# A Formal Approach to Modeling and Model Transformations in Software Engineering

Adrian Rutle

Department of Computer Engineering
Faculty of Engineering
Bergen University College

17.06.2008
EXPO 2008, Integration and Architecture, Bergen, Norway

# History of the Project

- **2003** Wolter and Lamo: decided to use Generalized Sketches approach in software engineering
- **2004** Ørjan Hatland (master student):
  - implementation of tools for GS
  - .NET framework
- **2006** Adrian Rutle (PhD student):
  - MDE, Eclipse platform
  - DPL
- **2006** Stian Skjerveggen (master student):
  - implementation of graphical editor (EMF/GMF)
- **2008** Thomas Slï¿½k Tvedt (master student):
  - serialization of diagrammatic specifications
  - continuation of Stian's tools

# Outline

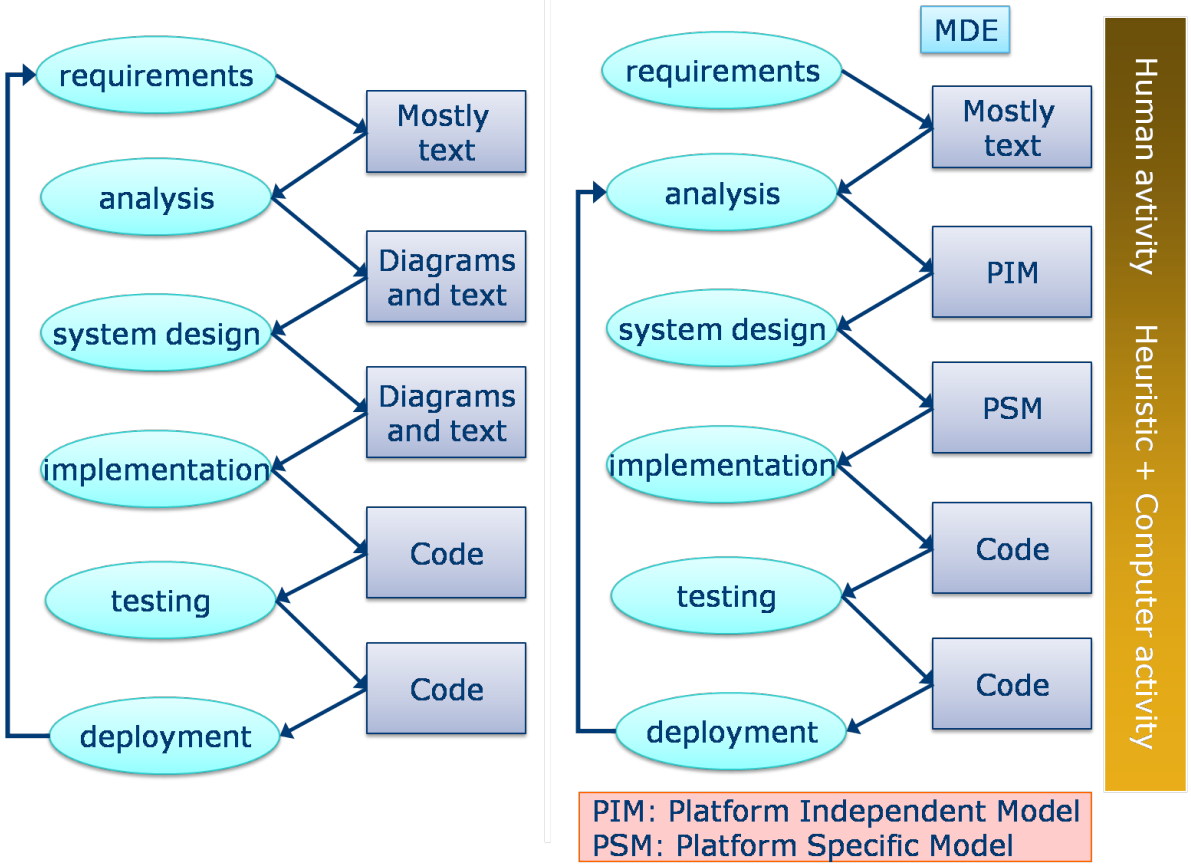
# MDE

- Engineering techniques where models are first-class entities
- Evolved from the popularity of diagrammatic languages such as UML and ER and their popularity for specification and documentation of software systems
- Aims to raise the abstraction level of software development from text to models
- The development process is based on software models, not on code

# Industry "standard" for MDE

- Model Driven Architecture (MDA) approach from Object Management Group (OMG) where:
  - First step is building platform independent domain models (PIM)
  - PIMs are refined to platform specific models (PSM)
  - Refinement process more or less automatized as model transformations
  - Code generation and model refinement can be seen as model transformations
  - Reuse of software models by using model transformations

# MDE vs Traditional Development Processes



# Key-points for the Success of MDE

Models are no longer only for documentation

- Models should be **both**:
  - intuitive to ease their interception by **software engineers** and **domain experts**
  - formal such that **machines** can have a precise understanding of (and able to reason about) the models

⇒ Diagrammatic approach enhances human understanding
⇒ One challenge is to combine intuition with formalization
⇒ Another challenge is to implement the formalization approach

# Key-points for the Success of MDE

Models are no longer only for documentation

- Models should be **both**:
  - intuitive to ease their interception by **software engineers** and **domain experts**
  - formal such that **machines** can have a precise understanding of (and able to reason about) the models

⇒ Diagrammatic approach enhances human understanding

⇒ One challenge is to combine intuition with formalization

⇒ Another challenge is to implement the formalization approach

# Key-points for the Success of MDE

Models are no longer only for documentation

- Models should be **both**:
  - intuitive to ease their interception by **software engineers** and **domain experts**
  - formal such that **machines** can have a precise understanding of (and able to reason about) the models

⇒ Diagrammatic approach enhances human understanding

⇒ One challenge is to combine intuition with formalization

⇒ Another challenge is to implement the formalization approach

# Key-points for the Success of MDE

Models are no longer only for documentation

- Models should be **both**:
  - intuitive to ease their interception by **software engineers** and **domain experts**
  - formal such that **machines** can have a precise understanding of (and able to reason about) the models

⇒ Diagrammatic approach enhances human understanding
⇒ One challenge is to combine intuition with formalization
⇒ Another challenge is to implement the formalization approach

# Outline

# Some facts about UML

- UML is considered *de facto* industry standard for modeling
- UML can only express constraints on binary relations
- UML has serious issues regarding:
  - Semantics; UML models may be ambiguous and have semi-formal semantics
  - Complexity; UML uses 13 different types of diagrams
  - Expressibility; To express constraints over higher order relations one need to use string based logic, (Object Constraint Logic, OCL)

# Some facts about Eclipse Modeling Framework

- EMF is considered a low-cost entry to the employment of model-driven engineering
- EMF is used increasingly by companies which adopt the Eclipse platform
- The metamodel of EMF is very close to Essential MOF
- Less complex $\Leftrightarrow$ clearer semantics
- Not just a standard, but an implementation
- Uses XMI for serialization of models

# Formal Modeling Languages

- Precise semantics, proven abilities and several fundamental Software Engineering problems solved by use of formal methods
- Set based semantics (logic, algebras etc...)
- Hard for software engineers to apply in practice
- Lack of good software development tools based on formal approaches
- Only used by well trained experts
- High cost $\Leftrightarrow$ low productivity

# Outline

# DPL

- Aims to combine the intuition from graphical modeling languages with the semantic rigor of formal methods
- Based on Generalized Sketches
- Sketches are graphical representations of category theory, constraints expressed by universal constructions (limits and co-limits) of diagrams
- Generalized Sketches are sketches extended with constraints as arbitrary constructions, not only universal constructions
- In DPL, sketches are generalized and adapted to model driven software engineering

# More on DPL

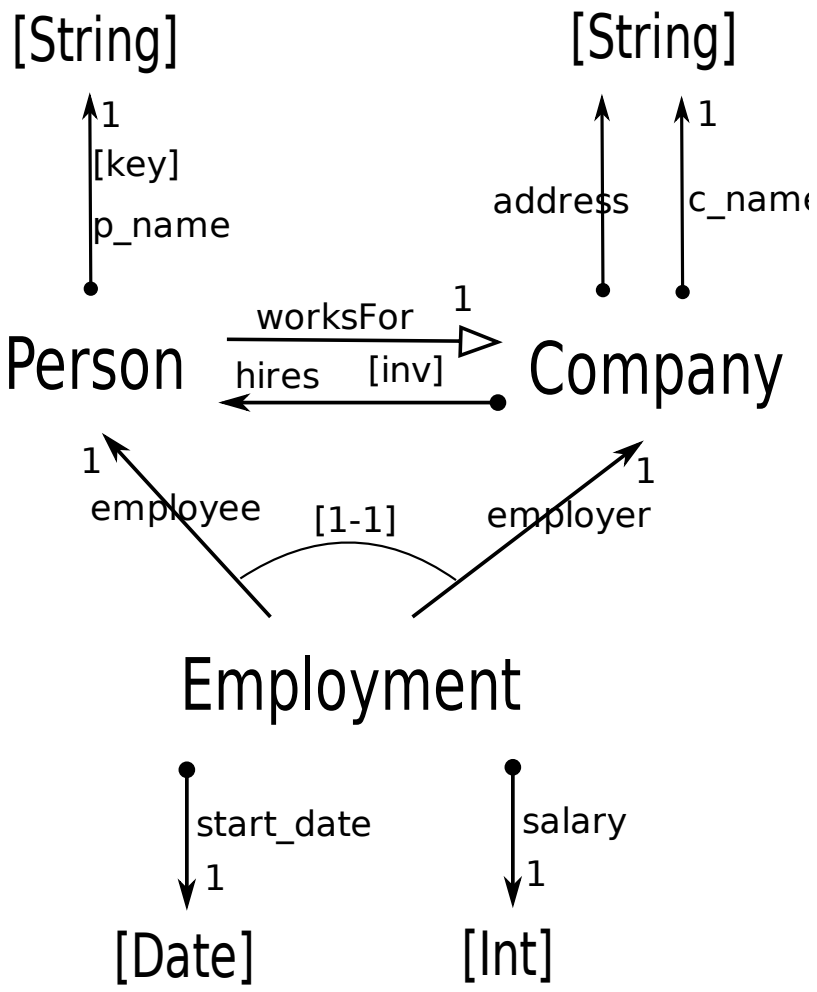
- DPL will be used as a framework for defining modeling languages and transformations between them in a formal, graphical way
- We work continuously on Investigating, Analyzing, Adapting, Verifying and Evaluating DPL
- Potentials to use the machinery from category theory to implement MDE tools

# Outline

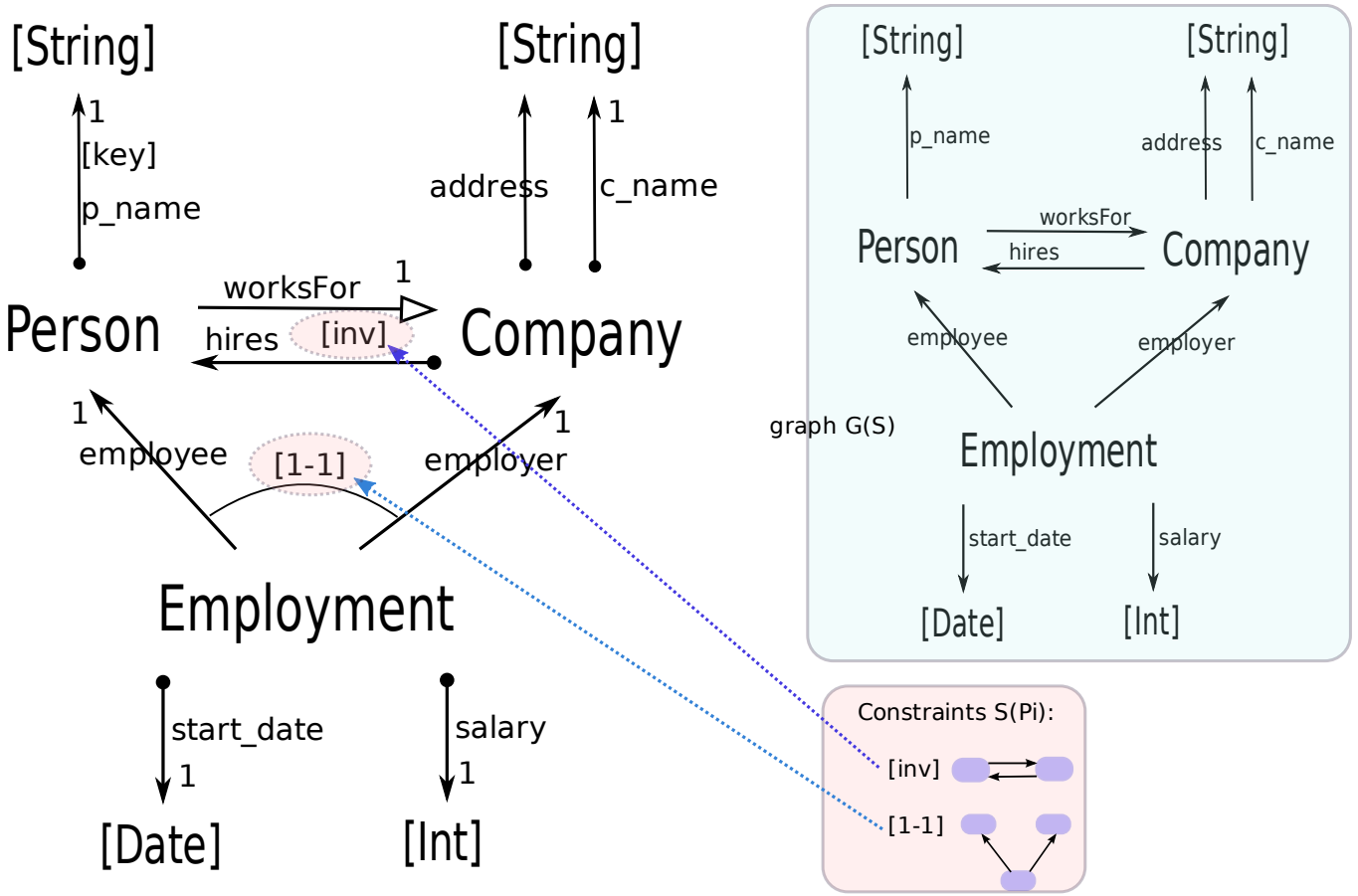
# Signature – Example

| $\Pi$ | $\alpha$ | visualization | intended semantics |
|---|---|---|---|
| [total] | $1 \xrightarrow{x} 2$ | $A \xrightarrow{f} B$ | $\forall a \in A : \lvert f(a) \rvert \geq 1$ |
| [key] | $1 \xrightarrow{x} 2$ | $A \xrightarrow{f\ [key]} B$ | $\forall a, a' \in A : a \neq a'$ implies $f(a) \neq f(a')$ |
| [inv] | $1 \underset{y}{\overset{x}{\rightleftarrows}} 2$ | $A \underset{g}{\overset{f}{\underset{[inv]}{\rightleftarrows}}} B$ | $\forall a \in A$ , $\forall b \in B : b \in f(a)$ iff $a \in g(b)$ |
| [jointly-key] or [1-1] | $\begin{tikzcd} 1 \arrow[r, "x"] \arrow[d, "y"] & 2 \\ 3 & \end{tikzcd}$ | $\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "g"'] & B \arrow[dl, bend left, "{[1-1]}"] \\ C & \end{tikzcd}$ | $\forall a, a' \in A :\ a \neq a'$ *implies* $f(a) \neq f(a')$ or $g(a) \neq g(a')$ |

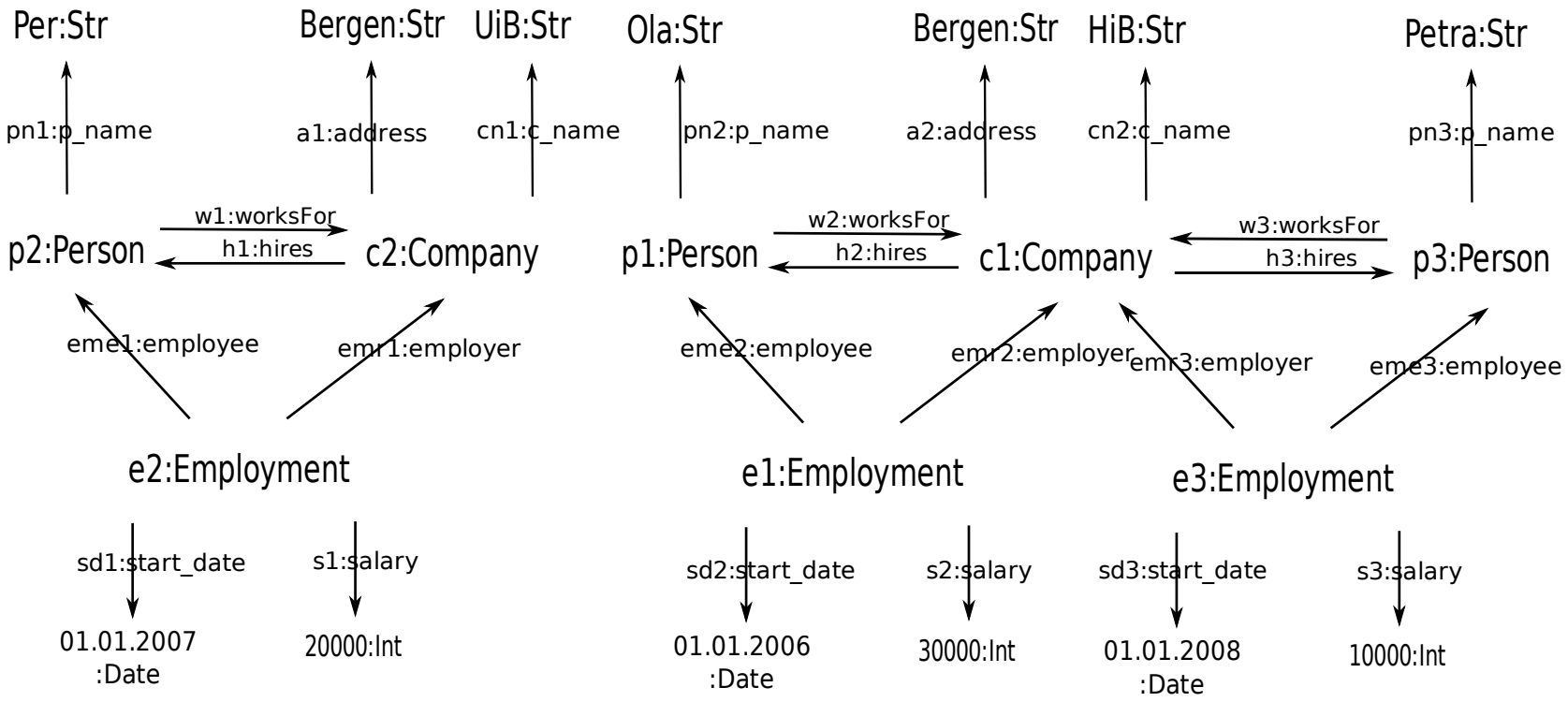
# Diagrammatic Specification – Example



# Diagrammatic Specification – Example



# Instance of Diagrammatic Specification – Example



# Outline

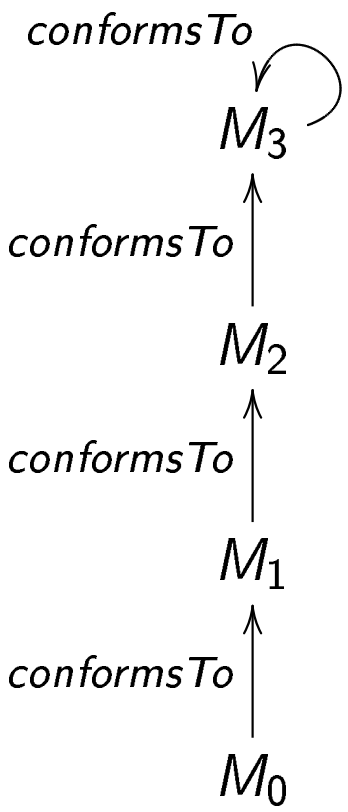
# OMG Metamodel Levels

| OMG levels | OMG Standards/examples |
|---|---|
| $M_3$: Meta-metamodel | MOF |
| $M_2$: Metamodel | UML language |
| $M_1$: Model | A UML model: Class "Person" with attributes "name" and "address" |
| $M_0$: Instance | An instance of "Person": "Ola Nordmann" living in "Sotraveien 1, Bergen" |

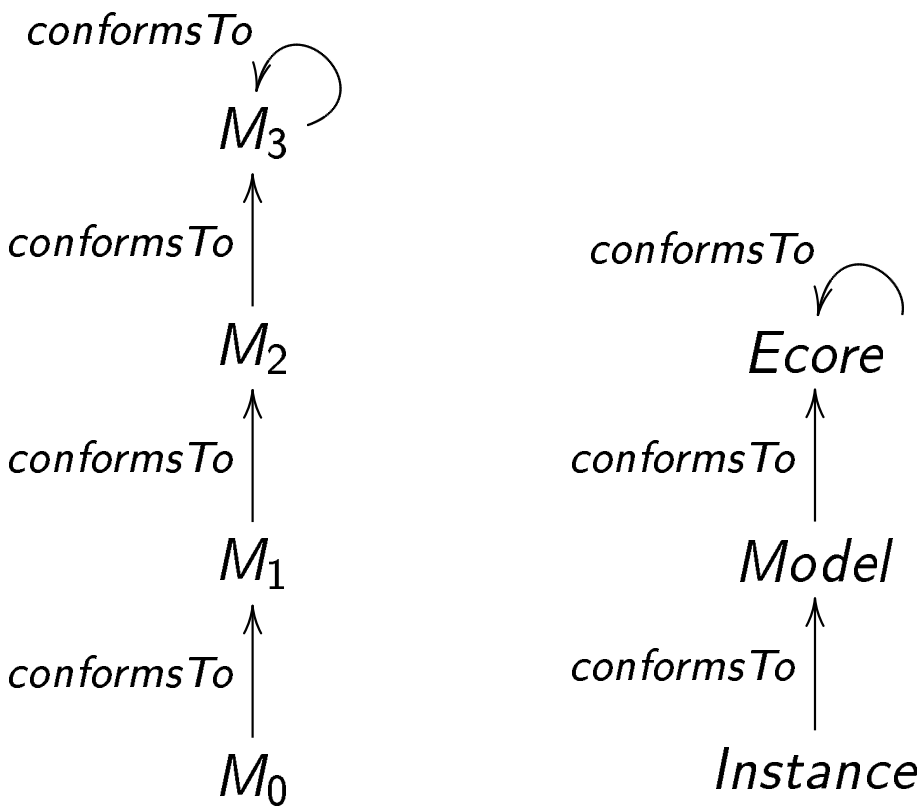
# OMG Metamodel Levels and EMF

| OMG levels | EMF/examples |
|---|---|
| $M_3$: Meta-metamodel | Ecore |
| $M_2$: Metamodel | Ecore |
| $M_1$: Model | An Ecore model: `EClass` "Person" with `EAttributes` "name" and "address" |
| $M_0$: Instance | An instance of "Person": "Ola Nordmann" living in "Sotraveien 1, Bergen" |

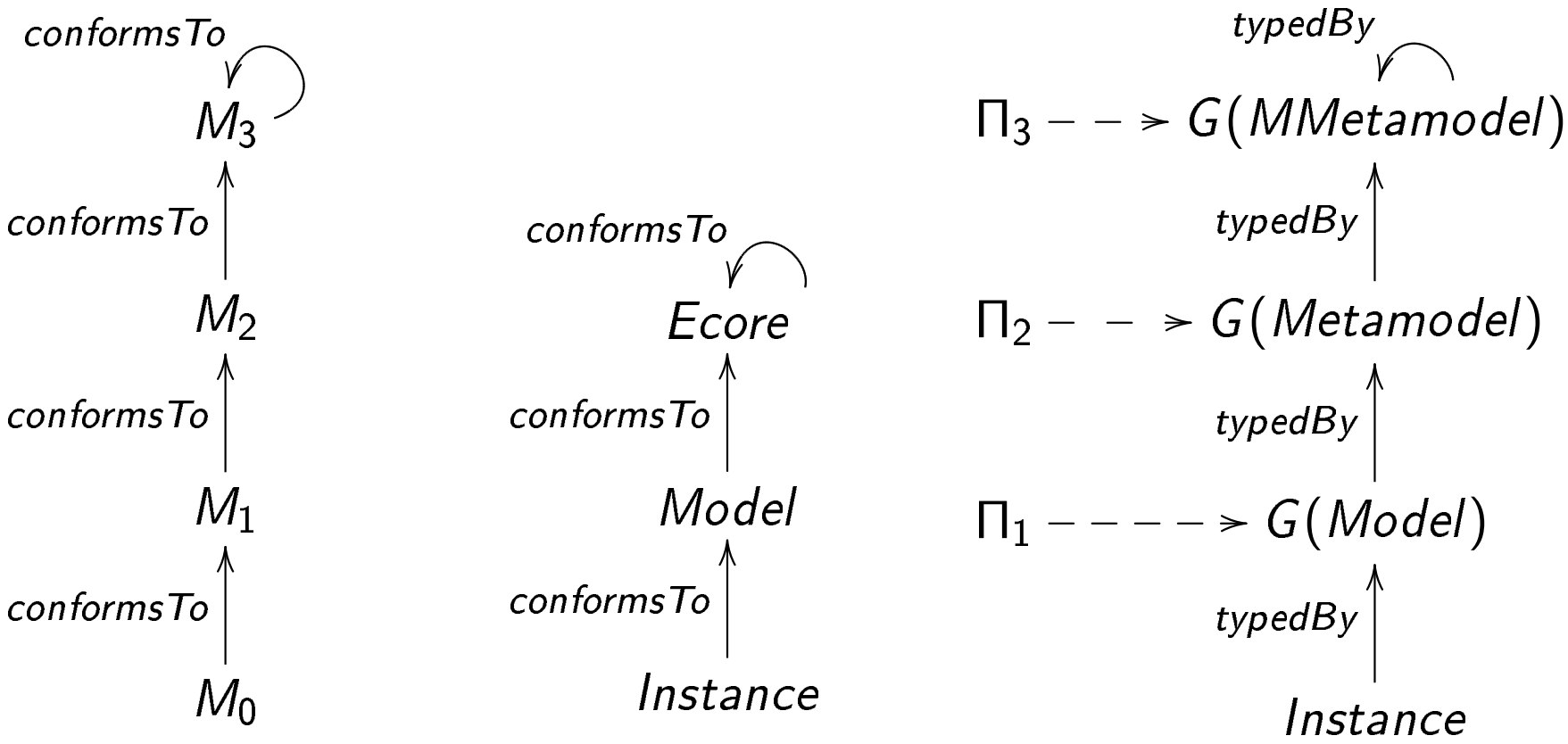
# DPL Methodology and Modeling Formalisms



# DPL Methodology and Modeling Formalisms

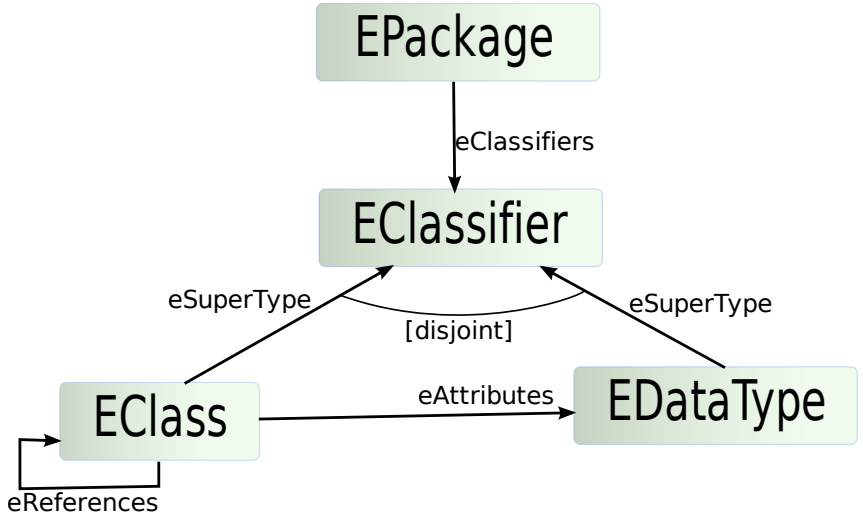


# DPL Methodology and Modeling Formalisms

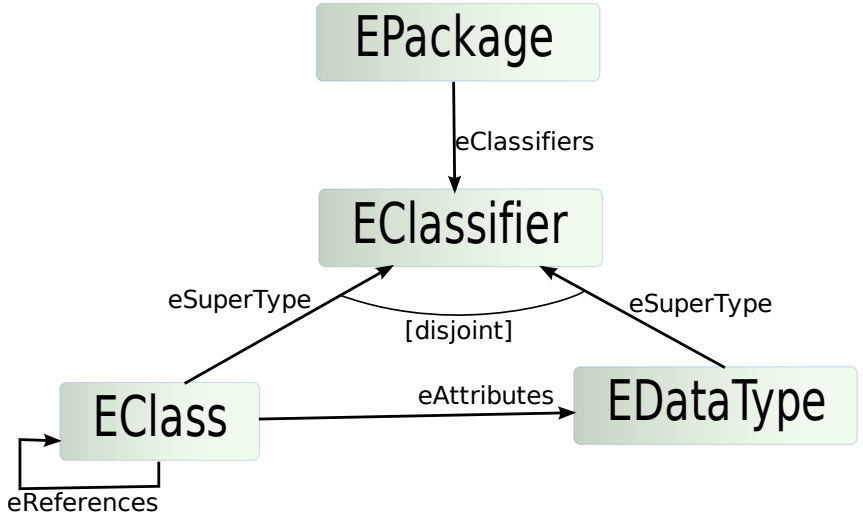


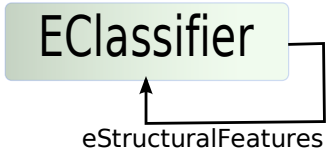
# Outline

# Ecore model expressed in DPL

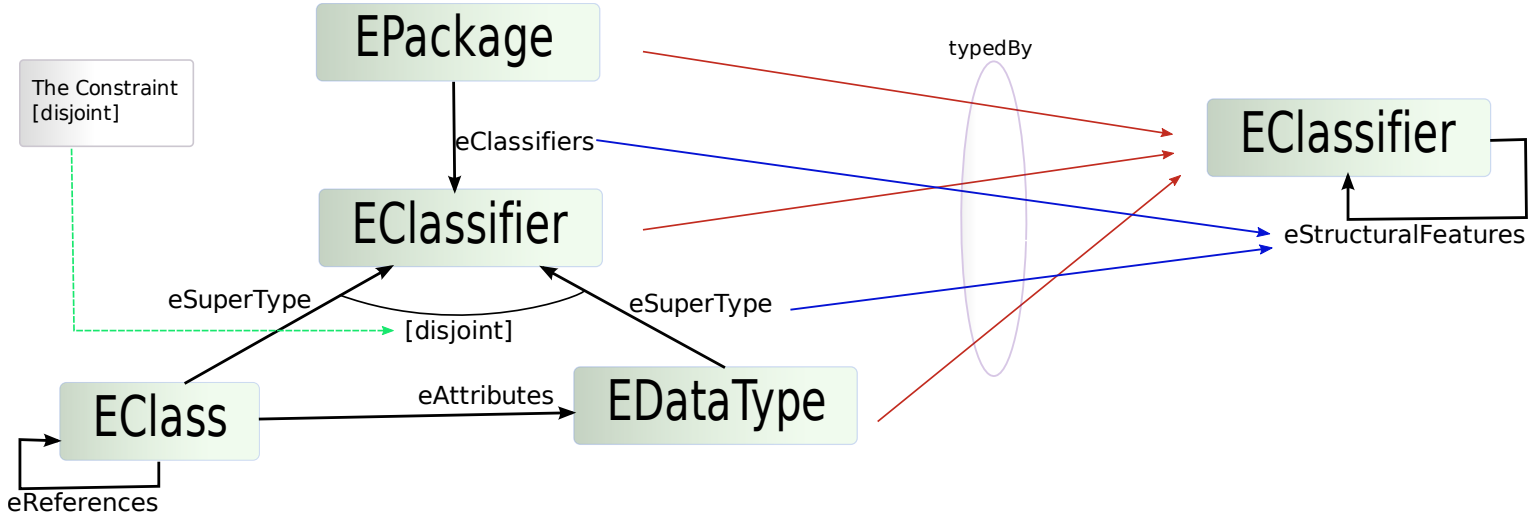


# Ecore model expressed in DPL



# Ecore model expressed in DPL



# Outline

# Model Transformations

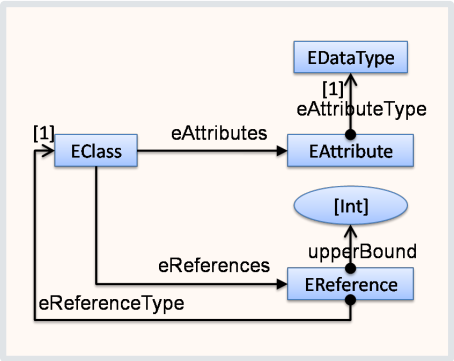
- Generation of target models from source models
- Examples:
  - Development process: Code generation, refinement etc
  - Model management: integration, decomposition etc
  - Migration: from a platform/implementation technology/programming language to another
  - Technology mappings: Java classes to Relational tables, objects to rows in database tables

# Outline

# Model Transformation Concepts

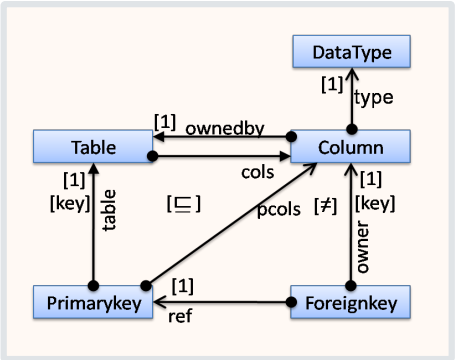
- Model Transformation = set of Transformation Rules + Coordinations ($\Rightarrow$ operational)
- Model Transaction = declarative definition of the transformation
- Transformation Engine = executes the rules

# Model Transformation

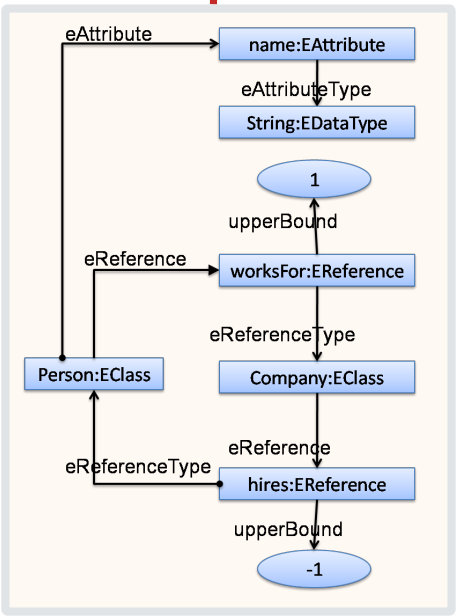



MT Language

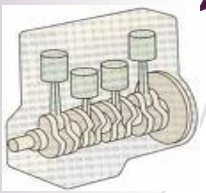
MT Definition




conformsTo

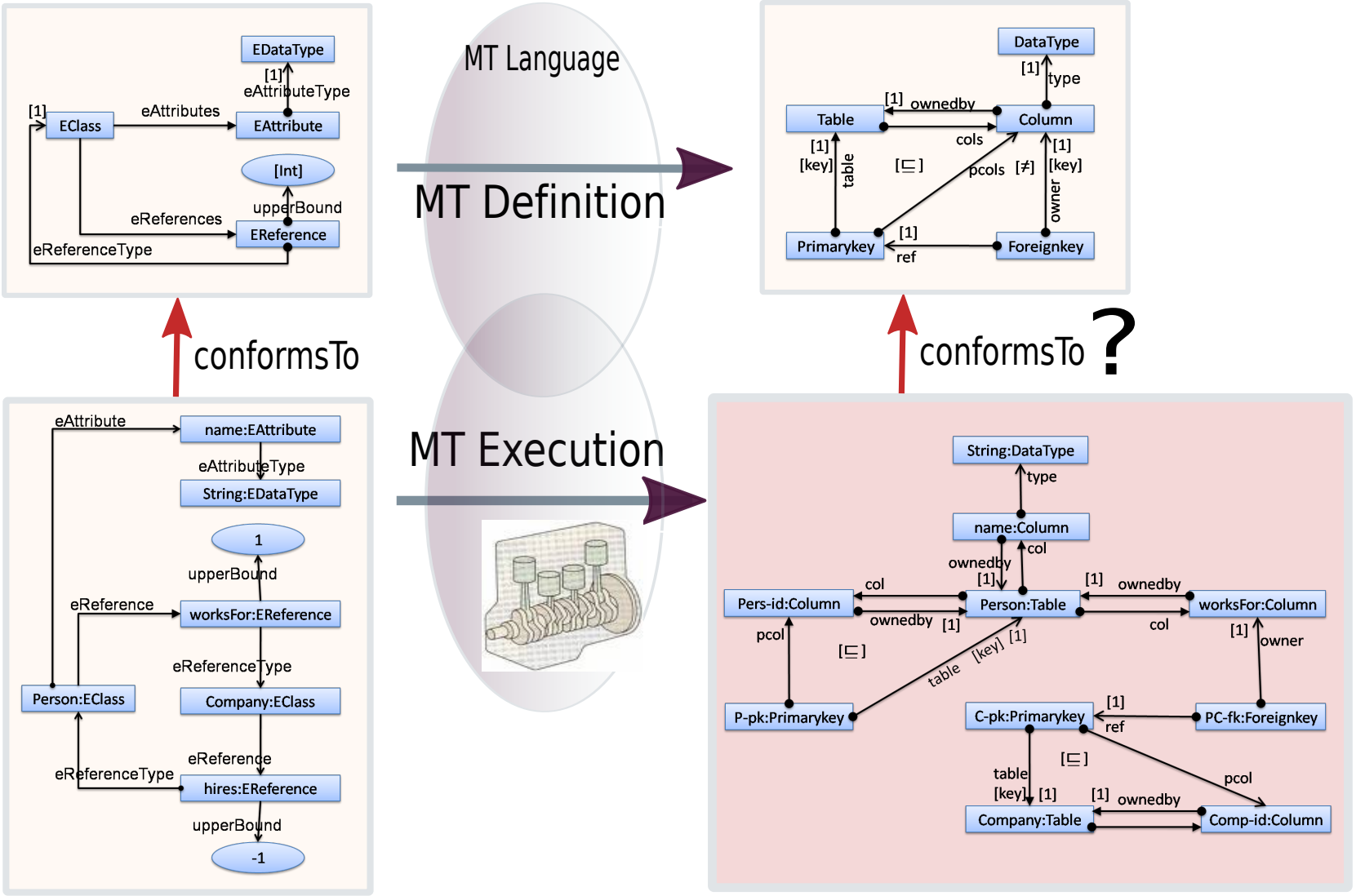



MT Execution



?

# Model Transformation



# Transformation Rules

$$
\begin{tikzcd}
 & P \arrow[dl, "\iota_P"'] \arrow[d] \arrow[r, mapsto, "m"] \arrow[dr, phantom, "[=]"] & \llbracket t \rrbracket(m) \arrow[d] & P' \arrow[d] \arrow[dr, "\iota_{P'}"] \arrow[dl, phantom, "[=]"] & \\
G(M_1) & I \arrow[l, "\iota"] \arrow[rr, dotted, "trace"'] & & I' \arrow[r, "\iota'"'] & G(M_2)
\end{tikzcd}
$$

- Input and output patterns $P$ and $P'$
- Input and output instances $I$ and $I'$
- The models $M_1$ and $M_2$
- The matches $m$ and $\llbracket t \rrbracket(m)$

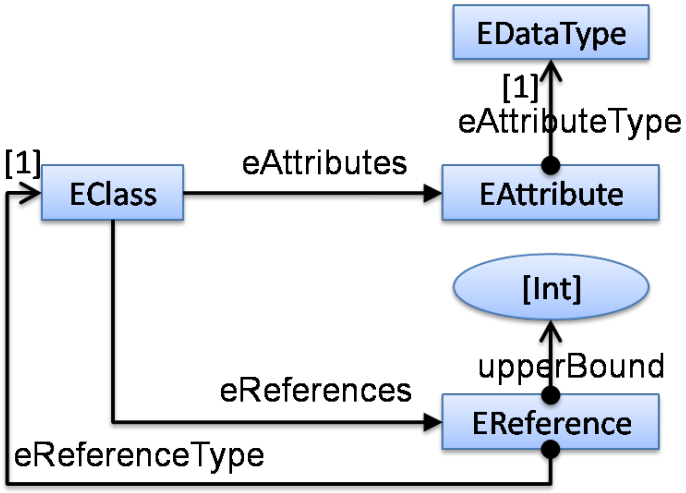
# Transformation Rules

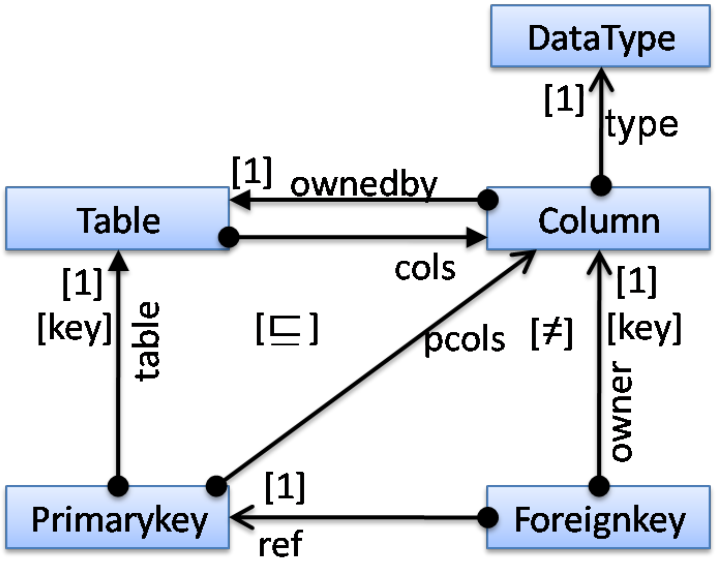
$$
\begin{tikzcd}
 & P \arrow[dl, "\iota_P"'] \arrow[d, "m"] \arrow[r, mapsto] \arrow[dr, phantom, "[=]"] & P' \arrow[d, "[\![t]\!](m)"] \arrow[dr, "\iota_{P'}"] \arrow[dr, phantom, "[=]"] & \\
G(M_1) & I \arrow[l, "\iota"] \arrow[r, dotted, "trace"'] & I' \arrow[r, "\iota'"'] & G(M_2)
\end{tikzcd}
$$

- Input and output patterns $P$ and $P'$
- Input and output instances $I$ and $I'$
- The models $M_1$ and $M_2$
- The matches $m$ and $[\![t]\!](m)$

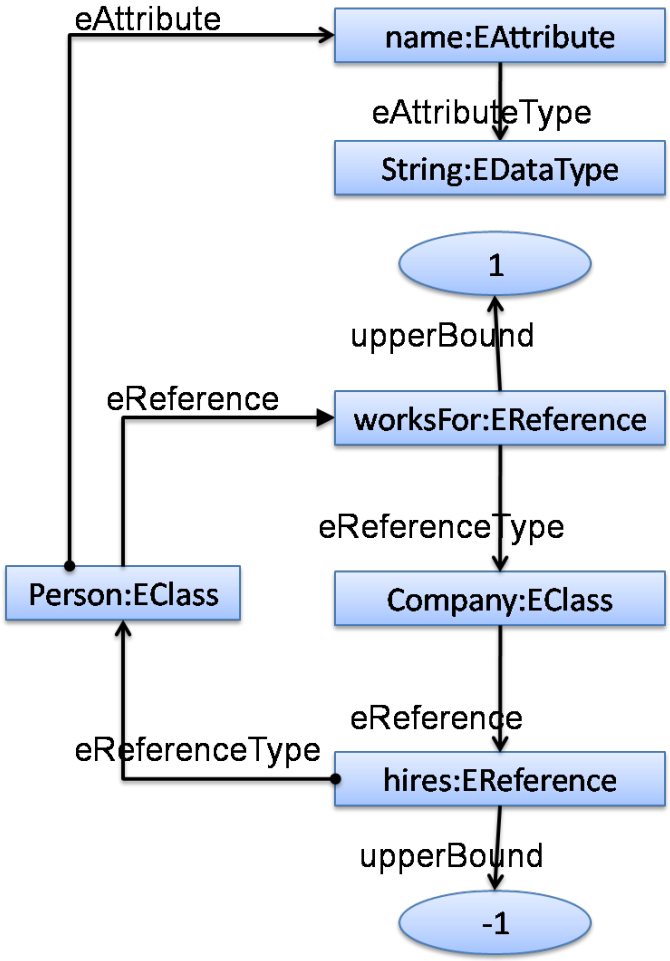
# Outline

# MT Definition: Ecore to RDB

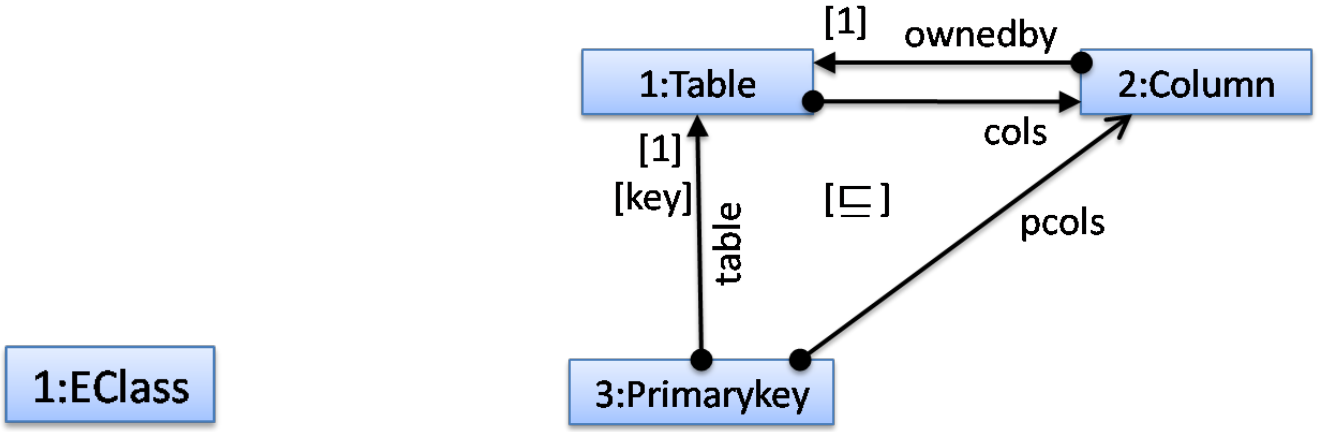


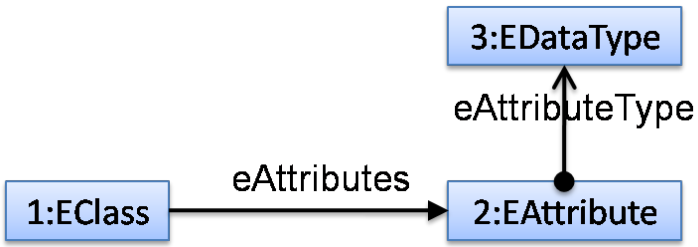
# The Input EMF Model

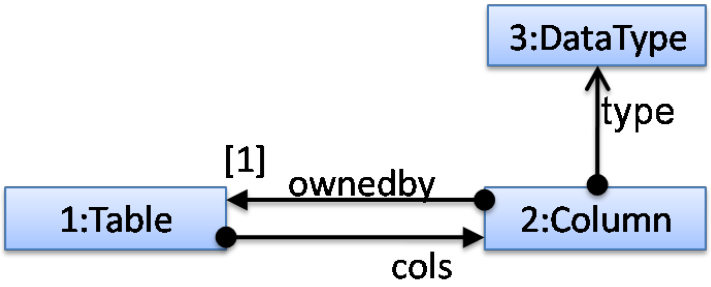


# Rule1: EClass to Table + PrimaryKey

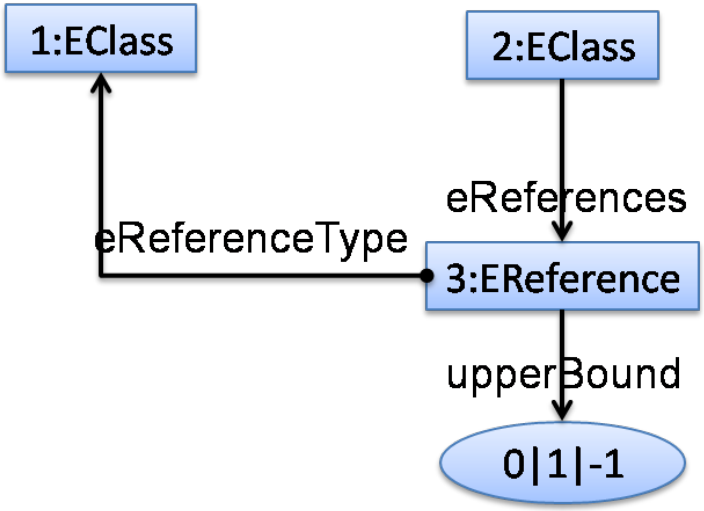


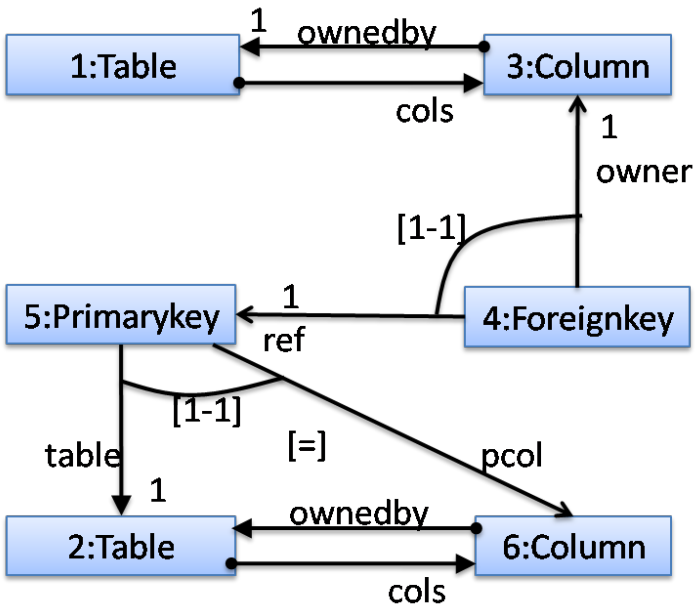
# Rule2: EAttribute to Column



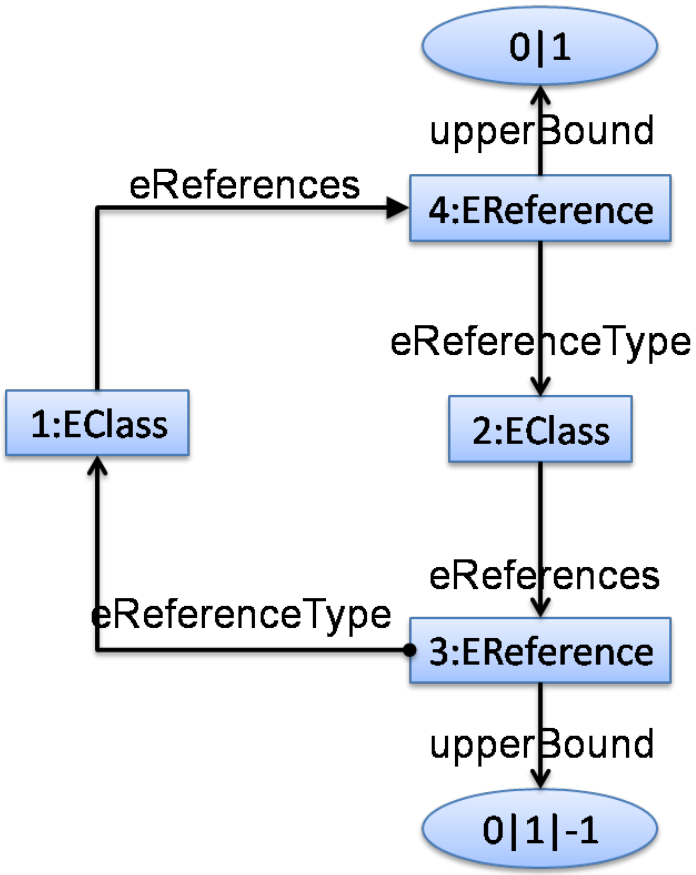



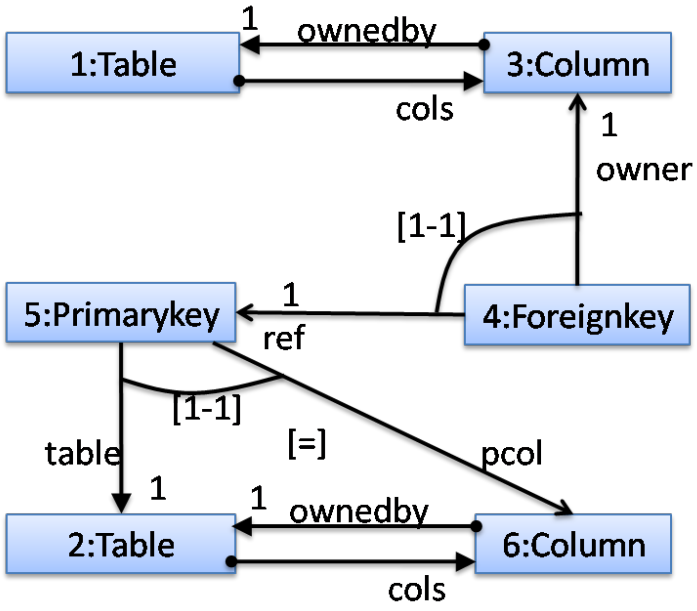
# Rule3 EReference to ForeignKey



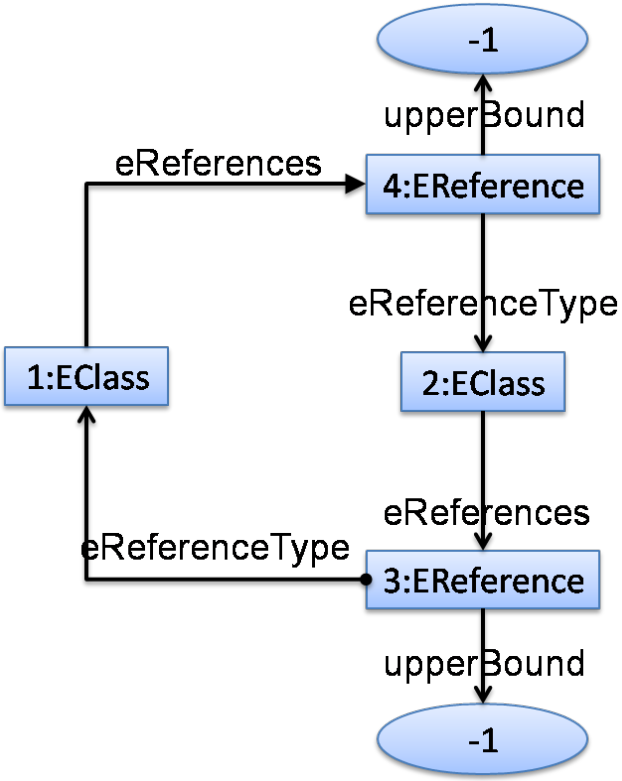



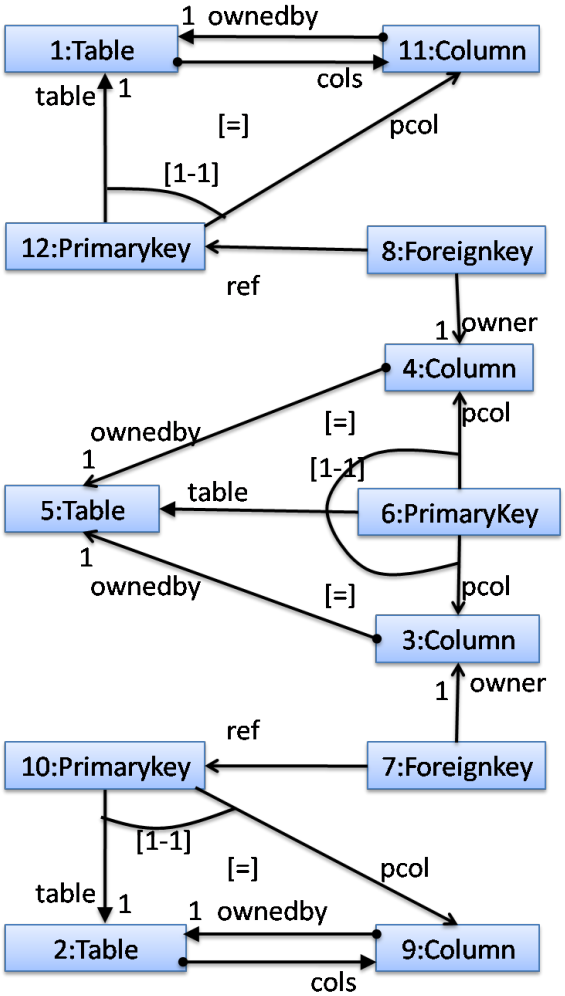
# Rule4: 1-* EReferences to ForeignKey



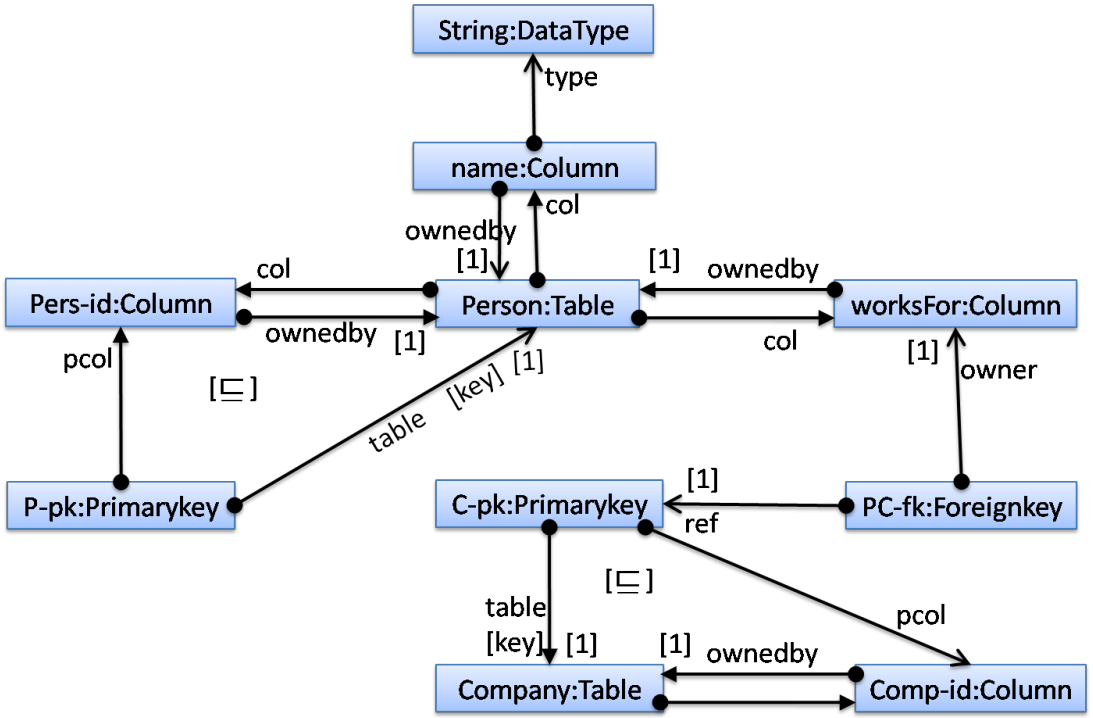



# Rule5: *-* EReferences to ForeignKey



# The Output RDB Model

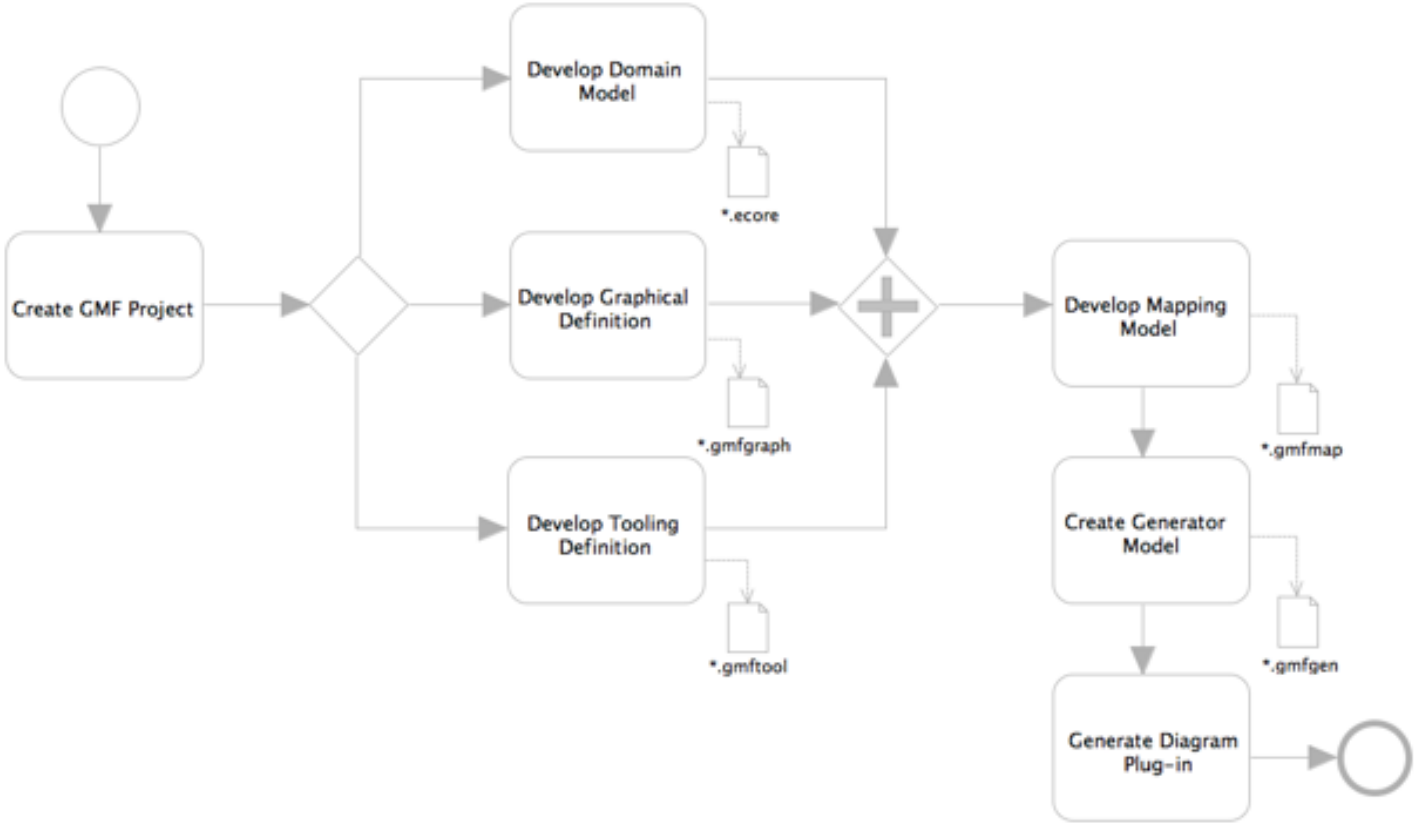


# Outline

# Tools

- Tools implemented as plug-ins to the Eclipse platform
- Need tools for:
  - Drawing of models (GMF-based editor), partly finished June 2008
  - Serialization of models, work started
  - Validation of models, future research project
  - Model transformation tools, research started

# Generation of GMF-based Editors



# Summary

- The MDD is a promising approach for software development processes
- The MDD requires tools for automatic model management
- Automatic model management requires formal modeling and model transformation languages
- Using DPL as a generic framework for formalization of modeling and model transformation languages
- Lessons learned from studying the Eclipse platform