Software Language Engineering: Transformation

Tijs van der Storm

Recap

- Grammar -> Parser -> Parse Tree -> AST
- Name resolution: recover referential structure
- Checking: find errors not captured by syntax
- Semantics: interpret or compile
- Today:
  - transformations everywhere

Transformations

- Instruction selection
- DSL code generation
- Desugaring
- Function inlining
- Constant folding
- Constant propagation
- Refactoring
- Renovation
- Simplification
- Interpretation (?)

Outline

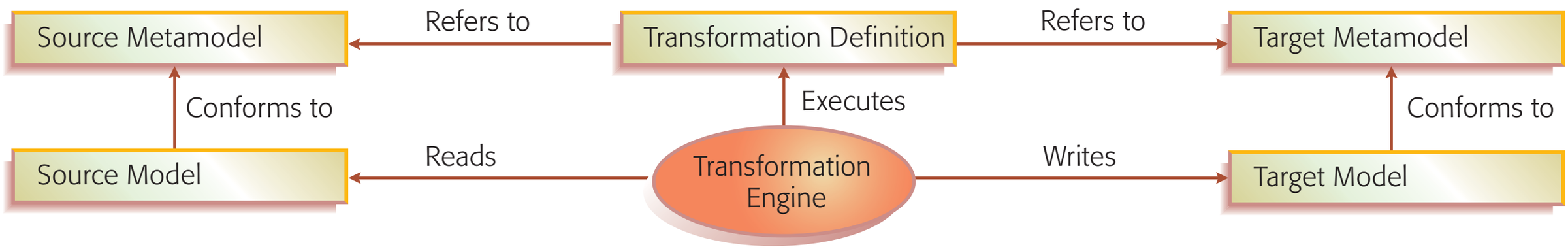
- Basic concepts of model transformation
- Scope of a transformation
- Direction of a (model) transformation
- Tools for model transformation

Feature-based survey of model transformation approaches

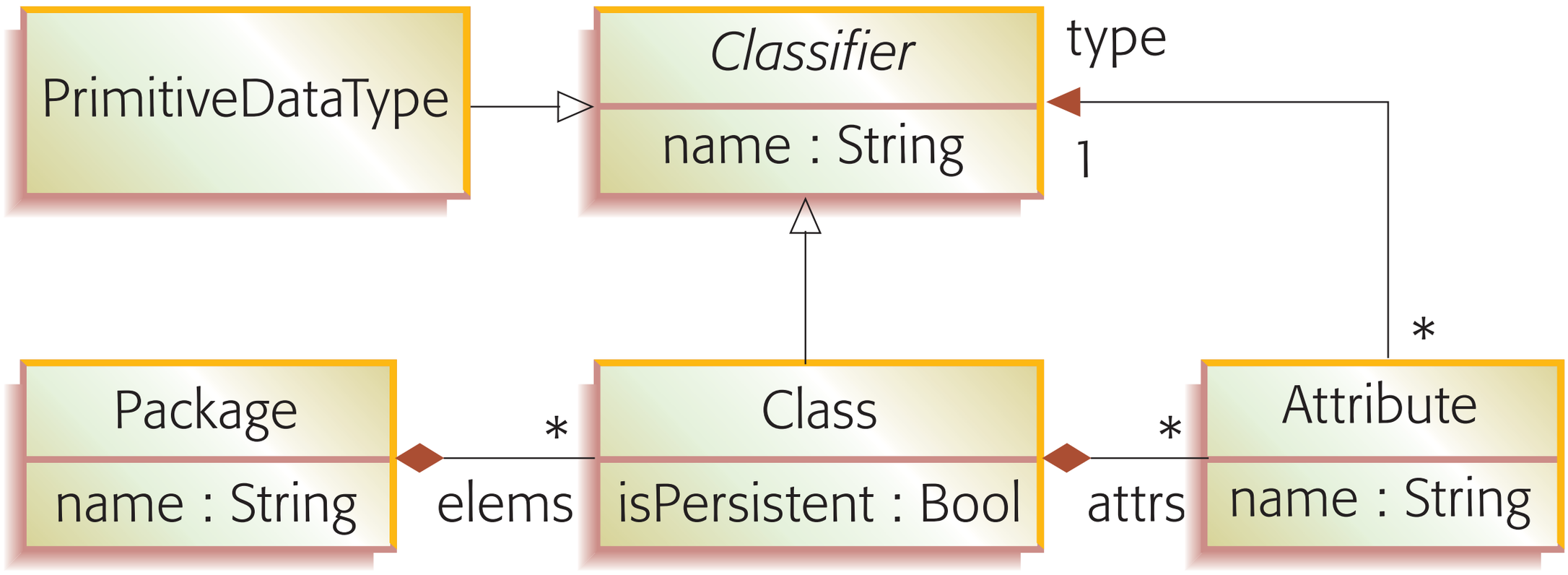
K. Czarnecki
S. Helsen

Model transformations are touted to play a key role in Model Driven Development™. Although well-established standards for creating metamodels such as the Meta-Object Facility exist, there is currently no mature foundation for specifying transformations among models. We propose a framework for the classification of several existing and proposed model transformation approaches. The classification framework is given as a feature model that makes explicit the different design choices for model transformations. Based on our analysis of model transformation approaches, we propose a few major categories in which most approaches fit.

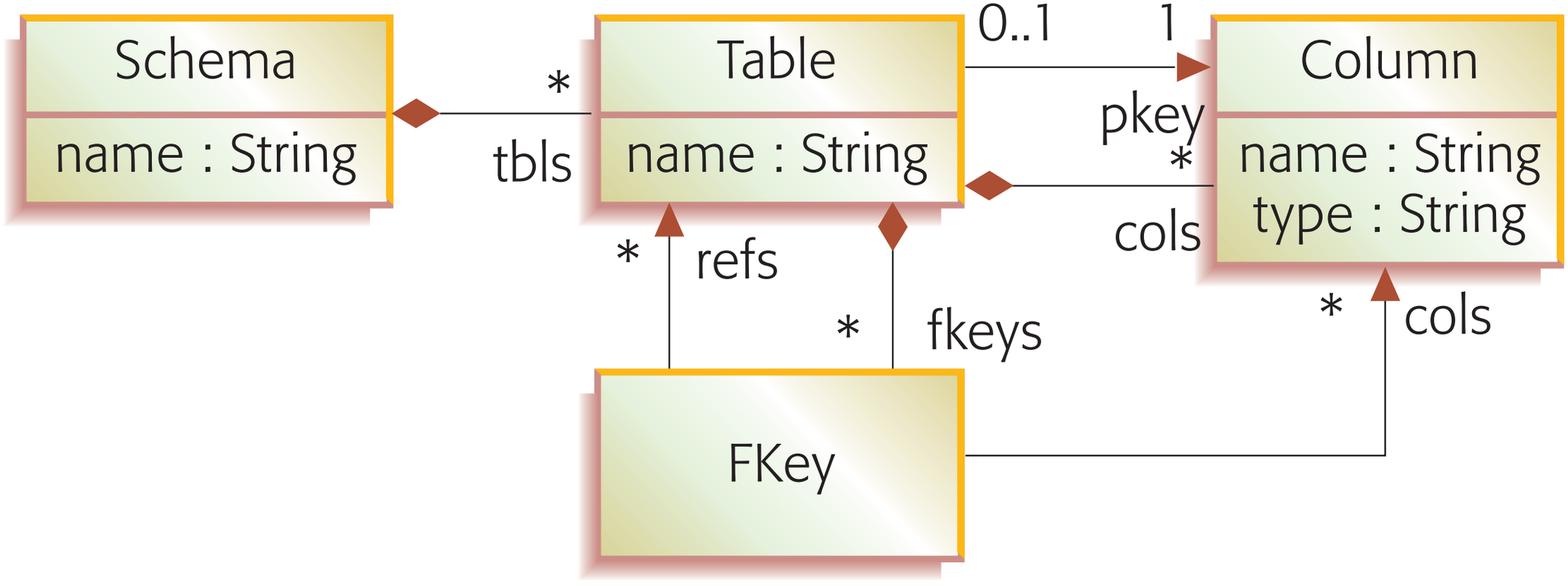
Basic concepts



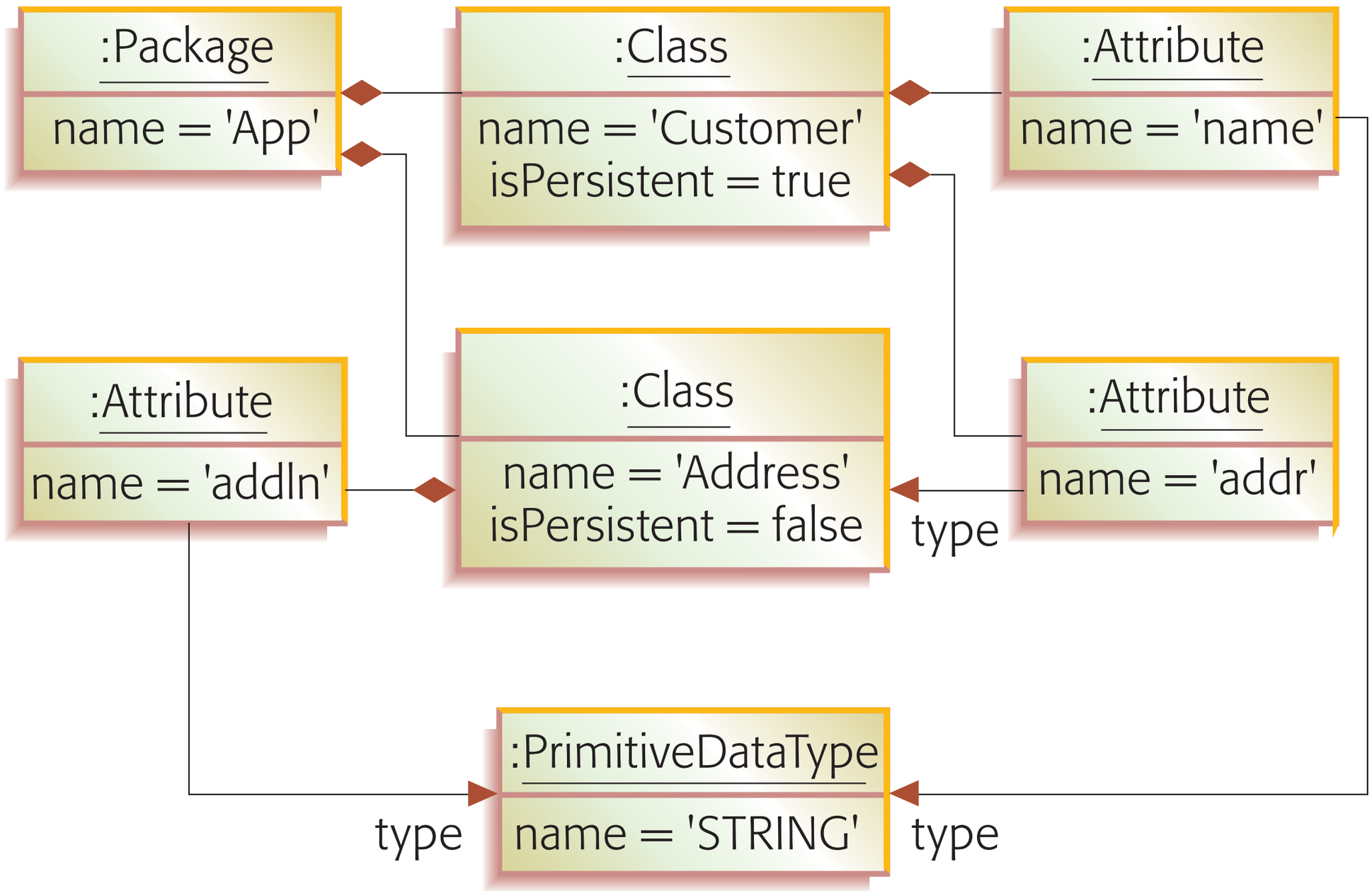
A Simple UML metamodel



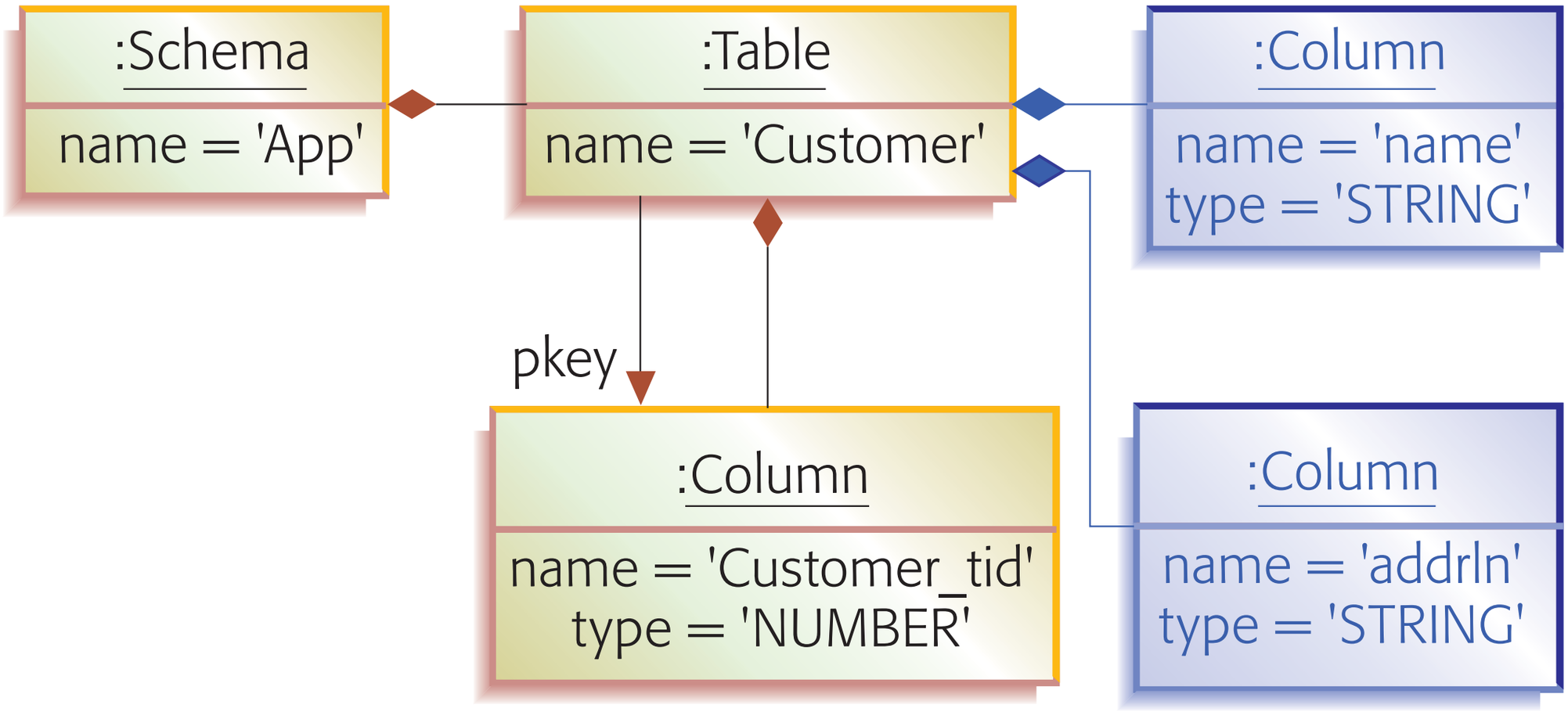
B Simple RDBMS metamodel



C UML sample model



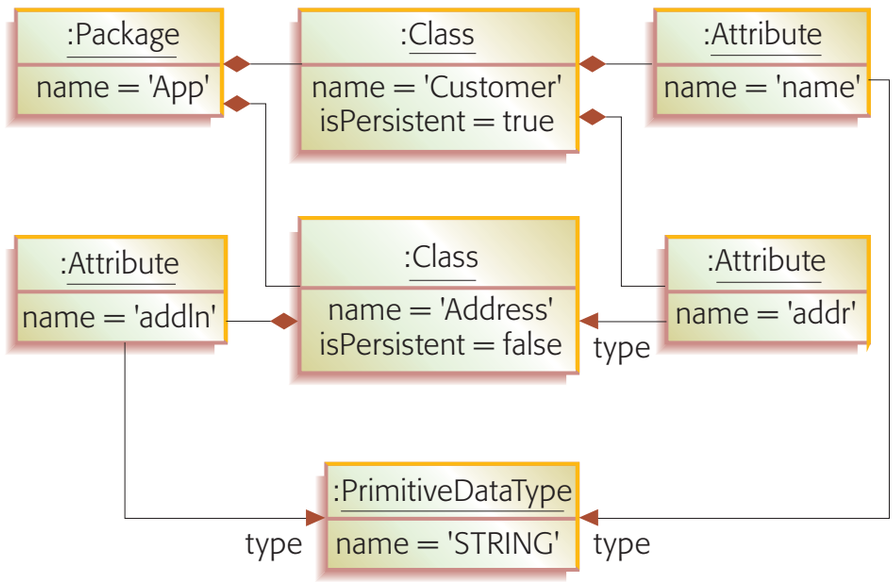
D RDBMS sample model

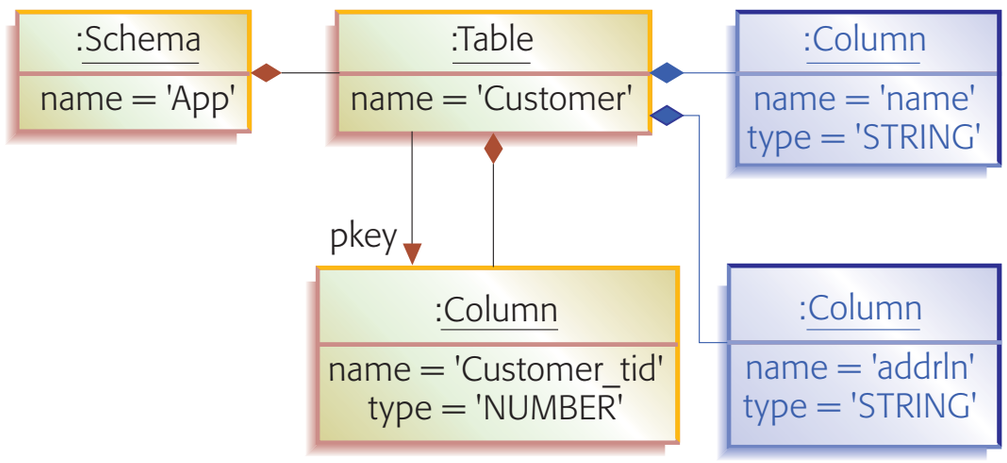


Model transformation

UML

Relational





QVT Relations

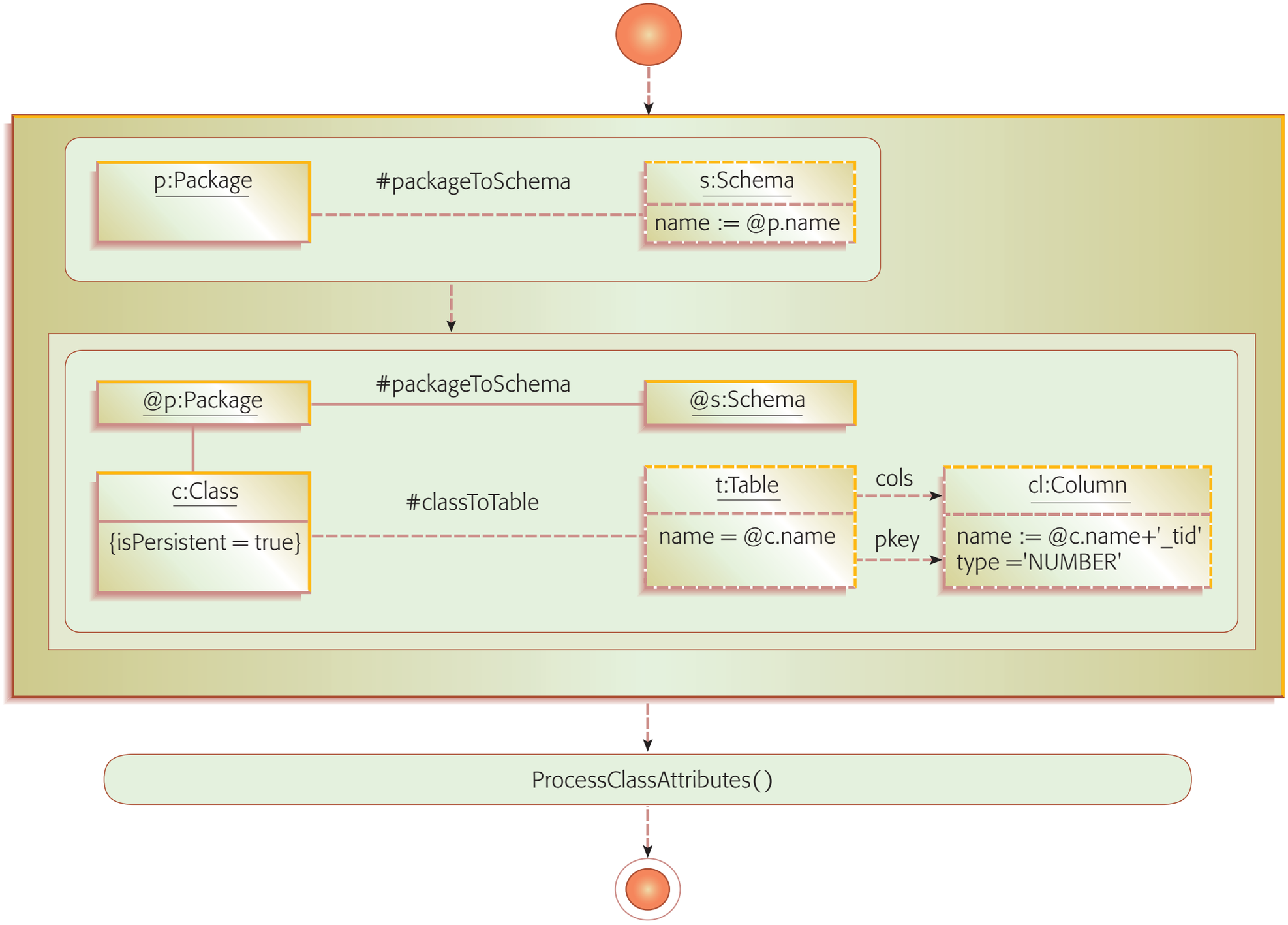
```
transformation umlRdbms {
  uml : SimpleUML, rdbms : SimpleRDBMS) {
  key Table (name, schema);
  key Column (name, table);

  top relation PackageToSchema {
    domain uml p:Package {name = pn}
    domain rdbms s:Schema {name = pn}
}

top relation ClassToTable {
  domain uml c:Class {
    package = p:Package {},
    isPersistent = true,
    name = cn
  }
  domain rdbms t:Table {
    schema = s:Schema {},
    name = cn,
    cols = cl:Column {
      name=cn+'_tid',
      type='NUMBER'},
    pkey = cl
}
when {
  PackageToSchema (p, s);
}
    }
    where {
      AttributeToColumn (c, t);
    }
  }

  relation AttributeToColumn {
  ...
  }
  ...
}
```

Graph transformation (Mola)



Model-to-Text: UML2Java

Templates!

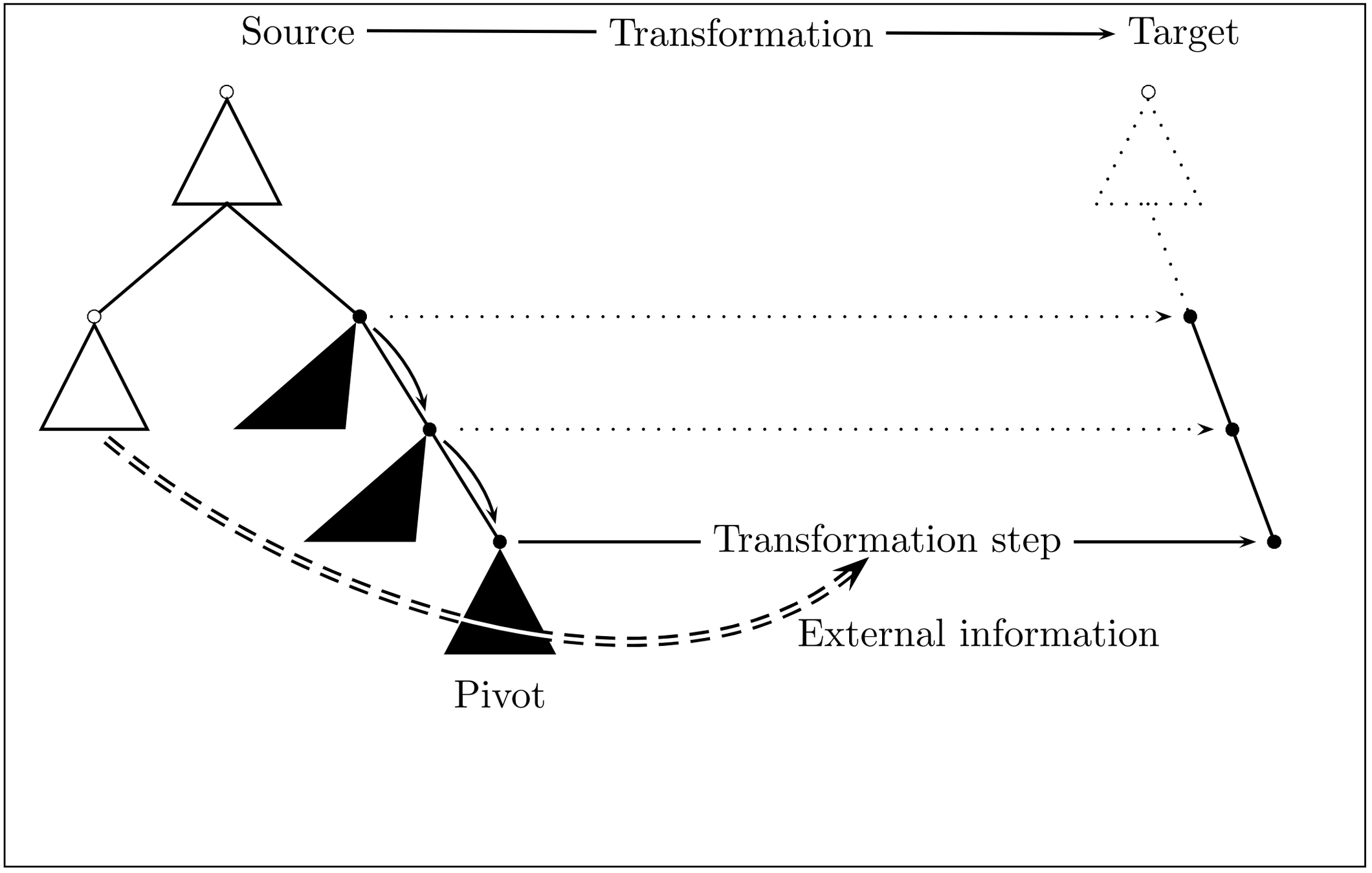
```
<<DEFINE Root FOR Class>>
    public class <<name>> {
      <<FOREACH attrs AS a>>
         private <<a.type.name>> <<a.name>>;
      <<ENDFOREACH>>
      <<EXPAND AccessorMethods FOREACH attribute>>
    }
<<ENDDEFINE>>

<<DEFINE AccessorMethods FOR Attribute>>
    public <<type.name>> get<<name.toFirstUpper>>(){
      return this.<<name>>;
    }
    public void set<<name.toFirstUpper>>(
      <<type.name>> <<name>> ) {
      this.<<name>> = <<name>>
    }
<<ENDDEFINE>>
```

Program Transformation Mechanics

A Classification of Mechanisms for Program Transformation with a Survey of Existing Transformation Systems

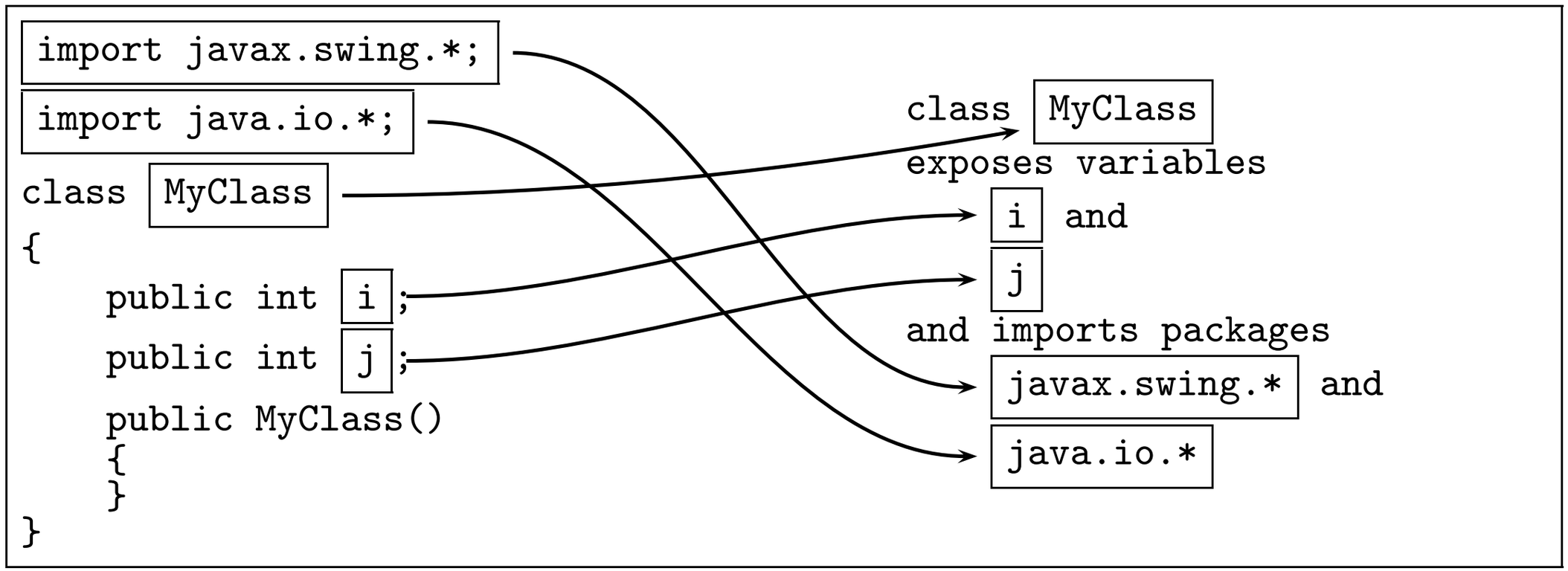
Jonne van Wijngaarden
Eelco Visser


Source
Transformation
Target
Transformation step
External information
Pivot

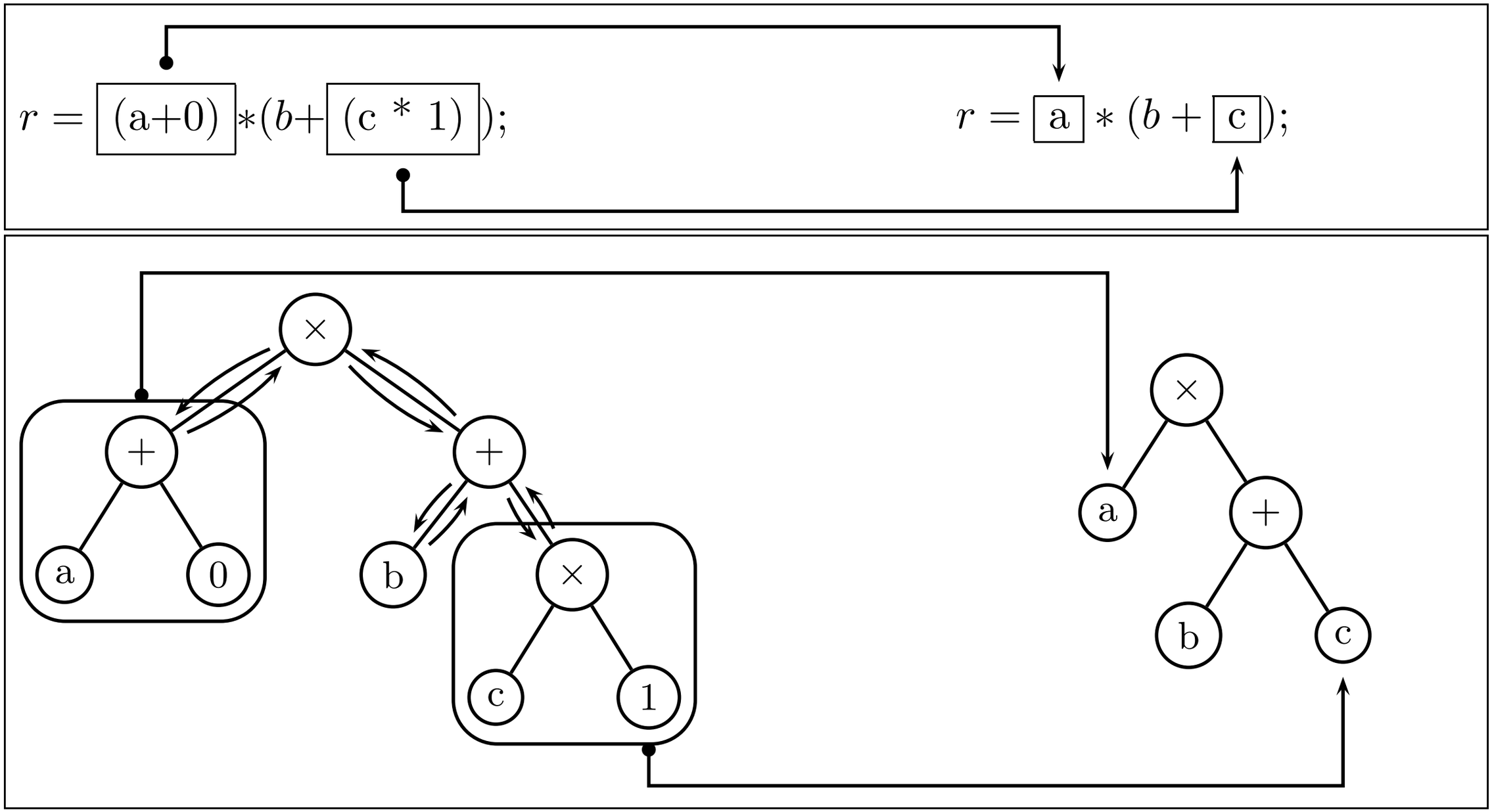
Transformation scope

- Local source to local target
- 1-to-1
- Global source to local target
- Local source to global target
- Global source to global target

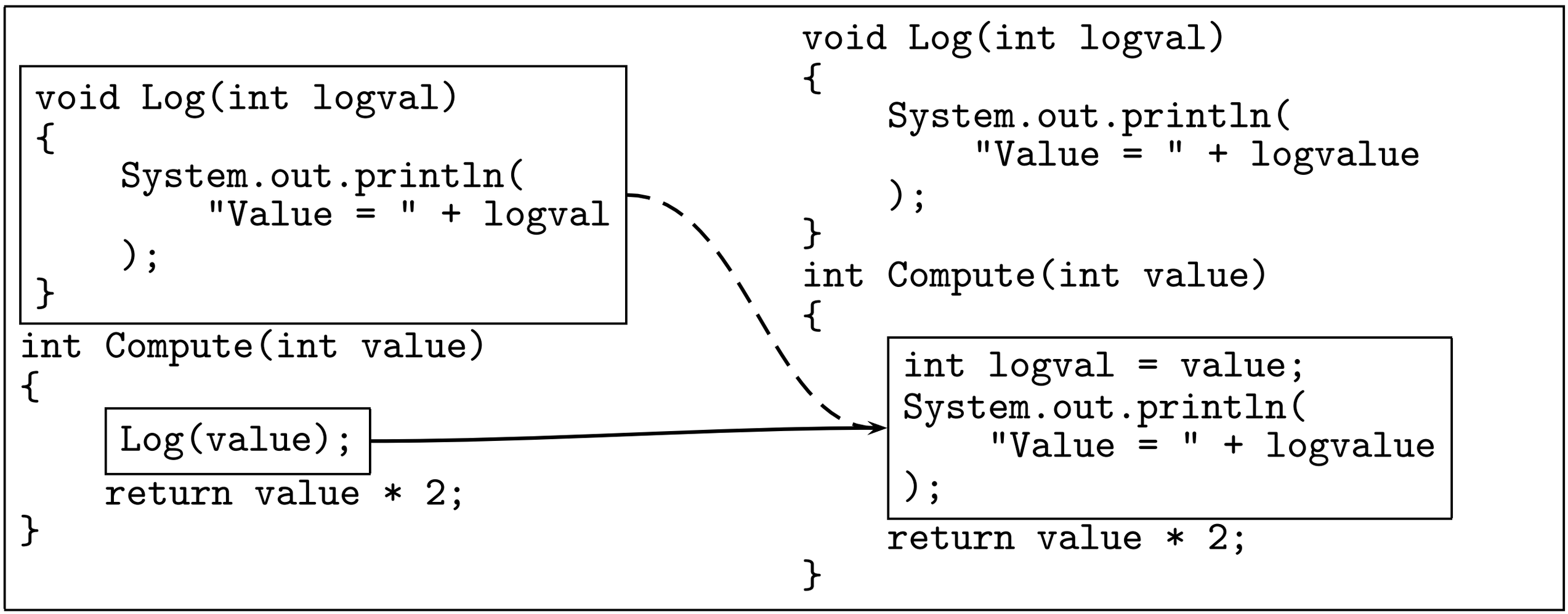
Local to local: documentation generation



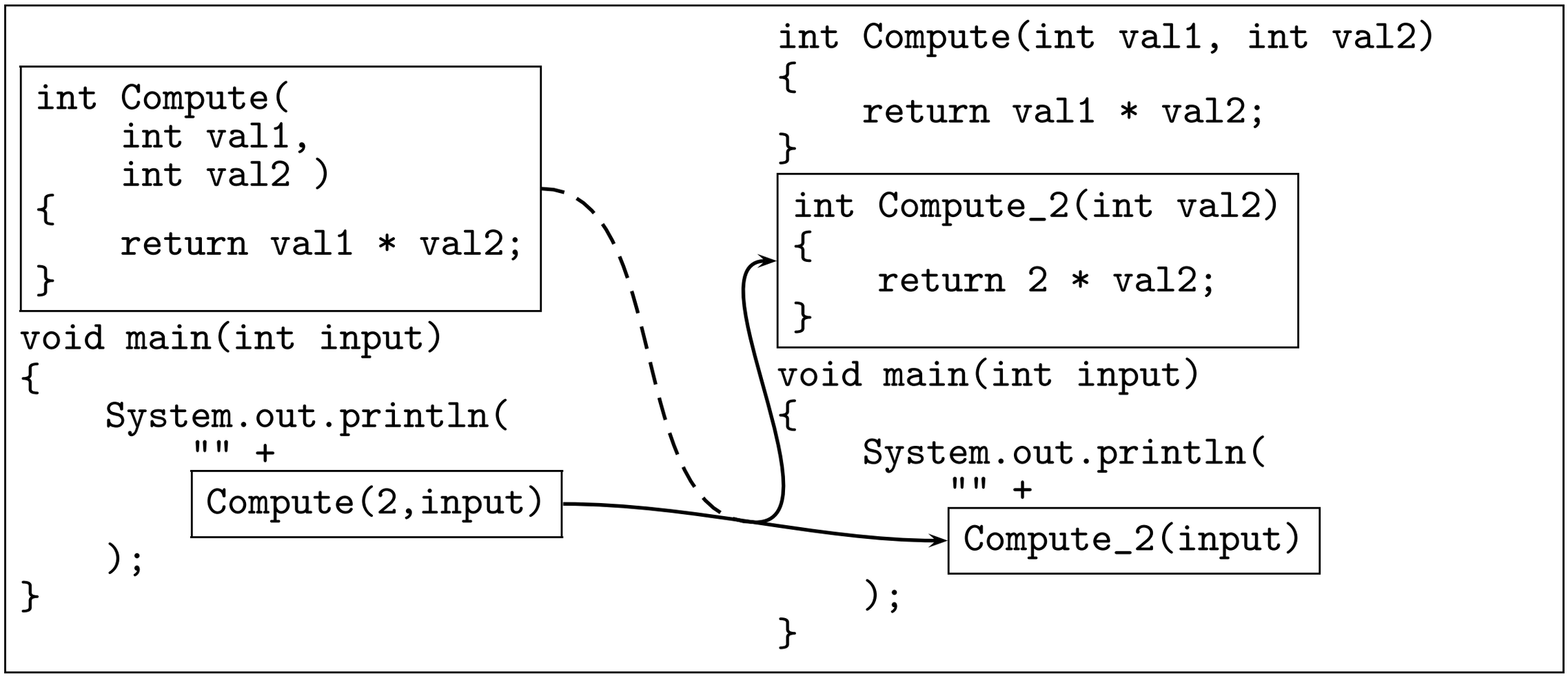
1-to-1:
expression simplification



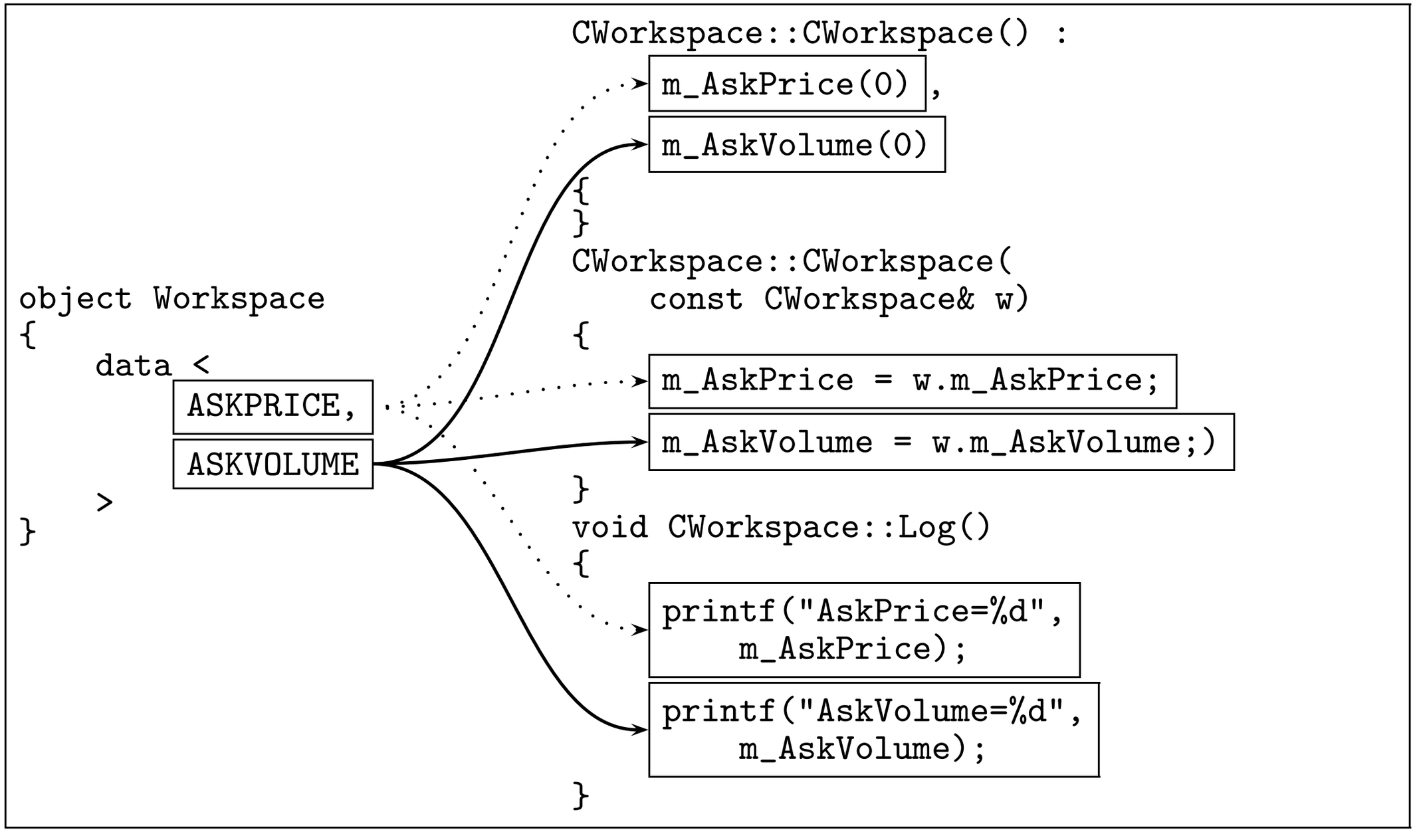
Global to local function inlining



Global to global function specialization



Local to global
DSL code generation



A Taxonomy of Model Transformation

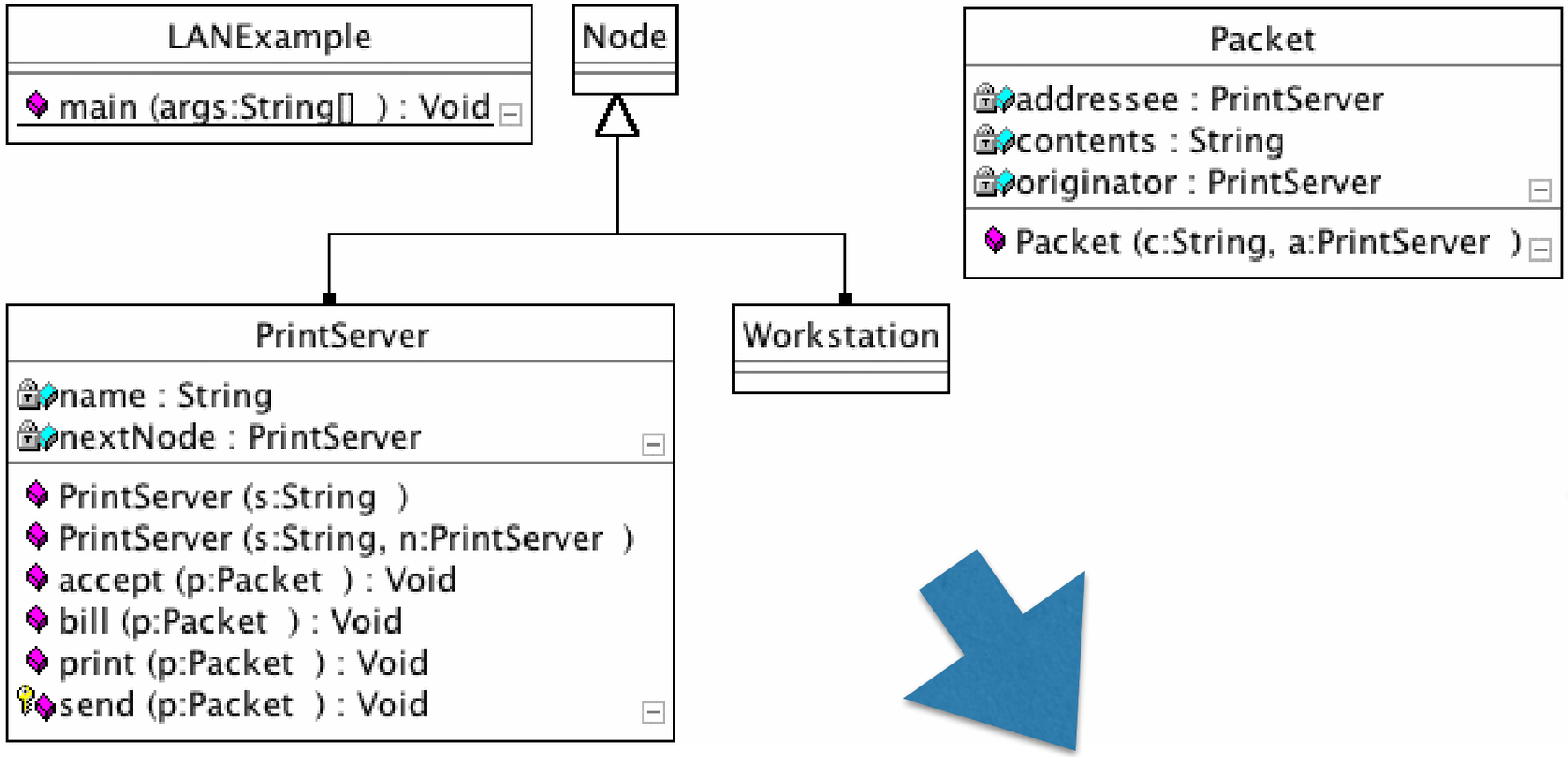
Tom Mens[1]

Software Engineering Lab
Université de Mons-Hainaut
Mons, Belgium

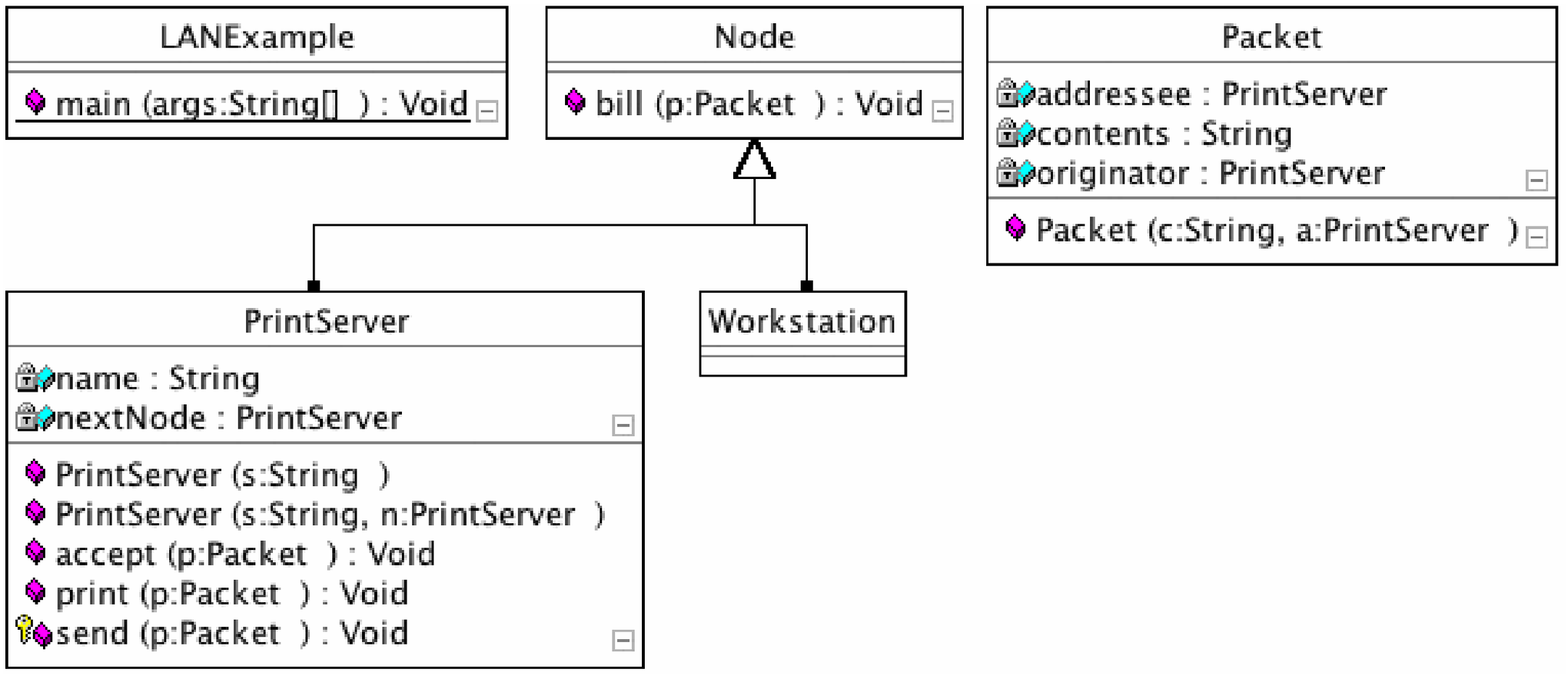
Pieter Van Gorp[2]

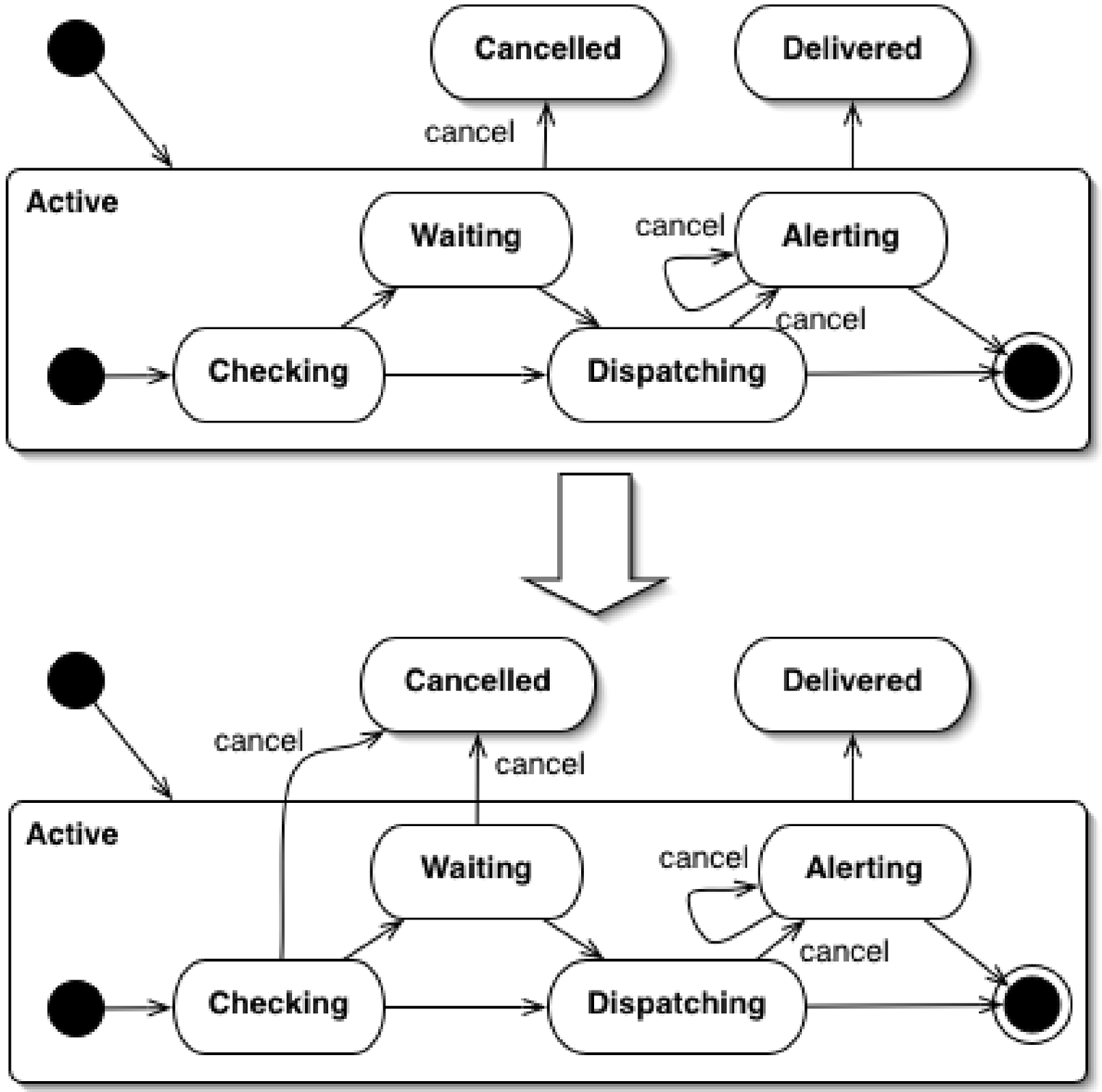
Formal Techniques in Software Engineering
Universiteit Antwerpen
Antwerpen, Belgium

	horizontal	vertical
endogenous	*Refactoring*	*Formal refinement*
exogenous	*Language migration*	*Code generation*



Pull up method





flatten state chart

Name	Cat	Kind	Description
family2persons	Exo	M2M	Extract persons from a family
family2graph	Exo	M2M	Convert family to graph
renameEvent	Endo	M2M	Rename events in SM (Section 2.3)
addResetTransitions	Endo	M2M	Add transitions to SM (Section 2.3)
regexp2Statemachine	Exo	M2M	Convert regular expression to SM
statemachine2DFA	Endo	M2M	Determinize state machine
parallelMerge	Endo	M2M	Merge states in SM
statemachine2Graph	Exo	M2M	Convert SM to graph
flattenInheritance	Endo	M2M	Push down fields (Section 2.3)
generalizeTypeRefs	Endo	M2M	Change field types to largest super class.
metaModel2Relational	Exo	M2M	MM to relational schema
metaModel2Java	-	M2T	From MM to Java code (text)
metaModel2Graph	Exo	M2M	Convert MM to graph
metaModel2ADT	Exo	M2M	Convert MM to ADT
source2Activity	-	T2M	Textual activity model to Activity Model
activity2Graph	Exo	M2M	Activity model to graph
executeActivity	Endo	M2M	Execute Activity Model

Paul Klint, Tijs van der Storm, “Model Transformation with Immutable Data”, ICMT’16

Transformation tools

- Program transformation
  - ASF+SDF
  - TXL
  - Stratego
  - Kiama
  - Rascal ;-)
  - …
- Model transformation
  - QVT
  - Epsilon
  - ATL
  - Viatra
  - …

Summary

- Basic concepts of model transformation
- Scope: local-to-local, global-to-local, etc.
- Endogenous vs exogenous
- Horizontal vs vertical
- Text2model (parsing)
- Model2text (code generation)
- Tools: program transformation/model transformation

Transformations in QL?

- Parsing
- CST2AST
- Name resolution
- Type checking (?)
- Code generation
- Normalize
- Refactoring

Next up: rename refactoring for state machines