

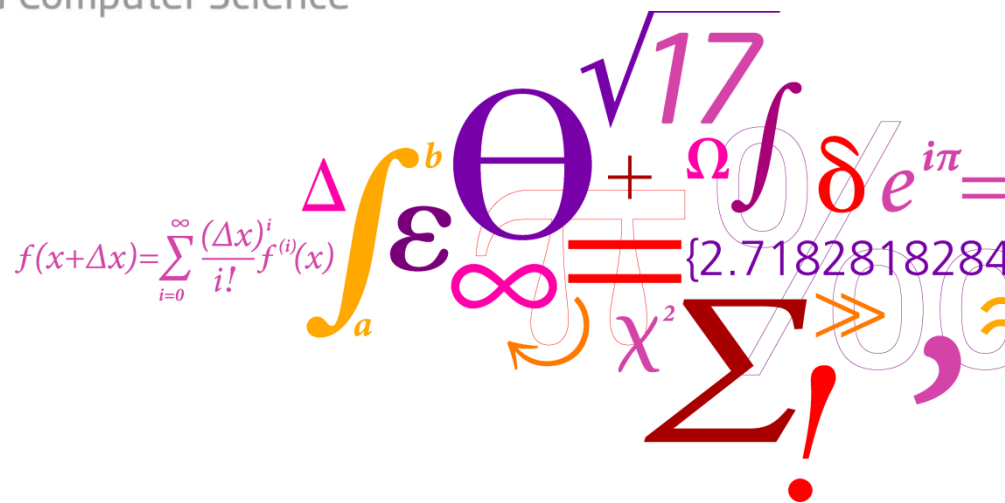
Model-based Software Engineering

(02341, spring 2016)

Ekkart Kindler

DTU Compute

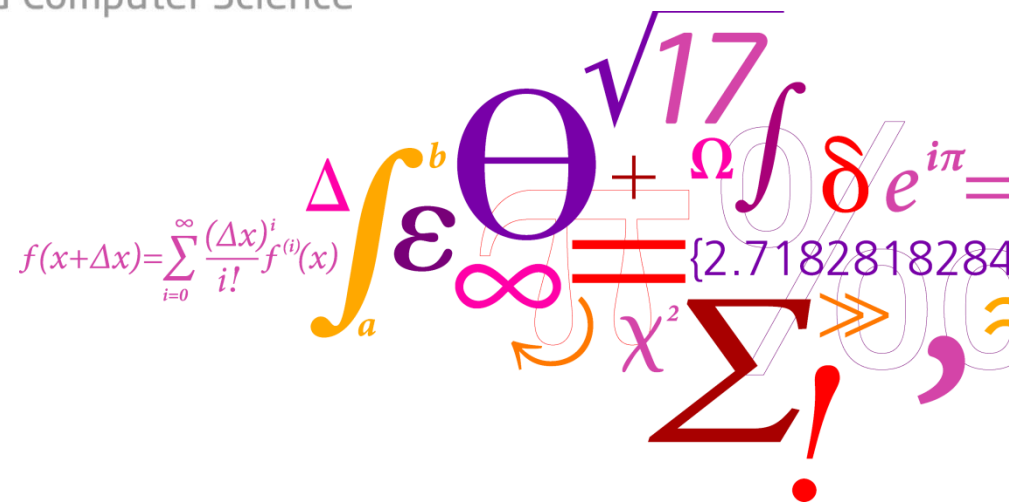
Department of Applied Mathematics and Computer Science



I. Introduction

DTU Compute

Department of Applied Mathematics and Computer Science



- What is “software engineering”?
- What is “software”?
- software ~~=~~ program
- software engineering ~~=~~ programming

is much more
than

Software >> Program

Software Engineering >>> Programming

is much much
more than

Software

Software Engineer

Software Engineering

If somebody has built a garage, would we let him built a skyscraper? No, never!

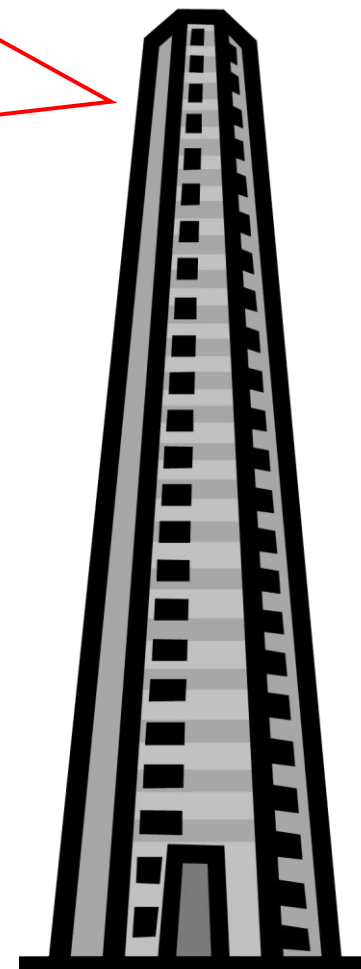
If somebody has written a program, would we let him built software? Yes we would!

Program

Programmer

Programming

But, of course we should not!



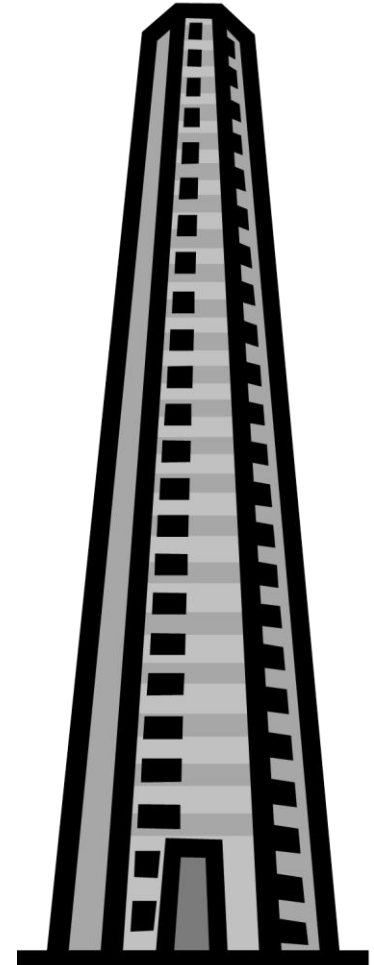
Die Menge aller **Programme**, **Prozeduren** und **Objekte**, zusammen mit den zugehörigen **Daten** und der **Dokumentation**, die für eine lauffähige Anwendung nötig oder wünschenswert sind.

[frei nach Informatik DUDEN und Hesse]

The sum of all **programs**, **procedures** and **objects** along with the associated **data** and **documentation**, which are necessary (or at least desirable) for running an application on a computer system.

... and a glimpse of how software can be developed by using models – without doing any programming at all.

Models are the “floor plans” of software engineers, and are the key to the success of software projects.

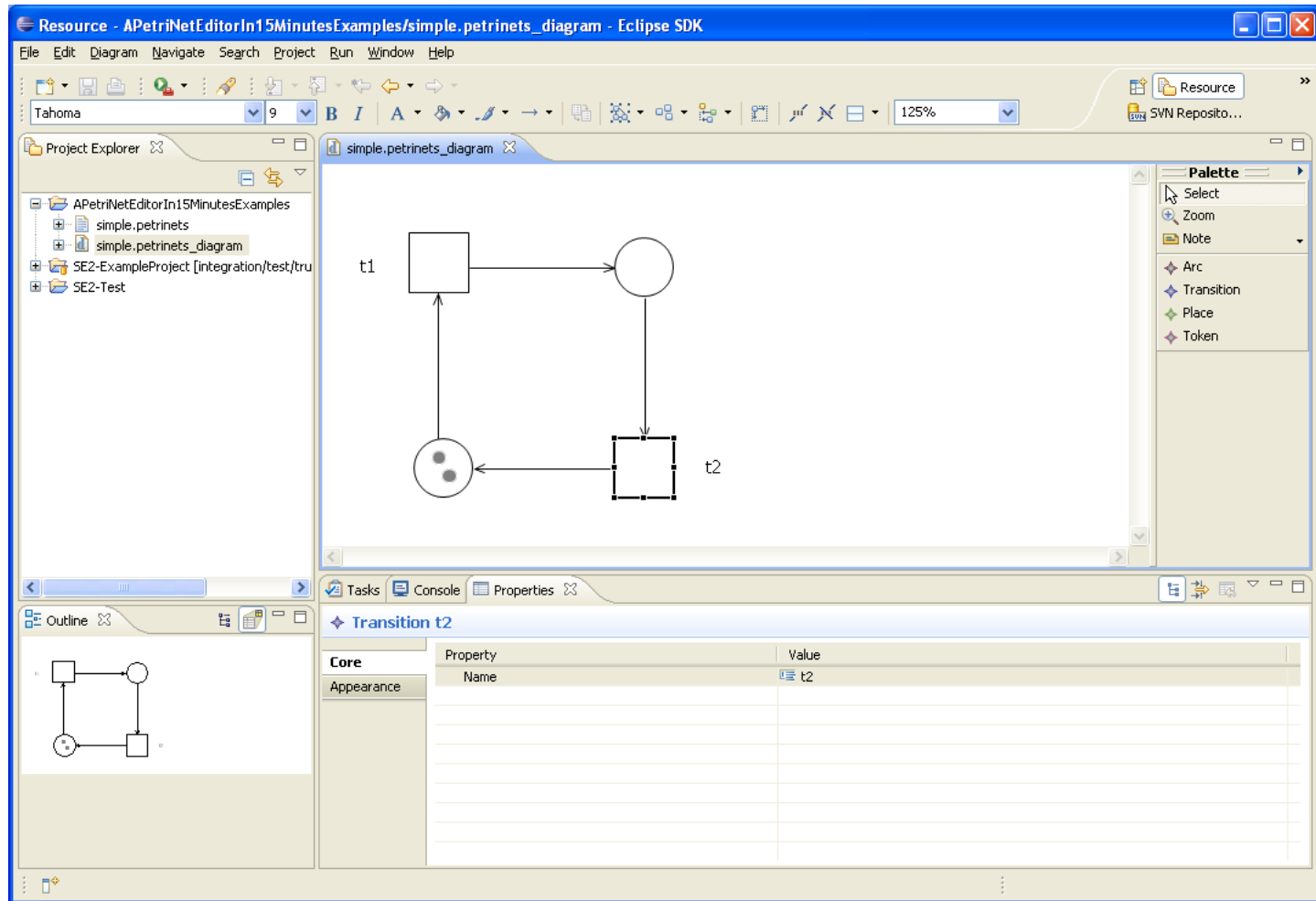
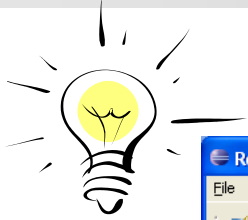


Idea for some Software

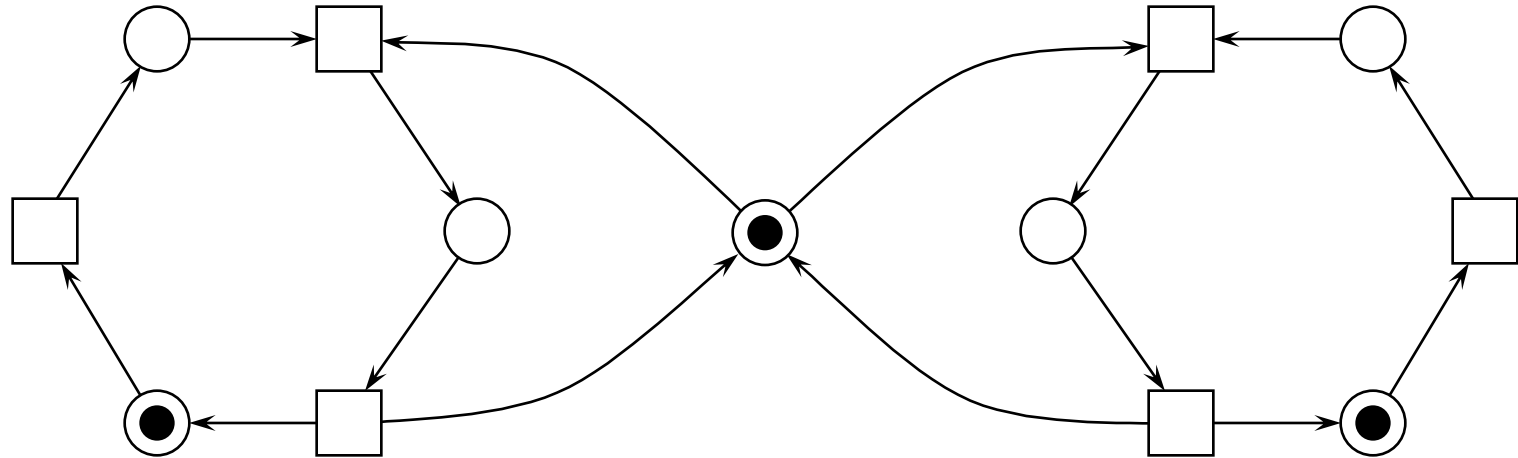
DTU Compute

Department of Applied Mathematics and Computer Science

Ekkart Kindler

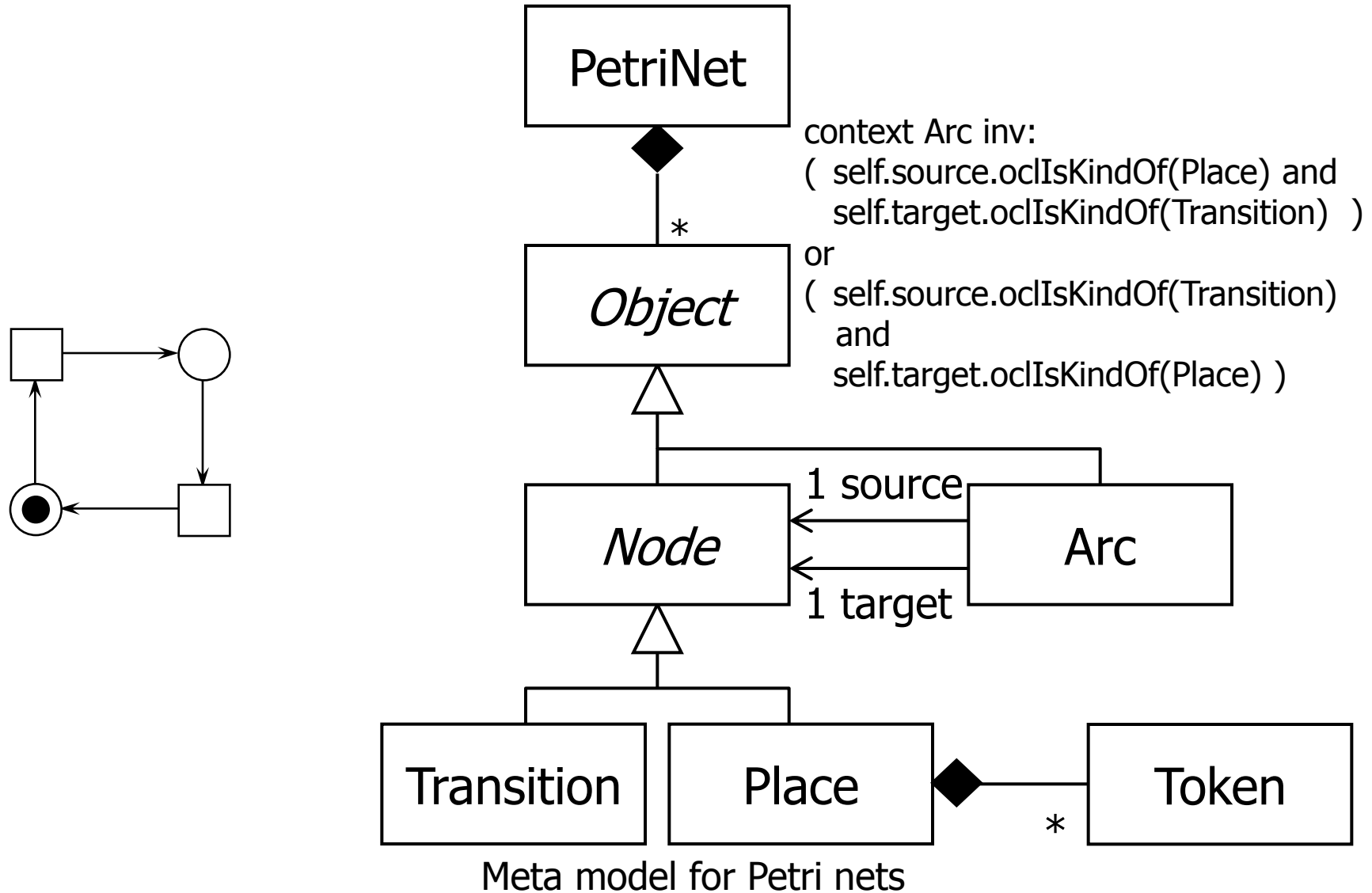


Example of a Petri net

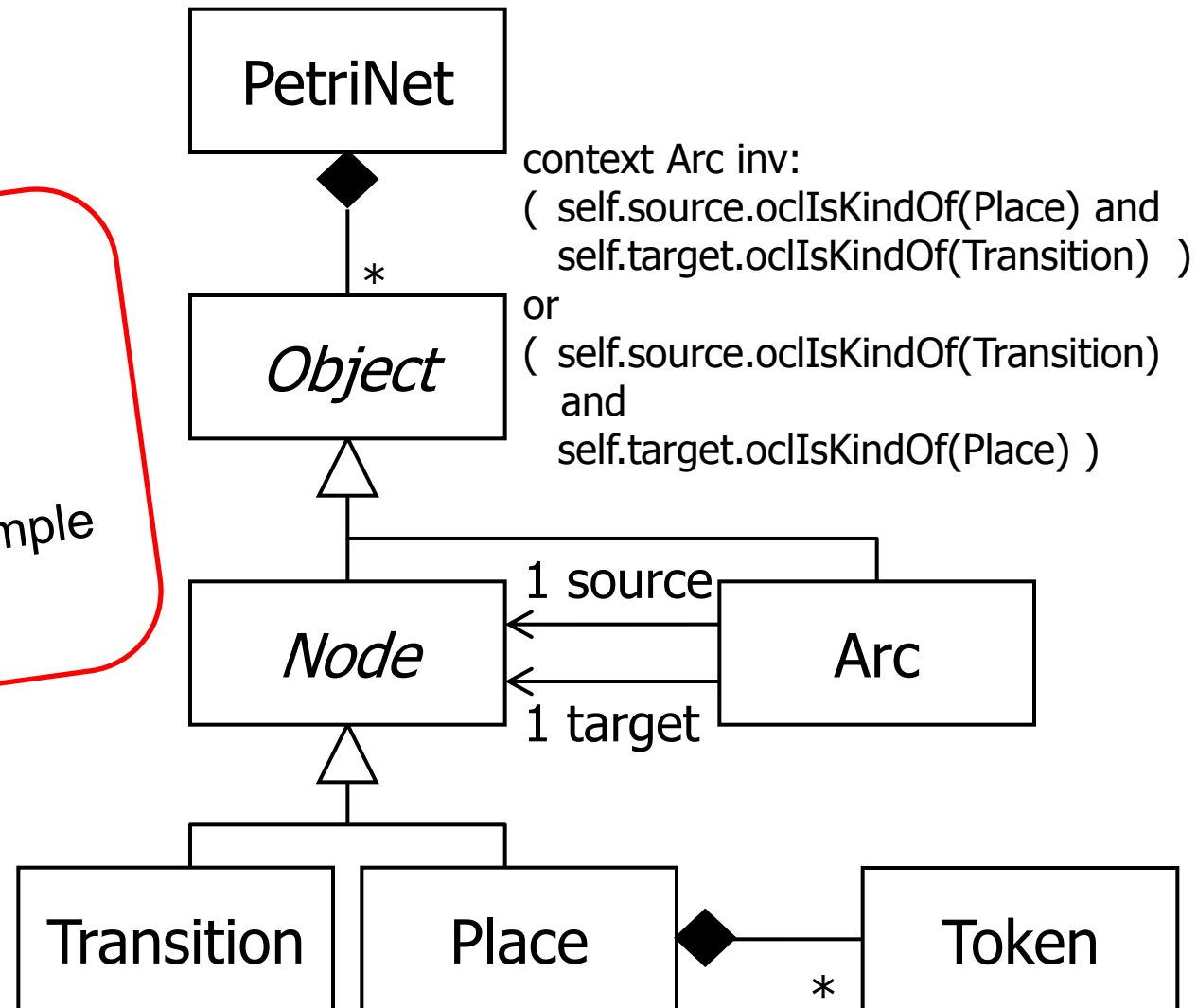


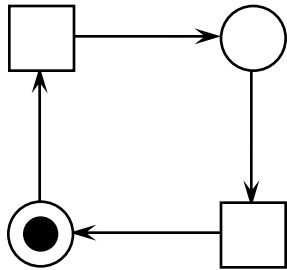
- Examples
- Taxonomy (done on blackboard)
- Glossary
- Model (developed on blackboard)

Rule: Never ever start making a UML model without having looked at some examples first and naming the main concepts (taxonomy)!

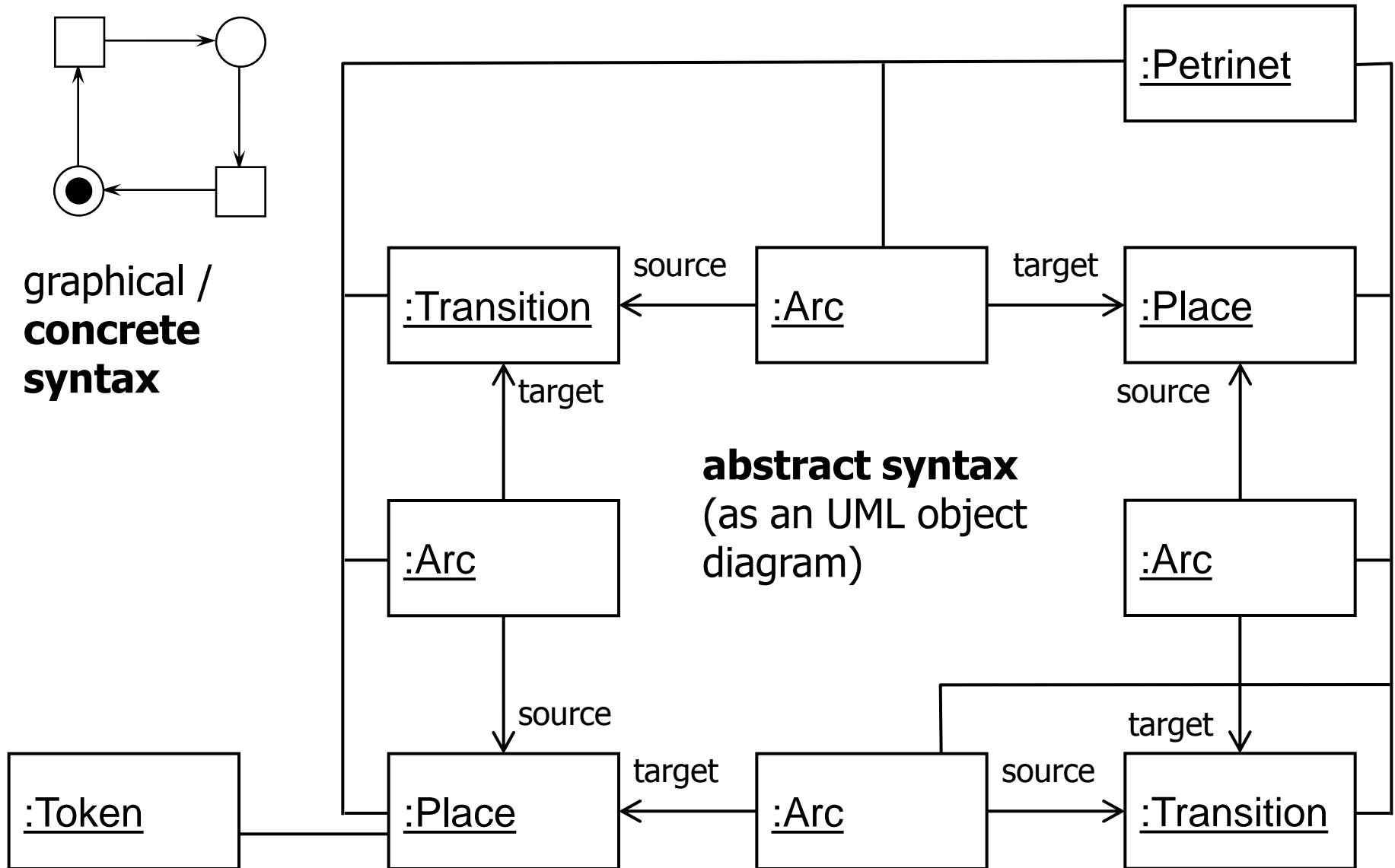


Rule: Don't think of programming for now!
These models are on concepts **only**: the concepts of "our" example domain: Petri nets!



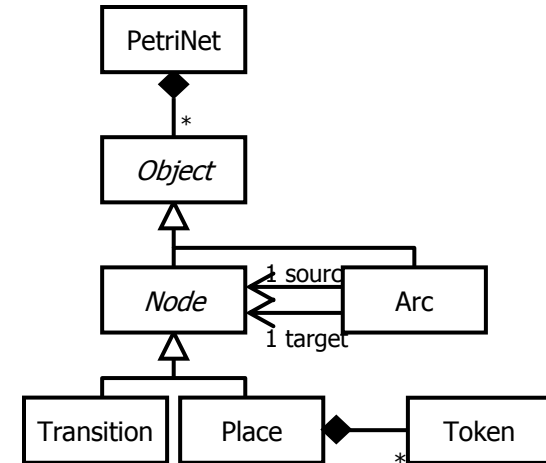


graphical /
concrete
syntax



meta model

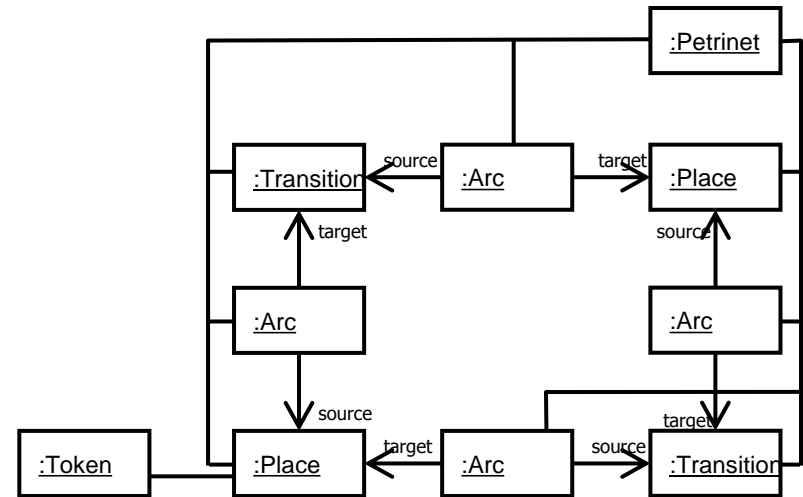
build-time



is an
instance of

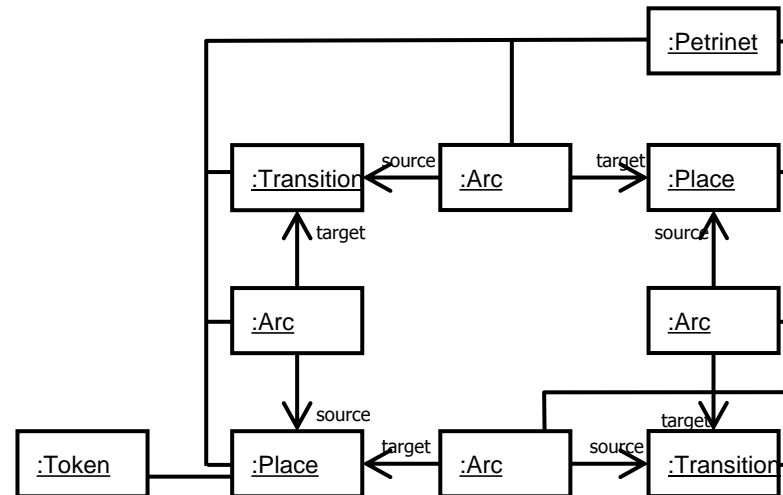
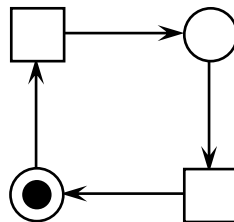
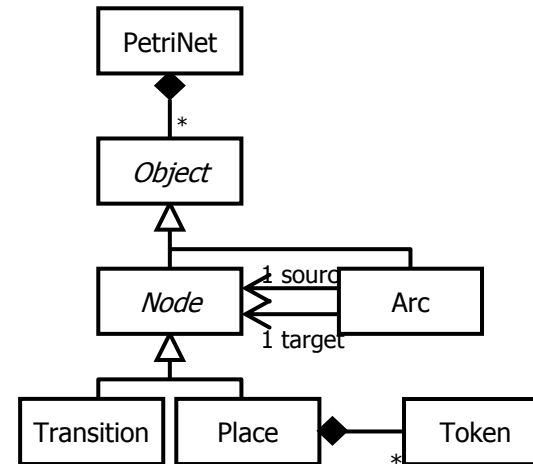
model

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)

Where does the concrete syntax come from?



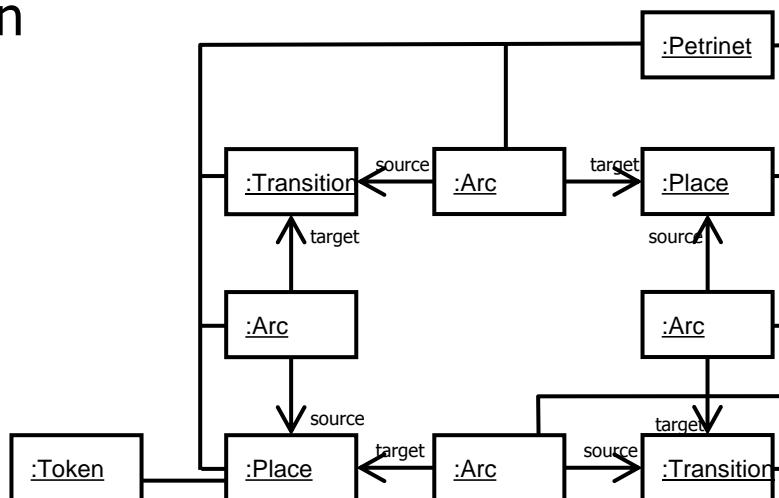
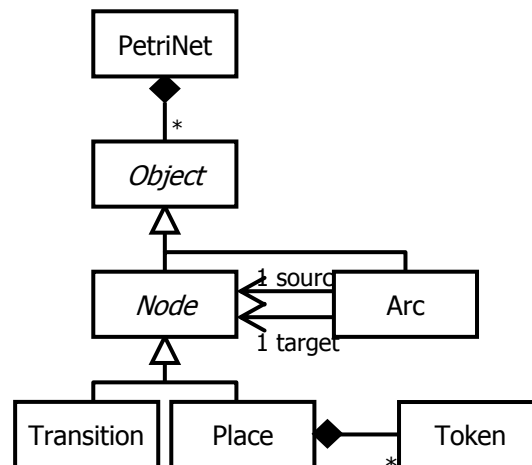
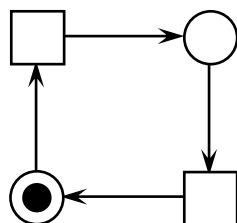
- Program an editor
- Standard technology for mapping abstract to concrete syntax: EMF / GMF / ...

Not a good answer here!

Details on GMF later:
“A graphical editor for
Petri nets in 15
minutes”



generate an
editor



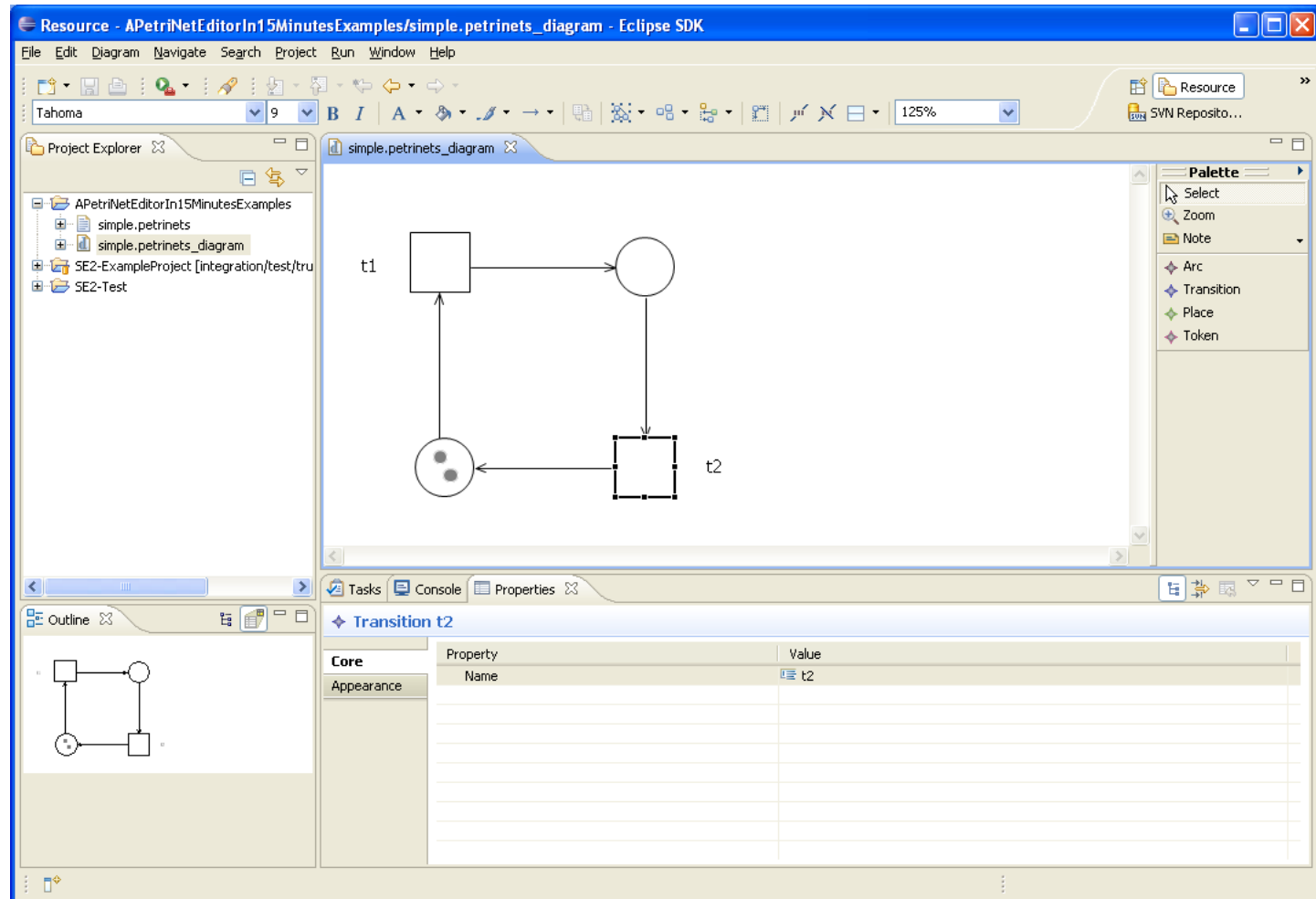
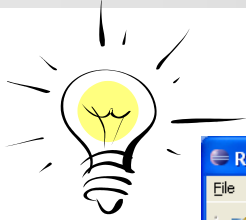
- Better Understanding
- Mapping of instances to XML syntax (XMI)
- Automatic Code Generation
 - API for creating, deleting and modifying model
 - Methods for loading and saving models (in XMI)
 - Standard mechanisms for keeping track of changes (observers)
 - Editors and GUIs

Idea for some Software

DTU Compute

Department of Applied Mathematics and Computer Science

Ekkart Kindler



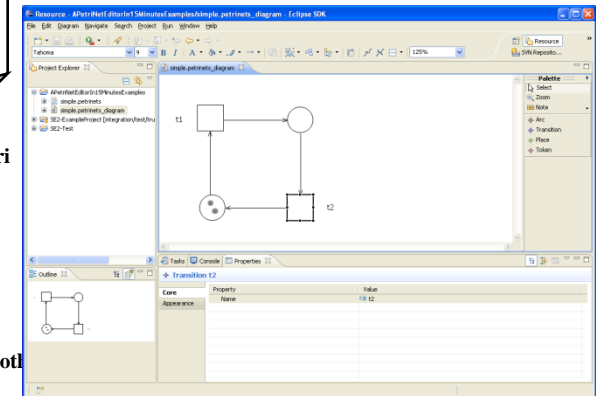
Get rid of
technical
artefacts!

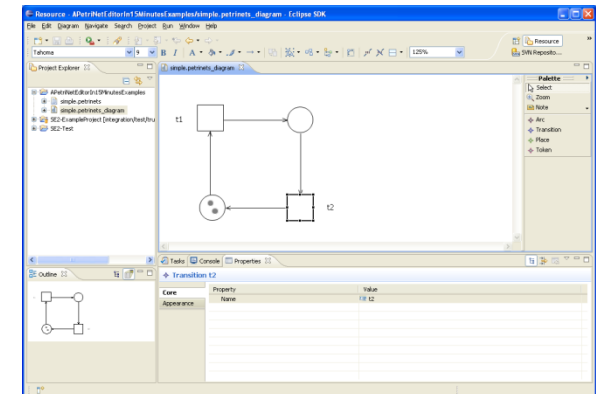
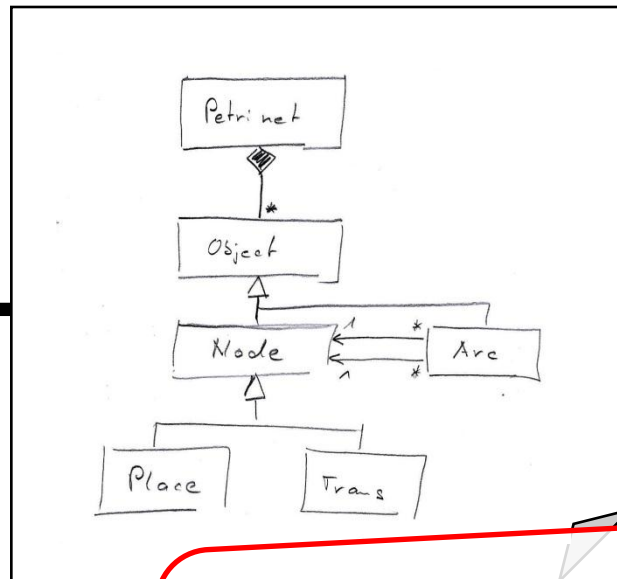
```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: %pluginName
Bundle-SymbolicName: APetriNetEditorIn15Minutes.diagr
Bundle-Version: 1.0.0
Bundle-Classifier:
Bundle-Location:
Bundle-Vendor:
Require-Bundle: org.eclipse.emf.ecore, org.eclipse.emf.ecore.xmi, org.eclipse.gmf.runtime.diagram.ui, 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
Bundle-Activator: APetriNetEditorIn15Minutes;visibility:=reexport

```

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

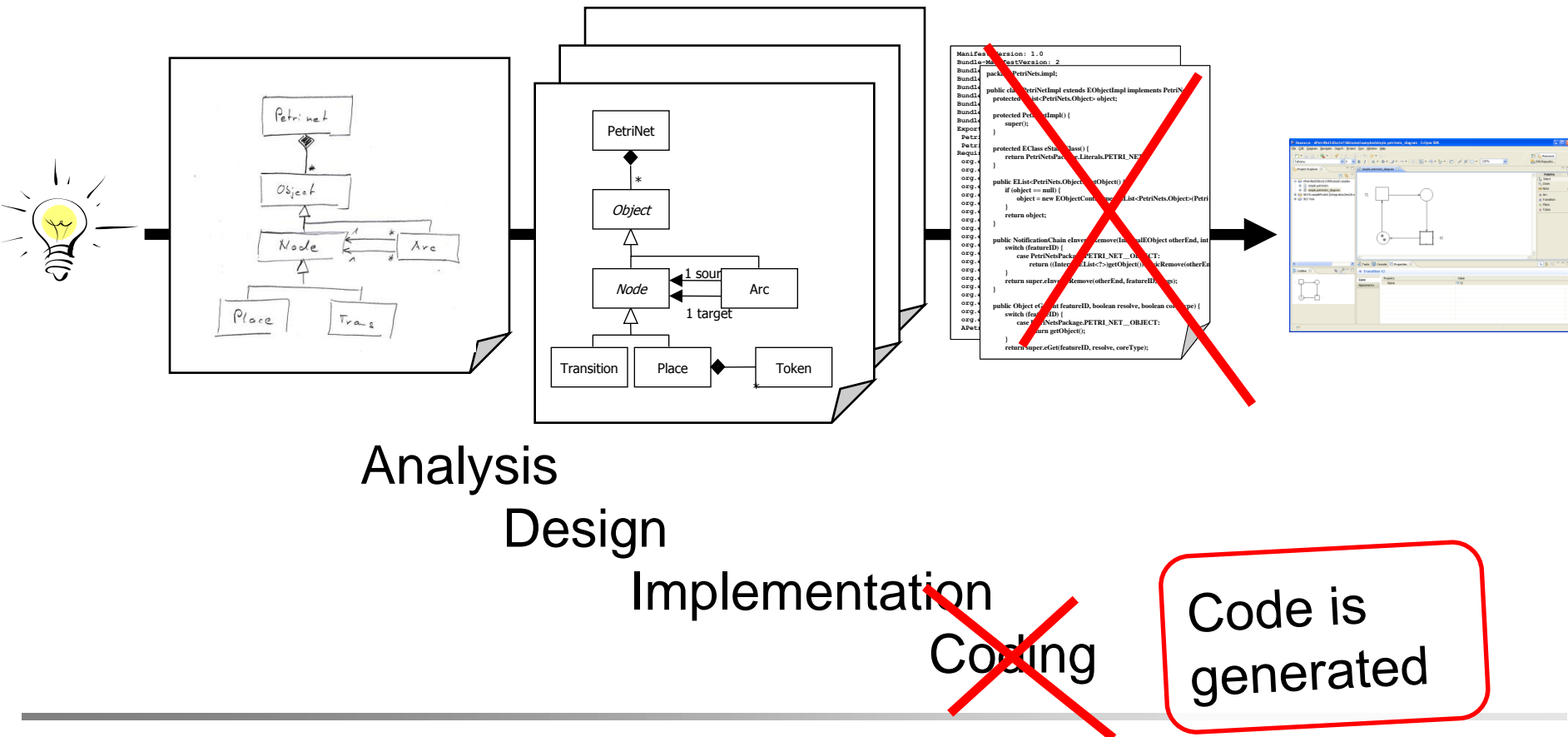
`return super.eGet(featureID, resolve, coreType);`





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

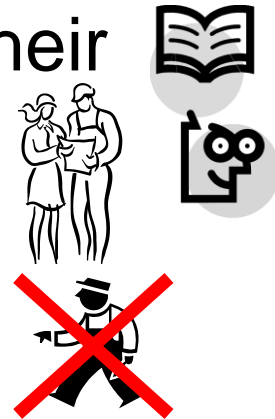
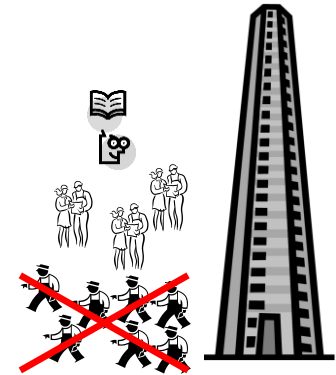
There are tools that partially support this idea already today (e.g. Eclipse and EMF/GMF). You will learn how to use them – and the general idea behind them in this course!



- 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.

- We will always have programming and programmers!
- We should always teach programming!
- But, software engineers should be trained in their engineering and modelling skills!
- And this is where they should be at their best!
- Most of the rest can be automated!
- Eventually, programming will be for software engineers as assembler is today for programmers.

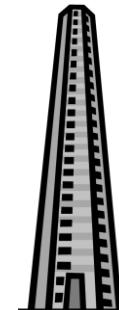
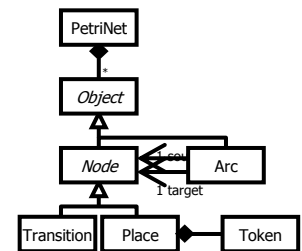
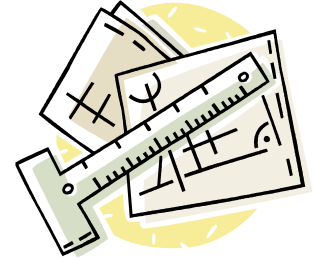


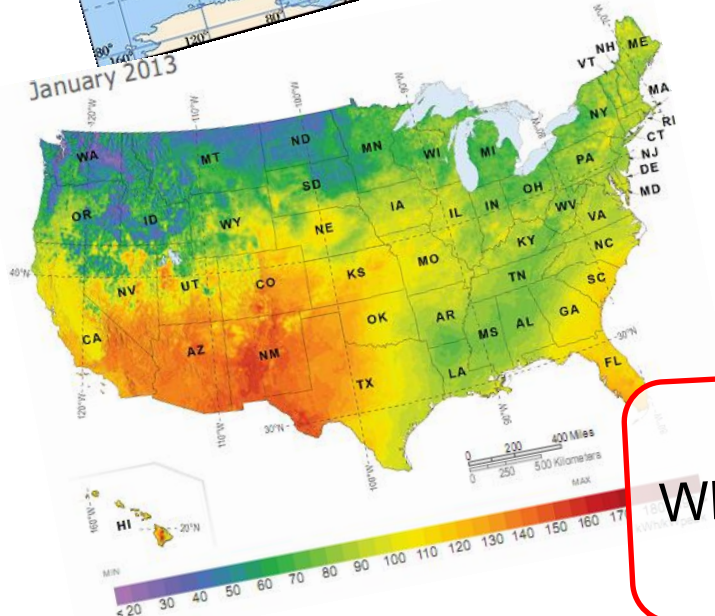
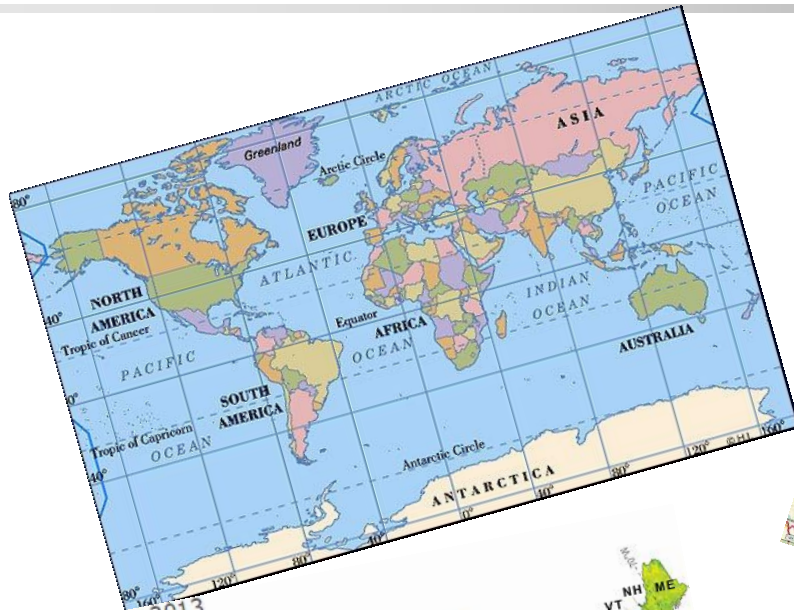
3. Modelling with a Purpose

→ Lecture 2

Analogies:

- Models as floor plans (see earlier slides)
 - Architects and construction engineers use quite different kind of plans – driven by the purpose
 - They even use models (miniatures)
- Models as maps
 - Understand the world (→ domain)
 - Find your way round in the software





Which of them is the best?



- Different level of abstraction and detail
- Different focus
- Different aspects


One map/ model is not enough

→ **Different purpose**

- For programs (small software) models are often not needed, and making them might be a waste of time.
- For software they are essential for building something which works out and the different pieces fit to each other

To be continued ...

Lecture 2:
Modelling with a purpose!

- Lecture 
 - Introduction and Motivation
- Organization
 - Organisation of this course
 - Project
- Tutorial
 - Tutorial 1: Getting started with EMF (Petrinet example)
 - Overview of task and steps
 - DO IT