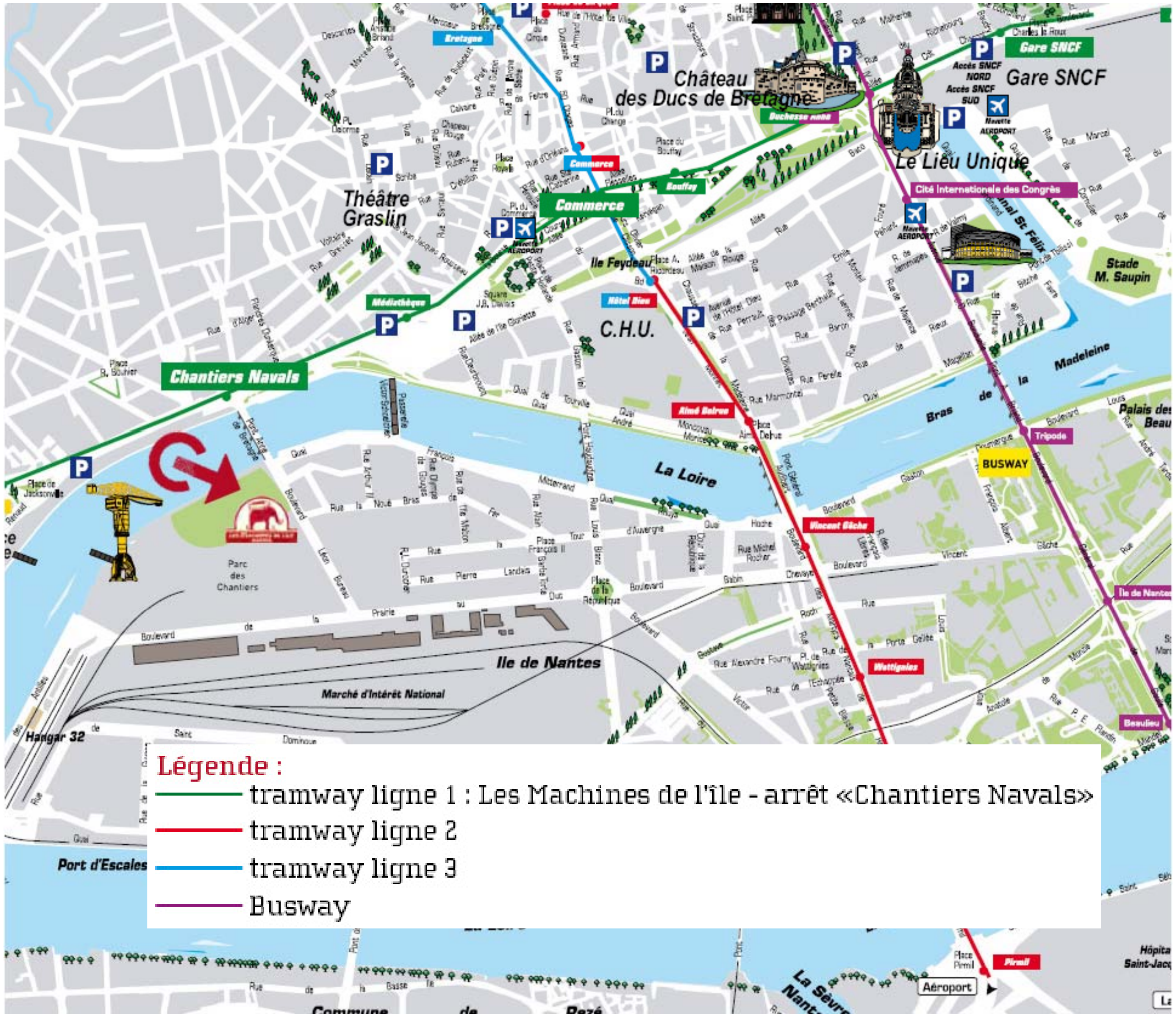
2 x Models = Software

Models = 1 / 2 Software ???????

What is a model

City of Nantes = "system" to be modeled

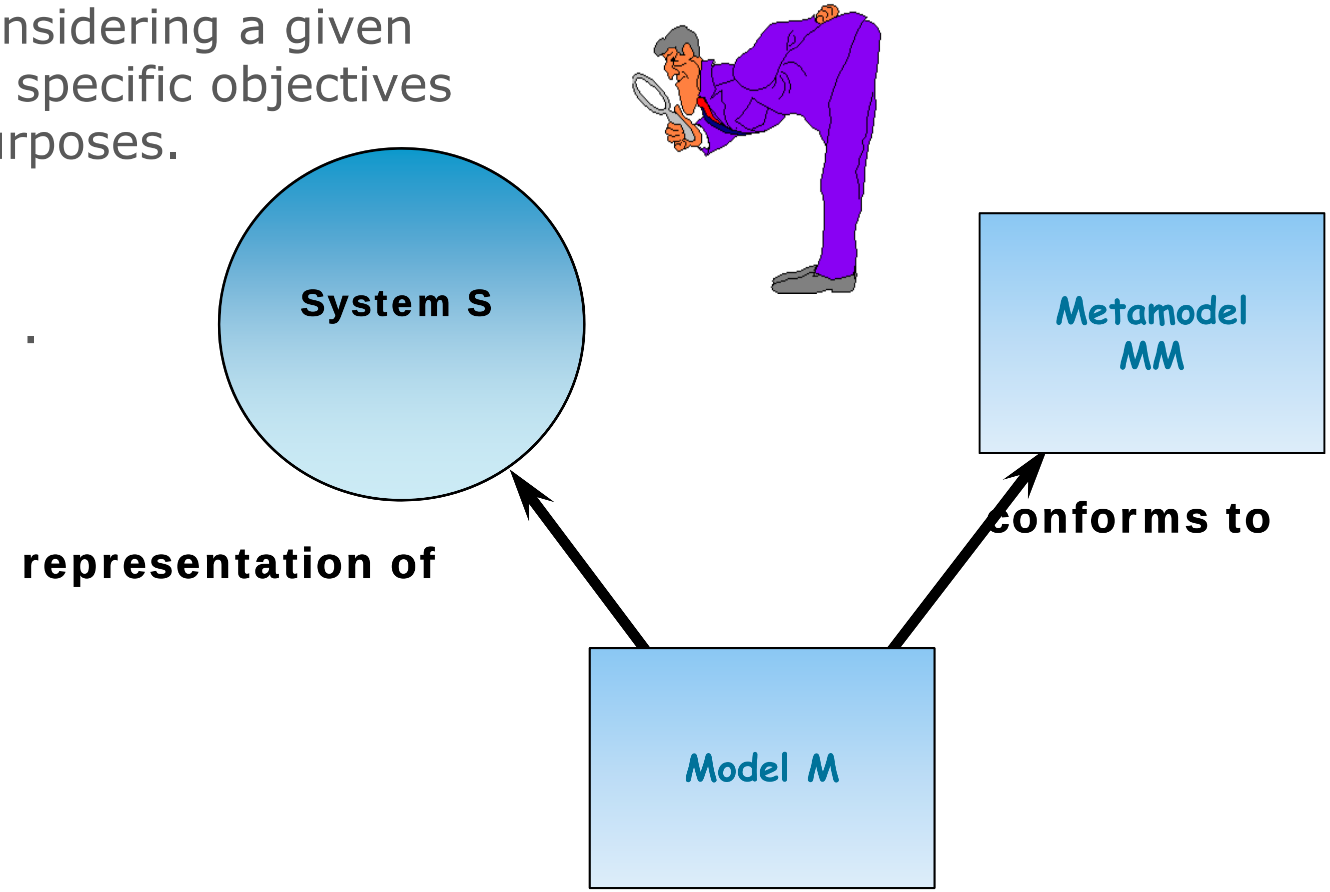
A map is a model o this system



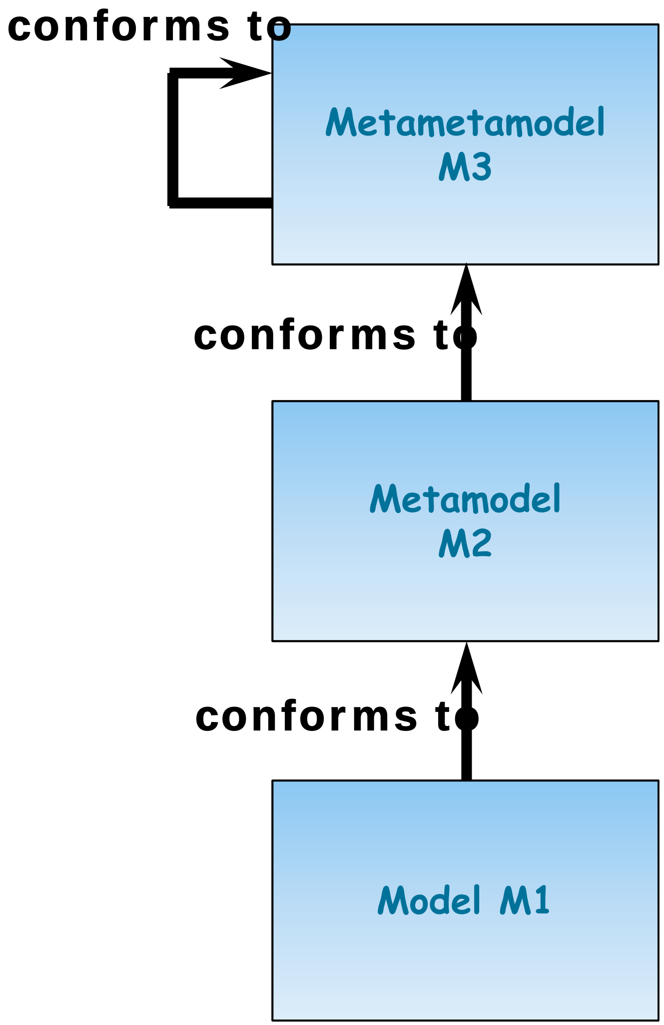
Its legend (what elements can appear, how they can be combined,…) is the grammar/ metamodel

Models & Metamodels

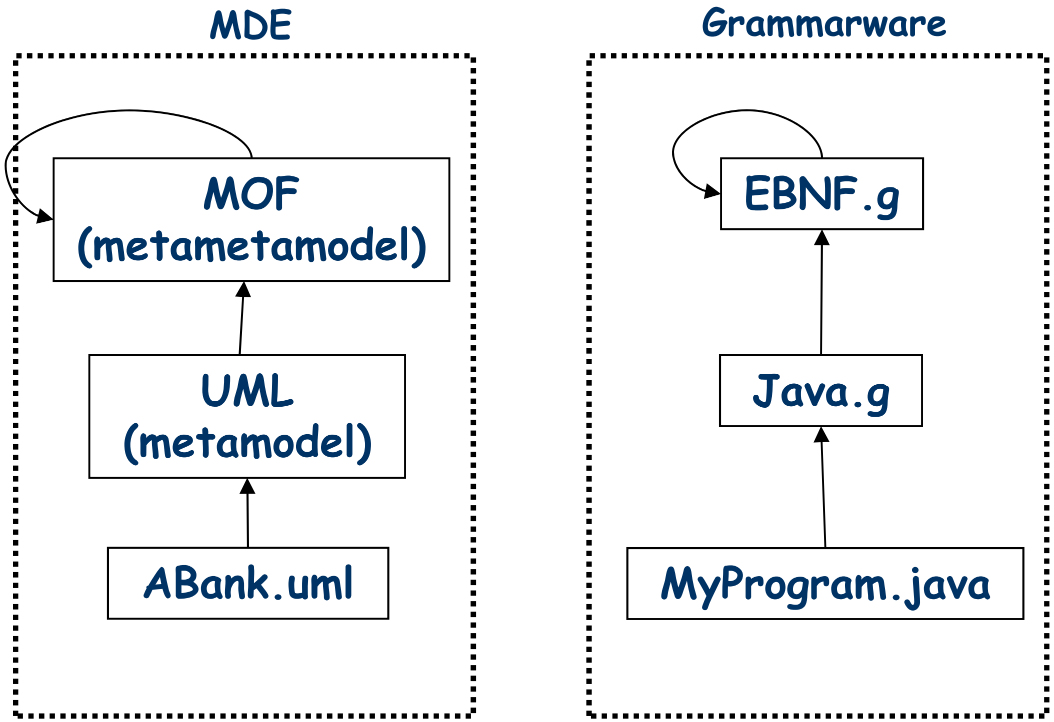
A model is the result, of the observation of a system considering a given metamodel with specific objectives or purposes.



The 3-level Modeling Stack

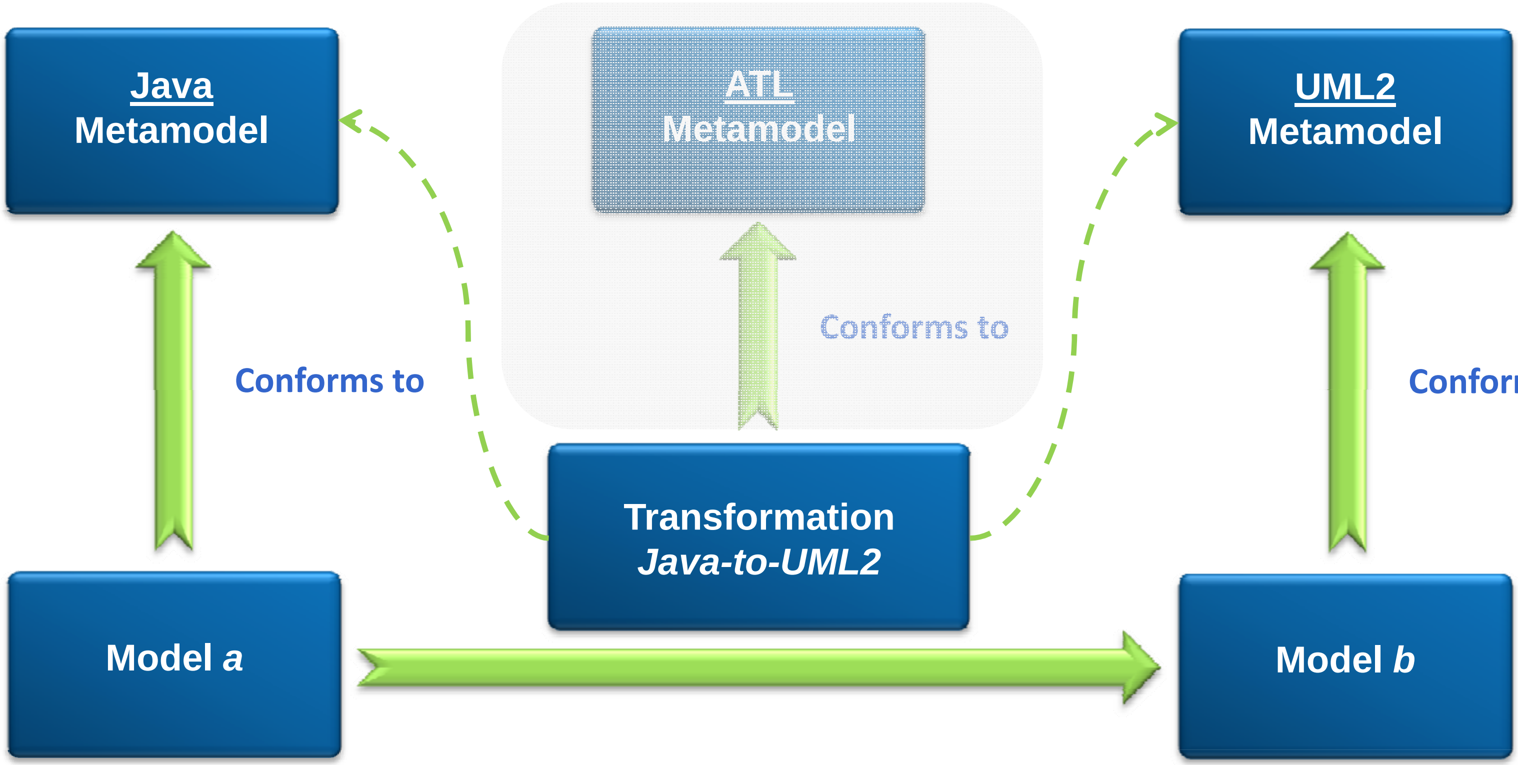


Btw, same approach as other Technical Spaces



MDE Core Technique/Operation: Model Transformation

- Model-to-Model Transformation (M2M)



Index

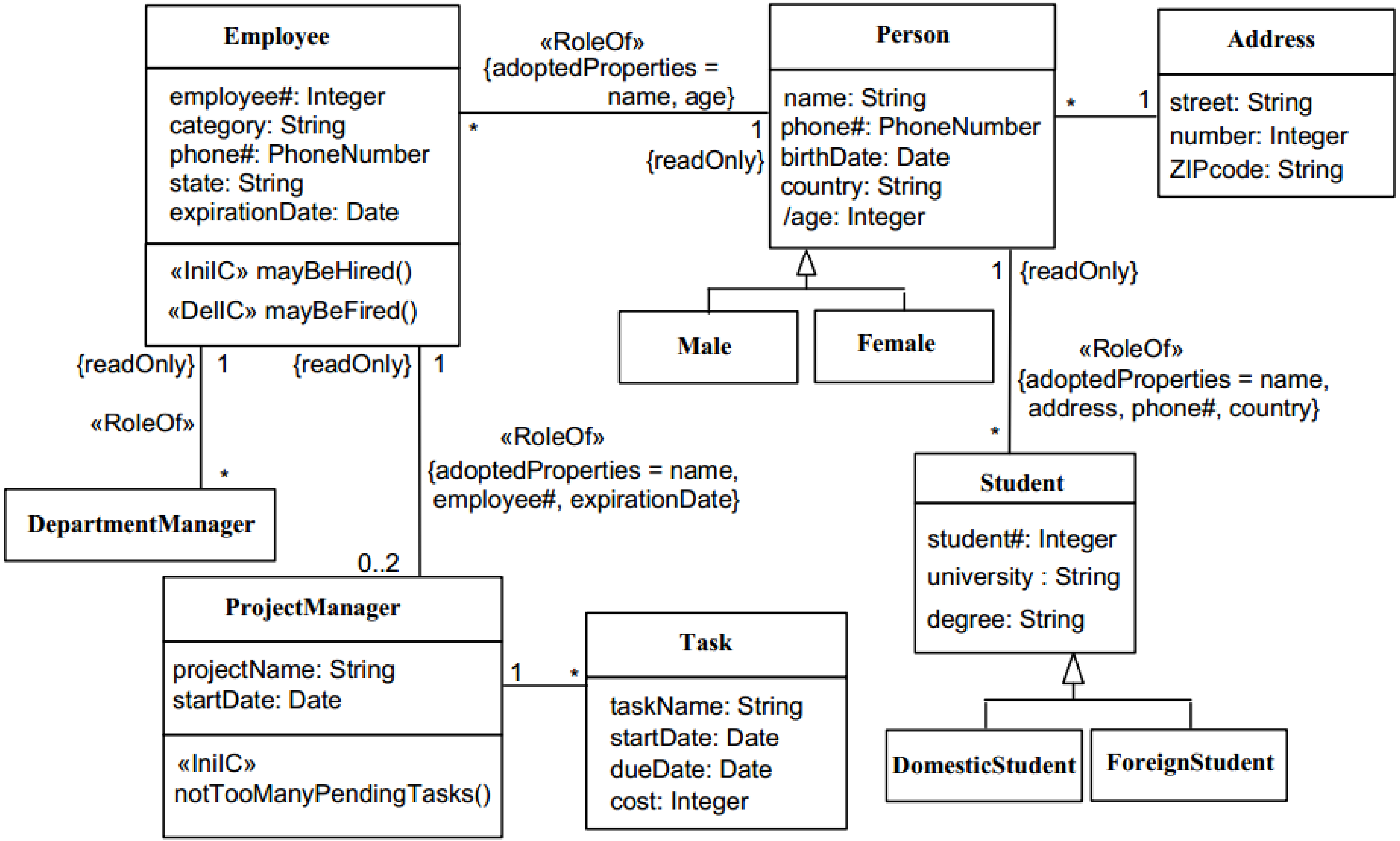
- Introducing MDE
- **Research in MDE**
- A Research Agenda for MDE 2.0
  - Models & Quality
  - Legacy systems
  - Social aspects
  - Very Large models
- Dissemination and technology transfer
- Credits
- Conclusions

Research in MDE

We have advanced a lot on the core techniques

- UML and profiles
- DSLs & Language workbenches
- Model-to-model and model-to-text transformations
- Model management and evolution
- …

with some contributions: Conceptual Modeling



J Cabot, R Raventós: Conceptual Modelling Patterns for Roles. Journal on Data Semantics

with some contributions: Rule modeling

Table 2
Equivalences for collection operators

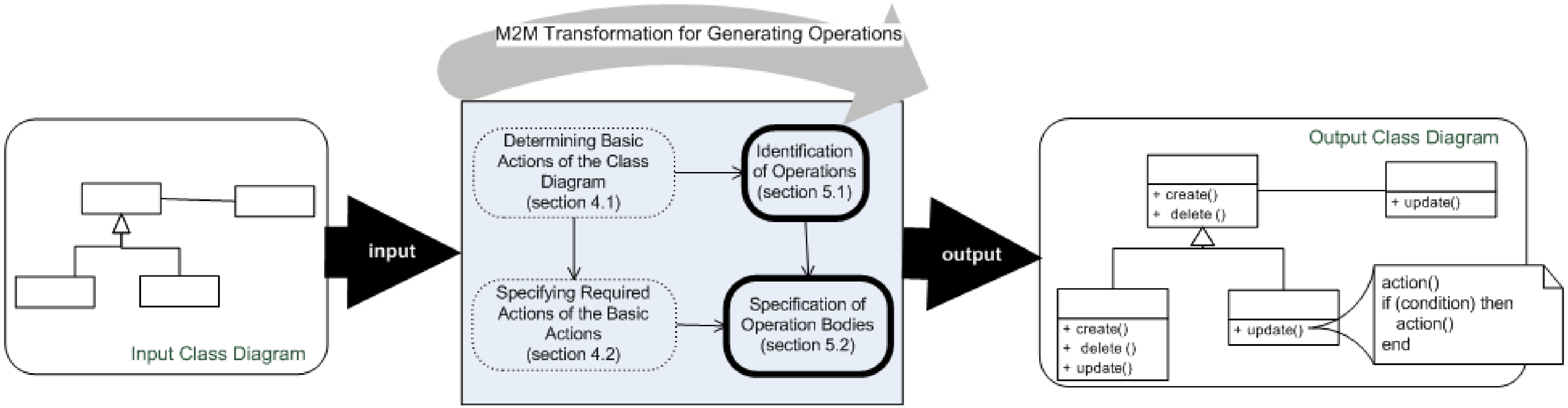
X->includes(o) ↔ X->count(o)>0	X->excludes(o) ↔ X->count(o)<1
X->includesAll(Y) ↔ Y->forAll($y_1 \mid$X->includes(y_1))	X->excludesAll(Y) ↔ Y->forAll($y_1 \mid$X->excludes(y_1))
X->isEmpty() ↔ X->size()=0	X->notEmpty() ↔ X->size()>0
not X->isEmpty() ↔ X->notEmpty()	not X->notEmpty() ↔ X->isEmpty()
X->excluding(o) ↔ X-> −(Set{o})	X->including(o) ↔ X->union(Set{o})
X->size()<=0 ↔ X->size()=0	not X->size()=0 ↔ X->size()>0
X->last() ↔ X->at(X->size())	X->first() ↔ X->at(1)
X->union(Y).$r_1 \ldots r_n$->forAll($z_1 \mid$Z) ↔ X.$r_1 \ldots r_n$->forAll($z_1 \mid$Z) and Y. $r_1 \ldots r_n$-> forAll($z_1 \mid$Z)	

Table 3
Equivalences for iterator expressions

X->exists(Y) ↔ not X->forAll(not Y)	not X->exists(Y) ↔ X->forAll(not Y)
X->select(Y)->size()>0 ↔ not X->forAll(not Y)	X->select(Y)->size()=0 ↔ X->forAll(not Y)
X->select(Y)->forAll(Z) ↔ X->forAll(Y implies Z)	X->select(Y)->exists(Z) ↔ X->exists(Y and Z)
X->reject(Y) ↔ X->select(not Y)	X->any(Y) ↔ X->select(Y)->asSequence()->first()
X->isUnique(Y) ↔ X->forAll(x_1,$x_2 \mid$ x_1<>x_2 implies x_1.Y<>x_2.Y)	X->one(Y) ↔ X->select(Y)->size()=1
X->select(Y)->size()=X->size() ↔ X->forAll(Y)	

J Cabot, E Teniente: Transformation Techniques for OCL Constraints. Science of Computer Programming Journal

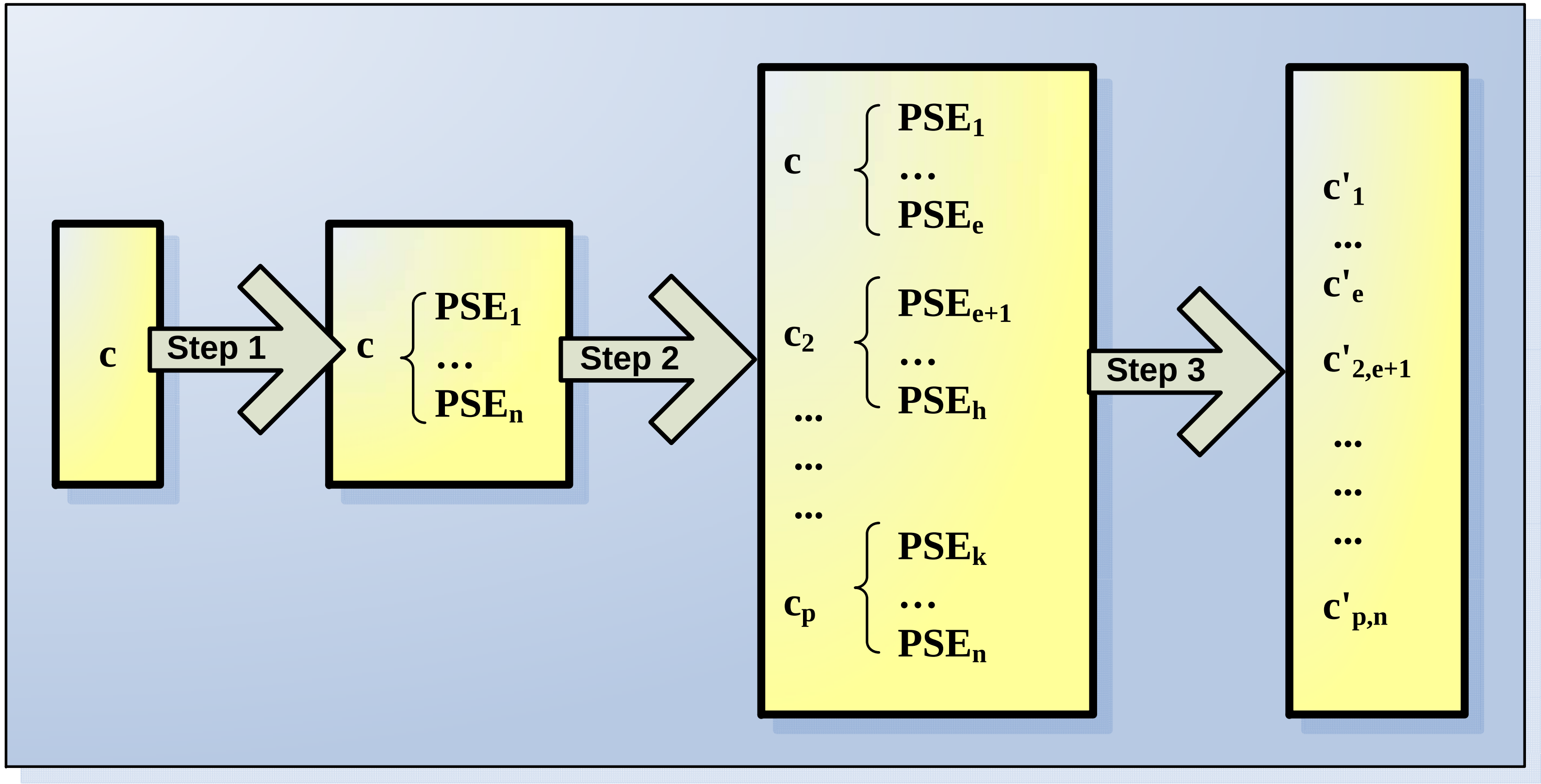
with some contributions: Code Generation



Albert, Cabot, Gómez, Pelechano: Automatic Generation of Basic Behavior Schemas from UML Class Diagrams. Software and Systems Modeling

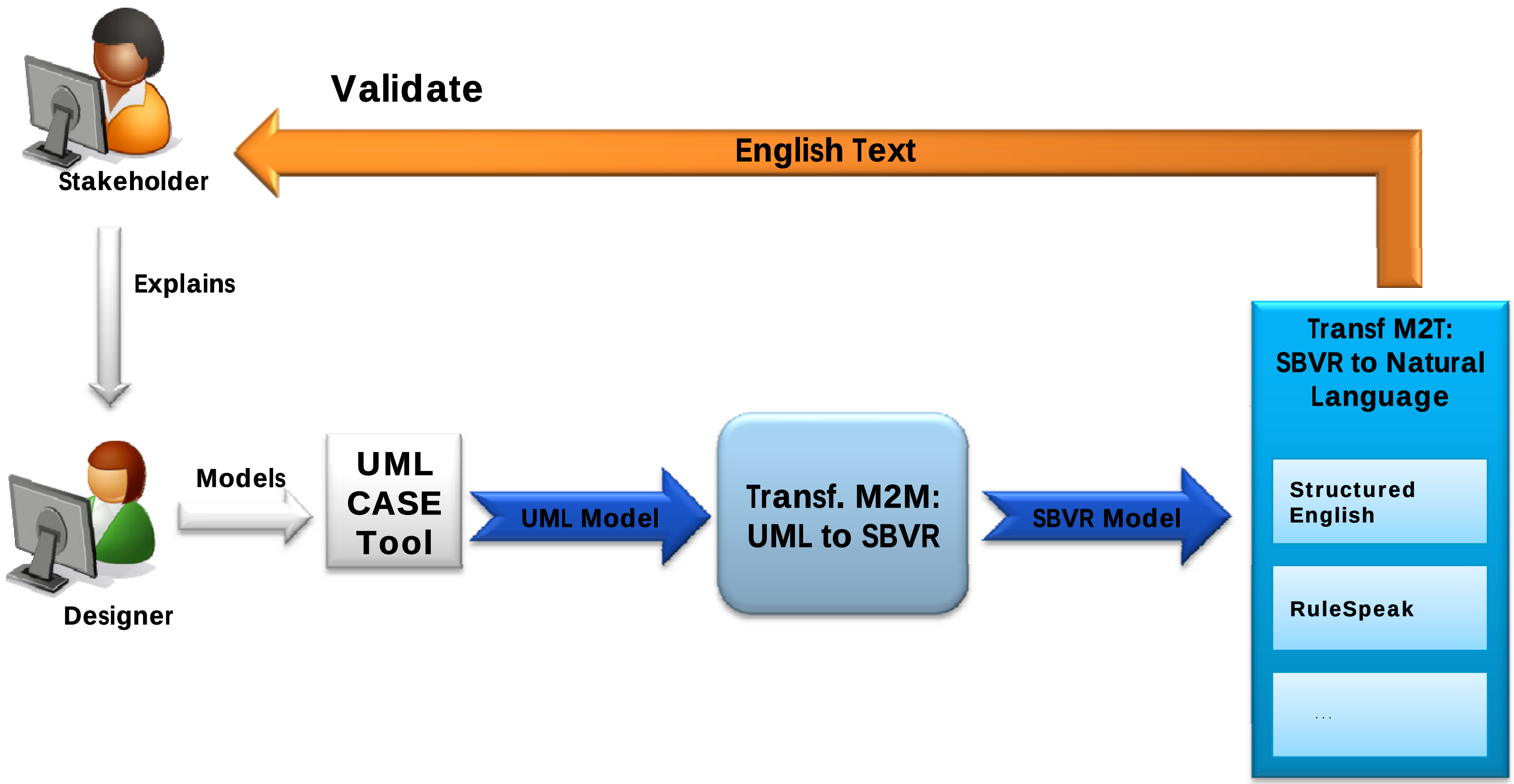
Albert,Cabot, Gómez,Pelechano. Generating operation specifications from UML class diagrams: A model transformation approach. Data & Knowledge Engineering

with some personal contributions: Code Generation



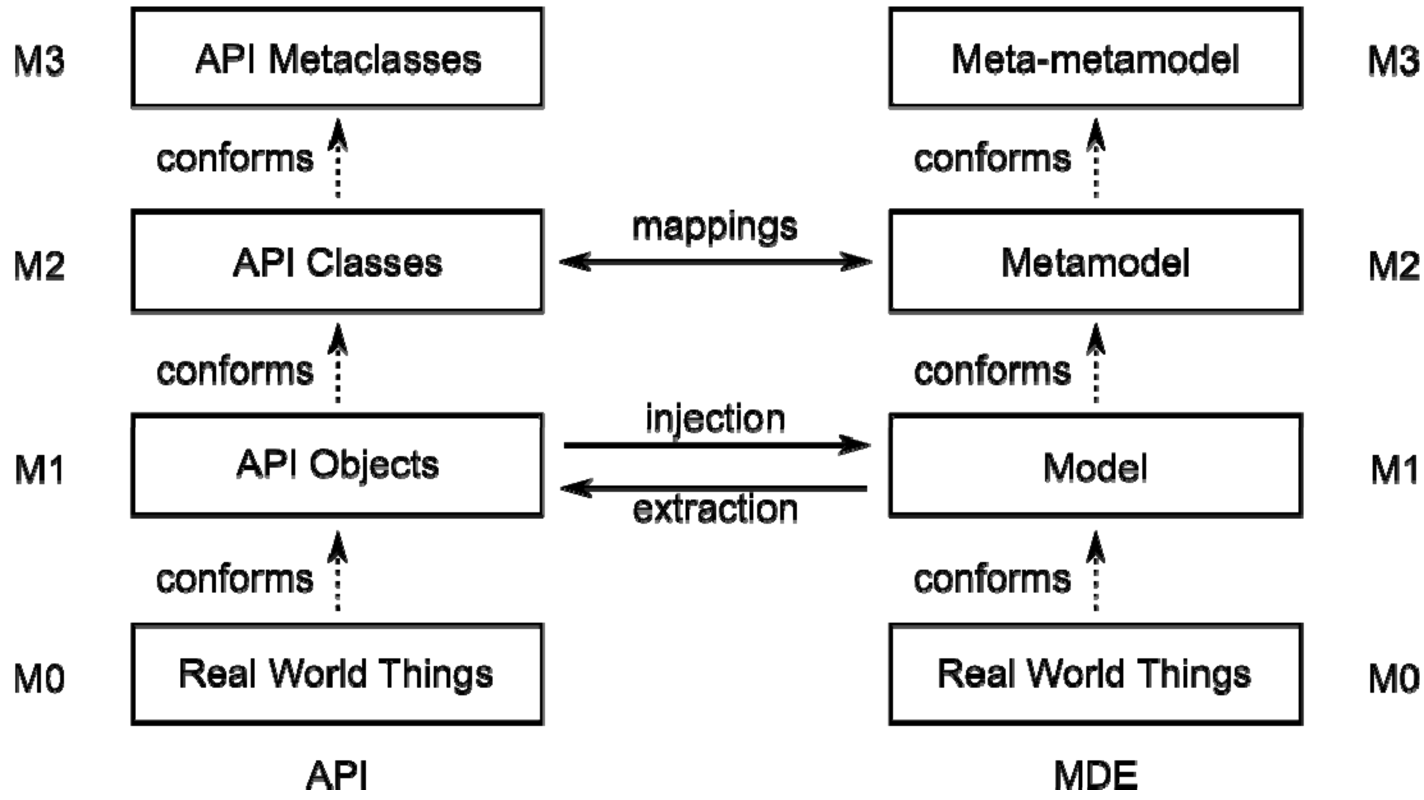
J Cabot, E Teniente: Incremental Integrity Checking of UML/OCL Conceptual Schemas. Journal of Systems and Software

with some contributions: UML Validation



J Cabot, R Pau, R Raventós: From UML/OCL to SBVR Specifications: a Challenging Transformation. Information Systems

with some contributions: API integration

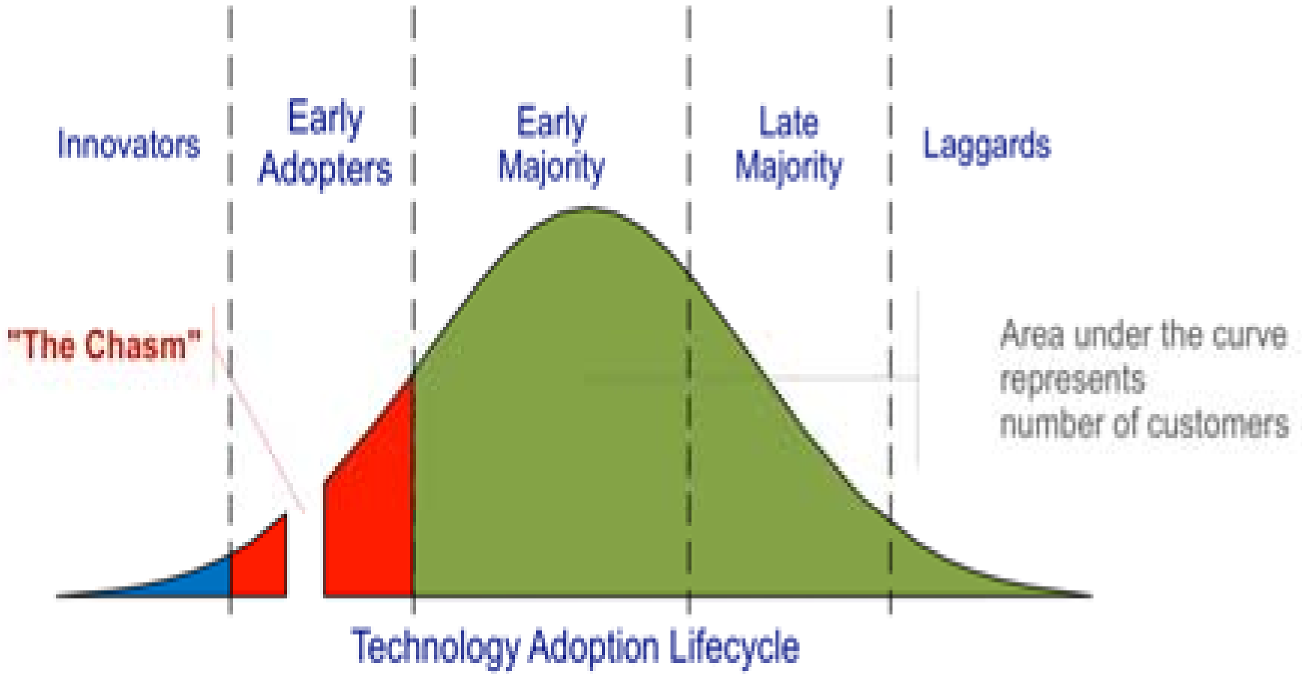


J Cánovas, F Jouault, J Cabot, J García Molina. API2MoL: Automating the building of bridges between APIs and Model-Driven Engineering. Information and Software Technology.

But it's clearly not enough

- Modeling will be commonplace in 3 years time – S. Mellor

  Though he is giving the same answer for the last 20 years



Index

- Introducing MDE
- Research in MDE
- **A Research Agenda for MDE 2.0**
  - Models & Quality
  - Legacy systems
  - Social aspects
  - Very Large models
- Dissemination and technology transfer
- Credits
- Conclusions

What else do we need?
MDE 2.0

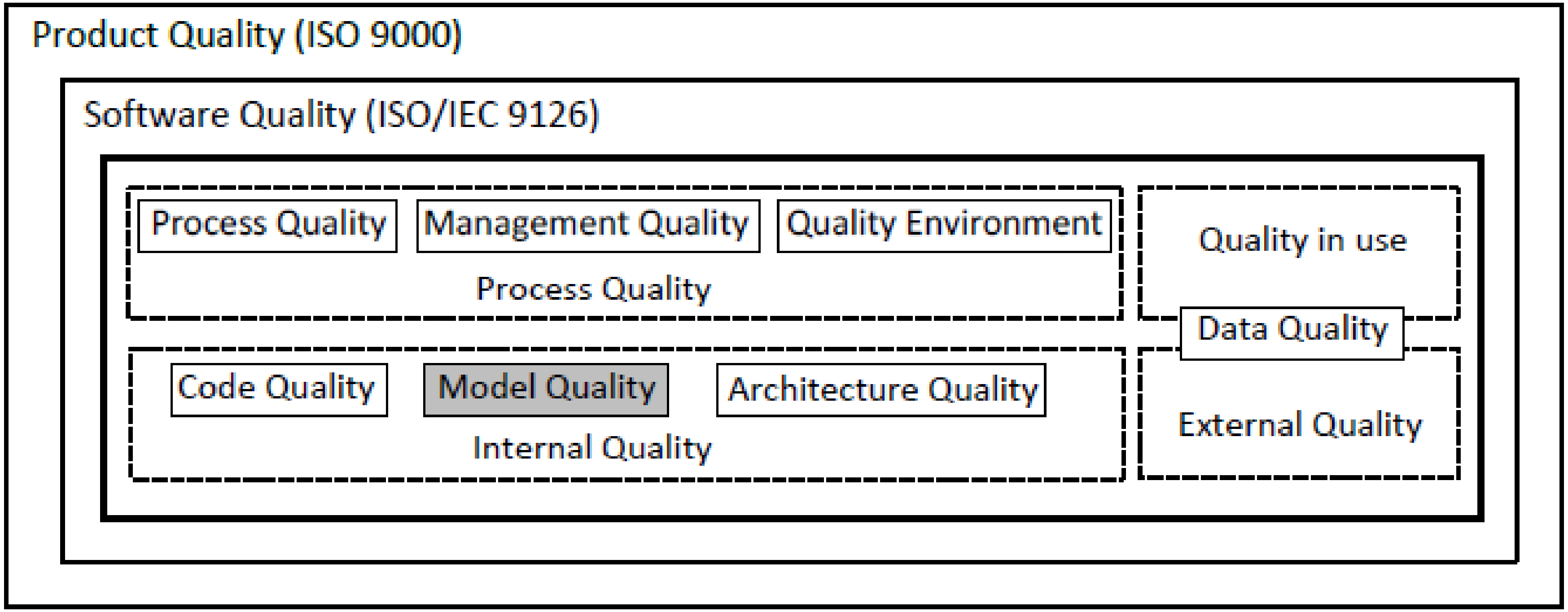
Four main challenges

1. Quality of models 
2. Support for legacy systems 
3. Social aspects of MDE
4. Very large models / Scalability

Quality

Quality

- Quality is a very broad concept

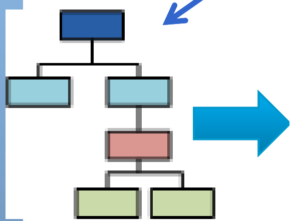


- We focus on the verification of models (are we building the models right?) and, partially, on their validation (are we building the right models?)

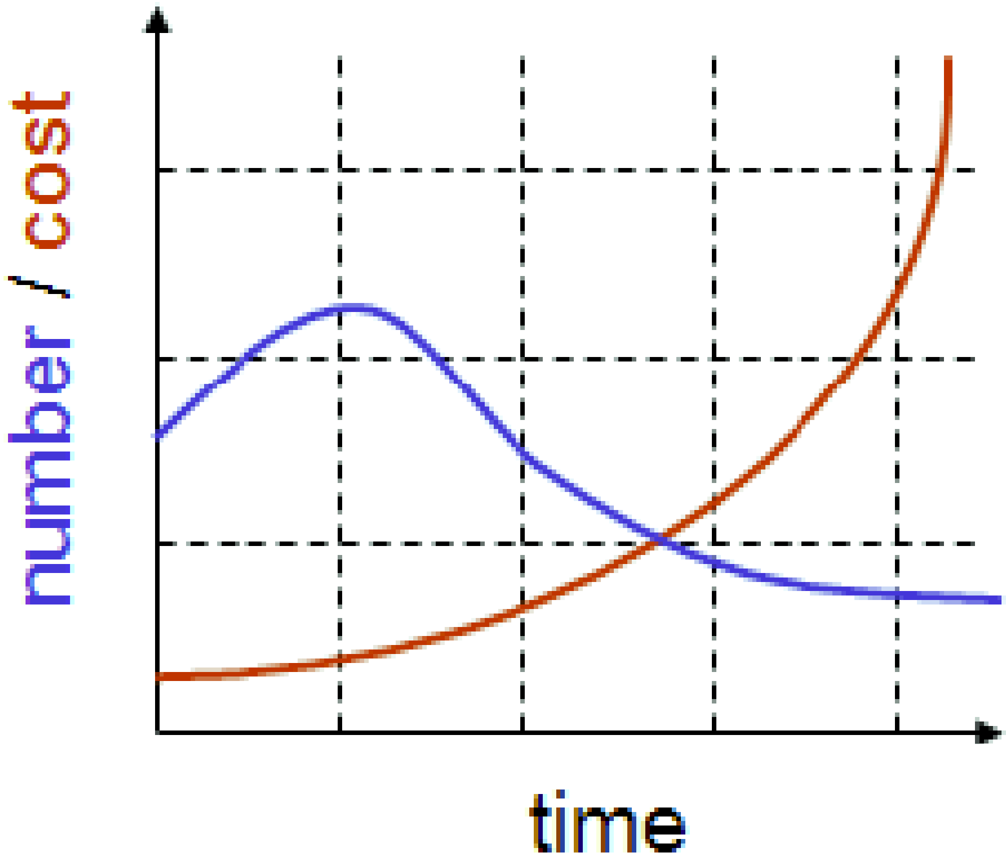
Importance of Quality in models

MDE- … process

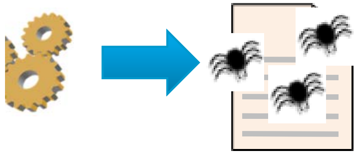
Original model



Tran



Source Code



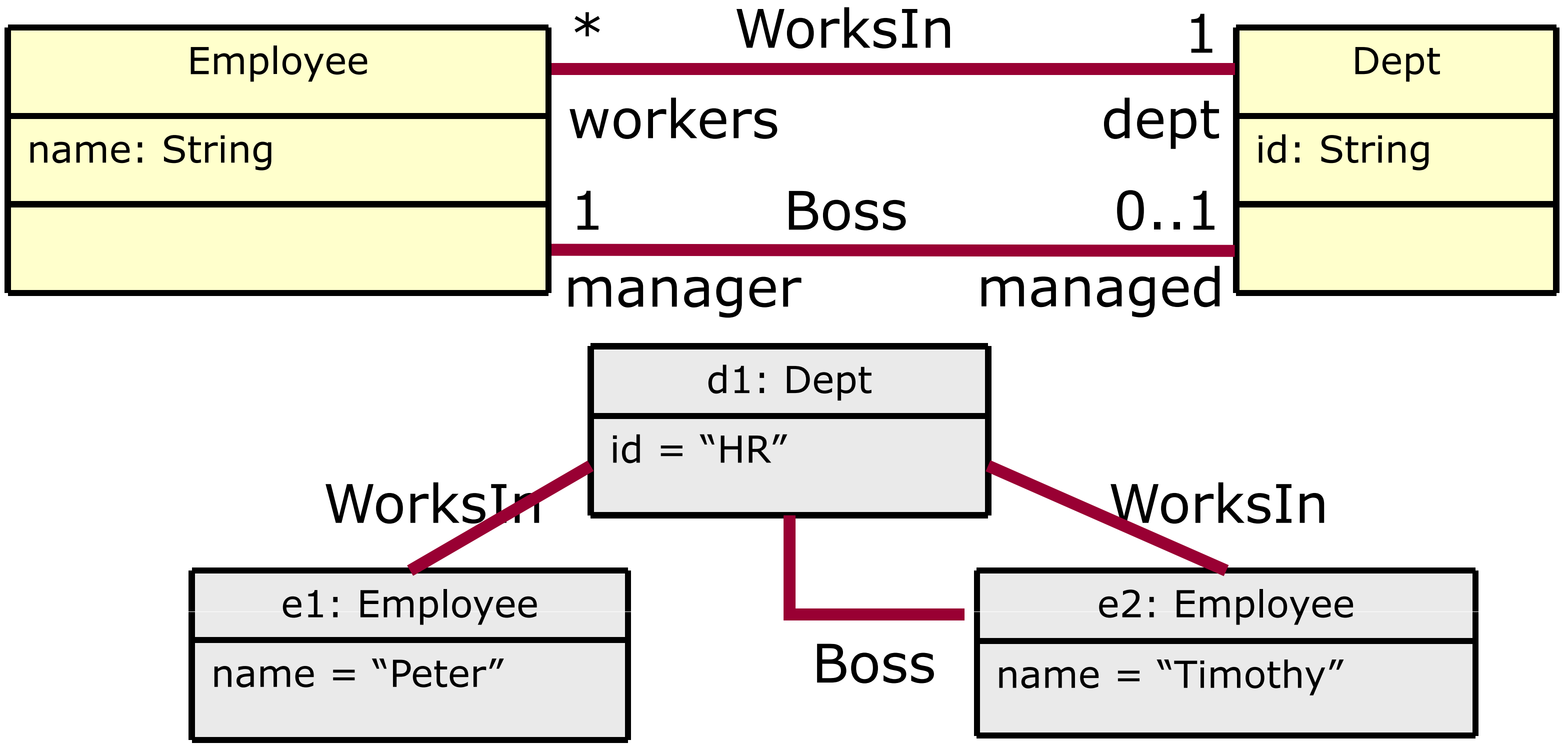
lodel
formation

Err … rs in

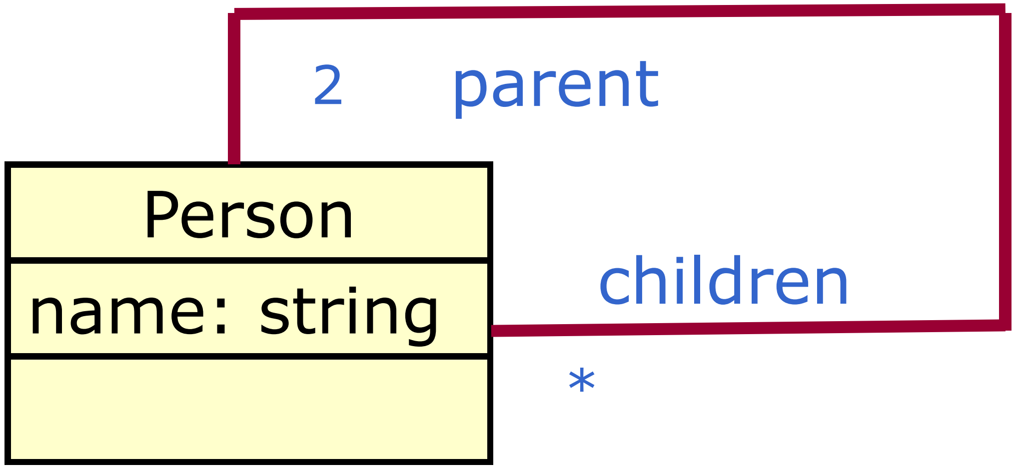
the resulting software

A Basic quality property: Satisfiability

- Satisfiability is the most basic correctness property for static models. Liveliness, redundancy,… can be expressed in terms of this one
- A model is satisfiable if it is possible to create a valid instantiation of that model. Otherwise it is useless, users won't be able to work with the model
- A instantiation is valid if it satisfies all model constraints

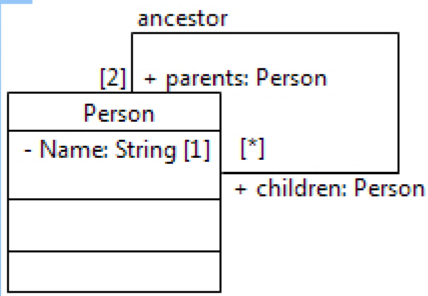


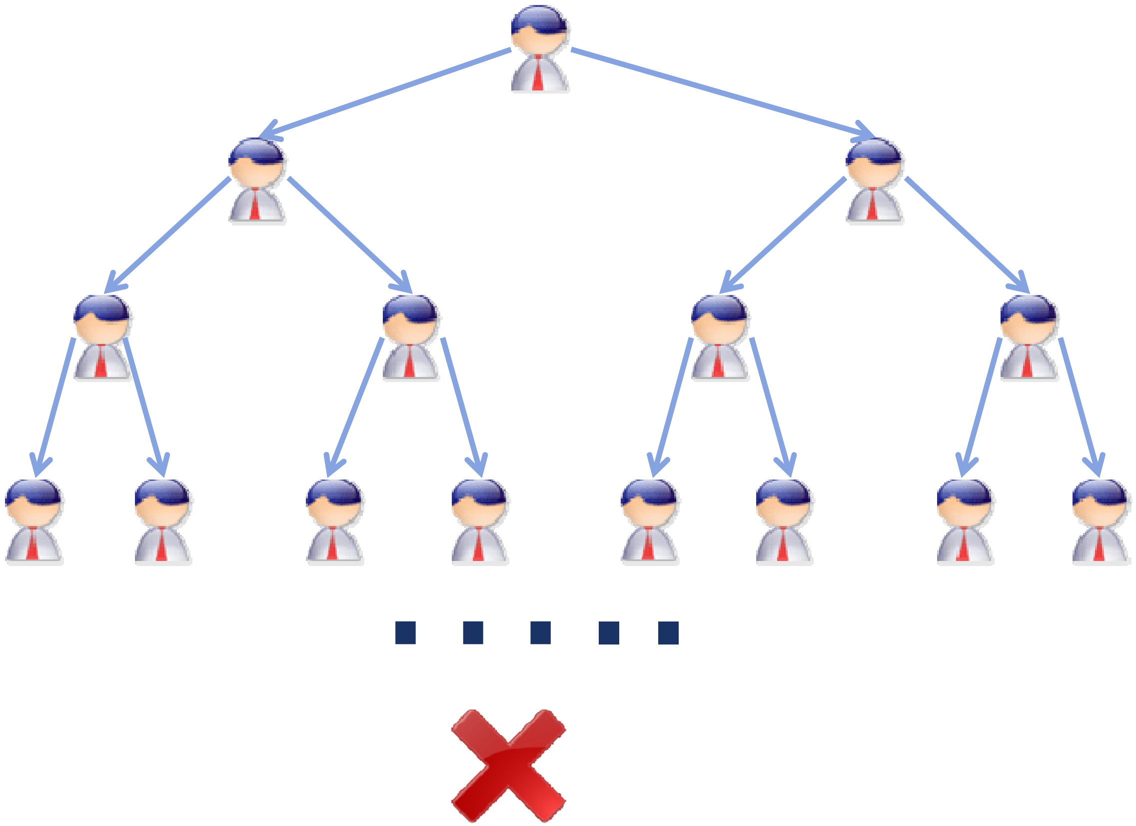
Example: Is it satisfiable?



+ constraint : Nobody can be his own ancestor

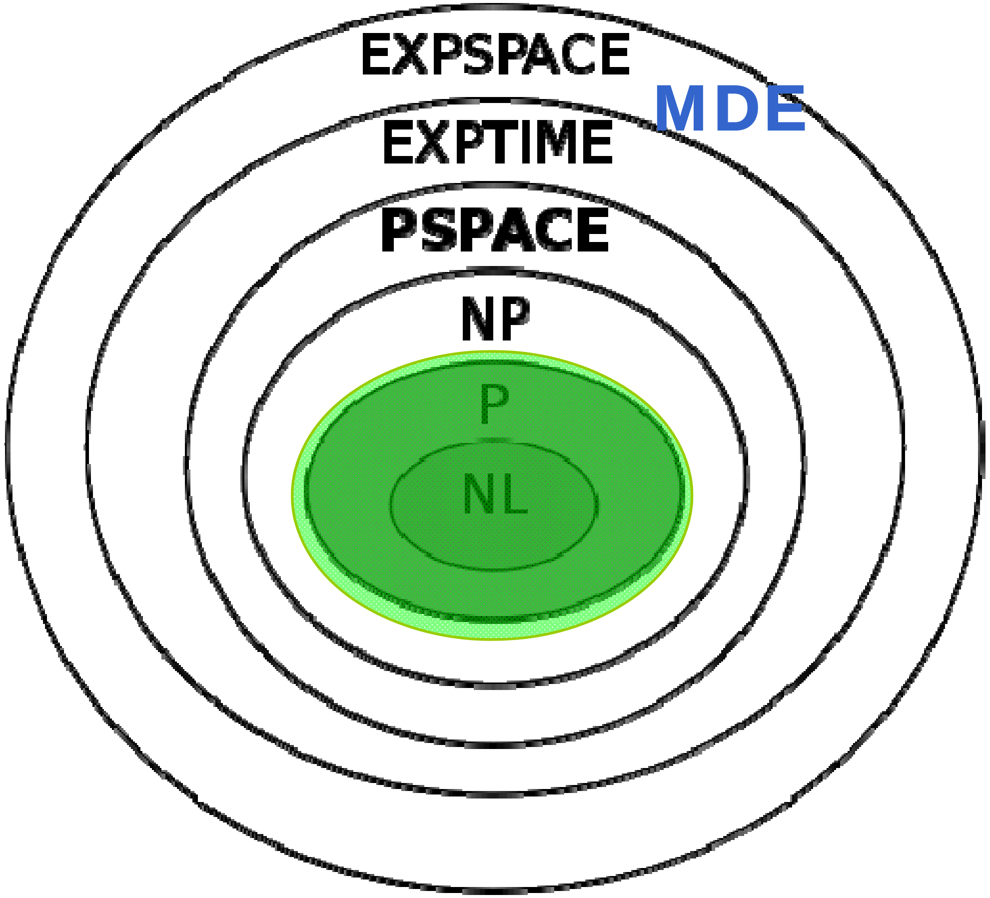
How models are verified?

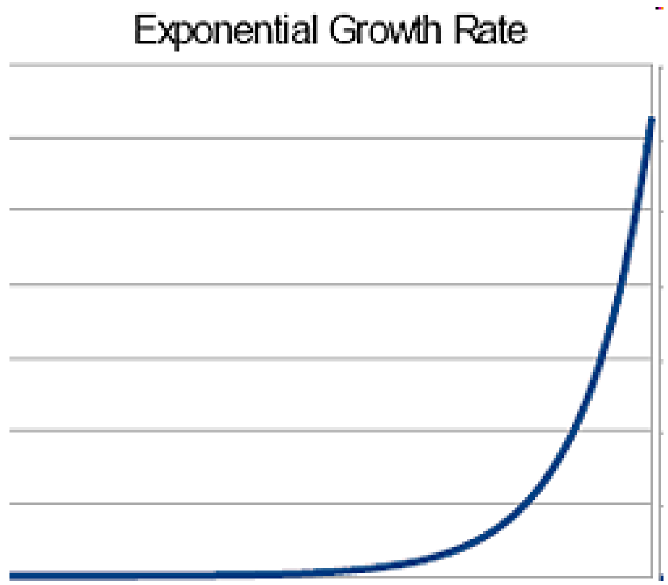




Strong Satisfiability

But Quality classified as a Grand Challenge



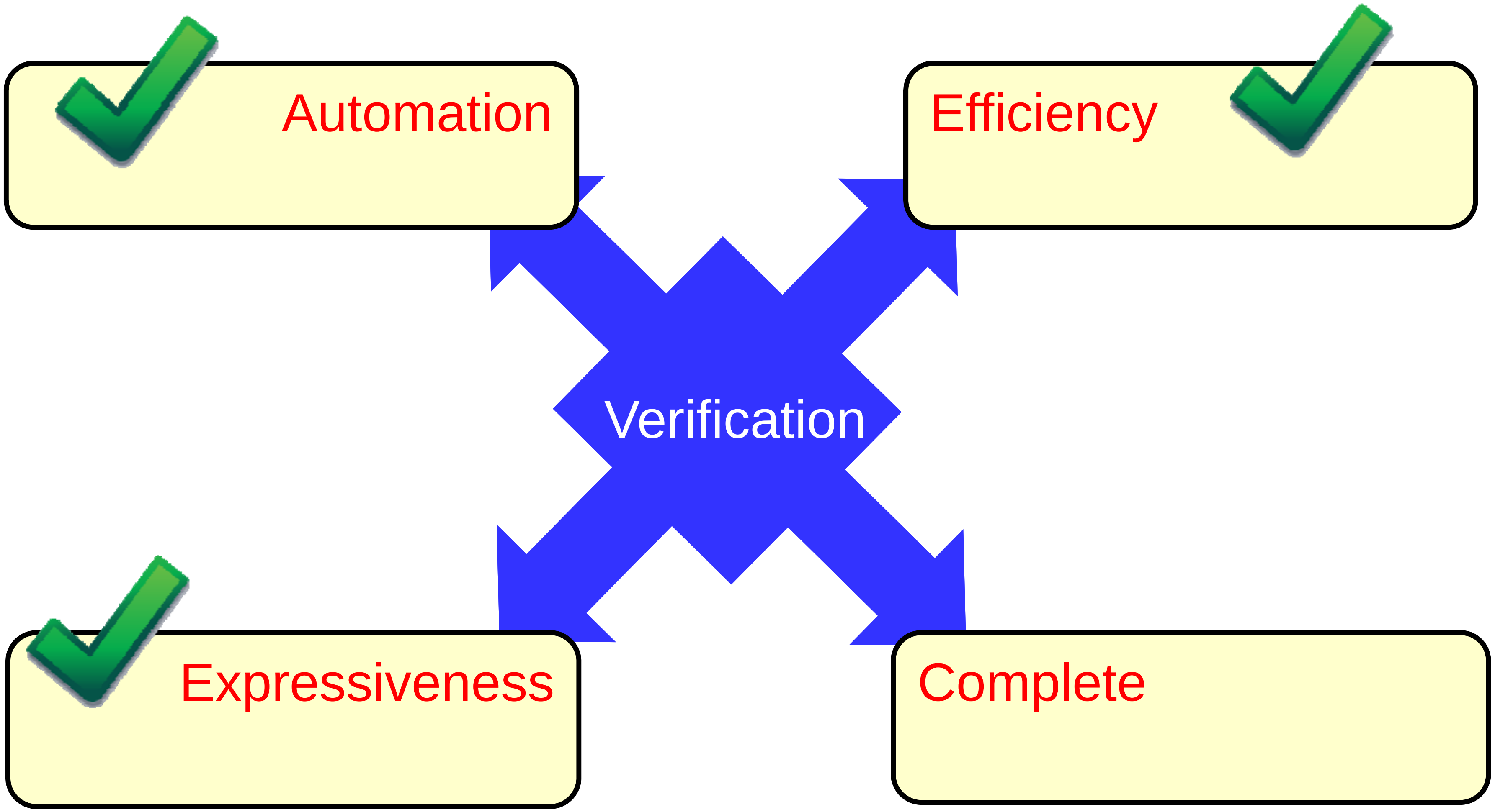


1E+18 scenarios for small models (classes=10)

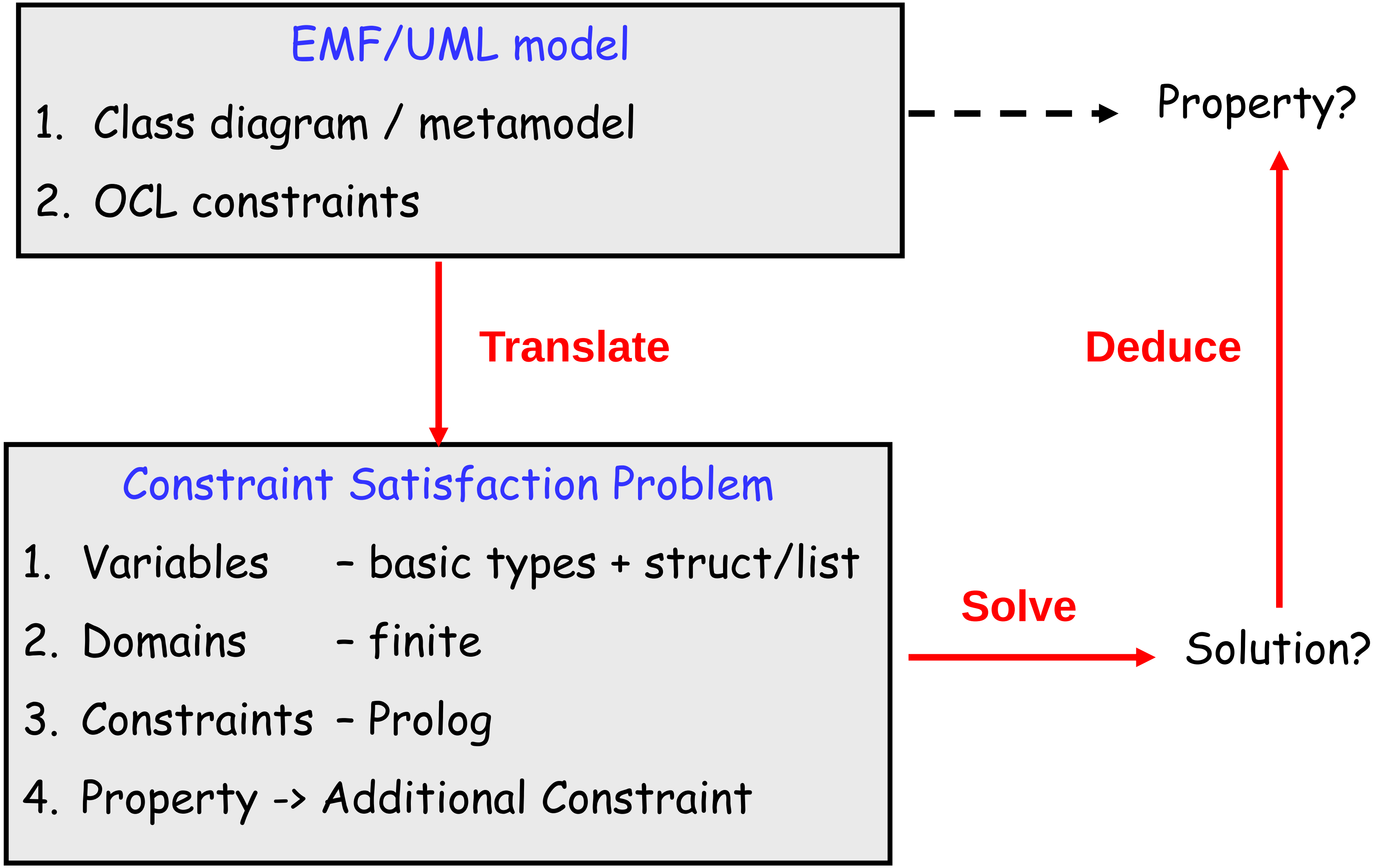
More than cells in the human body!!!

Finding the right trade-off

- No perfect solution exists



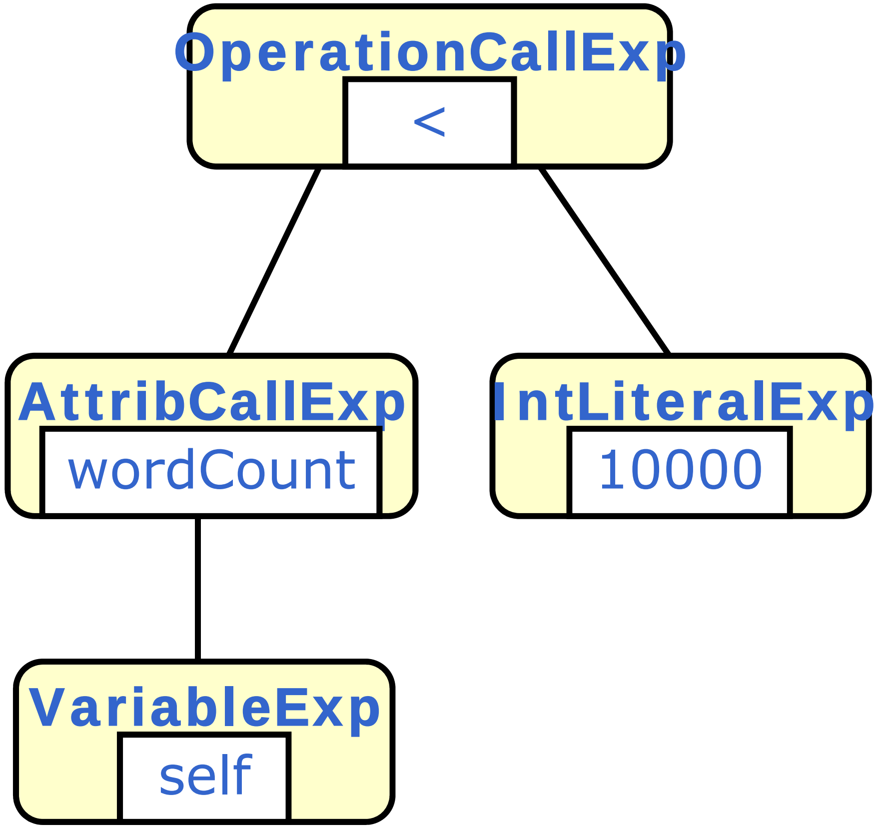
Our “pragmatic” approach



Translation of OCL invariants

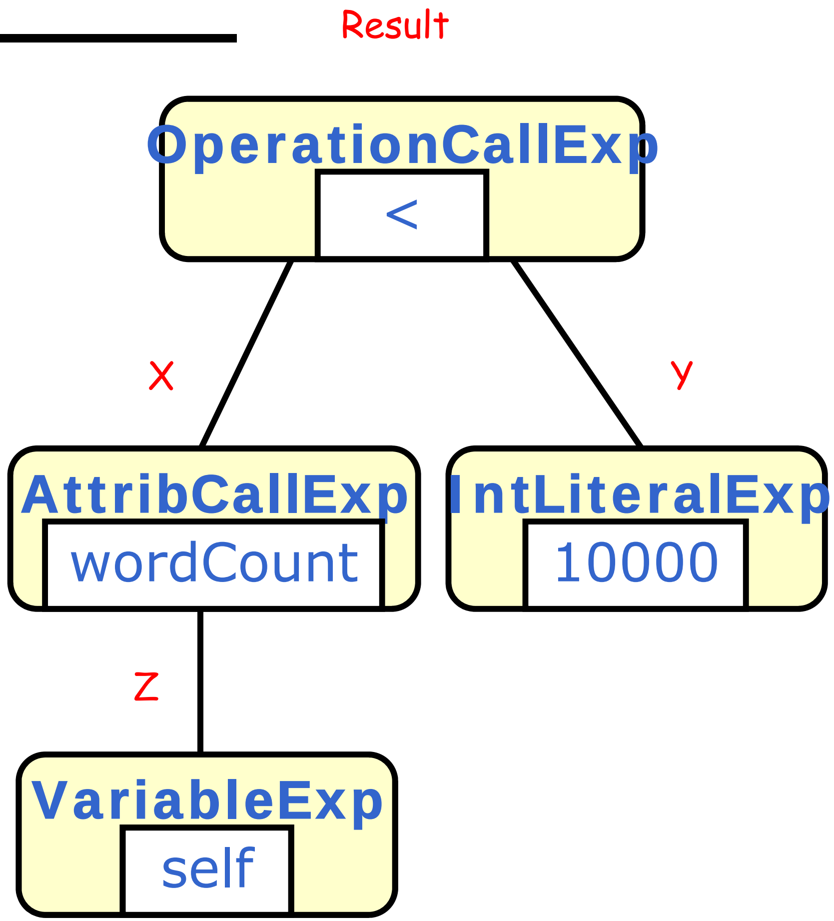
context Paper **inv**: self.wordCount < 10000

- OCL invariant = instance of OCL metamodel
- Invariant becomes a constraint of the CSP



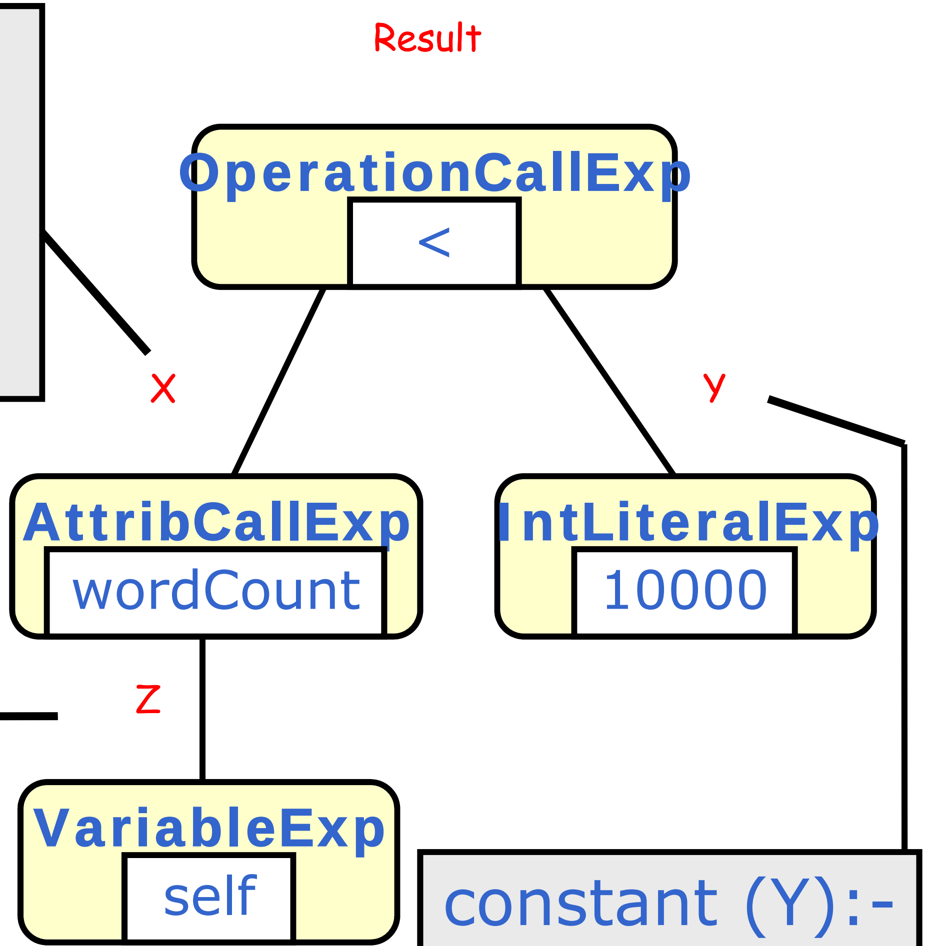
Translation of OCL invariants

```
LessThan(Vars, Result):-
  // Evaluate subexpressions
  Attribute(Vars, X),
  Constant(Y),
  // Compute result
  #<(X,Y, Result).
```



Translation of OCL invariants

```
attribute(Vars, X) :-
  // Evaluate subexpressions
  self(Vars, Z),
  arg(Z, "wordCount", X).
```




```
self(Vars, Z) :-
  // Z is the only visible var
  nth1(1, Vars, Z).
```

```
constant (Y):-
  Y = 10000.
```

Translation of OCL invariants

context Paper **inv**: self.wordCount < 10000

```
invariant(Papers):-
// Expression must be true
// for each Paper
(  for (Paper, Papers)
   do
   LessThan([Paper],Result),
   Result #= 1 ) .
```

OCL Prolog library

- To analyze OCL constraints, it is necessary to provide a translation for all OCL constructs in terms of the Prolog-based language used by the ECLiPSe solver.
- We have extended ECLiPSe with a new library dedicated to the translation of OCL constraints
- The library makes use of the ECLiPSe support for higher-order predicates

  e.g. ocl_set_collect(Instances, Vars, Set, Predicate, Result),
- Removal of symmetries and the suspension mechanism are used to optimize the performance of the library

Resolution of the CSP

Define cardinality variables

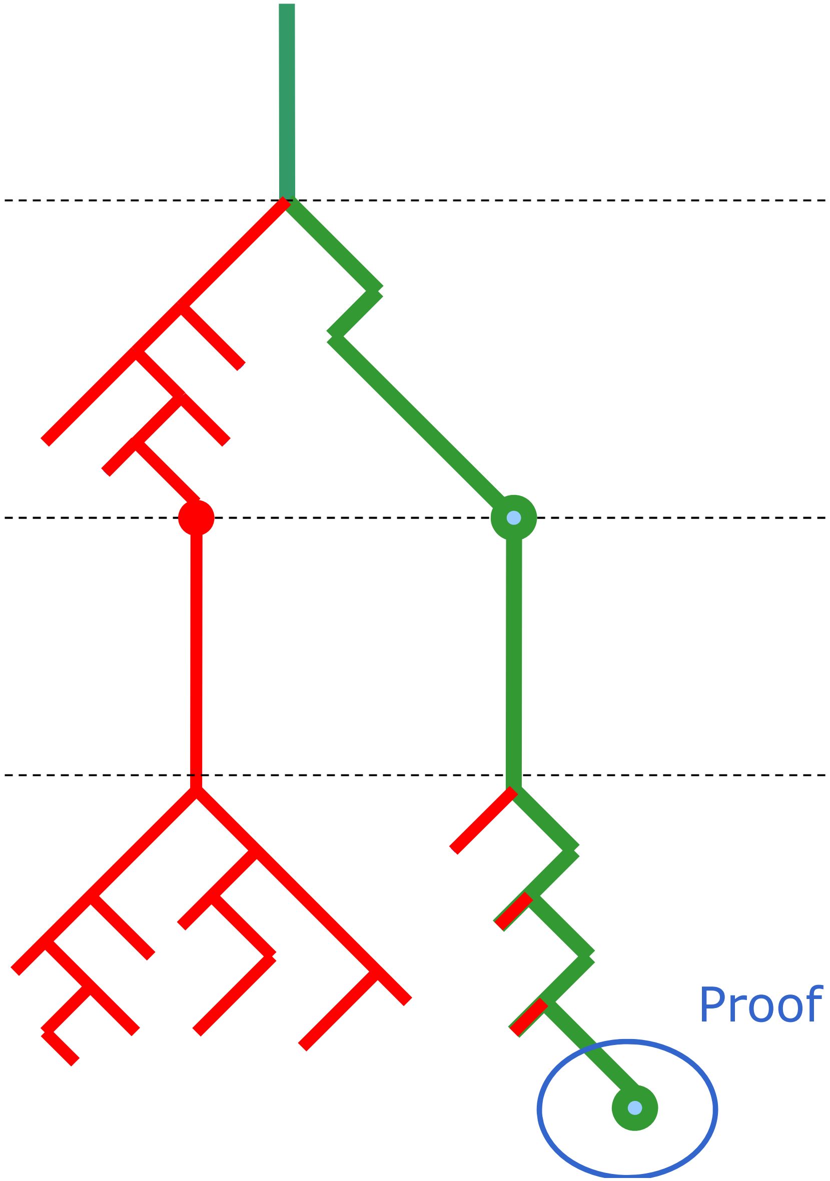
Constraints on cardinalities

Assign cardinalities

Define attribute variables

Constraints on attributes

Assign attributes



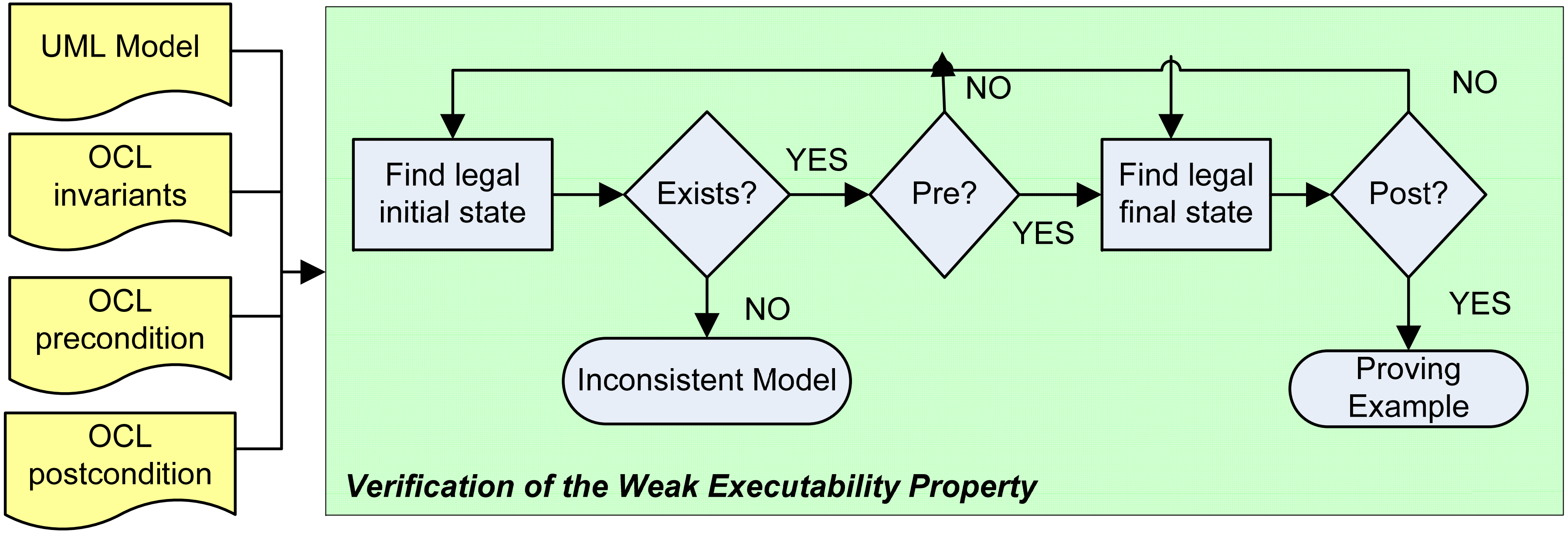
Trade-offs in verification

- Decidability: is automation possible? Yes
- Completeness: proof for any input?
  No, bounded verification
  (but Small Scope hypothesis – D. Jackson)
- Expressiveness: full OCL support? Yes
- Efficiency: Controlled by the user

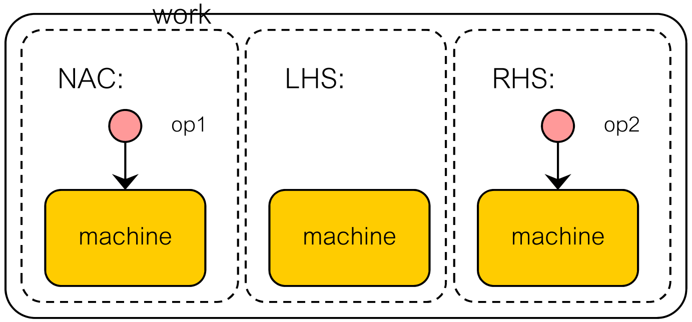
- ***Validation is also possible: we can generate valid instances from partial models***

Other applications (1): Operation contracts

- Verification of contracts: Applicability and executability of operations, determinism,…



Other applications(2): Graph Transformation to OCL



context System::work()

LHS

NAC

pre: Machine::allInstances()−>exists(m|
 not (Operator::allInstances()−>exists(op1|m.operator−> includes(op1)))

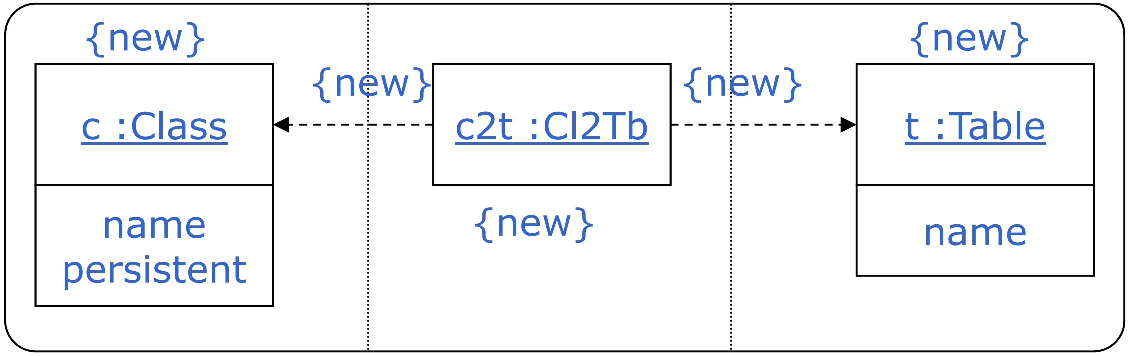
post: Machine::allInstances()@pre−>exists(m| not
 (Operator::allInstances@pre()−>exists(op1|m.operator->includes(p1)))
 and op2.oclIsNew() and op2.oclIsTypeOf(Operator) and m.operator−>includes(op2

RHS

Cabot, Clarisó, Guerra, de Lara: A UML/OCL framework for the analysis of graph transformation rules. Software and System Modeling

Other applications(3): TGG / QVT / ATL to OCL

work



ATTR COND: c.name=t.name and c.persistent=true

Invariants:

context Cl2Tb Inv:
self.class.size()=1 and self.table.size()=1 and self.class.name=self.table.name

context Class Inv:
self.persistent=true implies self.Cl2Tb.size() >=1

context Table Inv:
self.Cl2Tb.size() >=1

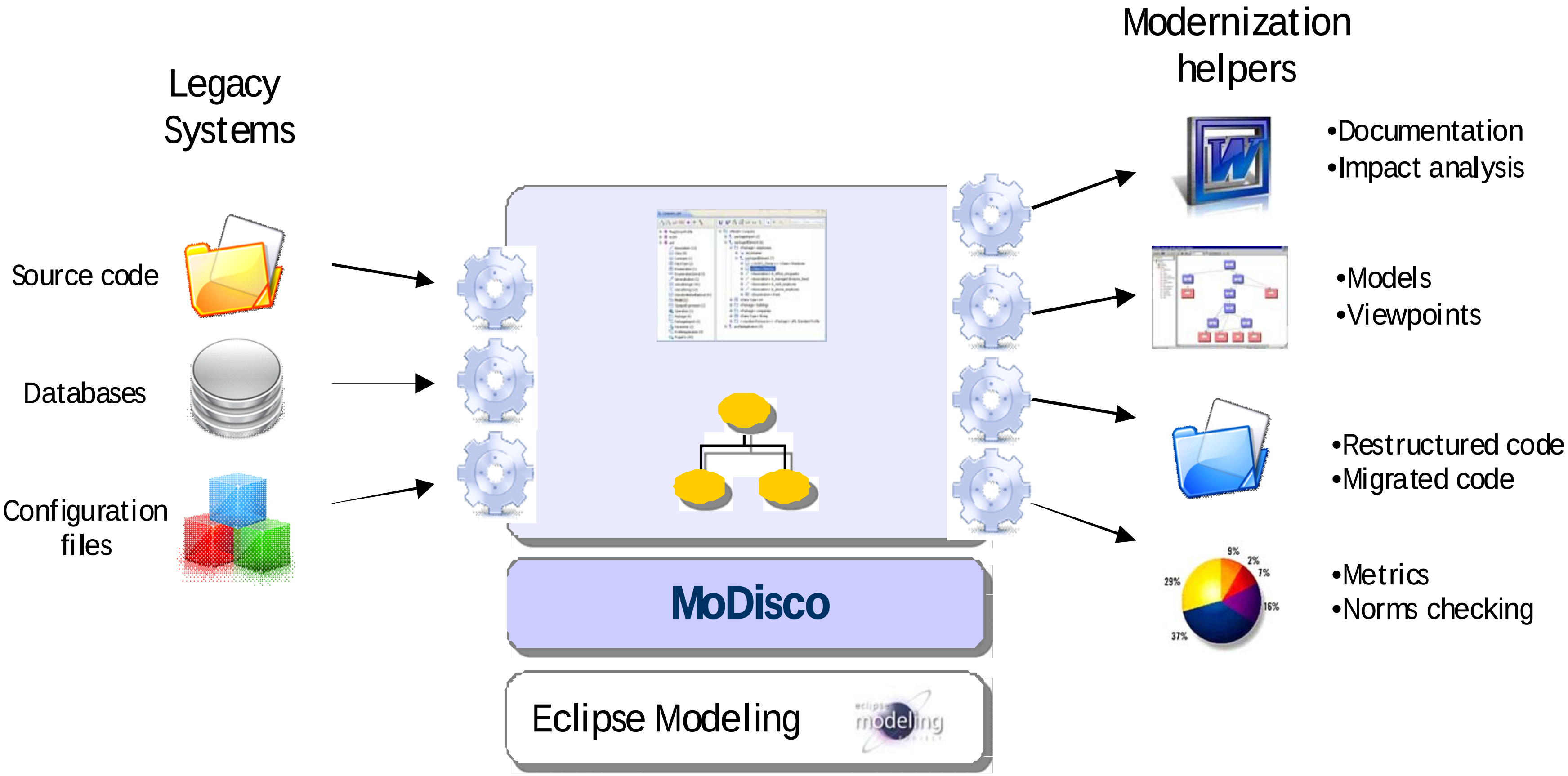
Cabot, Clariso, Guerra, de Lara: Verification and validation of declarative model-to-model transformations through invariants. Journal of Systems and Software

Challenges / Work in Progress

- Incremental verification
- Combine CSPs with SMT solvers to provide a complete verification approach when possible.
- More meaningful feedback (e.g. the subset of constraints that cause the inconsistency)
- Model slicing for parallel verification

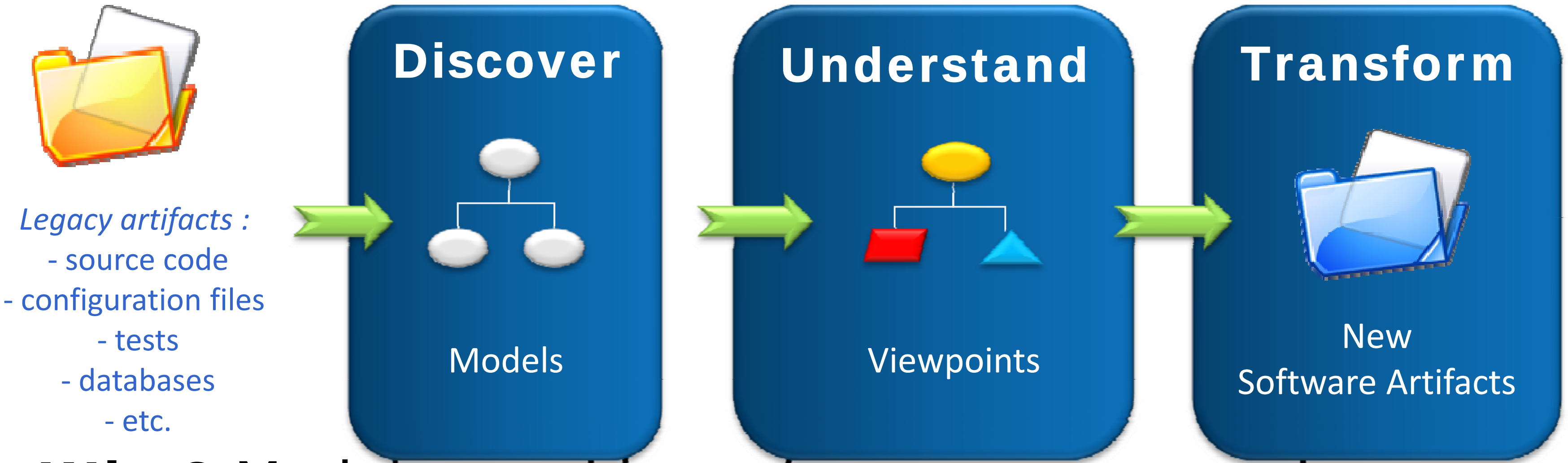
Model-driven Reverse Engineering

MoDISCO: a MDRE framerwork



- *Instead of adhoc Rev. Eng. Solutions, we use an intermediate model-based representation of the legacy system*

MDRE phases in MoDisco



- **Why?** Models provide an homogeneous and interrelated representation of all legacy components.

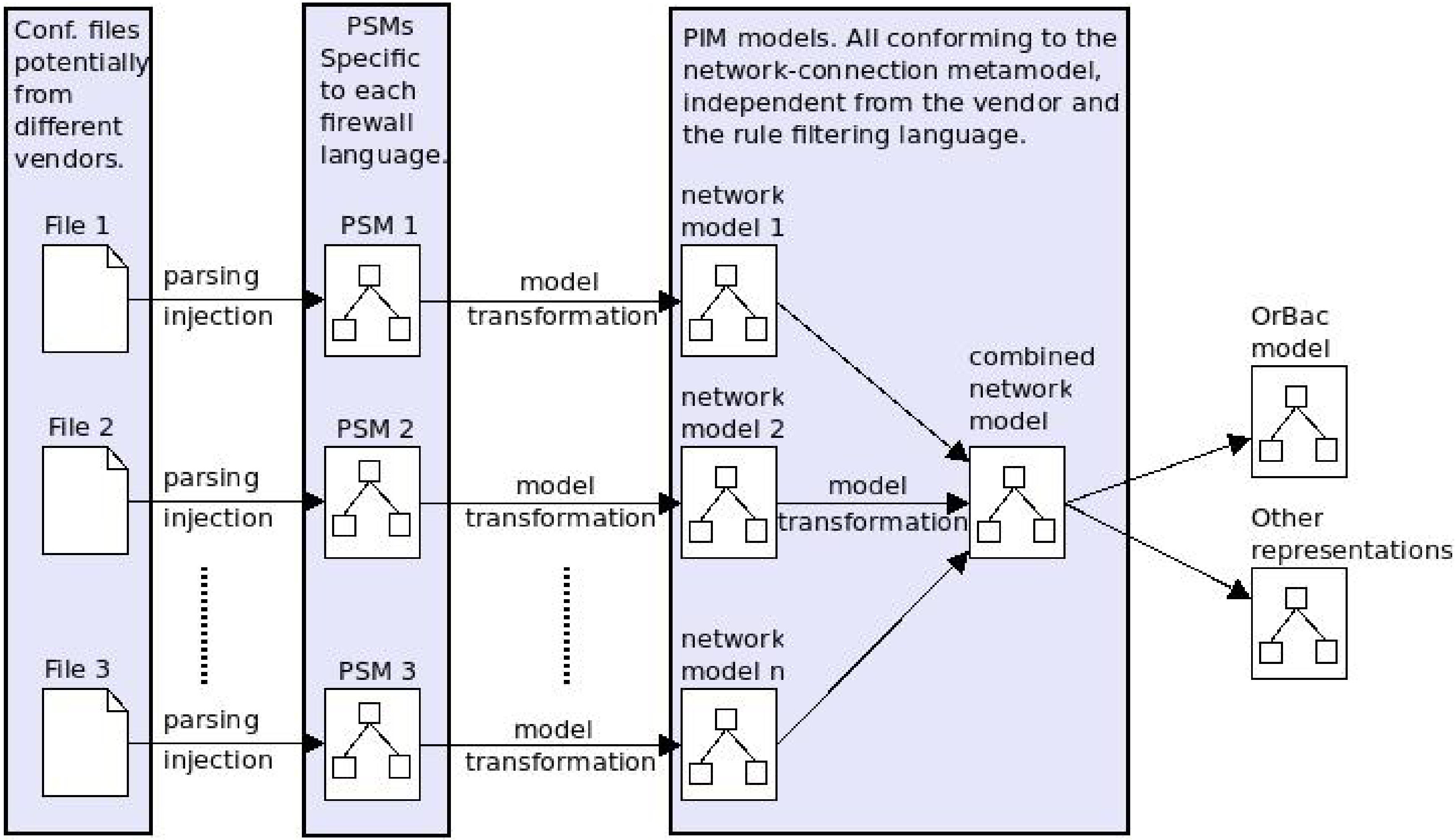
  No information loss: initial models have a 1:1 correspondance with the code

Challenges / Work in Progress

- Extraction of business rules
  - Getting a model of the code helps but it’s still too low-level for a stakeholder to look at it
  - Semi-automatic approach with IBM.
- Rev Eng of the whole software system
  - We know how to extract a model from a single component but we don’t take into account its relationship with other components (esp. in other layers)
  - Companies are interested in knowing the enterprise architecture (e.g. TOGAF) of a system

Challenges / Work in Progress

- Reverse Engineering of security policies



Social aspects of MDE

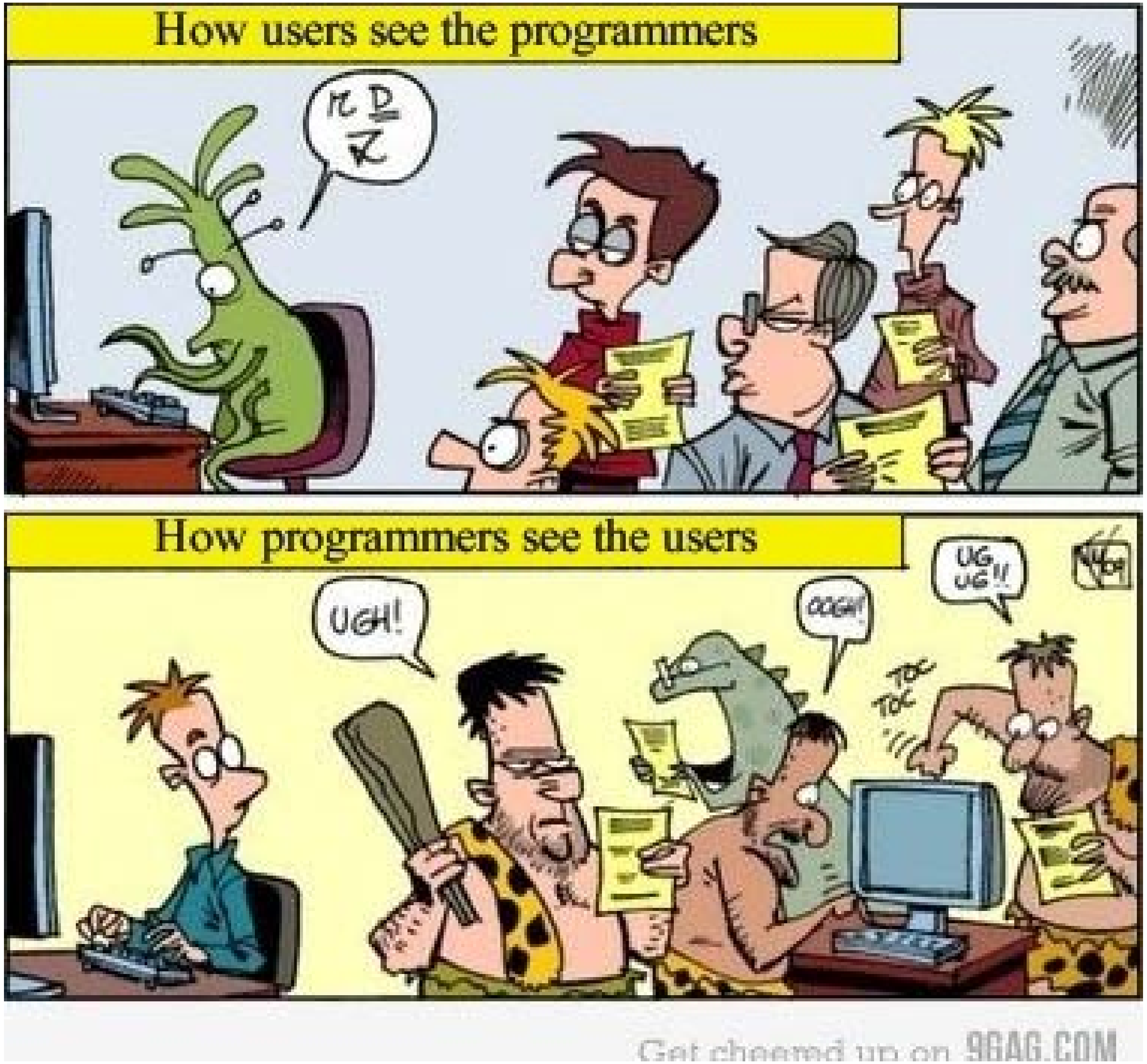
Social aspects of MDE

- We need a better understanding of the needs of users (technical and non-technical) to make sure we solve their actual problems (and not the ones we think they have).

- One example: Huge amount of research on providing methods for the specification, validation, etc of non-functional requirements:
  - Nontechnical constraints (like cost, type of license, specific providers) are as prominent as technical requirements like performance or security
  - Modern technology platforms already cover many applications' quality requirements so explicit NFR management not so useful
  - NFRs are hardly ever documented and poorly validated

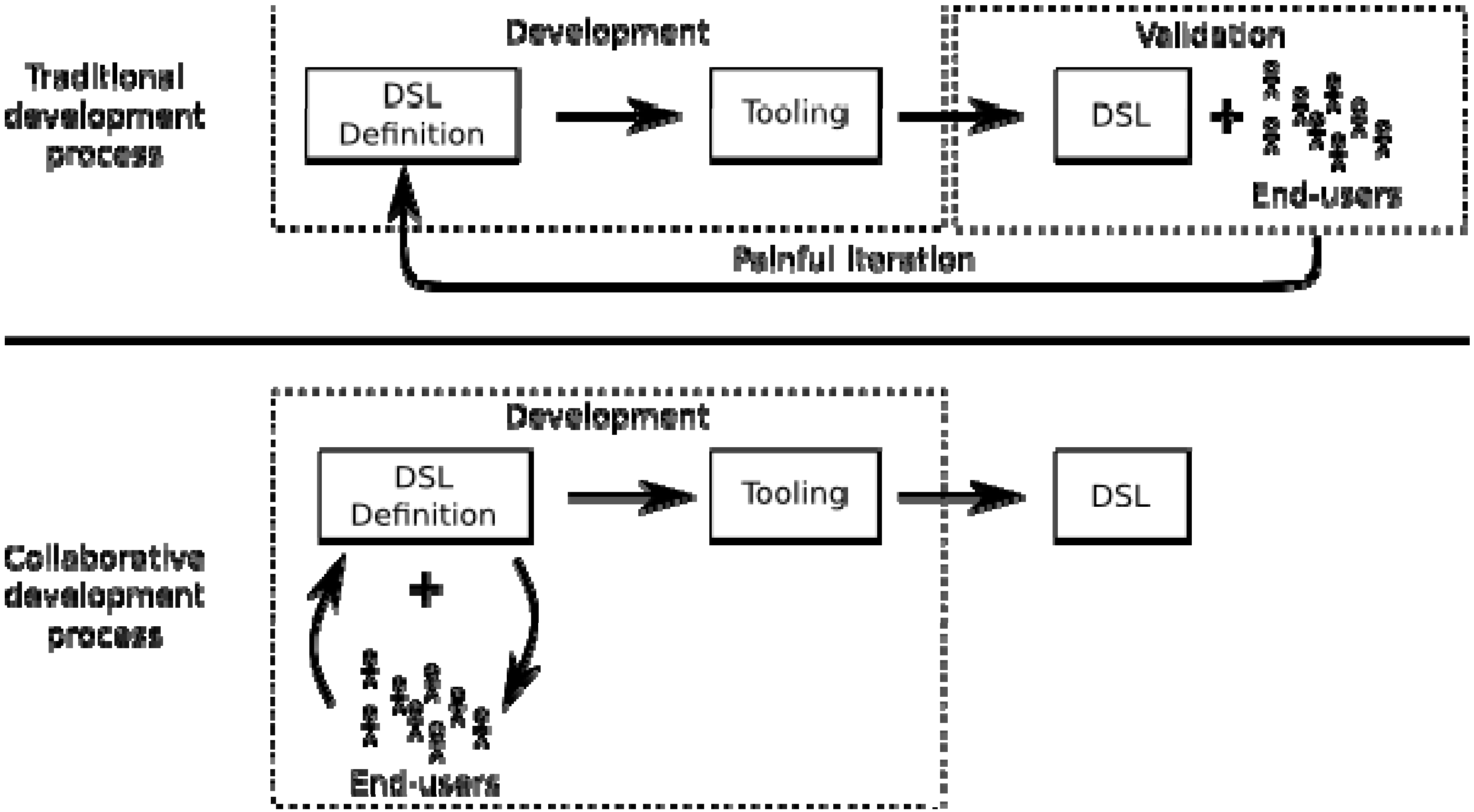
Architecture Quality Revisited. Buschmann, Frank; Ameller, David; Ayala, Claudia P.; Cabot, Jordi; Franch, Xavier. IEEE Software, vol. 29 (4)

Dealing with users is not easy

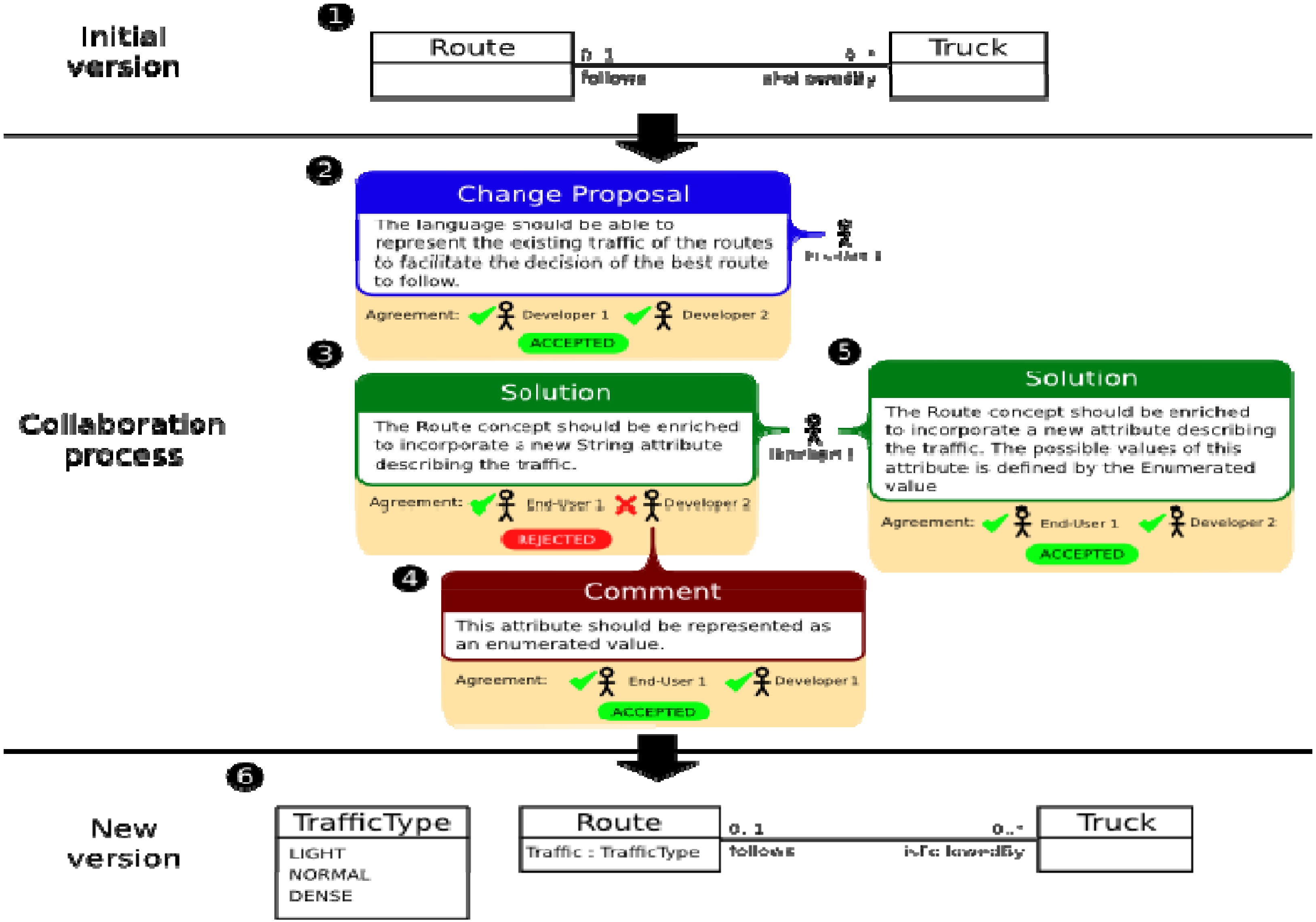


Collaborative development of DSLs

- Participation of end-users is specially relevant when creating DSLs since we are creating a language for them



Collaborative development of DSLs



Collaborative development of DSLs

- Collaboro: process + DSL + tool to enable the collaborative development of DSLs
- Users suggest changes to both abstract and concrete syntax levels
- The community comments and votes changes and solutions
- Once an agreement is reached (based on a given decision policy, e.g. unanimity) the solution is added to the current language version
- We get:
  - Languages that better satisfy the users' needs
  - Traceability to justify the rational behind the language design decisions

Challenges / Work in progress

- Gamification techniques to promote more user interaction
- What constitutes a good concrete syntax for DSLs?
- Learning from web designers: AB testing for DSLs
  - E.g. evolve the concrete syntax based on which alternative gets more “conversions”).

Very large models

Scalability

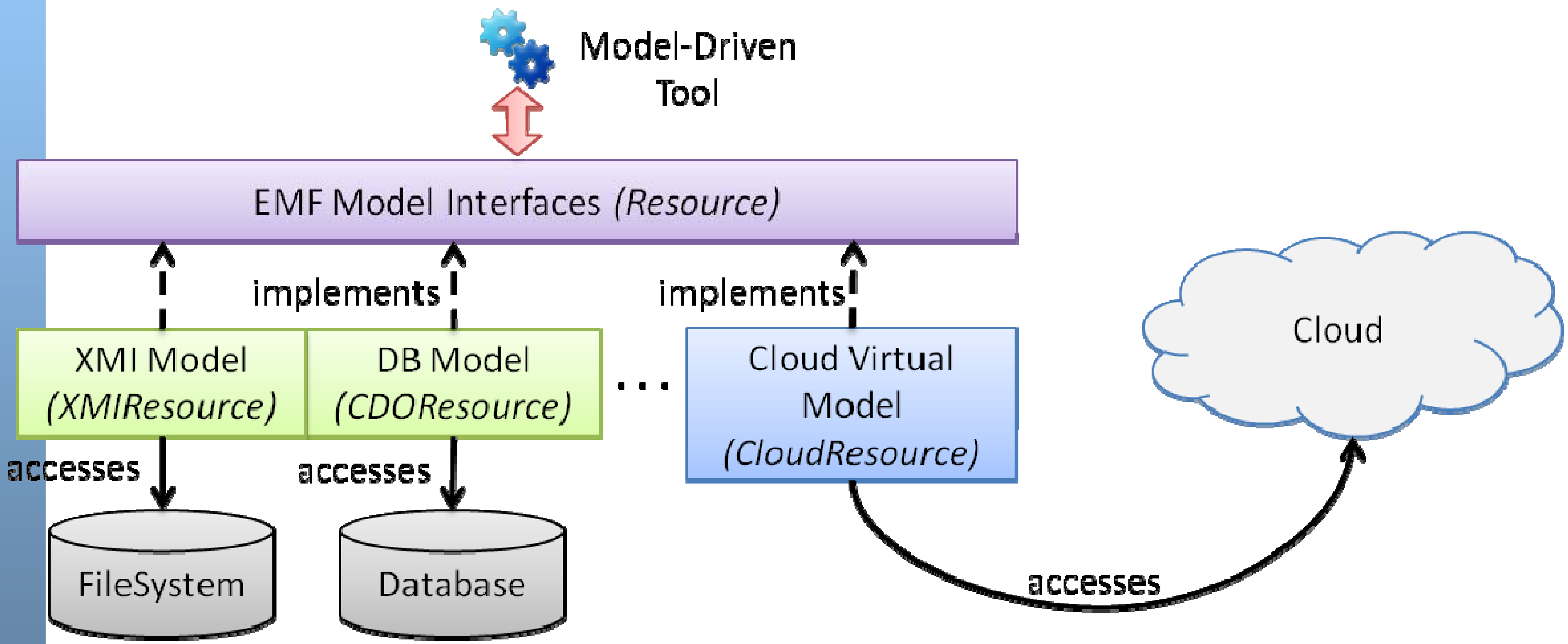
- MDE tools fail when dealing with very large models
  - E.g. reverse engineering of the Eclipse platform generates a model of more than 5M instances
  - EMF just crashes with these volumes
- Scalability important both at the model (loading very large models) and model manipulation level (executing complex transformations on large models)
- Key problem in industrial scenarios but far from a trivial one
  - Very Large DataBases is the main conf. in the database domain with 37 editions and still looking for solutions

Modeling in the cloud

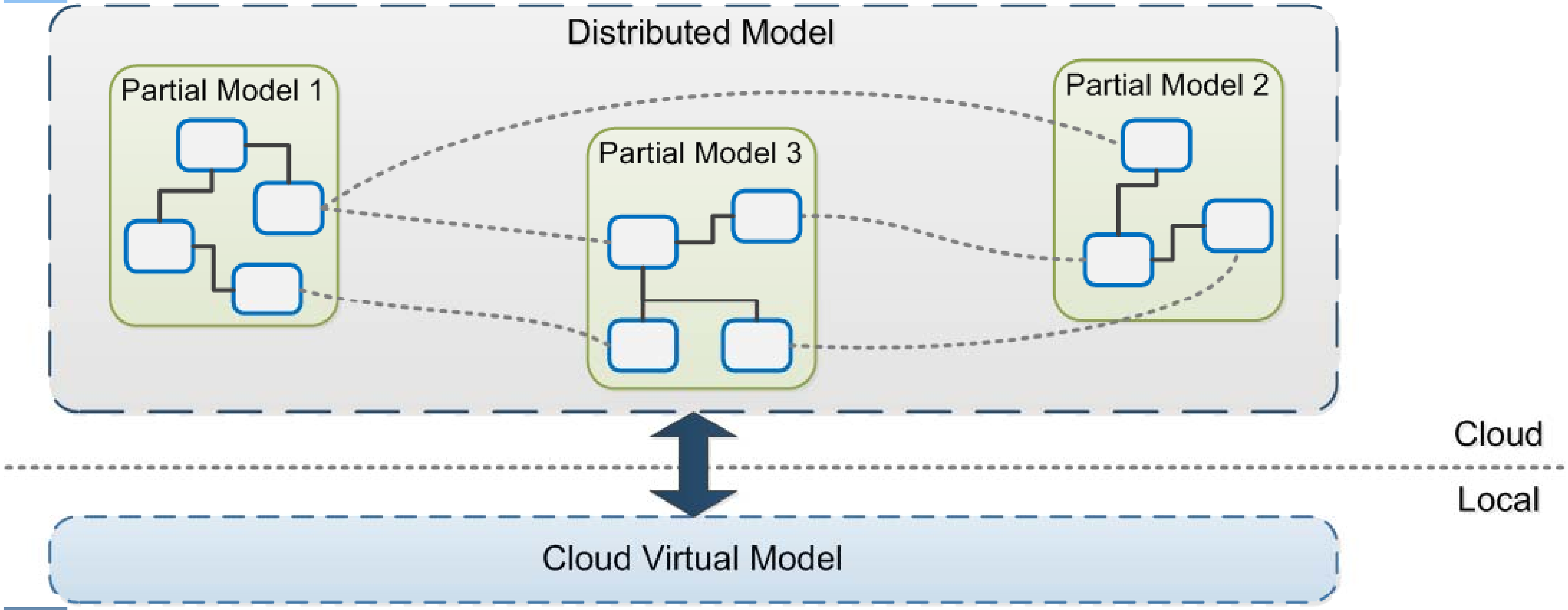
- Can the Cloud be used to handle VLMs?

- Two key aspects:
  - Model storage in the Cloud for efficiently storing and loading VLMs
  - Model transformation in the Cloud for distributing the computation of the transformation
    - Each virtual node executes the full transformation on a subset of the model
    - Each virtual node executes part of the transformation on the full model

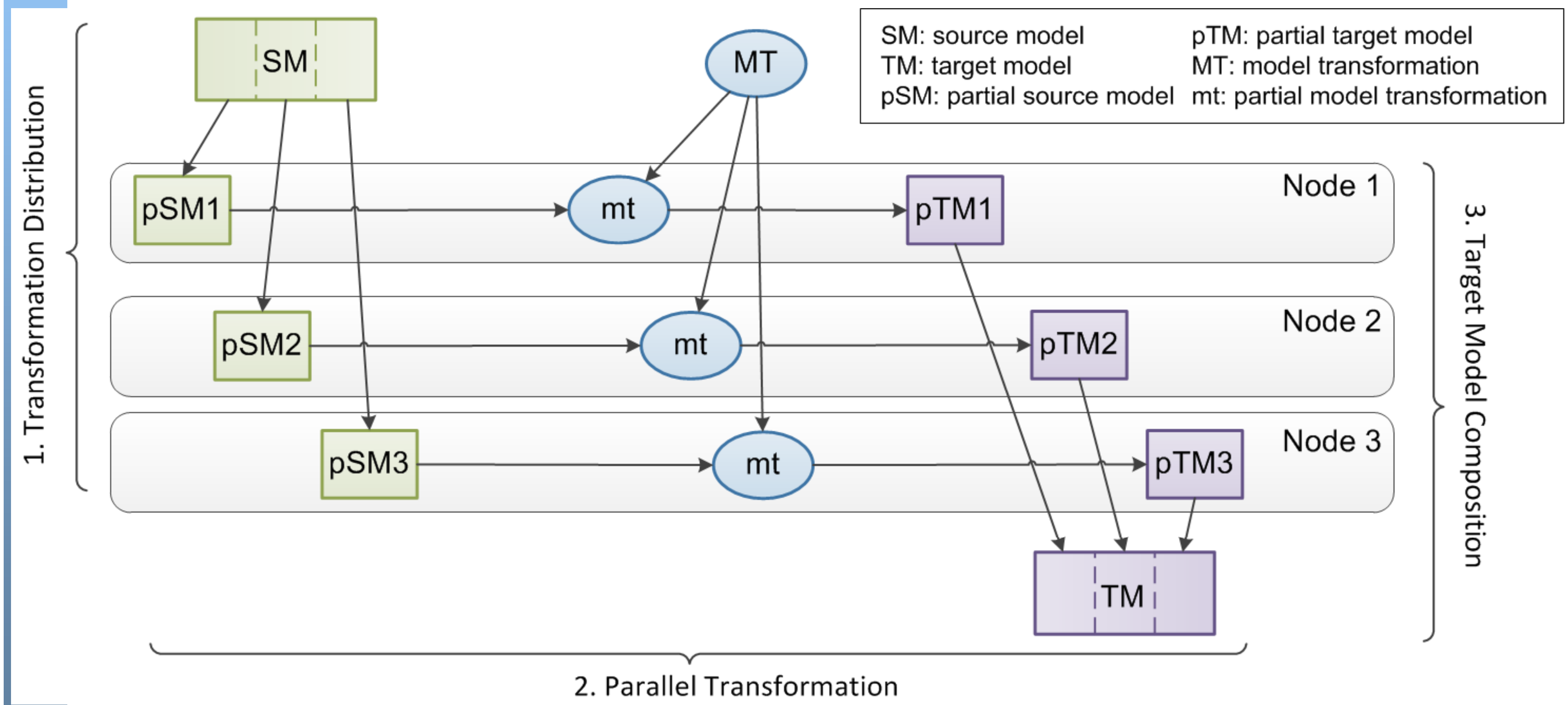
Cloud Virtual Model



Cloud Virtual Model



Parallel Transformation Overview



Challenges / Work in Progress

- Optimal way to split the models and transformations
  - And when not to split them
- Best technology for the backend?
  - Hadoop for storing the models plus MapReduce for the transformations?
- Reactive transformation engine:
  - Automaticaly activate only the strictly needed computation in response to updates or requests of model elements.
  - Incremental, minimal set of recalculations

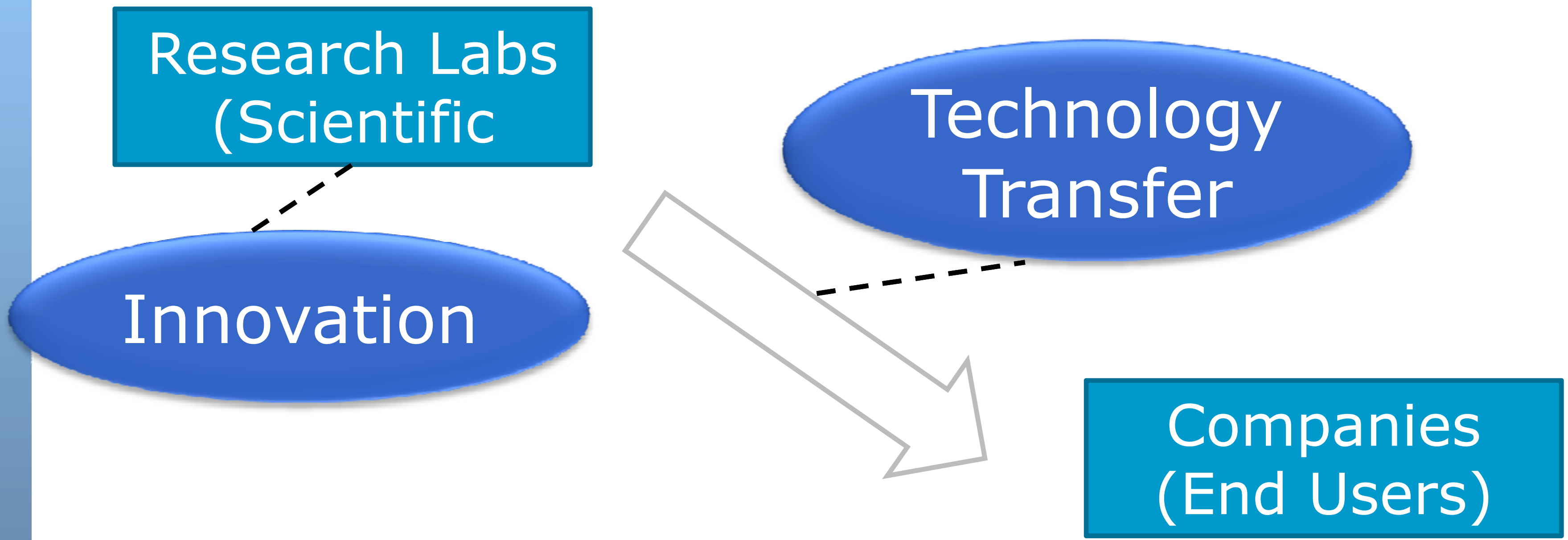
Index

- Introducing MDE
- Research in MDE
- A Research Agenda for MDE 2.0
  - Models & Quality
  - Legacy systems
  - Social aspects
  - Very Large models
- **Dissemination and technology transfer**
- Credits
- Conclusions

Always with technology transfer in mind

How do we bridge the gap?

- Traditional direct approach

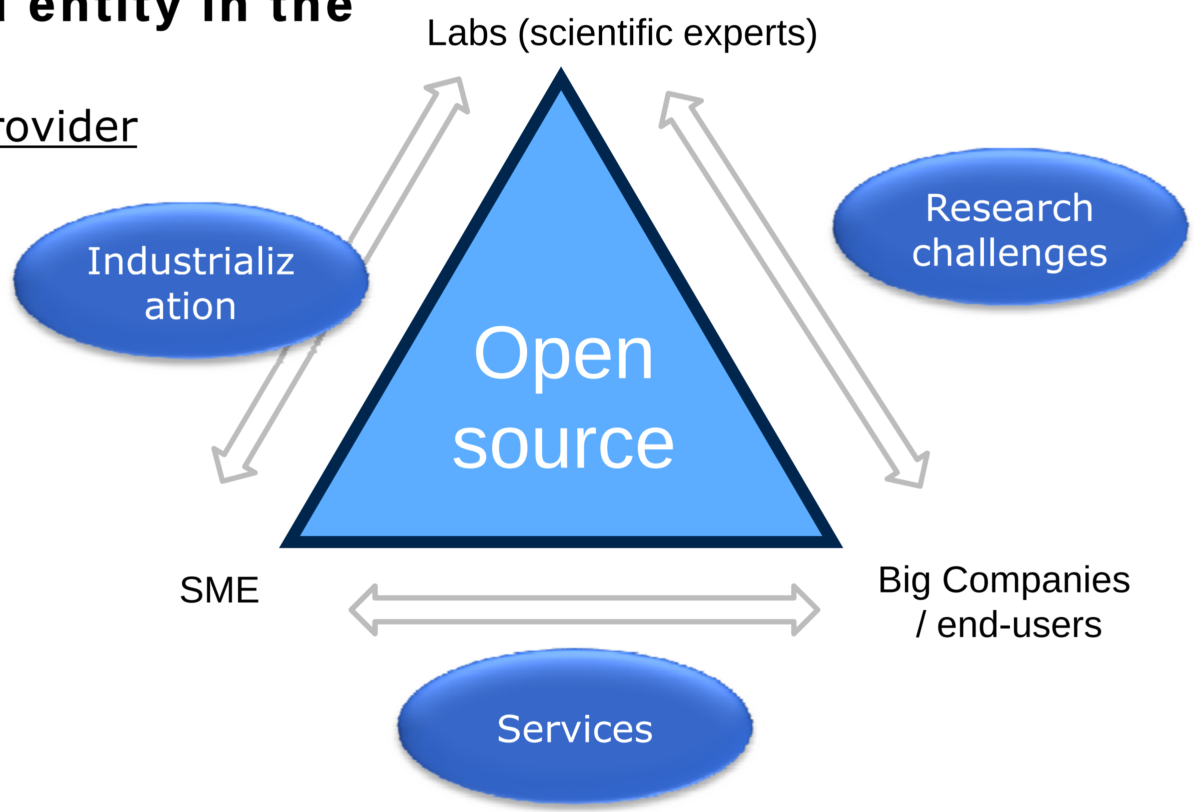


Direct technology transfer doesn't work!: Our tools are not good and they won't magically become great

We need a new business model

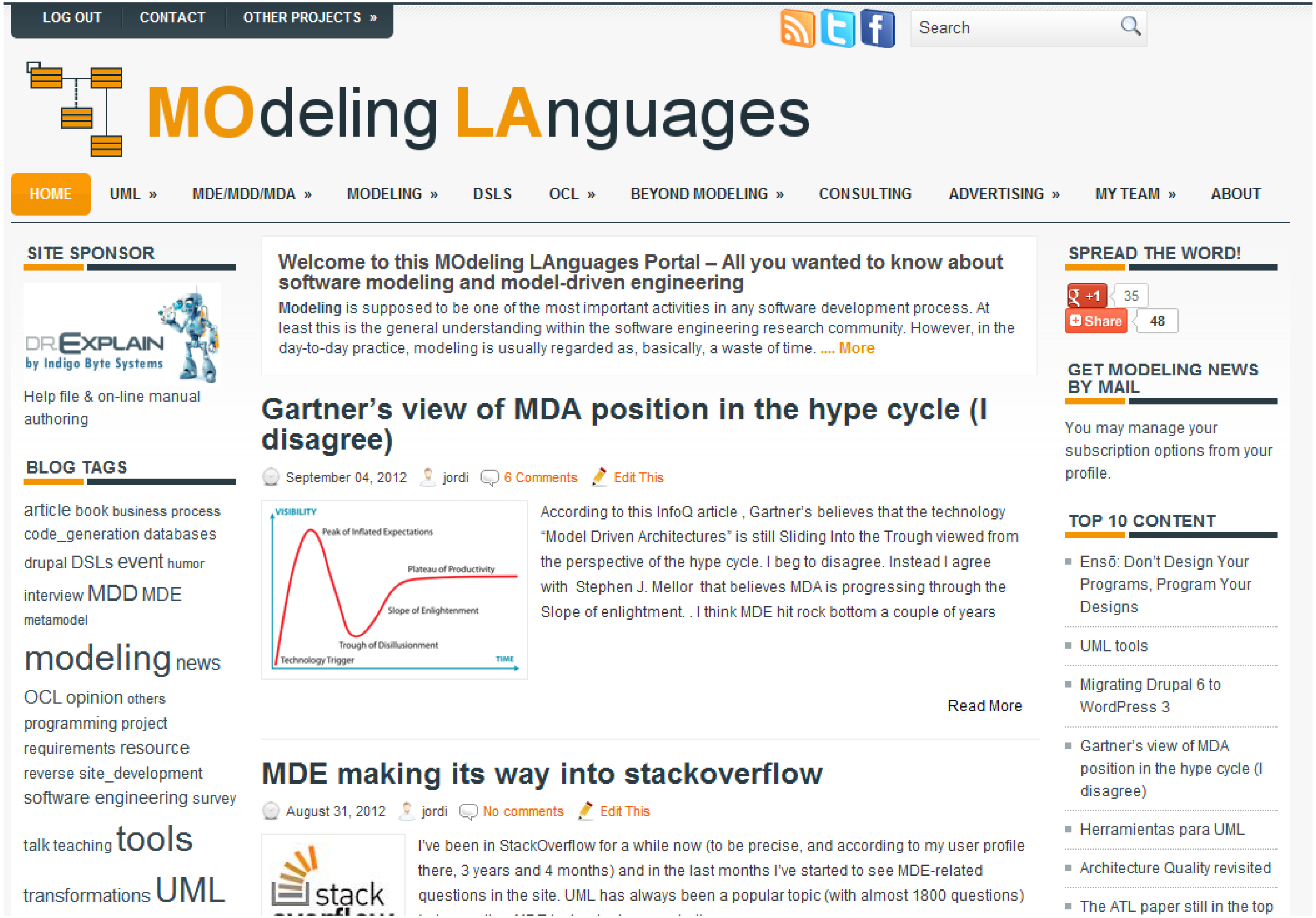
- Three-entity approach
- **Introduction of a third entity in the process**
  - A SME as Technology Provider

  - Play the role of Interface between researchers and users



Dissemination

The Modeling Languages portal



MORGAN & CLAYPOOL PUBLISHERS

Model-Driven Software Engineering: MDE in Practice

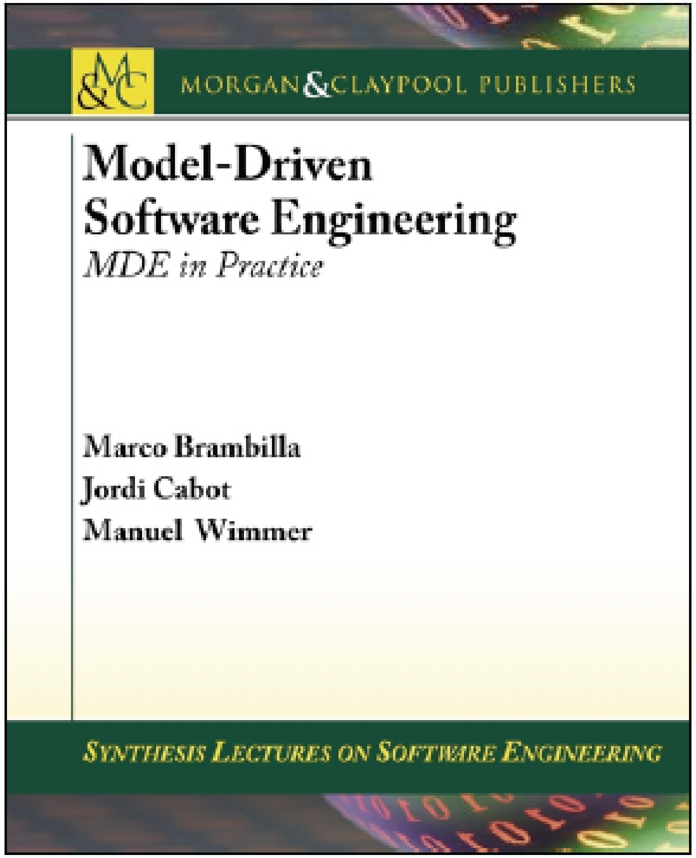
Marco Brambilla, *Politecnico di Milano*
Jordi Cabot, *Ecole des Mines de Nantes*
Manuel Wimmer, *Vienna University of Technology*

Foreword by Richard M. Soley, OMG

Series: Synthesis Lectures on Software Engineering

This book discusses Model Driven Engineering (MDE), which is the use of model-based approaches to improve the daily practice of software professionals. MDE practices have proved to increase efficiency and effectiveness in software development, as demonstrated by various quantitative and qualitative studies. MDE adoption in the software industry is foreseen to grow exponentially in the near future, e.g. due to the convergence of software development and business analysis.

The aim of this book is to provide the reader with an agile and



ISBN
9781608458820

Credits

Credits

- Research is always a team activity
  - Works on conceptual modeling and code-generation together with colleagues from Technical University of Catalonia and Politecnico di Milano
  - Quality line started together with R. Clariso (Open University of Catalonia) and J. de Lara and E. Guerra (Autonomous university of Madrid)
  - Univ. of Toronto taught me the importance of social and organizational aspects of Software Engineering
  - … thanks also to and many more co-authors that I can't list here

- Current research lines possible thanks to the members of the AtlanMod team (EMN / INRIA / LINA)

- AtlanMod was created in 2008 by Jean Bézivin. He is the "father" of MoDisco and of the technology transfer model we are following.

Wrapping up

Conclusions

- MDE is changing the way we build software
  - Though this “we” is still limited
- We have explored some of the (IMHO) promising research directions to improve MDE (and its adoption):
  - Quality, scalability, human aspects, legacy systems
- MDE as a means to an end → Better Software Engineering
- It is ok to renounce to perfect solutions in exchange of useful ones
  - The good-enough revolution (Wired – 09/2009)