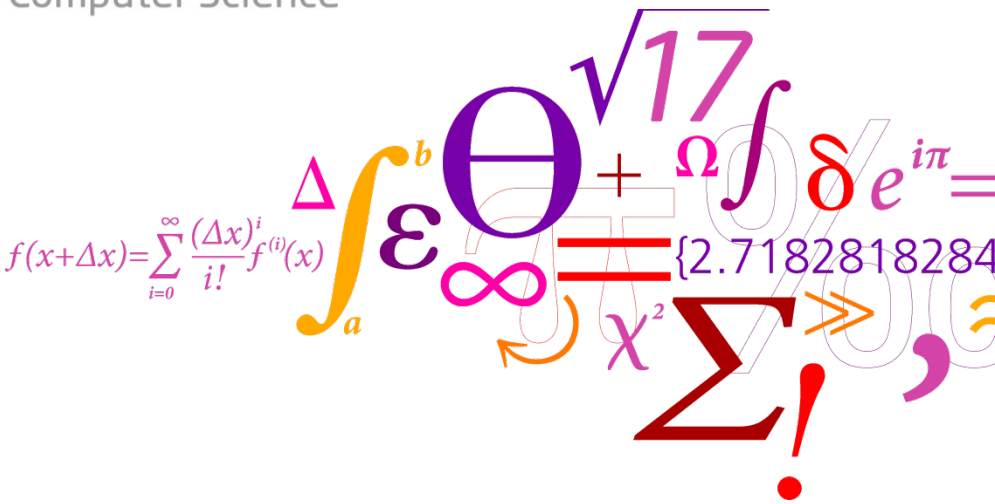
Model-based Software Engineering for/with Petri nets

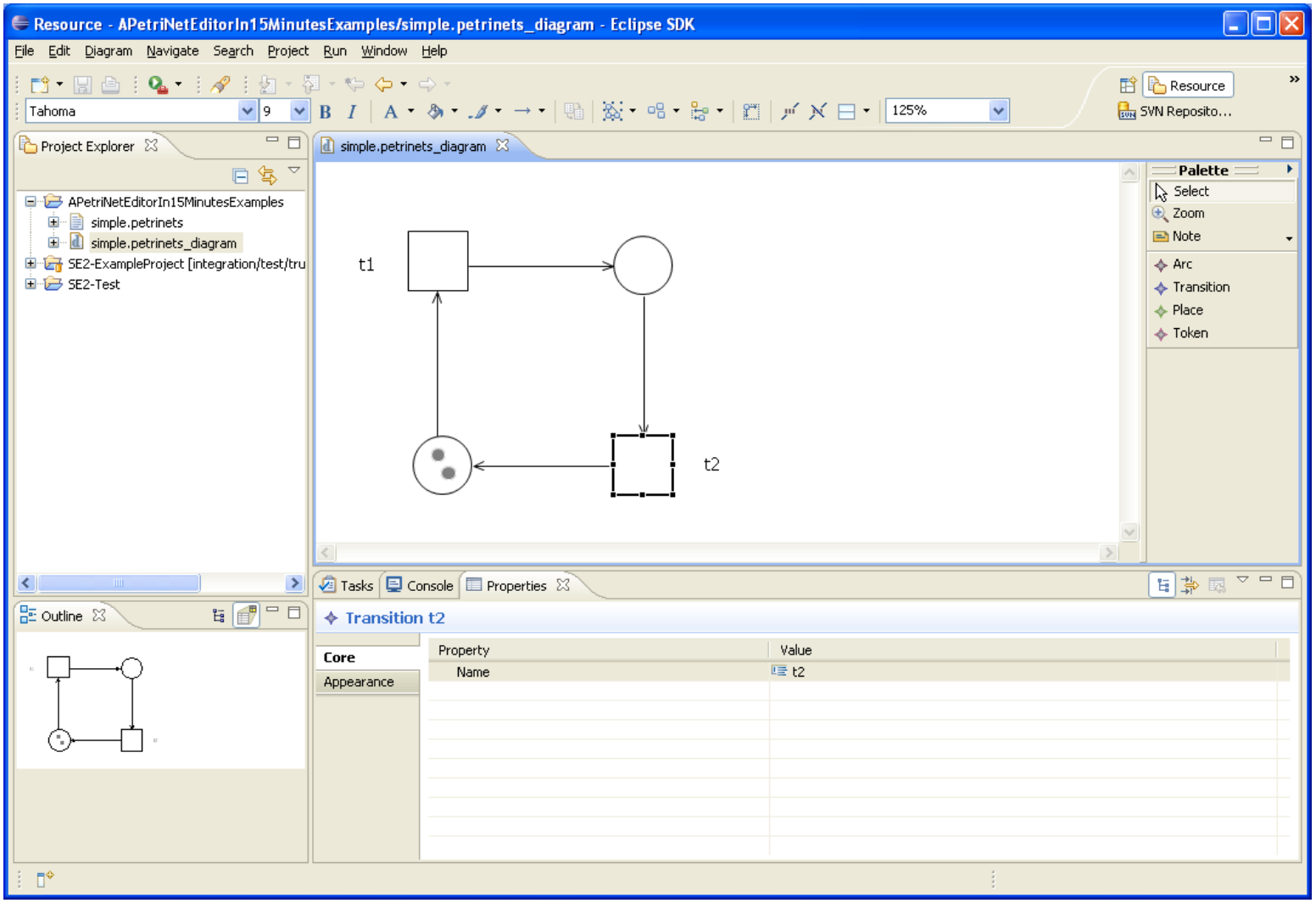
Ekkart Kindler

DTU Compute
Department of Applied Mathematics and Computer Science



1. Motivation

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Today

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

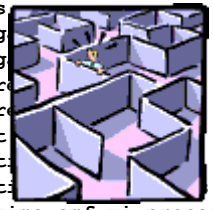
Technical artefacts!

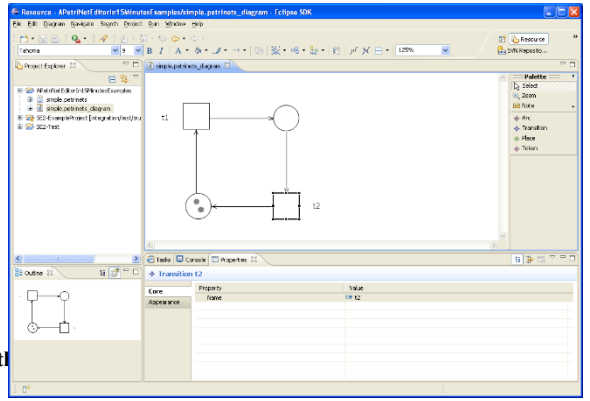
```
Manifest-Version: 1.0
Bundle-Name: %pluginName
Bundle-SymbolicName: APetriNetEditorIn15Minutes.diagr
Require-Bundl
 org.eclipse.
 org.eclipse.
 org.eclipse.
 org.eclipse.ui.ide,
 org.eclipse.ui.views
 org.eclipse.ui.navig
 org.eclipse.ui.navig
 org.eclipse.emf.ecor
 org.eclipse.emf.ecor
 org.eclipse.emf.edit
 org.eclipse.gmf.runti
 org.eclipse.gmf.runti                .core,
 org.eclipse.gmf.runtime.emf.ui.properties,
 org.eclipse.gmf.runtime.diagram.ui,
 org.eclipse.gmf.runtime.diagram.ui.properties,
 org.eclipse.gmf.runtime.diagram.ui.providers,
 org.eclipse.gmf.runtime.diagram.ui.providers.ide,
 org.eclipse.gmf.runtime.diagram.ui.render,
 org.eclipse.gmf.runtime.diagram.ui.resources.ed
 org.eclipse.gmf.runtime.diagram.ui.resources.e
 APetriNetEditorIn15Minutes;visibility:=reexpor
```

```
<plugin>

  <extension point="org.eclipse.emf.ecore.generated_pa
    <package
       uri = "PetriNets"
       class = "PetriNets.PetriNetsPackage"
       genModel = "model/PetriNet.genmodel" />
  </extension>

</plugin>
```

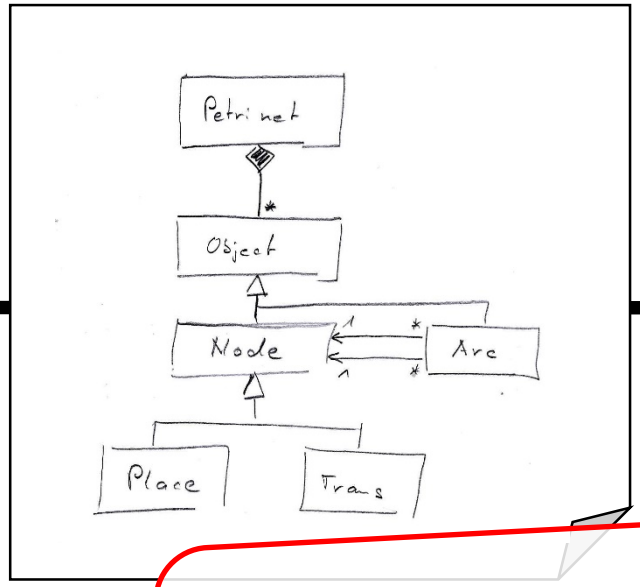


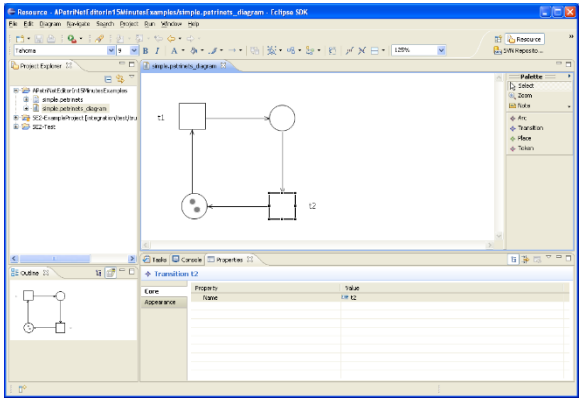


Get rid of them!

Vision

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



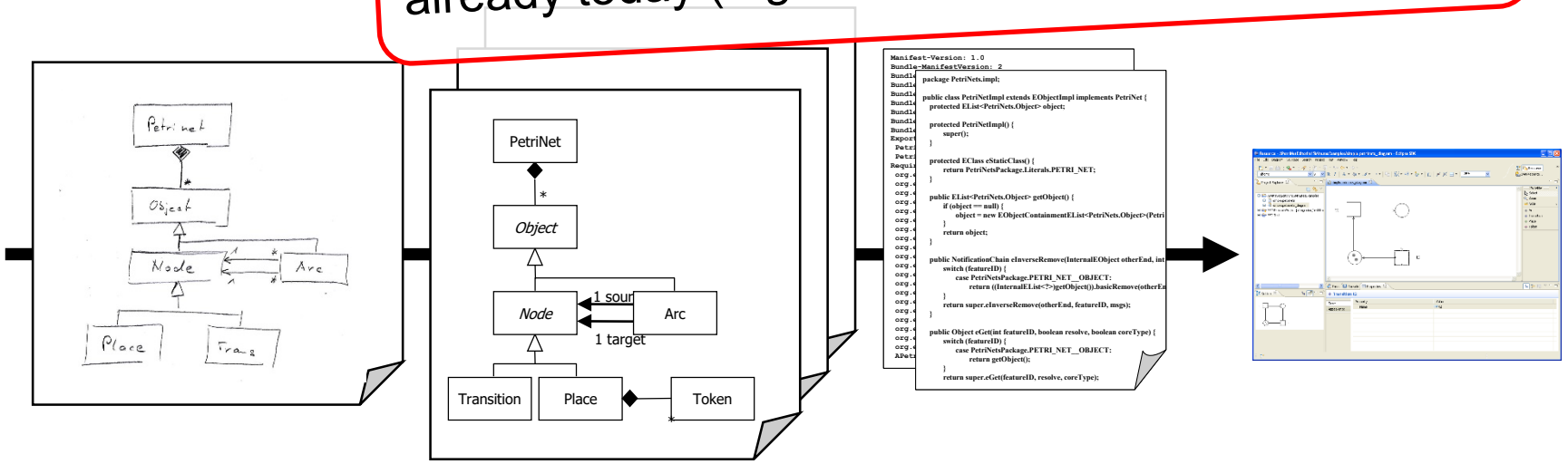


Exploit conceptual artefacts instead (and generate software from them).

Vision

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

There are tools that partially support this idea already today (e.g. Eclipse and EMF).



Analysis

Design

Implementation

~~Coding~~

+ analysis and verification!

Code is generated

”Buzzwords”

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Model Driven Architecture® (MDA®)
  OMG™ software development approach for separating business logic from platform specific details
  - using models
  - transformations
  - automatic generators (for code and other models)
- Model-based Software Engineering (MBSE)
  General term for making “better” use of models for easing the software development

MDD®

DSL

M2M, M2T

JET

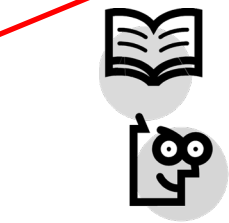
Ultimately: Getting rid of programming resp. technical artefacts.

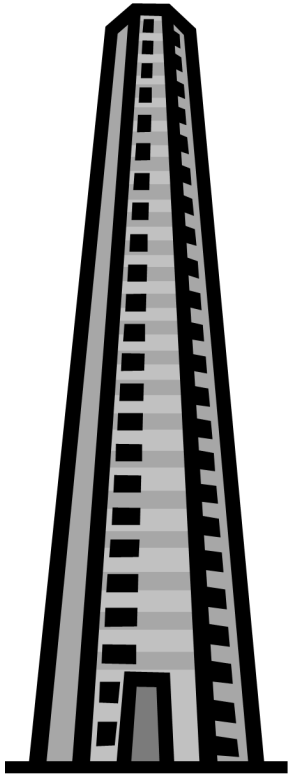
Software Engineering

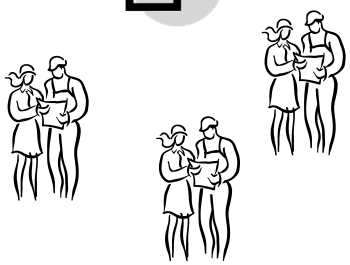
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

If somebody has built a garage, we would never let him built a sky scraper, would we?

But, if somebody knows how to program, we let him do software engineering, don't we?

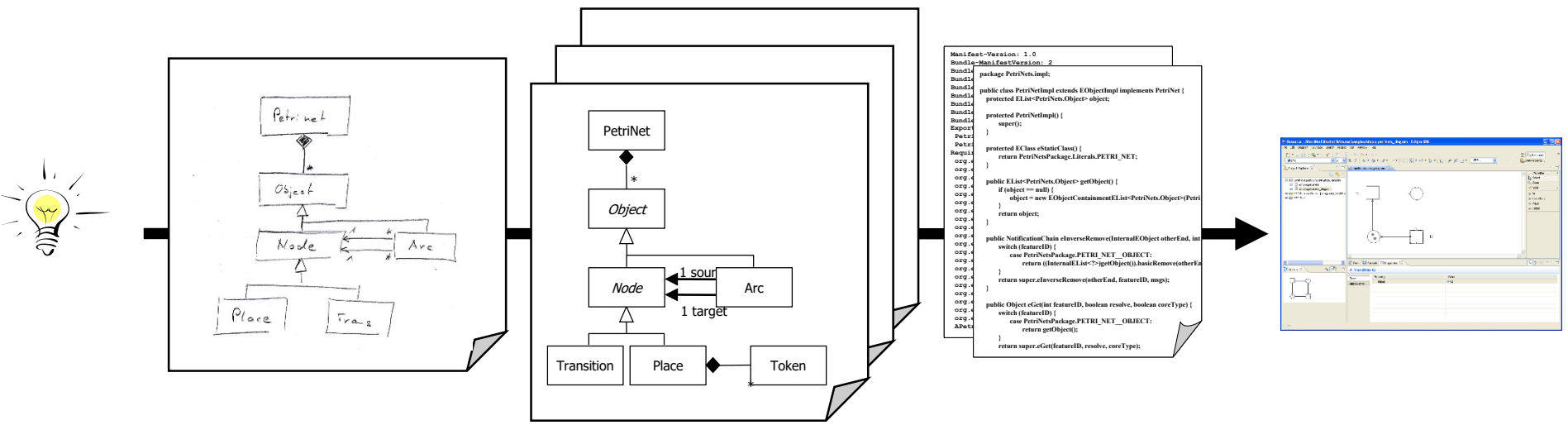






Vision

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Analysis

Design

Implementation

~~Coding~~

Vision

How do Petri nets come into this vision?

- How can we exploit this for developing Petri net tools?
- How can Petri nets be used for making software?

We do a bit of both in this course!

Schedule

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

9^{00}-10^{30}: Session 1

MBSE overview

- vision, idea, concept
- technologies

11^{00}-12^{30}: Session 2

Technology details
Modelling behaviour

- Petri nets and/in software
- Behaviour coordination

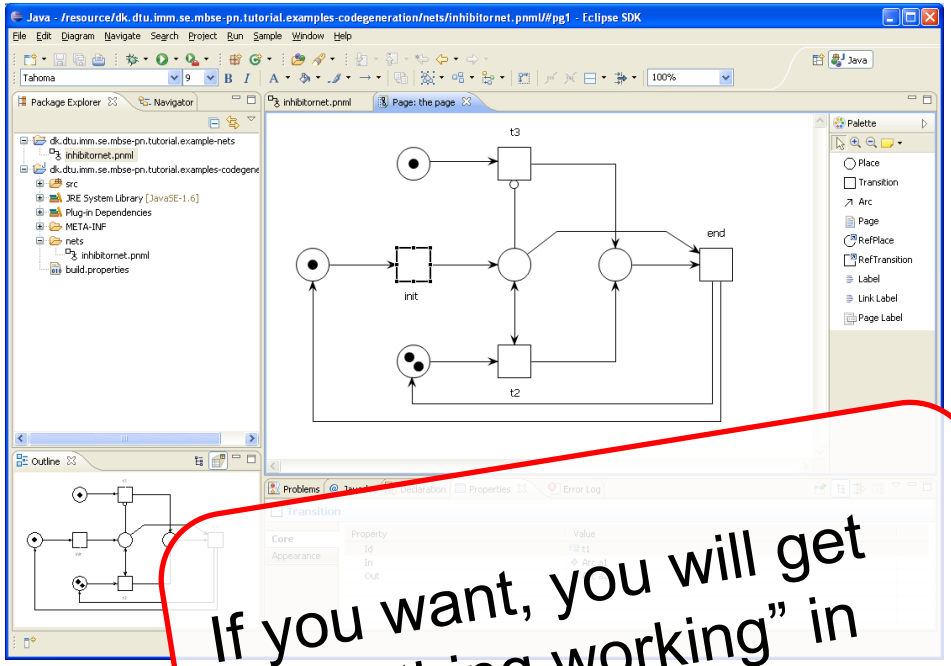
13^{30}-15^{00}: Session 3

Model-based technology in Petri nets

- PNML (overview)
- ePNK (overview)

15^{30}-17^{xx}: Session 4

Project (hands-on)



If you want, you will get "something working" in today's hands-on session.

If you are interested in ECTS points: + 30hours work on project after today's sessions

Model-based Software Engineering for/with Petri nets

MBSE: Introduction and Overview

Ekkart Kindler

DTU Compute
Department of Applied Mathematics and Computer Science

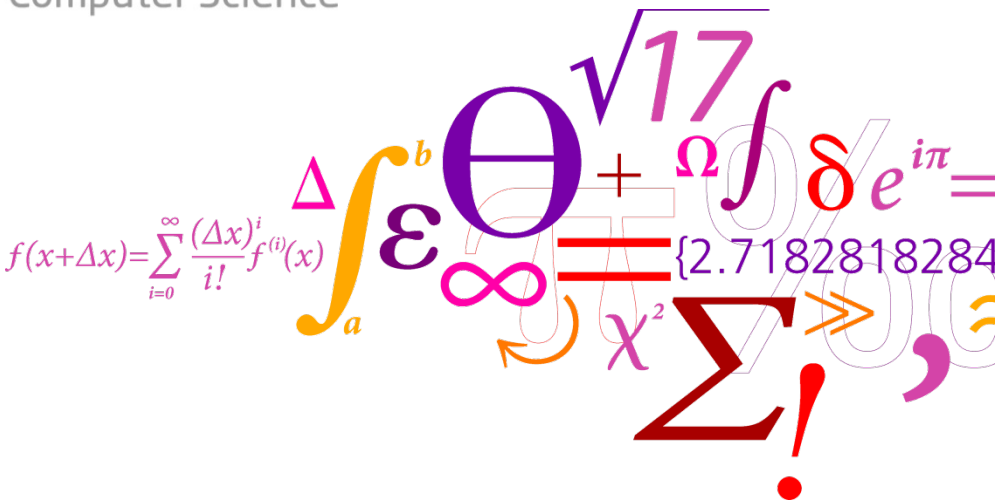
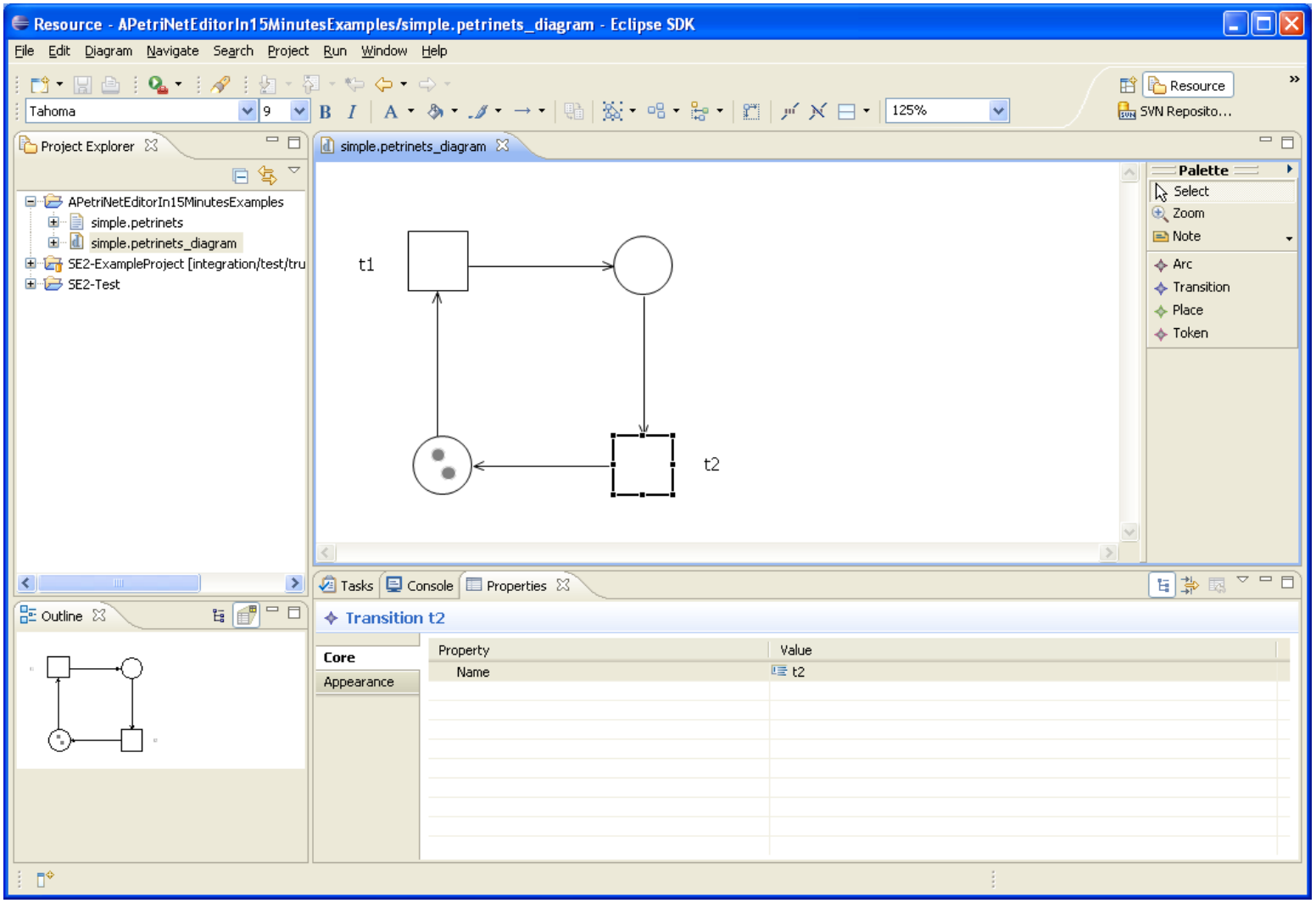


Table of Contents

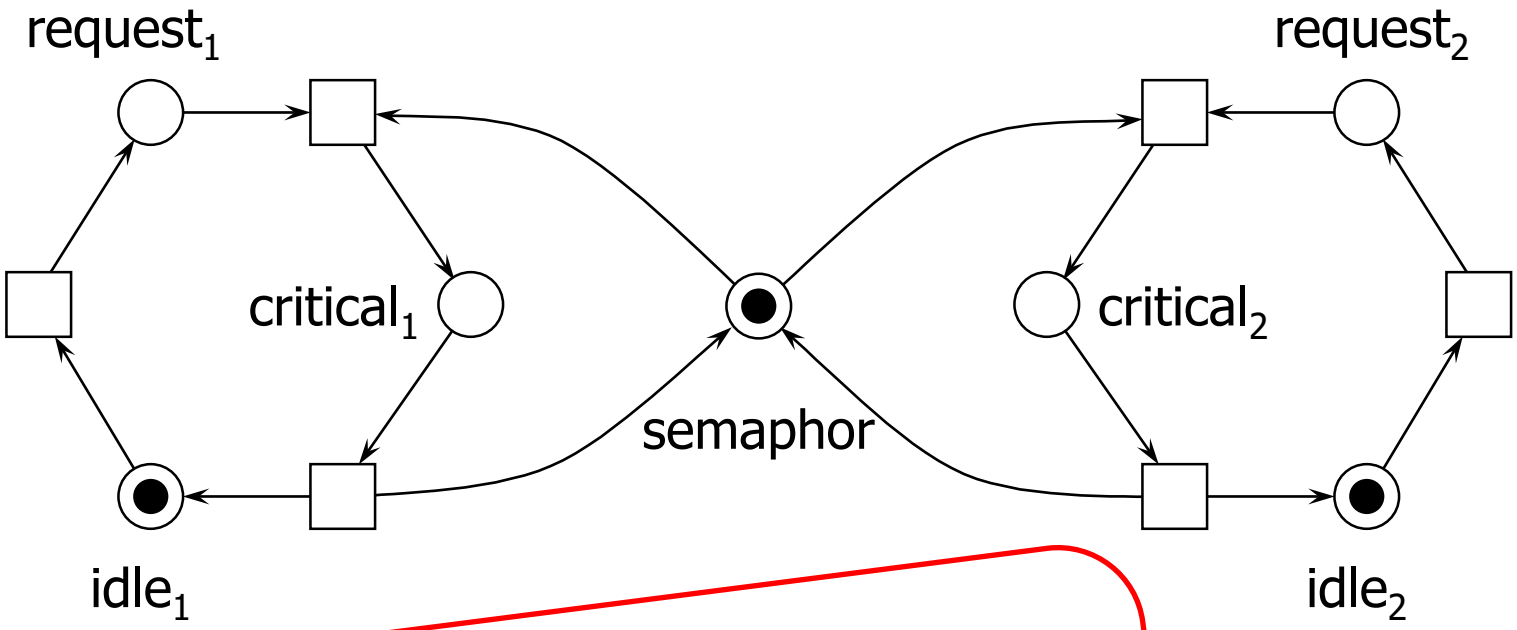
2.1. Example: A Net Editor

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



A Model (Petri net)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

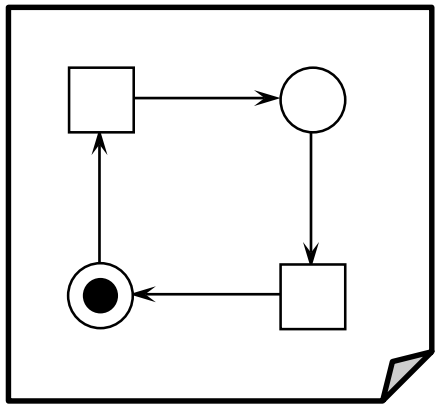


A rule of modelling:
→ Before making a model look at examples (instances)
→ Before making a meta-model look at the models (instances of meta-models)

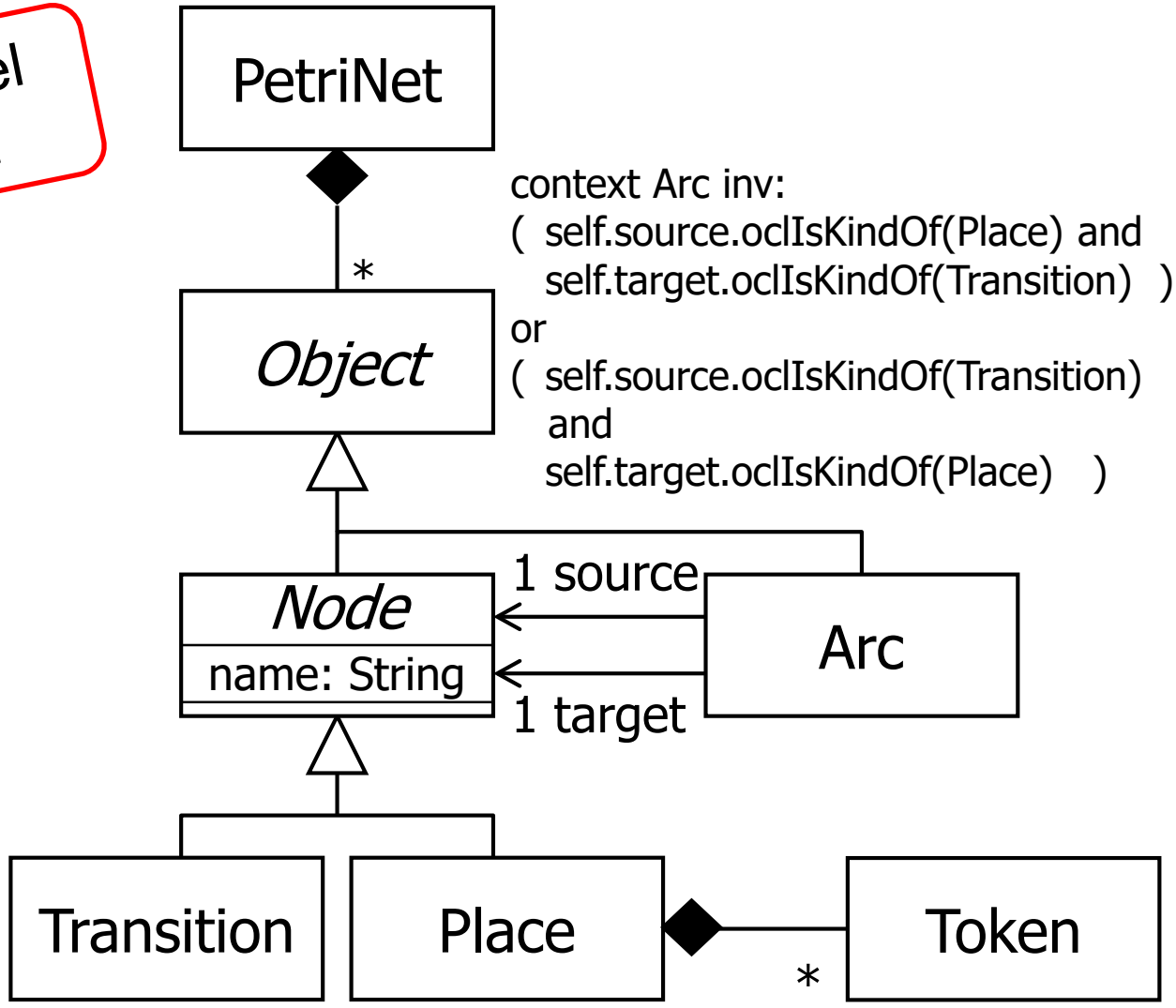
Models and Meta Models

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

→ PNML meta model ISO/IEC 15909-2



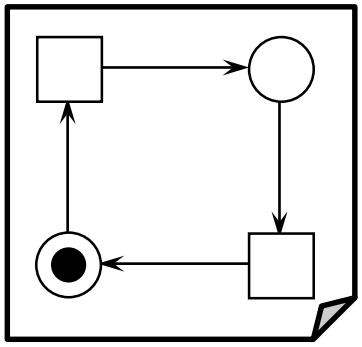
Petri net model



"Meta model" for Petri nets
(as a UML class diagram + OCL)

Syntax (abstract and concrete)

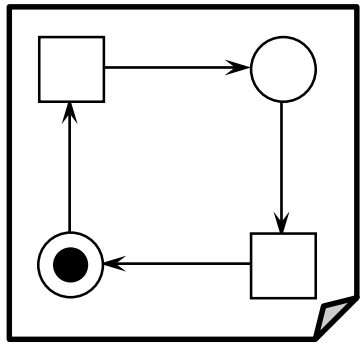
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



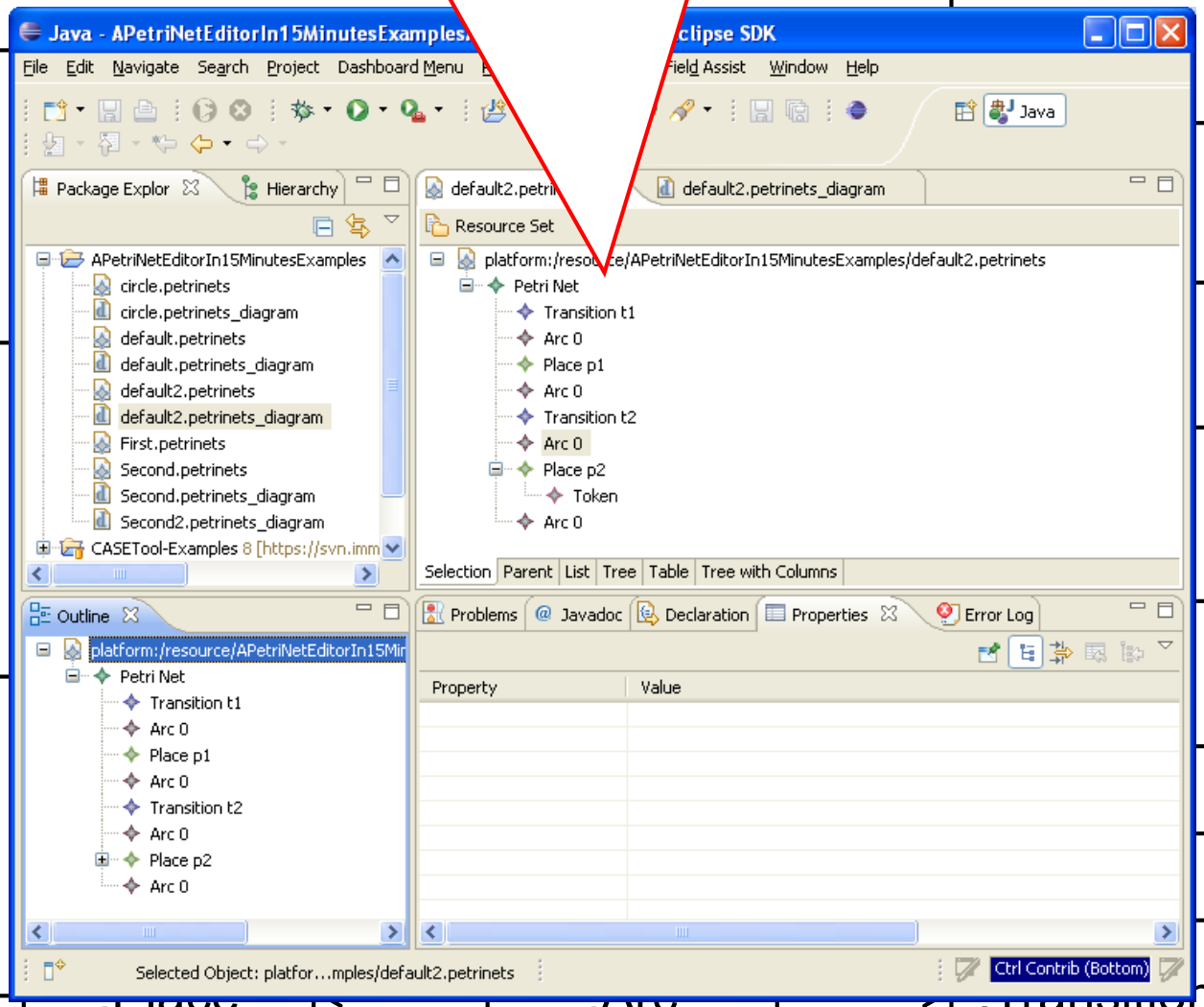
graphical /
concrete syntax

Syntax (abstract and co

Abstract syntax could also “look” like this (in EMF generated tree editor)!



graphical / **concrete syntax**



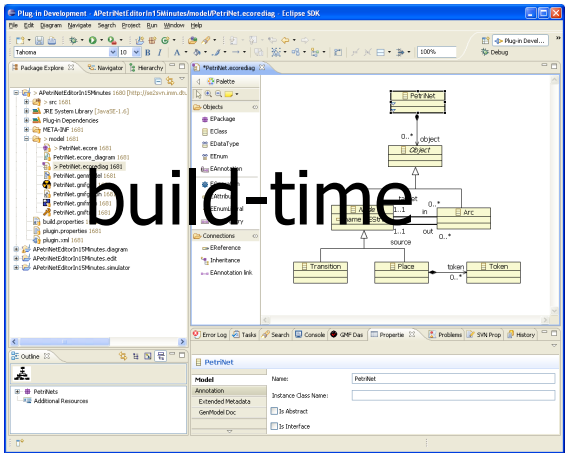
:Token

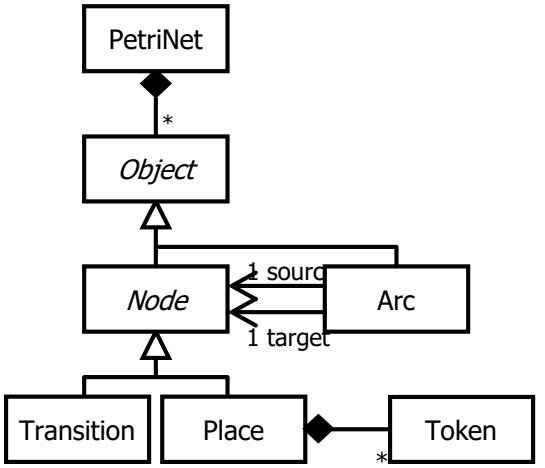
Overview

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

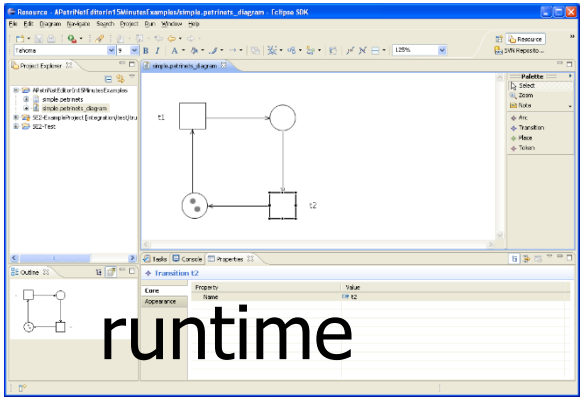
meta model

build-time



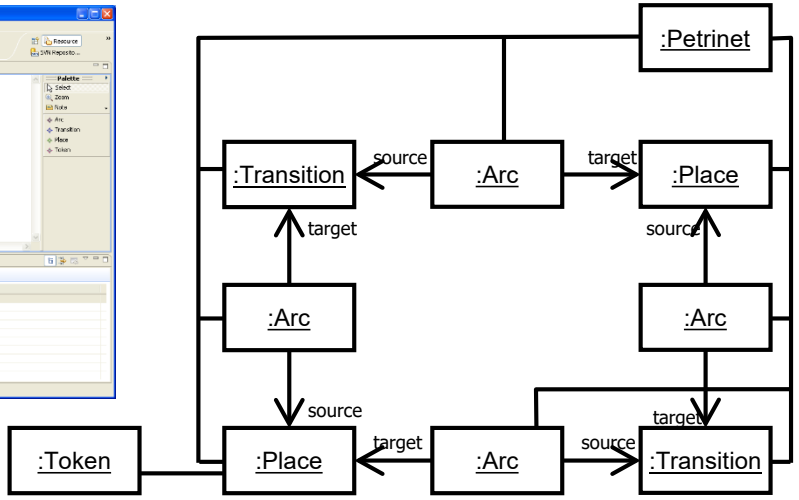


is an instance of



model

runtime



Benefits of Modelling

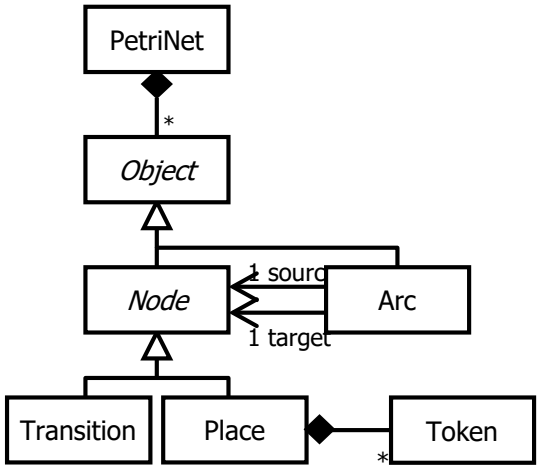
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

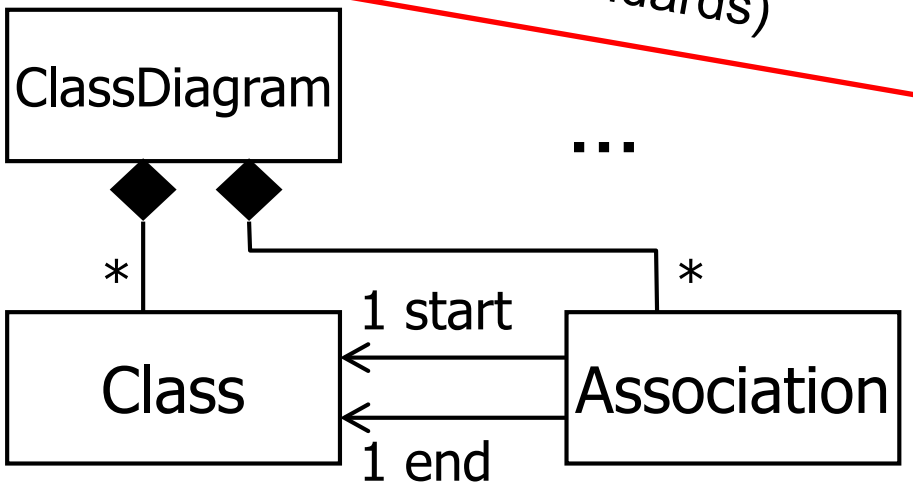
- Better understanding
- Mapping of instances to XML syntax (XMI) (canonical exchange format for instances)
- Automatic code generation
  - API for creating, deleting and modifying model
  - Methods for loading and saving models (in XMI)
  - Standard mechanisms for keeping track of changes (observers)
  - Transactional changes (ACI)
  - Database access / persistance (D)

Class Diagrams are Models too

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Disclaimer: The real model of UML is much more evolved (see MOF and UML standards)





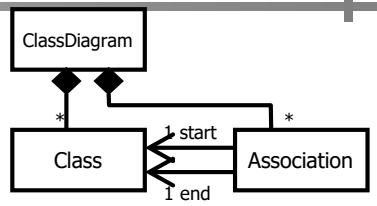
UML model
(class diagrams)

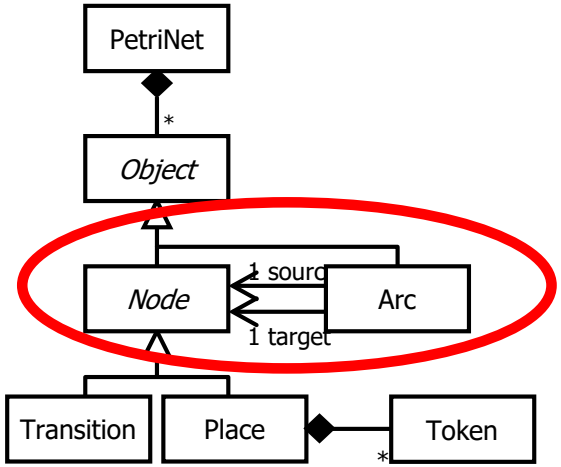
Meta model for UML
class diagrams

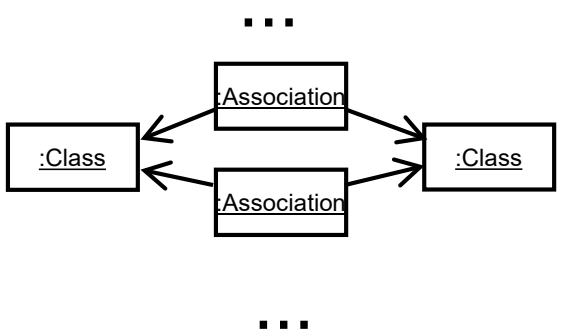
Now, the term “meta” model makes more sense!

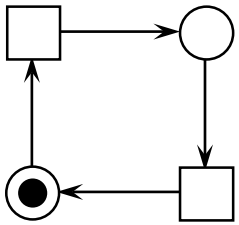
Different Meta-levels: MOF

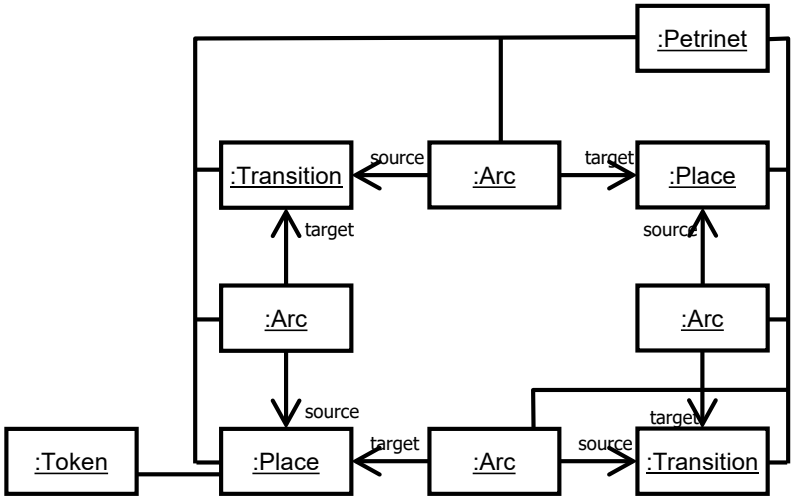
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler





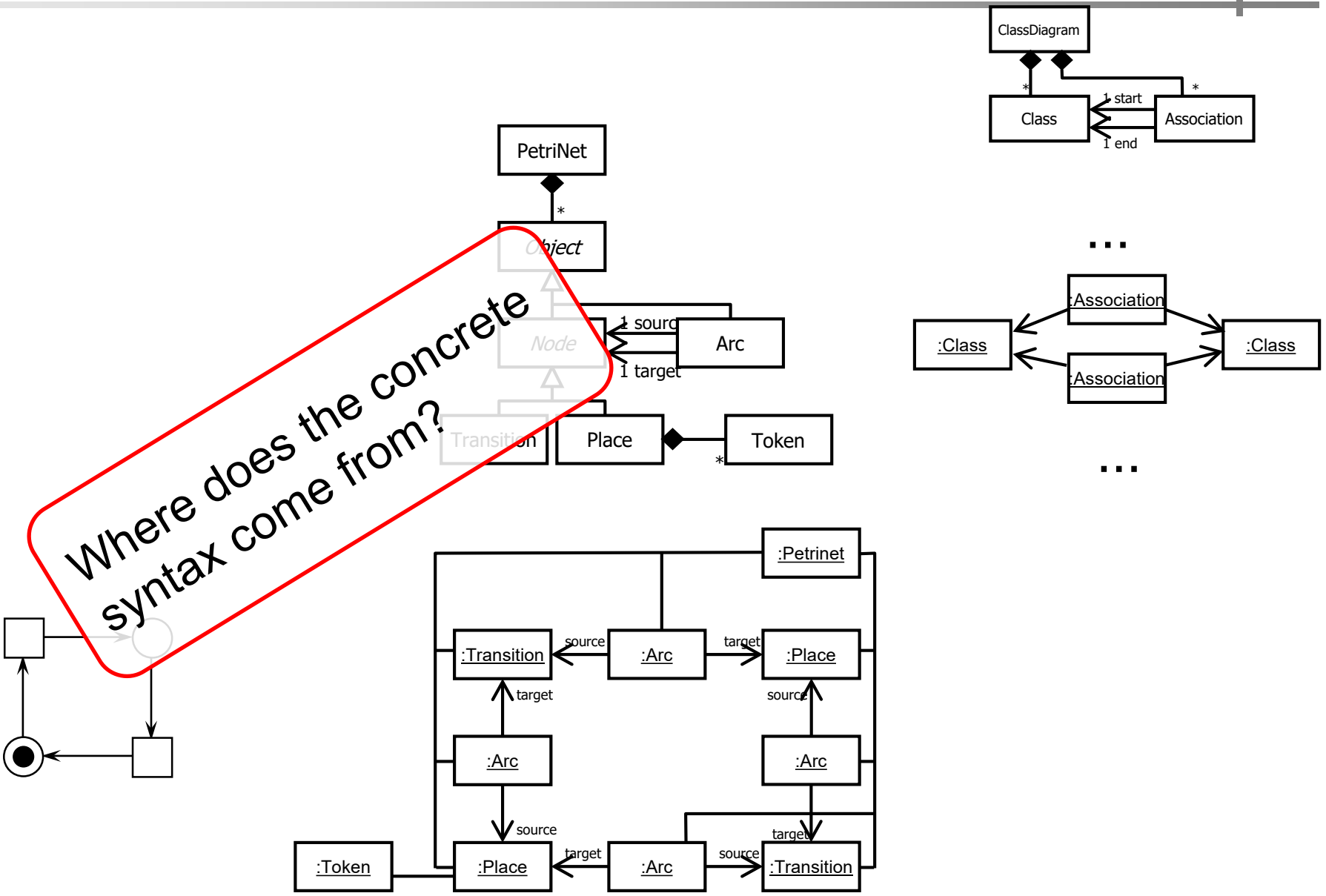






Different Meta-levels: MOF

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



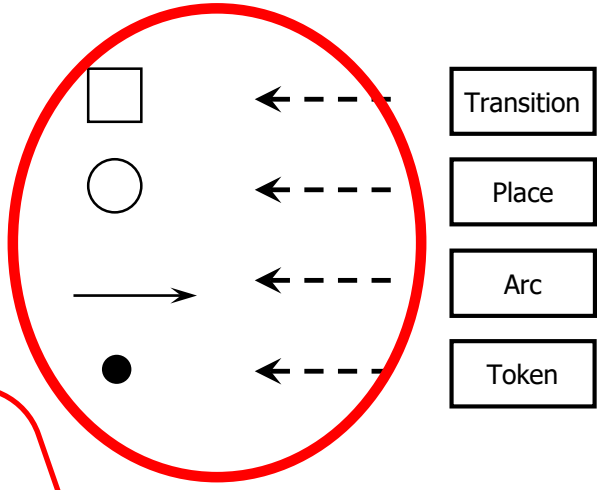
EMF/GMF Technology

meta model

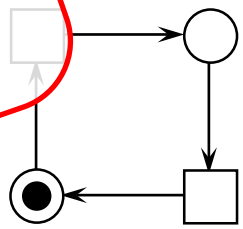
is instance of model

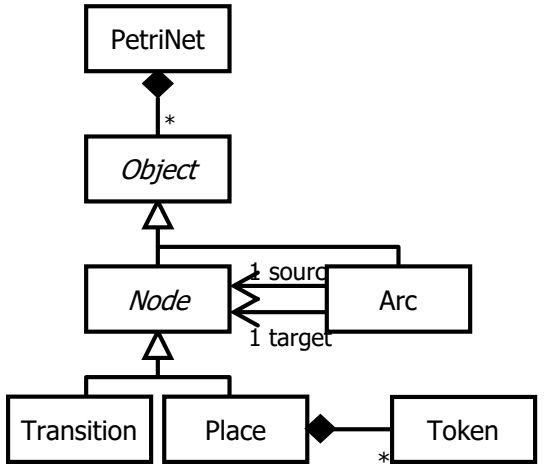
A Petri net editor in 15 minutes!

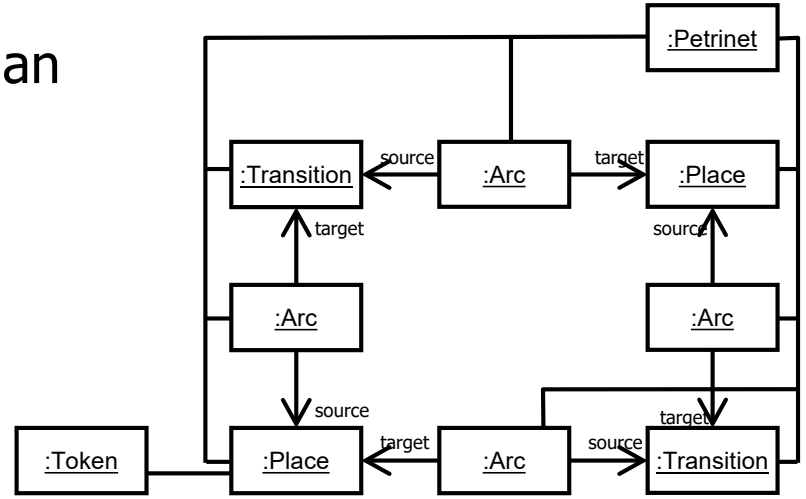
Not kidding!



generate an editor

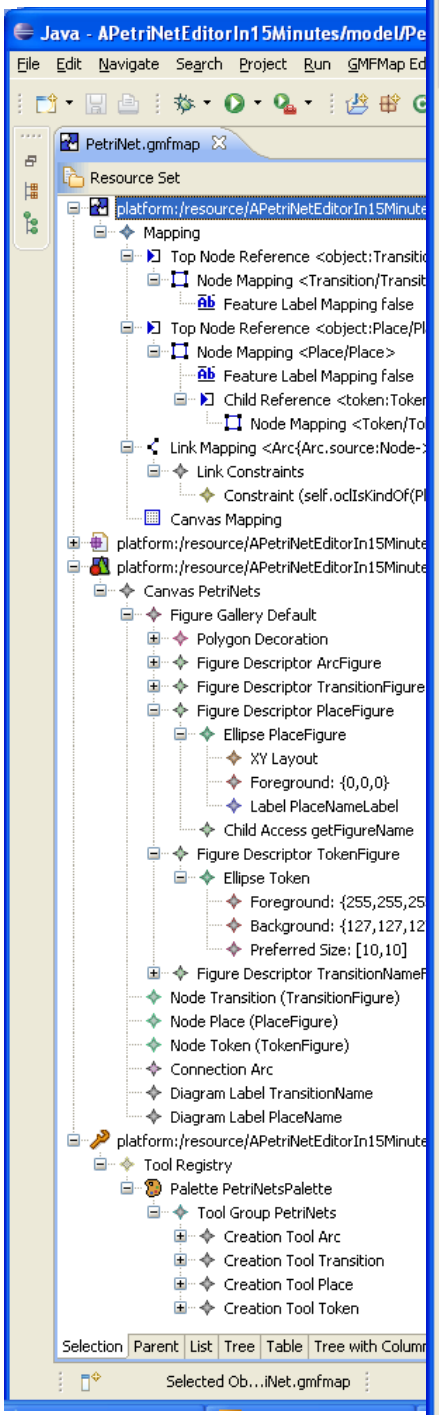






concrete syntax

abstract syntax



Resource Set

platform:/resource/APetriNetEditorIn15Minutes/model/PetriNet.gmfmap

- Mapping
  - Top Node Reference <object:Transition/Transition>
    - Node Mapping <Transition/Transition>
      - Feature Label Mapping false
  - Top Node Reference <object:Place/Place>
    - Node Mapping <Place/Place>
      - Feature Label Mapping false
      - Child Reference <token:Token/Token>
        - Node Mapping <Token/Token>
  - Link Mapping <Arc{Arc.source:Node->Arc.target:Node}/Arc>
    - Link Constraints
      - Constraint (self.oclIsKindOf(Place) and oppositeEnd.oclIsKindOf(Transition)) or...
  - Canvas Mapping

platform:/resource/APetriNetEditorIn15Minutes/model/PetriNet.ecore

platform:/resource/APetriNetEditorIn15Minutes/model/PetriNet.gmfgraph

- Canvas PetriNets
  - Figure Gallery Default
    - Polygon Decoration
    - Figure Descriptor ArcFigure
    - Figure Descriptor TransitionFigure
    - Figure Descriptor PlaceFigure
      - Ellipse PlaceFigure
        - XY Layout
        - Foreground: {0,0,0}
        - Label PlaceNameLabel
      - Child Access getFigureName
    - Figure Descriptor TokenFigure
      - Ellipse Token
        - Foreground: {255,255,255}
        - Background: {127,127,127}
        - Preferred Size: [10,10]
    - Figure Descriptor TransitionNameFigure
  - Node Transition (TransitionFigure)
  - Node Place (PlaceFigure)
  - Node Token (TokenFigure)
  - Connection Arc
  - Diagram Label TransitionName
  - Diagram Label PlaceName

platform:/resource/APetriNetEditorIn15Minutes/model/PetriNet.gmftool

- Tool Registry

GMF not in the focus of this course. But if you are interested, please, feel free to ask.

If I'd start the ePNK project today, I'd probably chose Graphiti:
https://www.eclipse.org/graphiti/

2.2 Benefits of Modelling

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Better Understanding
- Mapping of instances to XML syntax (XMI)
- Automatic Code Generation
  - API for creating, deleting and modifying model
  - Methods for loading and saving models (in XMI)
  - Standard mechanisms for keeping track of changes (observers)
  - Editors and GUIs

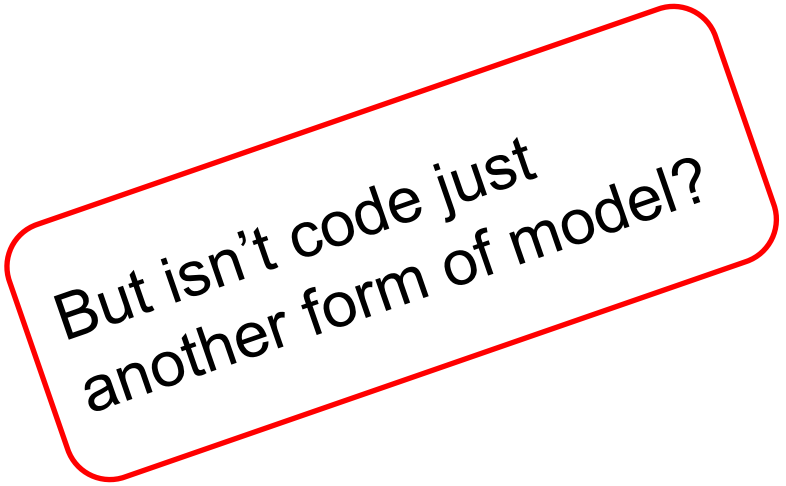
NB: All this is "standard functionality"?

No programming at all!

Purpose of (good) models

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- **Abstraction**
- Focus
- Simplification
- Separation



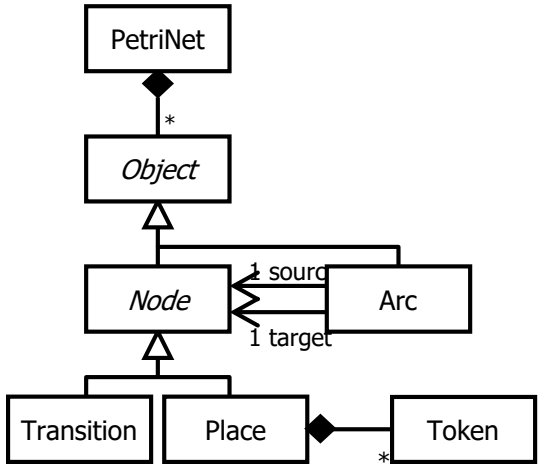
- Understanding
- Communication
- Analysis and verification
- Execution (interpretation / code generation)

My favourite example

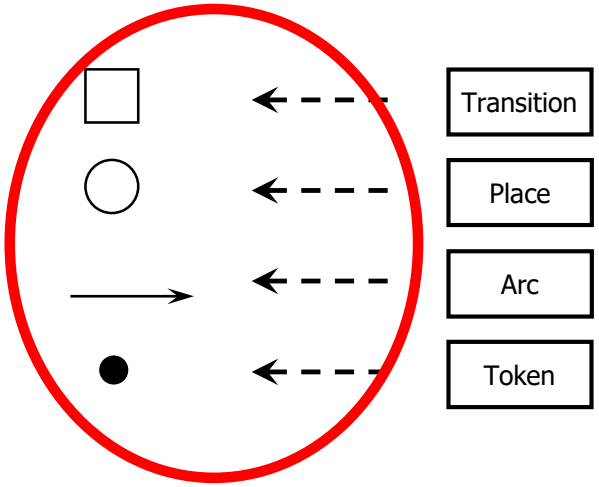
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

From this (EMF) model for Petri nets:
Generation of (Java) code for

- all classes
- methods for changing the Petri net
- loading and saving the Petri net as XML files (→XMI)



With this and some more GMF information:
Generation of the Java code of a graphical complete editor (with many fancy features). No programming at all.



Almost all you need to say about a Petri net editor.

Model-based SE

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Conventional: We can generate parts of the code form UML class diagrams automatically (MDA, MDE, **EMF**, EMFT/GMF)

- Class diagrams → Java class stubs with standard access methods (see basic project)
- Implementation of standard behaviour:
  - Loading and saving models
  - Accessing and modifying the models
  - Editors and graphical user interfaces
- The actual functions is implemented by hand (e.g. simulator, state-space generator, …)

Advanced: Actual functions also „modelled“ and code generated

We come back to that in the modelling behaviour session.

3. Principles and Concepts

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Model, view, controller
- ”Meta”, ”meta-meta”, who has more?
- MOF

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

The **domain models** are an (the) essential part of the software

In addition to that we need

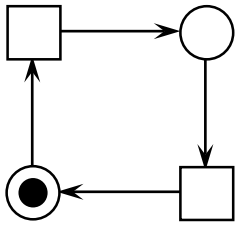
- Information about the presentation (view) of the model to the user
- The coordination / interaction with the user

Note: These parts of the software can also be modelled! Petri nets are probably particularly suited for this.

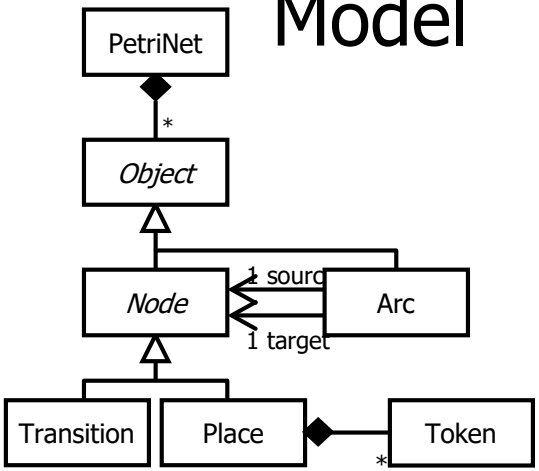
Model View Controller (MVC)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

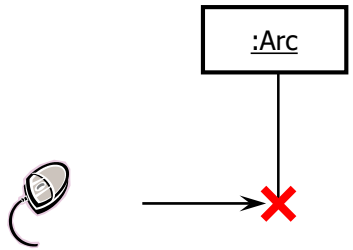
View



Model

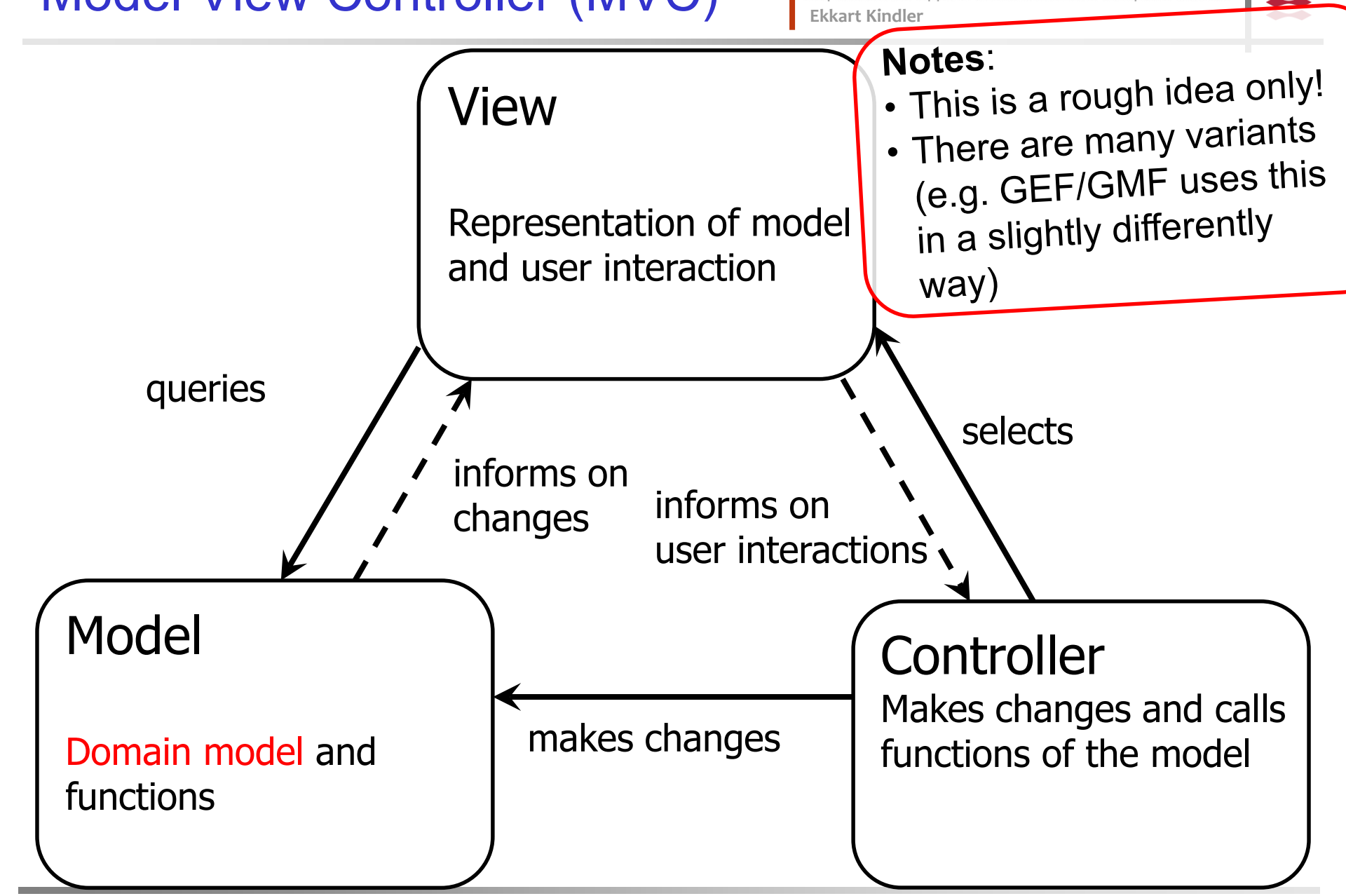


Controller



Model View Controller (MVC)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Notes:
- This is a rough idea only!
- There are many variants (e.g. GEF/GMF uses this in a slightly differently way)

Model View Controller (MVC)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

MVC is a principle (pattern / architecture) according to which software should be structured

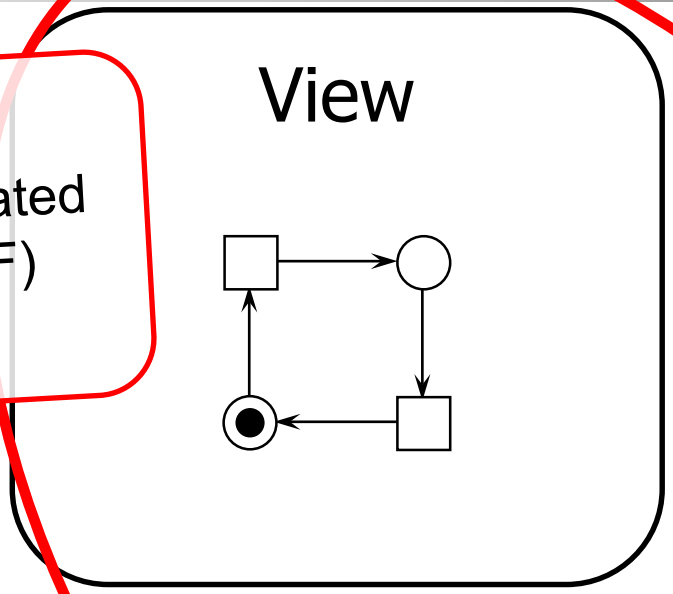
Eclipse and GEF (as well as GMF) are based on this principle and guide (force) you in properly using it

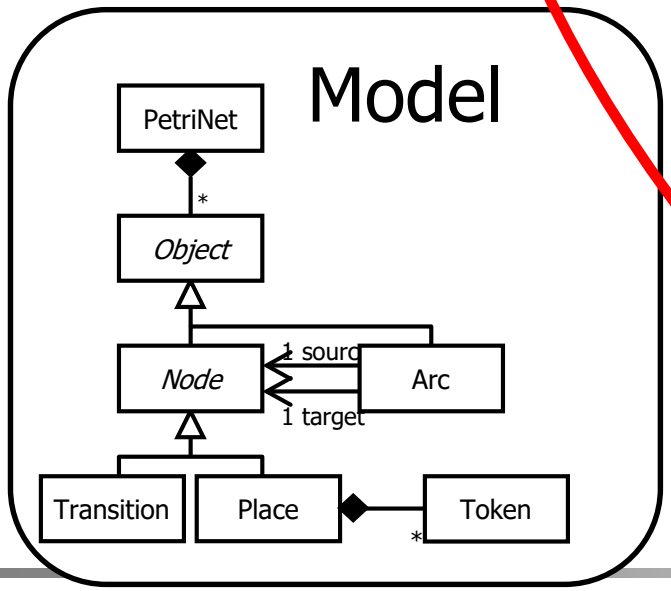
When using MBSE technologies, many more design patterns are relevant.

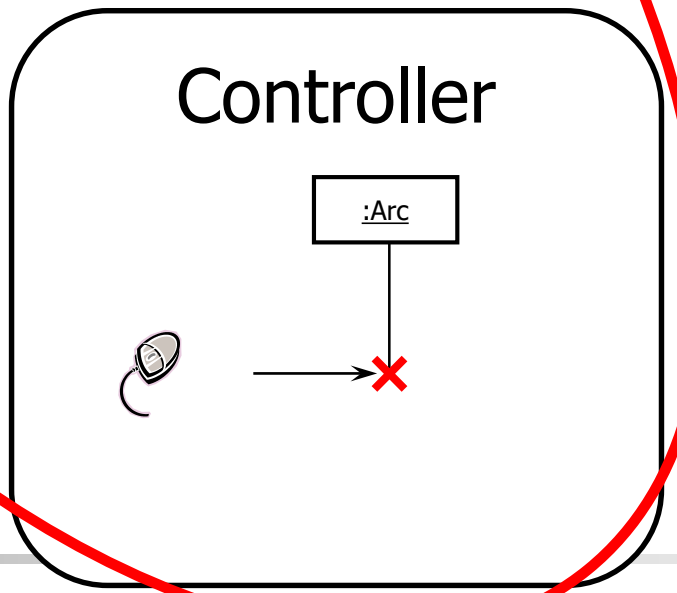
EMF, GMF and MVC

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

This part can be generated automatically (e.g. GMF)

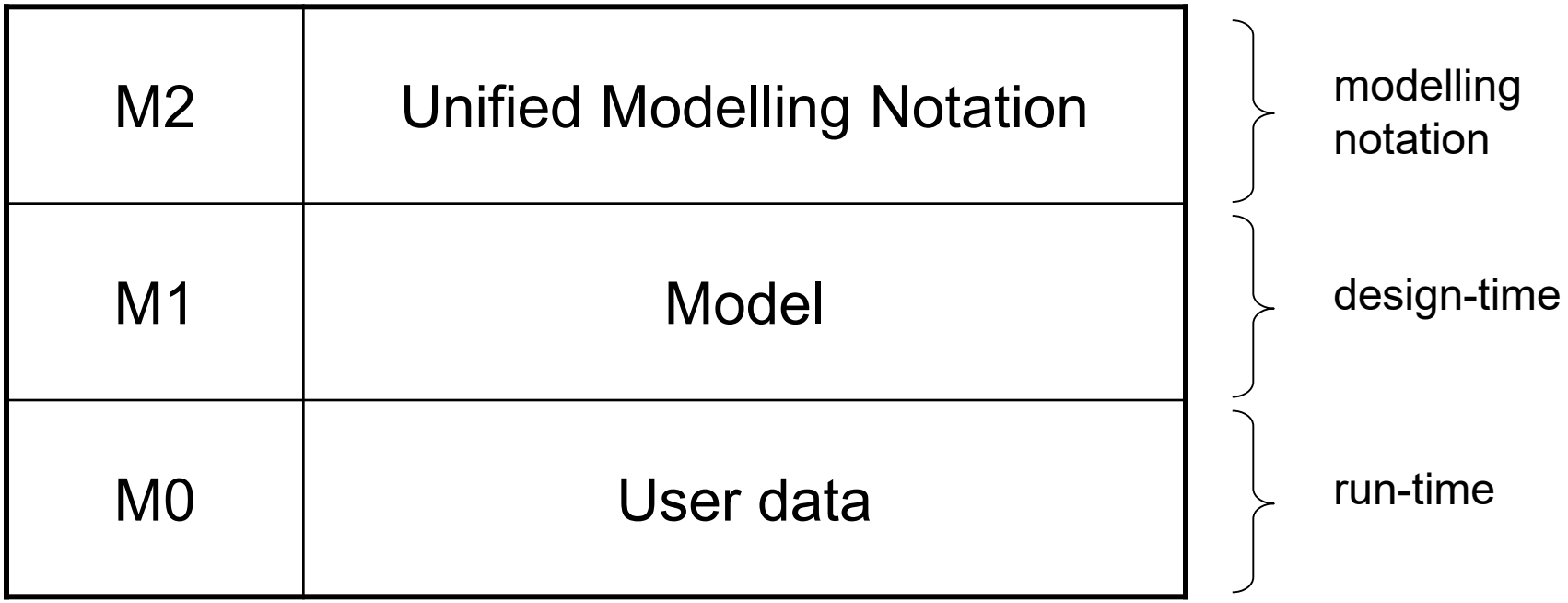






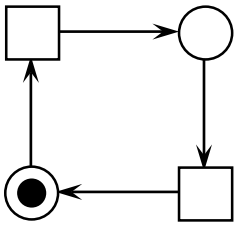
3.2 Meta Object Facility

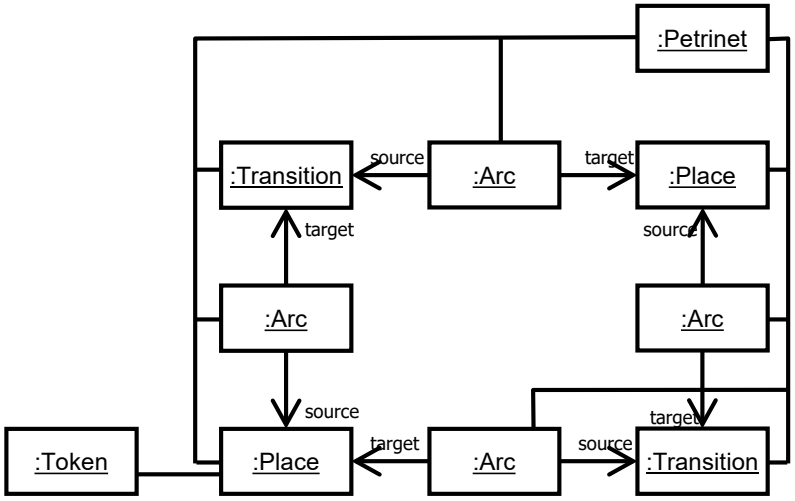
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

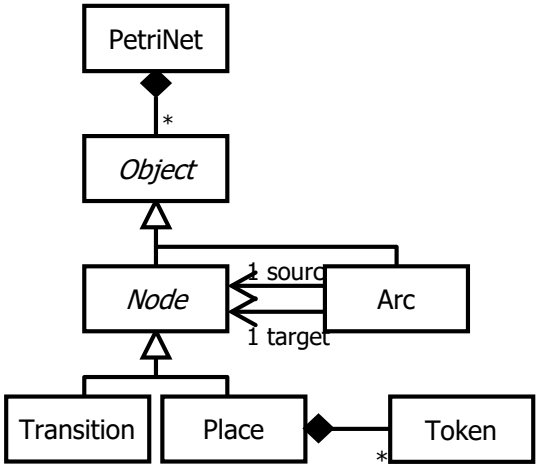


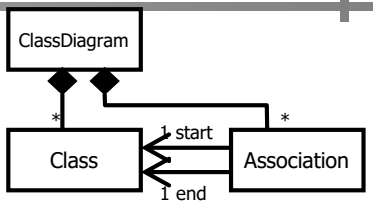
Re: Example

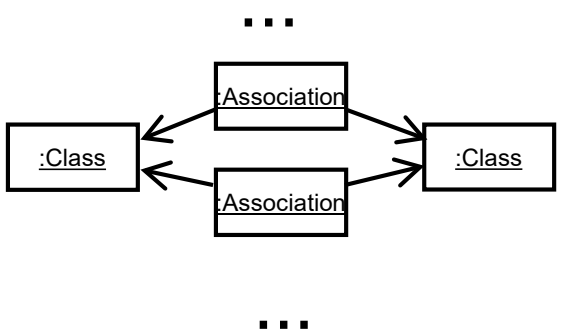
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler









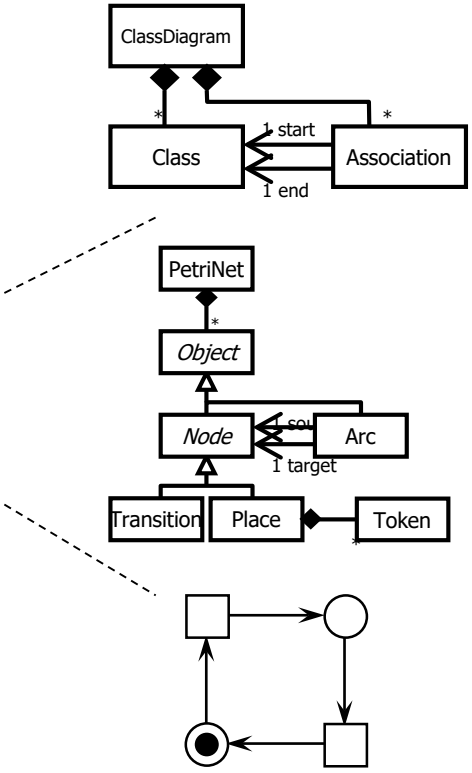


Outset

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

What, if we would like to use another modelling notation!

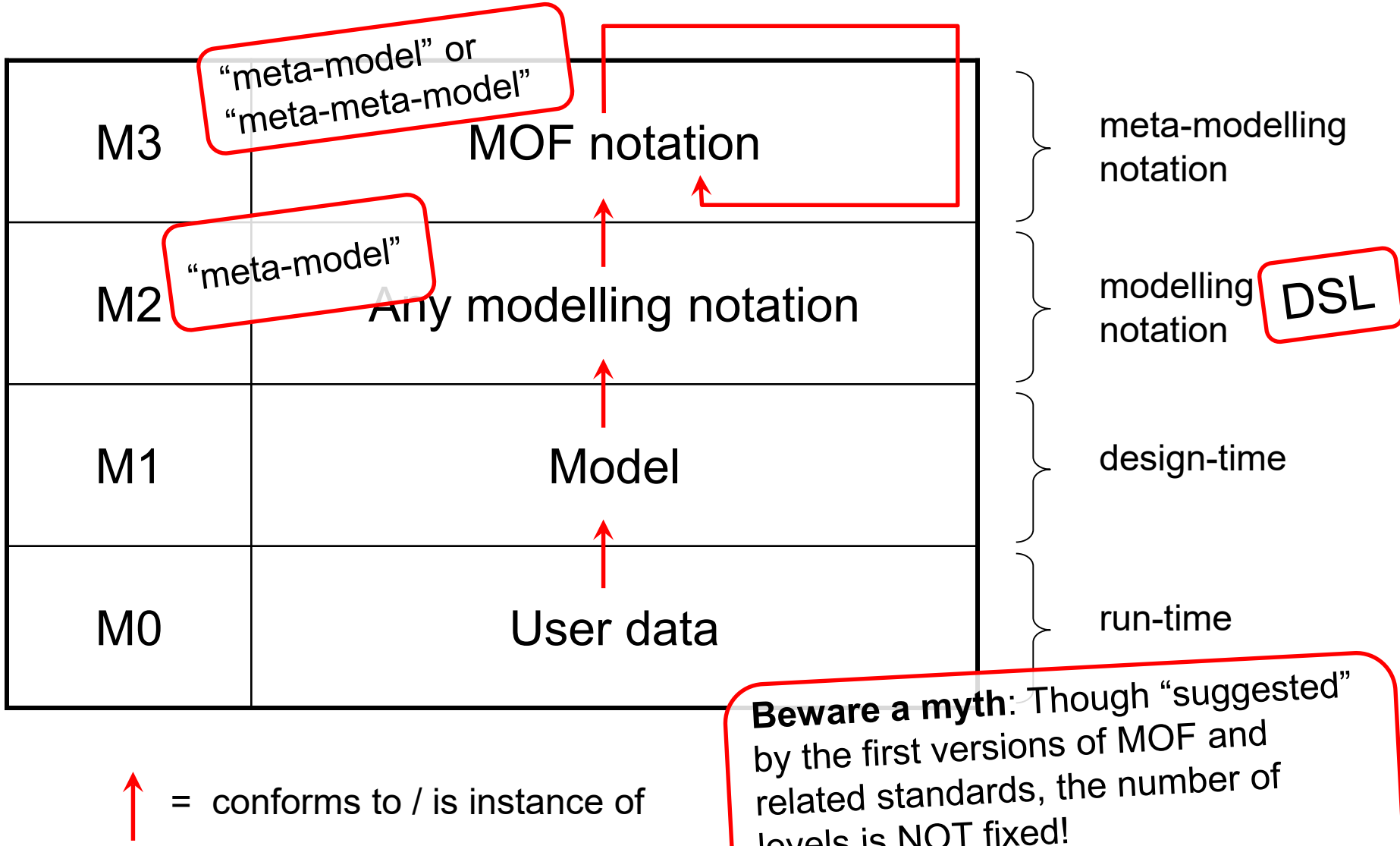
M2	Unified Modelling Notation
M1	Model
M0	User data



↑ = conforms to / is instance of

Meta Object Facility (MOF)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



MOF: Discussion

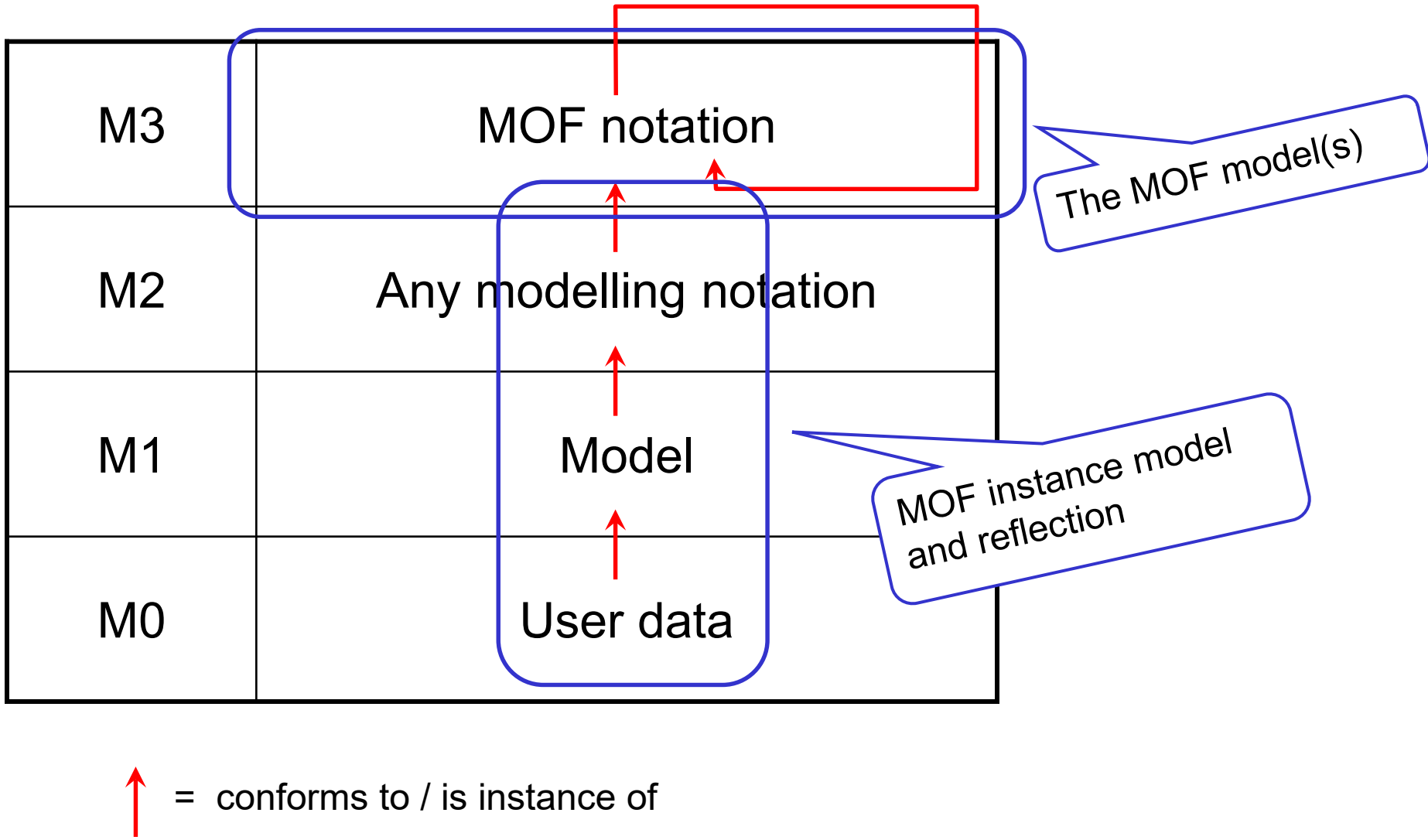
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Is that any good?
  - There is one level that we did not have before!
    So, this seems to be more complicated!
  - If UML can be defined in terms of itself, why should we define it in terms of something else?

What do **you** think?

Meta Object Facility (MOF)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



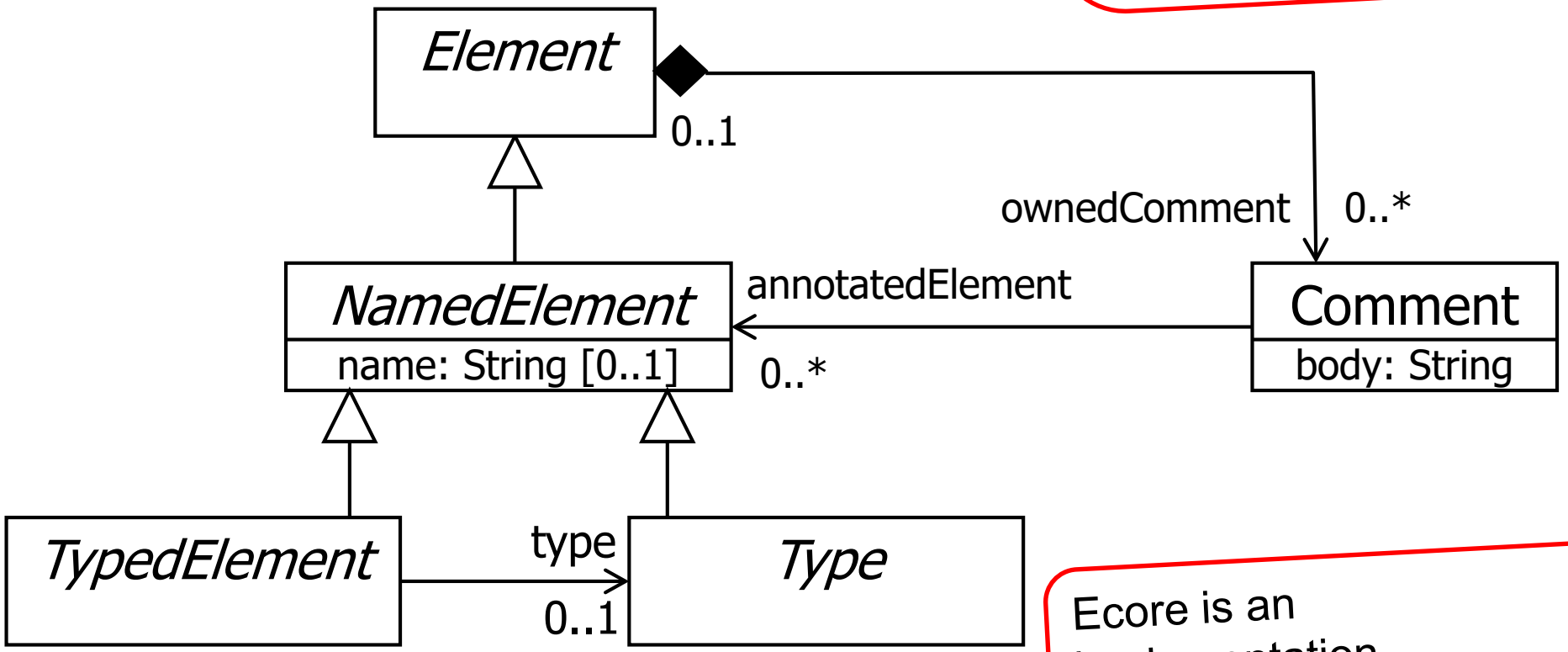
(Part of) the EMOF model

EMOF Types

the "E" stands for "essential"

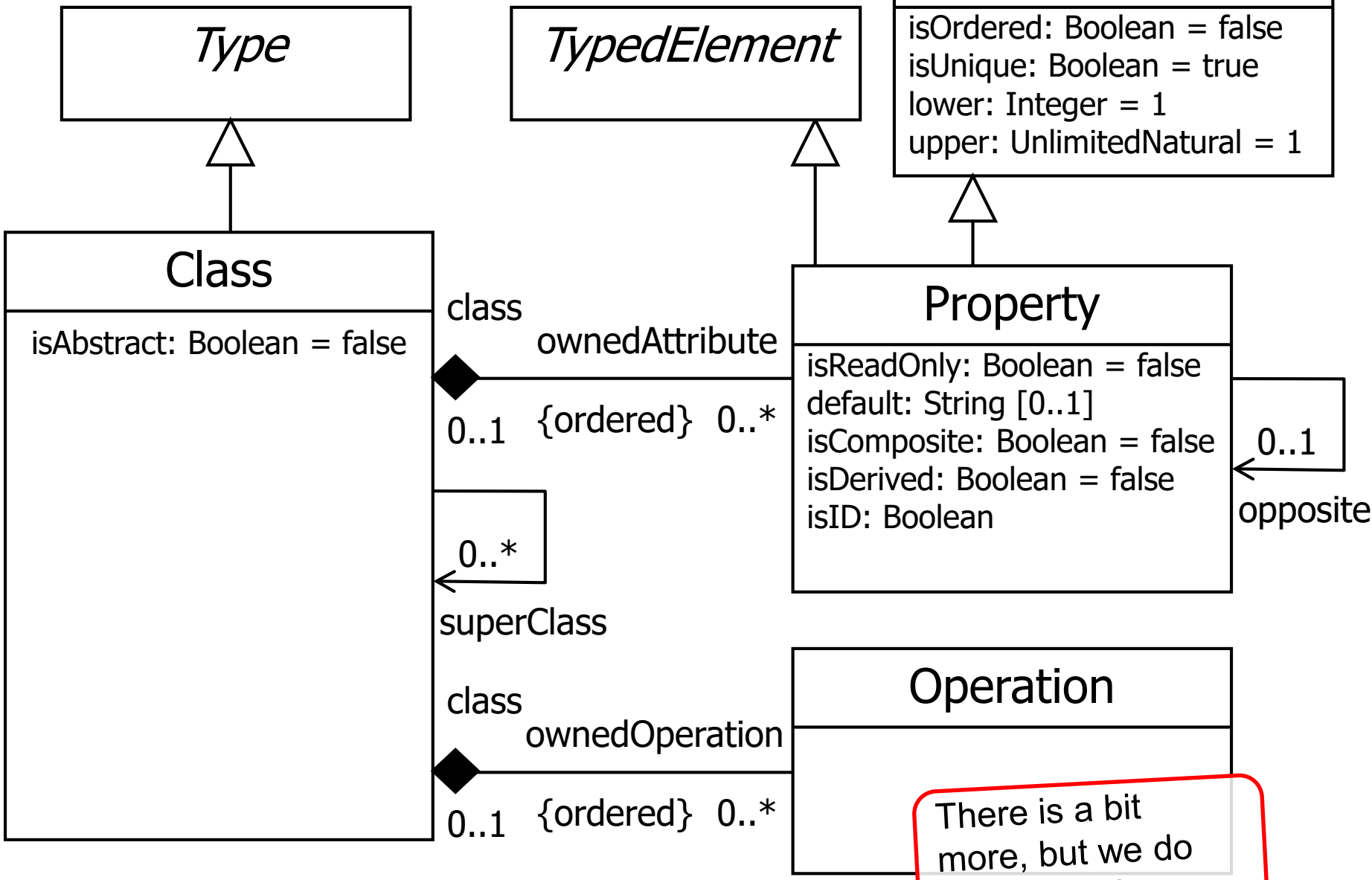
Actually, this comes from the UML infrastructure Core::Basic.

The MOF standard refers to and uses concepts and notations from the UML standard (infrastructure).



Ecore is an implementation of MOF (actually) EMOF.

EMOF Classes



Meaning of ”Meta-”

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Meta (from Greek: μετά = "after", "beyond", "with", "adjacent", "self"), is a prefix used in English in order to indicate a concept which is an abstraction from another concept, used to complete or add to the latter.

In epistemology, the prefix **meta-** is used to mean *about (its own category)*. For example, metadata are data about data, something about something (who has produced them, when, what format the data are in and so on). Similarly, metamemory in psychology means an individual's knowledge about whether or not they would remember something if they concentrated on recalling it. Furthermore, metaemotion in psychology means an individual's emotion about his/her own basic emotion, or somebody else's basic emotion.

Another, slightly different interpretation of this term is "about" but not "on" (exactly its own category). For example, in linguistics a grammar is considered as being expressed in a metalanguage, or a sort of language for describing *another* language (and not itself). A **meta-answer** is not a real answer but a reply, such as: "*this is not a good question*", "*I suggest you ask your professor*". Here, we have such concepts as meta-reasoning and meta-knowledge.

…

From: http://en.wikipedia.org/wiki/Meta

”The Meaning of Meta”

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Connotations and meaning of ”meta” in Software Engineering:

- beyond, ”one level higher”
- possibly self-referential
  (with all the problems of self-referentiality)

Self-references are at the core of all paradoxes.
Example: “This statement is wrong”!

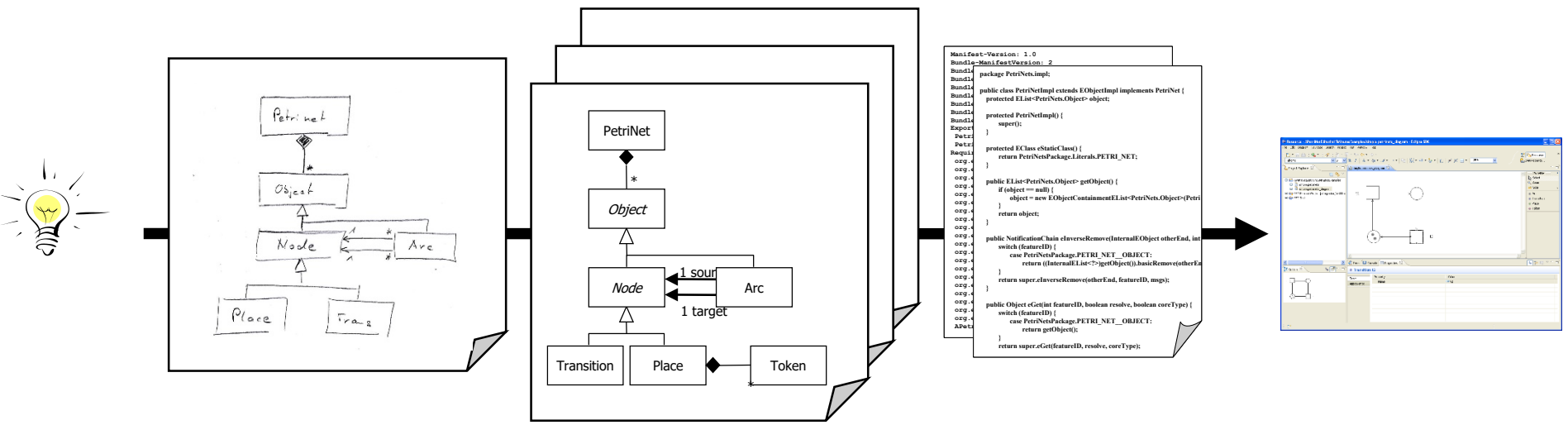
Often also:

- ~~a UML model~~
- ~~a class diagram~~

Abuse of language! I guess, introduced by people working only or just too much on the meta-level.

Re: Motivation

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Analysis

Design

Implementation

~~Coding~~

"Standard code" is generated

4. Technologies

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

If we want to get software automatically from models,

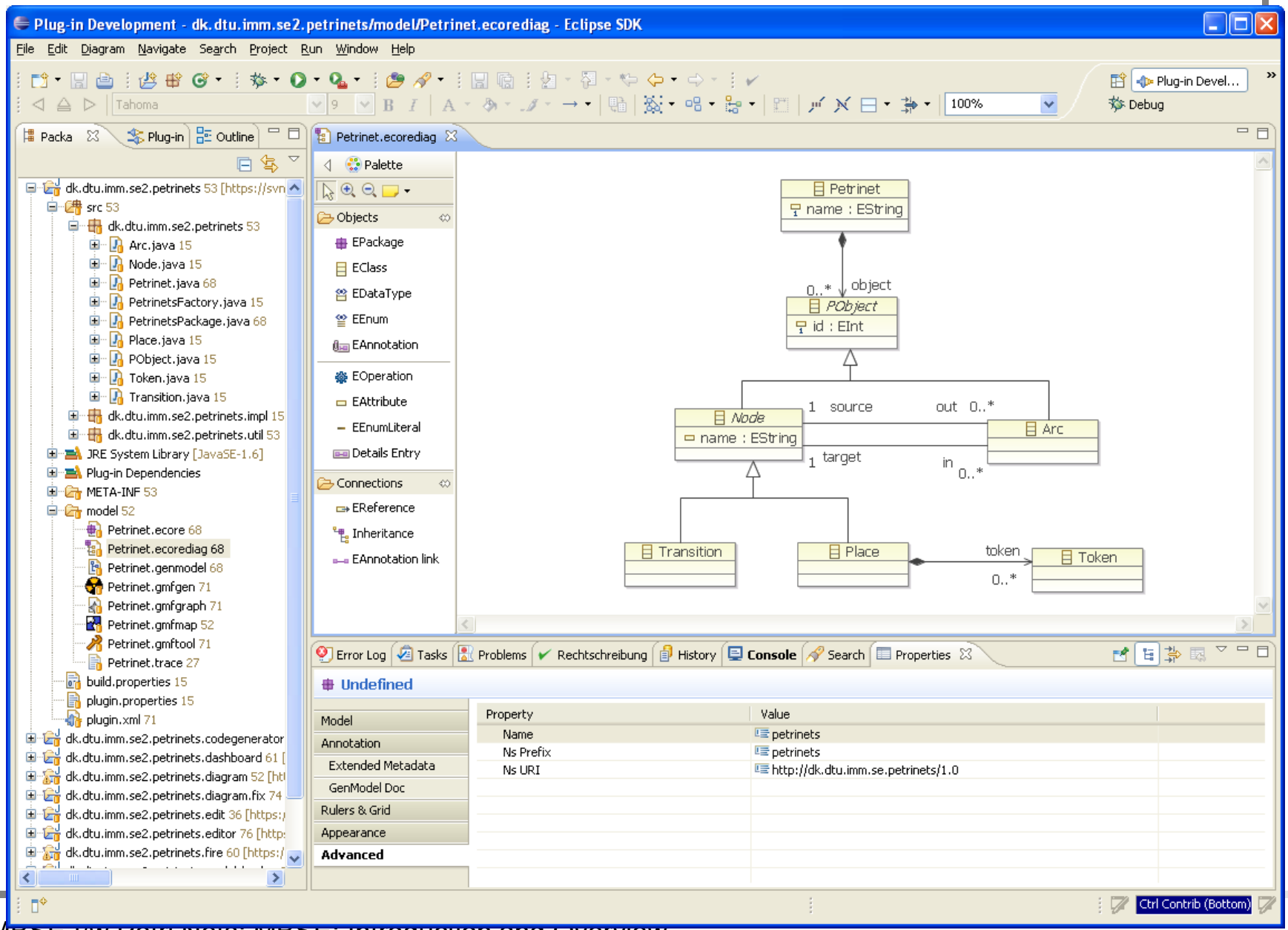
- we need to have a technology for making models
- transforming between models and into code

EMF provides specific mechanisms for making models and transforming them into (Java) code.

What if we want other modelling notations (DSL) or other transformations?

4.1. EMF/GMF for DSL

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



4.2. Transformations

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Basically, two kinds of transformations in MBSE

- model to model transformation (M2M)
- model to text (M2T)

In this course, we condsider Java Emitter Templates (JET) as an example of M2T

Characteristics of M2T:

- No restrictions of output format
- No need to define output syntax
- Flexibility

Main idea: Template

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Dear <Name>,

we are pleased to inform you that you you will be refunded <amount> in income tax.

The reason is that <reason>.

Best regards,
<clerkname>

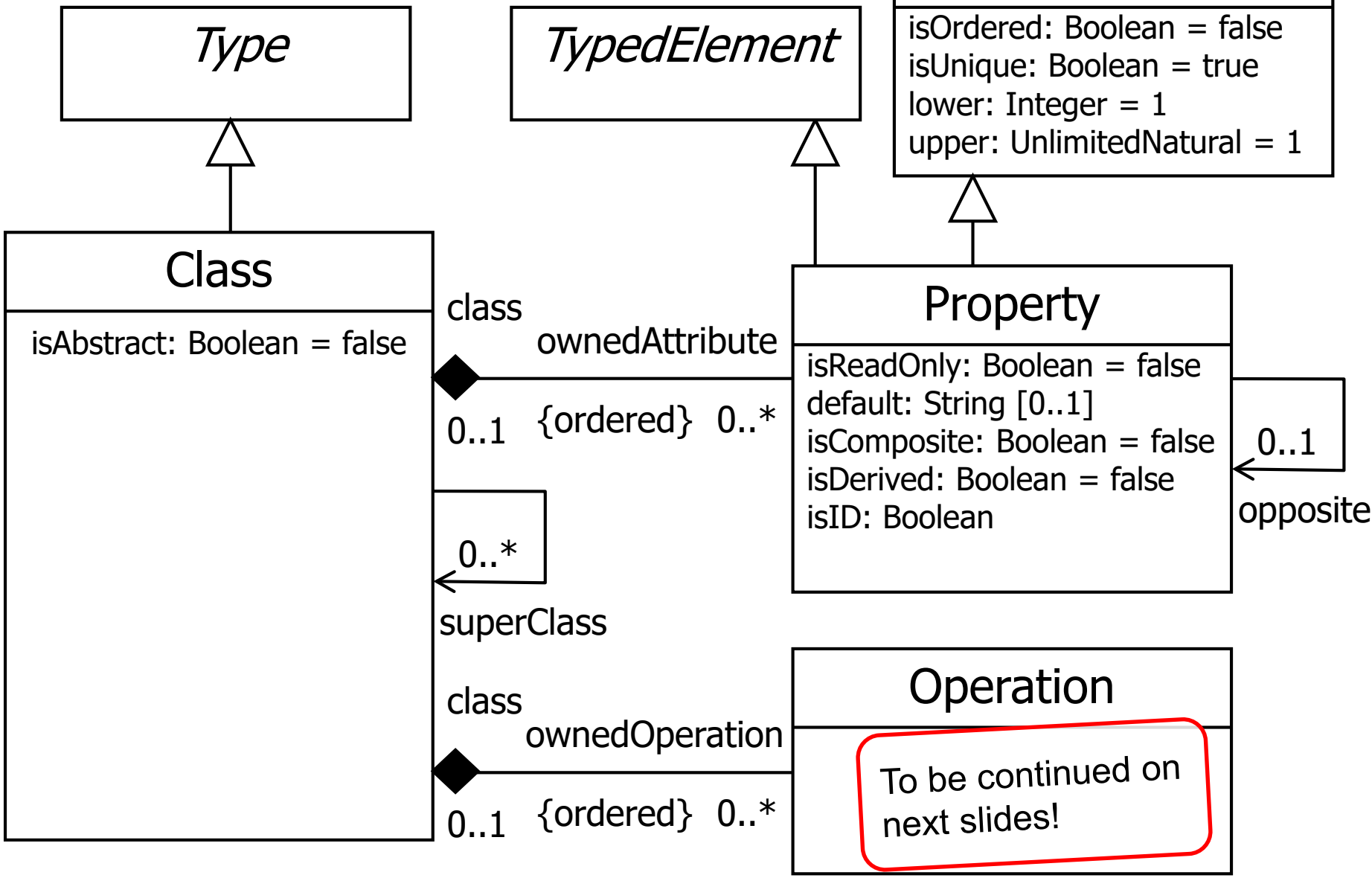
Today, this is much better known from web programming: PHP or JSP, ASP, …

Standard text in which some “specifics” will be filled in (attributes/parameters/fields).

TBC after break

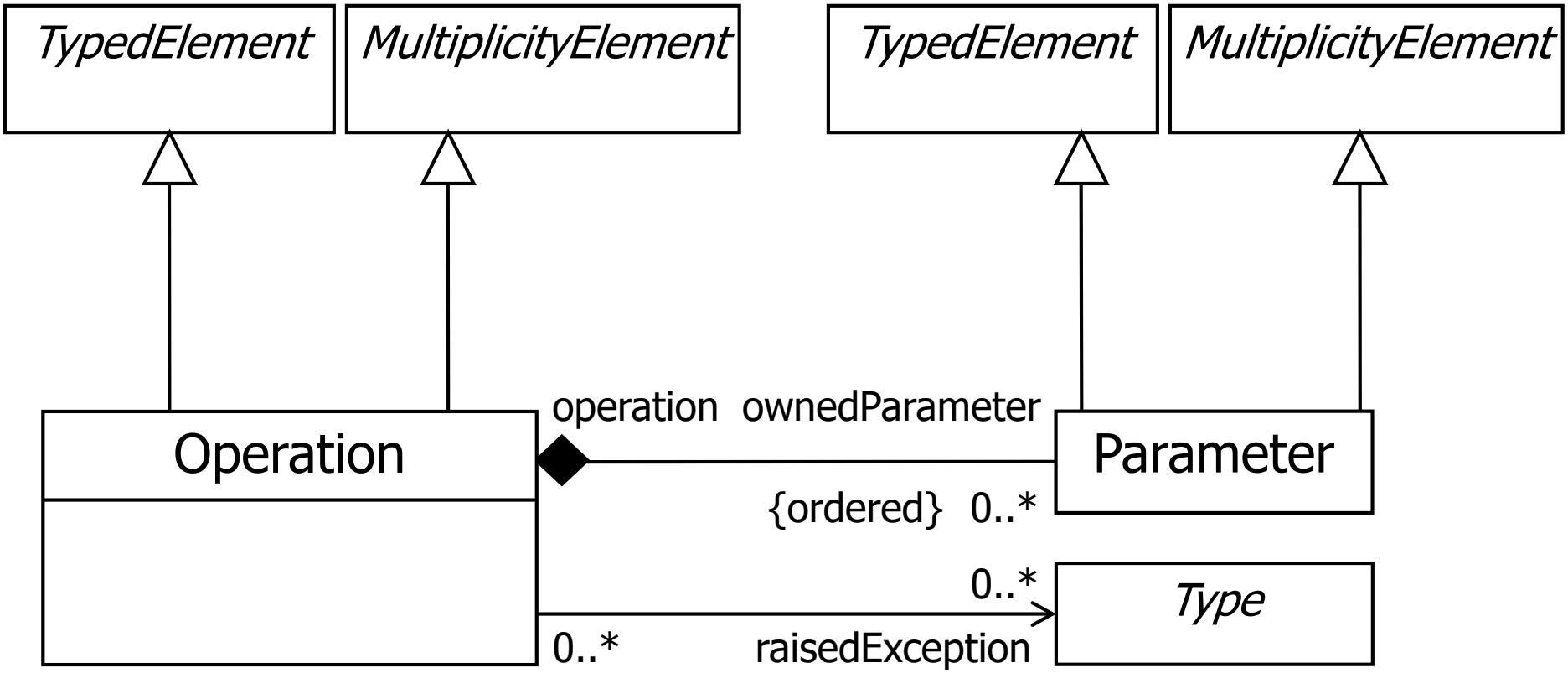
Appendix

EMOF Classes



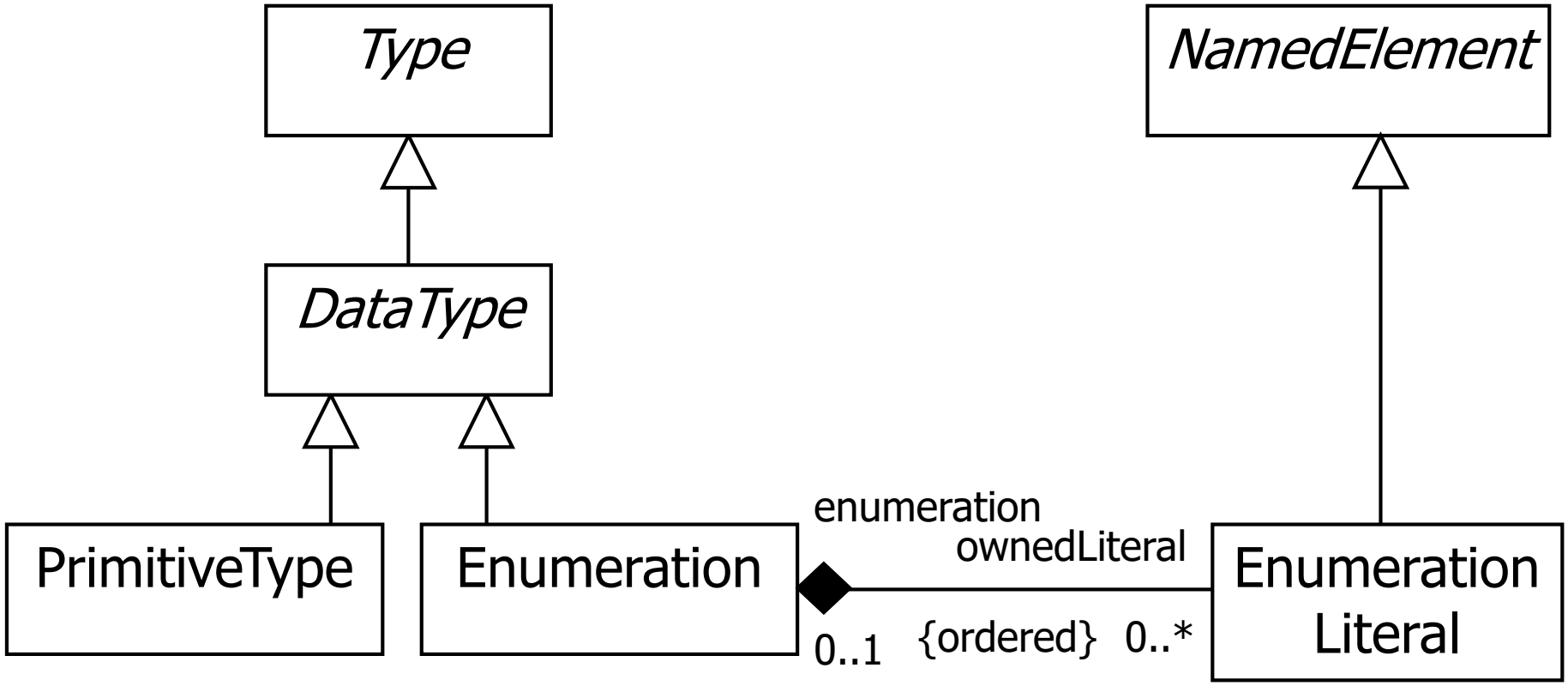
EMOF Classes (cntd.)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



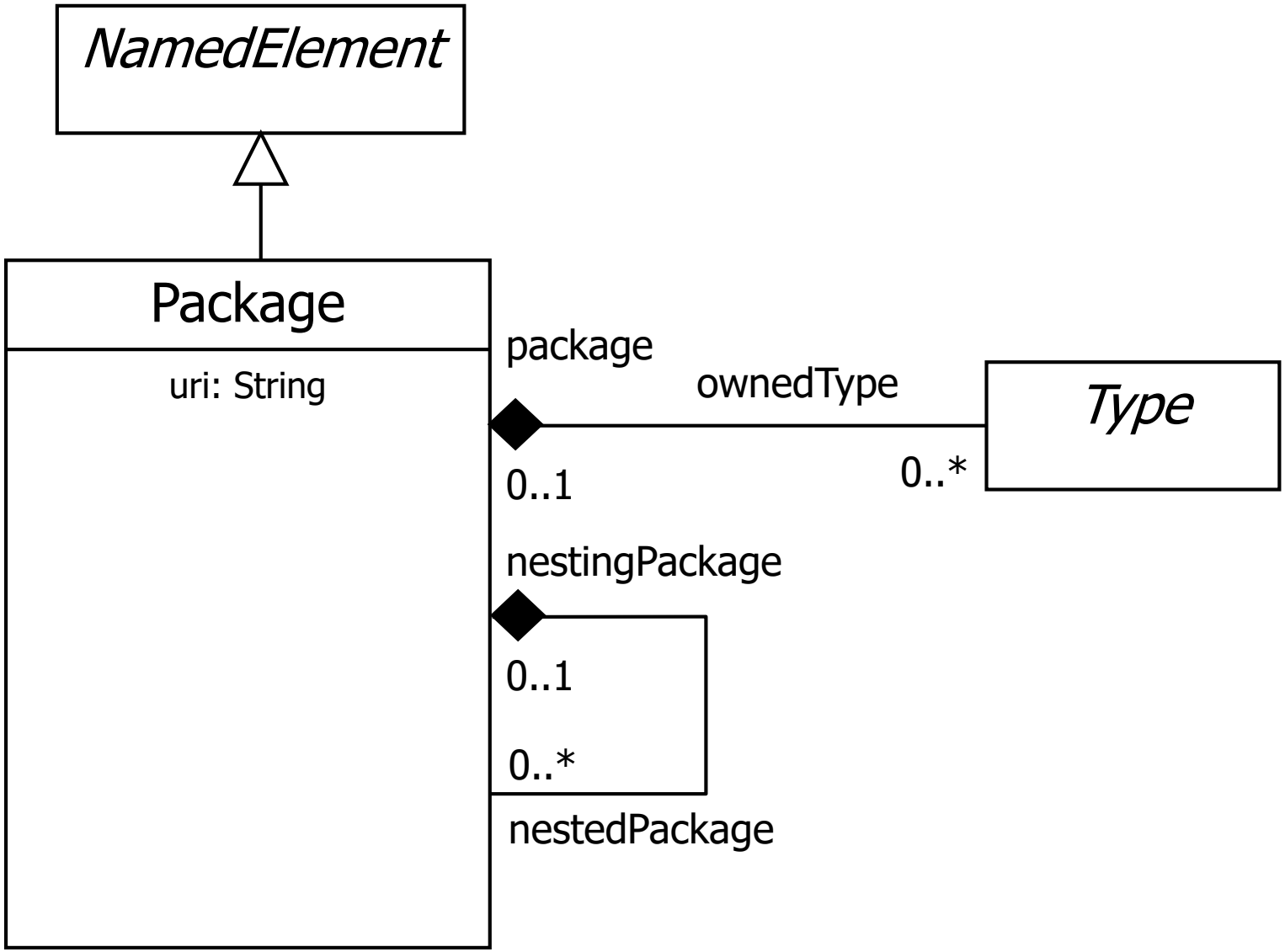
EMOF Data Types

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



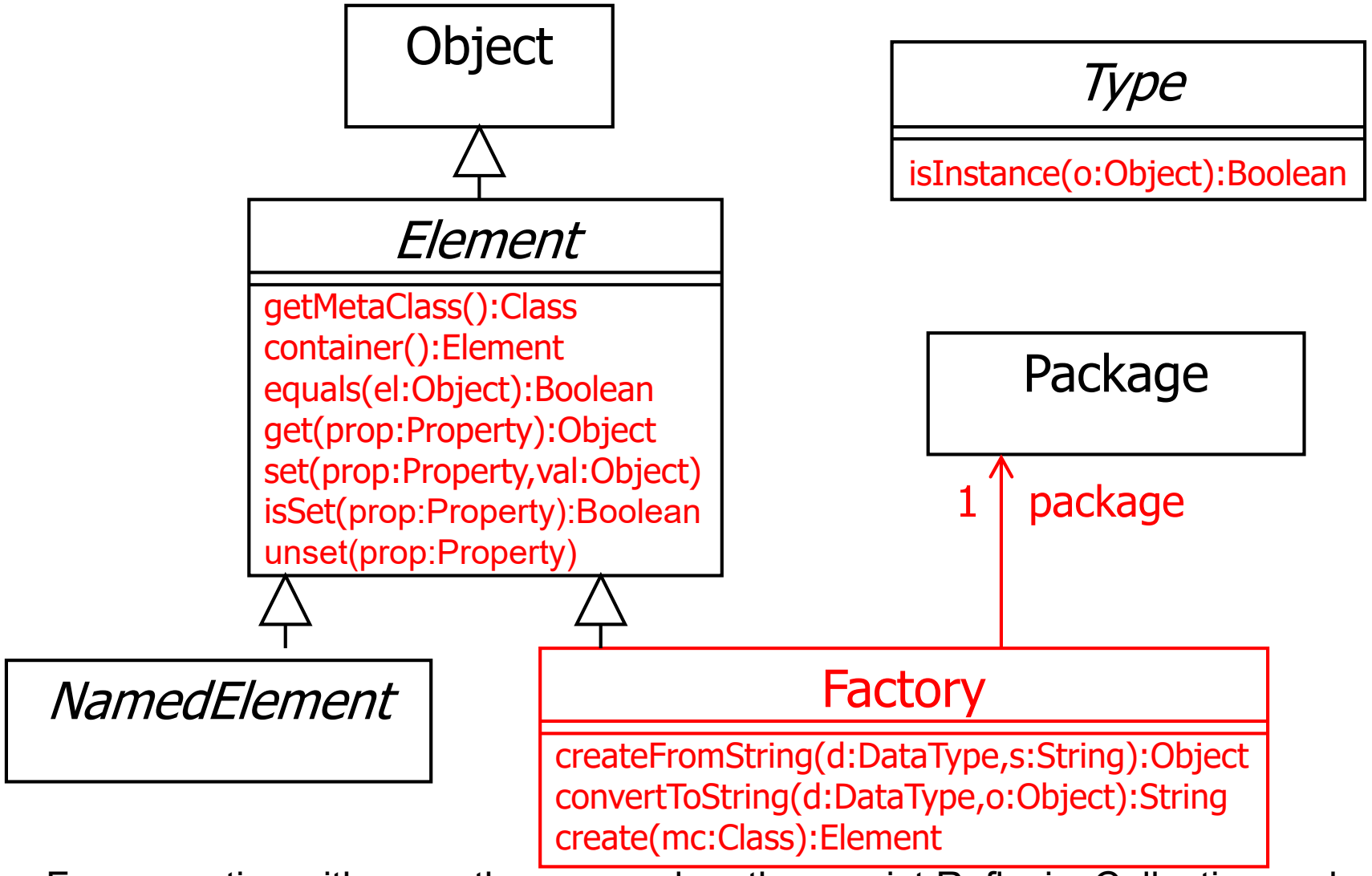
EMOF Packages

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Reflection package

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



For properties with more than one value, there exist ReflexiveCollection and ReflexiveSequence (similar to Java Collections)!