


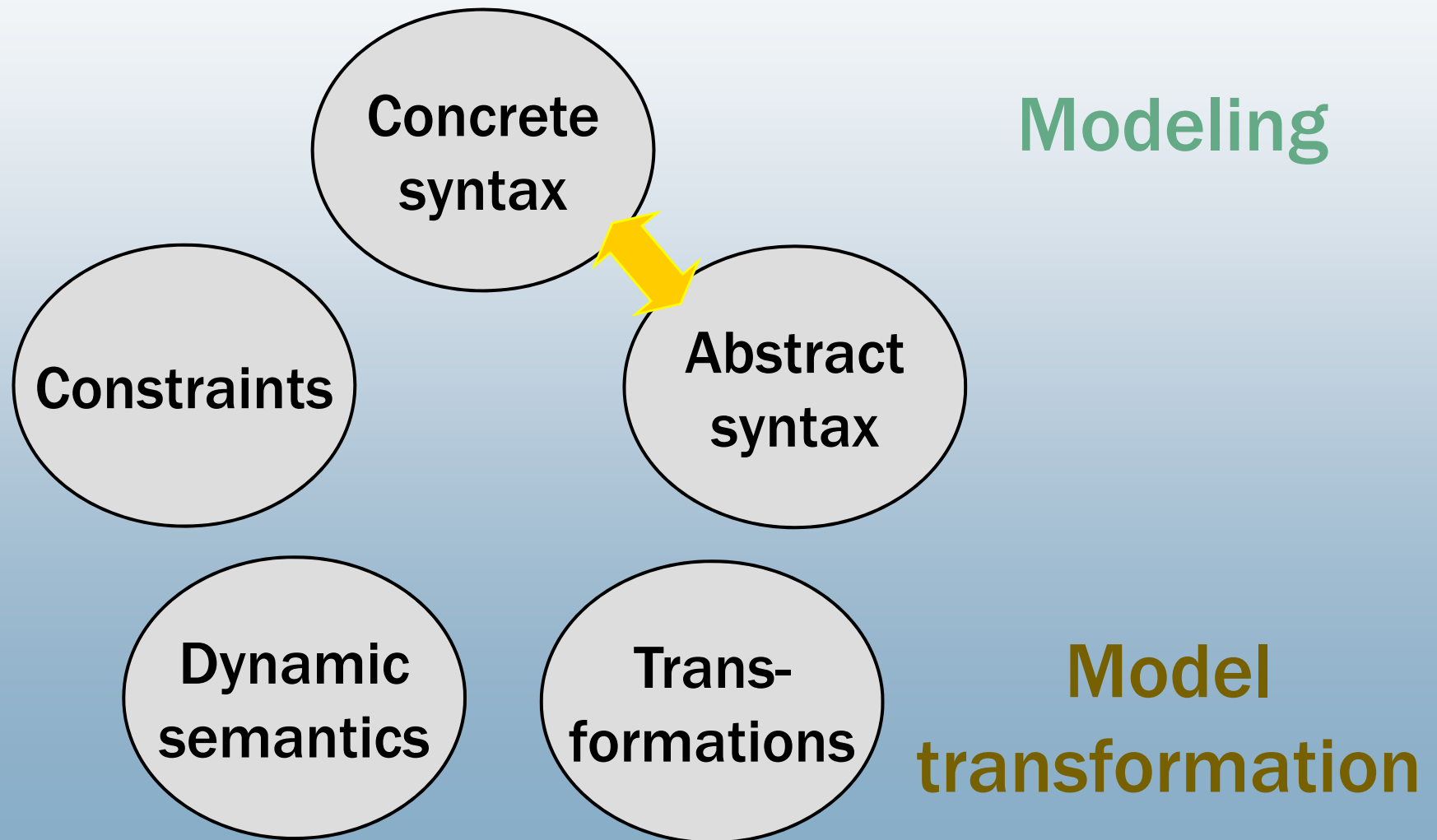
Challenges for advanced domain-specific modeling frameworks



István Ráth
Dániel Varró

Department of Measurement and Information Systems
Budapest University of Technology and Economics

Aspects of language engineering





ViatraDSM

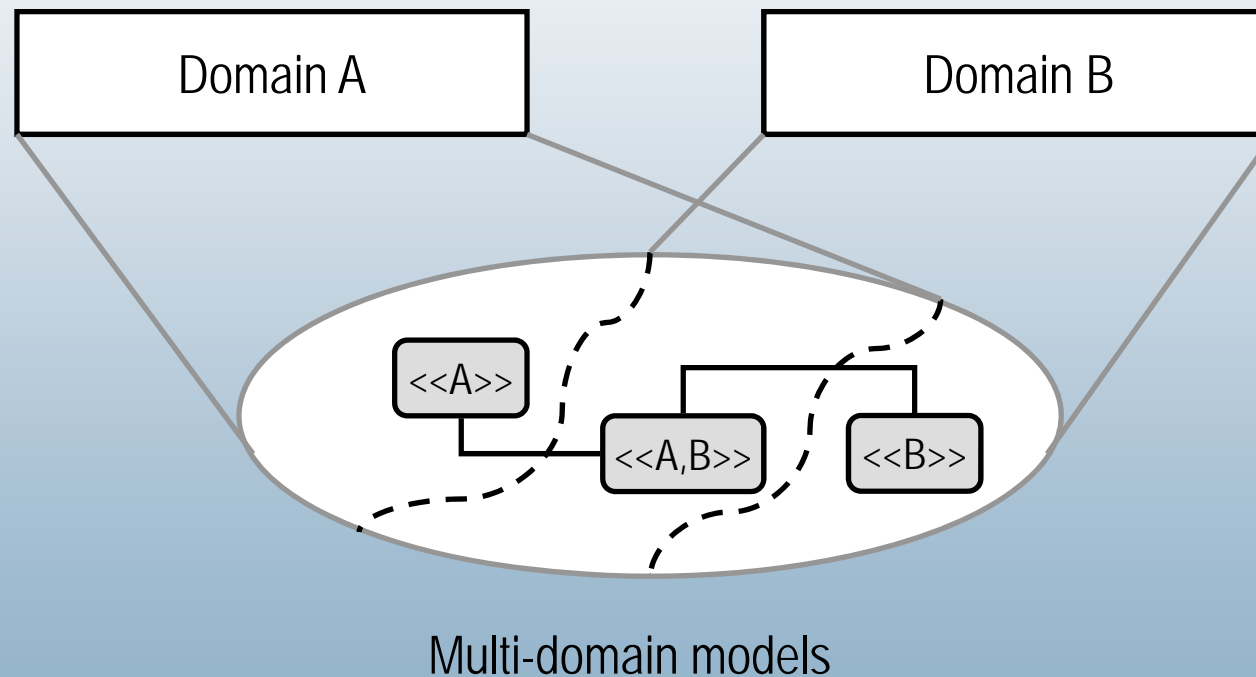
- Integrated modeling and simulation framework
 - integrated: learn one approach for **all** aspects
 - modeling: graphical domain-specific editors
 - simulation: editing-time *interactive* model simulation
- Based on VIATRA2 and Eclipse
 - Using the Graphical Editing Framework (GEF)



Challenges

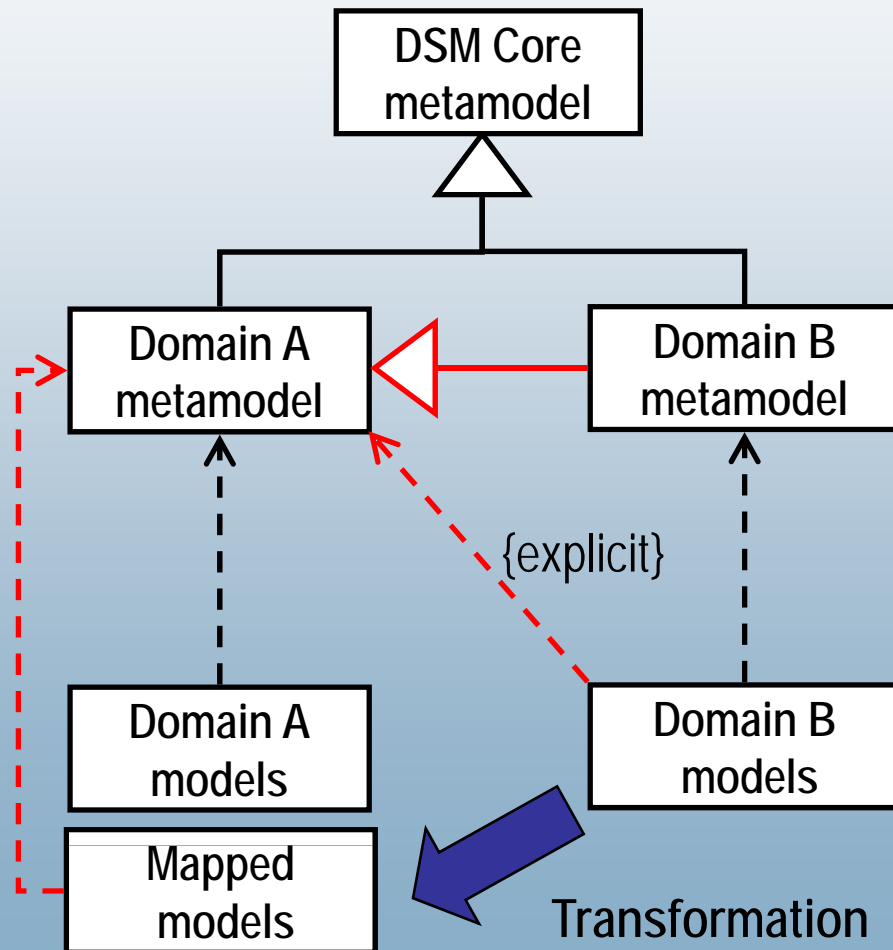
- Simplify diagrams
 - “association = edge, class = node” → arbitrary mapping
 - declarative mapping specification
 - minimize manual coding requirements
- Integrate dynamic modeling (simulation)
 - intuitively
 - fully customizable

Domain integration: Multi-domain modeling



Multi-domain modeling

- How to do it?
 - Light-weight approaches
 - Model-level tagging
 - Metamodel-level stereotyping
 - „Heavyweight” approach
 - Model transformation

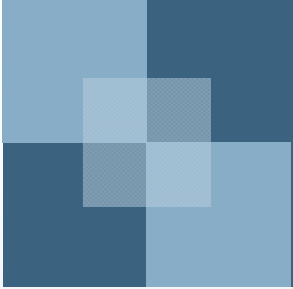


Multi-domain editors

- How to do it?
 - Light-weight approach
 - Model-level tagging
 - Metamodel-level stereotyping

- Show only relevant attributes
- Problem: differences in structure
 - Solution: transformations!

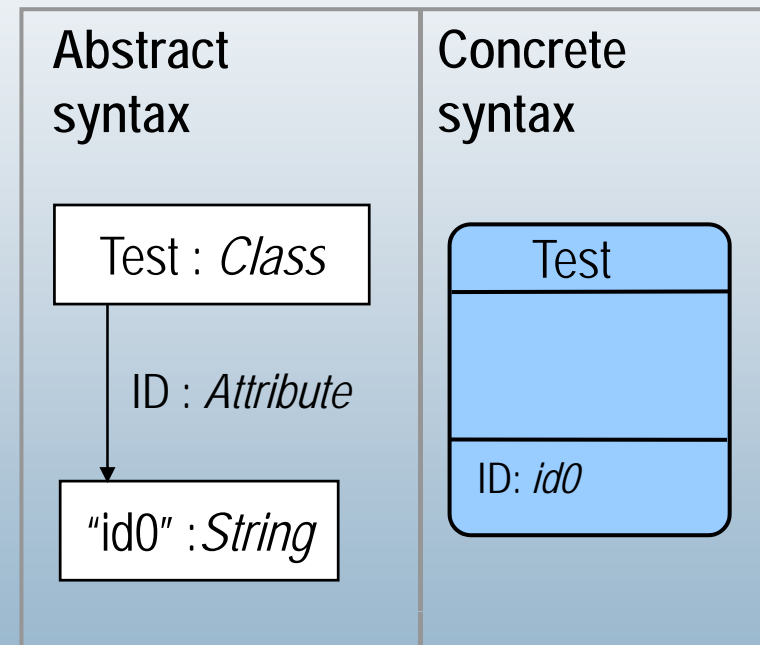
| Property | Value |
|------------------|-------|
| Modeling | |
| schedulingPolicy | |
| throughput | 23 |
| utilization | 0.4 |



Separating abstract and concrete syntax

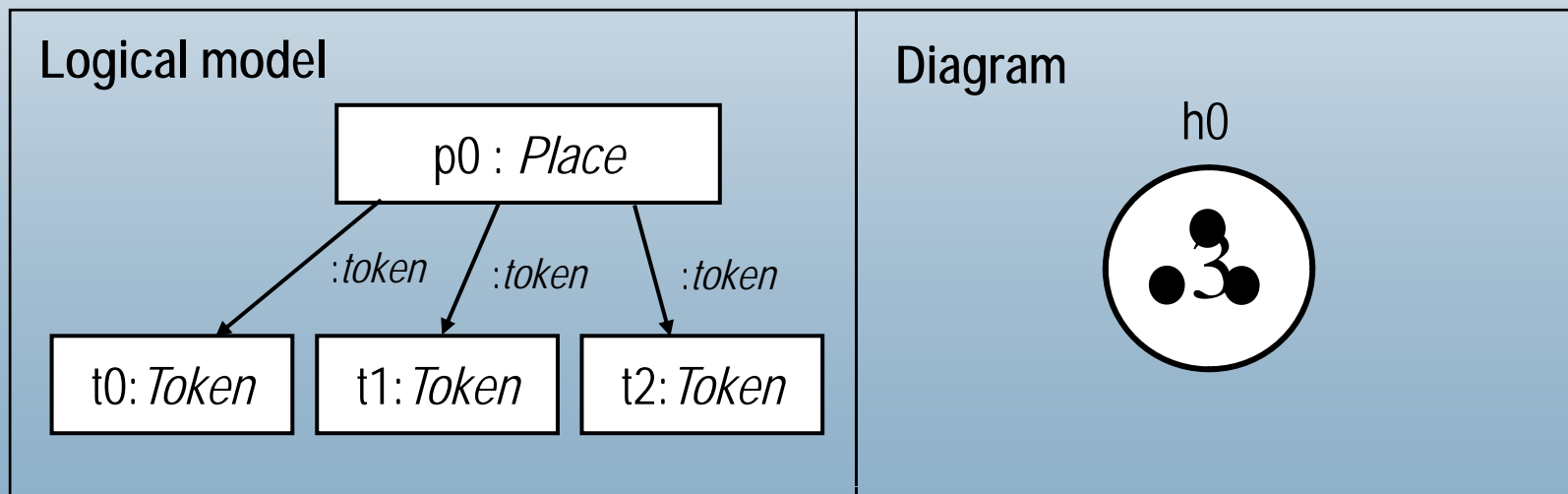
Separating abstract and concrete syntax

- What?
 - concrete syntax \approx diagrams
 - abstract syntax \approx logical model
- Why?
 - reduce complexity (for the user...)
 - more possibilities (for the language engineer)

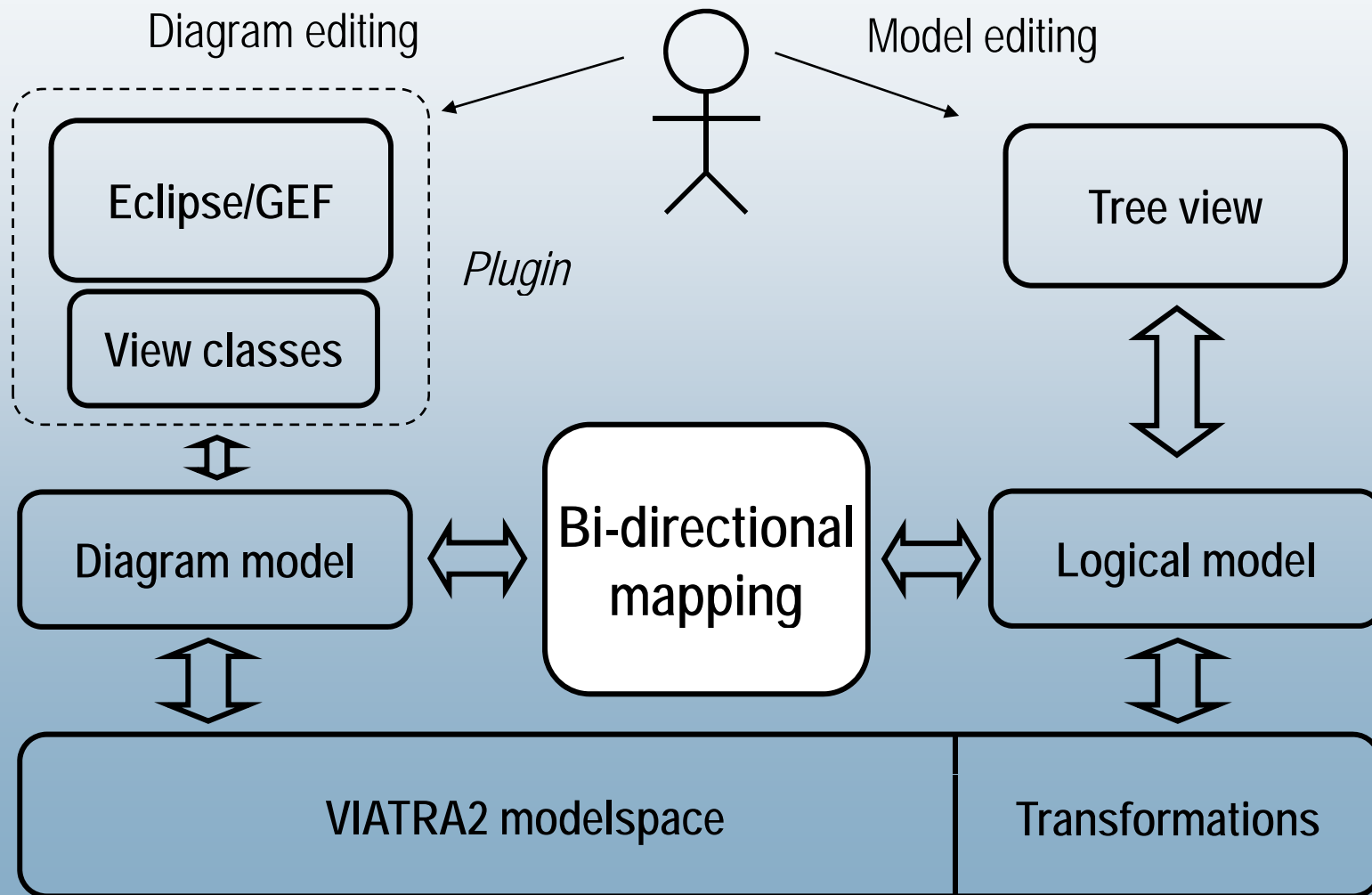


Objectives

- Arbitrary mapping
 - abstraction
 - aggregation
 - diagram-specific elements
 - decorators

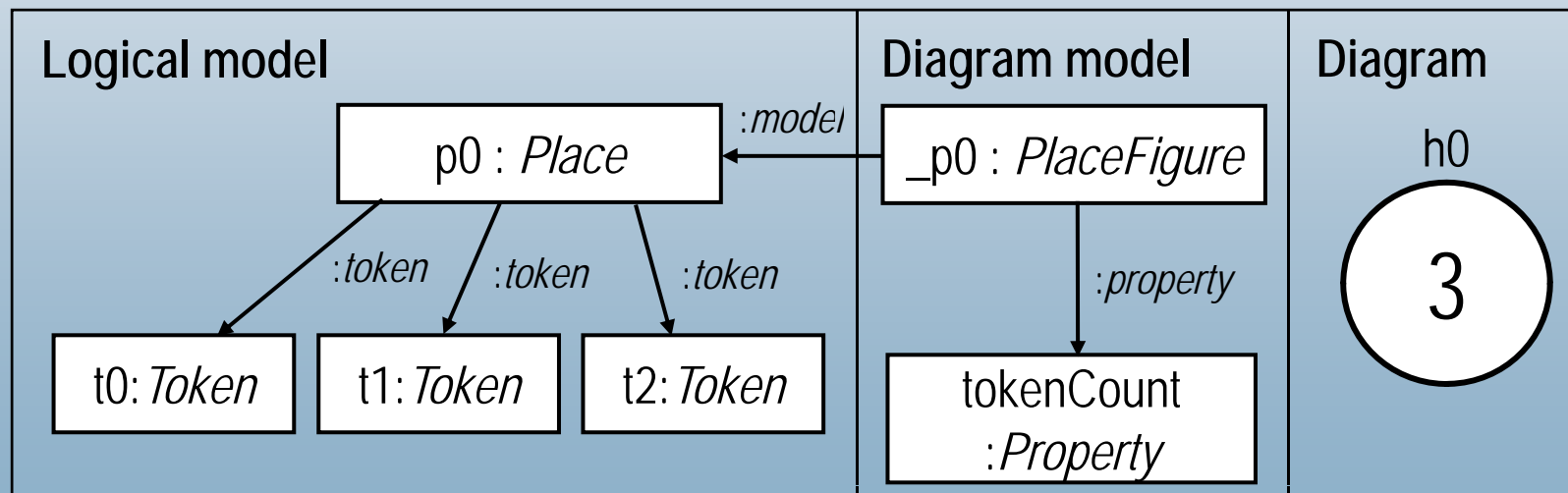


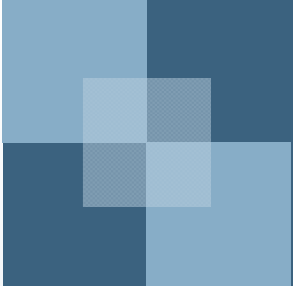
Architecture



Proposal

- Bi-directional mapping
 - goal: arbitrary mapping
 - means: metamodeling + **model transformations**





Separating abstract and concrete syntax

- Implementation
 - on the model level
 - the *user* decides what to show
 - most tools support it
 - on the metamodel level
 - the *language engineer* defines diagrams
 - uses a **separate modeling layer** for graphical representation
 - new approach!



Demo #1: TokenCount

Viatra R2 - dsm_coremeta.vpml - Eclipse SDK

File Edit Navigate Search Project Run Window Help

Outline

- PetriNet model elements
 - pn0
 - p0 (0)
 - p0_fork
 - p1 (2)
 - p1_join
 - uN1192_3f
 - uN1194_3f
 - p2 (0)
 - p3 (0)
 - p4 (2)
 - exit
 - fork
 - join
 - restart
 - PetriNet diagrams

Viatra R2 Frameworks

Petri net State machine UML Performance Security System Test Domain

Error Log Code Out View Properties

The diagram illustrates a Petri net with the following components and flow:

- Places:** p0 (0), p1 (2), p2 (0), p3 (0), p4 (2). Place p1 is highlighted with a dashed box.
- Transitions:** fork, join, exit, restart.
- Flow:** p0 → fork → p2 and p1. p2 and p1 → join → p3. p3 → restart → p0. p2 → exit → p4.



Dynamic modeling: simulation



Integrated model simulation

- Why?
 - constructing new languages: no existing tool support
 - existing languages: insufficient tool support
 - "model → generate → test" → see changes instantly
 - faster development
- What to simulate?
 - **Translator**: execute generated code
 - **Interpreter**: direct model manipulation

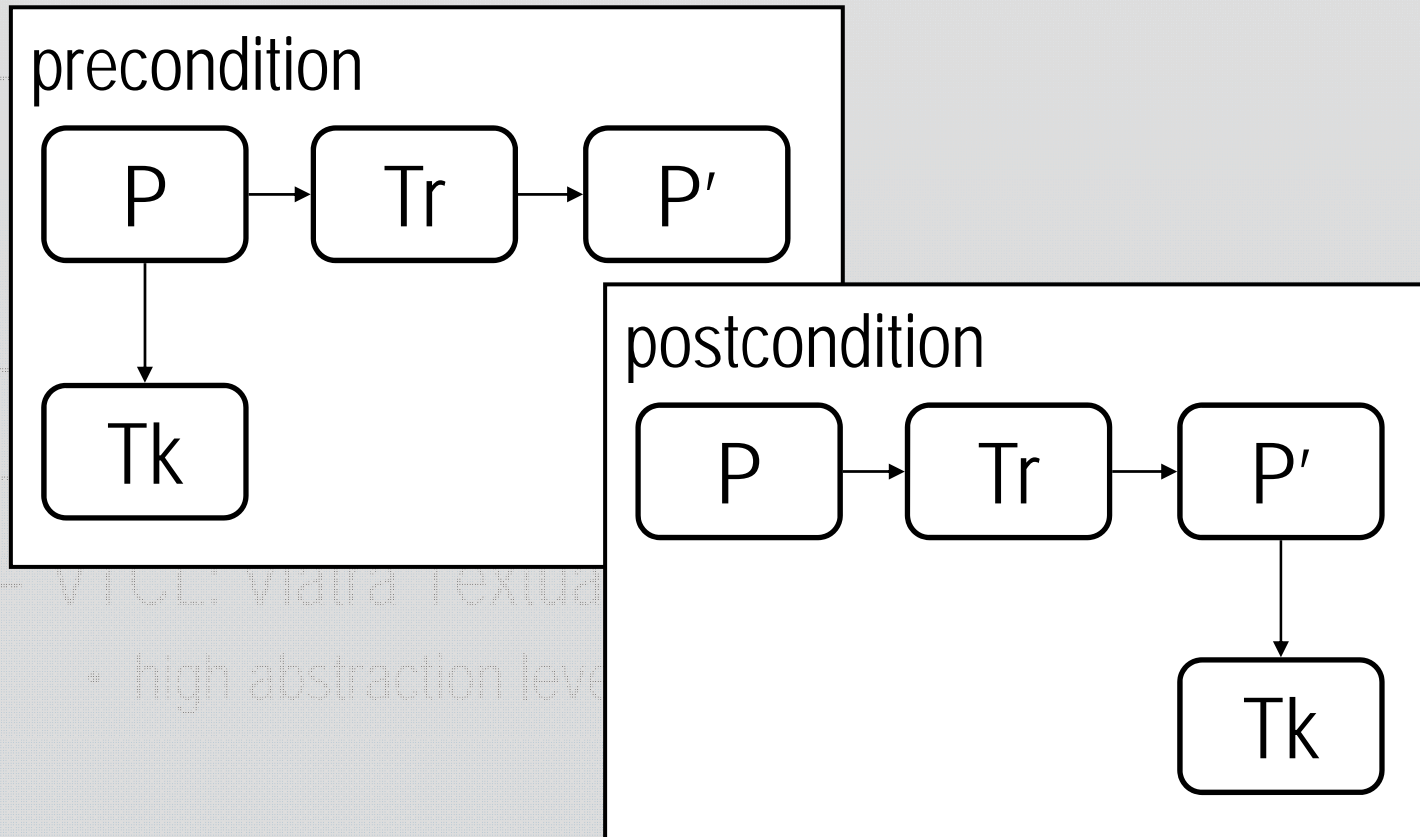


Integrated model simulation

- Definition of model simulators
 - VIATRA2 transformations
 - Guided (interactive) simulation
 - Automatic simulation
 - declarative semantics: graph patterns
 - imperative semantics: abstract state machines
 - VTCL: Viatra Textual Control Language
 - high abstraction level DSL

Integrated model simulation

- **Graph patterns**





Demo: Petri nets

Viatra R2 - dsm_coremeta.vpml - Eclipse SDK

File Edit Navigate Search Project Run Window Help

Outline » 2 □

PetriNet model elements

- pn0
 - p0 (1)
 - p0_fork
 - uN1228_3f
 - p1 (0)
 - p1_join
 - p2 (0)
 - p3 (0)
 - p3_restart
 - p4 (3)
 - exit
 - fork
 - join
 - restart
- PetriNet diagrams

Viatra R2 Frameworks □

- framework1
- framework2

dsm_coremeta.vpml *dsm_coremeta.vpml □

Select
 Mar...

```

    graph LR
      p0((p0: 1)) --> fork[ ]
      fork --> p2((p2: 0))
      fork --> p1((p1: 0))
      p1 --> join[ ]
      p2 --> join
      join --> p3((p3: 0))
      p3 --> restart[ ]
      restart --> p0
      p2 --> exit[ ]
      exit --> p4((p4: 3))
  
```

Petri net State machine UML Performance Security System Test Domain

Error Log Code Out View Properties □

| Property | Value | |
|---------------------------|-------|--|
| Diagram | | |
| NAME | null | |
| X | 250 | |
| Y | 150 | |
| Modeling | | |
| Place_Capacity [capacity] | 1111 | |
| Token count | 0 | |
| | | |
| | | |



Summary

- ViatraDSM
 - integrated language engineering environment
- Separation of abstract and concrete syntax
- Integrated interactive model simulation

<http://eclipse.org/GMT>

(VIATRA2 feature)