Multi-Level Language Architectures

A Foundation for Advanced Enterprise Systems

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

Ulrich Frank

LE4MM

Overview

- Enterprise Systems: Recent Developments … and Lack of a Vision
- Objectives and Challenges
- SRES: Early Vision .. and Obstacles
- Multi-Level Language Architecture to the Rescue
- Design and Implementation with the FMML^{x} and the XModeler^{ML}
- Conclusions and Future Research

Enterprise Systems – Where to Go?

MDD

Low-Code

In-Memory DB

SaS

Microservice

AI

IoT

Lack of a groundbreaking, feasible vision

Facets of Business Information Systems

- tremendous **complexity**
  -
  -
- re
  -
  - fluctuation of employees in key positions jeopardizes IT
- ir
- u
- pressure to justify/reduce IT costs

nd its

> "I see the artificial split between organizational and technical issues as dangerous and unnecessary, and the frequent cultural chasm between business people and information technology professionals as the one factor that can block the effective use of computers and communications."
>
> Peter G.W. Keen

> "Could it be that we are putting fifth generation technology in second generation organizations?"
>
> Charles M. Savage

BIS as linguistic artefacts!

What We Need

Problem	Measure
Complexity/Risk	Create transparency -> abstraction from irrelevant aspects
Communication	Focus on concepts different stakeholders are familiar with -> abstraction toward common concepts
Flexibility	Accounting for possible and probable change -> abstraction from existing structures, processes, solutions
Economics of IS development: Fostering reuse, integration and protection of investment	Identification of commonalities shared by different use contexts-> ■ abstraction from specific properties of particular domains ■ abstraction toward common properties (data structures, functions …) ■ abstraction from properties that may change over time, especially from particular technologies

Abstraction is the name of the game!

Problem	Measure
Complexity/Risk	Create transparency -> abstraction from irrelevant aspects
Communication	Focus on concepts different stakeholders are familiar with -> abstraction toward common concepts
Flexibility	Accounting for possible and probable change -> abstraction from existing structures, processes, solutions
Economics of IS development: Fostering reuse, integration and protection of investment	Identification of commonalities shared by different use contexts-> ■ abstraction from specific properties of particular domains ■ abstraction toward common properties (data structures, functions …) ■ abstraction from properties that may change over time, especially from

Need for conceptual models!!

Enterprise Model

An enterprise model integrates at least one conceptual model of the information system (e.g. a class diagram) with at least one model of the relevant action system (e.g. a business process model).

Frank, Ulrich, Strecker, Stefan, Fettke, Peter, Brocke, Jan vom, Becker, Jörg, and Sinz, Elmar J. “The Research Field “Modeling Business Information Systems".” *Business&Information Systems Engineering* 6, No. 1 (2014): 39–43.

CIM/OSA

SOM

Zachman Framework

TOGAF

ARIS

MEMO

4EM

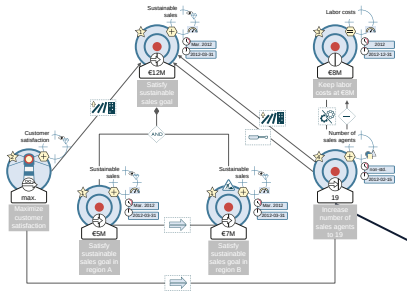
Multi-Perspective Enterprise Modelling (MEMO)

A multi-perspective enterprise model differentiates between various explicit **perspectives**, which will usually correspond to professional views. These perspectives are represented in models constructed with domain/purpose specific modelling languages.

- developed during the last 30 years
- accompanied by various tools
- still subject of ongoing research

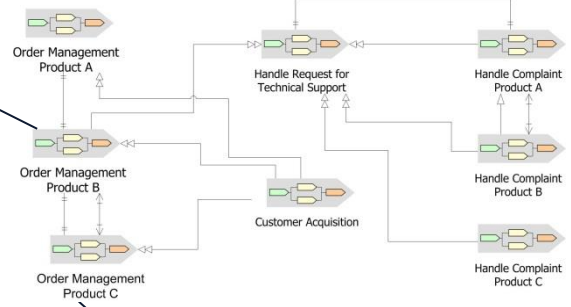
Frank, Ulrich. “Multi-Perspective Enterprise Modeling: Foundational Concepts, Prospects and Future Research Challenges.” *Software and Systems Modeling. Vol. 13, 2014, pp. 941-962*

MEMO: Illustration

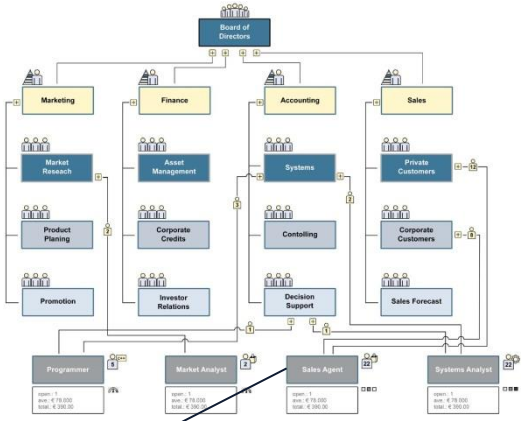


Goal Diagram

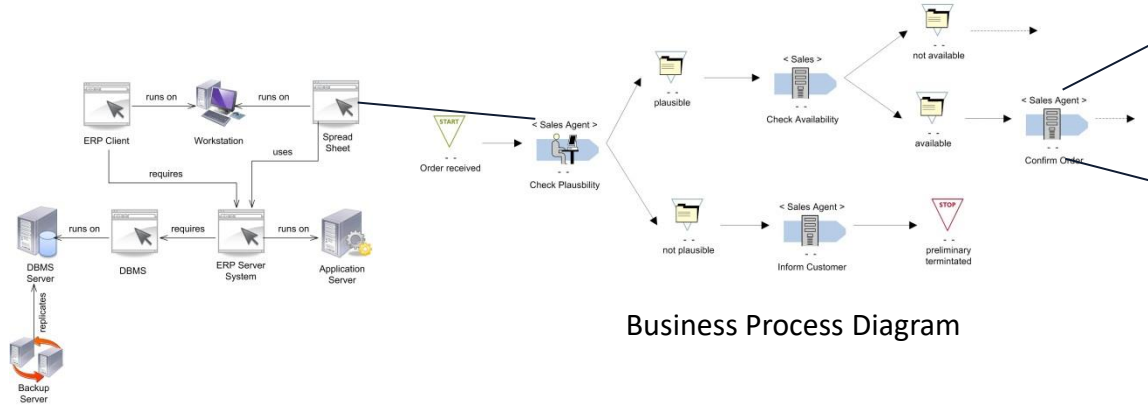
Value Chain Diagram



Business Process Map

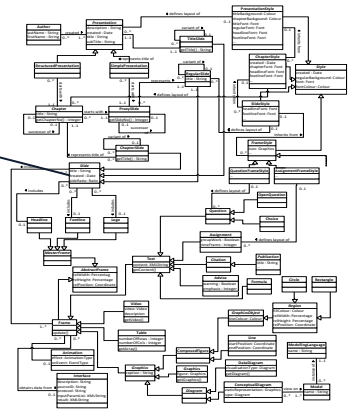


Organisational Chart



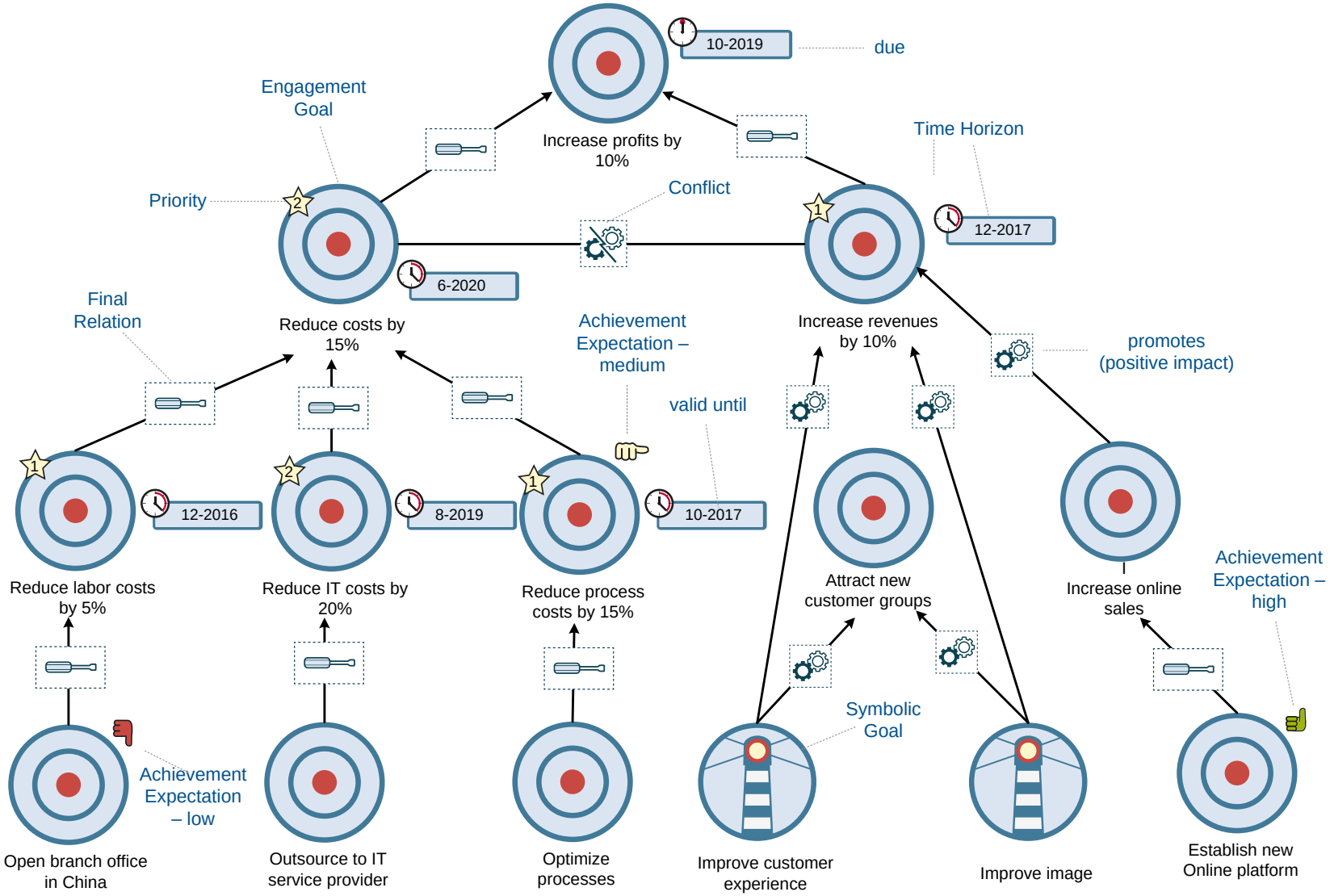
IT Resource Diagram

Business Process Diagram

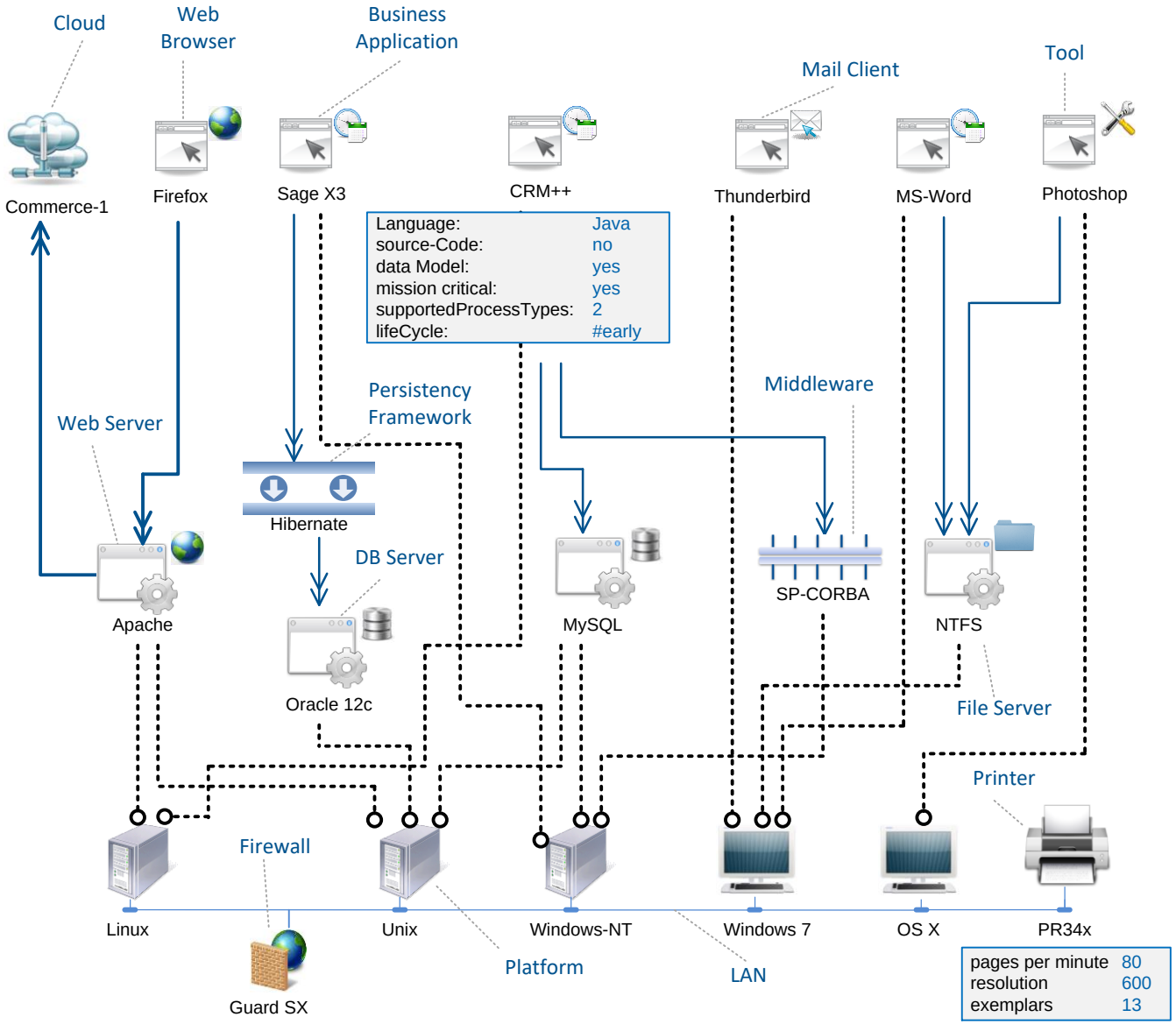


Object Model

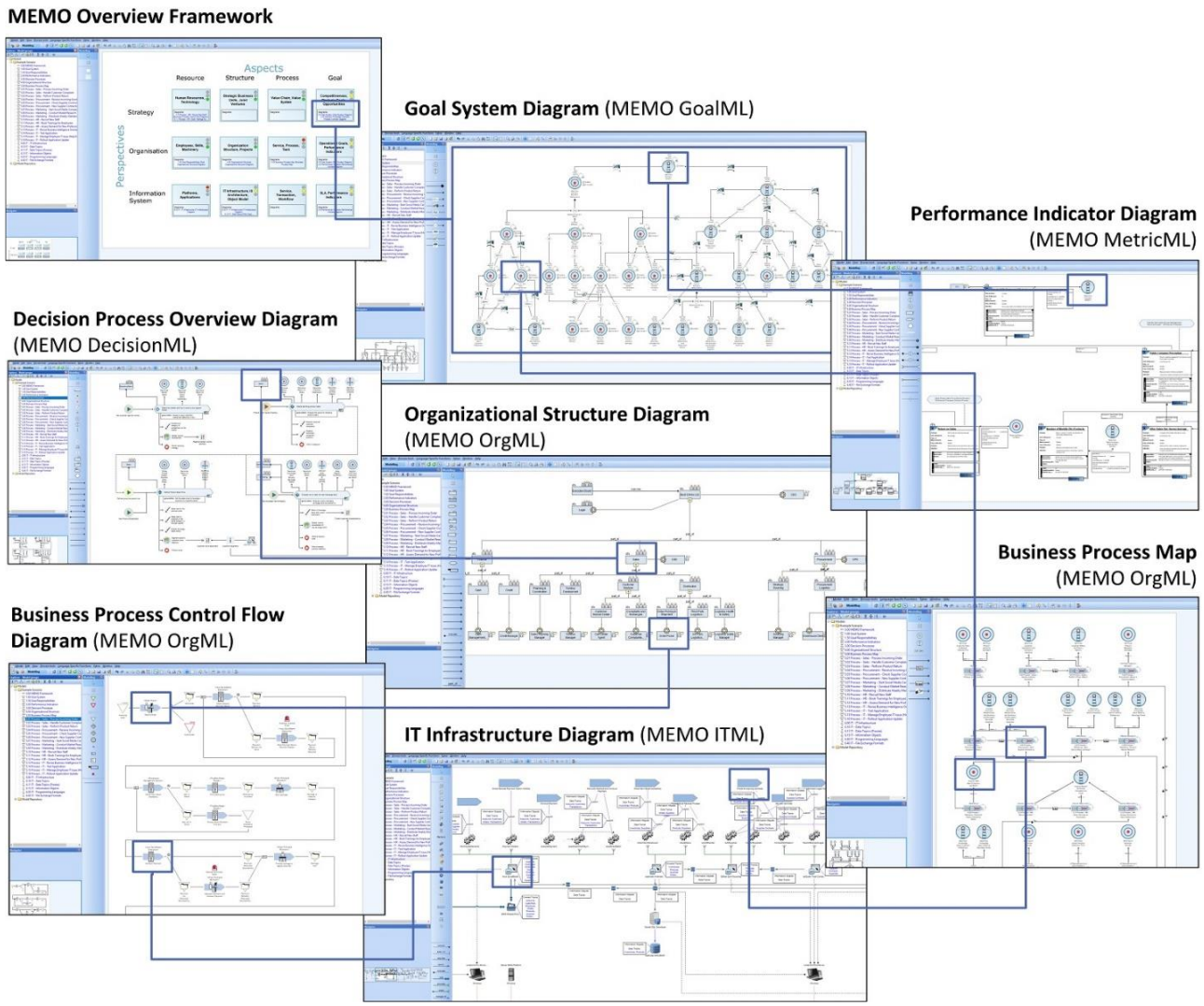
GoalML: Example



ITML: Example Diagram



Corresponding Tool: MEMO4ADO



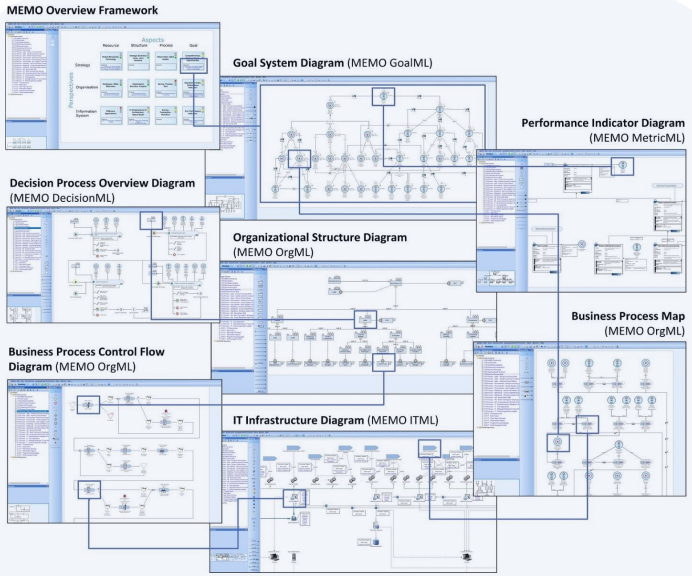
Vision: Self-Referential Enterprise System (SRES)

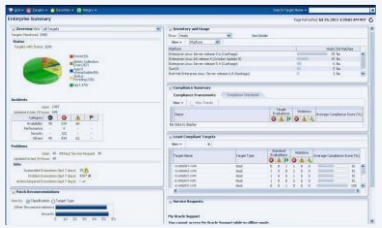
Development

Deployment

Use

Maintenance





DSMLs

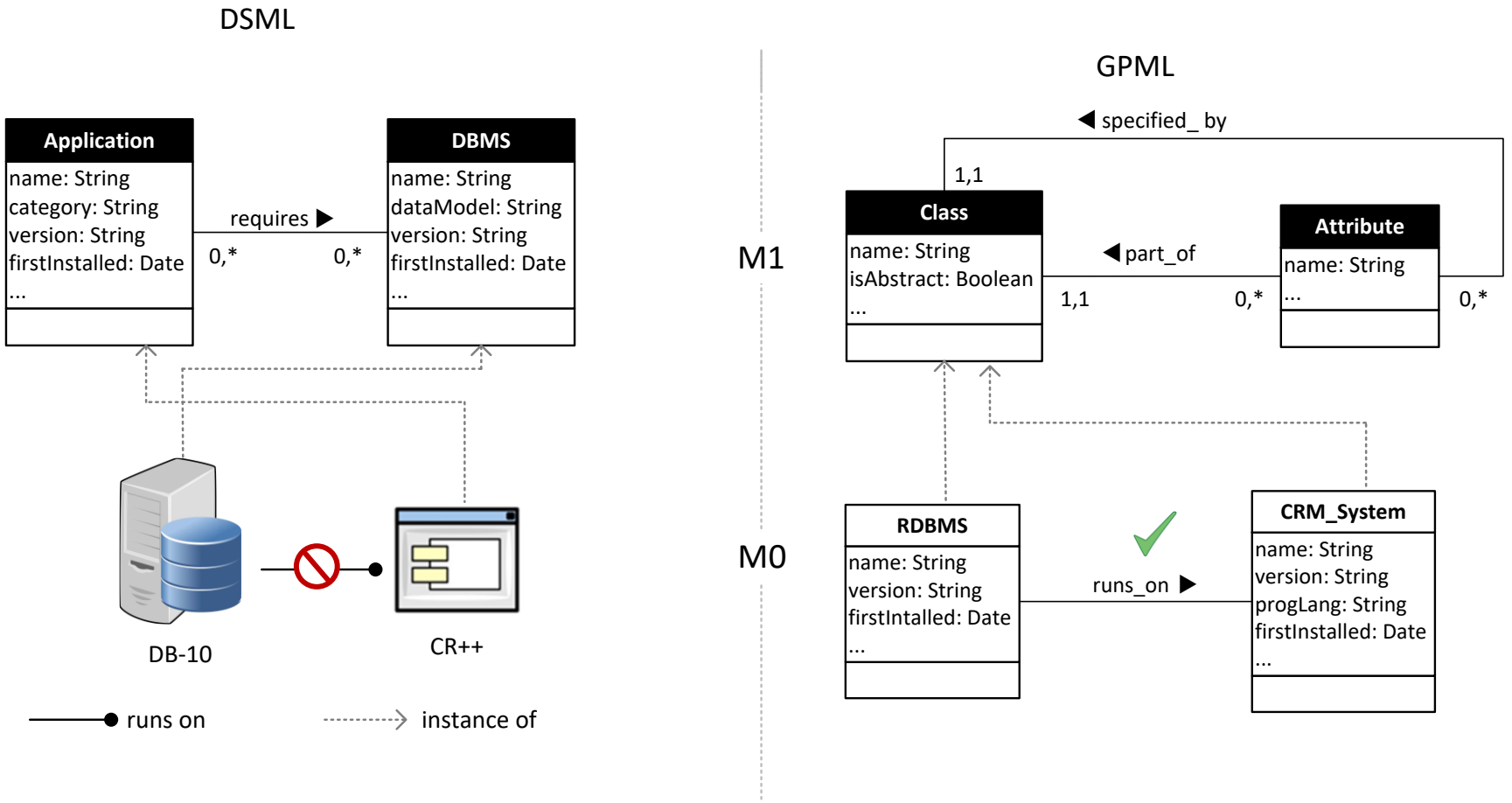
Enterprise Model

Enterprise Software

Focus on enterprise model during entire lifecycle.

Tight integration of modeling tool and enterprise software.

Domain Specific Language (DSML)



Model of IT infrastructure created with DSML

Model of IT infrastructure created with GPML

SRES would foster …

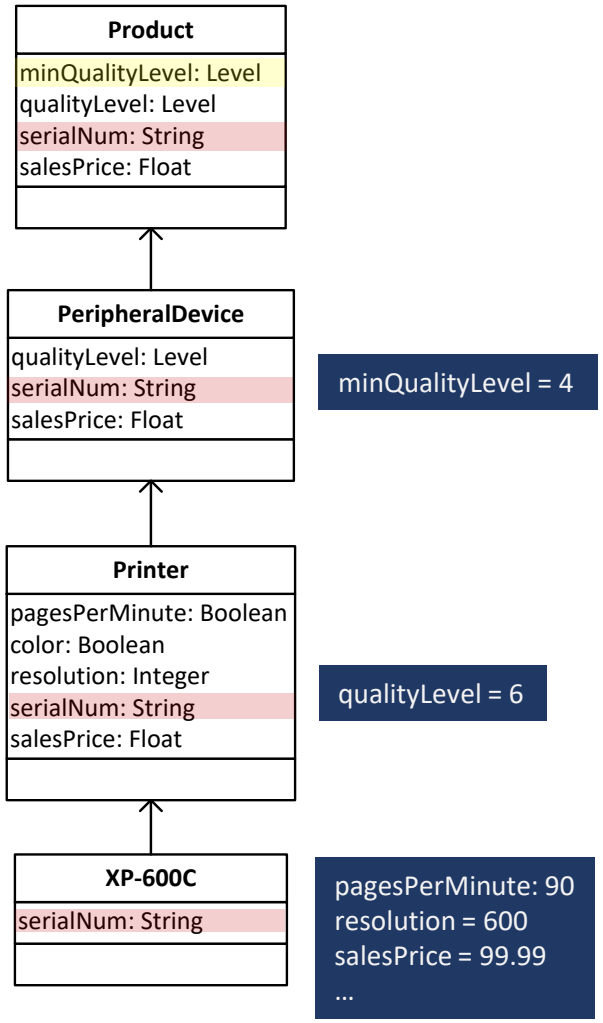
- conjoint analysis and design of IS and organization
- transparency by shedding light into the black box
- additional user empowerment
  - improved understanding of potential and limitations of IS
  - adaptation by authorized users of models they are familiar with
- reuse by abstracting on commonalities promoted by DSMLs
- integration
- decision making

Nice vision – but …

Serious Obstacles

- DSMLs unsatisfactory
  - substantial development effort
  - defined with general-purpose meta languages
  - lack of expressiveness
- tight integration of models and code hardly possible
  - mainstream programming languages require two different representations
  - synchronization painful and, in the long run, not effective
- fundamental design conflicts
  - ambivalent effects of semantics
  - e.g., trade-off between range and utility of reuse („generality/power“)

Problem 1: Lack of Expressiveness



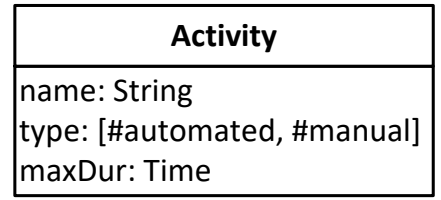
Sometime no clear distinction between instantiation or specialisation – both make sense.

Every (meta) class is an object (has state).

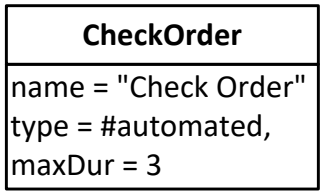
There are multiple levels of classification.

Instantiation may be deferred to lower levels.

No satisfactory solution within current paradigm!



M2



M1

actual duration at M0?

Problem 2: Integration of Model & Code

- model-driven development
- based on convincing idea
- at first restricted to build time
- even better: models available during the entire lifetime of a system
- in ideal case maintenance is restricted to models
  - however, usually not feasible
  - therefore: need for **synchronisation** of models and code

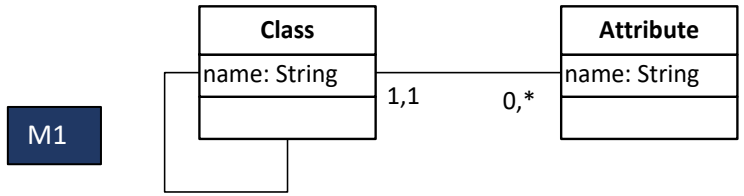
separate representations of model and program often show stopper

Problem: From Model to Code

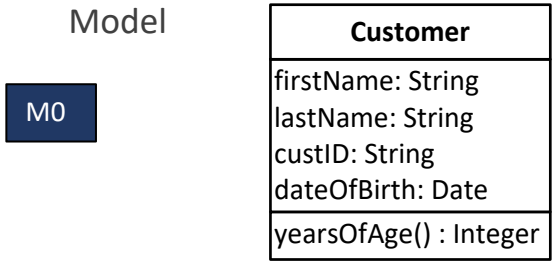
Modeling Environment

Progamming Environment

Meta-Model



Model



generate

Code

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

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

M1

```
if actKunde.alter() >= fullAge ...
```

M0

Why is there need to generate code anyway?

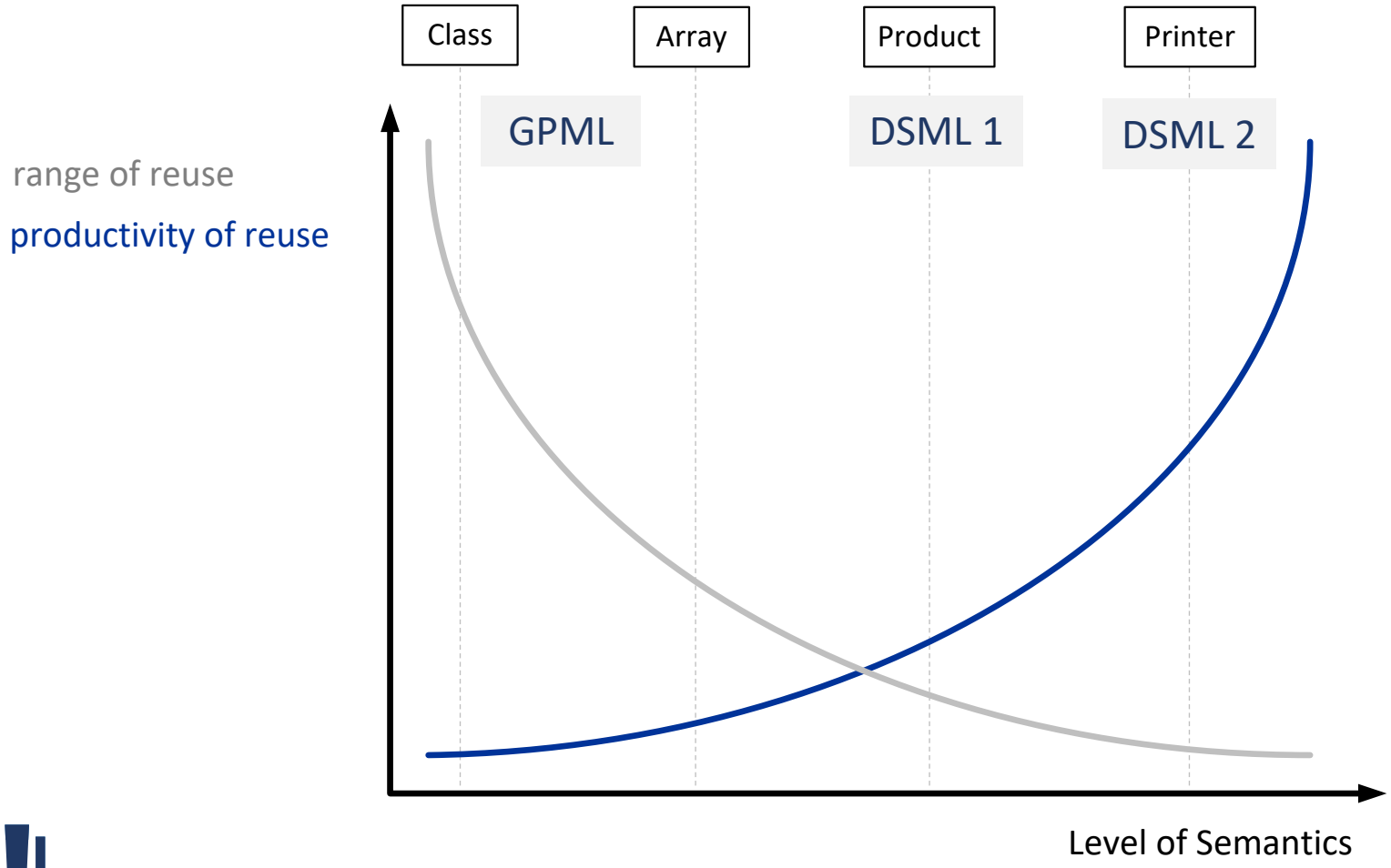
Problem 3: Fundamental Design Conflicts

- 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“

Range and Productivity of Reuse

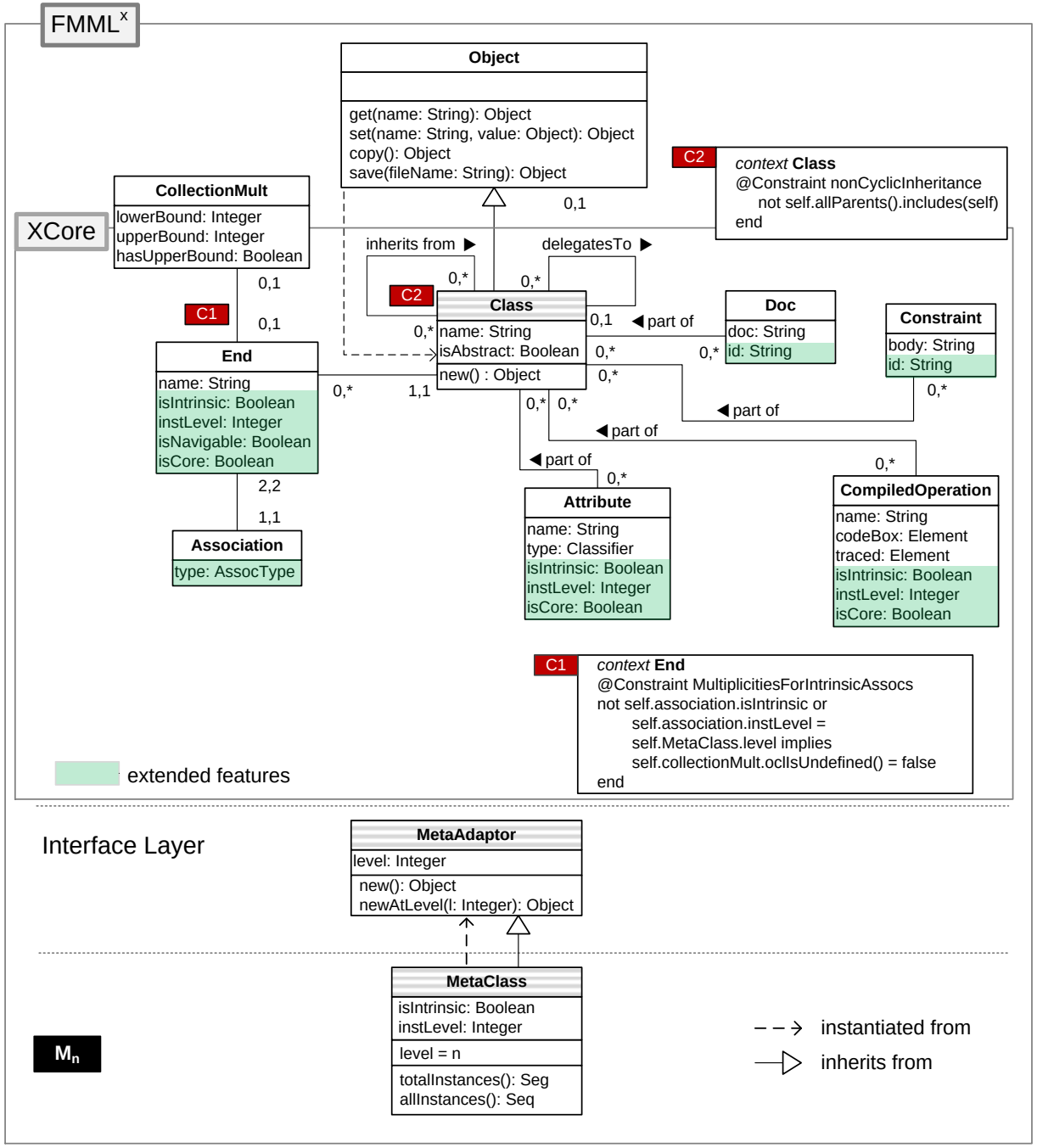


Need for (painful) trade-off between economies of scale and utility of reused artefact in particular case.

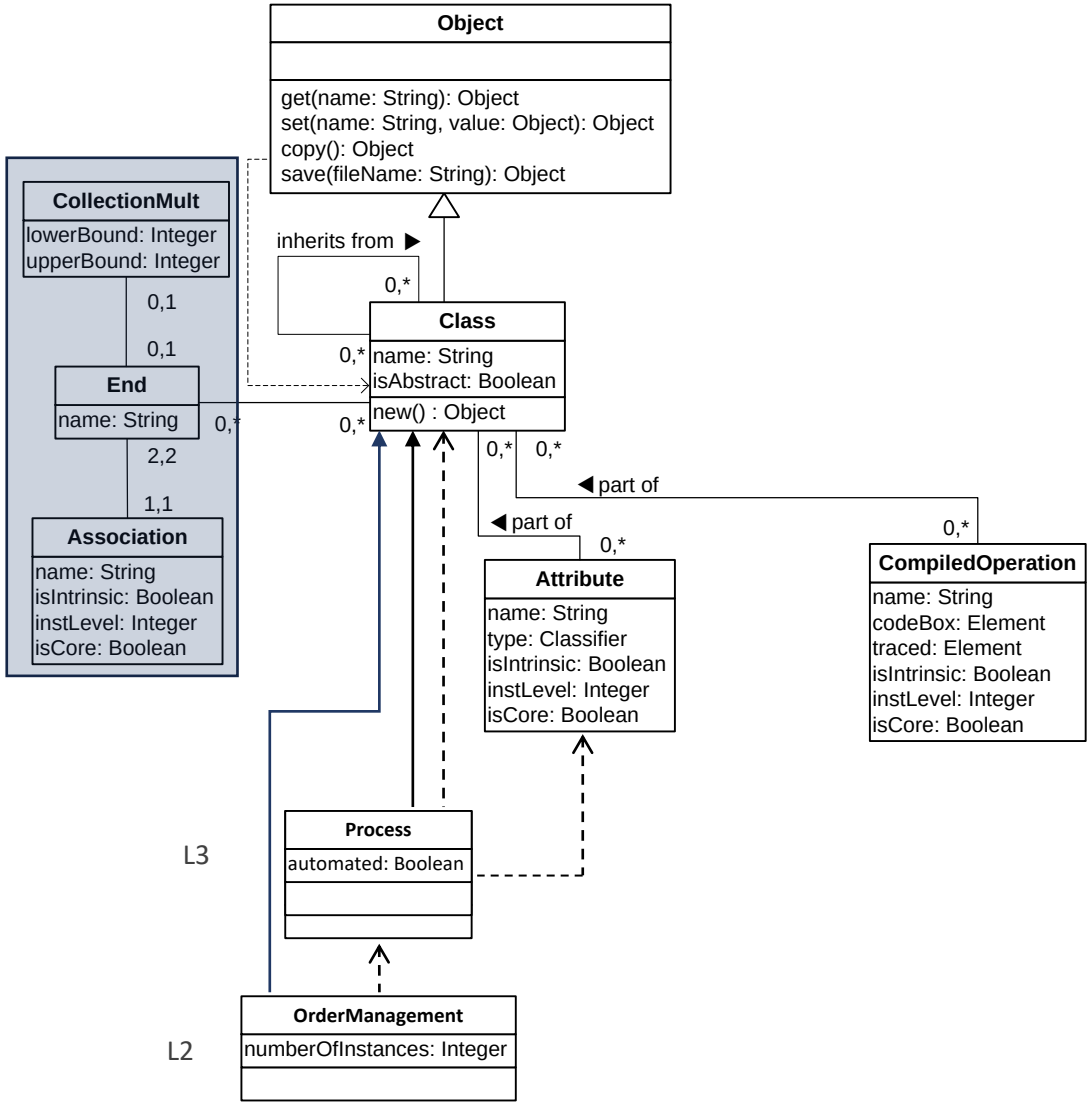
Multi-Level Language Architecture to the Rescue

- new language paradigm
- allows for an arbitrary number of classification levels
- motivated by the lack of abstraction in traditional, MOF-like language architectures
  - creates avoidable complexity
  - contributes to lack of flexibility
- first introduced in 2001 by Atkinson and Kühne
- with roots going back to the early 90s
- various approaches developed since then
- focus mainly on modelling, not on programming languages

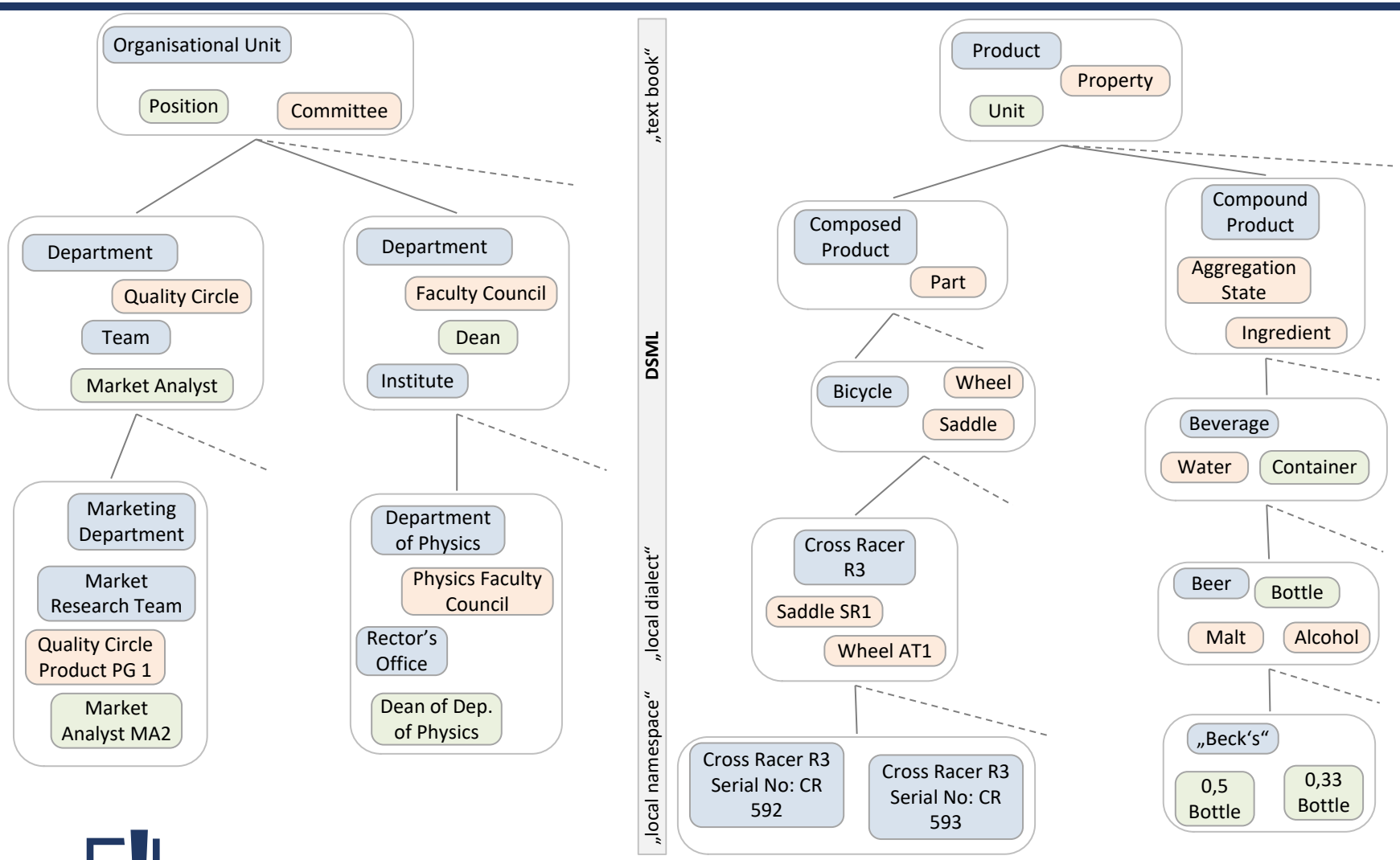
FMML^x: Metamodel



Background: Raising the Level of Classification



Inspired by Actual Use of Technical Languages



Language hierarchies with variable number of levels

FMML^{x} and XModeler^{ML}

- XCore and the XModeler
  - language engineering and execution environment
  - facilitates specification, implementation and execution of programming and modeling languages
  - developed by Tony Clark and colleagues
  - used and further developed in various projects with large companies
- FMML^{x}
  - executable multi-level modeling language
  - based on extension of XCore
  - common representation of models and code
- XModeler^{ML} implements FMML^{x}, extends orginal XModeler

Frank, Ulrich (2022): Multi-level modeling: cornerstones of a rationale. In: *Software and Systems Modeling* 21, pp. 451–480.

FMML^x: Illustration

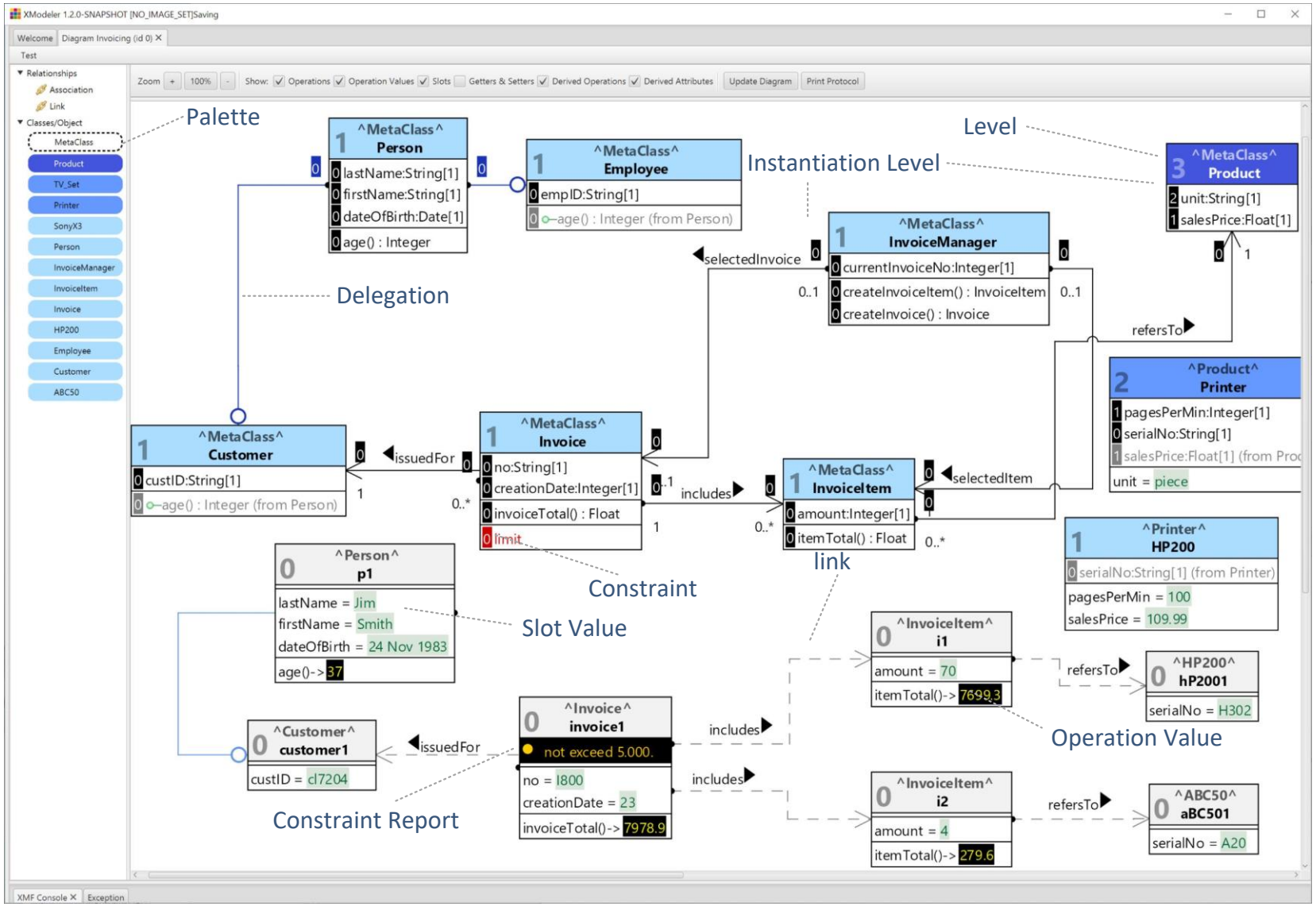
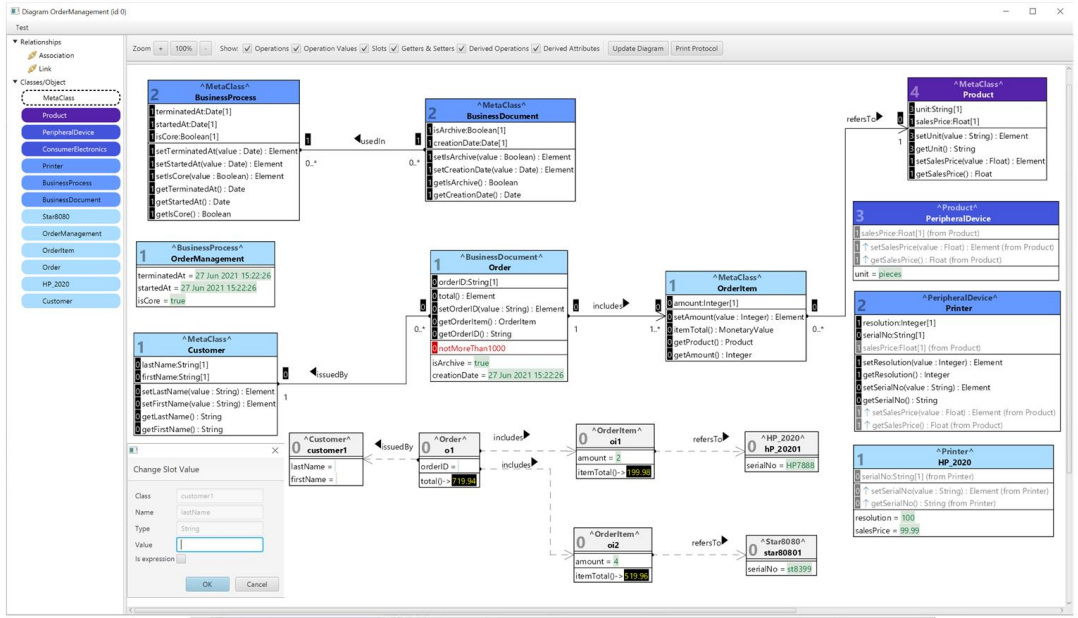
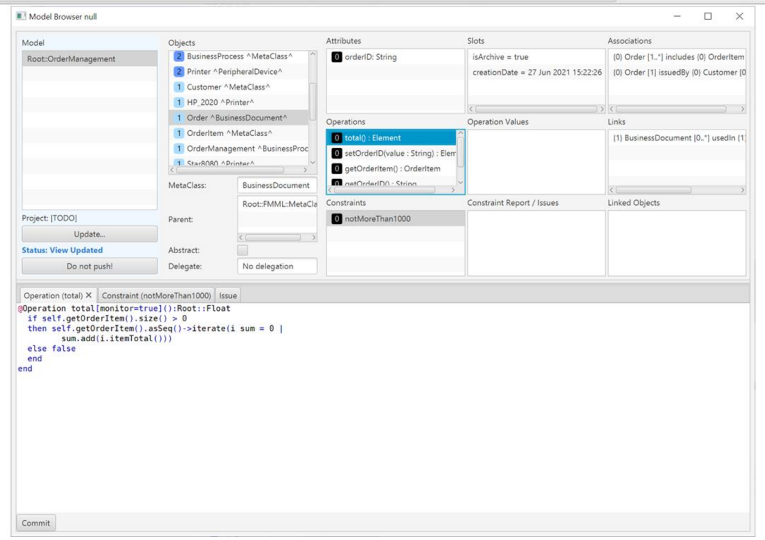


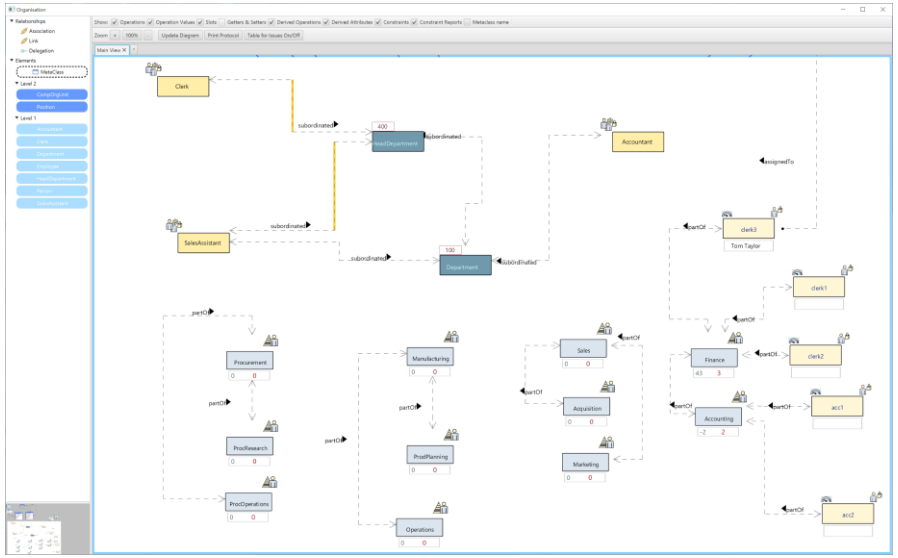
Diagram or Code – Different Views on Same Model



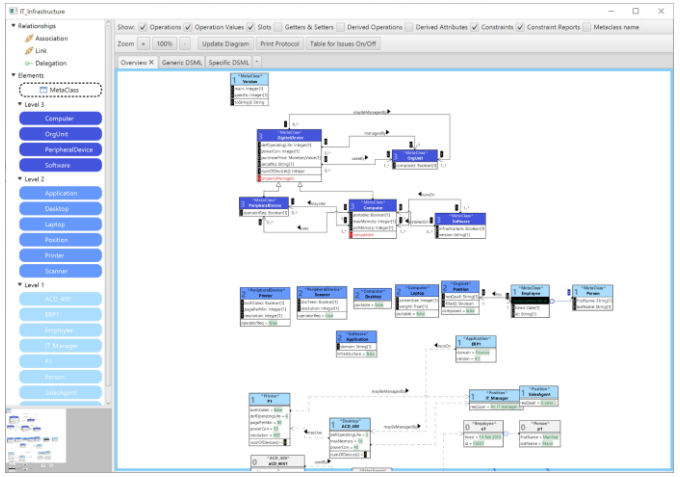


One Model – Many Representations

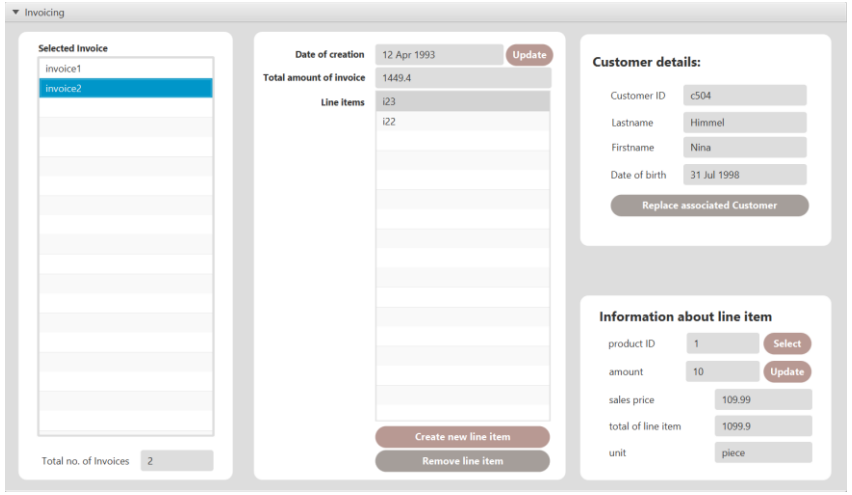
DSML with Graphical Notation



Default Notation



Generated/Revised GUI



Multi-Level SRES in a Nutshell

L2+

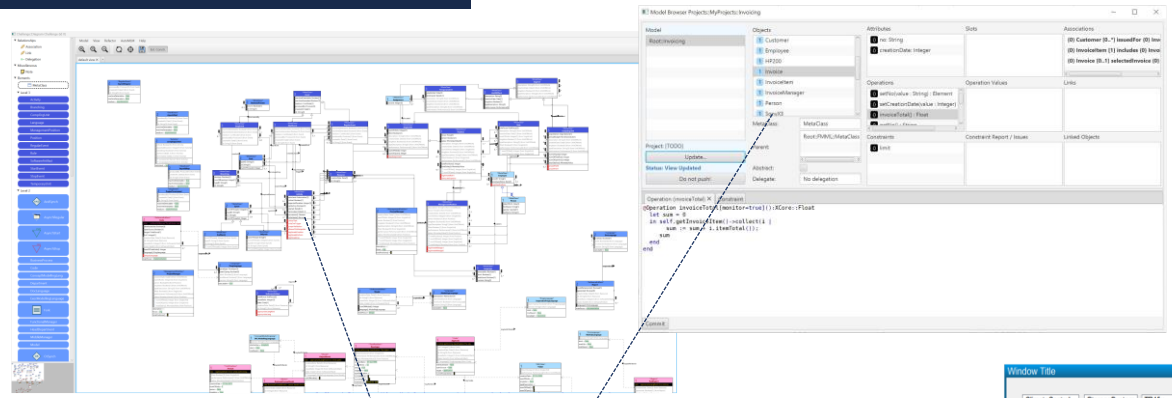
Multi-level model of the SRES presented as diagram or in a browser

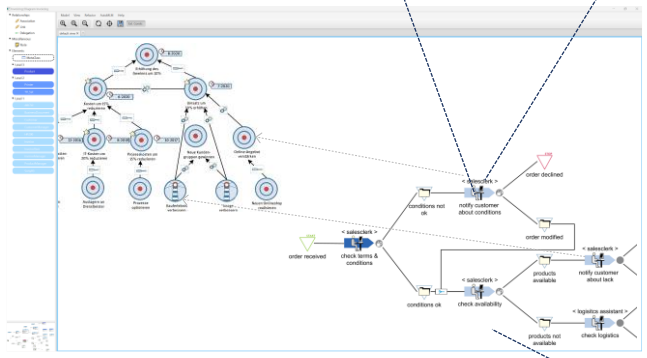
L1

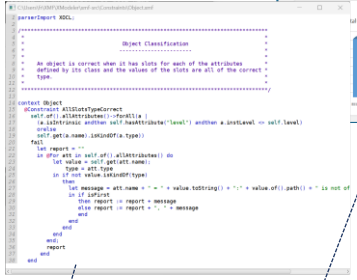
Graphical Editors, specific GUIs or text editors to access models of the enterprise and of the enterprise software

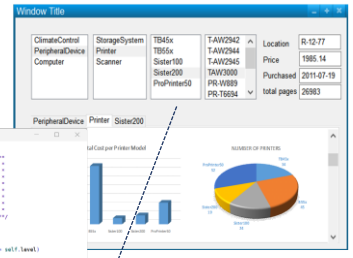
L0

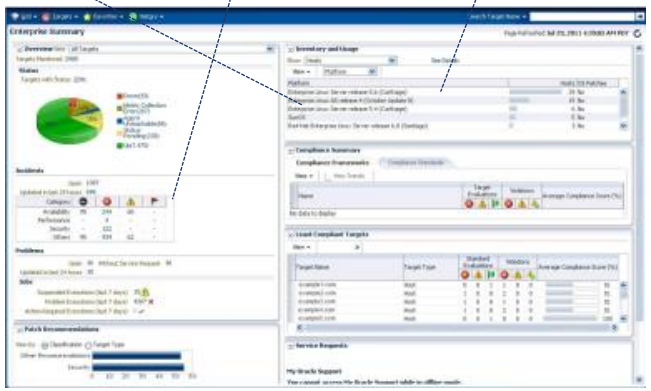
Traditional GUI of Enterprise System











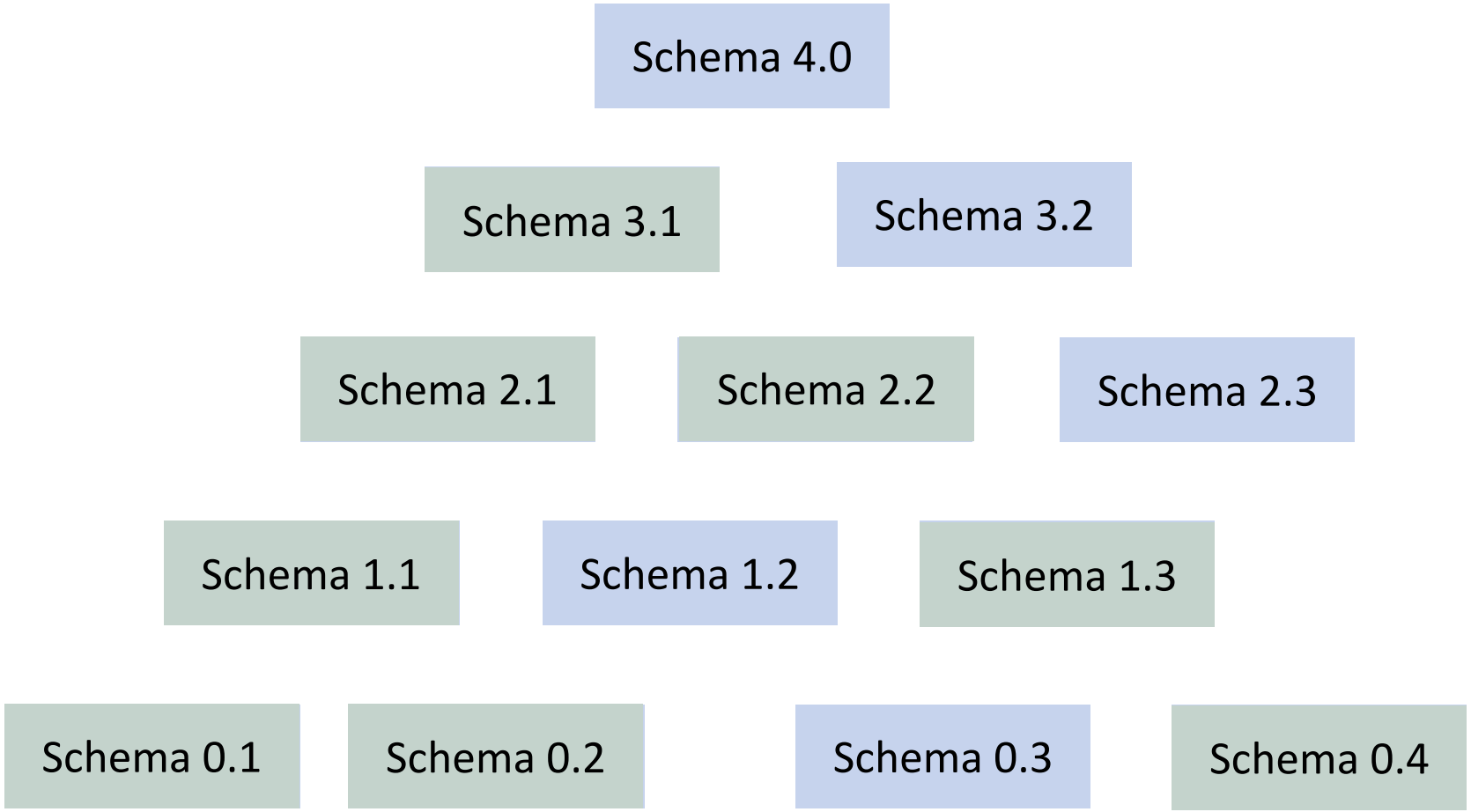
change of models at any level – immediate effect on software

Tight coupling requires specific refactoring measures in cases of non-monotonic extensions

also: need to cope with restrictions of dynamic typing

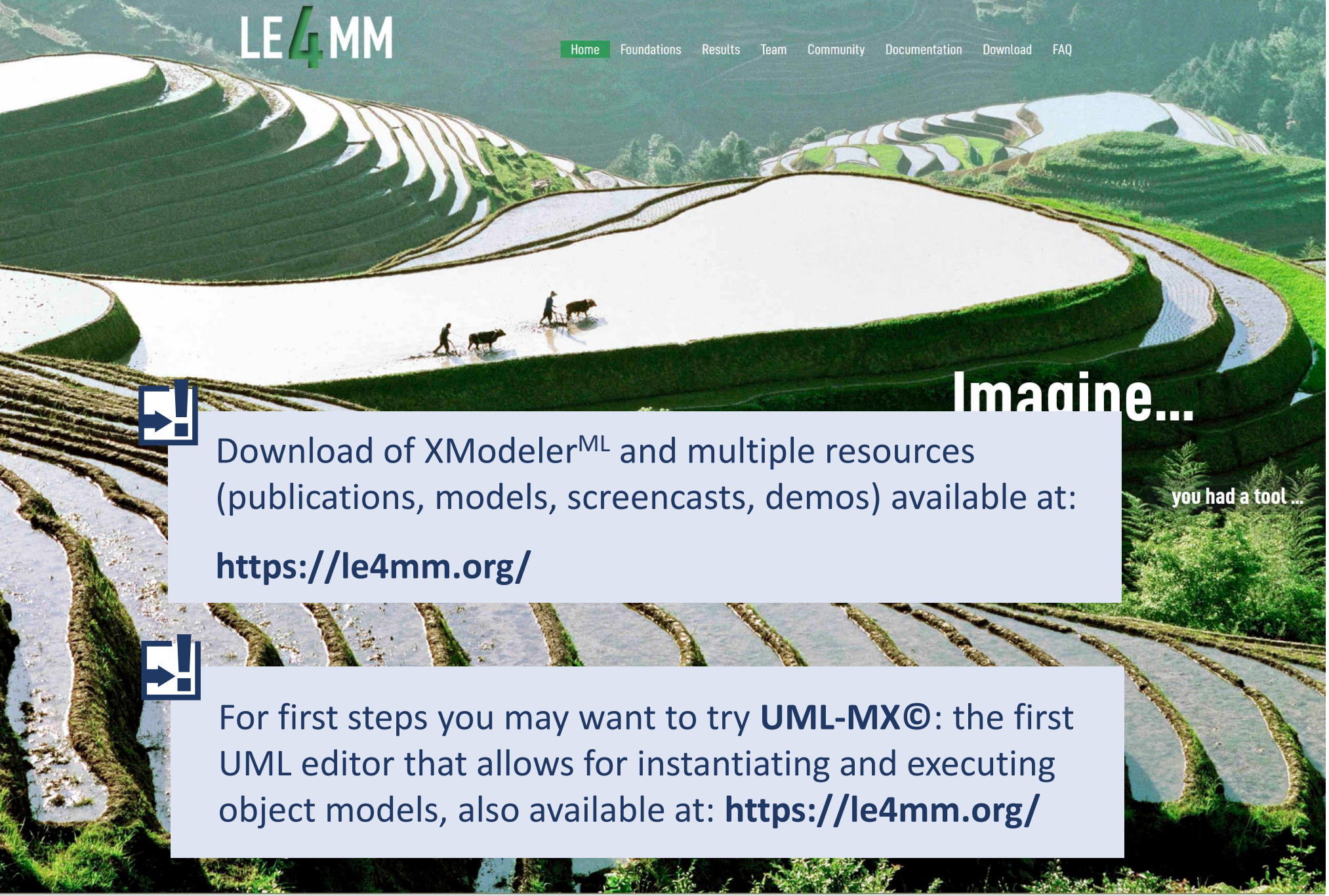
Relaxing the Generality/Power Conflict

Change of system through using a higher level (meta) schema (DSML)
and, may be, create a new instance of that -> clearly less effort and risk.



Conclusions and Future Research

- Multi-Level Language Architectures: Promising new Paradigm
  - suited to increase development productivity ..
  - and user empowerment
  - relaxes fundamental design conflict
  - enables advanced enterprise systems
  - (cross-) organizational integration
- awareness slowly increasing
  - in practice -> project with Oracle
  - in academia -> award for “most influential project” at CAiSE 2023
  - various promising application areas, e.g., digital twins, IoT, and many more
- Future work
  - multi-level process modelling – very challenging!
  - multi-level refactoring
  - multi-level reengineering of flat models and schemas
  - …



Download of XModeler^{ML} and multiple resources (publications, models, screencasts, demos) available at:

https://le4mm.org/

For first steps you may want to try **UML-MX©**: the first UML editor that allows for instantiating and executing object models, also available at: **https://le4mm.org/**