# MDA Reference Model

**Wim Bast[1], Tony Clark[2], Allan Kennedy[3], Laurence Tratt[4]**

[1] Wim.Bast@nl.compuware.com Compuware

[2] tony.clark@xactium.com Xactium

[3] allan.kennedy@kc.com Kennedy Carter

[4] laurie@tratt.net King's College London

# Introduction

- What is MDA?

# Introduction

- What is MDA? ...Exactly!
- Many meanings for many people.
- For MDA’s continued success, it is vital to gain more of a common understanding of what MDA is.

# Why is a definition useful?

- Capture existing knowledge.
- Provides the basis for a common understanding.
- Is both a reference and conformance point.
- Exposes errors in what is being defined.
- Exposes deficiencies in what is being defined.

# Previous work

- Much of this has been done by the ORMSC.
- The MDA Guide.
  Excellent first step towards defining MDA, but does so via informal diagrams and text.
- Allan Kennedy's reference model (available in two parts from the OMG website here and here).

# What we want to achieve

- A more formal definition of MDA, via a reference model.
- Hopefully feed back into a new version of the MDA Guide.

Try not to get bogged down in terminology!

# What is being defined

- We are in the early stages: better to be liberal than prescriptive.
- Currently we are only interested in defining the most basic and simple MDA concepts... although those are often the most contentious!
- As the definition is agreed upon, increase the scope and rigour of the description.

# Formality of definition

- General rule: start off informal, move gradually towards being more formal.
- Being fully formal is not always necessary, or helpful.
- Best of both worlds: formal(ish) underpinnings with explanatory prose.

# Definition style

- We strive to have a ‘modular’ definition; achieved by layering.
- Start with a very small, very expressive, kernel model and gradually layer more complex and restrictive concepts on top of that via translation or extension.

Advantages:

- Makes for ease of understanding of any given level.
- Allows forks of any given level (i.e. define your own semantics for your own extensions).

# The kernel

- ObjVLisp ‘everything is an object’ style.
- Advantages:
  - Proven approach.
  - Very simple, very general, very powerful.
- Not without its disadvantages, particularly the bootstrapping problem.

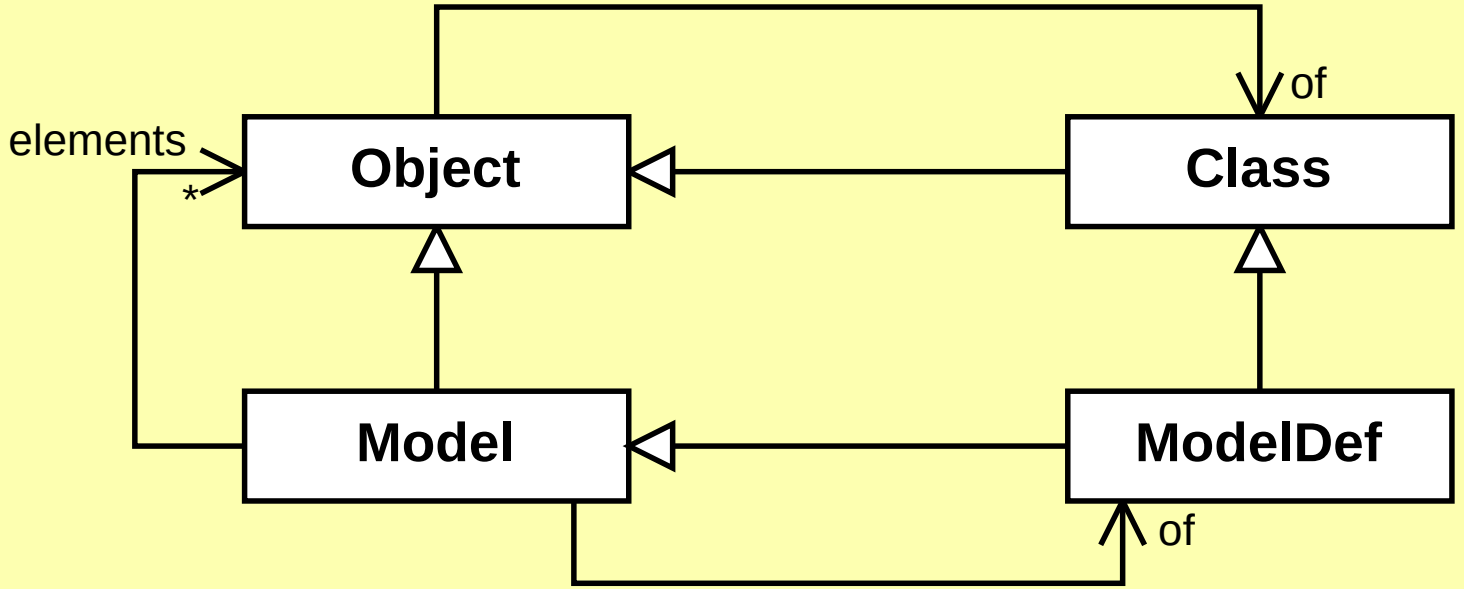
# Kernel

# Kernel

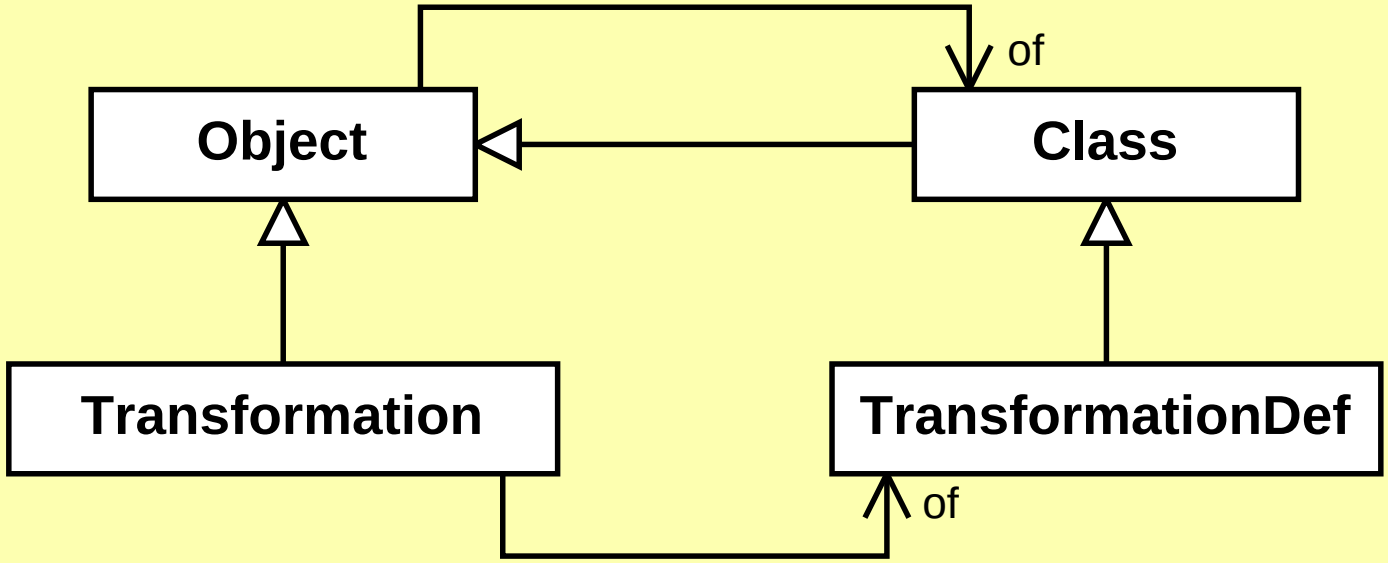

An important distinction (but don't dwell on it):

- Subclasses that subclass `Class` can create new classes - they are *metaclasses*.
- Those which directly subclass `Object` directly can create new objects.

# Models

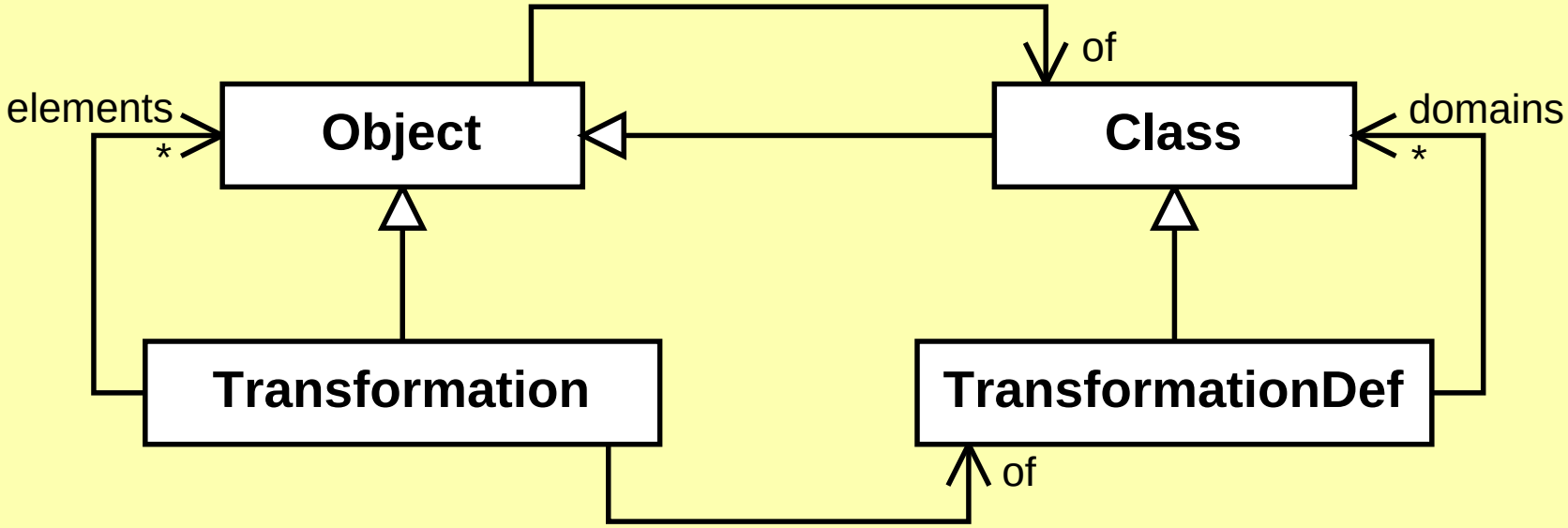


# Transformations




This is a ‘classic’ decision point: it’s tempting to have transformations be between one input and one output models...

# Transformations

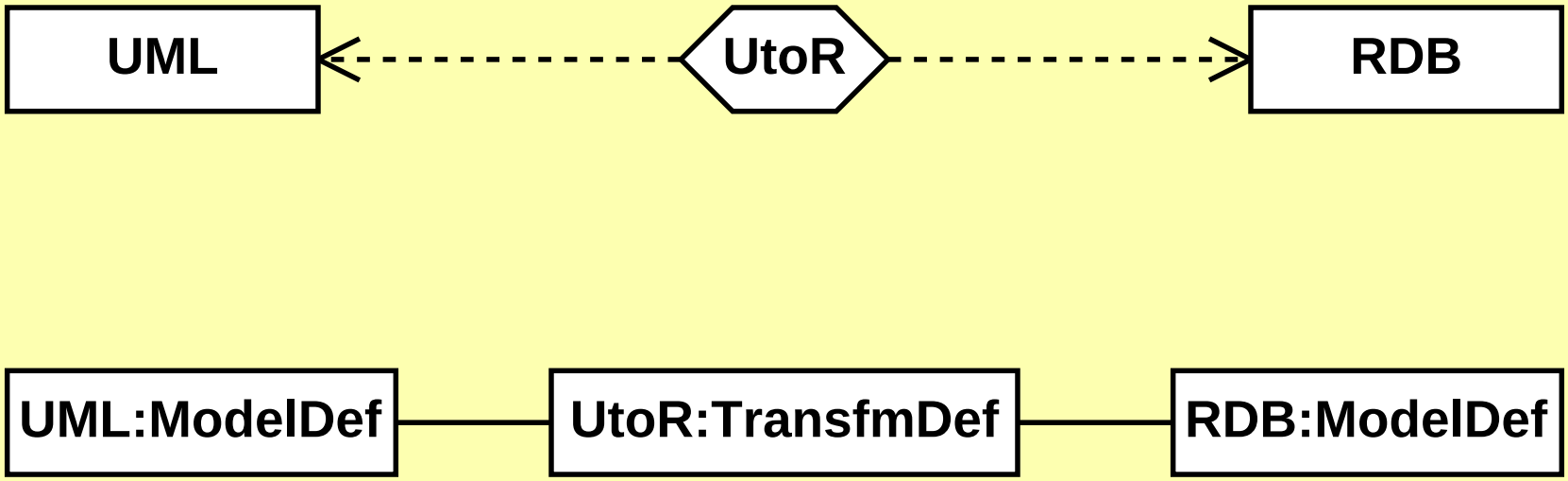



...but that wouldn’t cope with most of the QVT submissions!

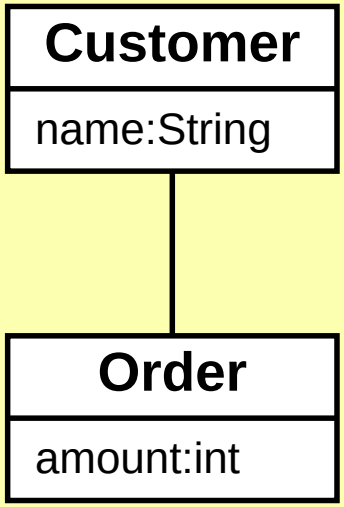
At this basic level one has to be liberal with the definition.

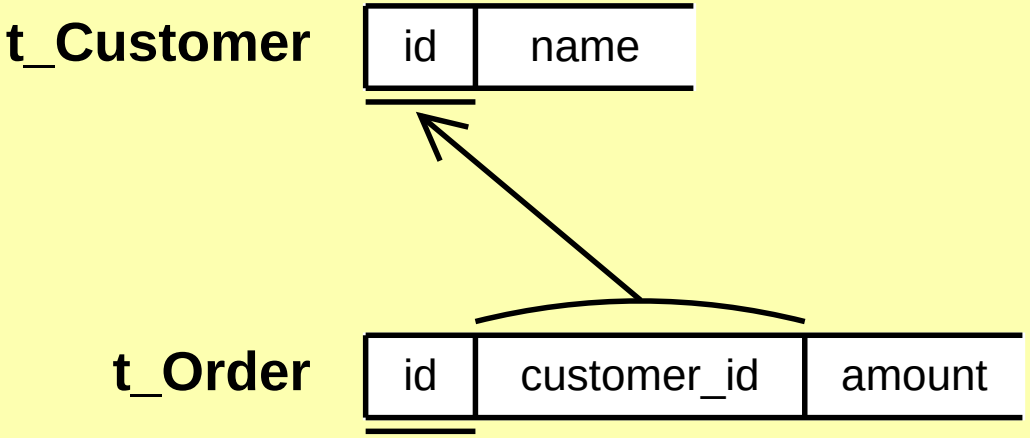
# Simple example

# Simple example

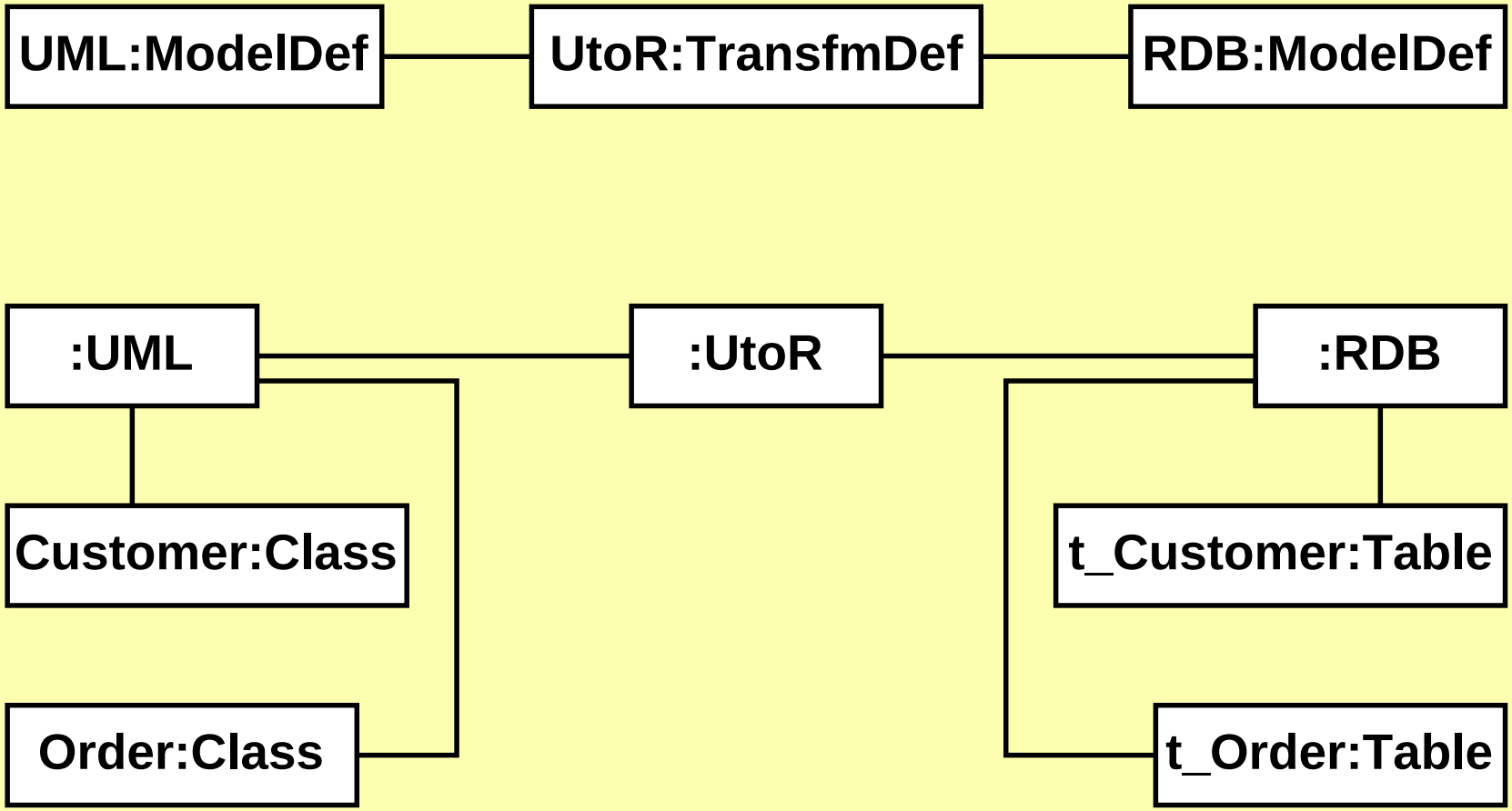


# Simple example

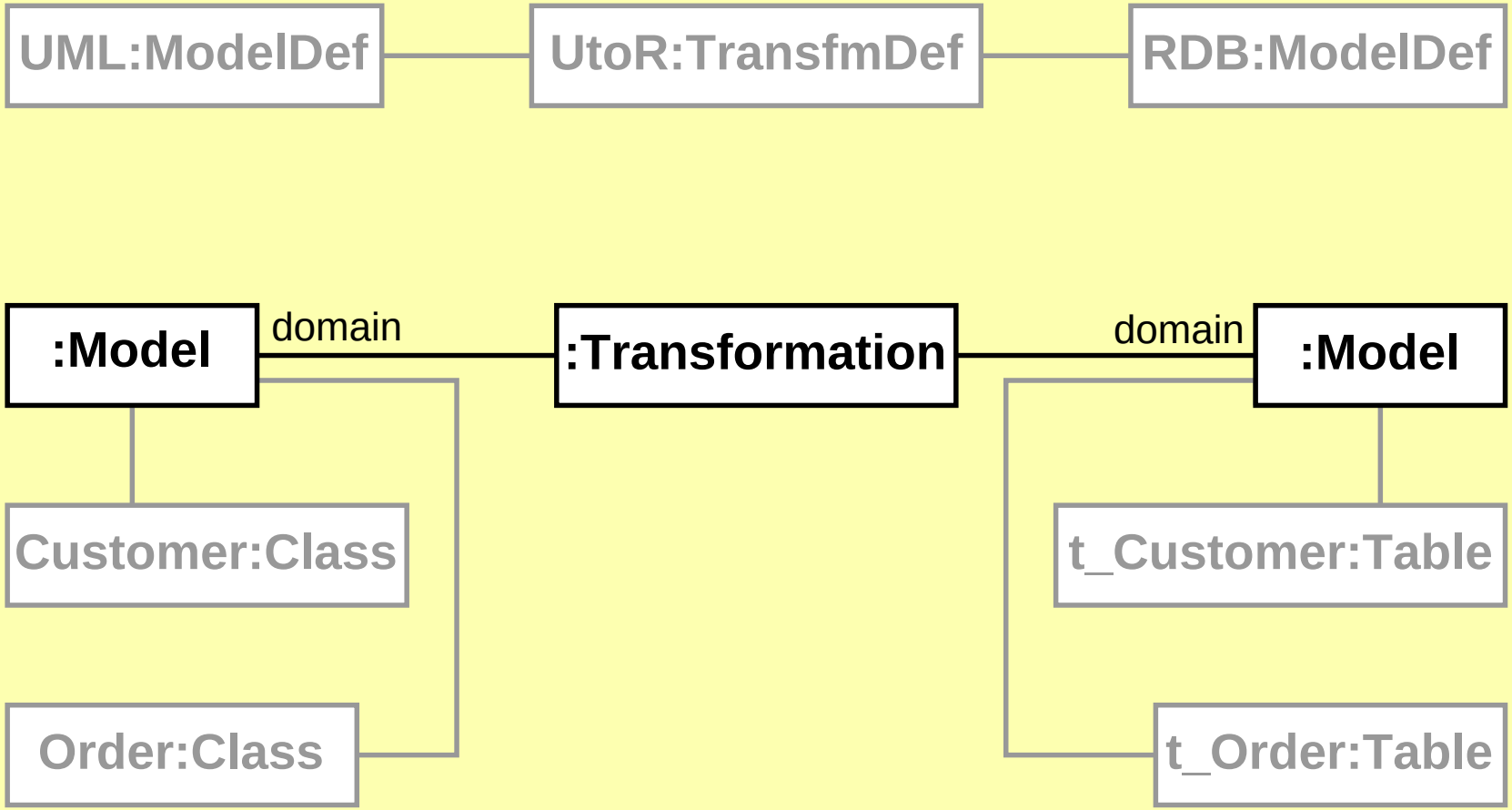



# Simple example



# Simple example



# Future work

- More of the same!
- Broaden participation.
- Integrate this into a more accessible document (e.g. the MDA Guide).