Multi-Level Language Engineering, Modeling and Software Development with the FMML^{x} and the XModeler^{ML}

LE4MM

Aston University
BIRMINGHAM UK

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Tony Clark
Ulrich Frank
Daniel Töpel
Michael Bartels
Luca Mattei

Before We Start …

What is the difference between a model and a modeling language?

Why would you generate code from models?

Are objects on M0 parts of models?

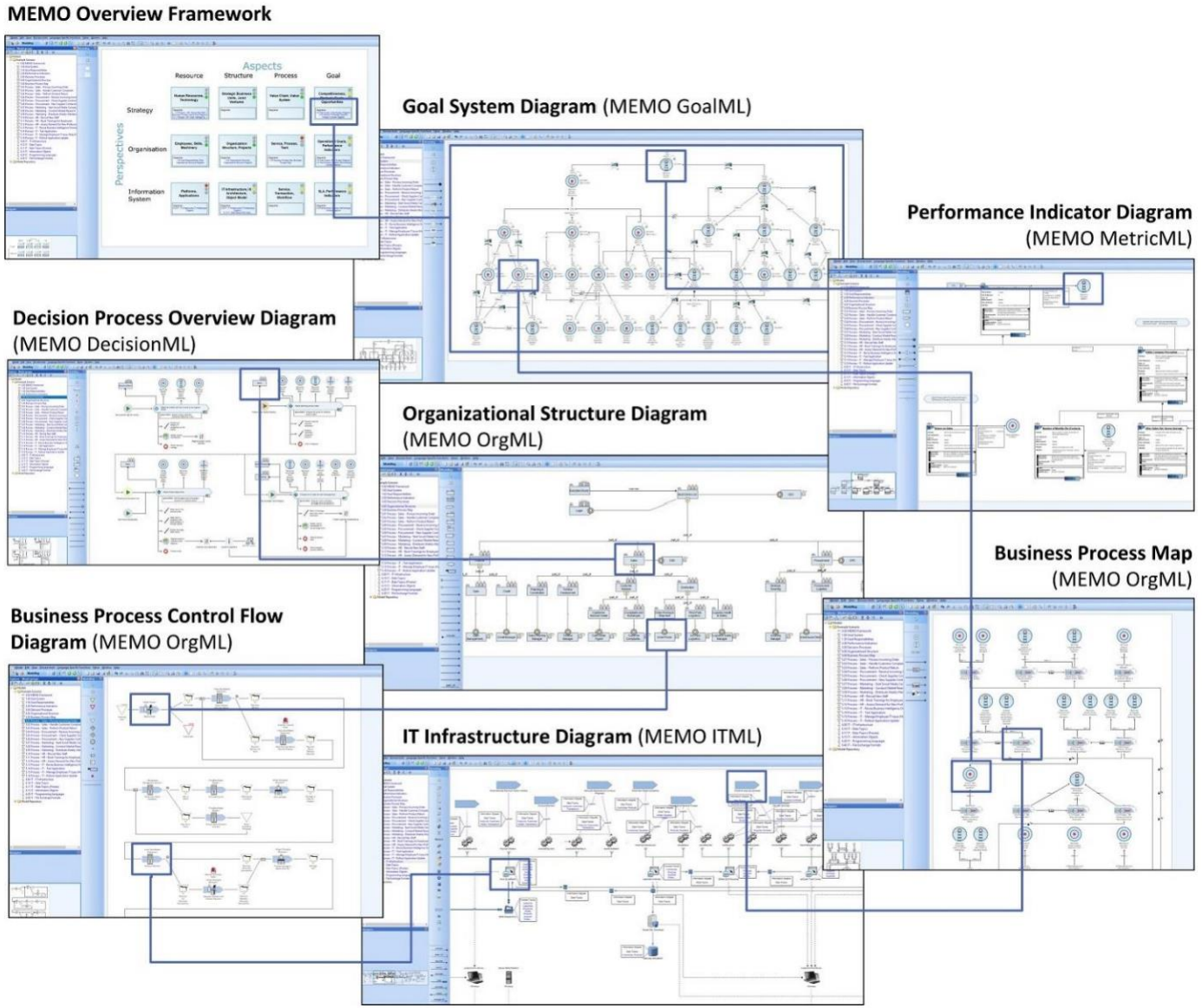
What is the difference between programs and models?

What is your vision of future enterprise software?

Overview

- Motivation
- FMML^{x}
- XModeler^{ML}
  - Components
  - Integrated Development and Use of Languages, Models and Apps
- Development and Use of Textual DSLs

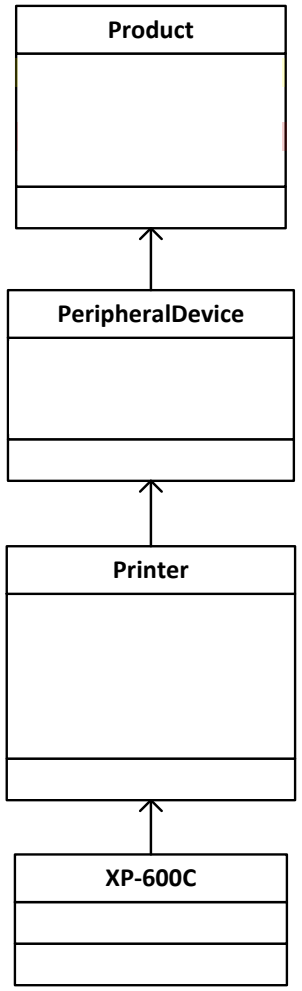
Background: DSMLs for Enterprise Modeling



DSML Development: Sources of Frustration

- lack of expressiveness
- unnatural dichotomy of language and model
- fundamental design conflicts
- pain of model/code synchronization
- ….

Lack of Expressiveness



Process

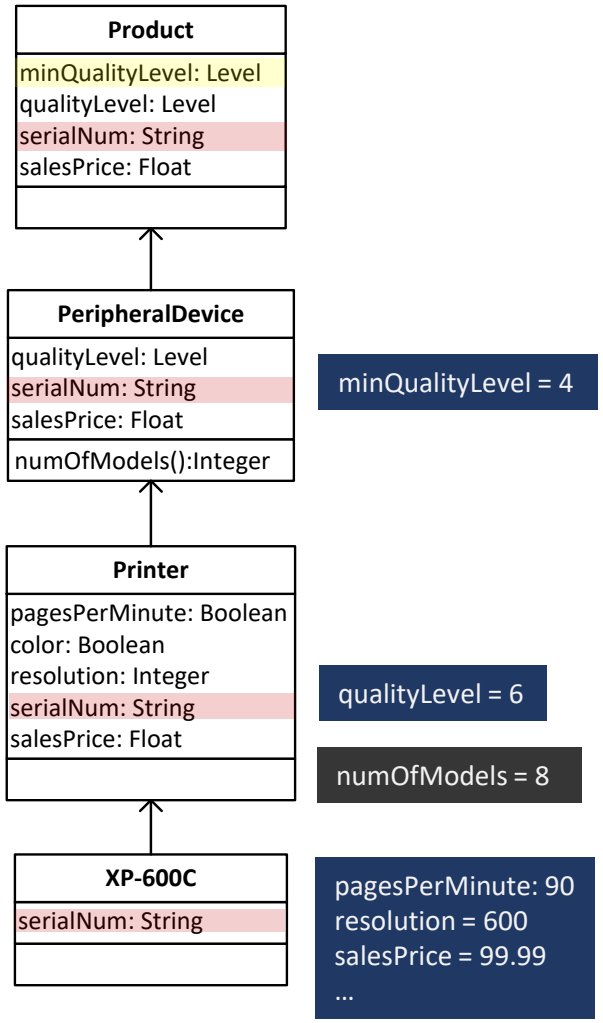
OrderManagement

p1: OrderManagement

Where to specify that every process execution has a start time?

Generalization or Classification?

Lack of Expressiveness



Sometime no clear distinction between instantiation or specialisation – both make sense.

Every (meta) class is an object (has state, can execute operations).

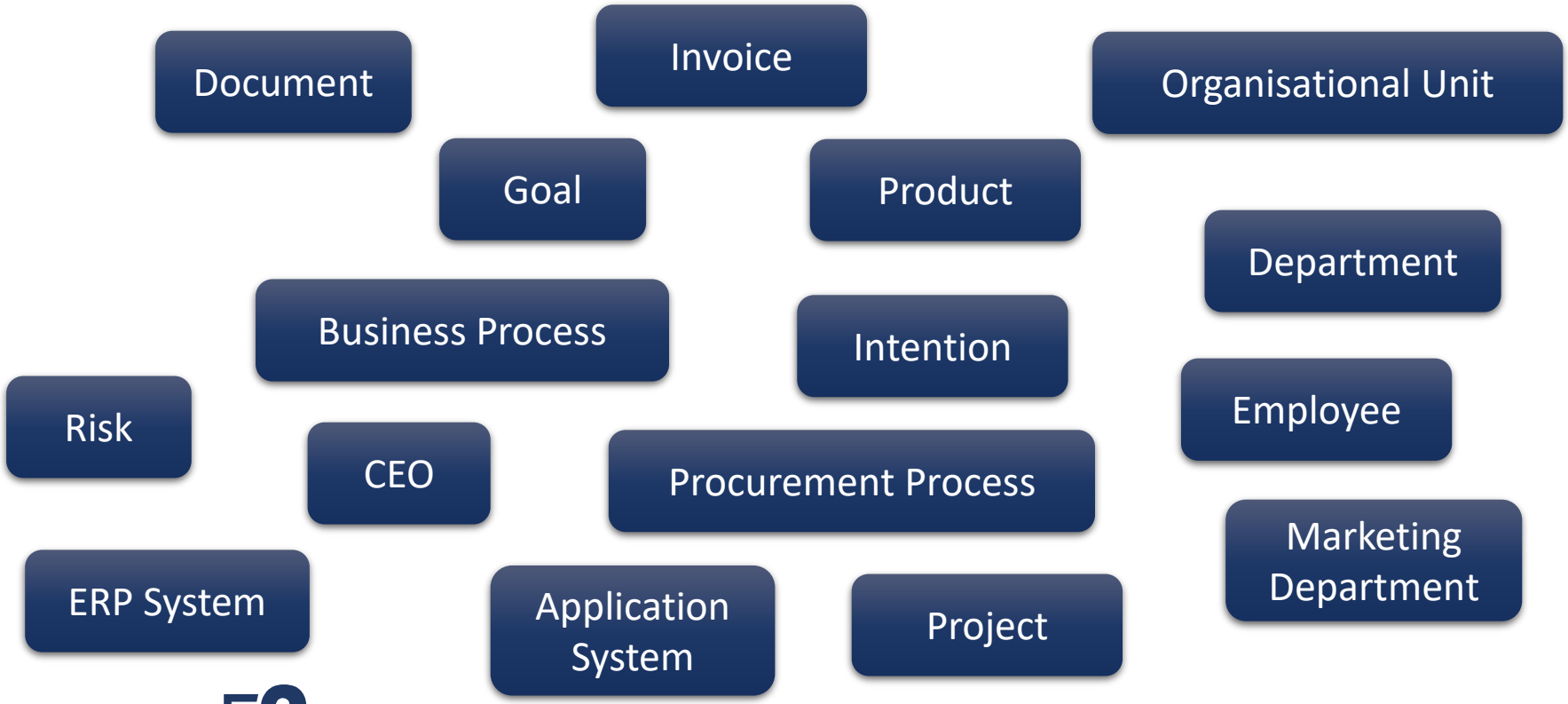
There are multiple levels of classification.

Instantiation may be deferred to lower levels.

Iron Law: “Express knowledge at the highest level.”

How could that be implemented?

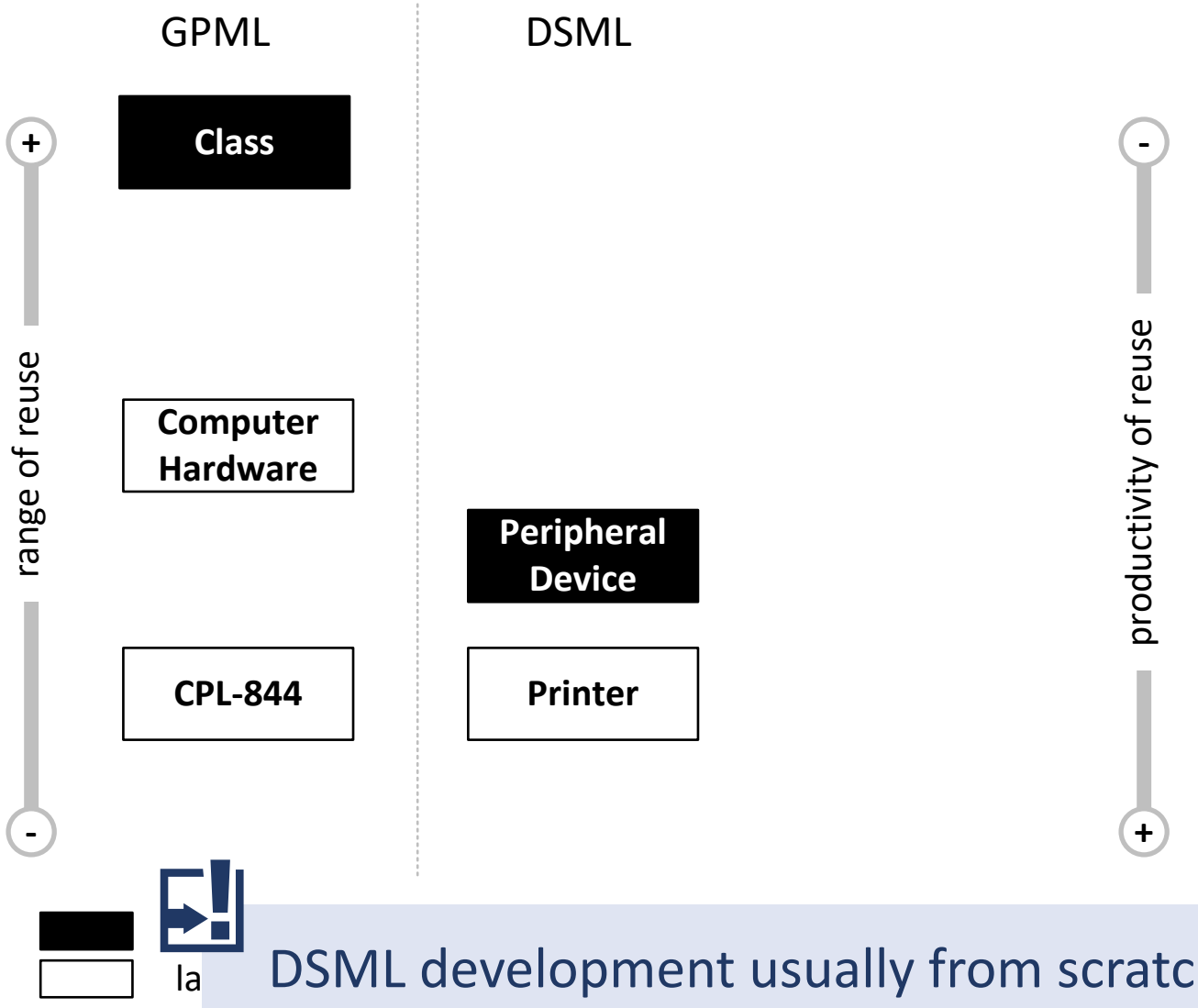
Distinction of Language and Model



Language concept or language application?

Traditional paradigm requires distinction, but how?

Power-Generality Conflict



DSML development usually from scratch!

Fundamental Design Conflicts

- Semantics promotes reuse. productivity of reuse
- Semantics compromises reuse. range of reuse (economies of scale)

- Semantics promotes integration. efficiency of communication
- Semantics compromises integration. openness of communication

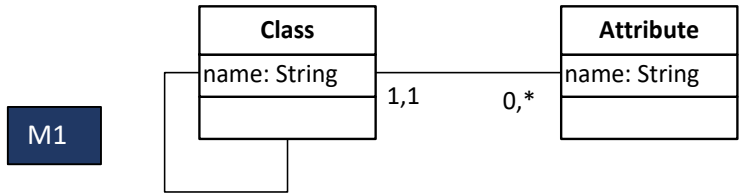
- Semantics promotes flexibility. through abstraction
- Semantics compromises flexibility. „loose coupling“

Problem: Synchronization of Model and Code

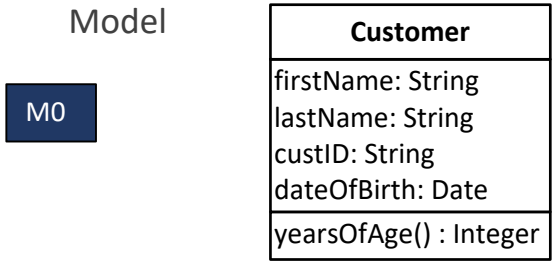
Modelling Environment

Progamming Environment

Meta-Model



Model



generate

Code

```
class Customer
{
    String firstName;
    String lastName;
    Date dateOfBirth;

    public int yearsOfAge()
           ....
}
```

M1

Program instance

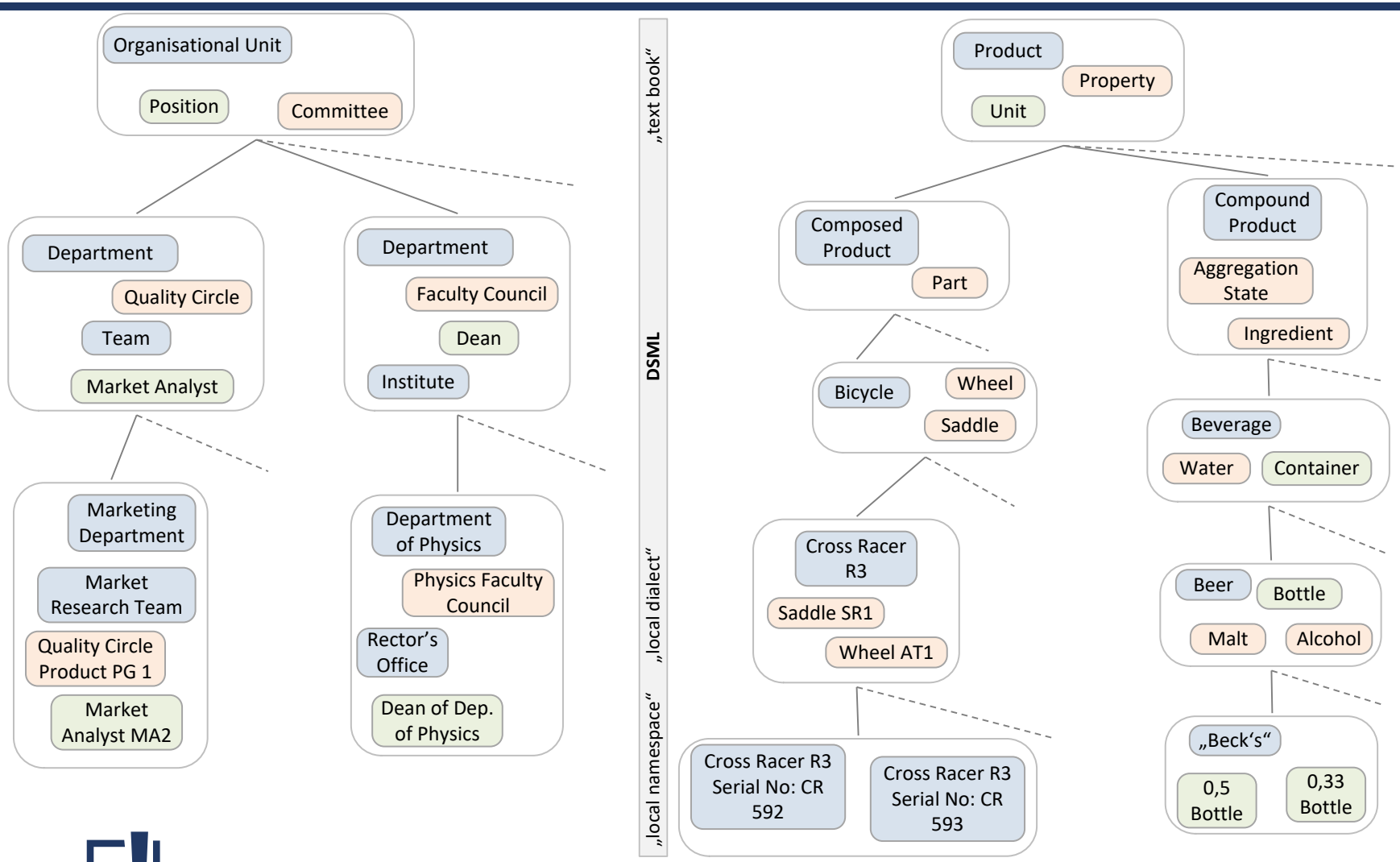
M0

Why is there need to generate code anyway?

Multi-Level Modeling to the Rescue

- new language paradigm
- allows for an arbitrary number of classification levels
- ... and deferred instantiation
- Every class at any level is an object.
- first introduced in 2001 by Atkinson and Kühne
- with roots going back to the early 90s
- focus mainly on modelling, not on programming languages

Inspired by Actual Use of Technical Languages



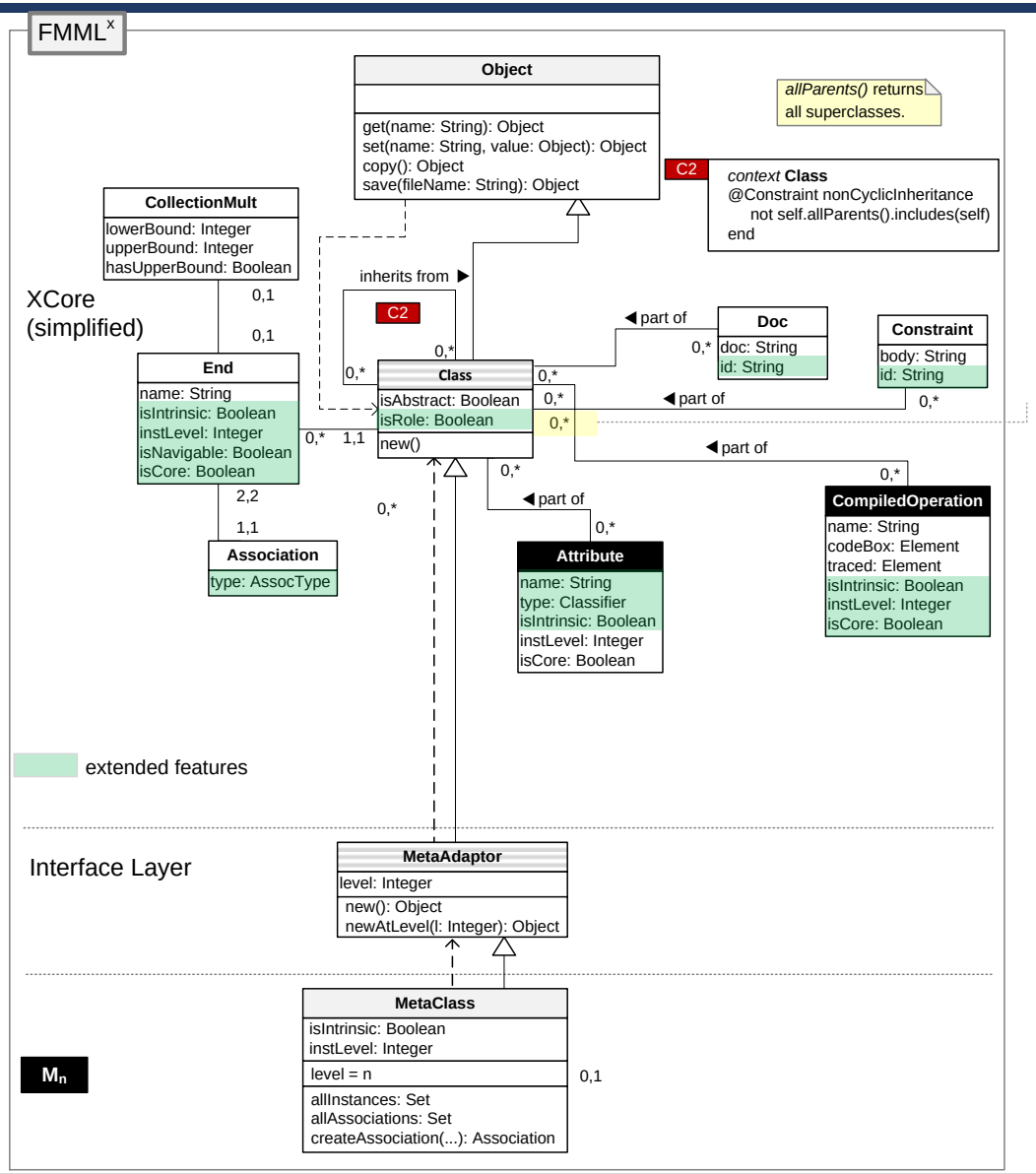
Language hierarchies with variable number of levels

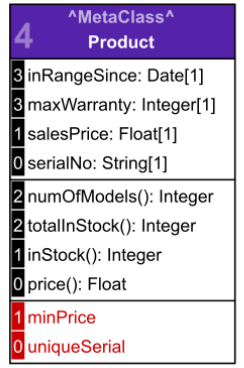
Foundations

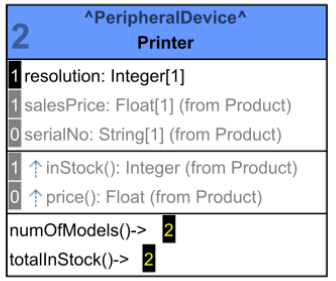
- XModeler
  - language engineering workbench
  - core language (Xcore) both reflexive and extensible
  - allows for an arbitrary number of (implicit) classification levels, but does not provide explicit levels and deferred instantiation
- both an instance of itself and a basis for defining a wide range of co-existing language variants – like the FMML^{x}
- common representation of programs and models
- models at any level executable
- [illegible]cs

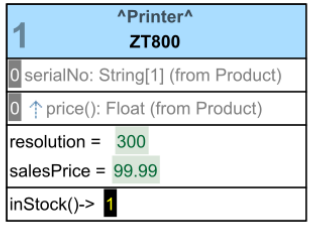
FMML^{x}	Multi-level modeling and execution language
XModeler^{ML}	Multi-level engineering, modeling, programming and execution environment, enables creation and use of graphical and textual DSL

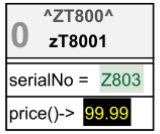
FMML^x: Metamodel & Generic Notation



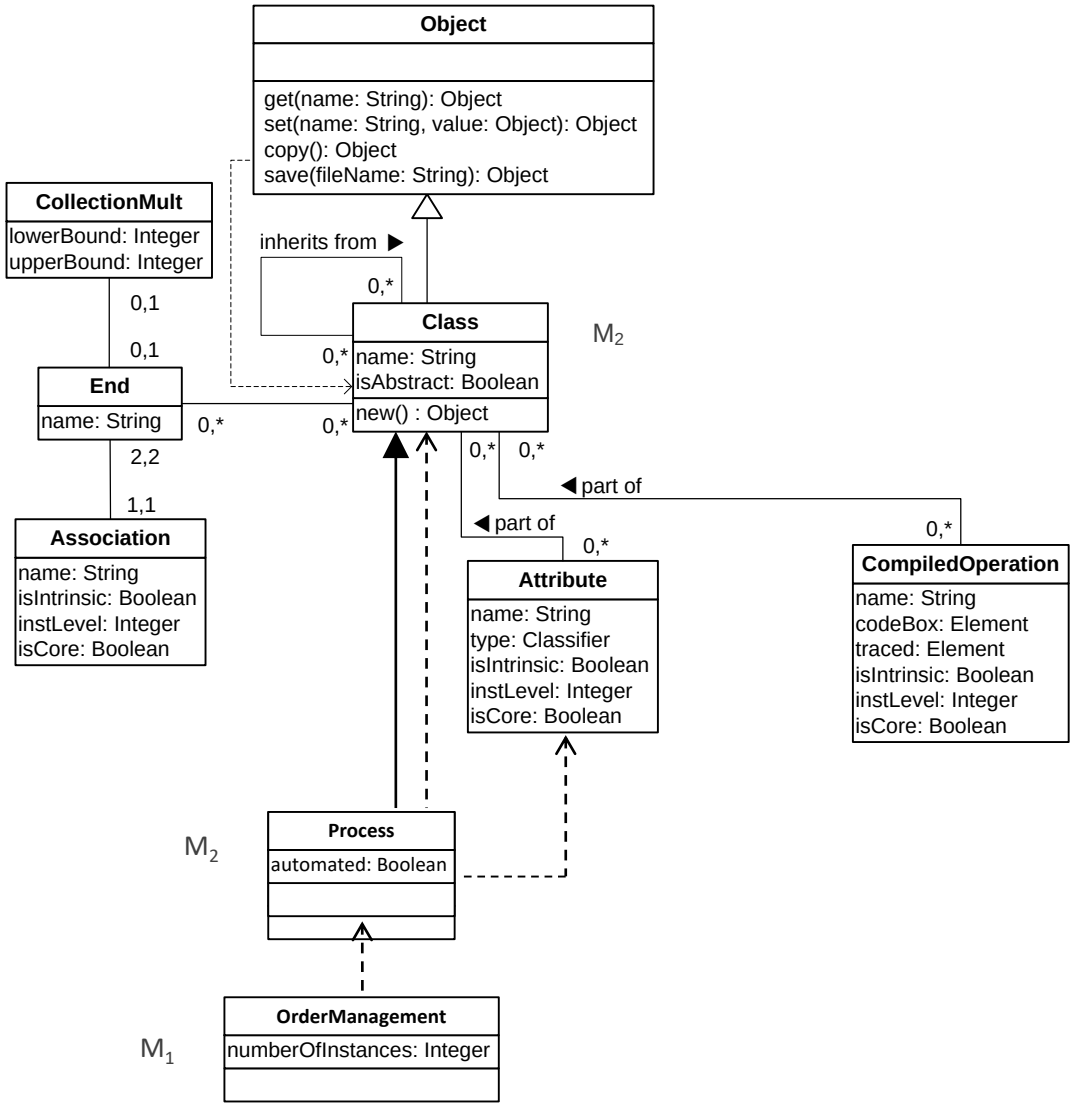






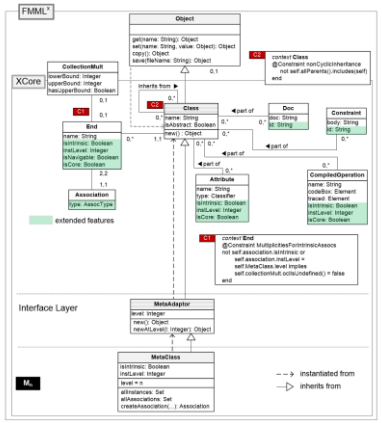


Background: Raising the Level of Classification



Architecture: Ballpark View

Every class is an object!



Meta Model (Meta-Modeling/Programming Language)

Representations

Text

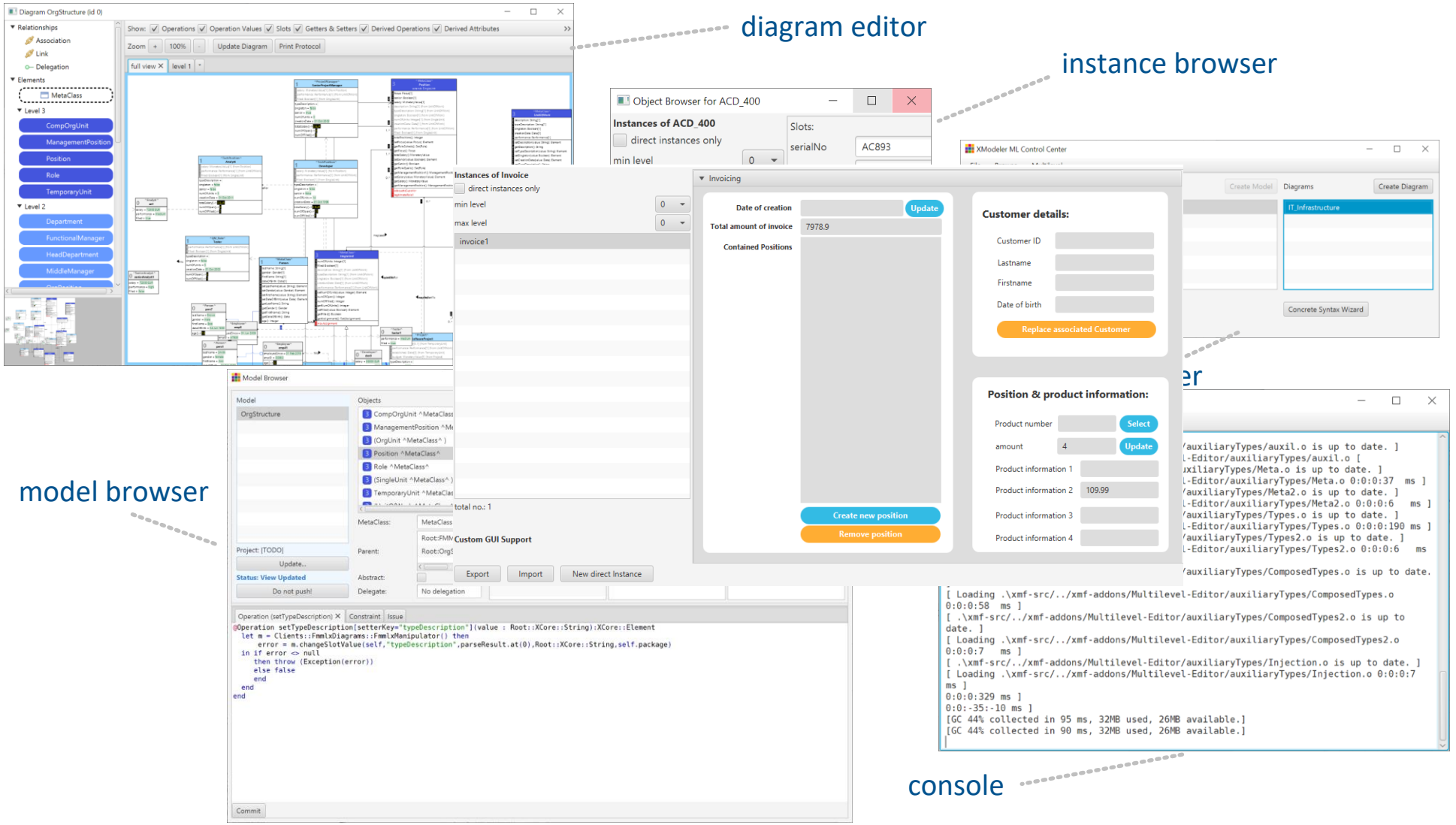
GUIs ..

Objects at any Level (Languages, executable models, programs)

Diagram

Code

XModeler^{ML}: Components of the Environment



Conclusions

- FMML^{x} allows for
  - conjoint design of models and modeling languages
  - at an arbitrary number of classification levels
- thus
  - enabling the specification of DSLs with more generic DSLs
  - hence, contributing to productivity, flexibility and integrity
  - relaxing fundamental design conflicts
- XModeler^{ML} enables
  - common representation of models and code
  - execution of models at any level
  - navigation from GUI to models (diagram or textual) at runtime
  - self-referential architectures of applications
  - contributes to user empowerment

Future Research

- Further development of multi-level modeling method
- Maintenance/management of multi-level models
- New version of editor for defining graphical notations
- Multi-level modeling of processes (dynamic abstractions)
- Reconstruction of existing DSMLs for enterprise modelling
- Prototypical implementation of self-referential ERP system

Download of Tool, Models, Screencasts ..

https://www.wi-inf.uni-duisburg-essen.de/LE4MM/