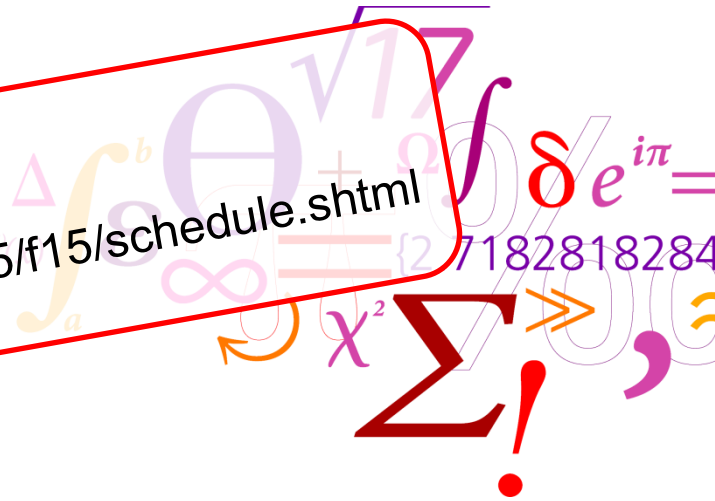
Advanced Topics in Software Engineering (02265)

Ekkart Kindler

DTU Informatics
Department of Informatics and Mathematical Modeling

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



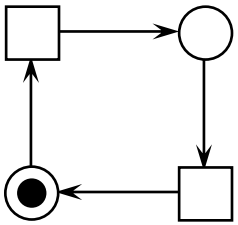
Recapitulation
(I. Introduction)

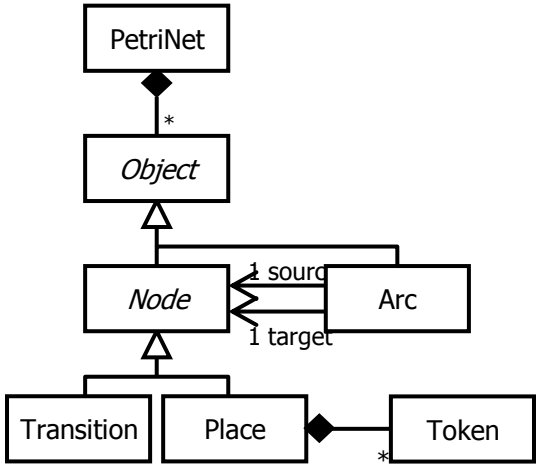
Levels of models

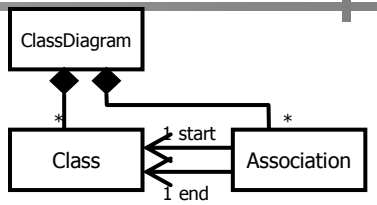
DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler

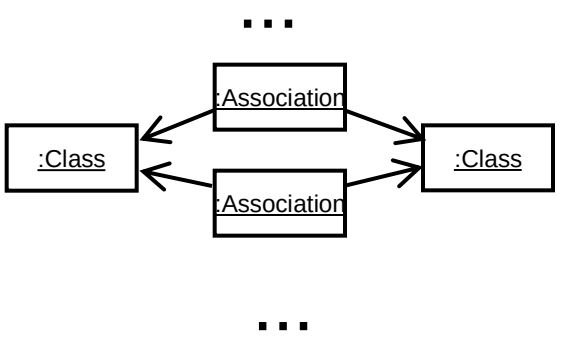
is an instance of

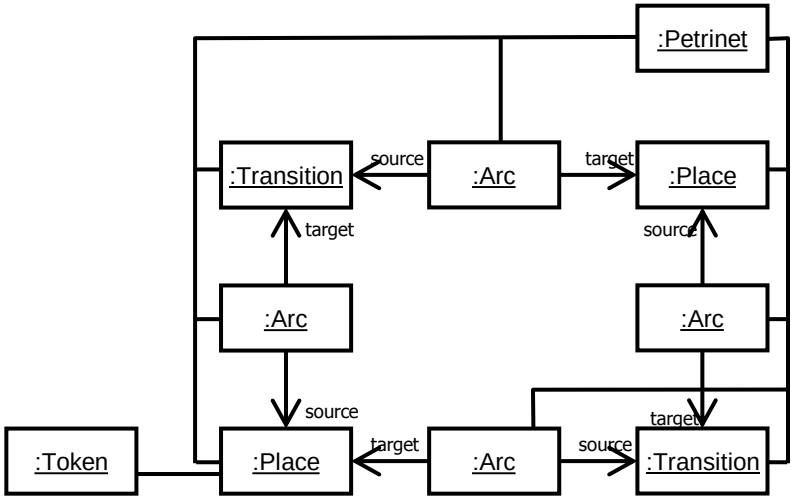
concrete syntax reprs. for











Goals of this course

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler

- Acquiring a bit more experience!

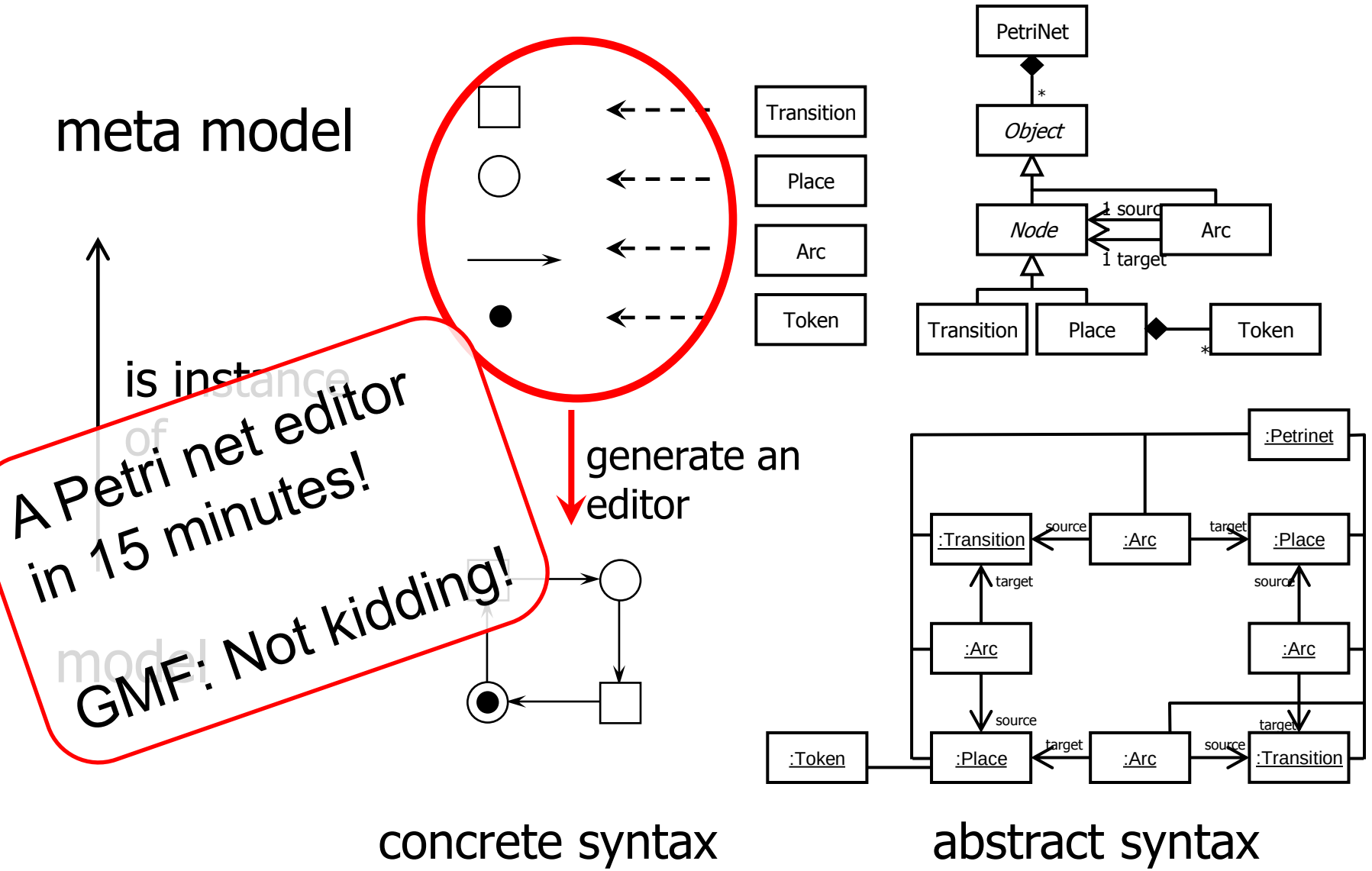
- Learn what is behind the scenes!
  - Understand concepts and technology
  - Apply (some of) them
  - Experiment and evaluate technology

- 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**

EMF/GMF Technology

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler



Basic terminology

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler

- Concept
- Formalism
- Method / methodology
- Model / meta-model
- Notation
- Principle
- Technique
- Technology
- Tool

- Software engineering

- Taxonomy
- Ontology

- Framework
- Approach

Course organisation

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler

- 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);

II. Domain Specific Languages

- Domain Specific Language (DSL)
- Domain Specific Languages (DSLs)

What do they mean?
What is their “spirit”?

DSL / DSLs

The terms DSL and DSLs are uses since the the mid 90ties; ”Domain Specific Automatic Programming” even dates back to the mid 80ties.

Still, there is not is not a uniform or universal understanding of what a DSL or what DSLs are; it depends a bit on the background which characterisitics of DSLs are considered to but important or relevant.

This lecture gives an overview – but with a model-based software engineering bias!

- DSL (singular):
  A single domain specific language,
  designed and realised according to some principles
  and for a specific purpose or a specific domain

- DSLs (plural):
  - Disipline and principles for designing and realising a DSL
  - A technology or set of technolgies for designing and realising a DSL
  - A way of ”thinking” software design

1. Examples

- COBOL
- Lisp
- PROLOG

Some examples named by some proponents of DSLs; if there, not all would agree!

- SQL (Structured Query Language → DB)
- BNF (Backus Naur Form → syntax definition)
- regex (regular expressions)
- lex, yacc (compiler construction)

- Shell scripting languages
- OCL

Some DSL existed even before the term DSL was invented!

- BPEL (Business process execution language)
  BPML (Business process modeling language)

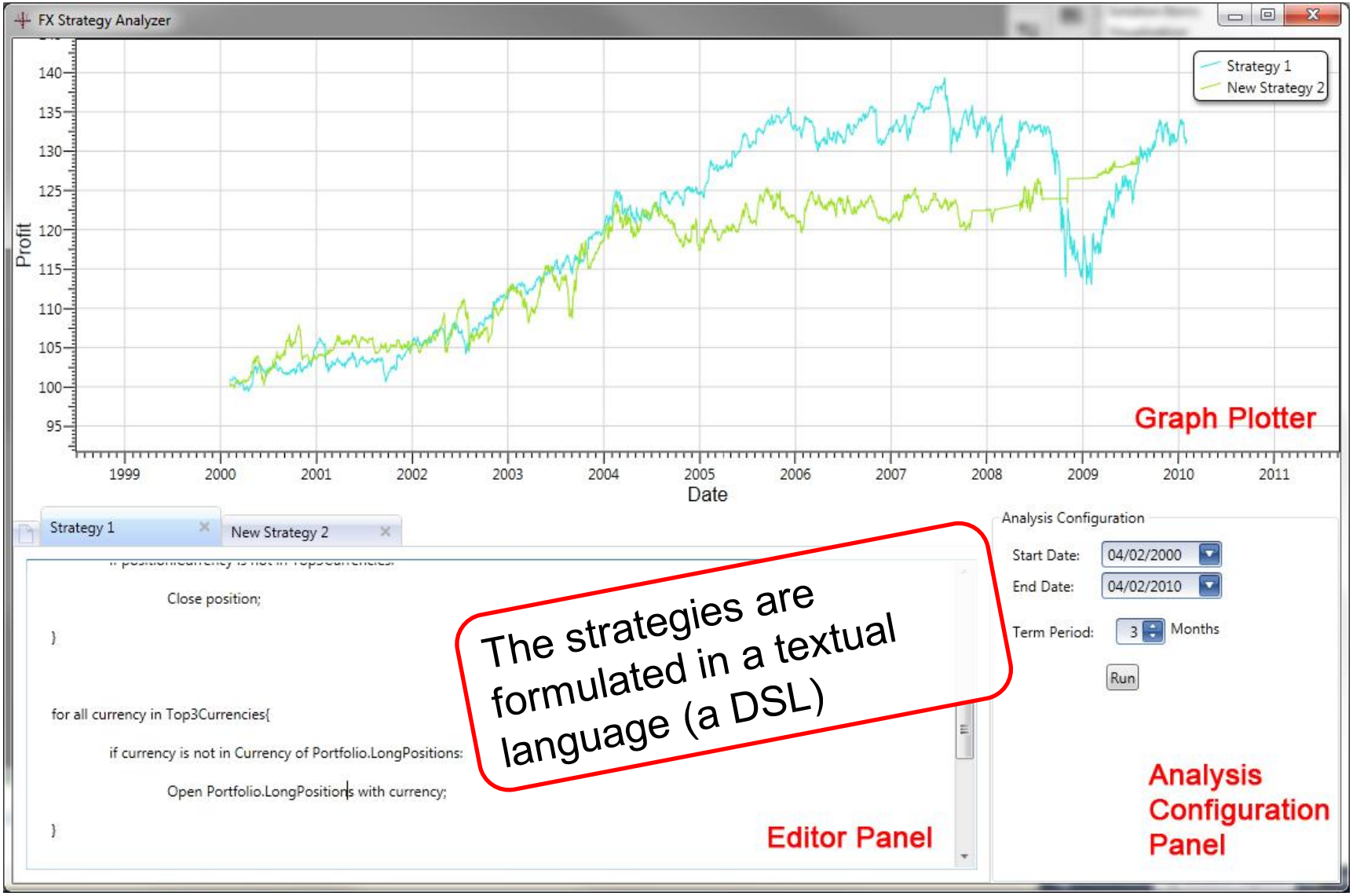
- Petri nets
- ECNO
- OCL

- Trading strategy language (see next slide)

- PDF / PostScript
- HTML / CSS

Tool for testing FX strategies

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler



Counter-examples

- C
- C++
- C#
- Java
- Ruby
- ...

- UML
- ...

2. Characteristics

Traditional distinction of ”programming languages”:

- General Purpose Languages (GPL):
  - universal
  - same thing can be achieved in many differnent ways
  - Turing complete
  - huge
- Special Purpose Language (SPL):
  - made for a specific purpose
    (adequate for this specific purpose)
  - succinct and highly expressive (for given purpose)
  - typically, not Turing complete
  - small

SPL and DSL?

- Is any SPL a DSL?
- Is every DSL a SPL?

DSL characteristics

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler

- Textual (language) vs graphical (notation)
- Programming vs. modelling
- Domain of application vs separation of concerns
- Way of thinking design vs
  use of specific”DSL technology”
- Abstraction vs technical
- User focus vs technical focus
- Language vs framework
- Idiom oriented vs. programming oriented

Classification

- **Embedded DSL**:
  Embedded to an existing programming language by adding some framework for some purpose (often some functional languages with syntactic sugaring features)

  - Typically textual languages!
  - Often programmed (with ”DSL thinking” in mind)

- **External DSL**:
  Standalone language (graphical/textual) which is then compiled or interpreted. Often realized by DSL development tech

  - Often: Focus on adequate concrete syntax!
  - Typically realized by using ”DSL technologies”

3. Parts of a DSL definition

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler

- **Abstract syntax** (see L01):
  language concepts and their relation
  (*API / domain model / framework)*

- **Concrete syntax** (see L01):
  syntactical representation of concepts
  (graphical or textual)

Actually, there could be different concrete syntax for the same abstract syntax.

- **Semantics** (what it does):
  Code generation or interpretation, which enacts what an instance of the DSL says

DSL Technolgies typically support the first two steps; and might help a bit with the last!

4. Philosophy

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler

- A DSLs should help decrease redundancy and unnecessary work
- A DSL should help separating the variable or generic parts of a software product from parts which do not change
- A DSL should improve reuse
- A DSL should support abstraction form irrelevant technical details
- A DSL should emphasize the domains idioms