TECHNISCHE UNIVERSITÄT DRESDEN

Department of Computer Science, Software Technology Group

ResUbic

Qualitune

# Role-based Object-Relational Co-Evolution

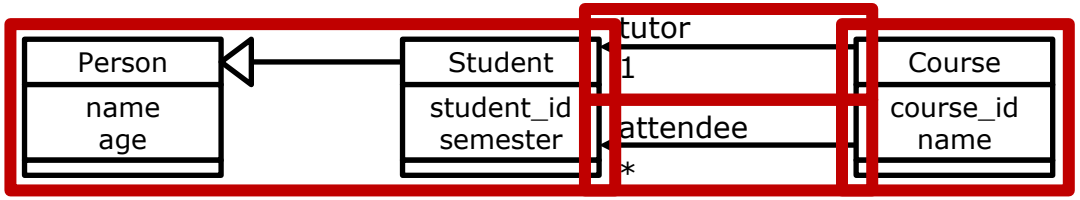
**Sebastian Götz**, Sebastian Richly and Uwe Aßmann

27.06.2011

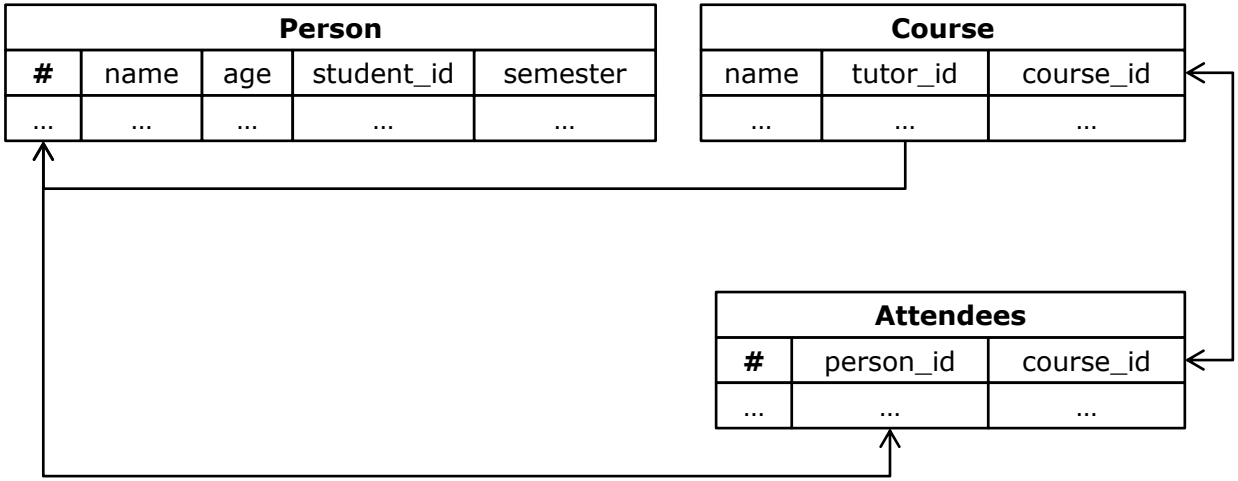
RAM-SE 2011 co-located with TOOLS 2011, Zurich, Switzerland

# Context

OO **Domain** Model:

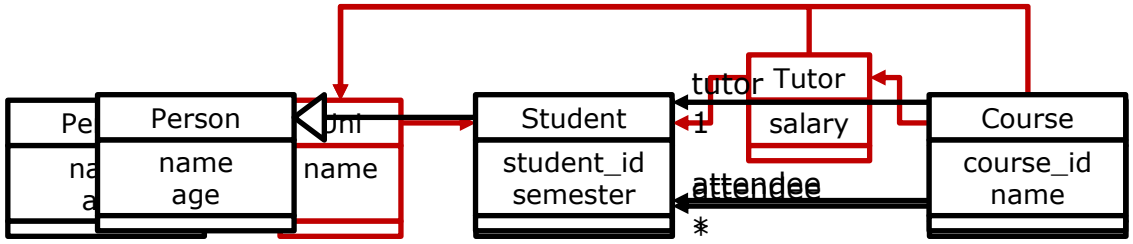

Relational Schema:




1) ORM = Object-Relational Mapper

# Problem

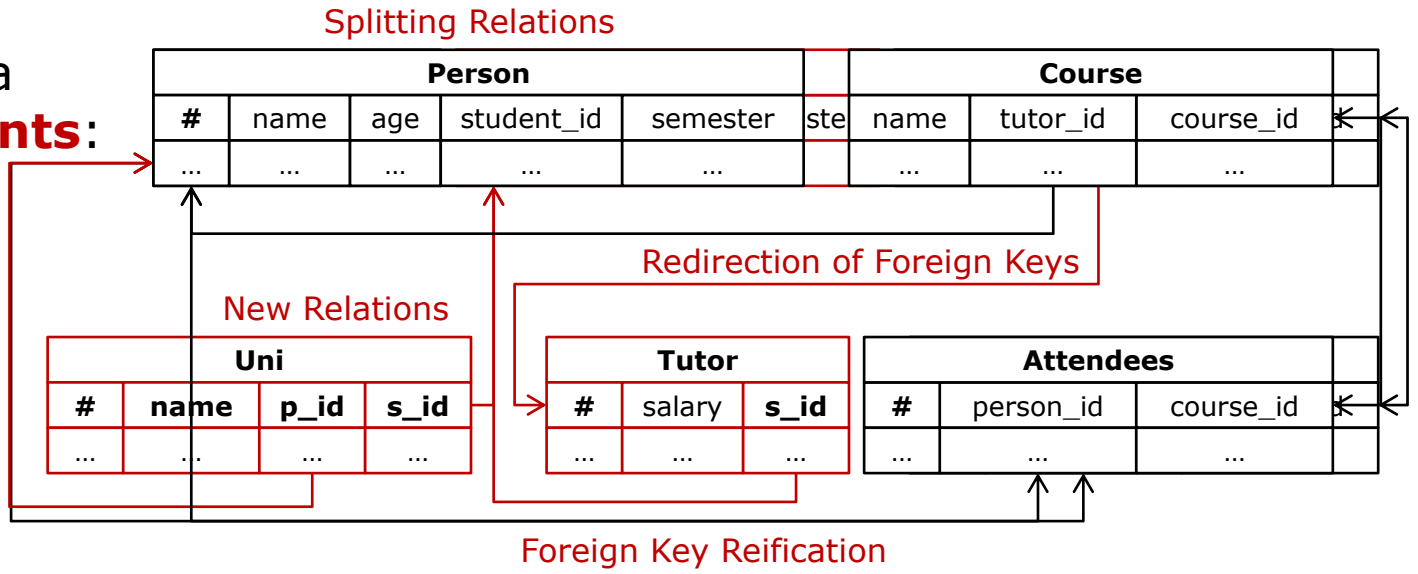
OO **Domain** Model:
**evolves**:




Relational Schema
**needs adjustments**:

➔ Time-consuming

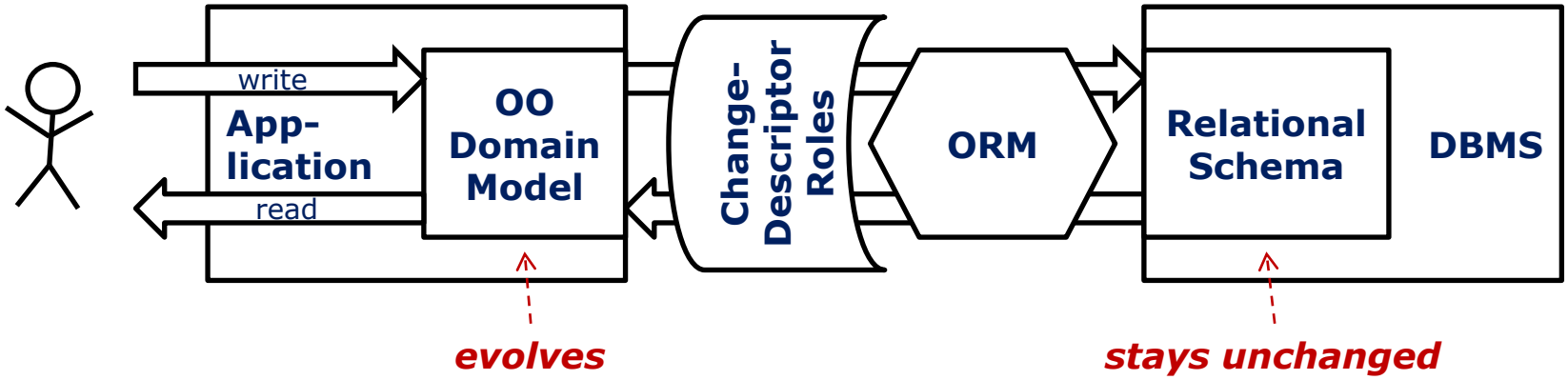
➔ Rare support for changes beyond addition



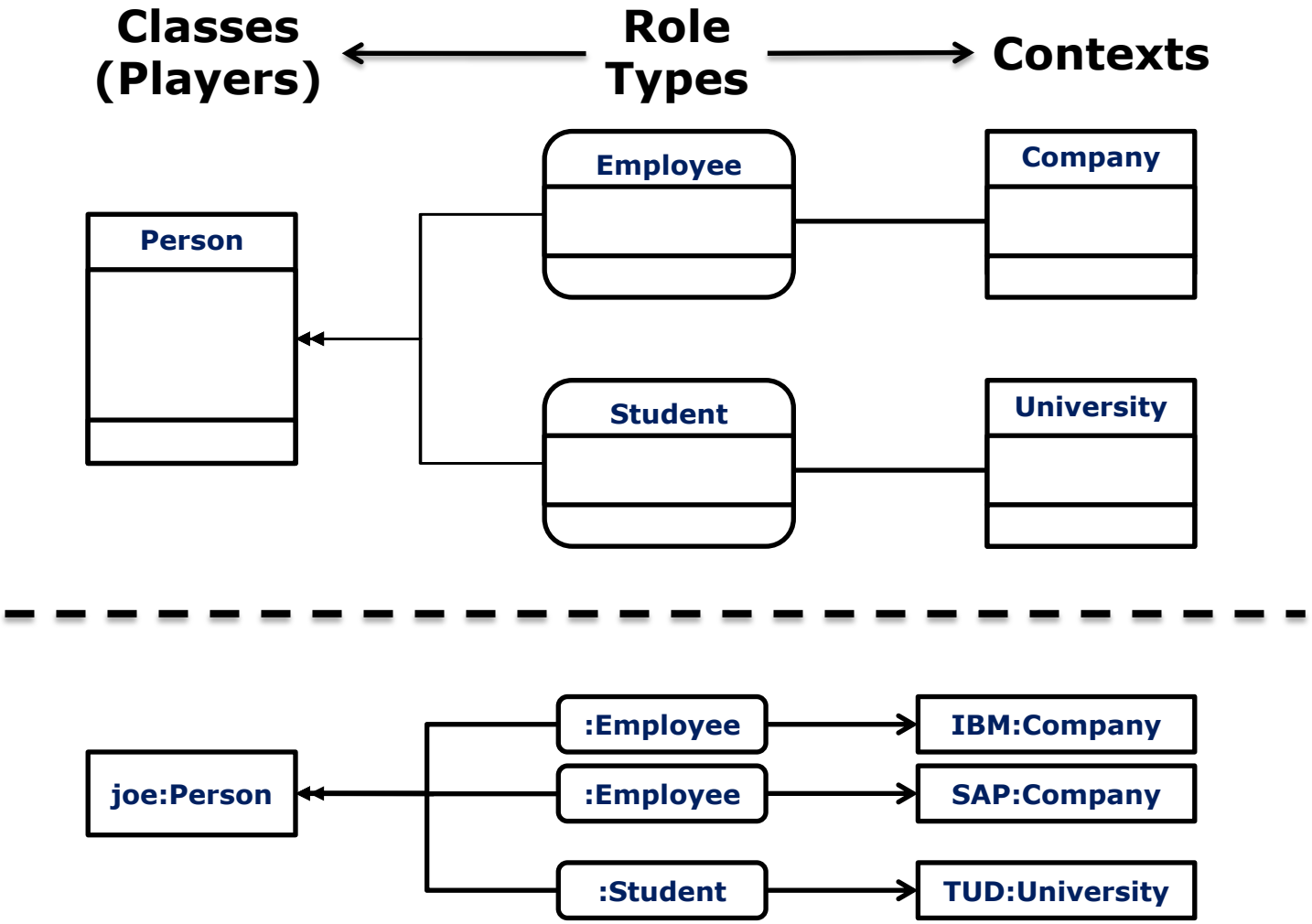
# Goal

➔ Automatic Adaptation of Object-Relational Mappings keeping the relational schema intact

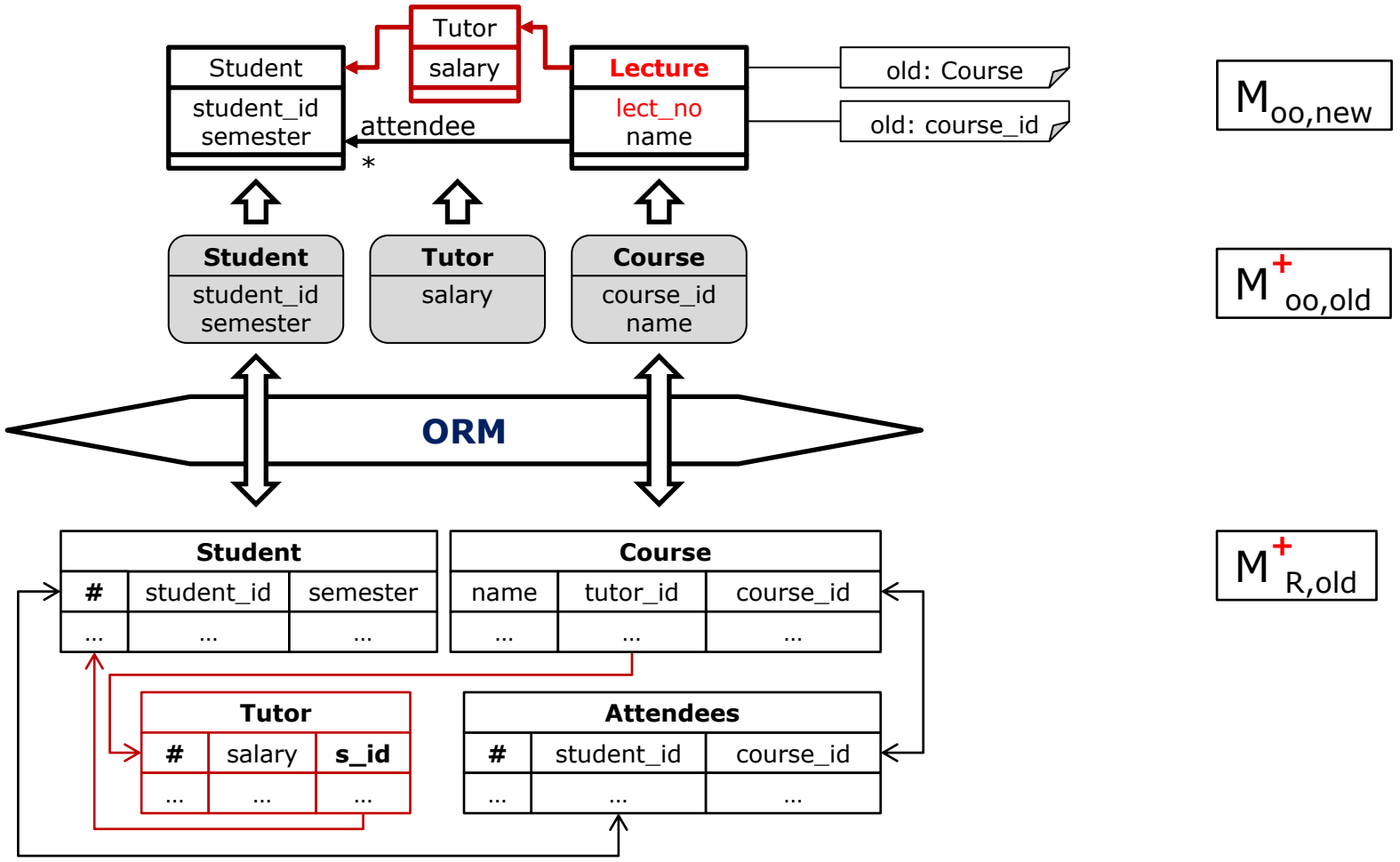
| Benefits | Drawbacks |
|---|---|
| Rapid Development | Performance Penalities |
| - postpone DB adjustments<br>- independent parallel work<br>- postpone query adjustments | - in large scale scenarios (many versions) |



# Object-Roles

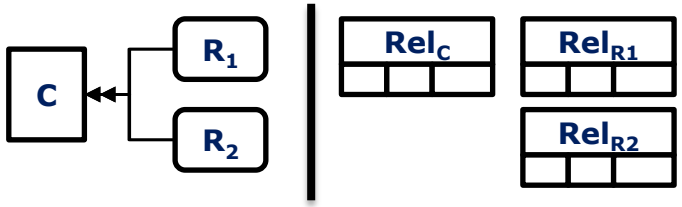


## ➔ Change Descriptor Roles (CDRs)

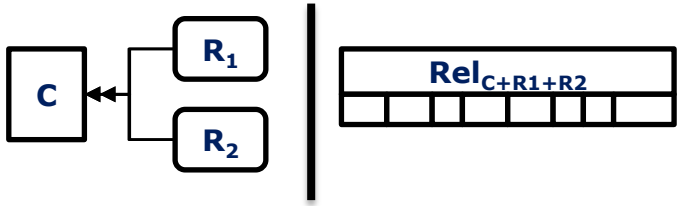


# Role-Relational Mapping

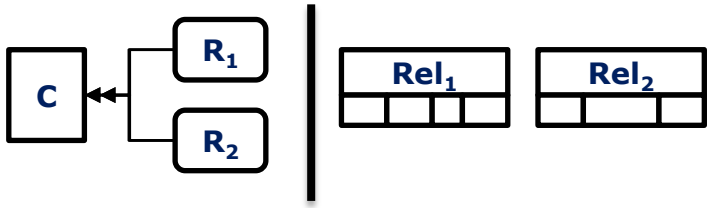
1. **Separate Relation per Role**
   - Complex queries / many joins



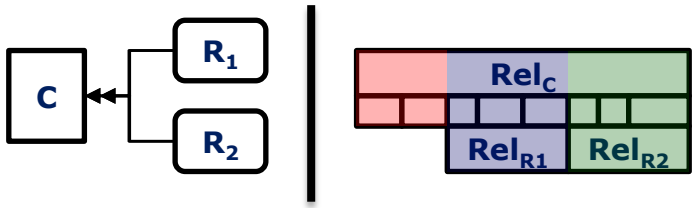

2. **Single Relation per Player-Role combination**
   - Sparse data
   - Semantic redundancy




3. **Normalized Single Relation**
   - Automatic normalization unfeasible
   - (due to high complexity)

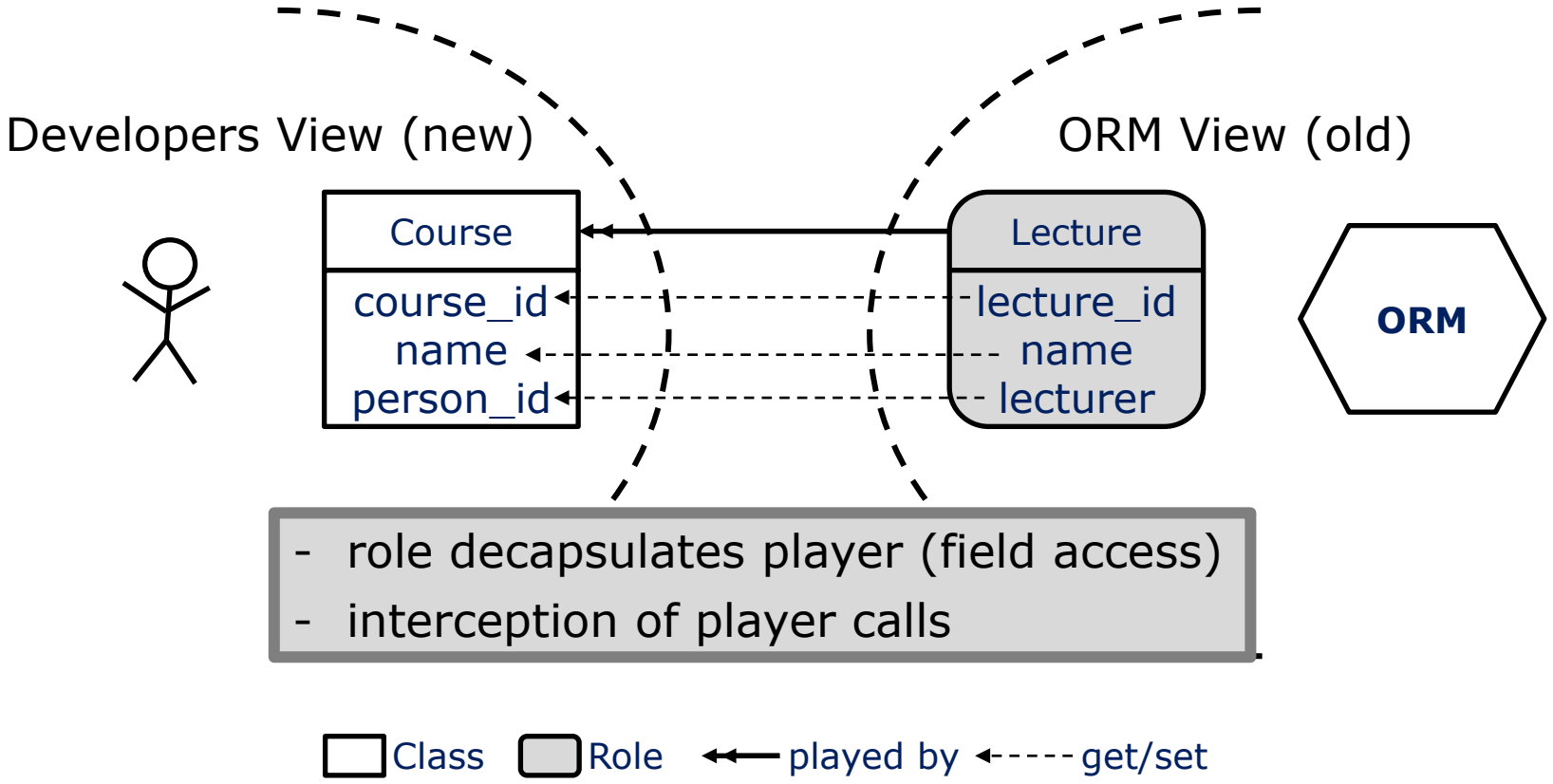



4. **SQL:99 subtable**
   - subtable = Role, supertable = player
   - context requires additional foreign key
   - rarely supported by DBMS

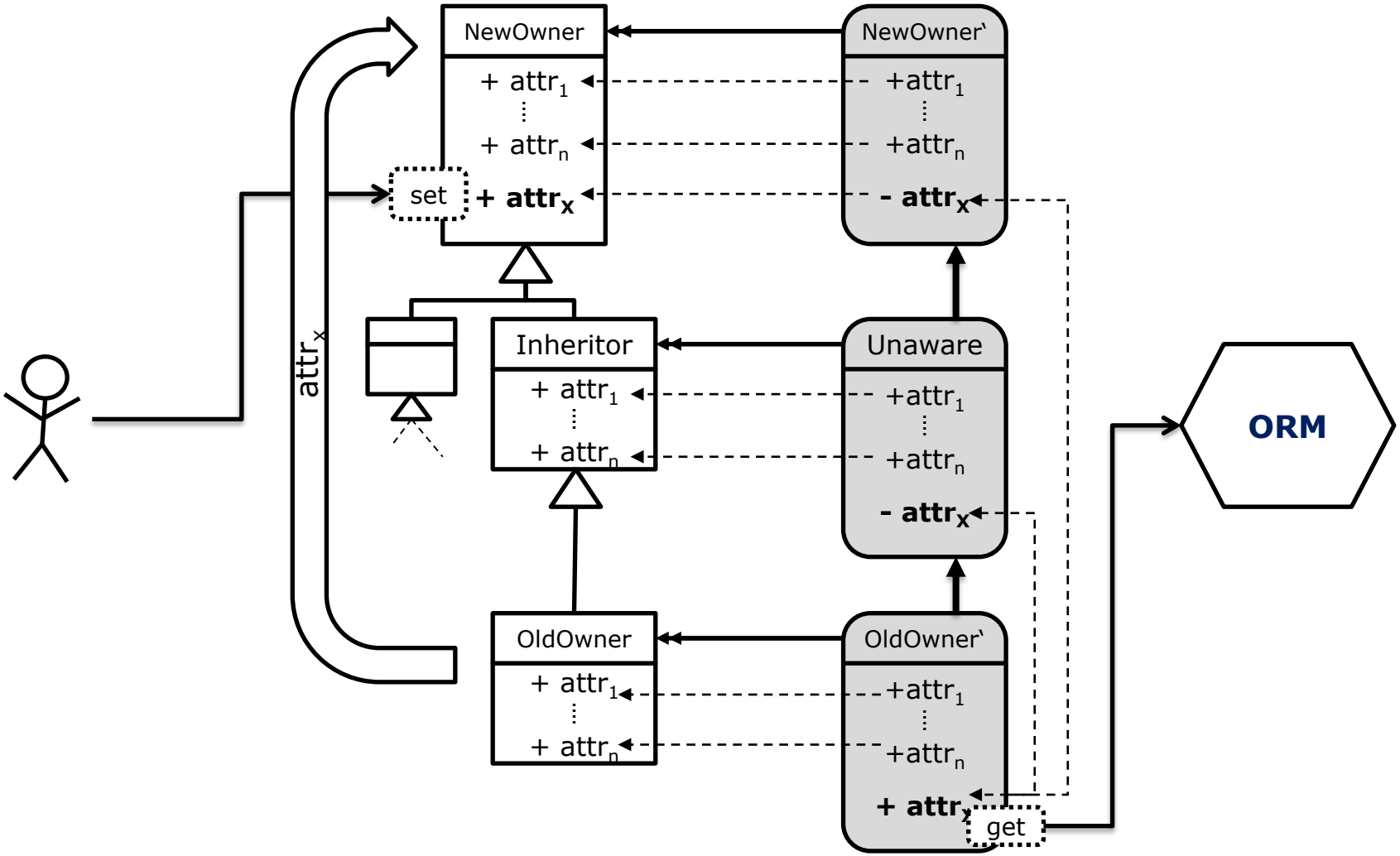


# Rename CDR

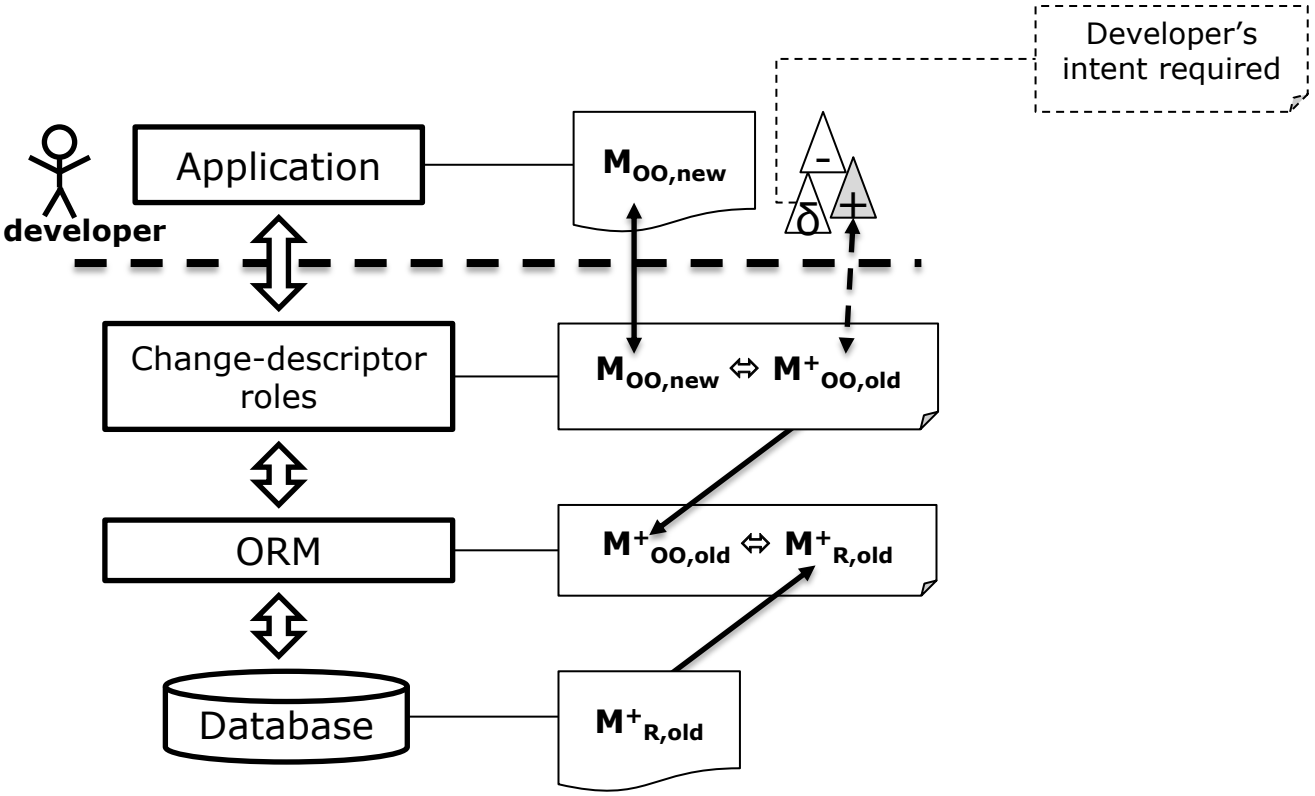
➔ Developer renames class **`Lecture`** to **`Course`**



# PullUp Attribute CDR



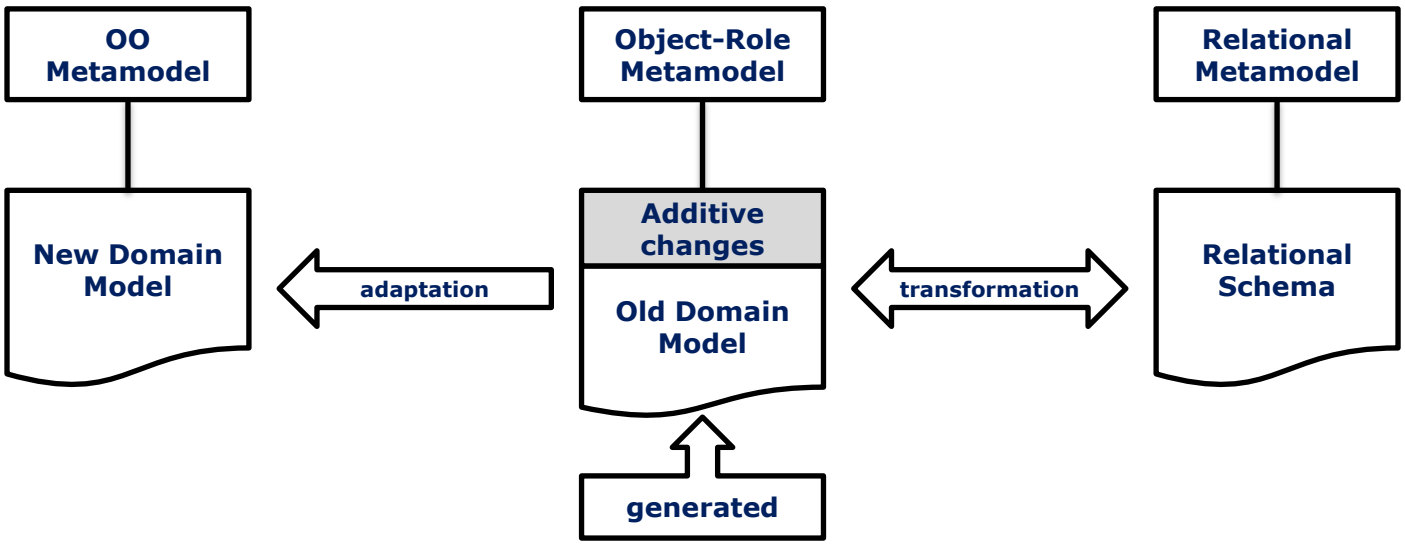
# Solution


Developer's intent required
developer
Application
Change-descriptor roles
ORM
Database
$M_{OO,new}$
$M_{OO,new} \Leftrightarrow M^{+}_{OO,old}$
$M^{+}_{OO,old} \Leftrightarrow M^{+}_{R,old}$
$M^{+}_{R,old}$
δ
-
+

# Related Work

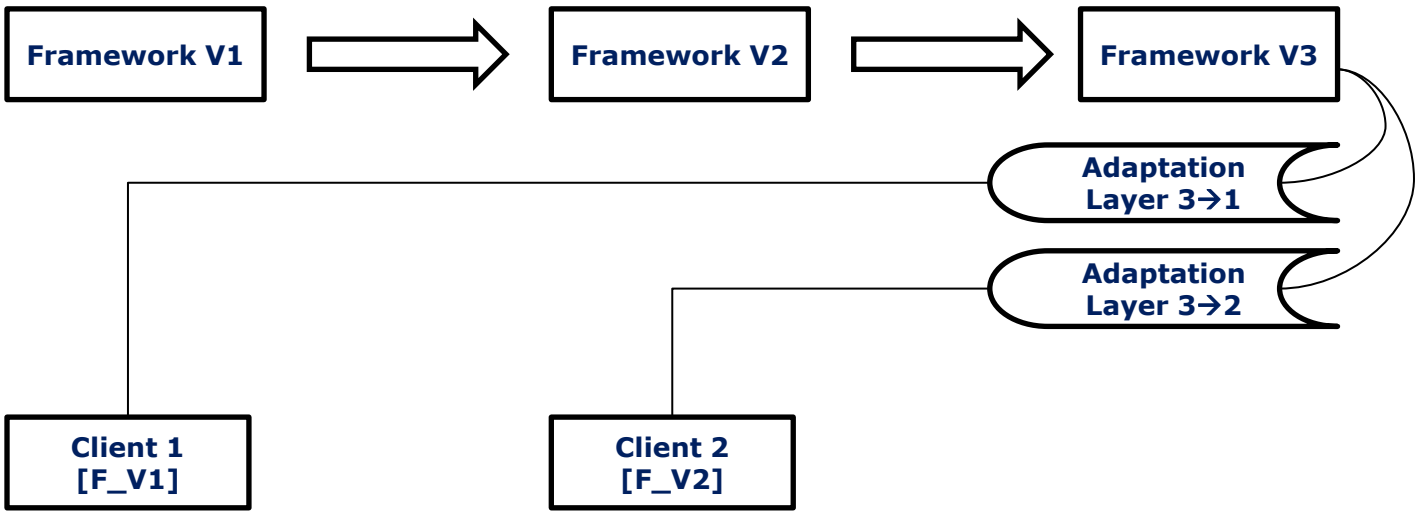
**Coupled Evolution in Model-driven Engineering** (e.g., [5, 6])

- Evolve model instances according to metamodel changes
- The work presented here does not consider metamodel changes
  - instead evolving source of model transformation (OO to Relational)
  - different goal: shielding the target from complex changes



**ComeBack** [10]

- shielding plugins/clients from framework evolution
- holistic adaptation layer
- Framework looks unchanged to clients vs.
  Application looks unchanged to ORM
- Focus on control-flow (not on data-flow)



# Related Work

- **MeDEA** [7] allows to apply application schema changes to the DB
  - user needs to provide ***migration script***

- **Terwilliger et al.** [8] handle object-relational co-evolution by transforming
  - Changes of the application to
    Changes of the mapping between the application and the DB
  - The goal is not to postpone model migration, but
    - To automatically perform model migration

- Approaches like **PRISM** [9] allow to automatically evolve database queries

# Summary

**Problem**

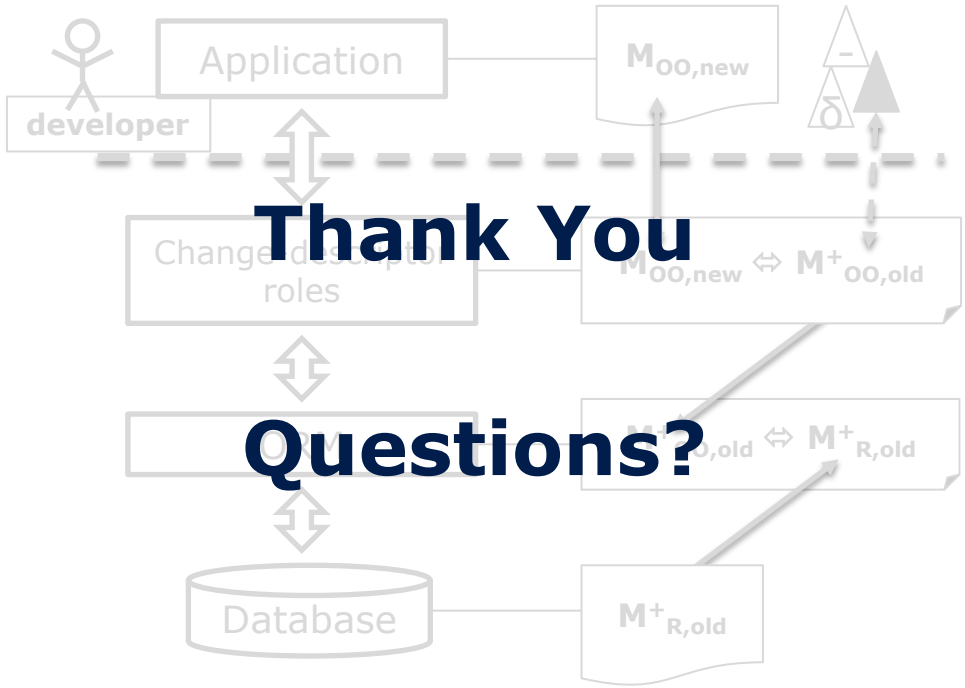
- Evolution of object-oriented domain model
- Adjustments to relational schema required
  - Time-consuming
  - Changes beyond additions poorly supported by DBMS

**Goal**

- Evolve OO domain model keeping relational schema intact
  - Fosters development productivity
    - ability to postpone changes
    - parallel development (on the same data)

**Solution**

- Change-Descriptor Roles / Holistic Adaptation Layer
  - Adapting new OO domain model to its old version
  - Hiding changes from the ORM




# Thank You

# Questions?

# Literature


[1] Baldoni, M.; Boella, G. & van der Torre, L.
**Roles as a Coordination Construct: Introducing powerJava**
*Electronic Notes in Theoretical Computer Science, Elsevier Science Publishers B. V.,* **2006**, *150*, 9-29

[2] Monpratarnchai, S. & Tetsuo, T.
**The Implementation and Execution Framework of a Role Model Based Language, EpsilonJ**
*Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2008. SNPD '08. Ninth ACIS International Conference on,* **2008**, 269 -276

[3] He, C.; Nie, Z.; Li, B.; Cao, L. & He, K.
**Rava: designing a Java extension with dynamic object roles**
*Engineering of Computer Based Systems, 2006. ECBS 2006. 13th Annual IEEE International Symposium and Workshop on,* **2006**, pp. 452-459

[4] Herrmann, S.
**ObjectTeams: Improving Modularity for Crosscutting Collaborations**
*NetObjectDays, Springer,* **2002**, *2591*, 248-264

[5] B. Gruschko, D. Kolovos, and R. Paige.
**Towards Synchronizing Models with Evolving Metamodels.**
In *Proceedings of the Work. MODSE, 2007.*

# Literature

[6] G. Wachsmuth.
**Metamodel Adaptation and Model Coadaptation.**
Proceedings of the 21st ECOOP, volume 4069 of LNCS. Springer-Verlag, July **2007**.

[7] E. Dominguez, J. Lloret, A. Rubio, and M. Zapata.
**Evolving the implementation of isa relationships in EER schemas.**
In Advances in Conceptual Modeling - Theory and Practice, volume 4231 of Lecture Notes in Computer Science, pages 237-246. Springer Berlin / Heidelberg, **2006**.

[8] J. Terwilliger, P. Bernstein, and A. Unnithan.
**Automated co-evolution of conceptual models, physical databases, and mappings.**
In Proceedings of Conceptual Modeling - ER 2010, volume 6412 of Lecture Notes in Computer Science, pages 146-159. Springer Berlin / Heidelberg, **2010**.

[9] C. Curino, H. Moon, and C. Zaniolo.
**Graceful database schema evolution: the prism workbench.**
In Proceedings of VLDB Endow., 1:761-772, August **2008**.

[10] I. Savga, M. Rudolf and S. Götz
**Rigorous and Practical Refactoring-Based Framework Upgrade**
In *Proceedings of 7th International Conference on Generative Programming and Component Engineering (GPCE'08),* **2008**

# Discussion

- **Why Object-Roles?**
  - Roles (only) perform structural adaptations
  - Two interacting collaborations
    - User interacts with new version objects
    - ORM interacts with old version objects
  - New role layers can be added on the fly

- **Indirection imposes problems:**
  - **Performance**
    → meant for rapid development
    → not for productive use
  - **Debugging**
    → Debugging the OR mapping is hard
    → Debugging of application is not effected

- **Languages supporting role-based OO**
  - powerJava [1], EpsilonJ [2], Rava [3], **OT [4] (most mature)**

# Discussion

- **Scalability**
  - Tradeoff between usability and scalability in terms of performance
    - Changes between first and current version → 1 layer of indirection
    - Changes between last and current version → N layers of indirection
- **Weaknesses of the Approach**
  - Annotations to be provided by the developer (to identify the semantics of changes)
- **Tool Support**
  - The annotations can be generated using, e.g., the IDE refactoring log
  - CDRs are generated by default
- **Changes hard or impossible to model**
  - changed visibility might look like hard to model
    - but is not, due to *decapsulation* [4] by roles

# Problems due to Additions?

➔ Developer renames class **`Lecture`** to **`Course`** and introduces new subclasses **`Lecture`** and **`Seminar`**.