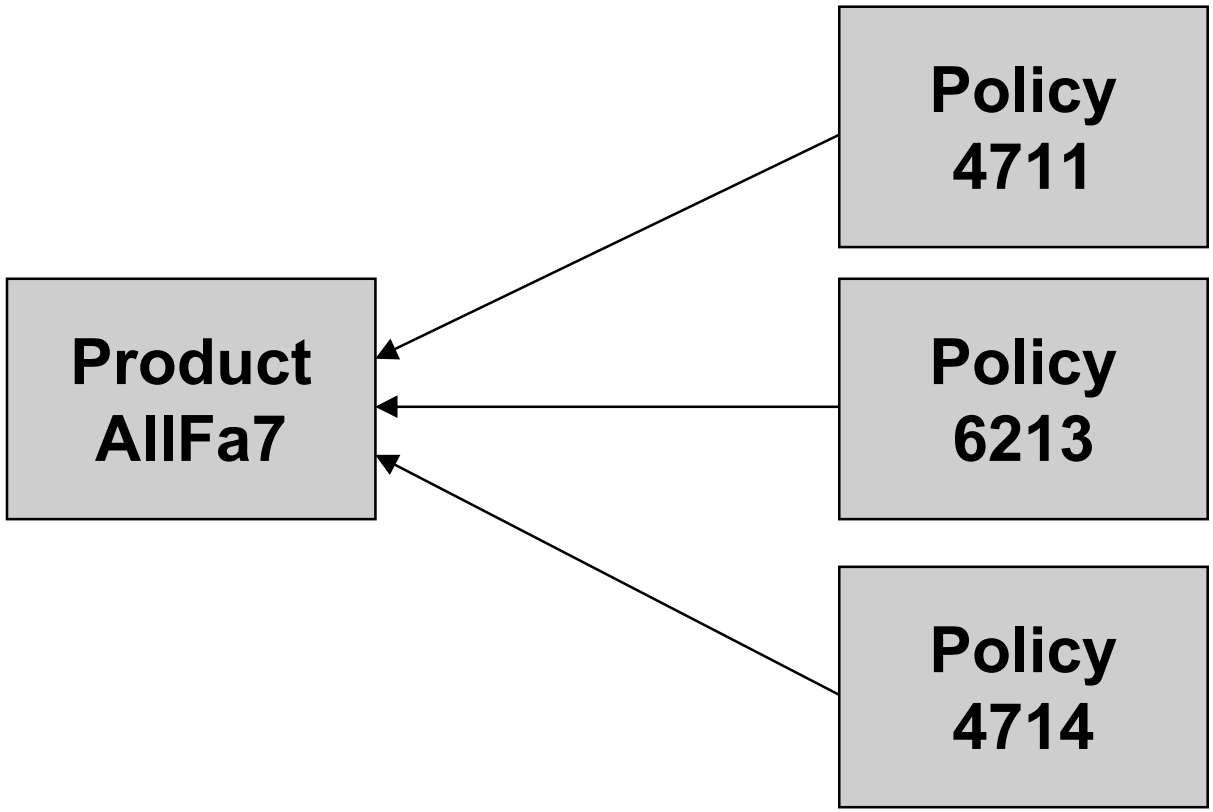
Flexible Insurance Systems

Part 4: Reflective Policy Systems

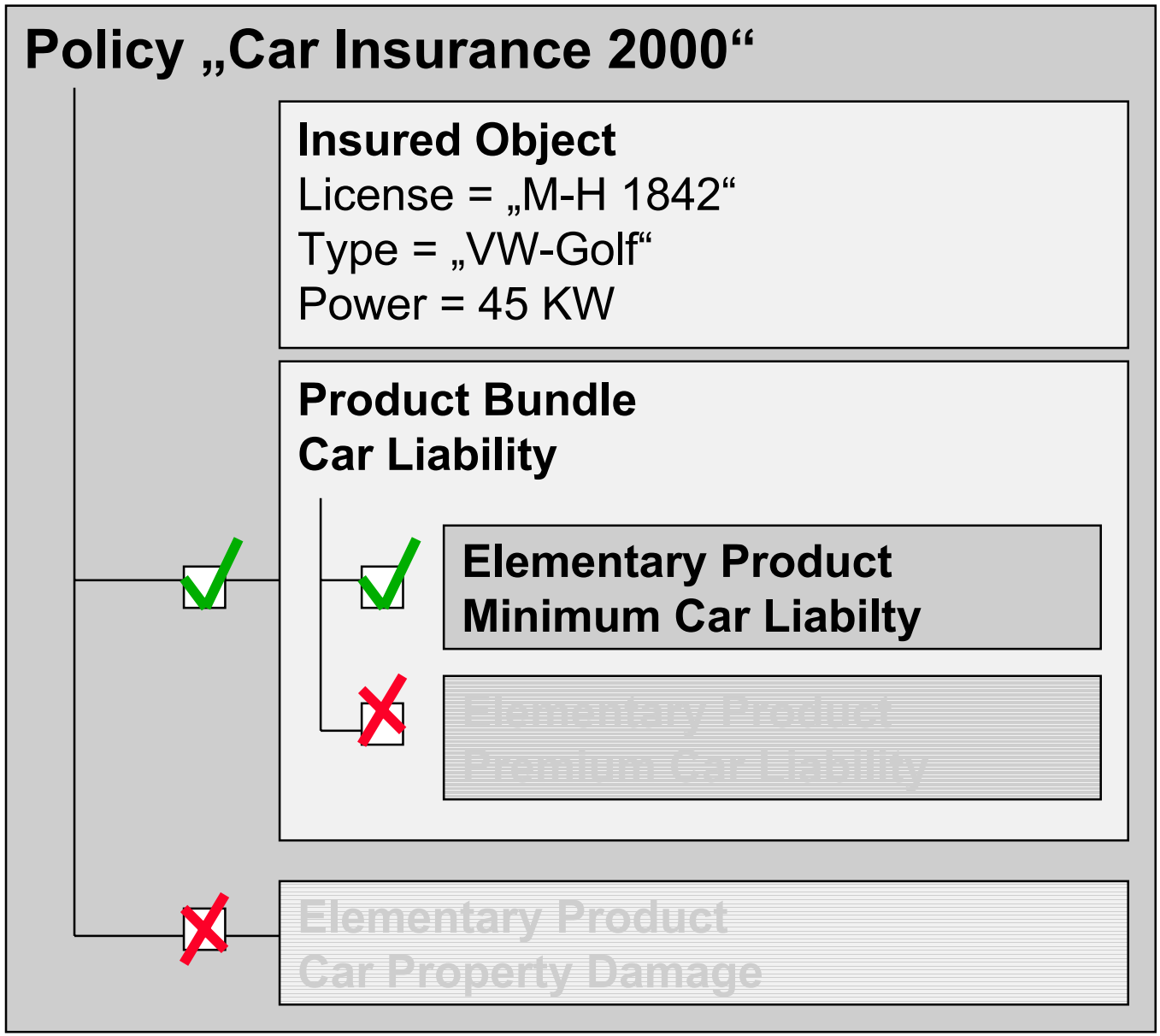
Agenda

- Now that I have a Product Definition,
  How do I implement a policy?
- The Reflection Pattern aka. Meta System
- Example: Flexible “Insured Objects”
- Is everything “meta”?
  - Partners are not, general ledger is not ,….
- How to provide generic funtionality
- How to control Performance?
  - Don’t make it too deep, Beispiele der Vertragsstruktur VIAS und PVS

How do I implement a policy?
Policies refer to Products



How do I implement a policy?
Policies mirror Products



How do I implement a policy?
Policies mirror Products

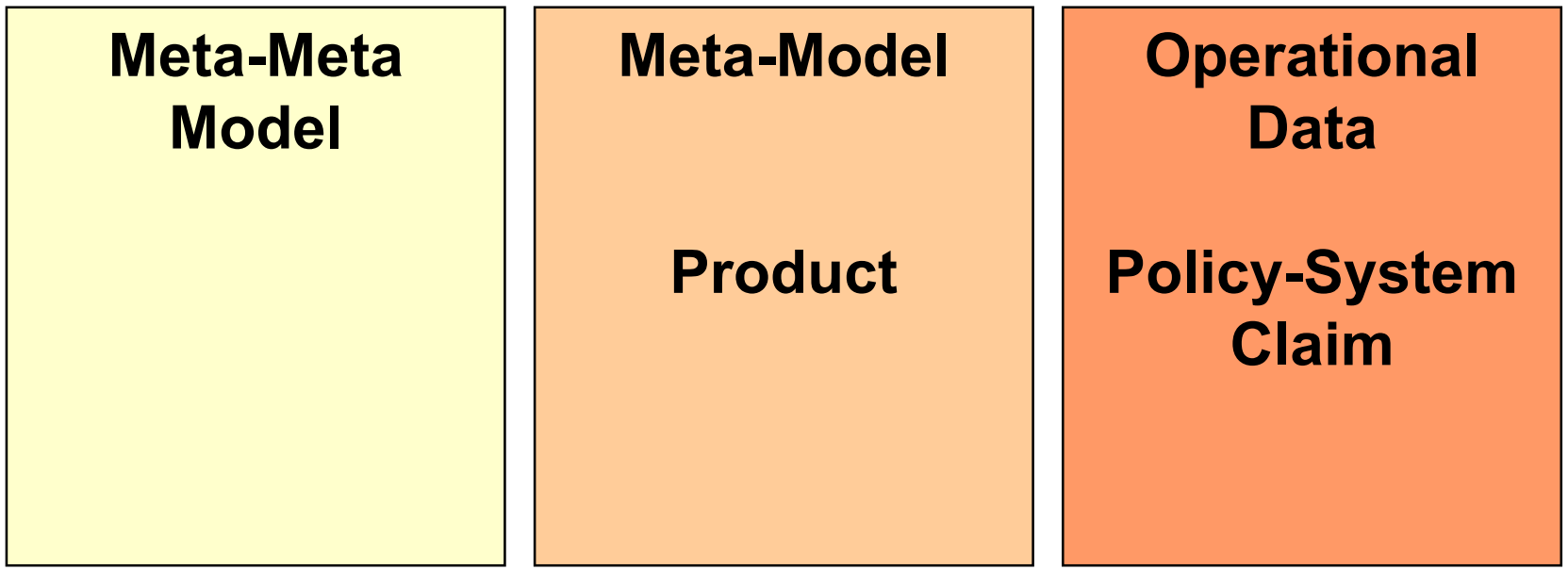
Insured Object Definition
Attribute License of type Text(10)
Attribute Type of type Text(20)
Attribute Power of type Num(4)
.....

Product Definition

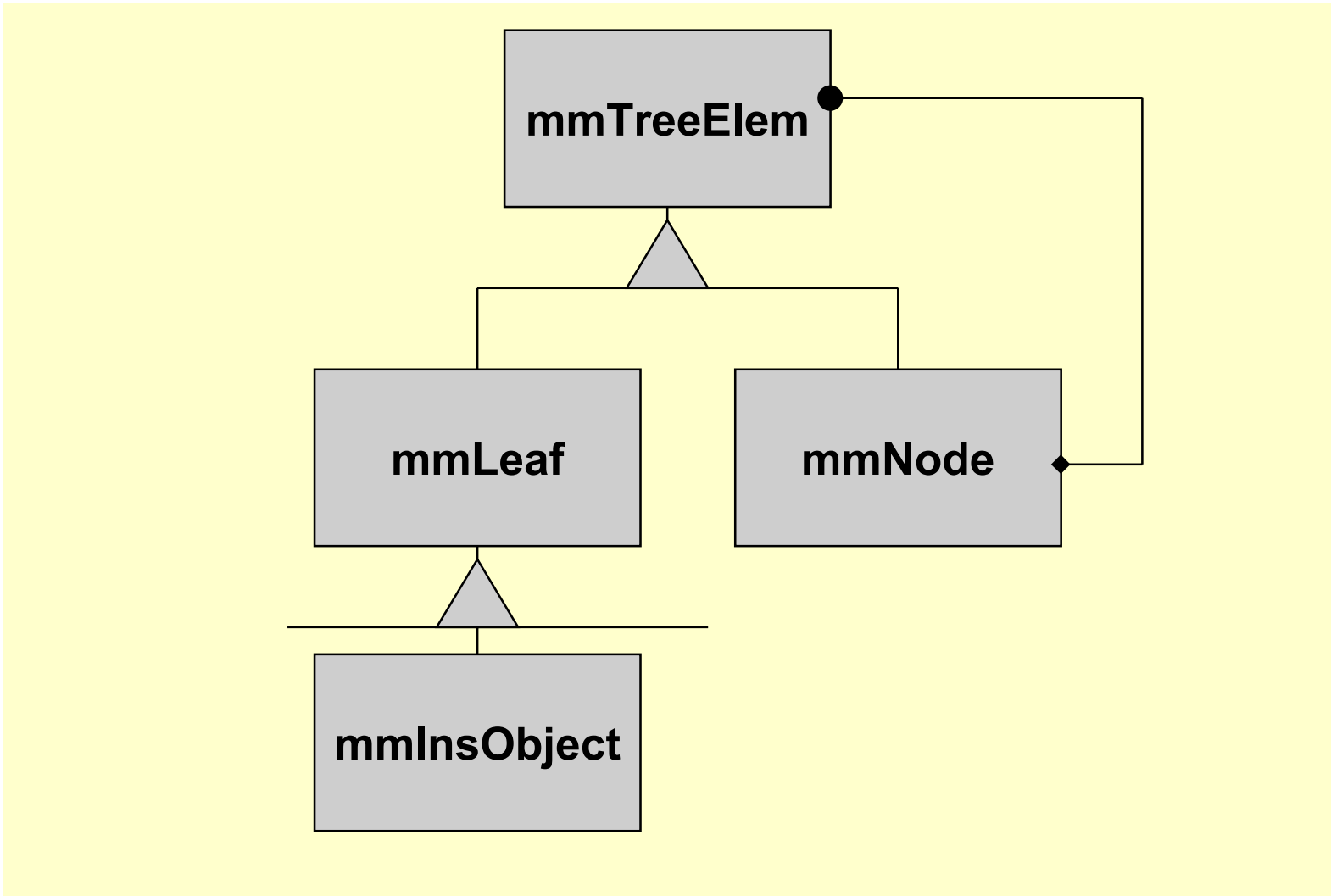
Policy Instance ..

Insured Object
License = „M-H 1842“
Type = „VW-Golf“
Power = 45
...

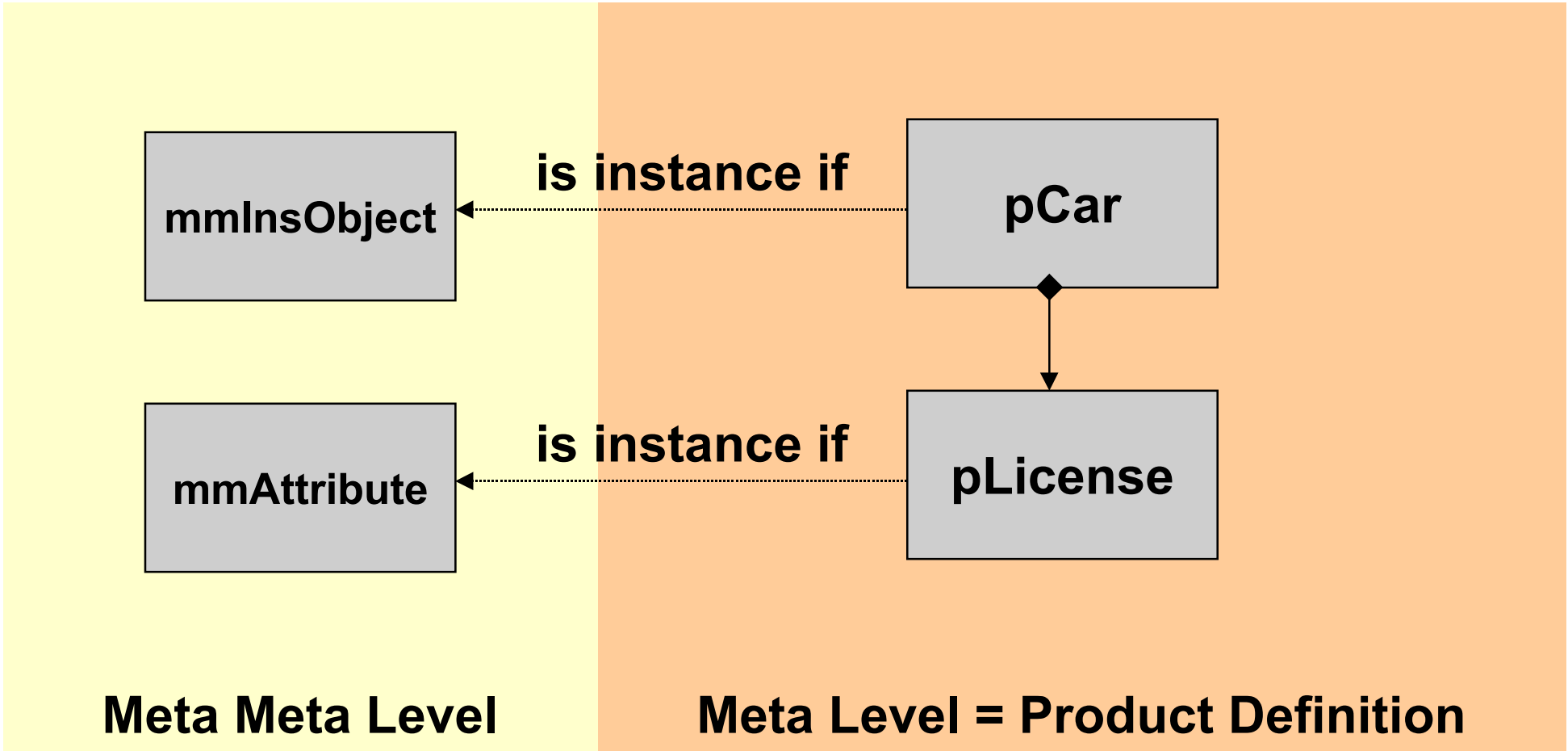
The Reflection Pattern aka. Meta System Running Example: Insured Object



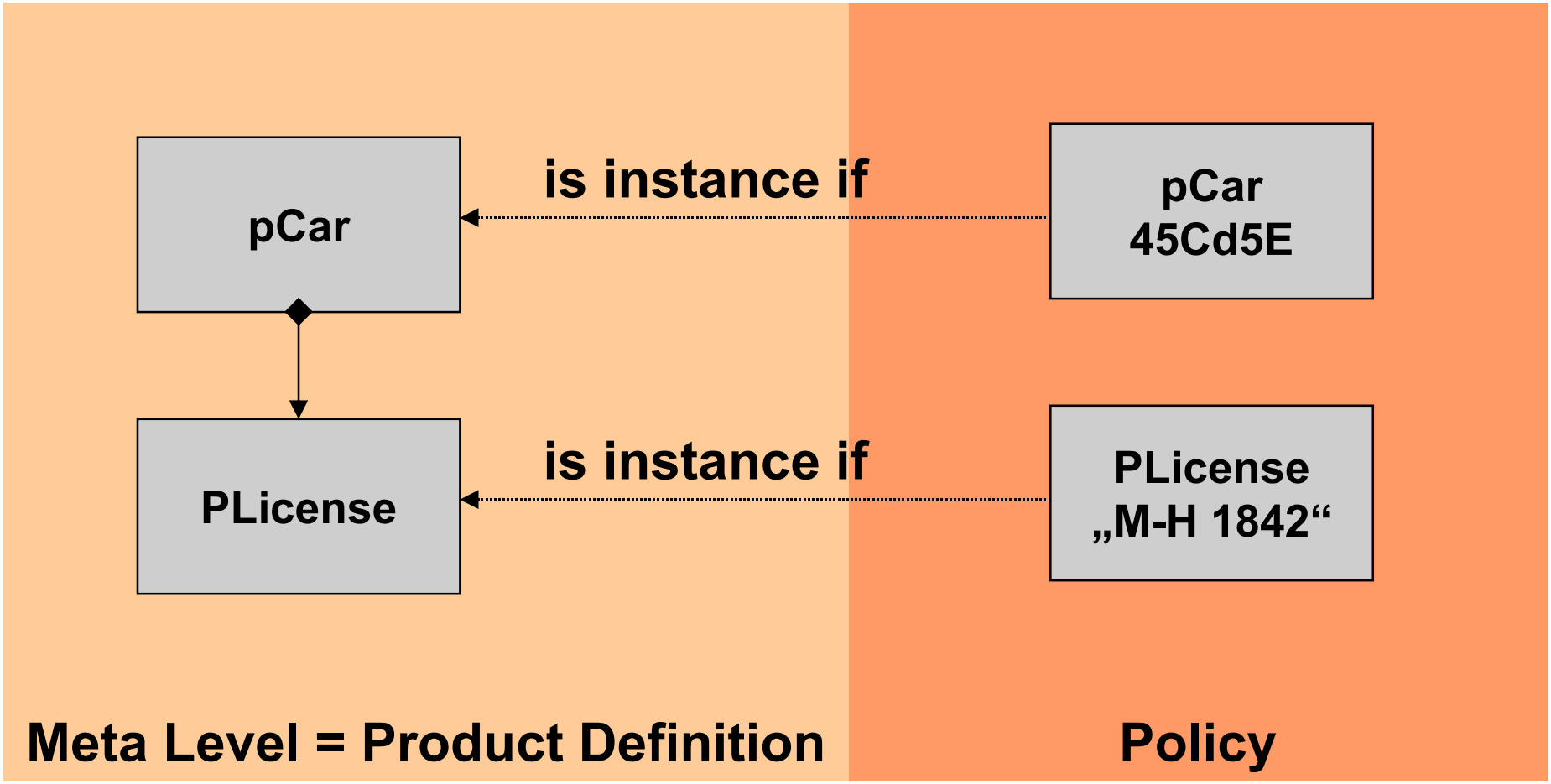
Meta Meta Model



Meta Model
aka Product Definition



Operational Data
aka Policy



How can I implement this in a RelDB Example: Flexible “Insured Objects”

Problem

- I do not want to add a new table for a new type of Insured Objects each time I add a new type of Insured Object (e.g. a Satellite) to the product definition.

Solution

- Use a generic table scheme like ...

Example: Flexible “Insured Objects” Generic Table Scheme

Types of Insured Objects

tObjectID	tObjectType
43B56C	pCar
43B57E	pBicycle
...	

Concrete Insured Objects

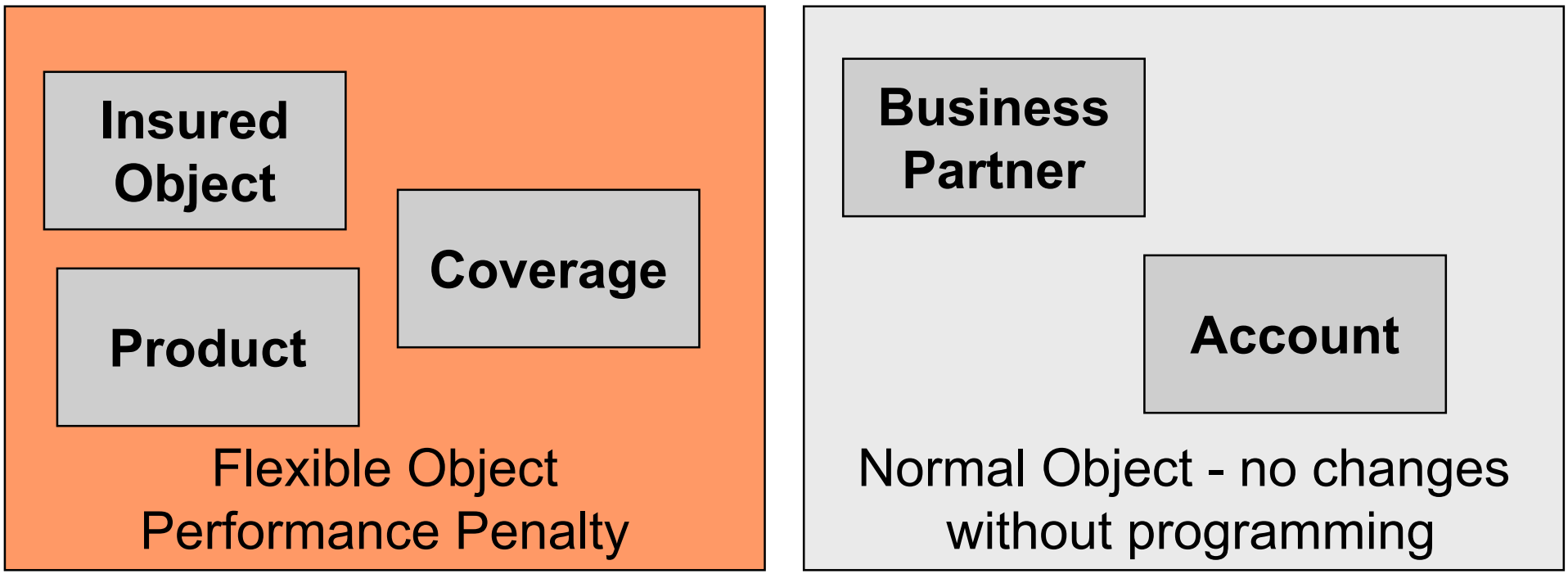
tObjectID	tAttribute	tValue
43B57E	Value	800,42
43B56C	License	M-H 1842
43B56C	Type	VW Golf
43B56C	Power	45
43B56D		

You can do the same for ...

- Coverages
  - Or Call them Events and Indemnities
- Elementary Products
- Sellable Products
- ....

Is everything “meta”? Design here is balancing performance versus flexibility ...

Where do I put my objects???

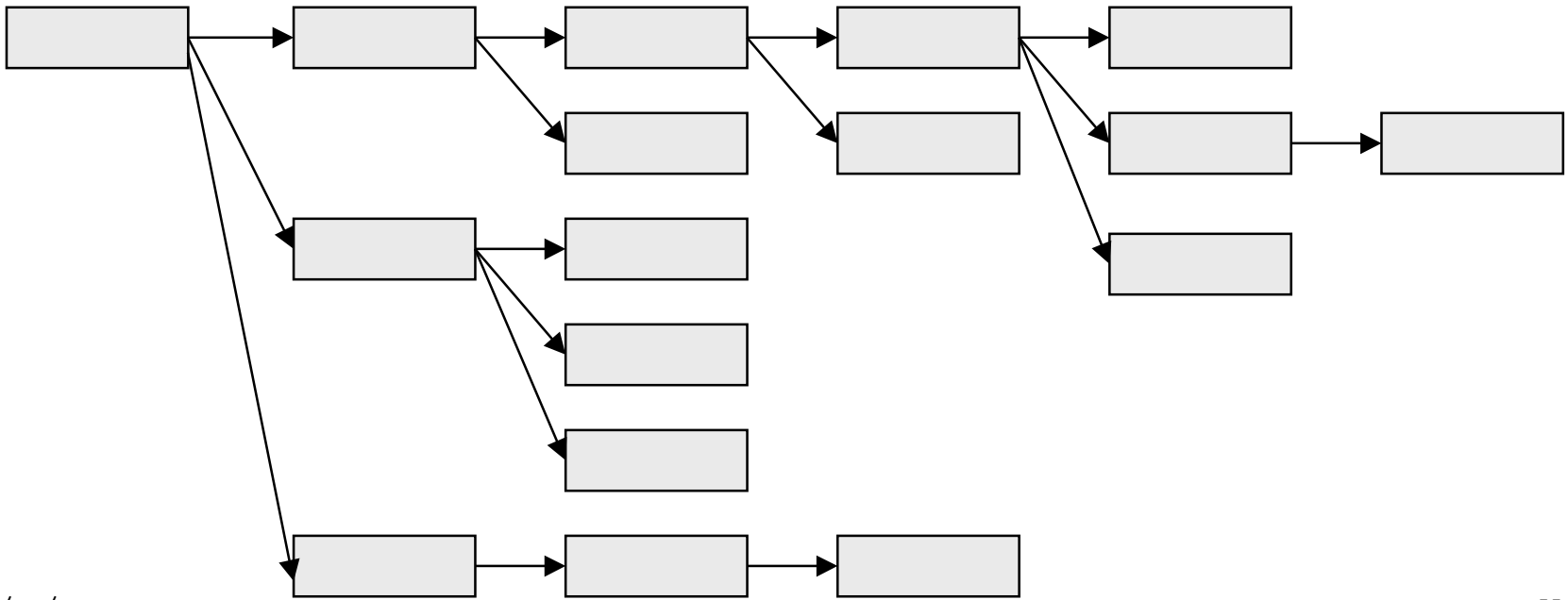


Is everything “meta”?
Tree depth and performance

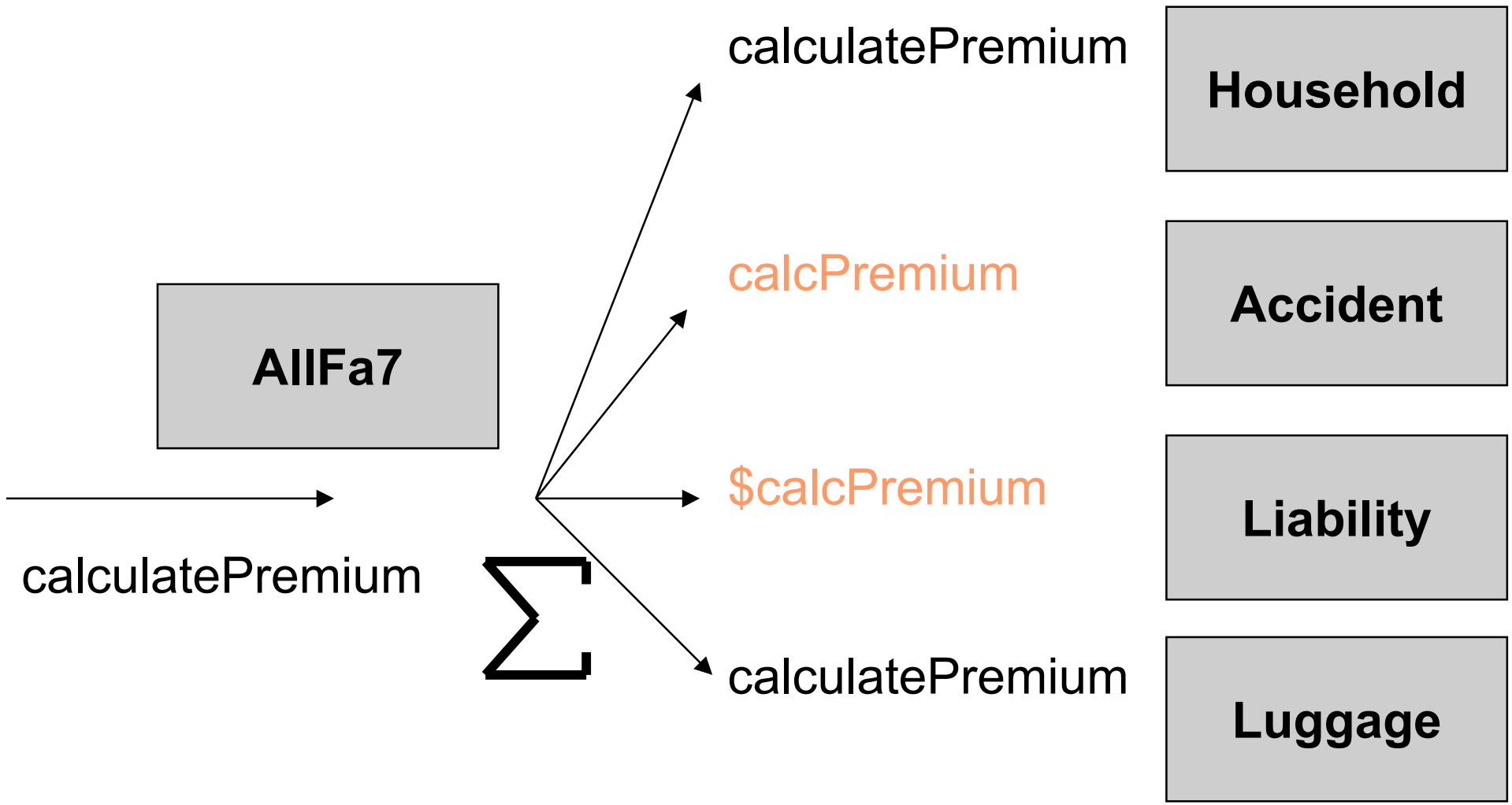
Better performance

depth

Slower System

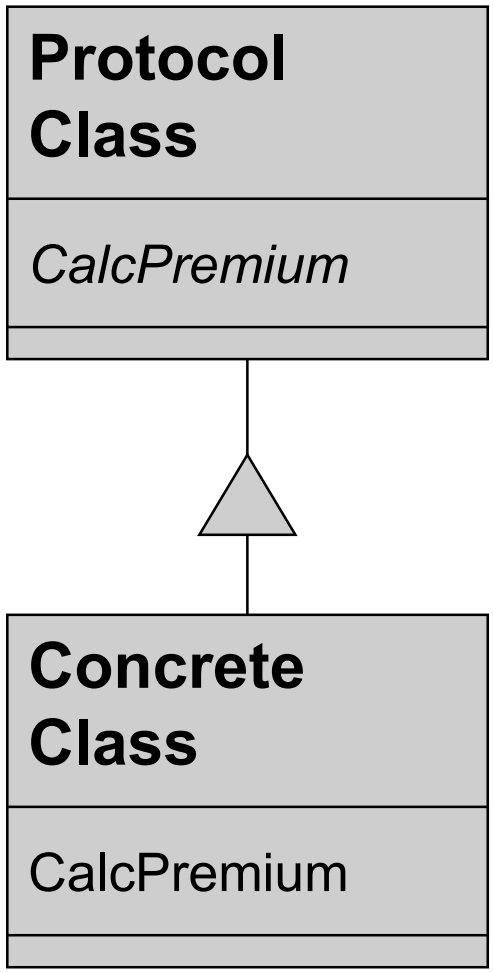


How to provide generic funtionality
Something that does not work



How to provide generic funtionality
Something that does work

- **In order to make product definitions work together with a policy system the model needs to follow a set of conventions that are mastered by the policy system**
- **In Object-Orientation this is often referred to as „protocol inheritance“**

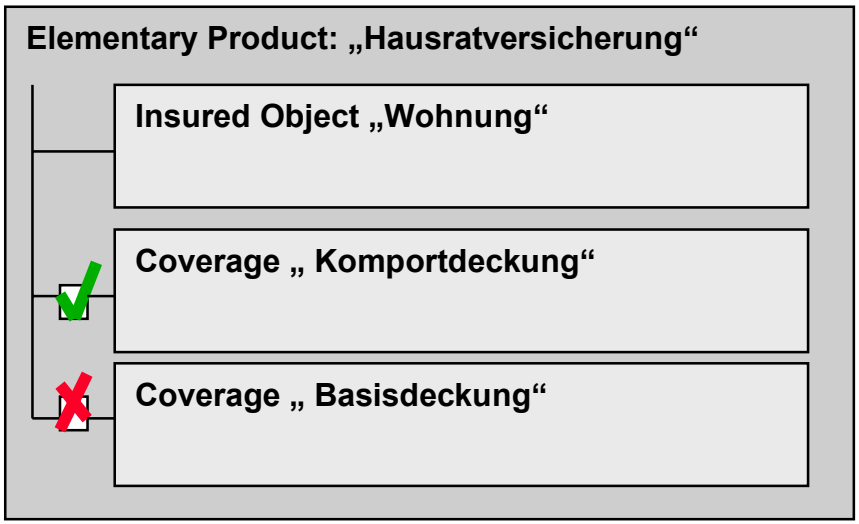


How to provide generic funtionality
Something that does work

- **Restrict tree structures to a defined set of patterns**

- **Like VP/MS style**
- **Like PDFS style**
- **Like VIAS style**
- **Or a collection of styles**

- **But NOT any style**



Literature and Related Stuff

- See the Reflection Pattern
  - **Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal:** *Pattern Oriented Software Architecture - A System of Patterns*, Wiley 1996.
- See the Type Object Pattern
  - **Ralph Johnson, Bobby Woolf: Type Object, in Robert Martin, Dirk Riehle, Frank Buschmann (Eds.): Pattern Languages of Program Design 3. Addison-Wesley 1998.**
- See the Open Implementation Homepage
  - **Gregor Kiczales et al.: Open Implementation Design Guidelines; see http://www.parc.xerox.com/oi/**
- See „Some Patterns for Insurance Systems“
  - **Wolfgang Keller; see http://www.objectarchitects.de/ObjectArchitects/papers/ or the STJA 1998 Proceedings**