

Advanced Topics in Software Engineering (02265)

Ekkart Kindler

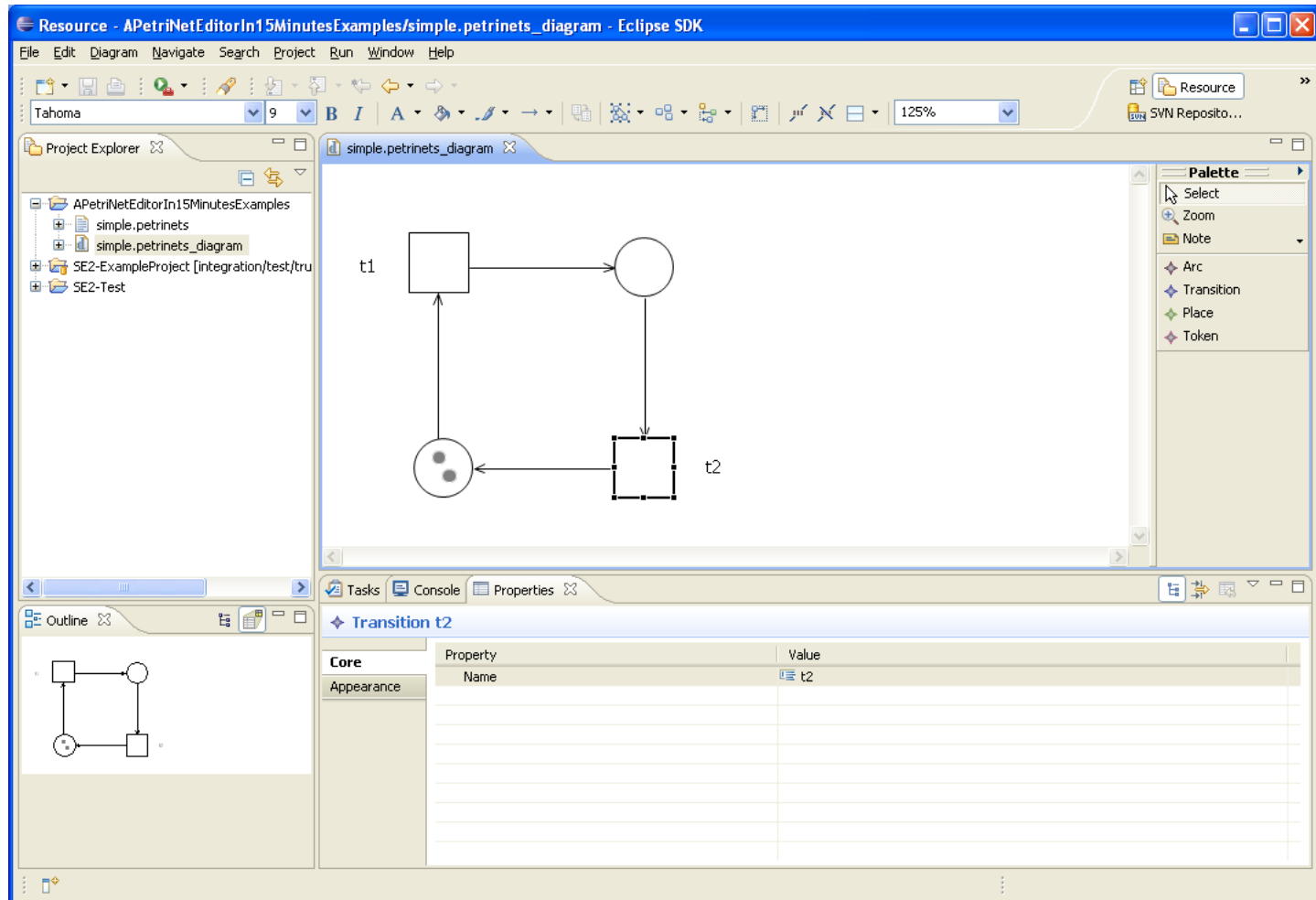
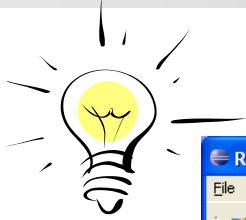
DTU Informatics

Department of Informatics and Mathematical Modeling

All presentations available at:
<http://www2.compute.dtu.dk/courses/02265/f15/schedule.shtml>

I. Introduction and Motivation

1. Vision

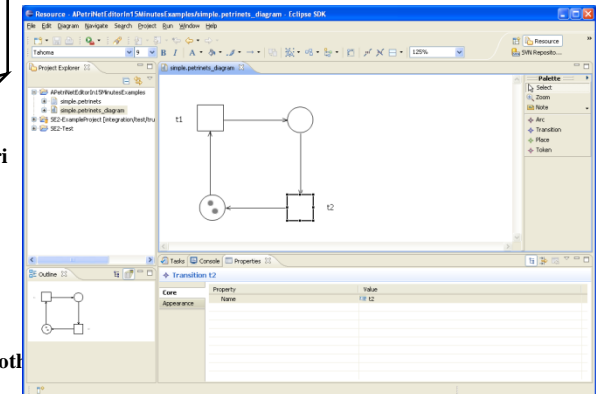
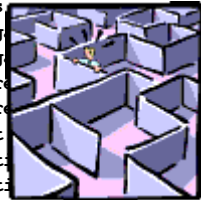


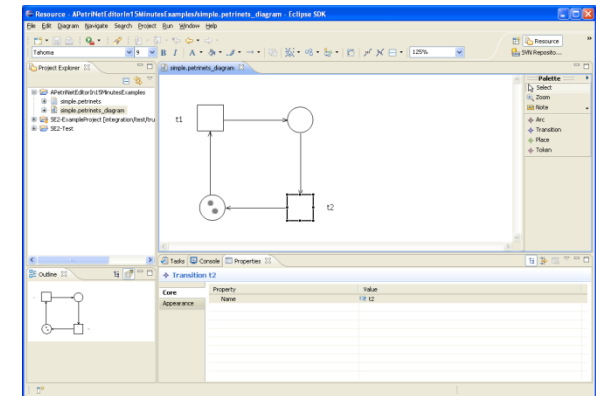
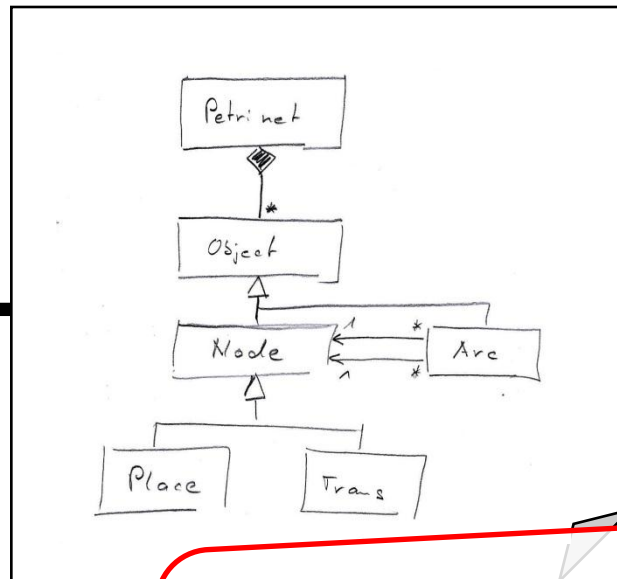
Get rid of
technical
artefacts!

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: %pluginName
Bundle-SymbolicName: APetriNetEditorIn15Minutes.diag
Bundle-Version: 1.0.0
Bundle-Classifier:
Bundle-Localization: plugin
Require-Bundle: org.eclipse.emf.ecore,
org.eclipse.emf.ecore.xmi,
org.eclipse.gmf.runtime.emf.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.editor,
org.eclipse.gmf.runtime.diagram.ui.resources.editor.ide
APetriNetEditorIn15Minutes;visibility:=reexport

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

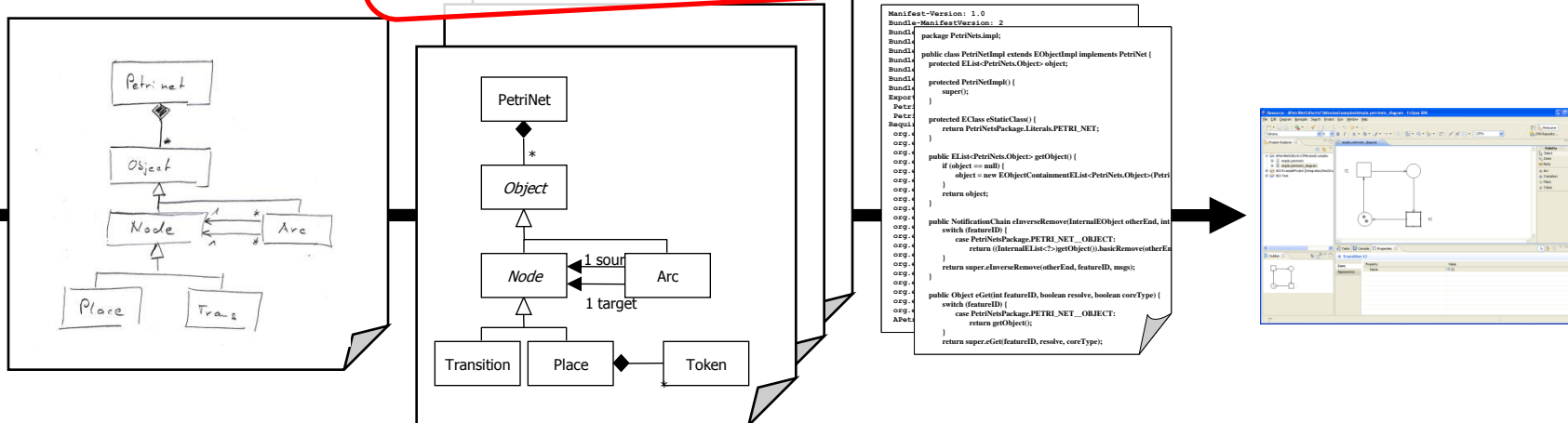
return super.eGet(featureID, resolve, coreType);





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

There are tools that partially support this idea already today (e.g. Eclipse and EMF as used in earlier editions of SE2 (2013 or earlier)).



Analysis

Design

Implementation

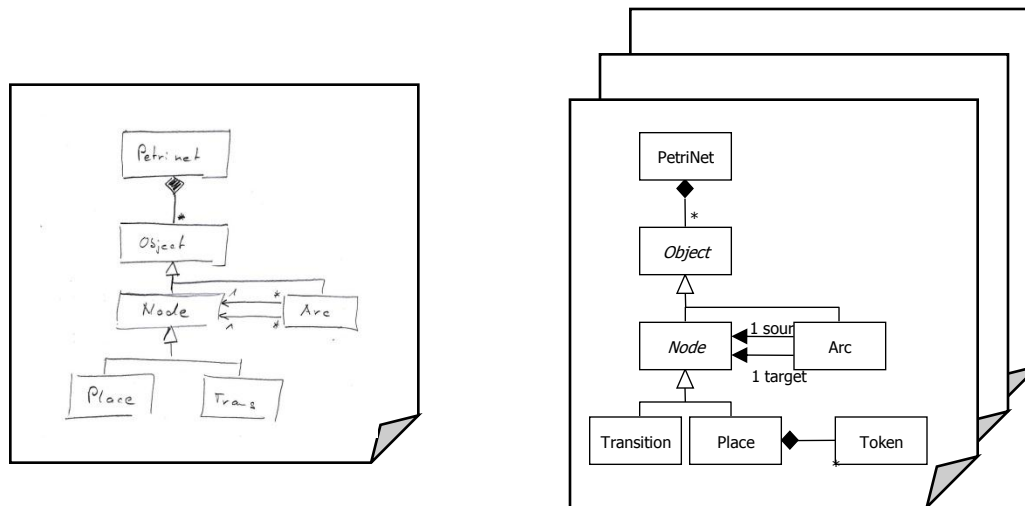
~~Coding~~

Code is generated

- Model Driven Architecture[®] (MDA[®])
OMG[™] software development approach for separating business logic from platform specific details
 - using models
 - 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

Ultimately: Getting rid of programming resp. technical artefacts.

- But, that's UML! We know UML already!
- Didn't we do all that already?
(e.g. in SE1 and SE2, etc.)
- Why do we need another course on that?



2. Goals of this course

- Acquiring a bit more experience!
- Learn what is behind the scenes!
 - Understand concepts and technology
 - Apply (some of) them
 - Experiment and evaluate technology
- Contribute to MBSE?
 - Extend (and develop new) technology
 - Combine them in a new way
 - Formalize and analyze them

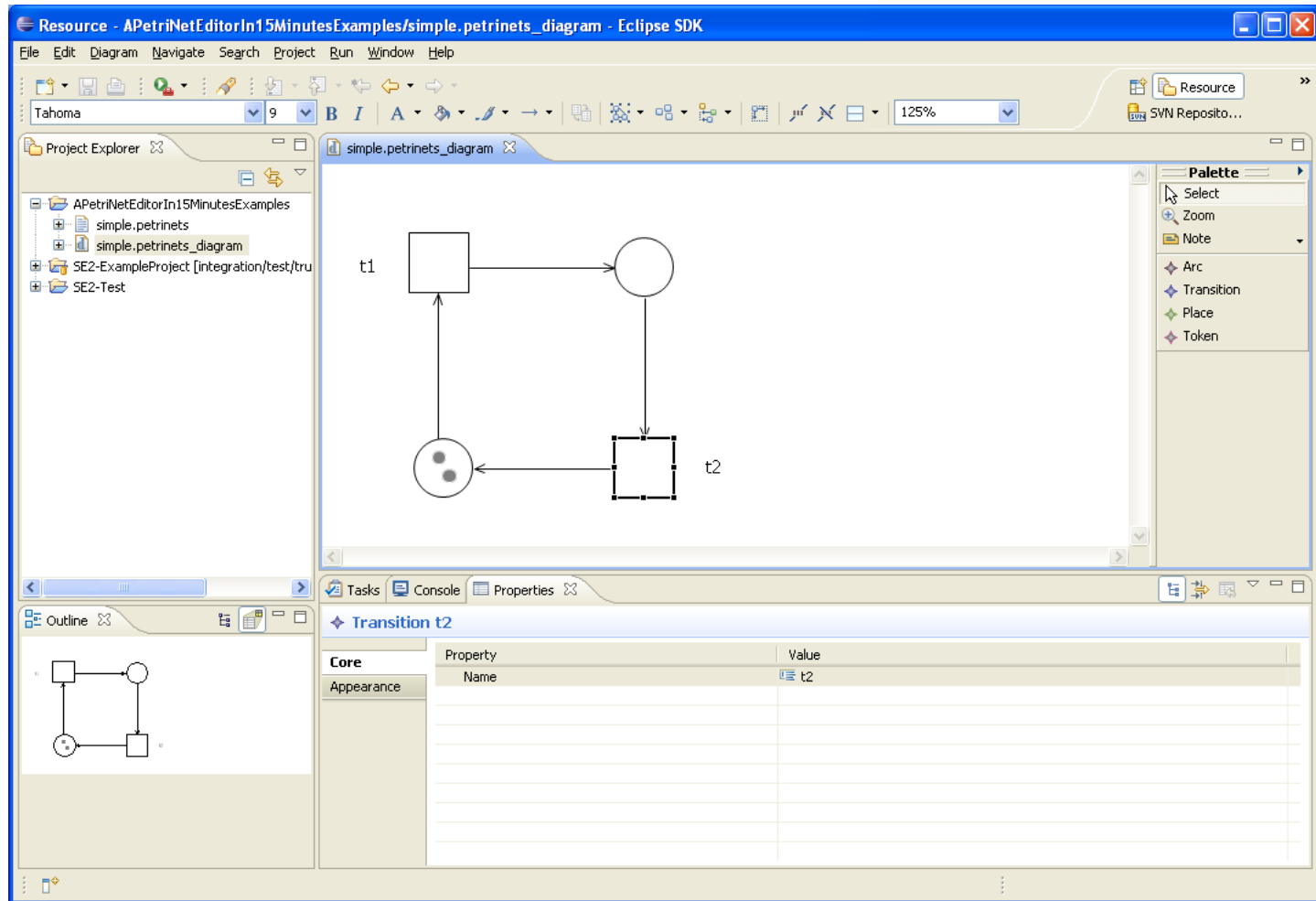
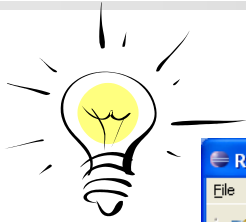
Understand
and work on
the **meta-level**

See <http://en.wikipedia.org/wiki/Meta>
for the meaning and co-notations of
the prefix “meta”. We will come back to
that later.

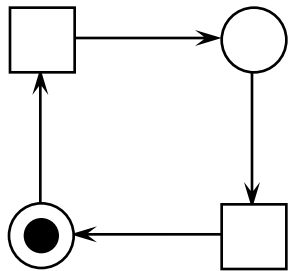
- Modelling notations
 - formalizing, implementing, exchanging them
 - integrating new ones
 - relating different models
 - model notations vs. / w. languages
- Transforming models
 - into other models
 - into text (generate code)
- Synchronizing and merging models
- Analysing and verifying models

- Working with models
- Behaviour models

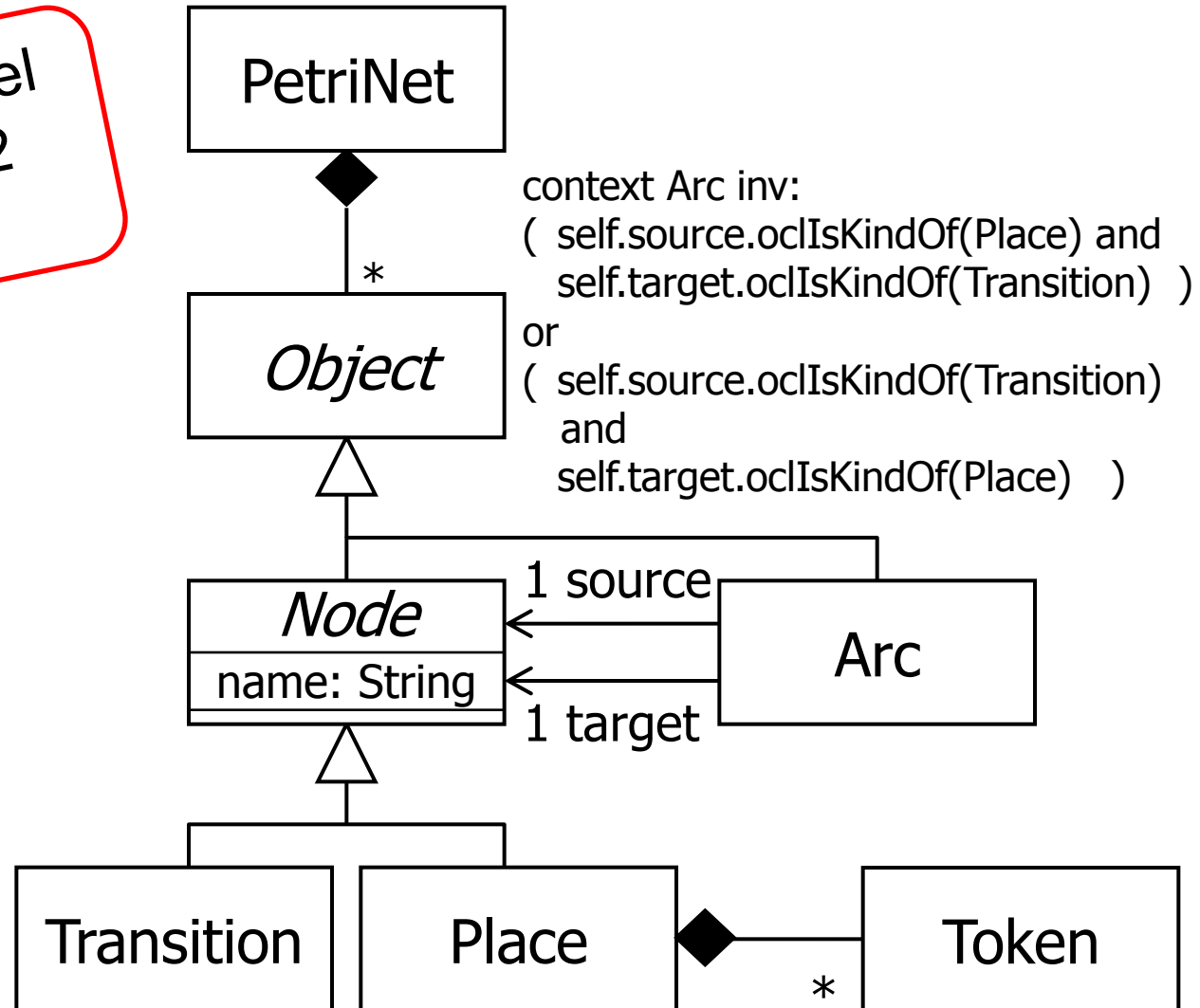
3. Example (a Petri net editor)



→ PNML meta model
ISO/IEC 15909-2
→ ePNK

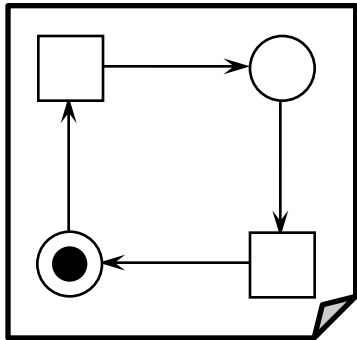


Petri net
model

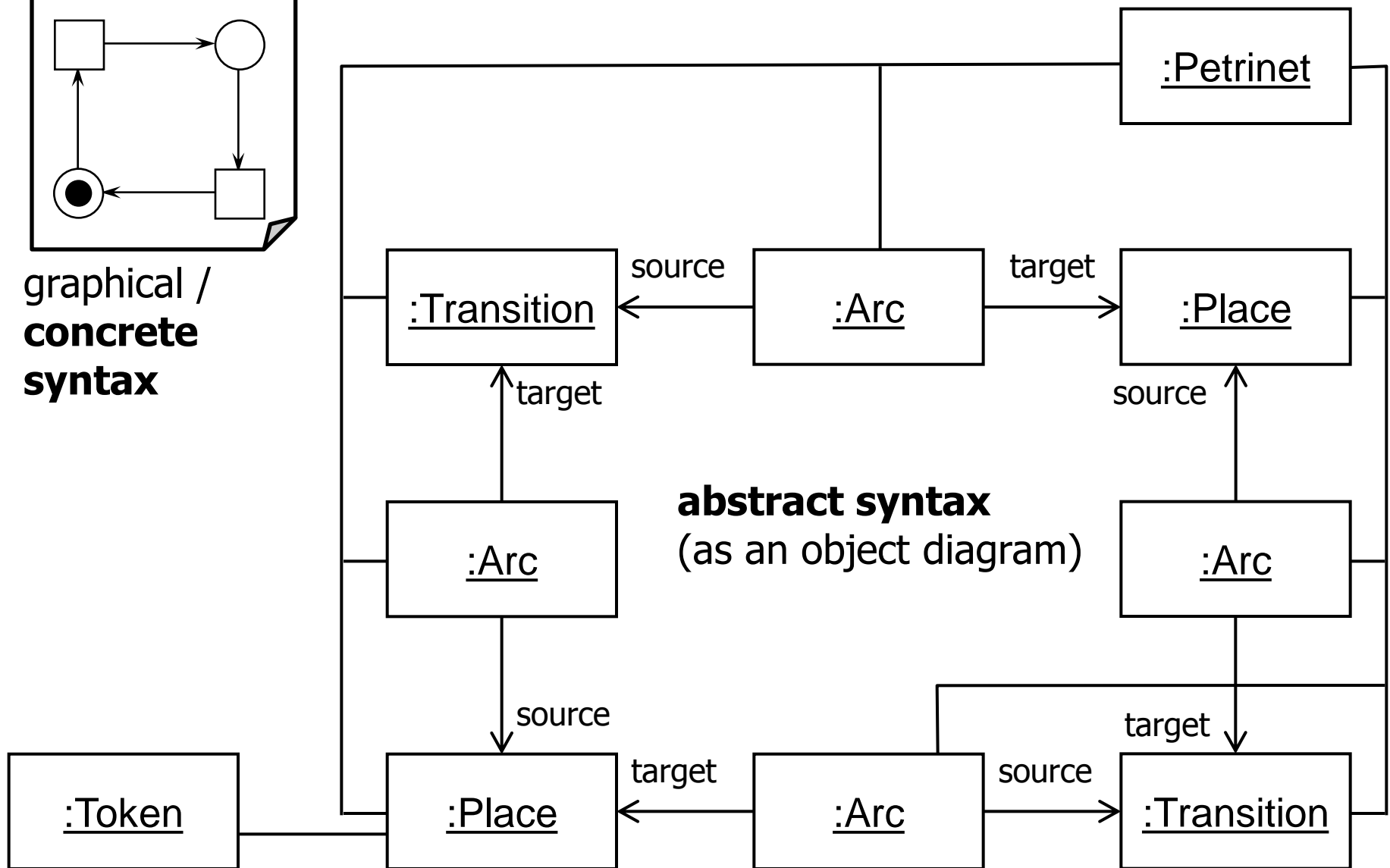


context Arc inv:
(self.source.ocIsKindOf(Place) and
self.target.ocIsKindOf(Transition))
or
(self.source.ocIsKindOf(Transition)
and
self.target.ocIsKindOf(Place))

Meta-model for Petri nets



graphical /
**concrete
syntax**



abstract syntax
(as an object diagram)

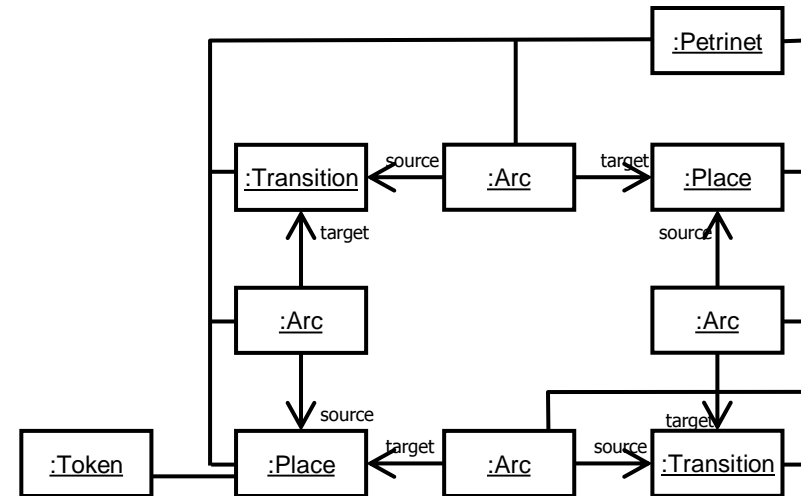
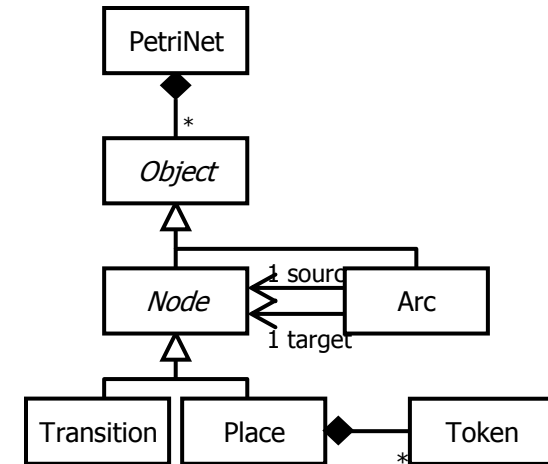
meta model

build-time

is an
instance of

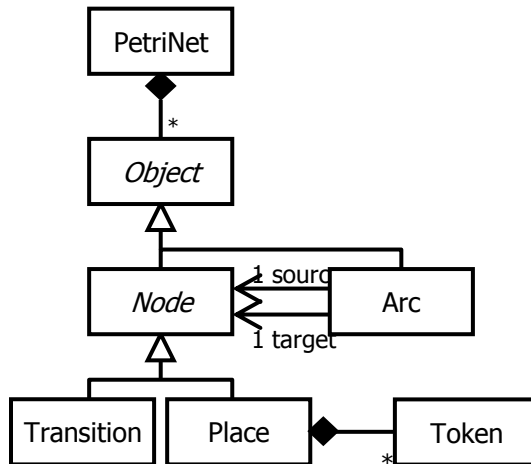
model

runtime

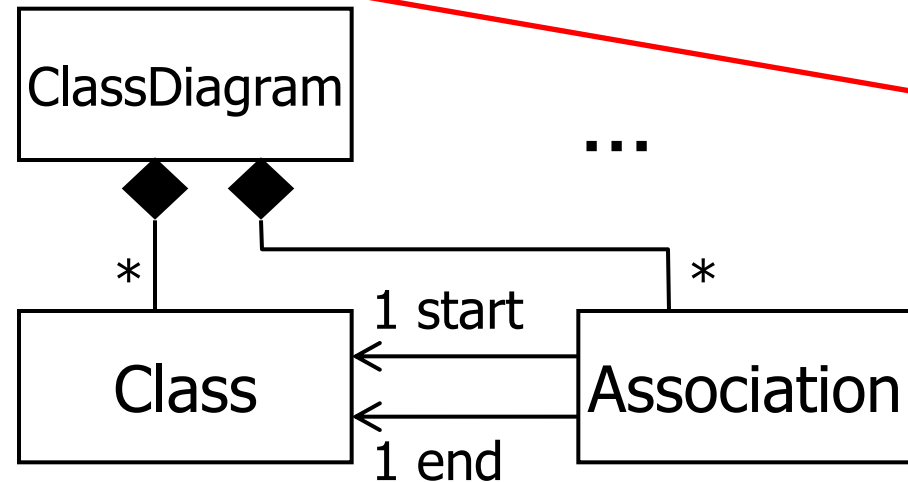


- Better understanding
- Mapping of instances to XML syntax (XMI)
- Automatic code generation
 - API for creating, deleting and modifying model
 - Methods for loading and saving models (in XMI)
 - Standard mechanisms for keeping track of changes (observers)

Note: We will see later that this is **much more** involved!



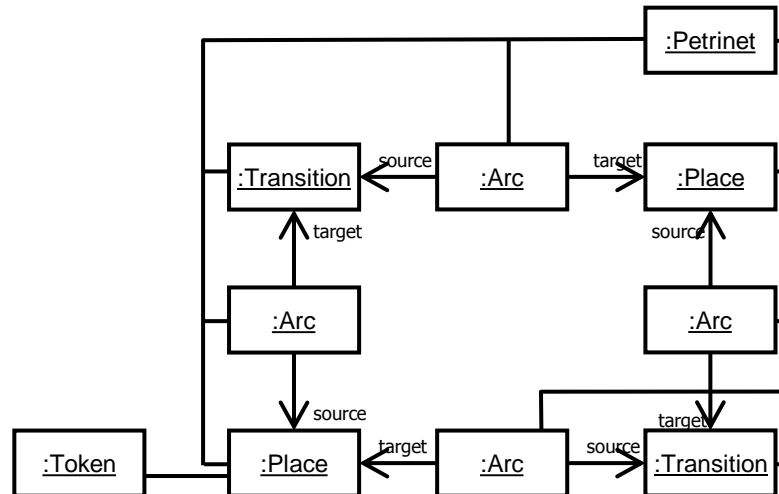
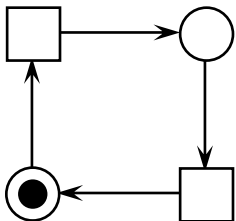
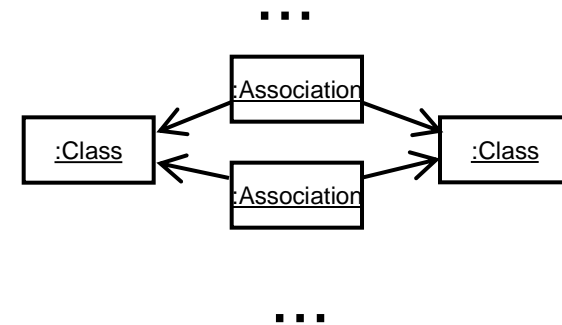
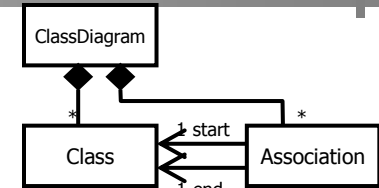
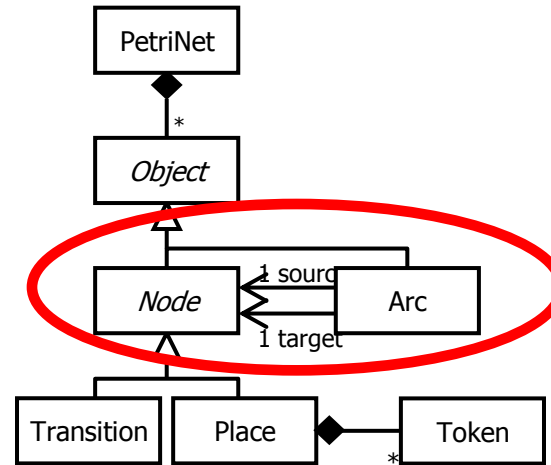
UML model

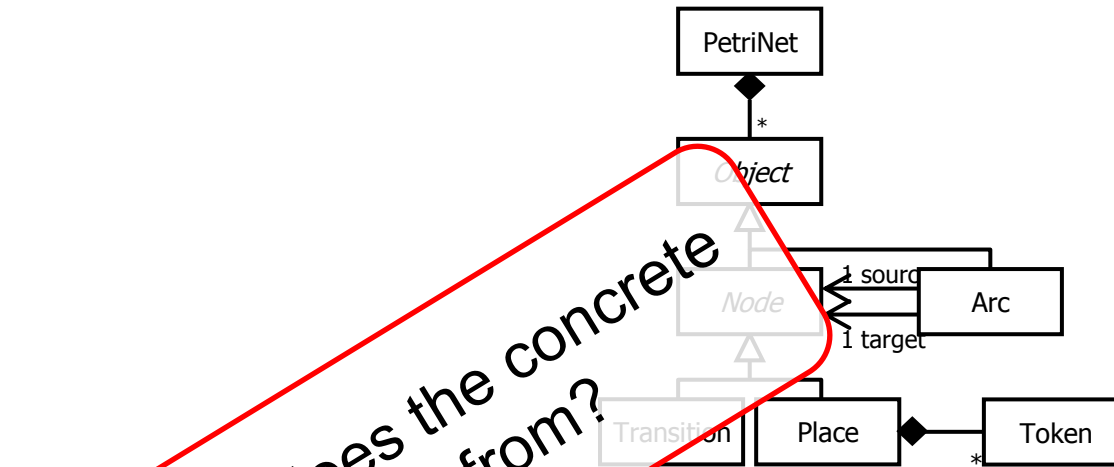
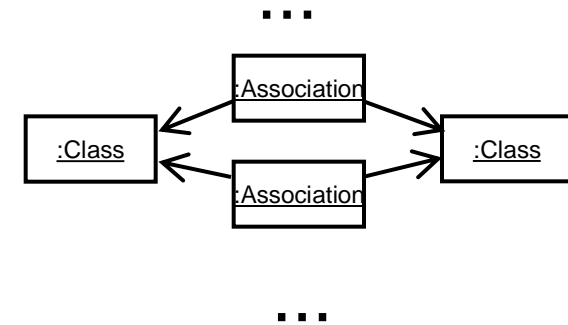
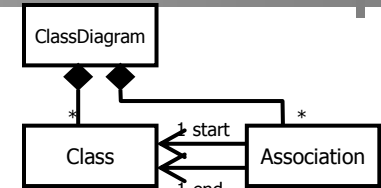


Meta-model for UML
(class diagrams)

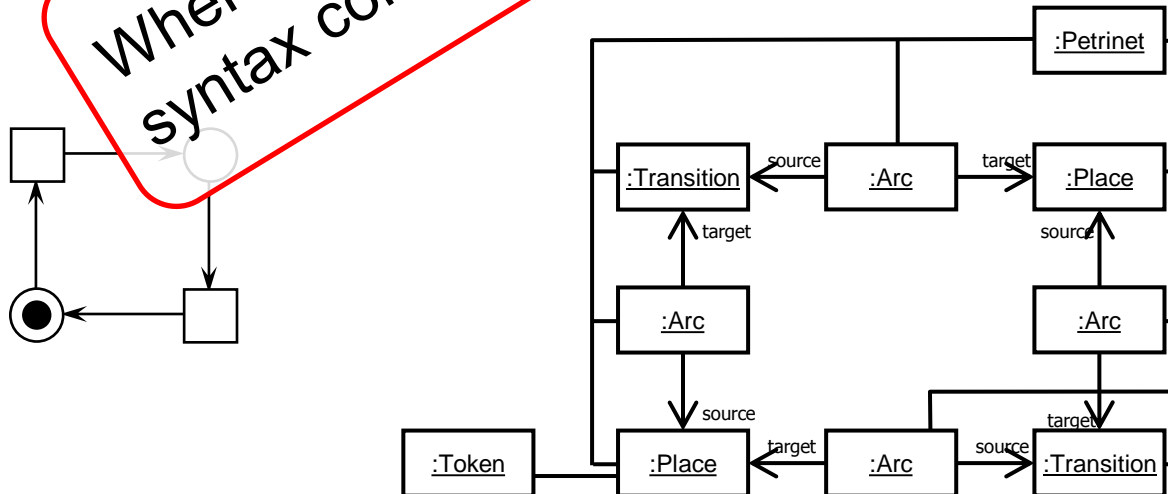
Now, the term “meta”
model makes sense!

We come back to that, when we learn more about the Meta Object Facility (MOF)

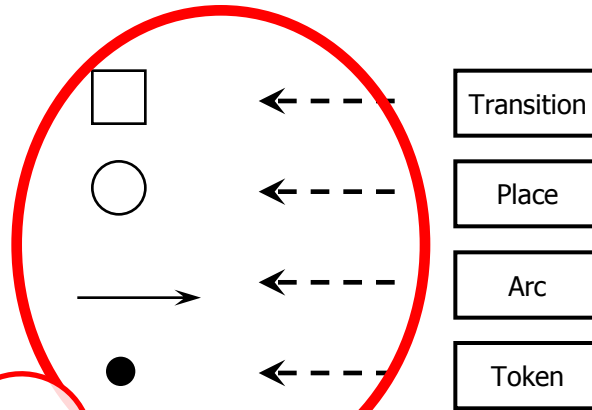




Where does the concrete syntax come from?

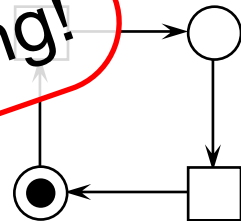


meta model

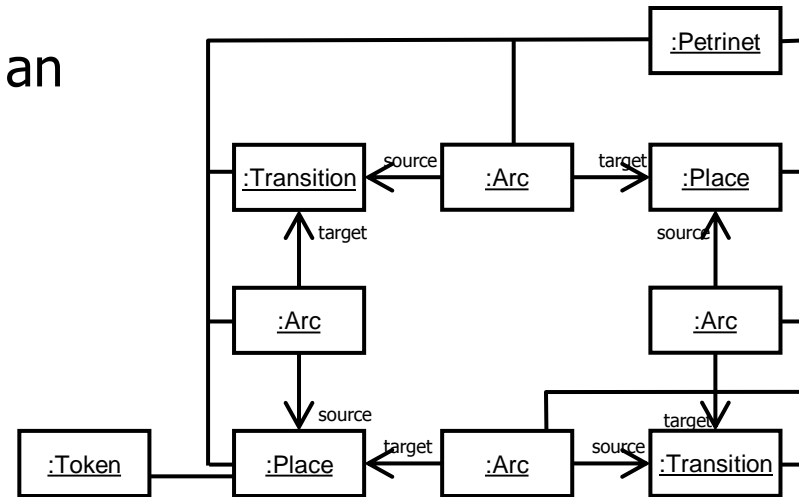
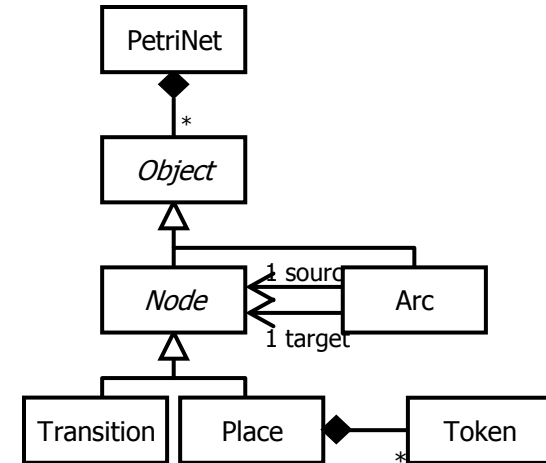


is instance of
A Petri net editor
in 15 minutes!
GMF: Not kidding!

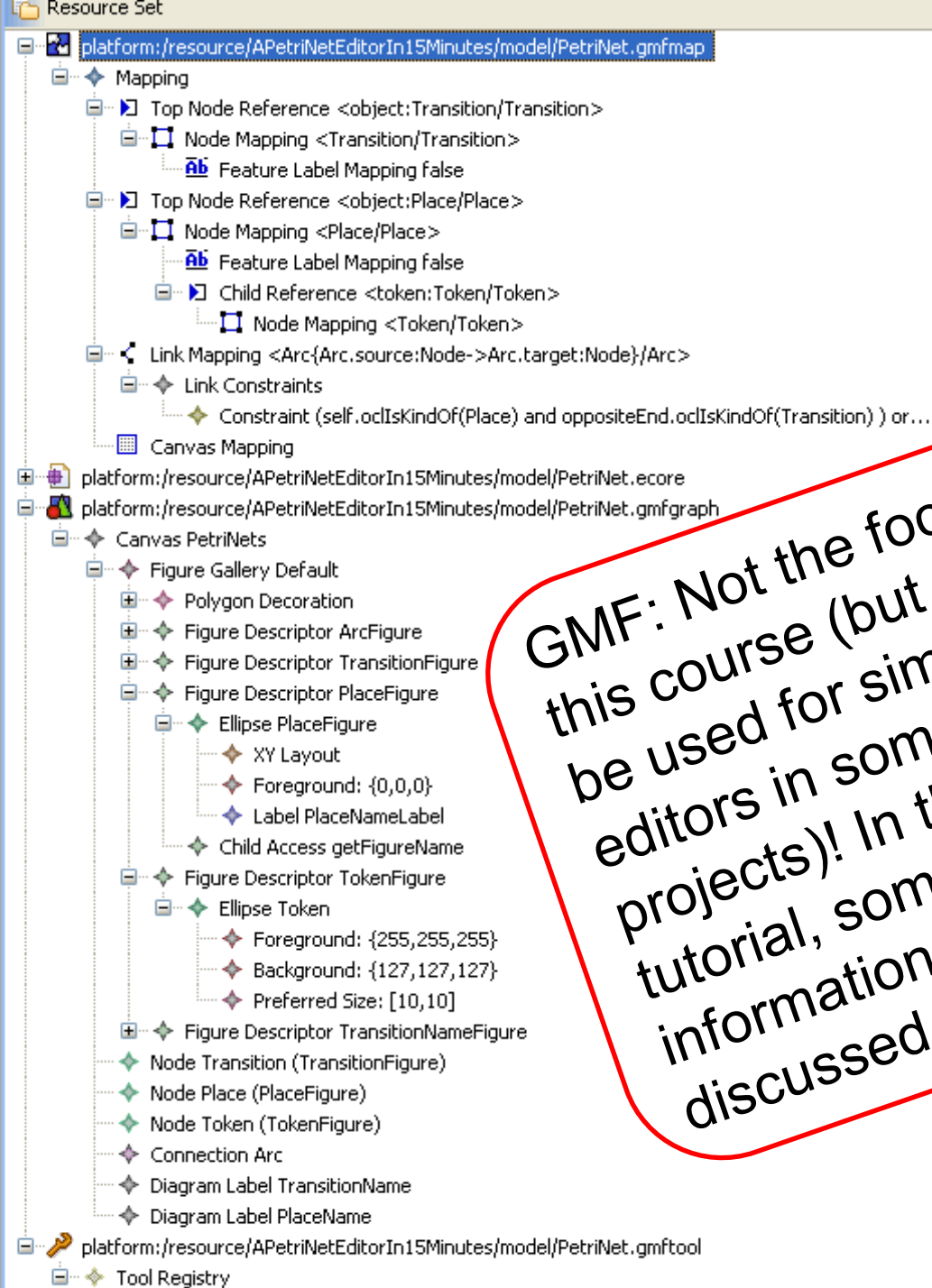
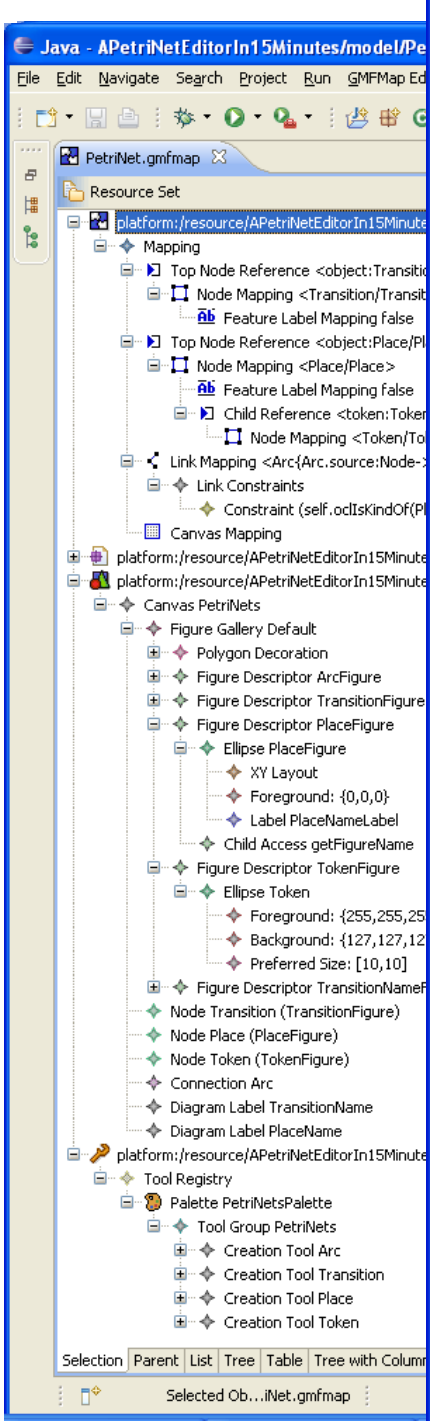
generate an editor



concrete syntax



abstract syntax



GMF: Not the focus of this course (but might be used for simple editors in some projects)! In the tutorial, some of this information will be discussed.

- 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

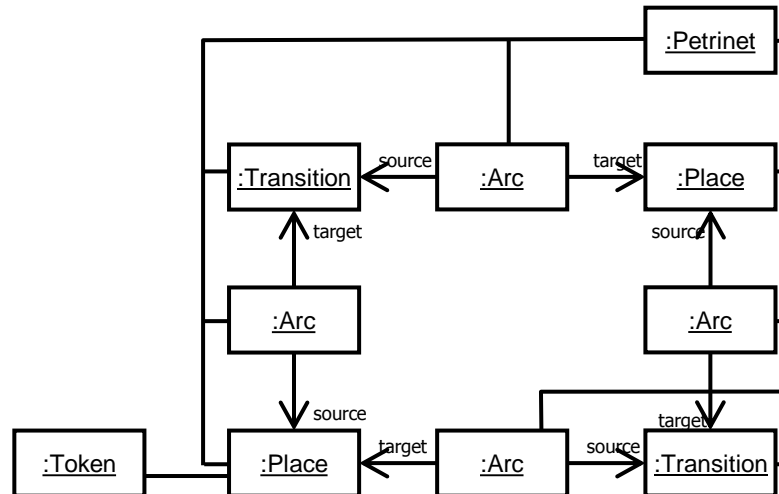
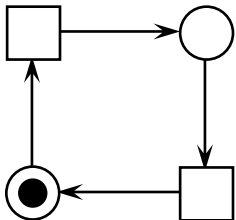
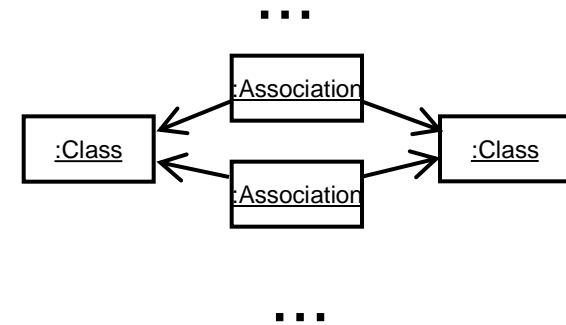
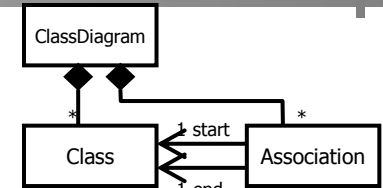
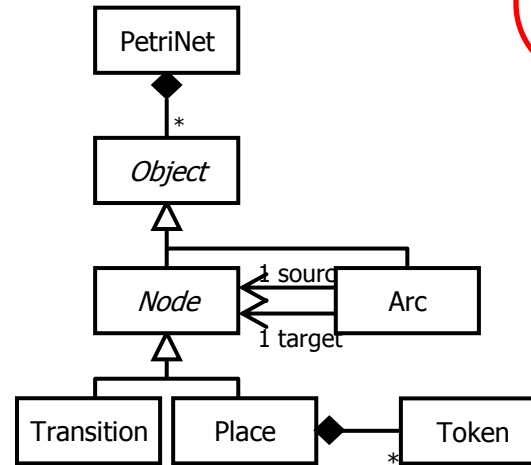
How about “real” functionality /
behaviour?
→ later in this course!

■ **Abstraction**

- Focus
- Simplification
- Separation
- Understanding
- Communication
- Analysis
- Execution (interpretation / code generation)

But isn't code just
another form of model?

Our focus in ATSE:



4. Basic terminology

- Concept
- Formalism
- Method / methodology
- Model / meta-model
- Notation
- Principle
- Technique
- Technology
- Tool
- Software engineering
- Taxonomy
- Ontology
- Framework
- Approach

Modell [*lat.-vulgärlat.-it.*] *das; -s, -e*:

...

7. die vereinfachte Darstellung der Funktion eines Gegenstands od. des Ablaufs eines Sachverhalts, die eine Untersuchung od. Erforschung erleichtert od. erst möglich macht.

...

[nach Duden: Das Fremdwörterbuch, 1990].

Model [*lat.-vulgärlat.-it.*]

...

7. the simplified description of the function, purpose, or process of something; it enables us investigating and analysing this thing.

...

[translated from Duden: Das Fremdwörterbuch, 1990].

Technology Technology is a term with origins in the Greek "*technologia*", "*τεχνολογία*" — "*techne*", "*τέχνη*" ("craft") and "*logia*", "*λογία*" ("saying"). However, a strict definition is elusive; "technology" can refer to material objects of use to humanity, such as machines, hardware or utensils, but can also encompass broader themes, including systems, methods of organization, and techniques. The term can either be applied generally or to specific areas: examples include "construction technology", "medical technology", or "state-of-the-art technology".

From: <http://en.wikipedia.org/wiki/Technology>

A technique is a procedure used to accomplish a specific activity or task:

- Technology, the study of or a collection of techniques
- Skill, the ability to perform a task
- Scientific technique, any systematic method to obtain information of a scientific nature

From: <http://en.wikipedia.org/wiki/Technique>

Principle

Etymology: Middle English, from Middle French *principe*, *principle*, from Old French, from Latin *principium* beginning, from *princip-*, *princeps* initiator — more at prince

Date: 14th century

- 1 a: a comprehensive and fundamental law, doctrine, or assumption
b (1): a rule or code of conduct (2): habitual devotion to right principles <a man of *principle*> c: the laws or facts of nature underlying the working of an artificial device

2 ...

From: <http://www.merriam-webster.com/dictionary/Principle>

A **concept** (abstract term: "**conception**") is an cognitive unit of *meaning*— an abstract idea or a mental symbol sometimes defined as a "unit of knowledge," built from other units which act as a concept's characteristics. A concept is typically associated with a corresponding representation in a language or symbology such as a word.

From: <http://en.wikipedia.org/wiki/Concept>

method

Etymology: Middle English, prescribed treatment, from Latin *methodus*, from Greek *methodos*, from *meta-* + *hodos* way

Date: 15th century

- 1: a procedure or process for attaining an object: as a
(1): a systematic procedure, technique, or mode of inquiry employed by or proper to a particular discipline or art
(2): a systematic plan followed in presenting material for instruction
b (1): a way, technique, or process of or for doing something (2): a body of skills or techniques
- 2: a discipline that deals with the principles and techniques of scientific inquiry
- 3 a: orderly arrangement, development, or classification : plan b: the habitual practice of orderliness and regularity

From: <http://www.merriam-webster.com/dictionary/Method>

methodology

- a set of methods for achieving something (typically along with some instructions on when and how to use them, and the rationale behind them)

Generally “X-ologies” comprise a body of knowledge and science on “X” (comes from Greek “logos”).

Often abused to make things to “sound more important”!

notation

Etymology: Latin *notation-*, *notatio*, from *notare* to note

Date: 1584

1: annotation , note

2 a: the act, process, method, or an instance of representing by a system or set of marks, signs, figures, or characters

b: a system of characters, symbols, or abbreviated expressions used in an art or science or in mathematics or logic to express technical facts or quantities

We call a notation a formalism, if it has a more or less precise meaning!

From: <http://www.merriam-webster.com/dictionary/notation>

- Concept
- Formalism
- Method / methodology
- Model / meta-model
- Notation
- Principle
- Technique
- Technology
- Tool
- Software engineering
- Taxonomy
- Ontology
- Framework
- Approach

More specific terminology explained throughout this course, e.g.:
MBSE, MDA, DSL, Meta-modelling,
M2M, M2T, ...

A method?

A methodology?

A principle?

A notation?

A formalism?

A concept?

A technique?

A technology?

is the sum of all means, facilities, procedures, processes, notations, methods, concepts and principles for developing, operating and maintaining a software system.

In this course, the focus is on the software development and technology for that purpose.

A method?

A methodology?

A principle?

A notation?

A formalism?

A concept?

A technique?

A technology?

ATSE in a nutshell

- Advanced SE technologies
(principles, concepts, methods, notations, tools, ...)
- and how to implement tool support for software development (and concepts and technologies for that purpose)

Projects implement a part of such a tool!

DSL:
Domain Specific Language

- DSLs and "its" flavours
- Meta-modelling
 - MOF (eMOF)
 - XMI
- Notations
 - OCL
 - ...
- Using models
 - Frameworks (ex. EMF+)
- Formalisation
 - Mathematical models
 - Model checking
 - Graph grammars
- MDA / MBSE
 - Idea and principles
 - Code generation (JET, ...)
 - Transformations (QVT, TGGs)
 - Behaviour modelling
 - "Textmodels" (ASTs, Xtext?)
- Further ideas
 - Merging and "diffing" models
 - Aspect oriented modelling
- IDE-Integration
- Behaviour modelling
- ...

- Lecture part:
 - Concepts and underlying theory of Model-based Software Engineering (with focus on the meta-level)
 - Relation between the concepts and rationale behind them
- Tutorial part:
 - Use of basic technology (Eclipse/EMF/ePNK/ECNO Tool)
 - Practical application of (some of the) concepts and techniques for small examples
- Project:
 - A simple tool for some aspect of software development (which requires to use a combination of some concepts of this course)
 - In groups of 2-4 students
 - There are different predefined topics from which the groups may chose (see other presentation);

Weekly Schedule

	Mon	Tue	Wed	Thu	Fri
8-10	<div>Plus actual work on tutorials an the project!!</div>				
10-12					
13-15			lecture		
15-17			tutorials / project		

There will some exceptions (check web pages regularly) :
<http://www2.compute.dtu.dk/courses/02265/f15/schedule.shtml>

Bookmark this page!!!

DTU Compute
Department of Applied Mathematics and Computer Science

02265: COURSE ON ADVANCED TOPICS IN SOFTWARE ENGINEERING

This is a preliminary schedule for the course Advanced Topics in Software Engineering (02265), material in spring 2015. The first lecture starts on Wednesday, Feb. 4 at 13.00 (sharp) in room 303A/44.

The plan also indicates, at which times there will be lectures (L:), tutorials (T:), and work on or projects (P:). For the lectures, the material will be made available within the timetable below respectively. The assignments will be available one week before the respective tutorial (and then discussed).

Please note that this is a preliminary schedule and the exact time slots for lectures and tutorial. In the end of the semester, the tutorial slot will be used for working on the project.

Note: For your convenience, you will find the links to all material that is made available here also on this page, that is organized chronologically: <http://www2.compute.dtu.dk/courses/02265/f15/updates.shtml>

Week	Wed 13.00-15.00 303A/44	Wed 15.00-17.00 302A/44	Deliverables (project)	Comments
wk 6 4. 2.	L: Introduction	P: The projects: a first overview T: Getting started with Eclipse and EMG/GMF (Assignment 1)	If not stated otherwise, the deadline in the respective week is Wednesday at 24:00	
wk 7 11. 2.	L: Model-based Software Engineering and Domain Specific Languages	P: More details and discussion of projects and the ePNK and ECNO T: Working with the ePNK ("flattening" a net)	Feb. 11: CN submission: from every group, a list of participants (2-4)	
wk 8 18. 2.	L: Meta-modelling	P: Project discussion and the ECNO Tool, Q & A (have another look at the projects, so that you can ask questions on the projects) T: ECNO (programming a GUI)	Feb. 18: CN submission: from every group, which project they chose	
wk 9 25. 2.	L: Modelling framework & OCL	P: Q & A T: EMF programming (needed advanced concepts)	Feb. 25: CN submission: from every group, a short first project description (1-2 pages) in PDF.	

ATSE: Tutorial 1

DTU Compute
Department of Applied Mathematics and Computer Science

02265: ADVANCED TOPICS IN SOFTWARE ENGINEERING (F15)

Assignment 1 (due wk 7)

Note that you do not need to submit these assignments! You should have the solution to the assignments ready in the indicated week, so that details and problems can be discussed in that week. The official submissions concern the project deliverables only.

In this assignment, you will become acquainted with the most relevant parts of Eclipse, EMF and GMF. To this end, you will install Eclipse and the Petri net editor example from the lecture, and you will slightly extend the Petri net example.

The details are explained in the following sub-tasks step by step:

1. Install Eclipse (version 4.4, Luna) on your computer with the necessary MBSE extensions. You will find the information on how to do that at the Eclipse installation page: <http://www2.compute.dtu.dk/courses/02265/f15/project/eclipse-installation.shtml>.
2. Download the Petri net editor example from [here](#) and import the two projects into your workspace by selecting "File -> Import... -> General -> Existing projects into workspace", and select the archive file (make sure to choose "select archive file") in wizard that popped up.

Now, you will find two projects in your workspace. Ignore the errors in project "APetriNetEditorIn15Minutes.simulator" for a moment (the reason is that this project has some dependencies to projects that are not yet generated).

In order to resolve these problems, generate the projects ".edit" and ".diagram": The ".edit" project can be generated once you opened the file "PetriNet.gemodel" in the models directory (by double-clicking on it). Press the right mouse button on the top-level Petri net element and choose "Generate edit code". For generating the diagram code, right-click on the file "PetriNet.gmfgen" and choose "generate diagram code".

The errors in the project "APetriNetEditorIn15Minutes.simulator" should be gone now.

Now you can start the run-time workbench by

- a. Select "Run -> Run Configurations..."
- b. Right-click "Eclipse Applications" and select "New"
- c. Enter a name like "Petri Net Example", change the location to "\${workspace_loc}/../runtime-ws-atse", and press "Run"
- d. After a while, a new Eclipse instance will start with an empty workspace (this is called the "runtime workbench", since it runs Eclipse including the projects you developed in the "original" workbench).

In the runtime workbench started above, create a new empty project and create and edit a new Petri net

- Teacher: Ekkart Kindler
 - Email: ekki@dtu.dk
 - Office: 303B.060
 - Consultation hours: whenever
(after the lecture/tutorial, just drop in, call, or email me)
- Material:
 - <http://www2.compute.dtu.dk/courses/02265/f15/>
(in particular check for schedule, and assignments on the web)