TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät für Informatik Institut für Software- und Multimediatechnik

Ringvorlesung Forschungslinie

Beyond Objects

From Object Modeling Via Role Modeling To Satellite Modeling

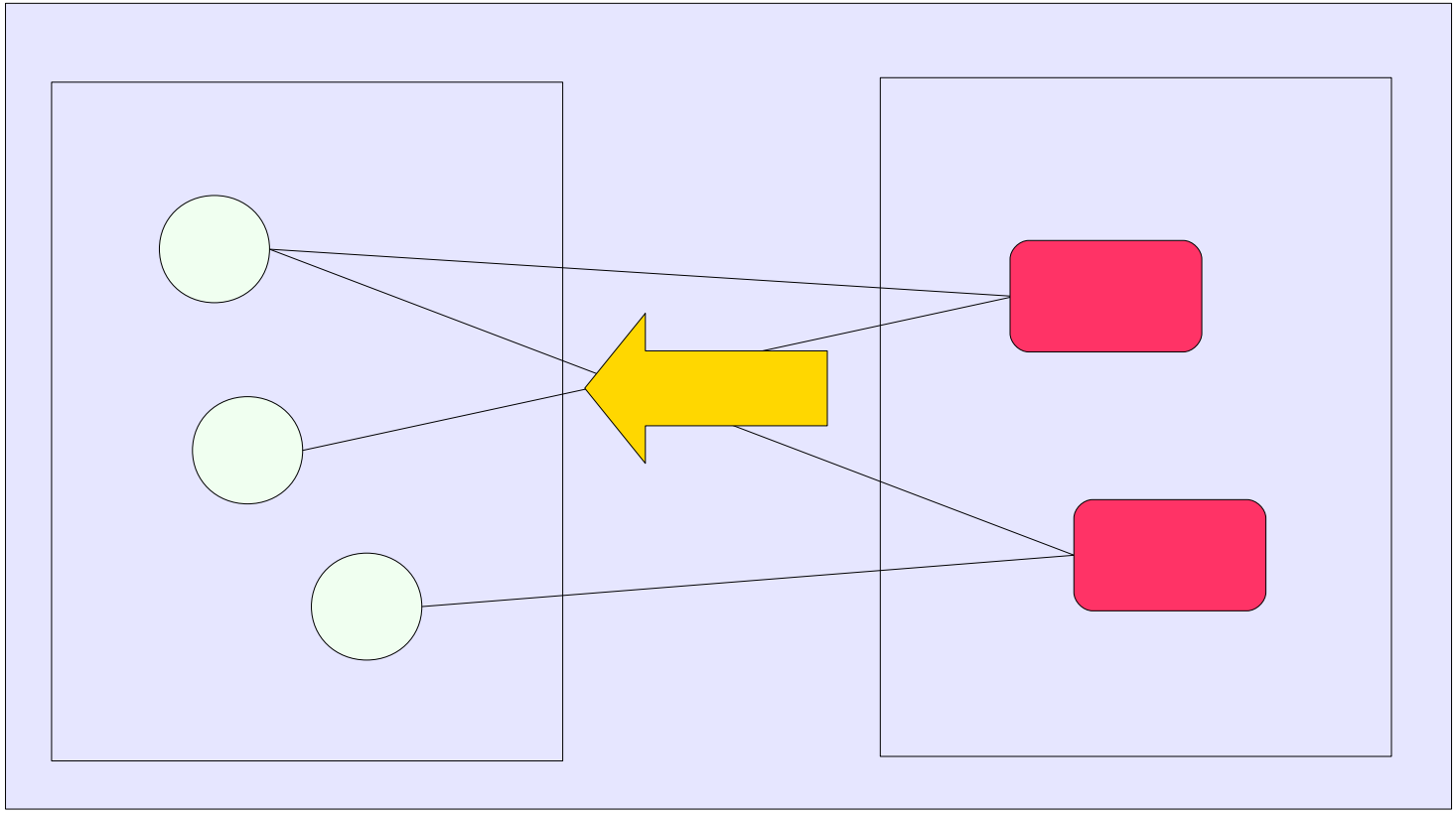
Prof. Dr. Uwe Aßmann
Institut für Software- und Multimediatechnik (SMT)
Fakultät für Informatik, TU Dresden
18-0.2, June 11, 2018

Overview

- Problems
  - The extensibility problem in modeling
  - The variability problem
  - The parallel programming problem (inheritance anomaly)
  - The view problem (what are concerns and aspects?)
  - The component problem (what are components?)
  - The substituability problem (what is conformance?)
- Step 1) The Steimann product-lattice factorization of types
  - Its advantages Scalability Extensibility Variability
- Step 2) Satellite-based development
  - Object-Satellite-Model (ORBIT)
- The relationship to the groups

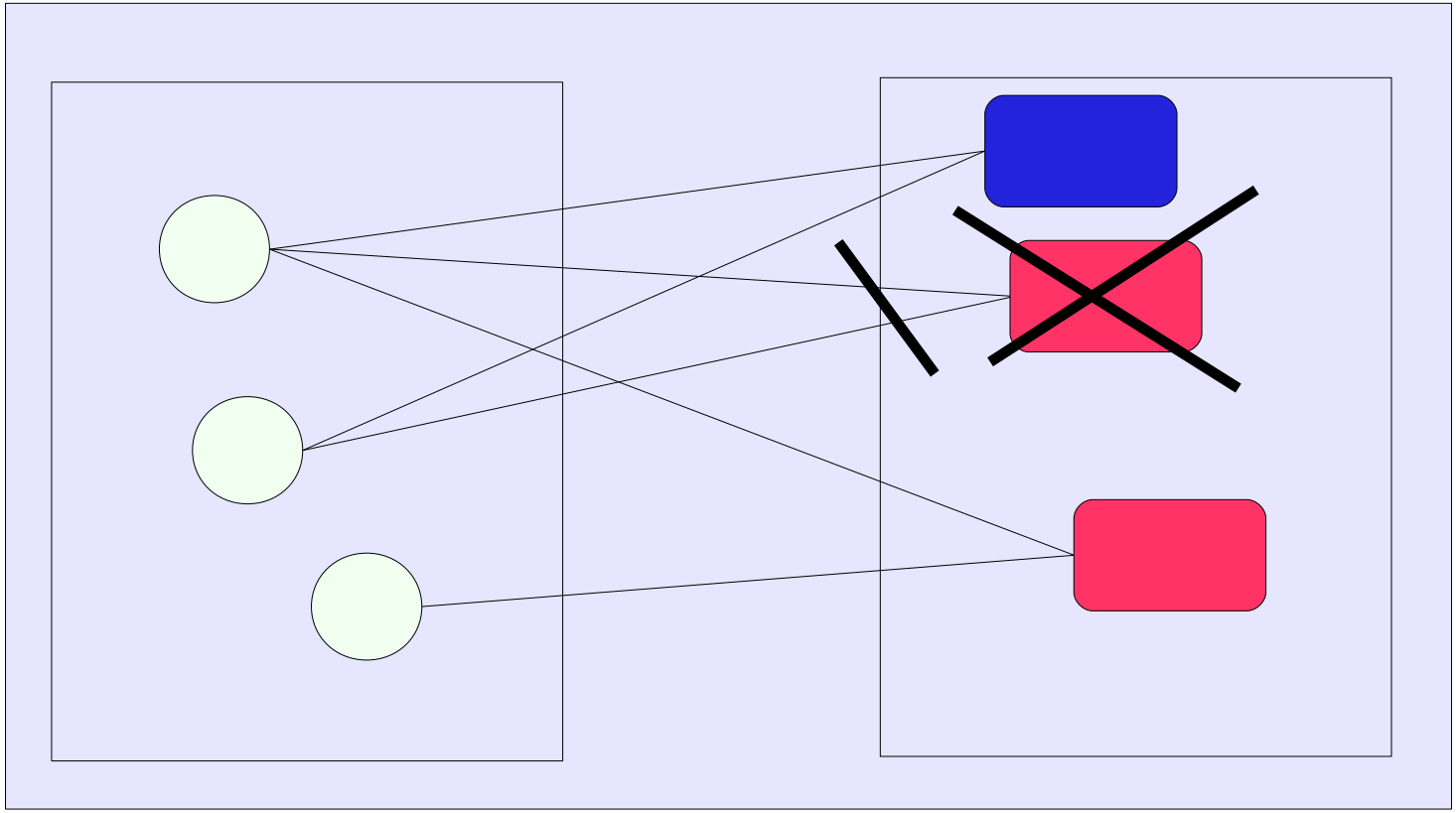
The Extensibility Problem

- How to extend software?



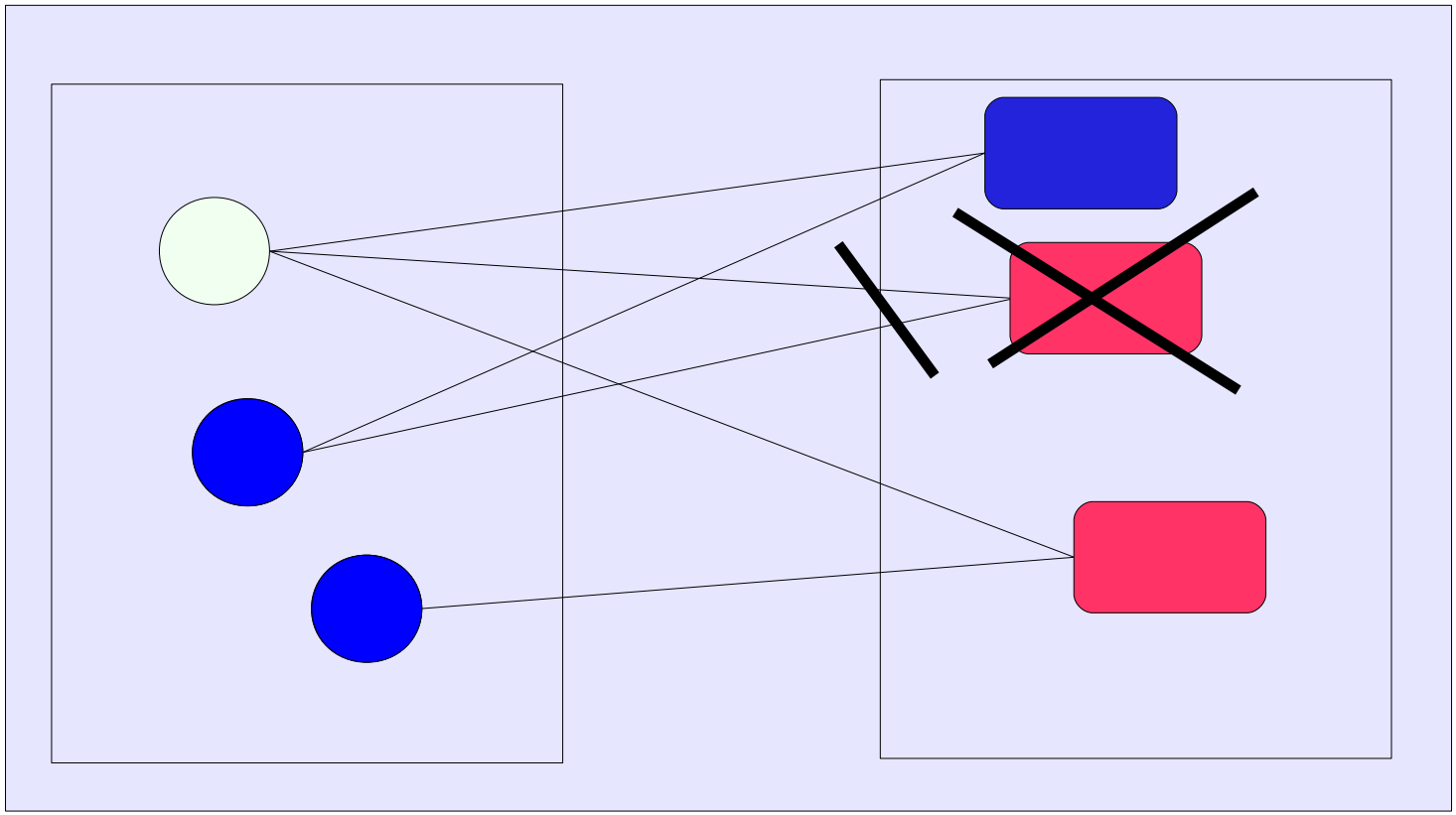
The Substitutability Problem

- How to substitute a component?



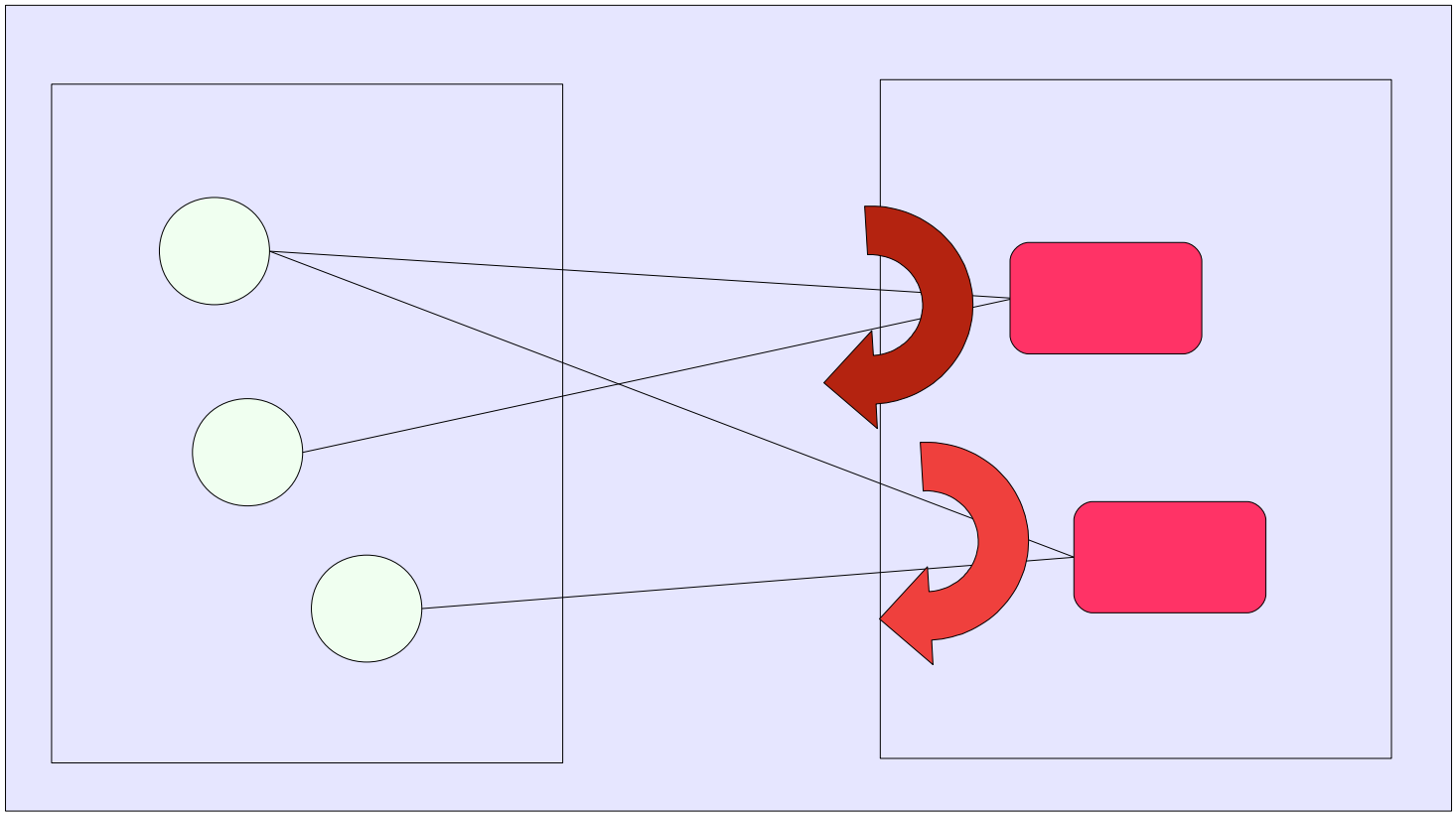
The Varability Problem

- How to vary many components or a subsystem?

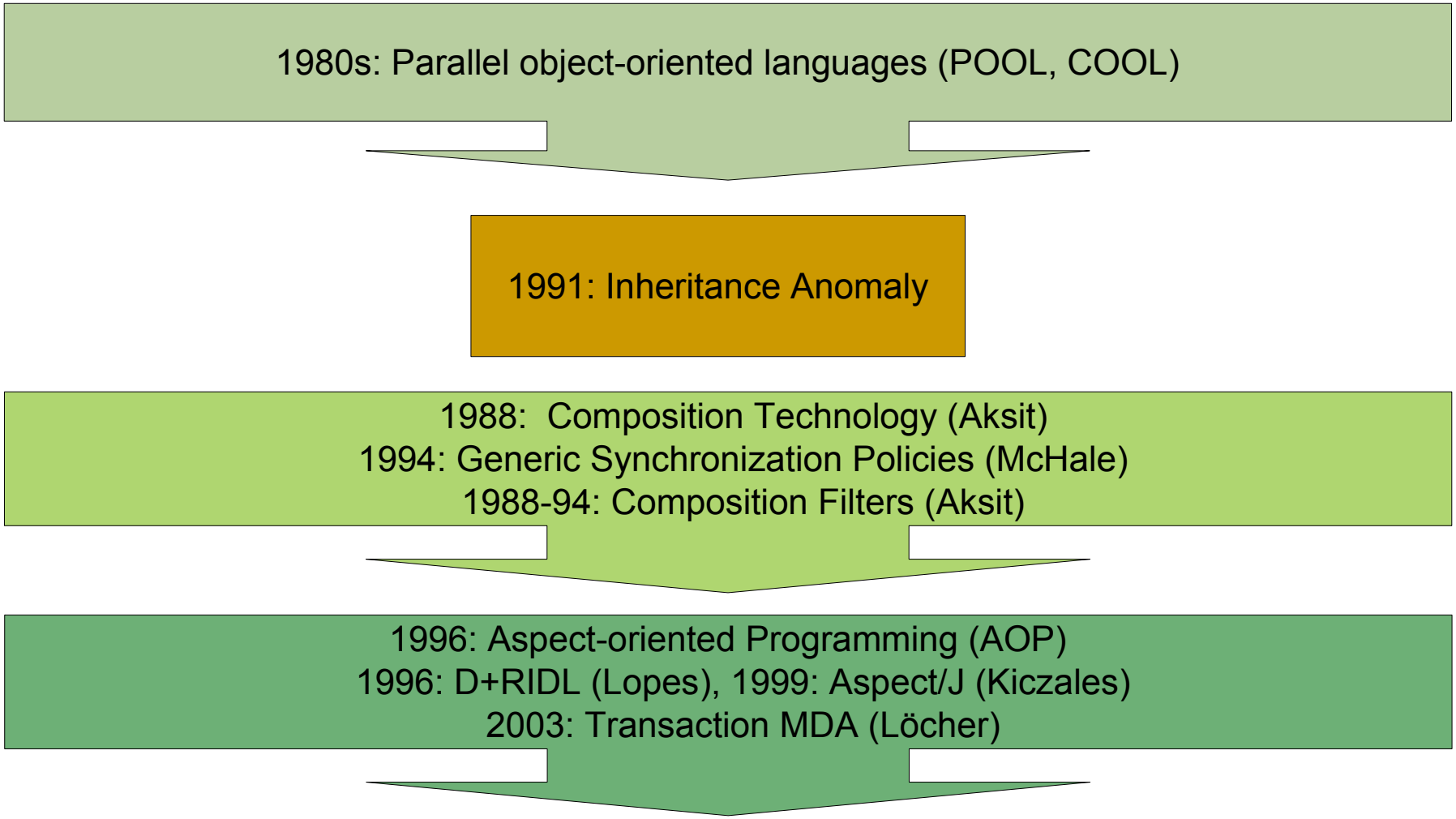


The Wrapping Problem

- How to wrap software with code, e.g., for protection?

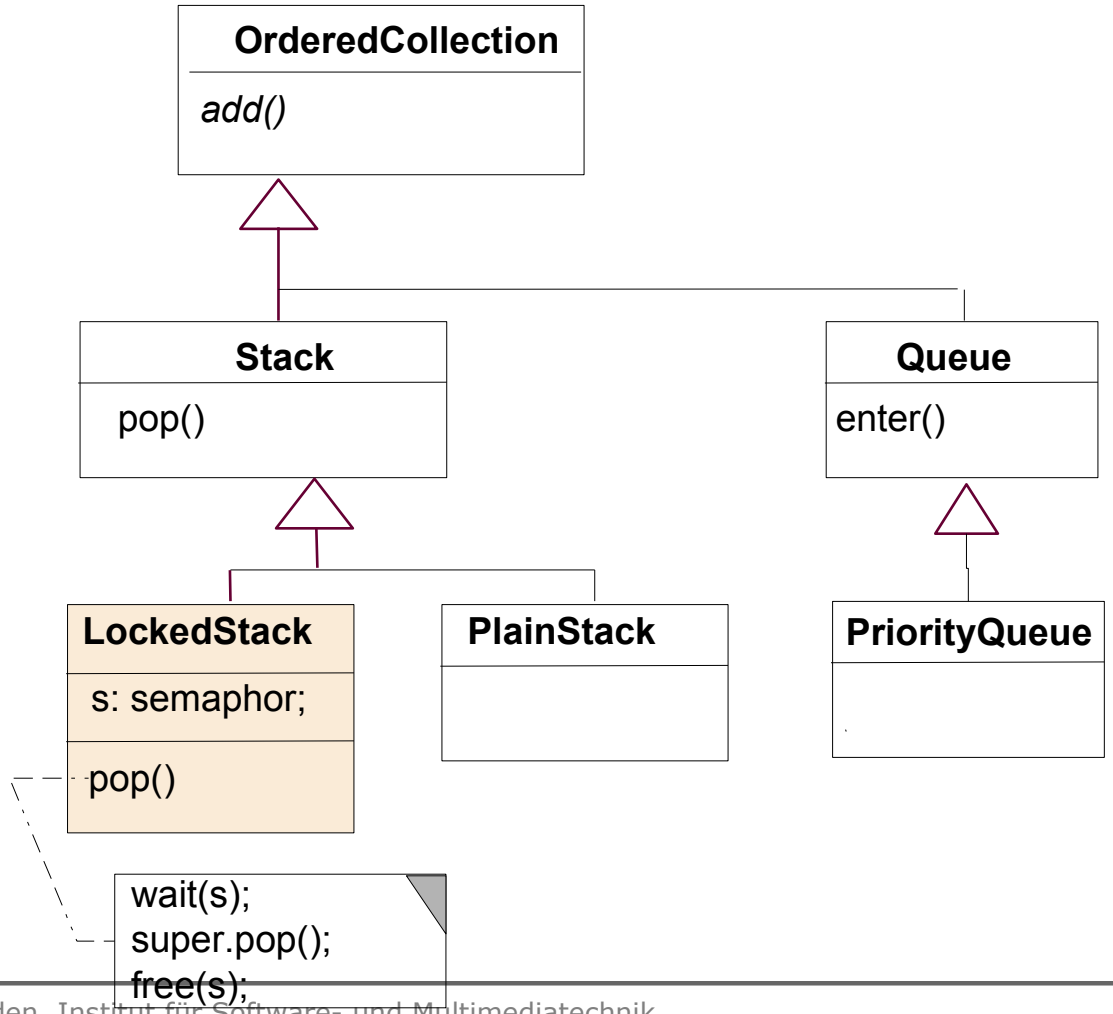


The Parallelism Problem (Inheritance Anomaly)



Inheritance Anomaly - Why Concerns are Necessary

- In a parallel program or library, where should synchronization code be inserted?
  - Stack?
  - Queue?
  - OrderedCollection?
  - Collection?
  - Object?

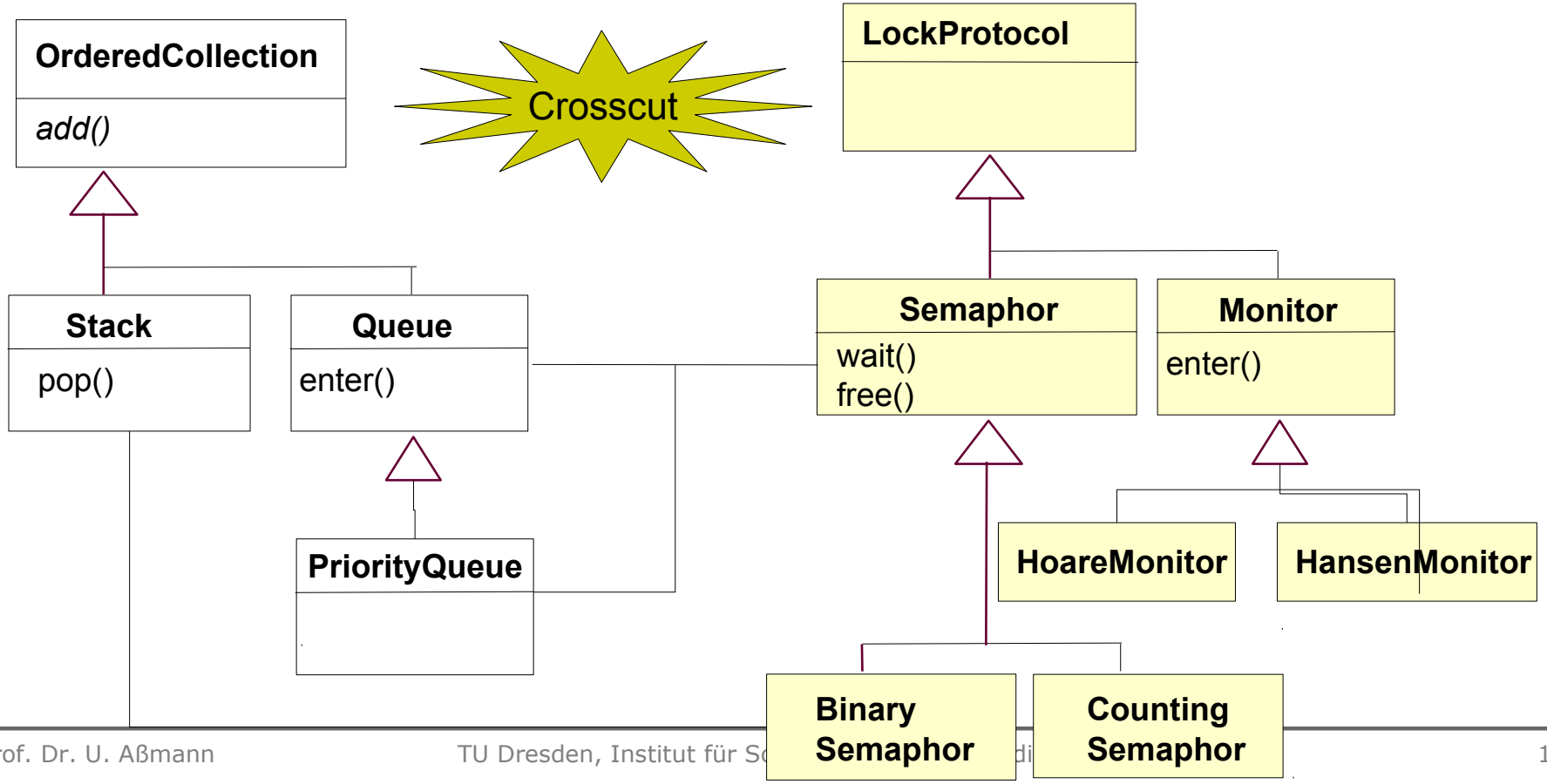


Inheritance Anomaly

- At the beginning of the 90s, parallel object-oriented languages failed, due to the inheritance anomaly
  - *Inheritance anomaly*: In inheritance hierarchies, synchronization code is intermingled with the algorithm and cannot be easily exchanged
  - *Synchronization tangling*: Because synchronization code *braces* code, it is *tangling*
  - *Synchronization crosscut*: Because synchronization code *is reused* code, it is *crosscutting*

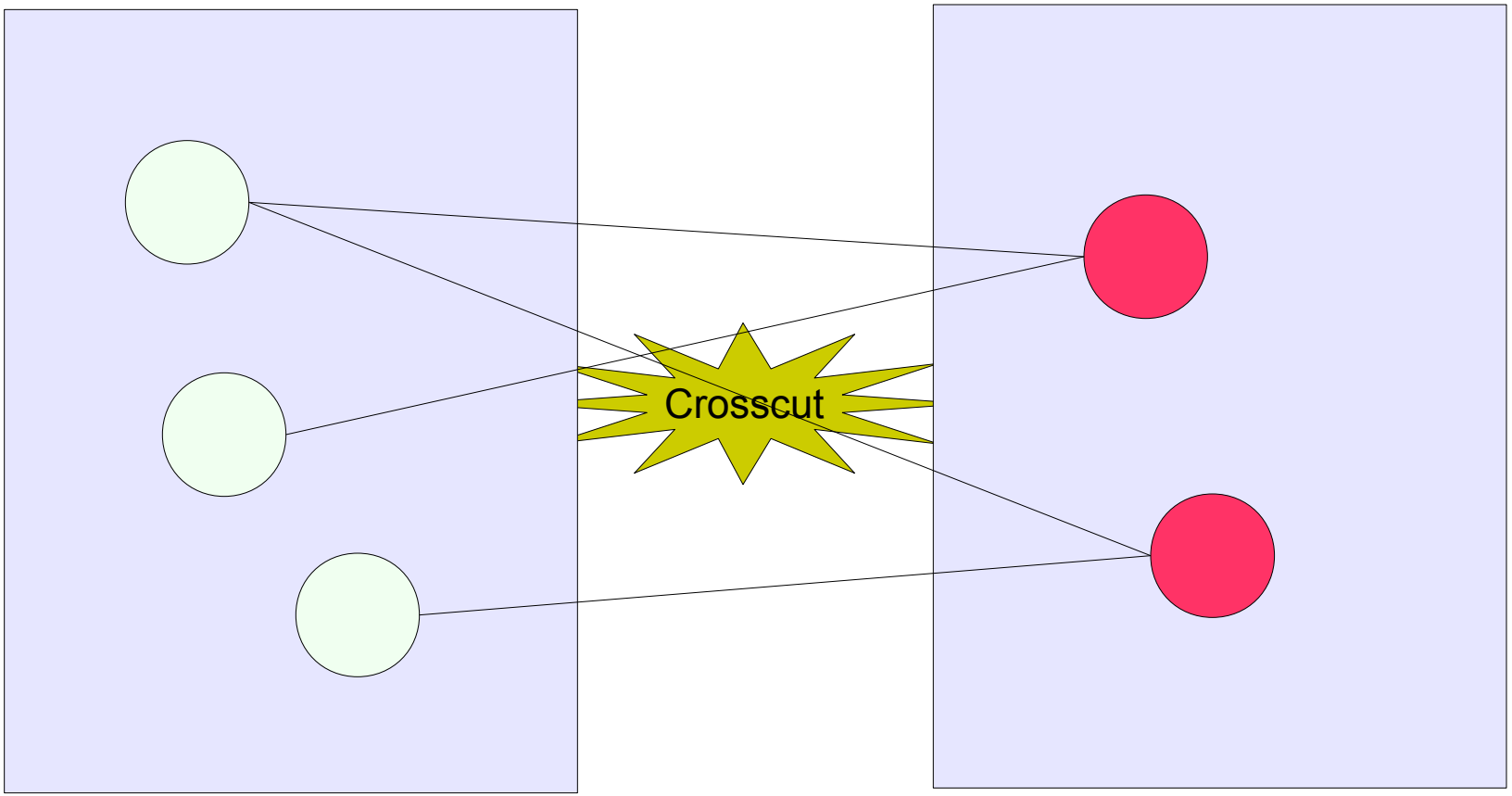
Algorithm and Synchronization are Core and Aspect

- Composition fixes crosscut between core and aspect



What are Crosscut Relations?

- Relations between “things” in a core and an aspect

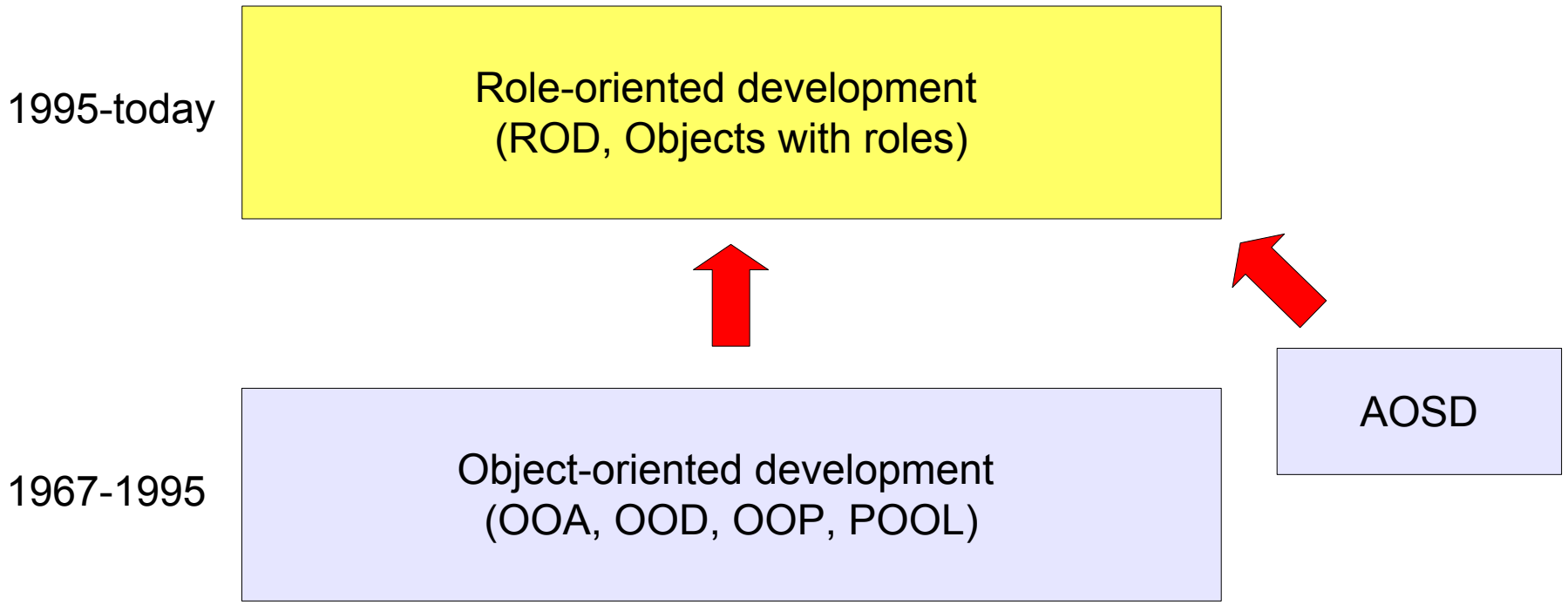


Challenges

What is a *concern (Sicht, Belang)?*

What is an *aspect (querschneidende Sicht, Belang)?*

Ladder of Paradigms



Step 1
Beyond Objects - Role Modeling and the Steimann Factorization

Splitting a type into a tuple of natural and founded parts

Rigid and Founded Types

If an object that has a *rigid* type, it cannot stop being of the type without loosing its identity [Guarini]

- Example:
  - Book is a rigid type, Reader is a non-rigid type
  - Reader can stop reading, but Book stays Book
- Rigid types are *tied to the identity* of objects
  - A *non-rigid type* is a dynamic type that is indicating a state of the object

A *founded type (relative type)* is a type that exists always in collaboration (association) with another class.

Role and Natural Types

A *role type* is a founded and non-rigid type.

Role types are in collaboration and if the object does no longer play the role type, it does not give up identity.

Natural types are non-founded and rigid.

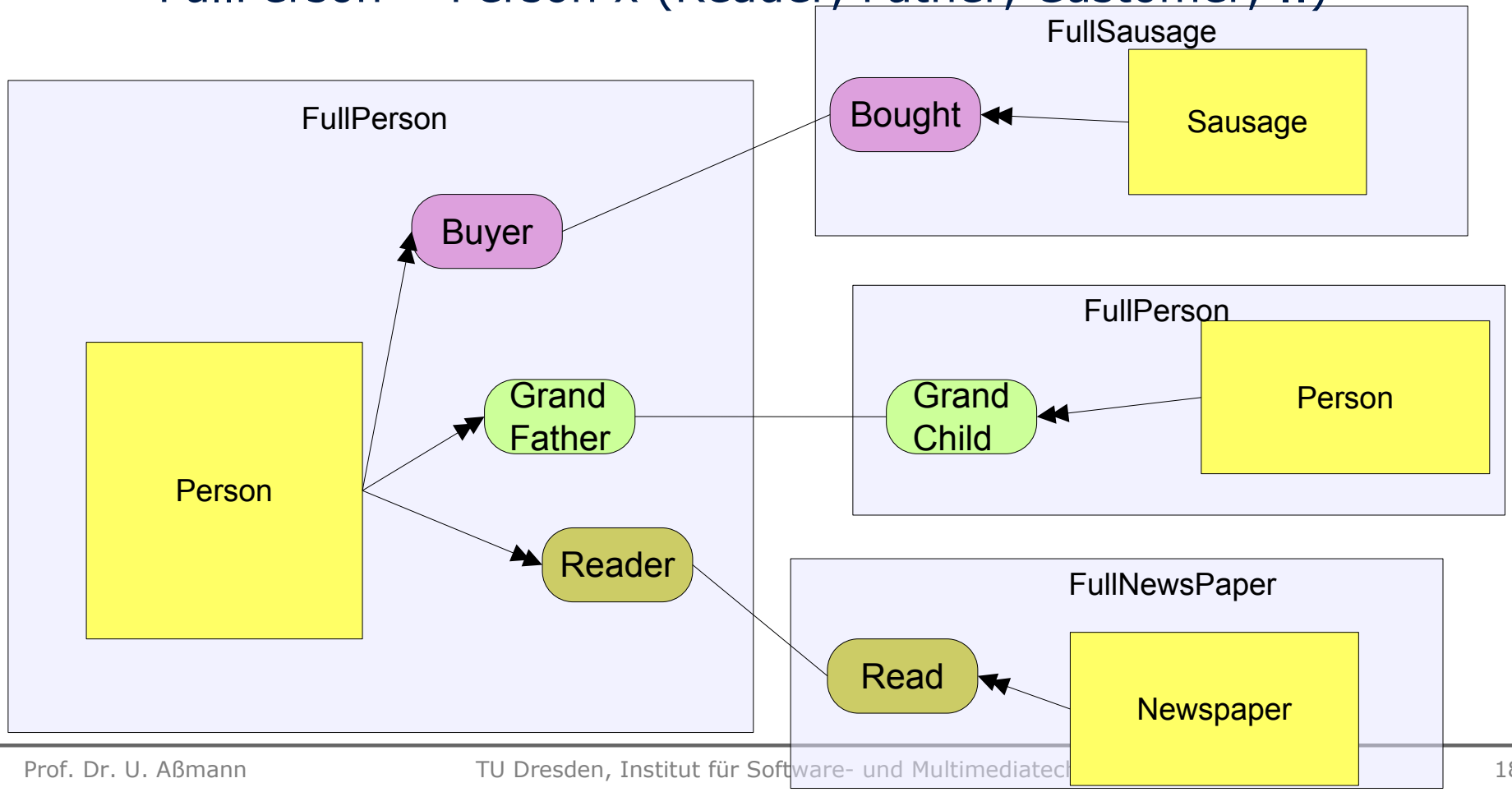
A natural type is *independent* of a relationship.
The objects cannot leave it.

Roles in the Literatur

- Databases (Bachmann 77)
- ER model (Chen 76); though hidden in association ends
- Design patterns (Riehle 98)
  - Course “Design patterns and frameworks” at TUD
- Product line engineering (Smaragdakis,Batory 02)
- Connectors in architectural languages (Garlan, Shaw 95)
- OO Modeling (Reenskaug 95)
- Security anyway
- ACL lists in operating systems
- Ontologies (Brachman, description logic)
- ... [Steimann DKE 2000] has many more and tries to unify them

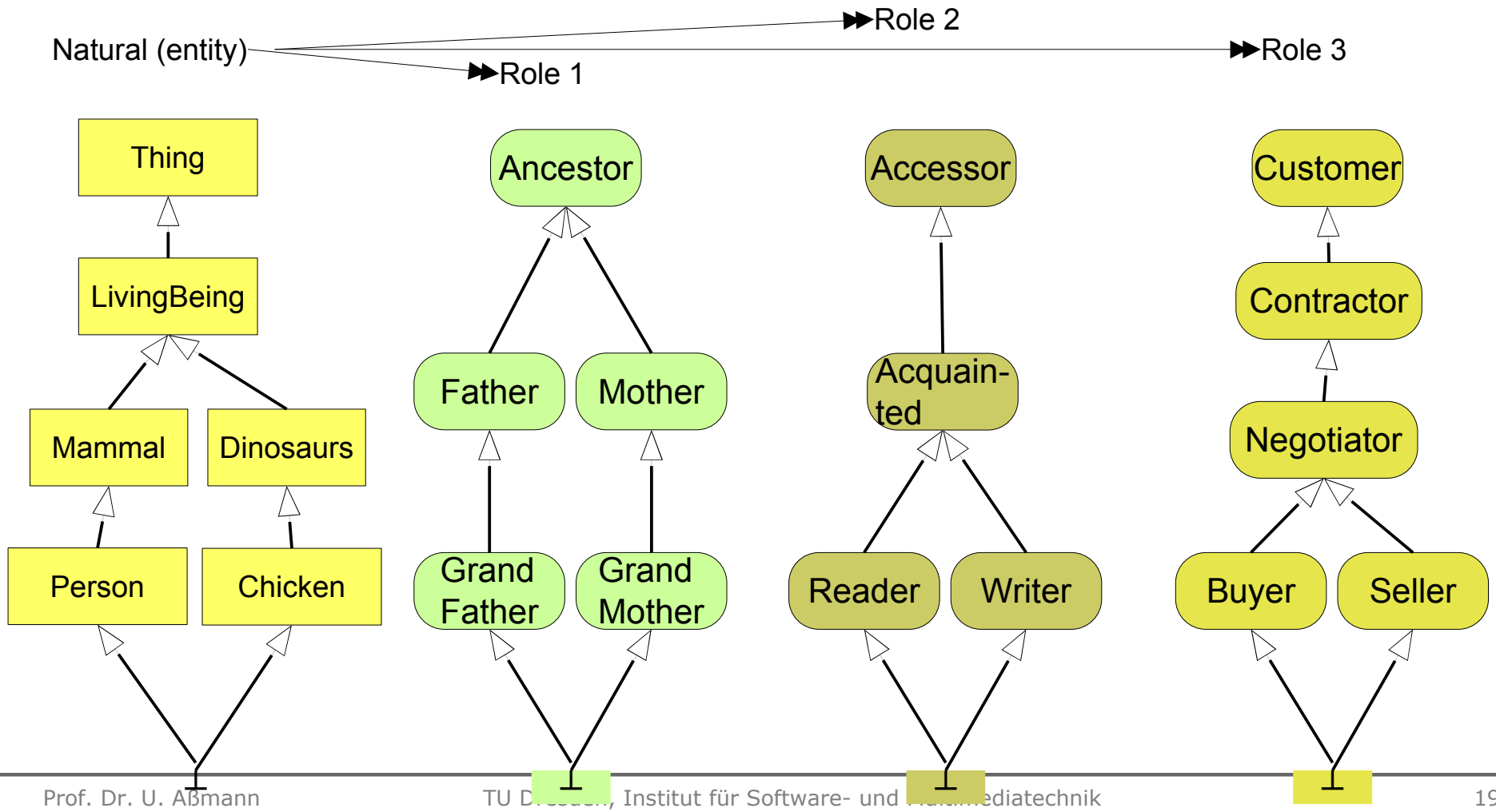
Steimann Factorization [Steimann, DKE 2000]

- Splitting a full type into its *natural* and *role-type* components
  - FullType = Natural x (role-type, role-type, ...)
  - FullPerson = Person x (Reader, Father, Customer, ..)



Full Type is from Inheritance Product Lattice

- What is a reading buying grandfather person?

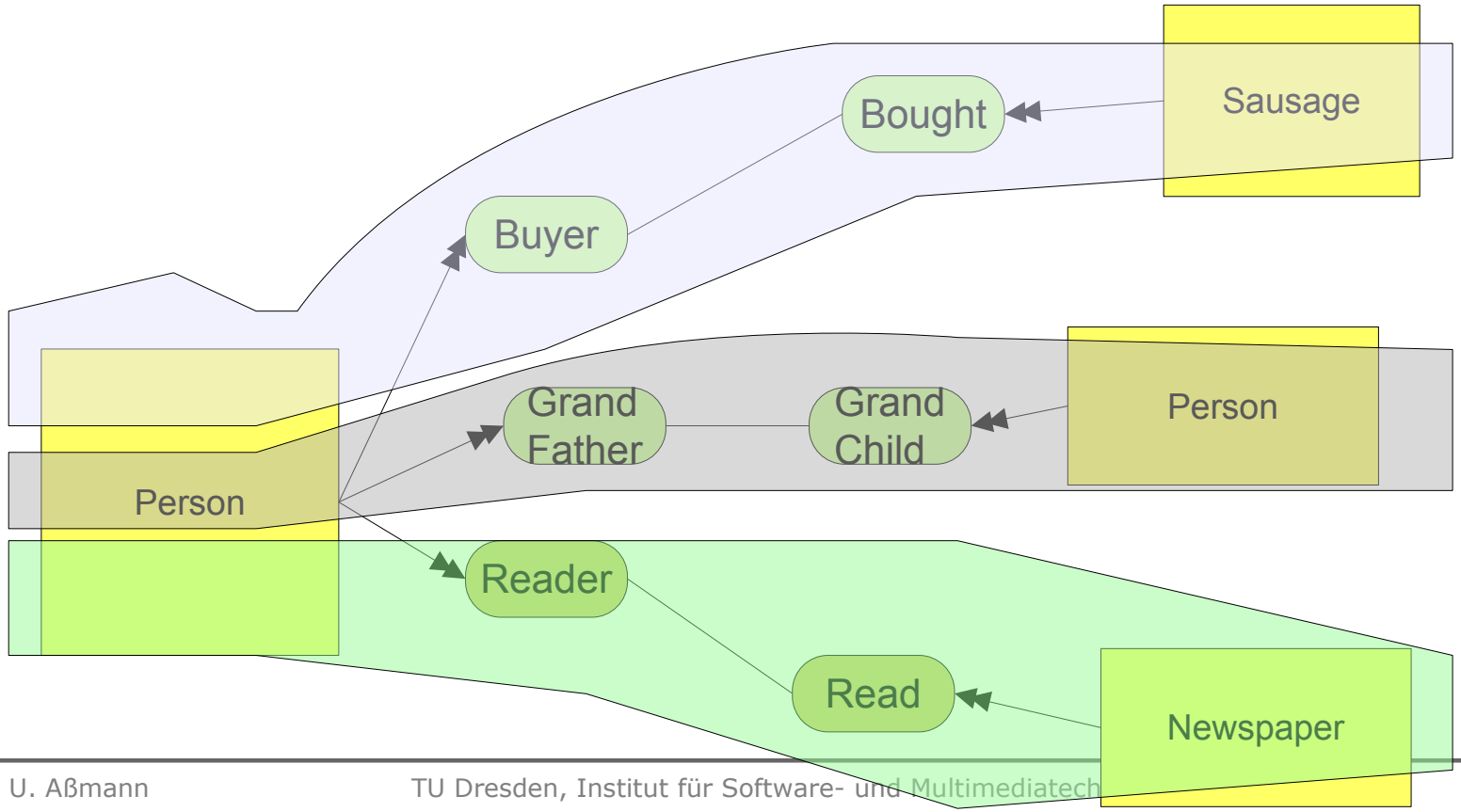


The Steimann Factorization

Divide a *type* into a *tuple type* over a
product lattice of a core dimension and n-1 role dimensions
(Core, Role_1, ..., Role_n)

Simplified Representation of Object Nets

- *Role models (collaborations)* are *interprocedural slices*
- Collaboration schemas (connector schemas) are schemas (types) for interprocedural slices



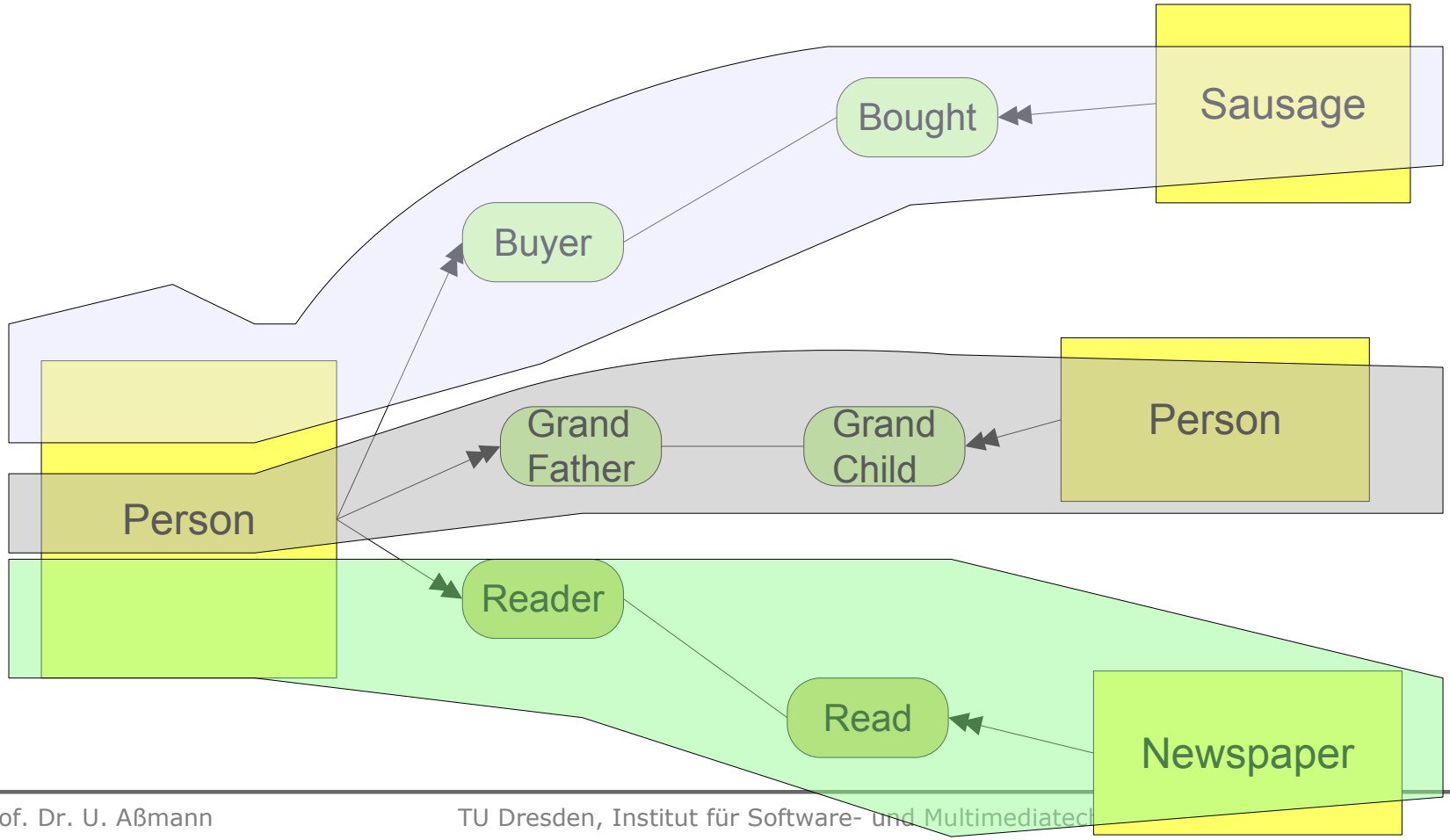
Advantages of the Steimann Factorization for System Construction

- Simplicity
- Representation of concerns and aspects
- Extensibility
- Substitutability (of roles and role models)
- Variability (delayed role embedding decisions)

Advantage 1: Extensibility

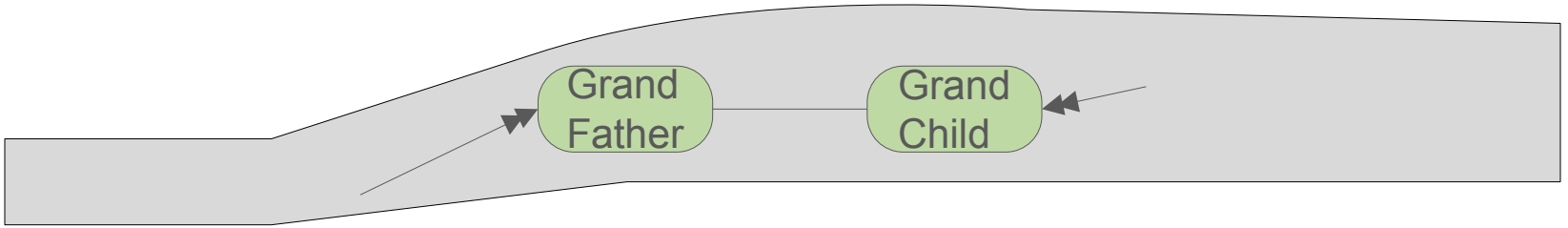
Simplified Extension with Connectors (Role Model, Collaboration)

- Object-role nets can be *extended by* new role models (connectors, collaborations)

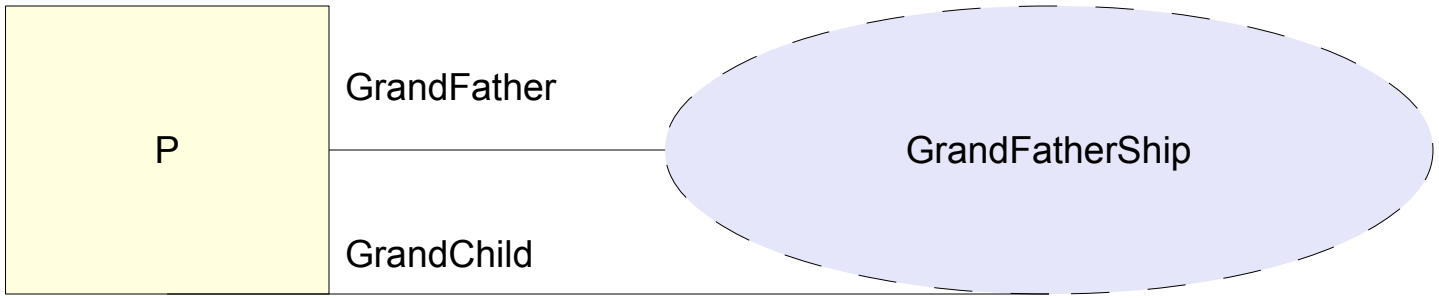


A Connector is a Relational Module (Collaboration)

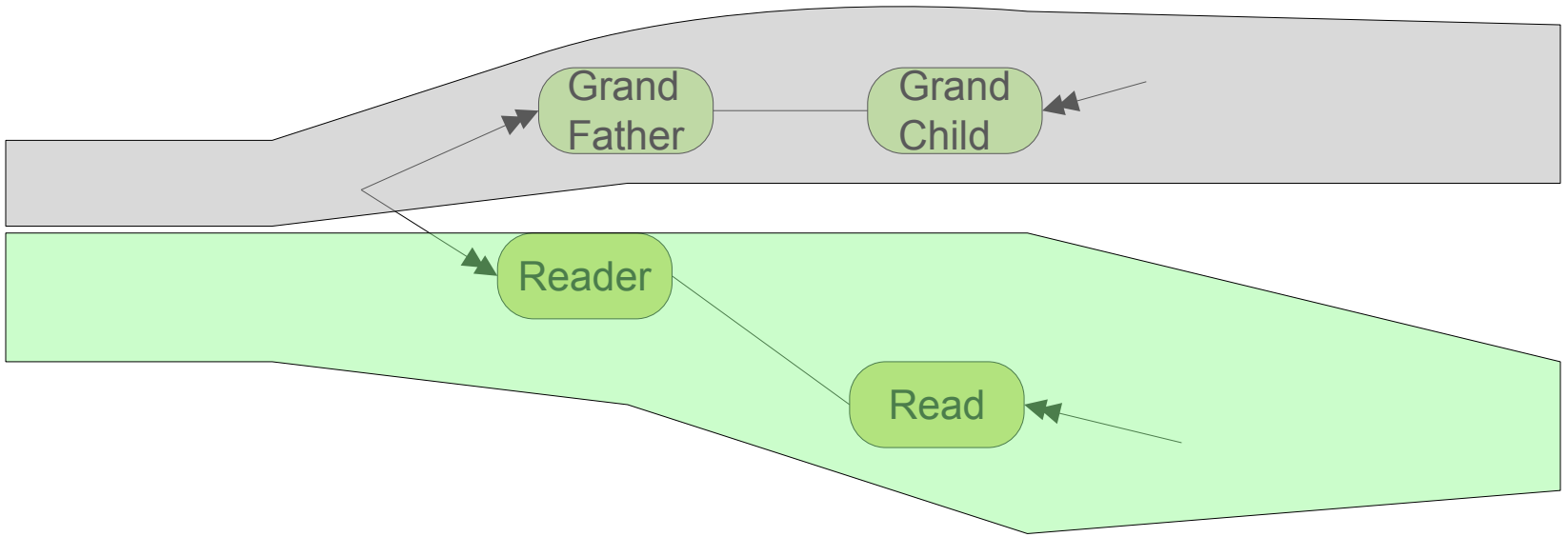
- Nets of roles with open ends, open *plays-a* tentacles,
  - to be attached to object cores



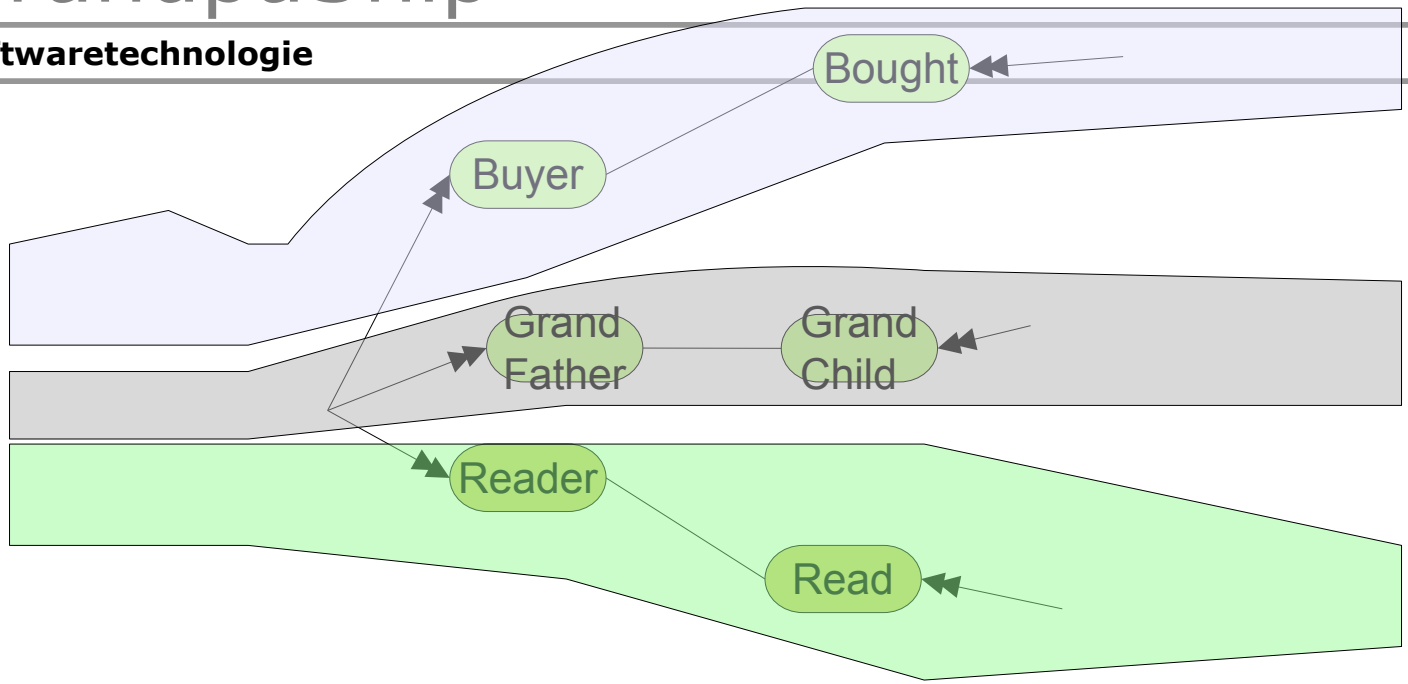
  - UML Notation with *role-type parameter* P:

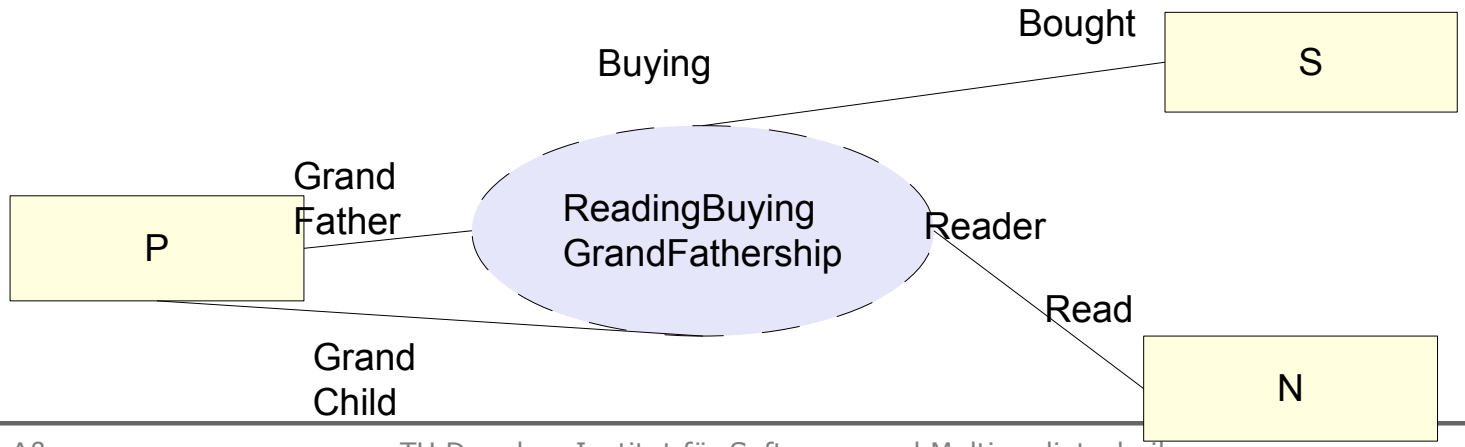


Newspaper-Reading GrandpaShip



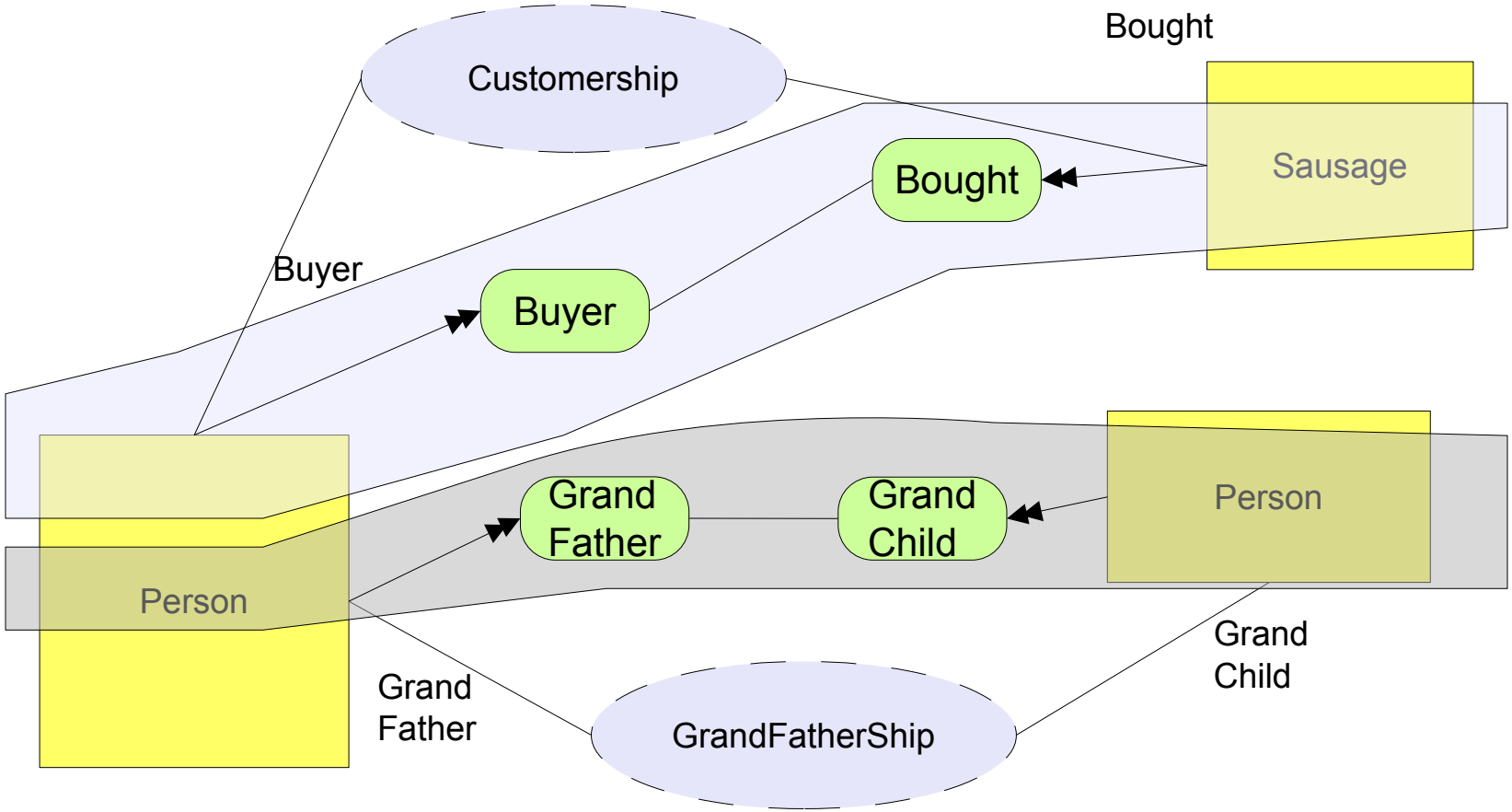
Newspaper-Reading Buying GrandpaShip



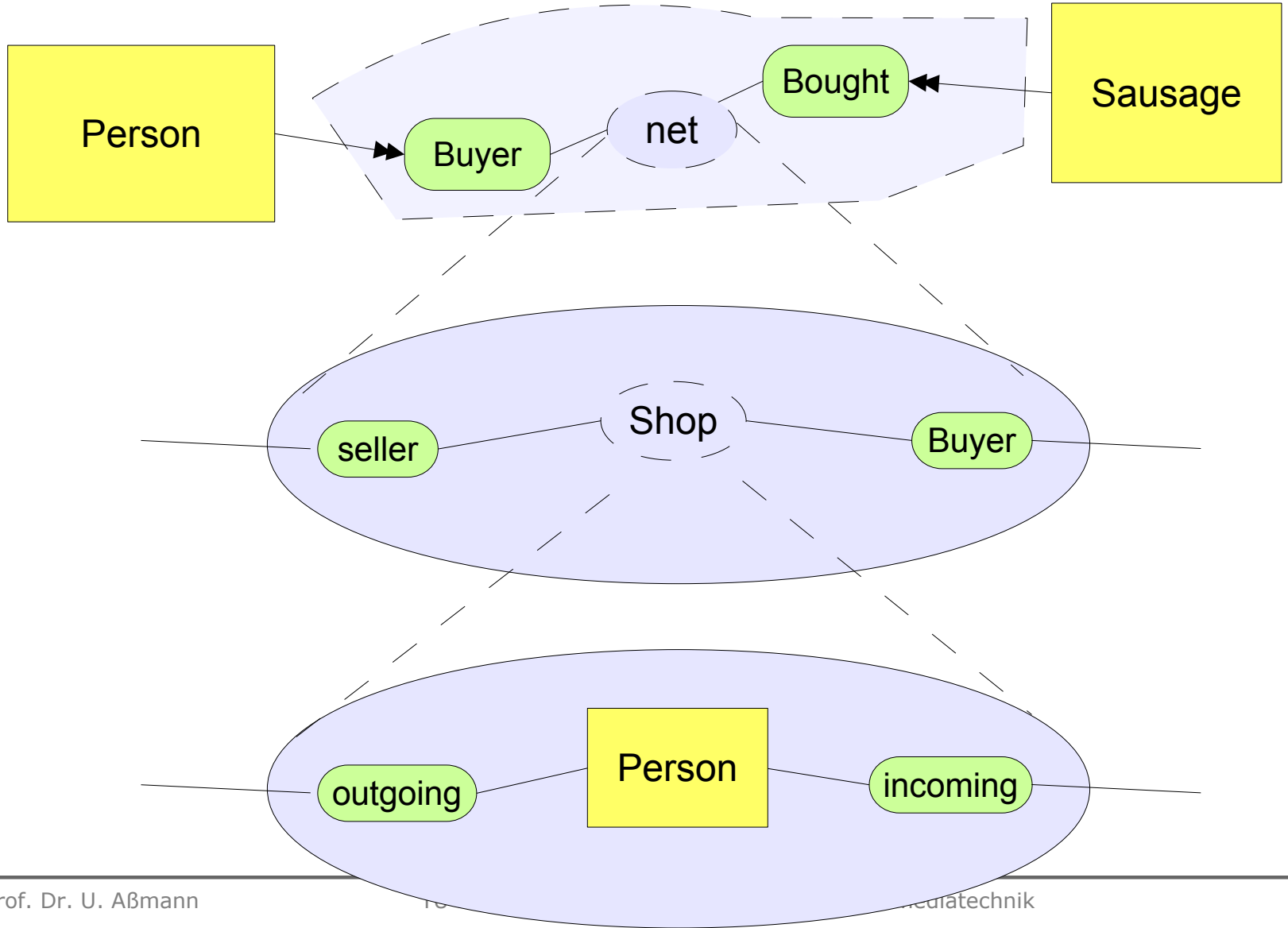


Connector Superimposition in UML

- Connector superimposition extends designs

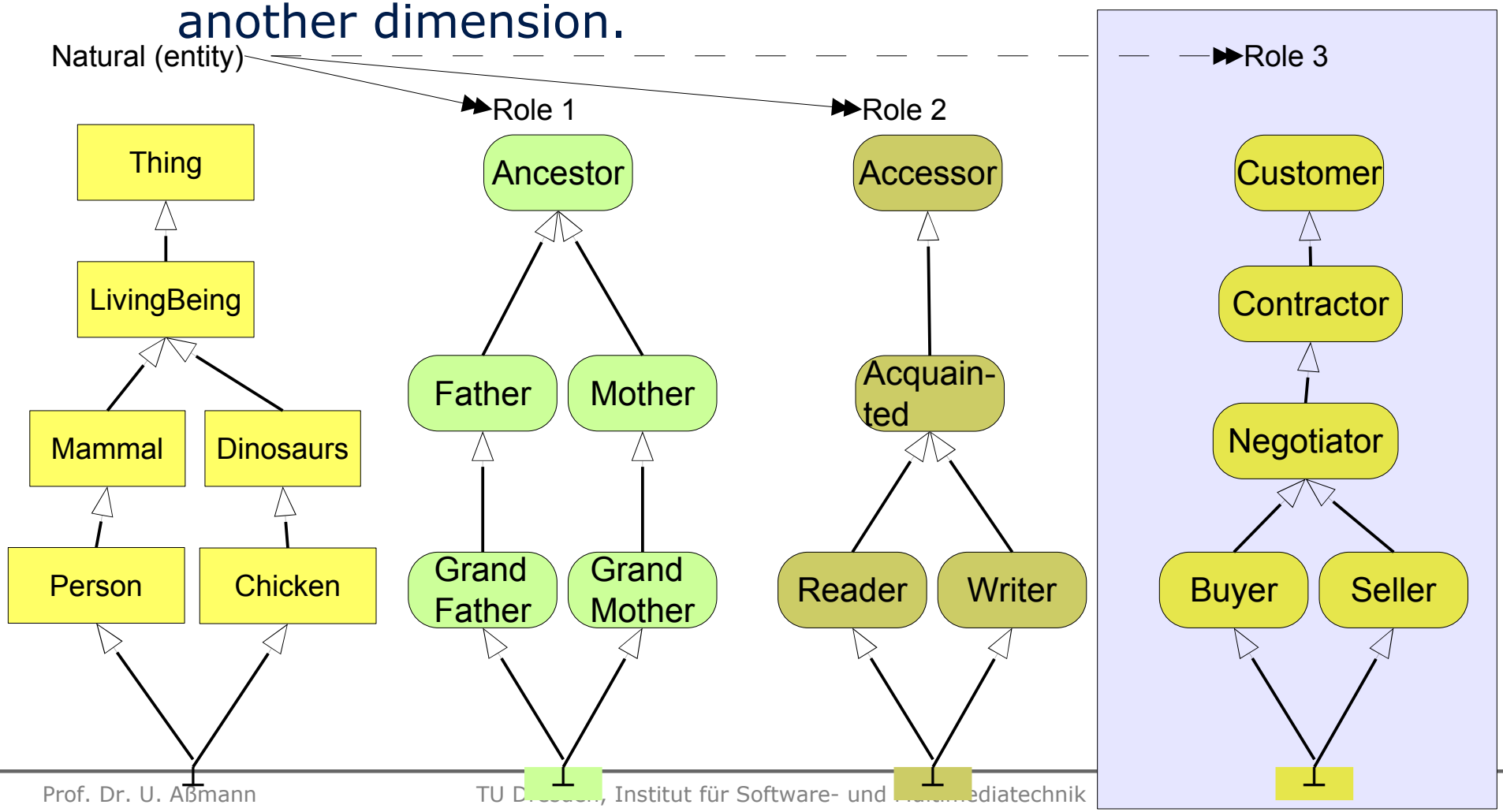


Connectors can be Refined

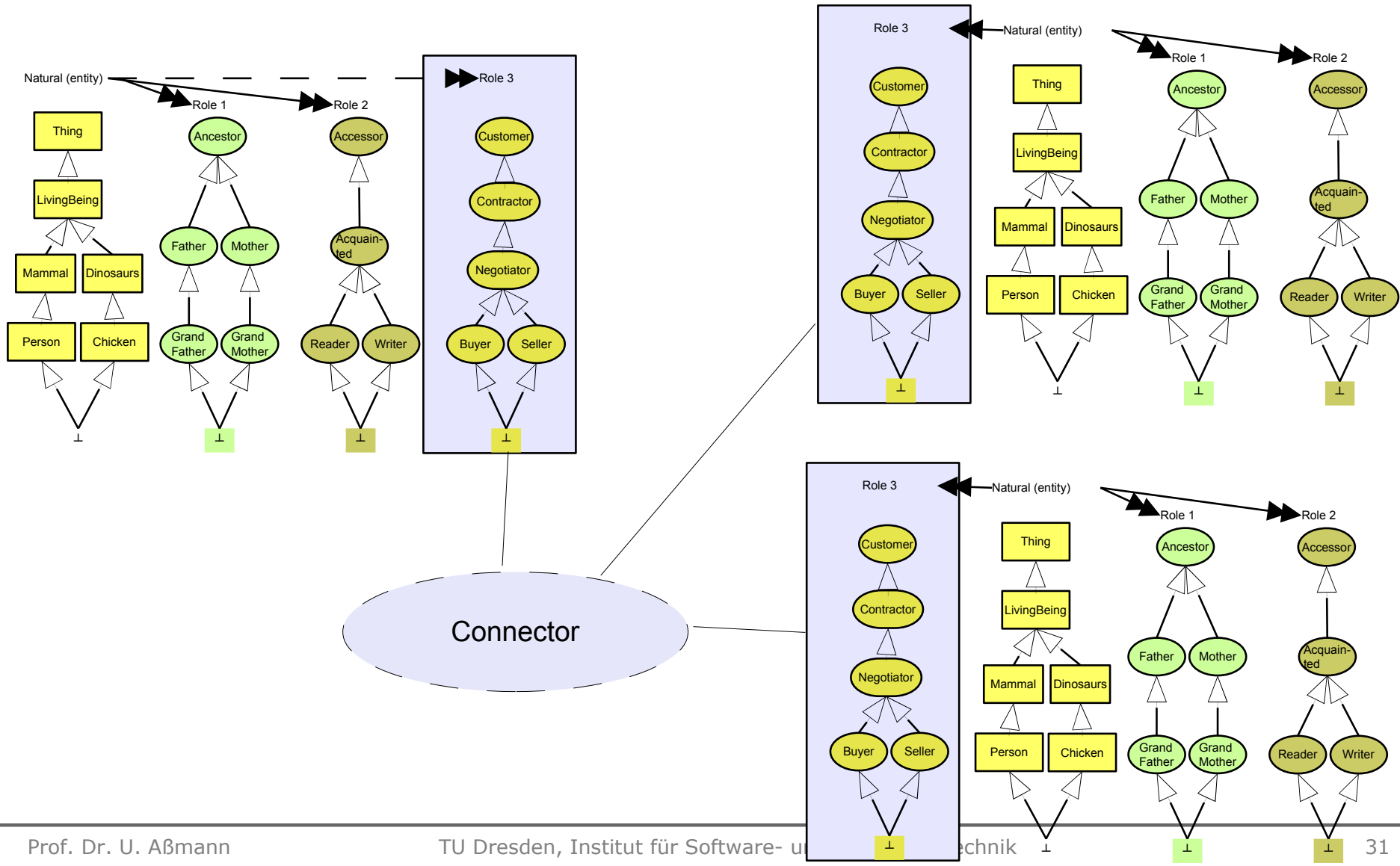


Extension on the Steimann Product Lattice

- A new role relationship extends the product lattice by another dimension.



Connector Superimposition extends the Steimann Lattices of all involved Classes



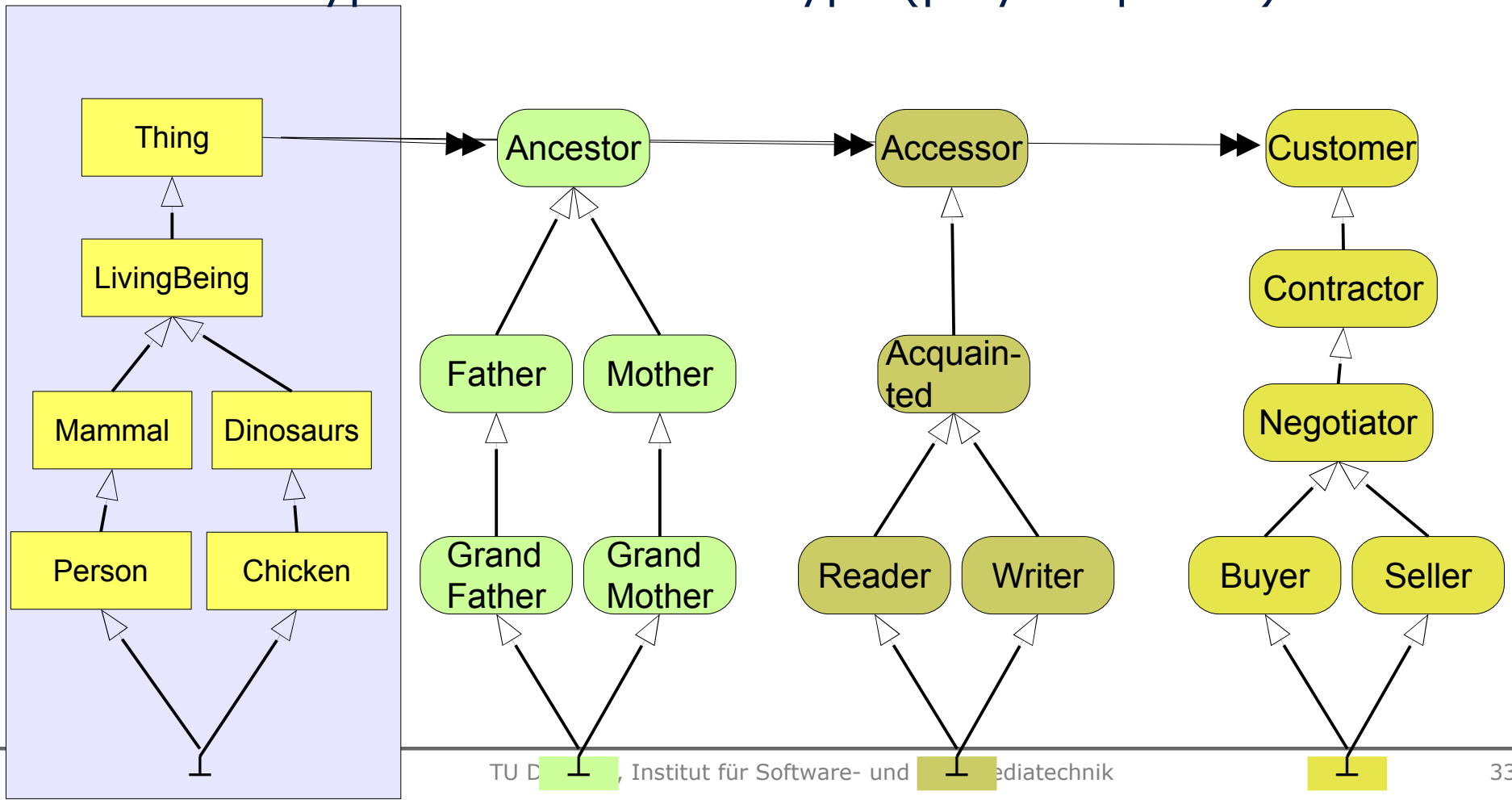
Extension in the Steimann Lattice Retains Inheritance

Superimposition of connectors to objects in Steimann-factored form retains all inheritance structures

- Stable entity inheritance hierarchies, if concepts are added *relationally* to a model
  - Otherwise: extension of superclasses necessary (role classes become superclasses of entity classes)
  - Adding of new *concerns* is simple (adding a collaboration)

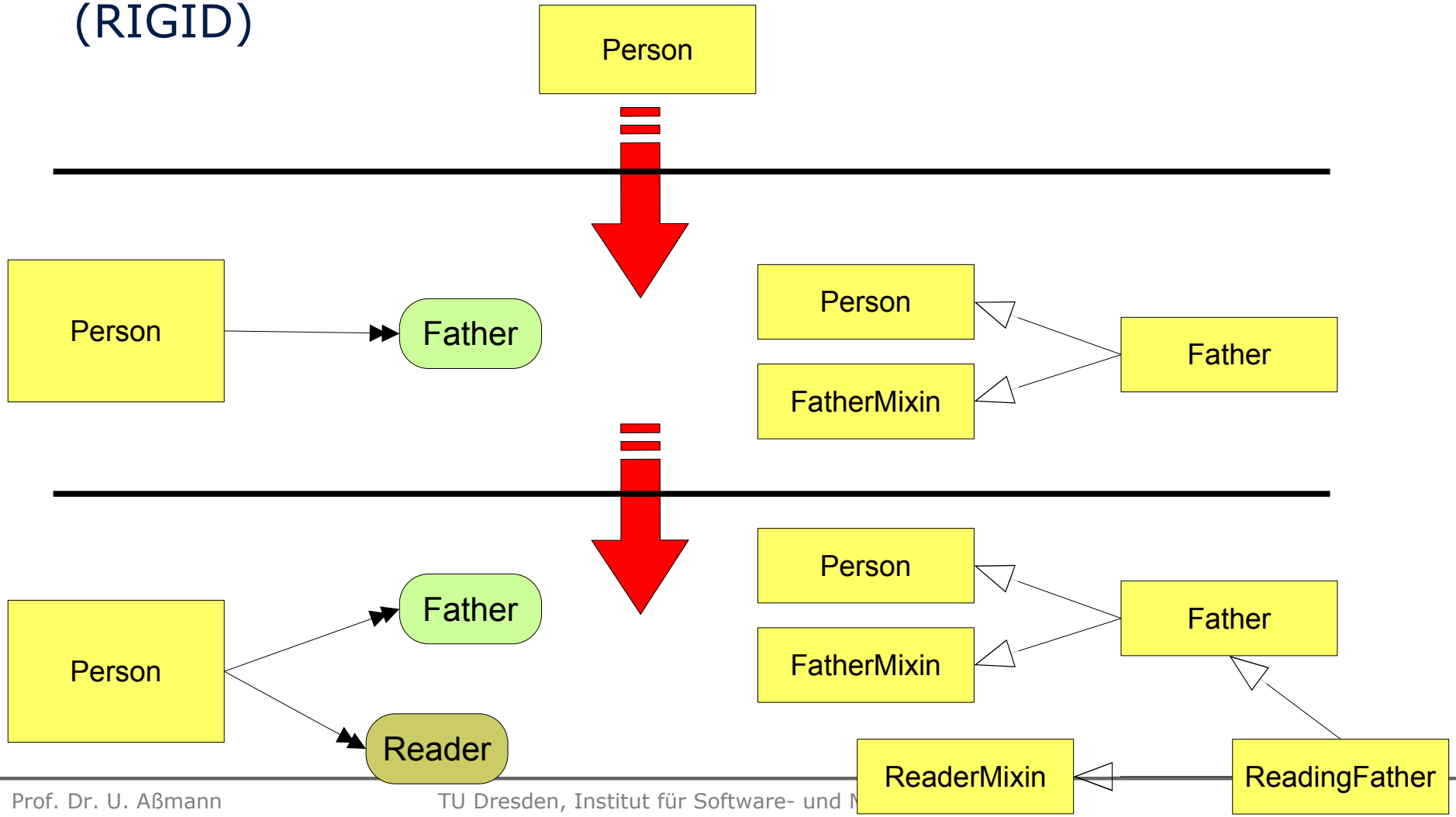
Identity is Fixed to Core Facet of Product Lattice

- Role type extensions does not change the name of the core type nor of the full type (polymorphism)



Role Extension Retains Core Identity

- Distinguish role types (NON-RIGID) from natural types (RIGID)



Modern Implementation of Connectors (Collaborations)

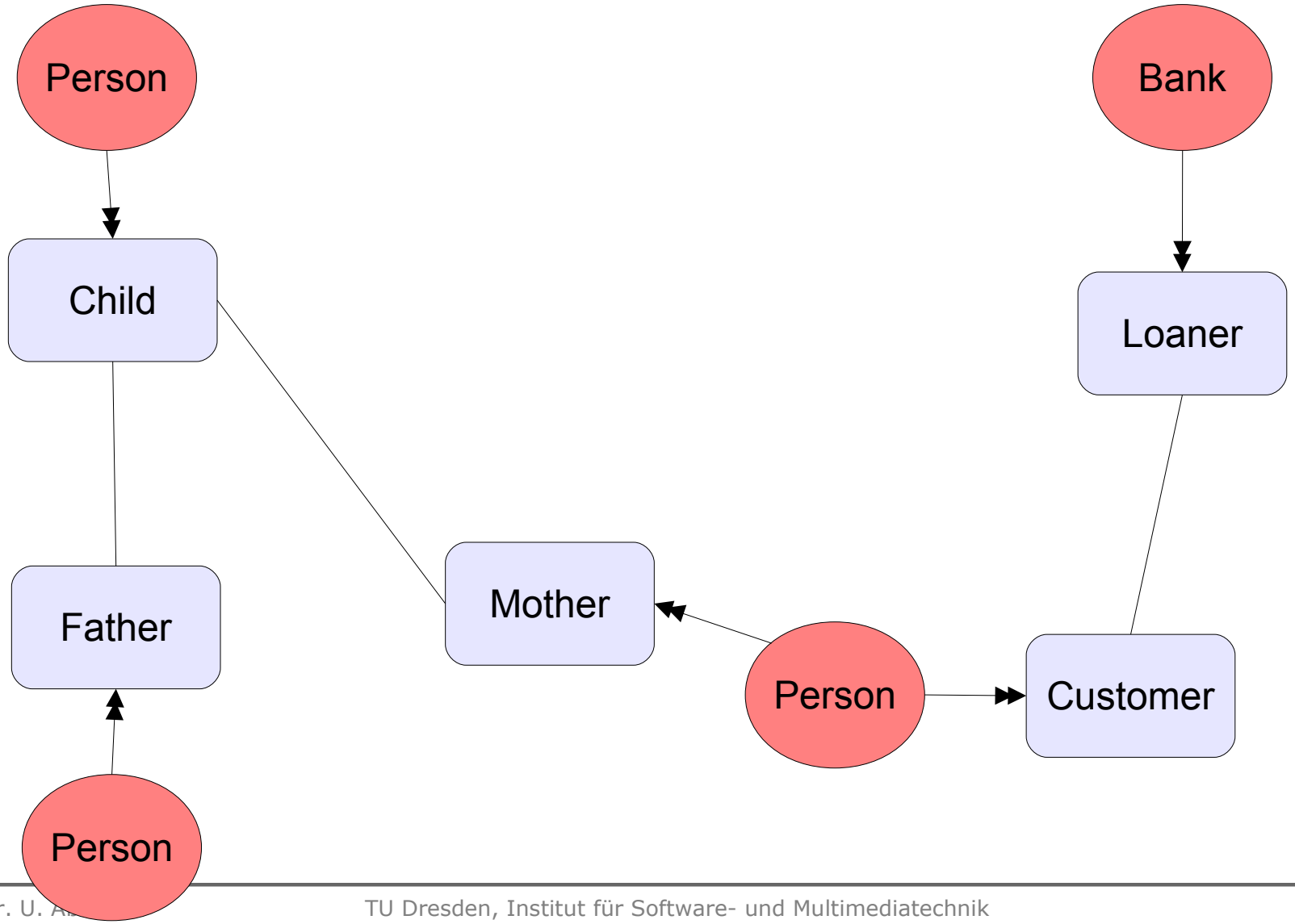
- Connectors (Collaborations, role models) are *type functors*
  - Functions on types
  - Model concerns and aspects
- Direct implementation
  - Mixin layers [Batory]
  - Role Object Pattern
  - Semantic macros
  - Generic templates (BETA, Invasive Composition Aßmann)
  - Aspects
  - Ceasar/J [M. Mezini, Darmstadt]
  - www.objectteams.org [S. Herrmann, Berlin]
- Rewriting to standard languages
  - Mapping (MDD process), e.g., with graph rewriting systems

Advantage 2) Scalable Refinement and Variability: Anchoring Roles into the Model-Driven Software Development (MDSD)

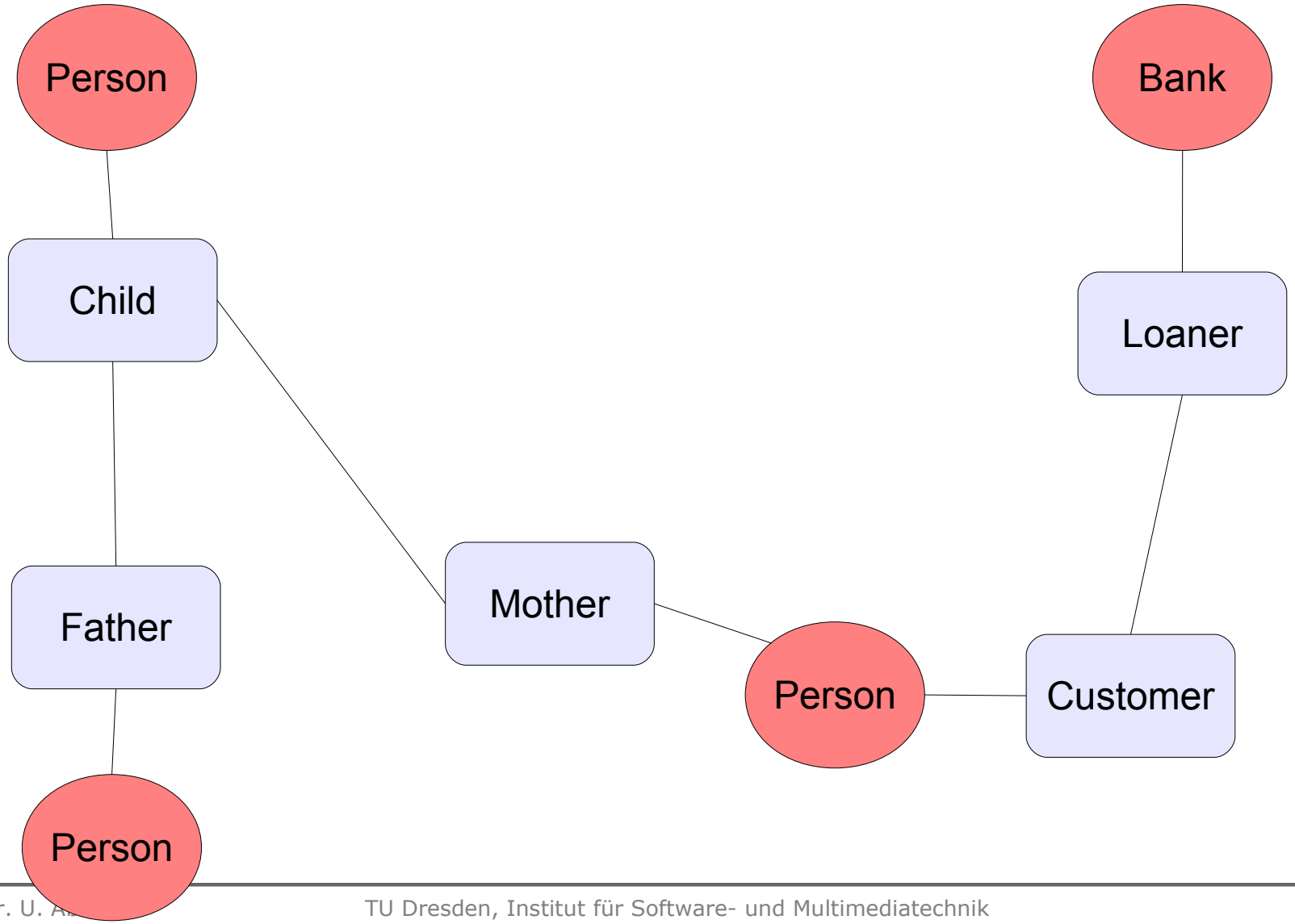
Variability by Role Embedding

- Mapping Roles to Implementation-Records is an *Embedding Decision*
- The plays-a relationship leaves the embedding open, defers the decision to implementation

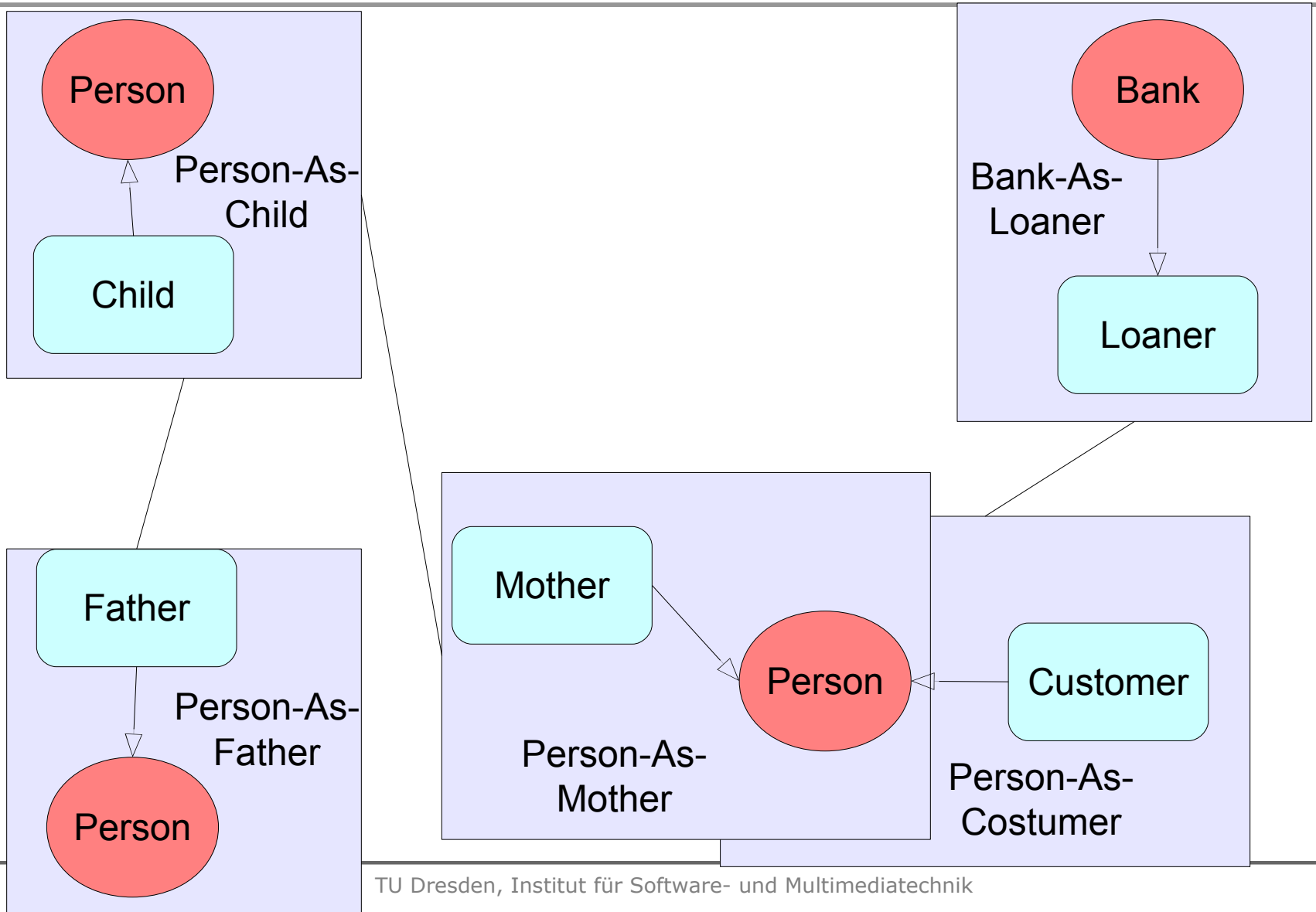
Families and Banks (in a role model)



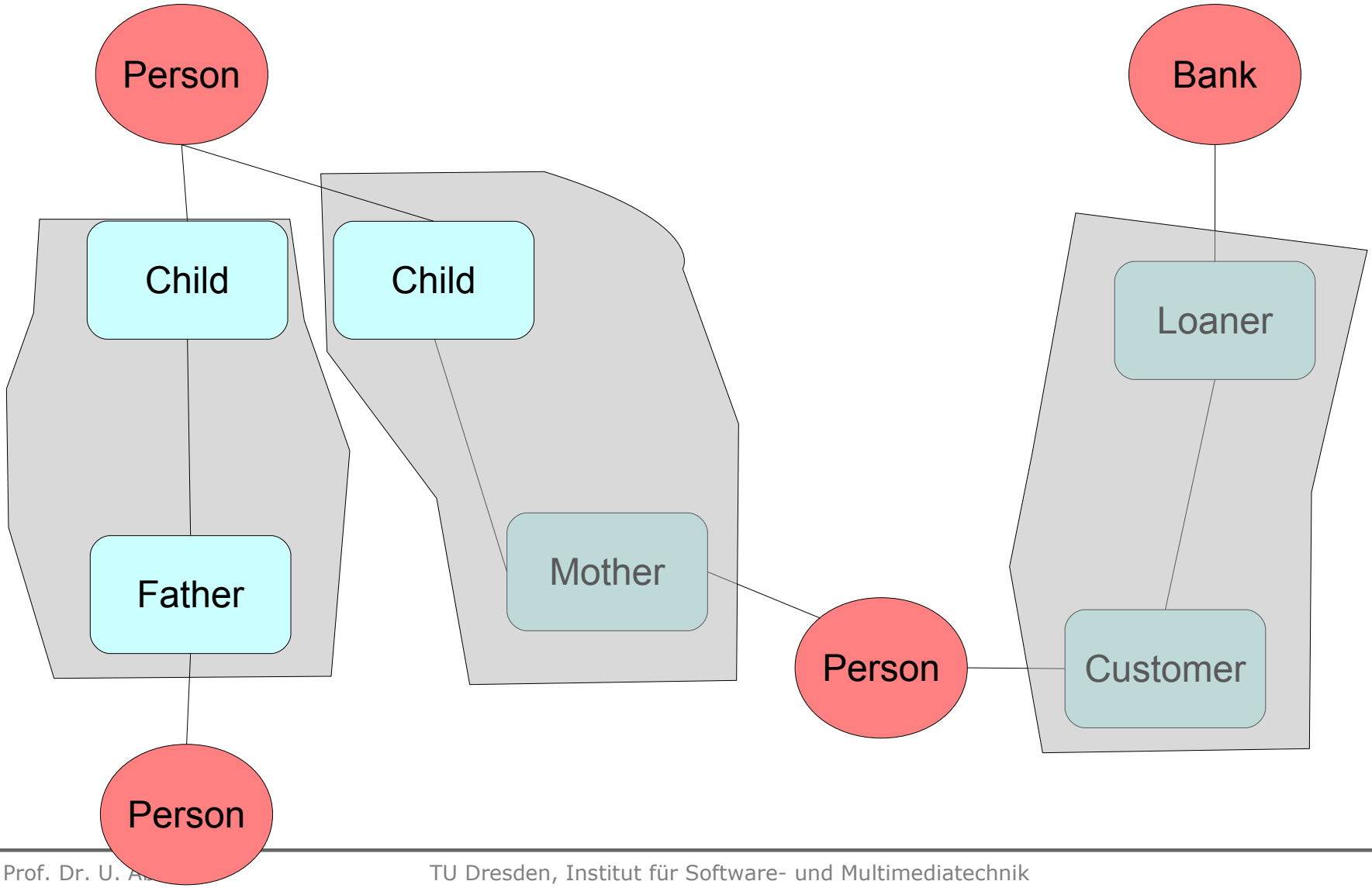
Families and Banks (Delegation, Split Design)



Families and Banks (Roles Embedded in Players)

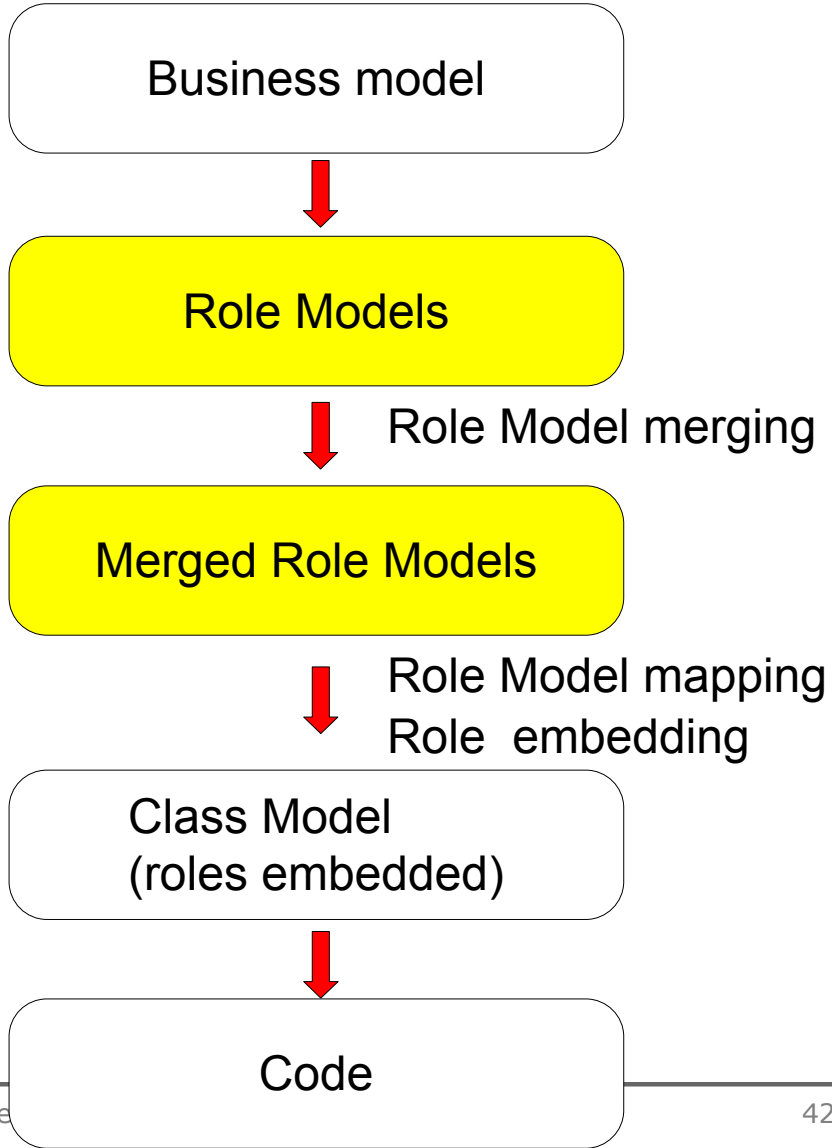


Families and Banks (Roles embedded in Relations)



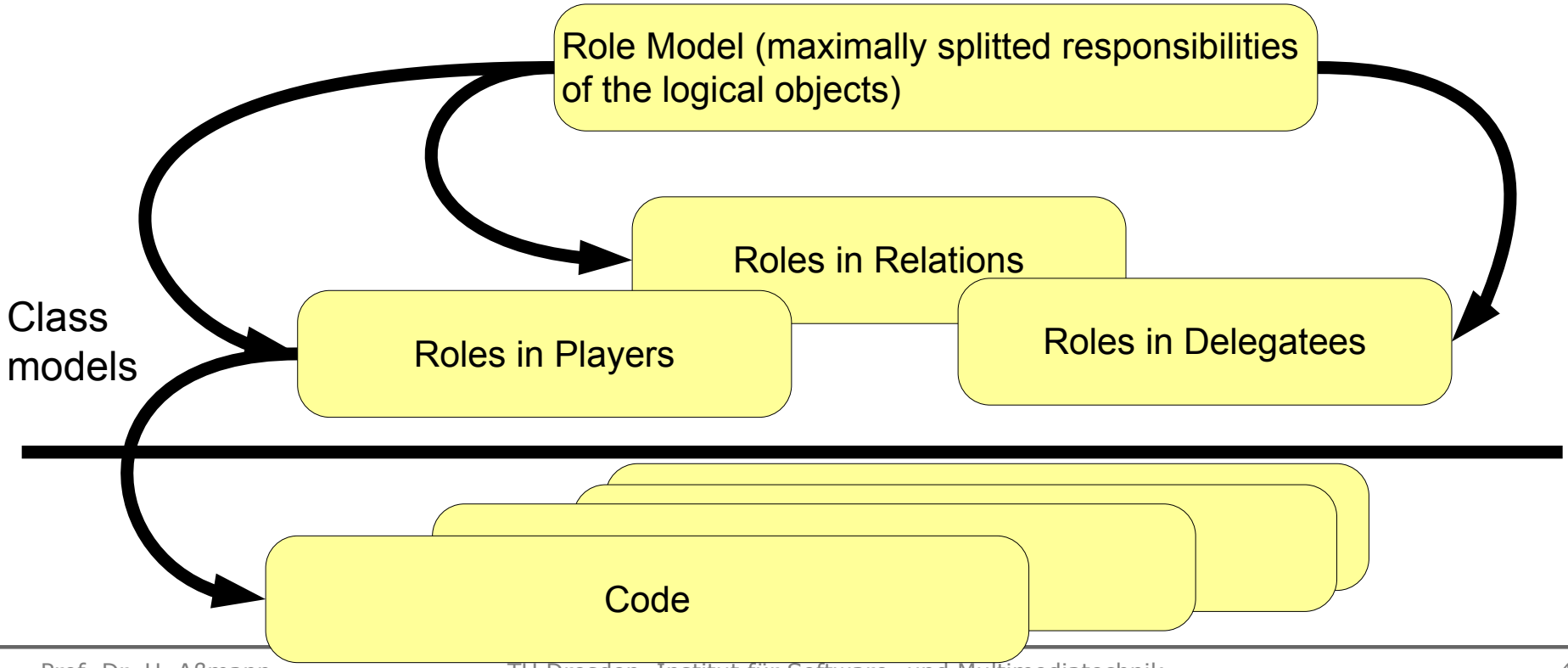
The Role Mapping Process and Model-Driven Architecture

- The question "*Where is a role embedded?*" is a *platform decision*
  - A role model is more *platform independent* than a class model
  - In MDA, role models are found on a more platform independent level than class models
- Role embedding is a task in MDA



Computing Physical Representation from Complex Objects

- The role embedding
  - determines, which physical objects inherit from which roles
  - *computes* the physical objects from maximal splits of the complex objects

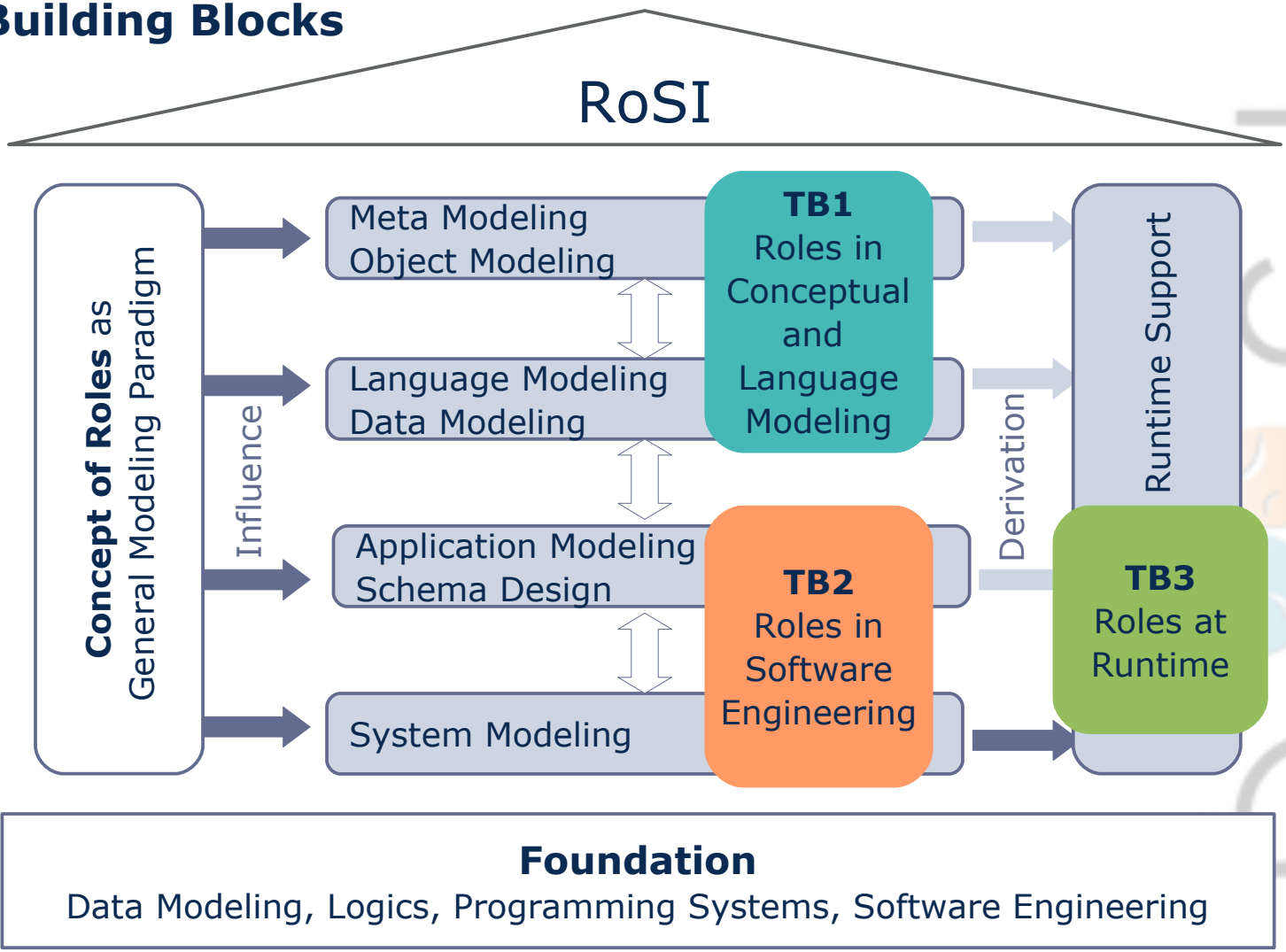


Role Embedding Yields Scalability

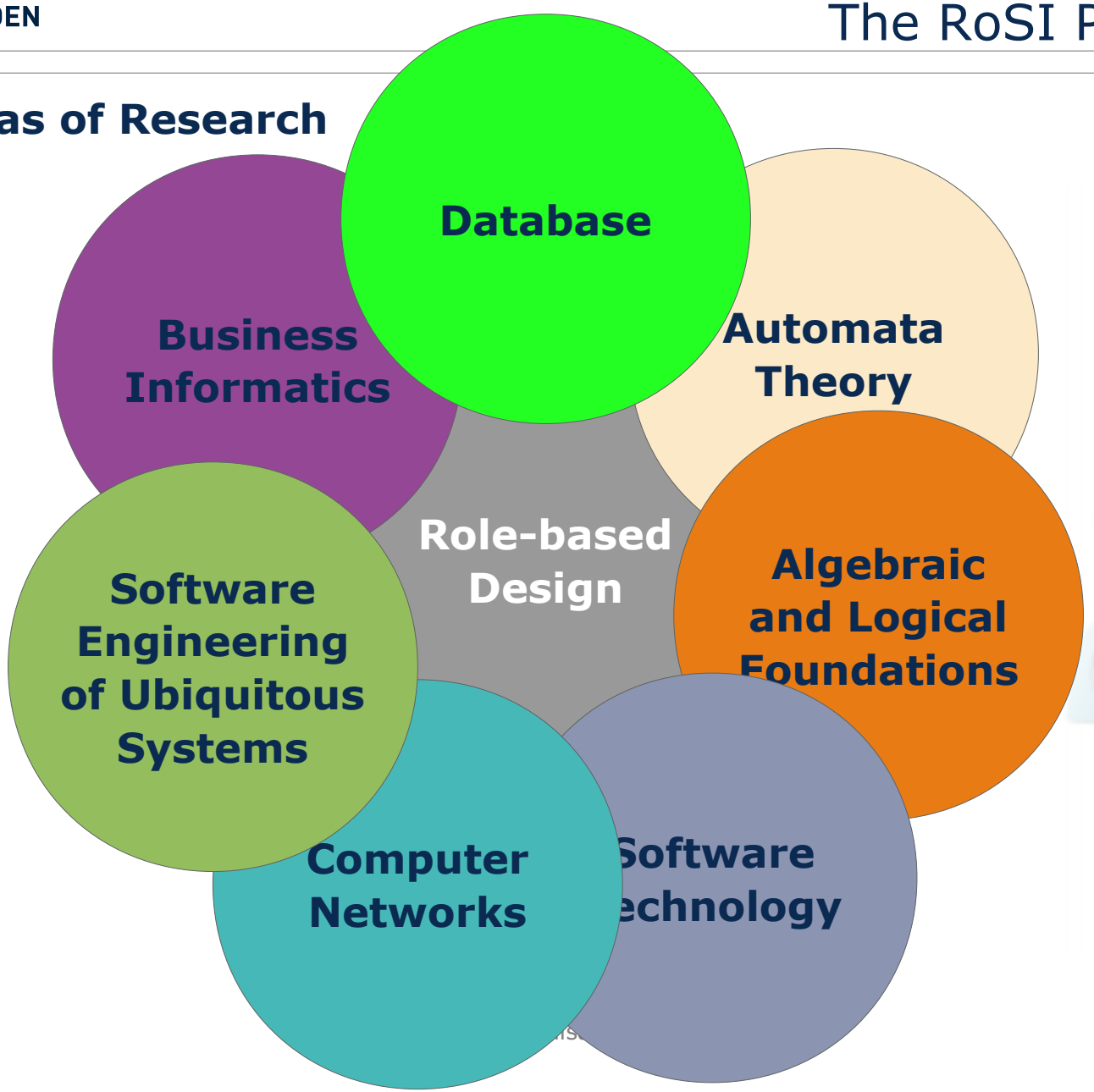
- Scalability means:
  - From one role design, derive many class designs
  - One can play with defering the embedding decision (compile-time, run-time)

Role embedding delivers variable implementations, scalable in splitting, locality and allocation

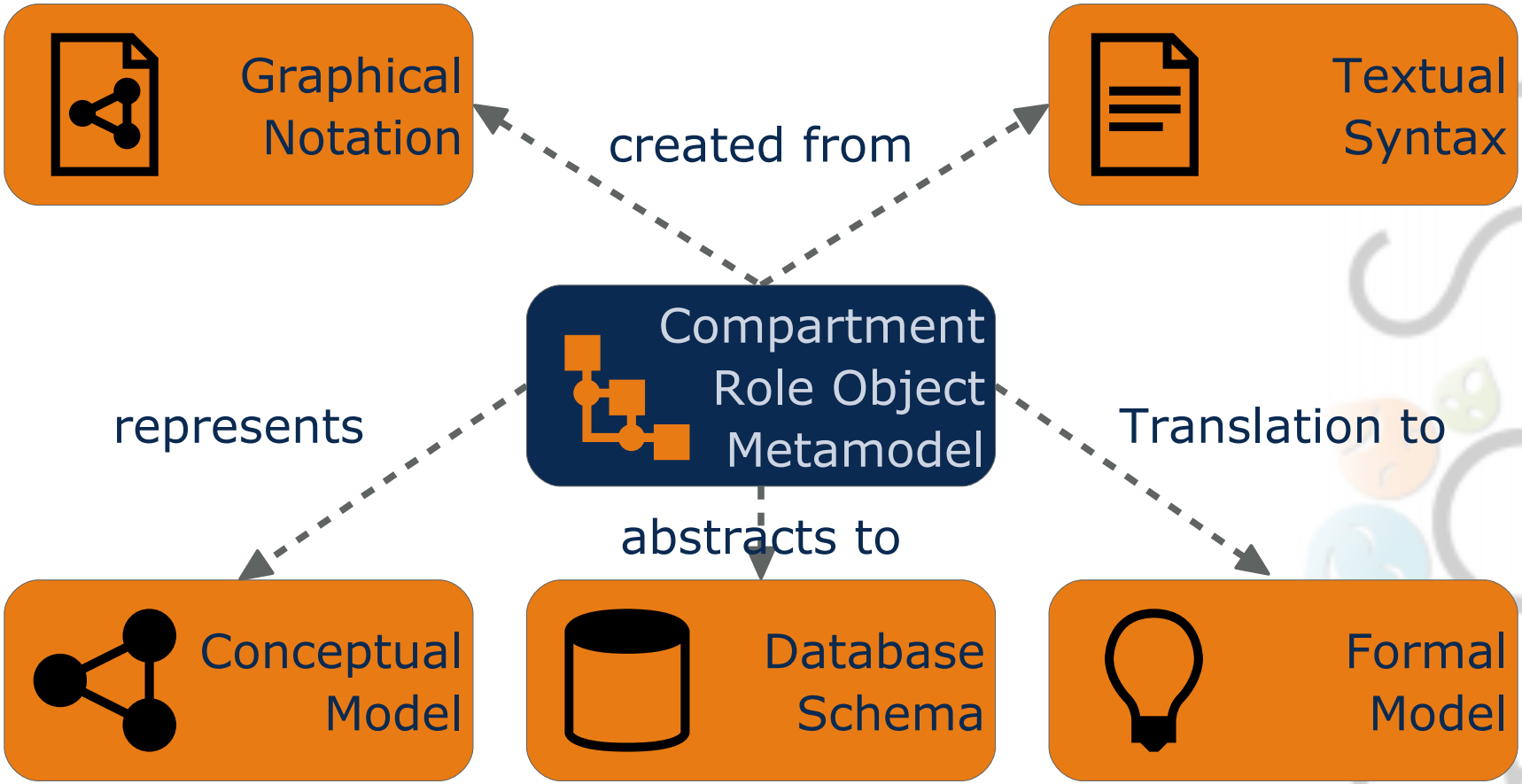
Building Blocks



Areas of Research

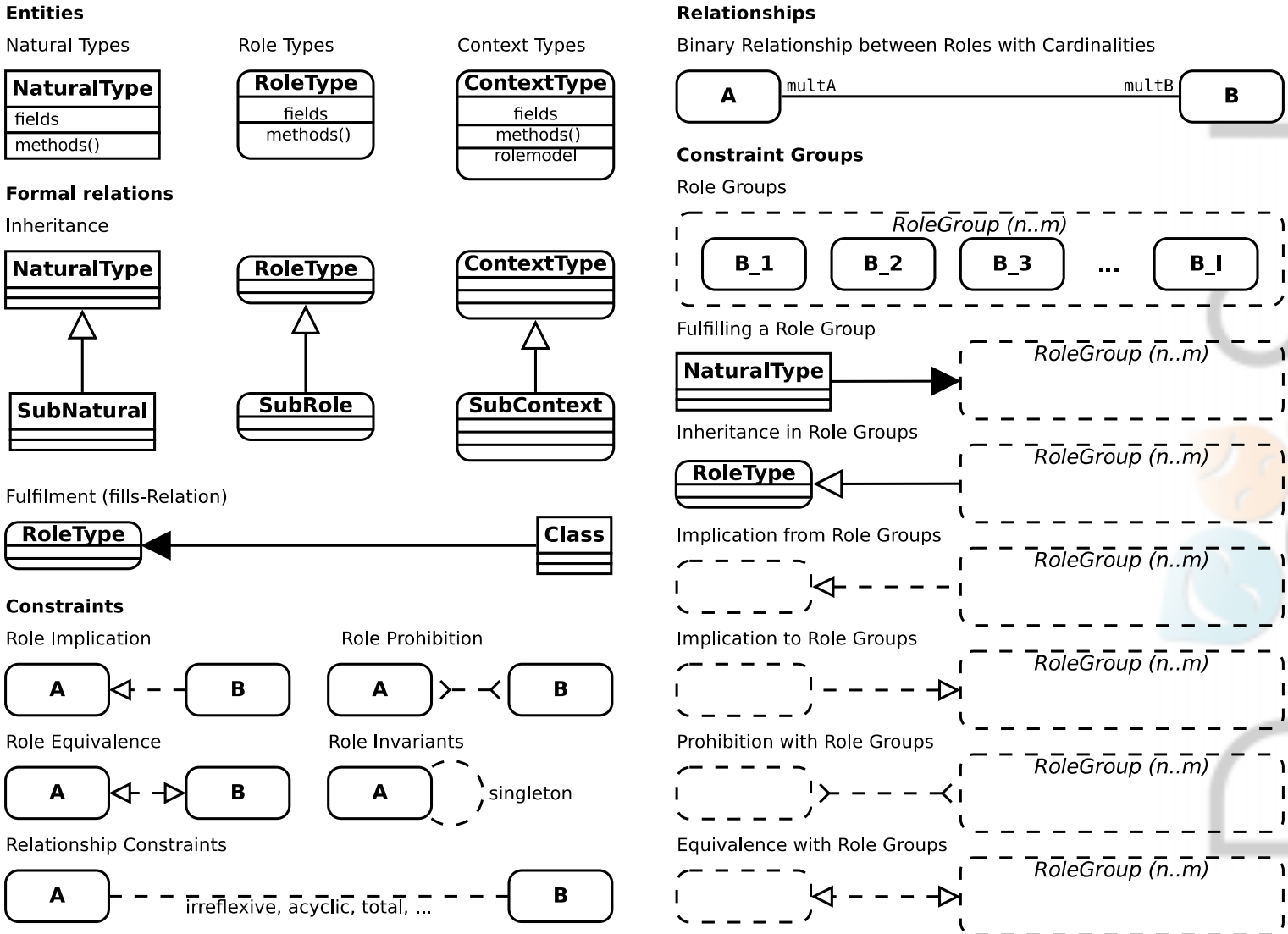


CROM - A Metamodel for RoSI

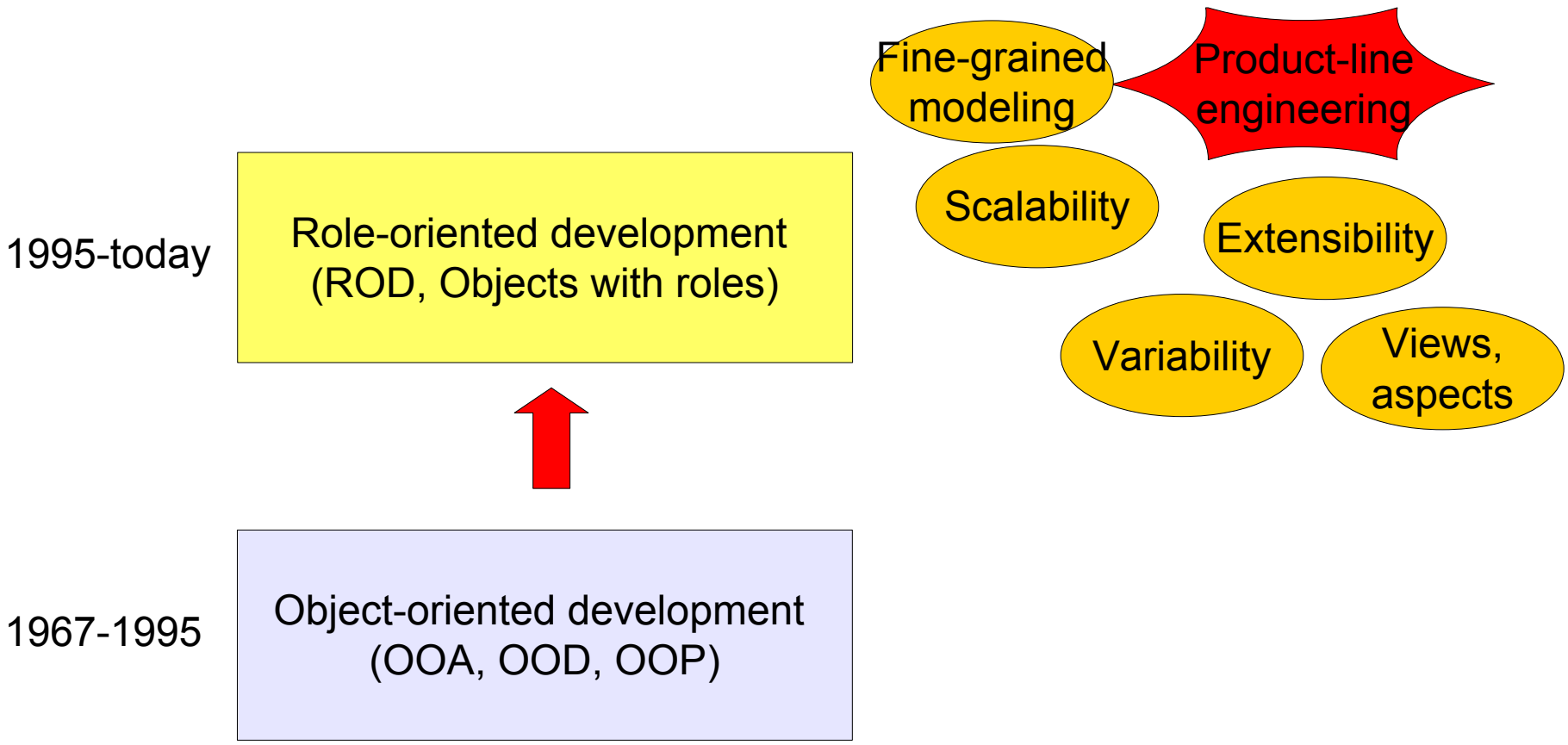


Models based on a common Metamodel

Compartment Role Object Model [Kühn]



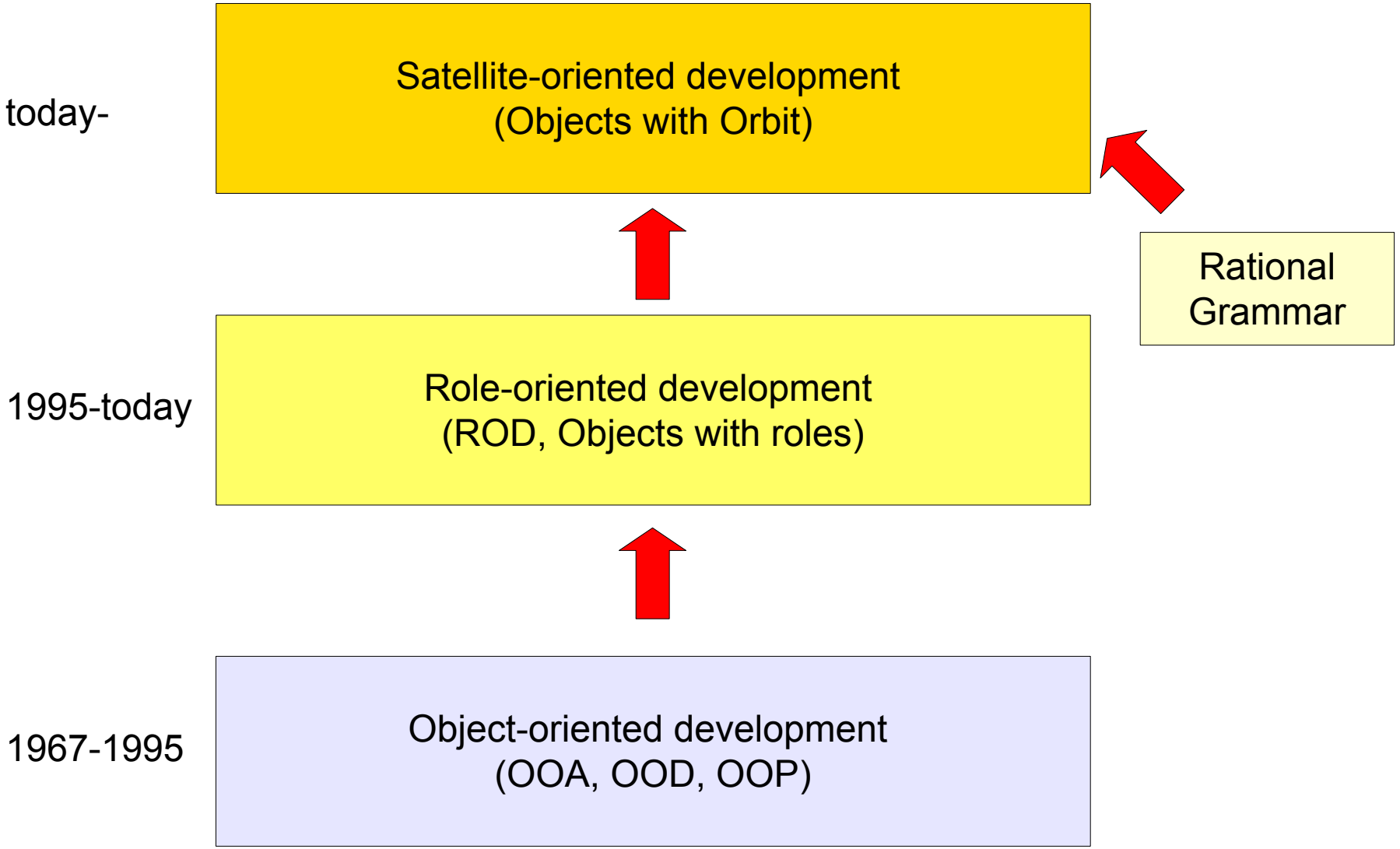
Ladder of Paradigms



Step 2
Beyond Role Modeling - Satellite-Based Modeling

(preliminary; unpublished; ongoing work)

Ladder of Paradigms (2)



Satellites (Mixins) as Predicates about Objects

as contrast to natural types (standalone things)

Satellite Objects and Types

A satellite (mixin object) is a subobject representing a predicate about a core object

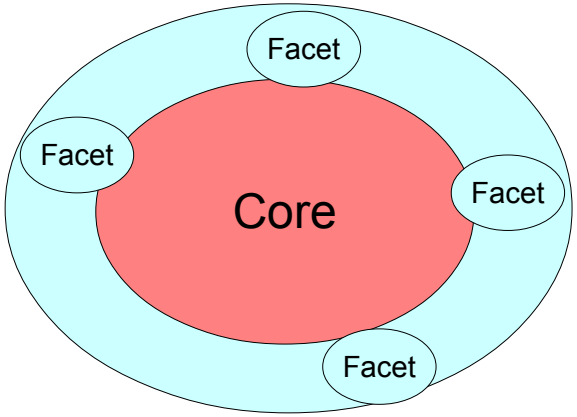
- Role (founded, anti-rigid)
- Facet (non-founded, rigid): classification value
- Phase (non-founded, anti-rigid)
- Integrated Part (non-founded)
- Other unary predicates

A satellite type (mixin type) is a type, dependent on a core type (sortal universal)

- Role type
- Facet type: classification dimension
- Phase type
- Integrated Part type

What are Facets?

- Classification dimensions
  - that belong intrisically to the object
  - non-founded
  - rigid (non-temporary)
- Library science [Ramagathan]

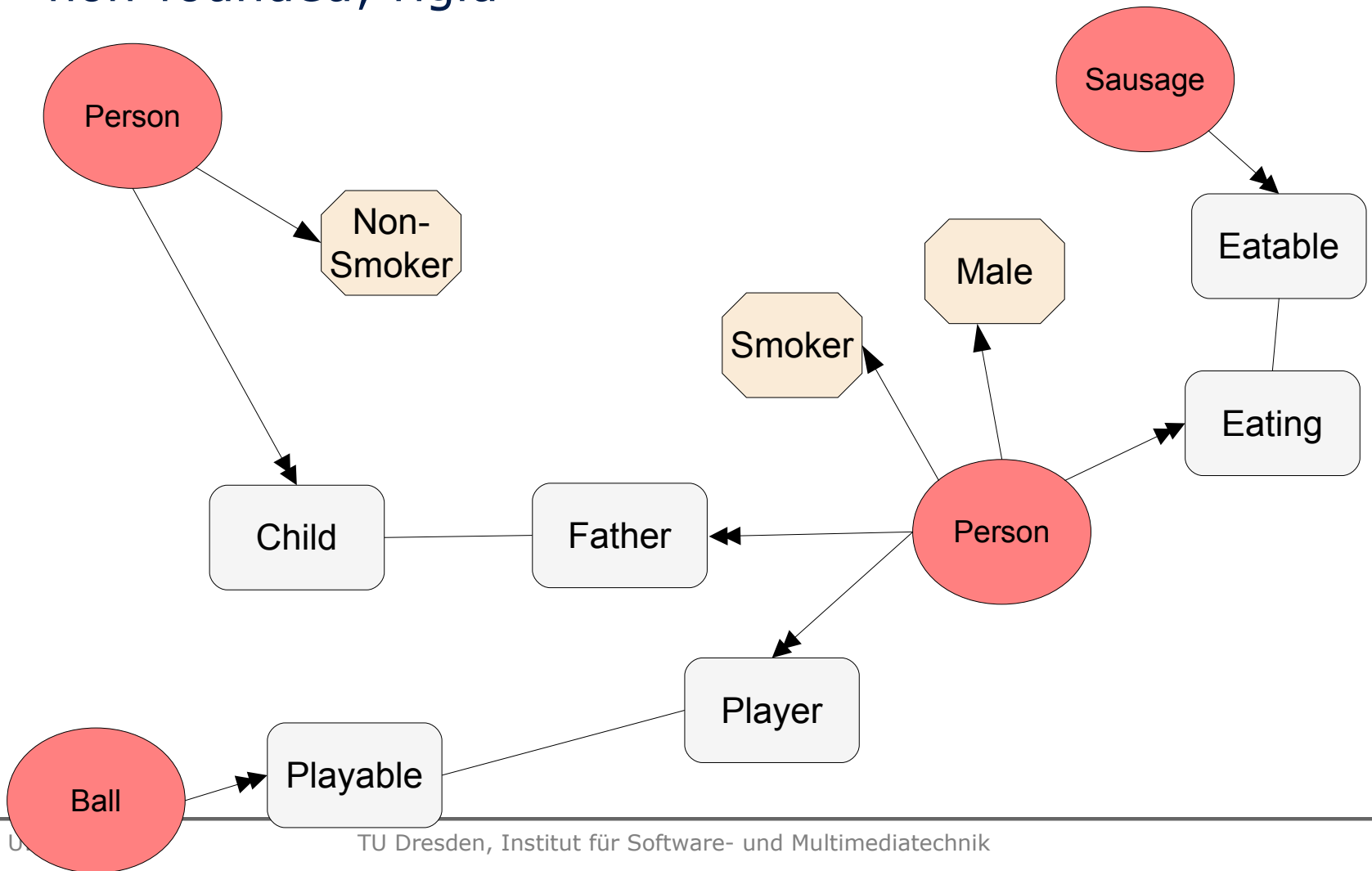


FIND07 -- International Workshop on
Dynamic Taxonomies and Faceted Search
Regensburg, Germany, September 3-7, 2007
in conjunction with DEXA 2007

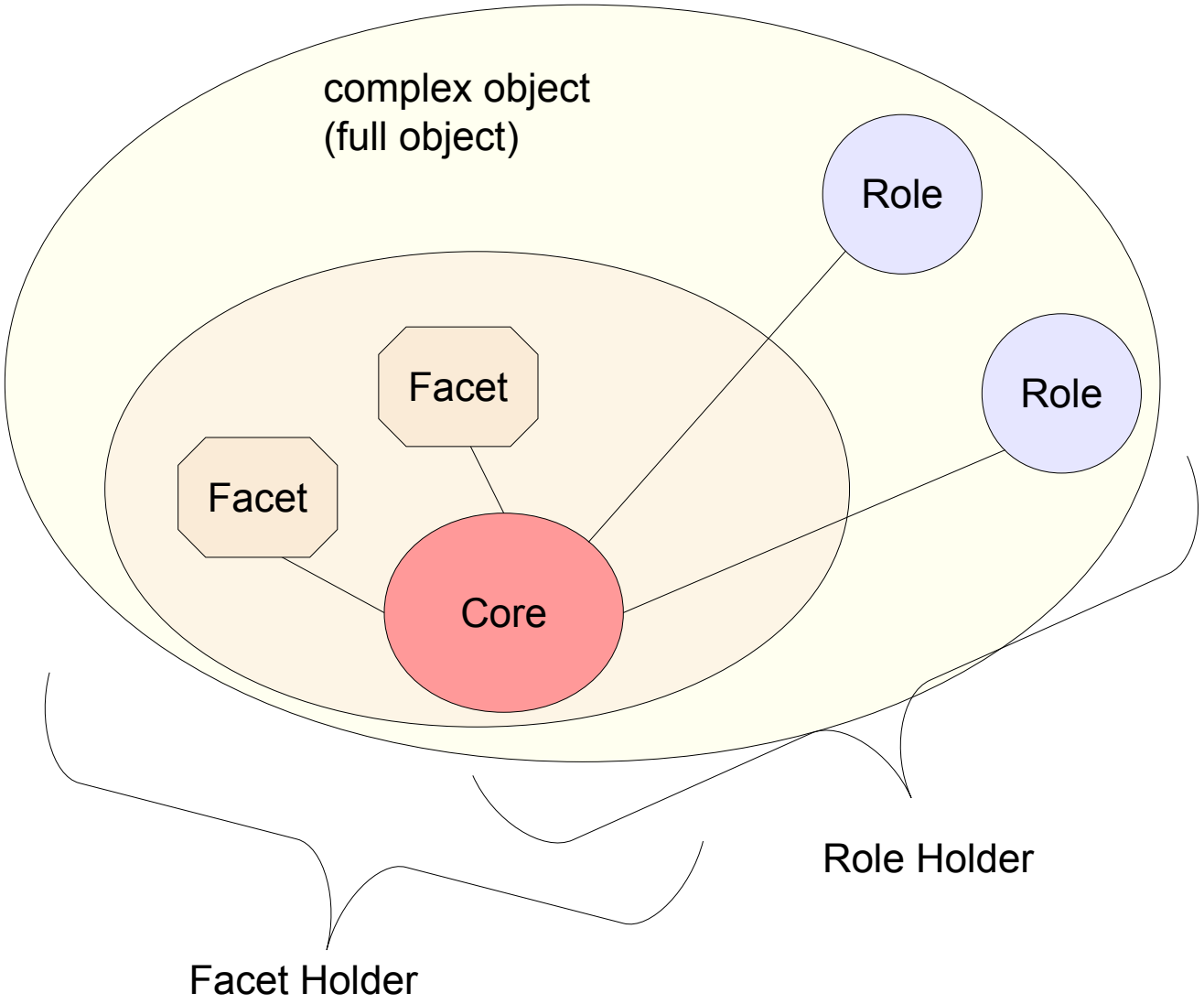
Dynamic taxonomies work on multidimensional taxonomies (usually organized by facets) and provide a single, coherent visual framework in which users can focus on one or more concepts in the taxonomy, and immediately see a conceptual summary of their focus, in the form of a reduced taxonomy derived from the original one by pruning unrelated concepts. Concepts in the reduced taxonomy can be used to set additional, dependent foci and users iterate in a guided yet unconstrained way until they reach a result set sufficiently small for manual inspection.

What are Facets?

- non-founded; rigid

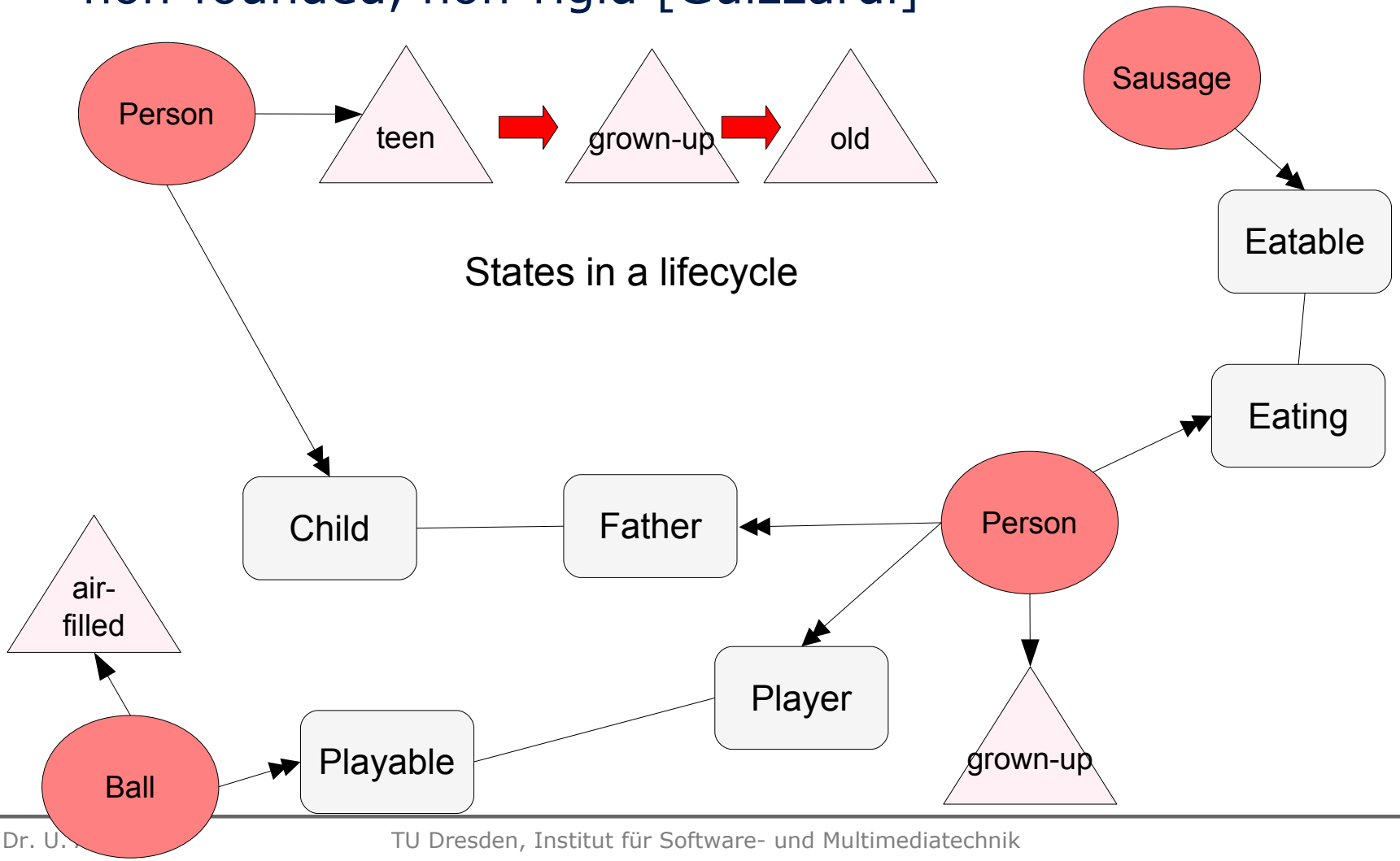


Satellite Holders (Complex Objects)

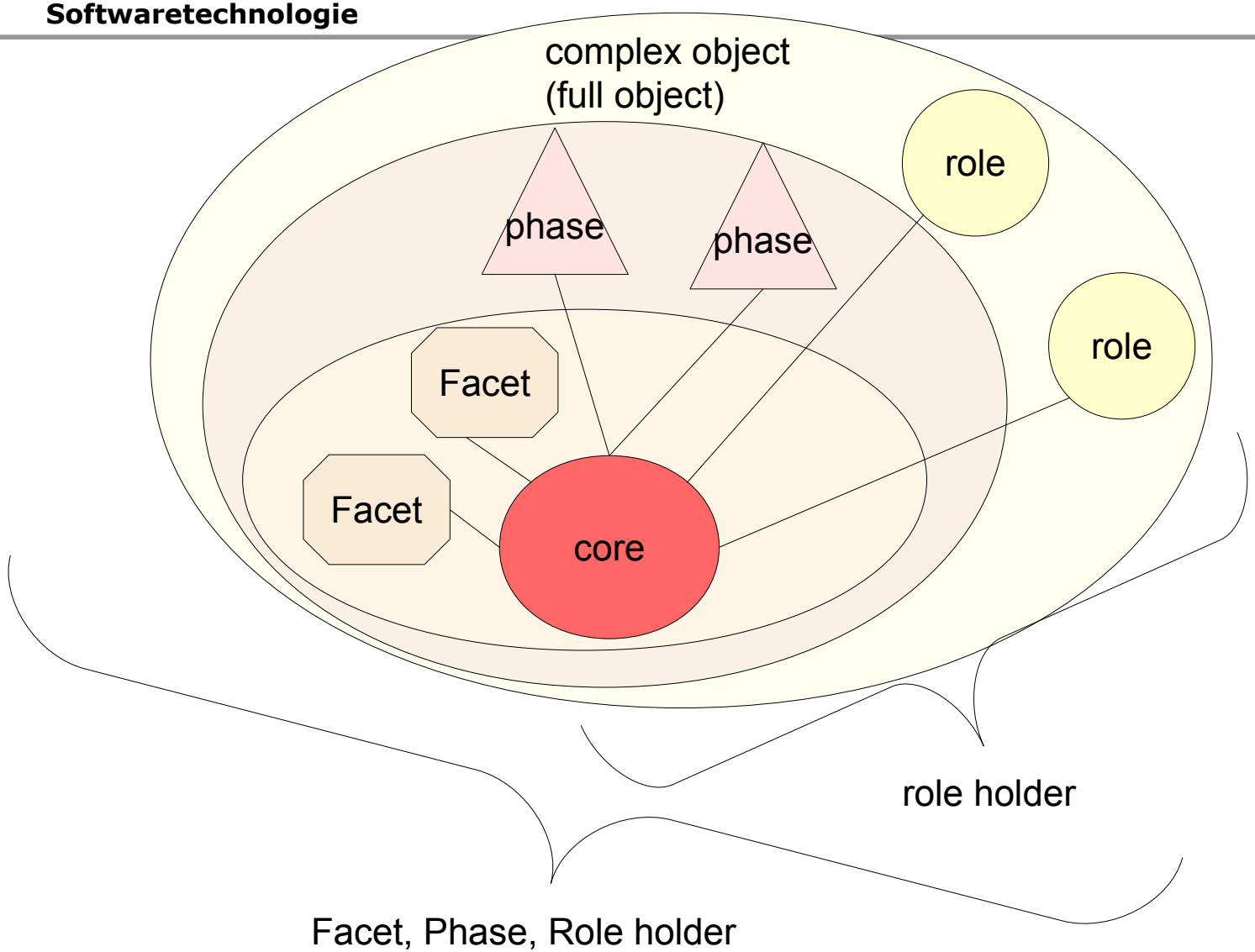


What are Phases?

- non-founded, non-rigid [Guizzardi]

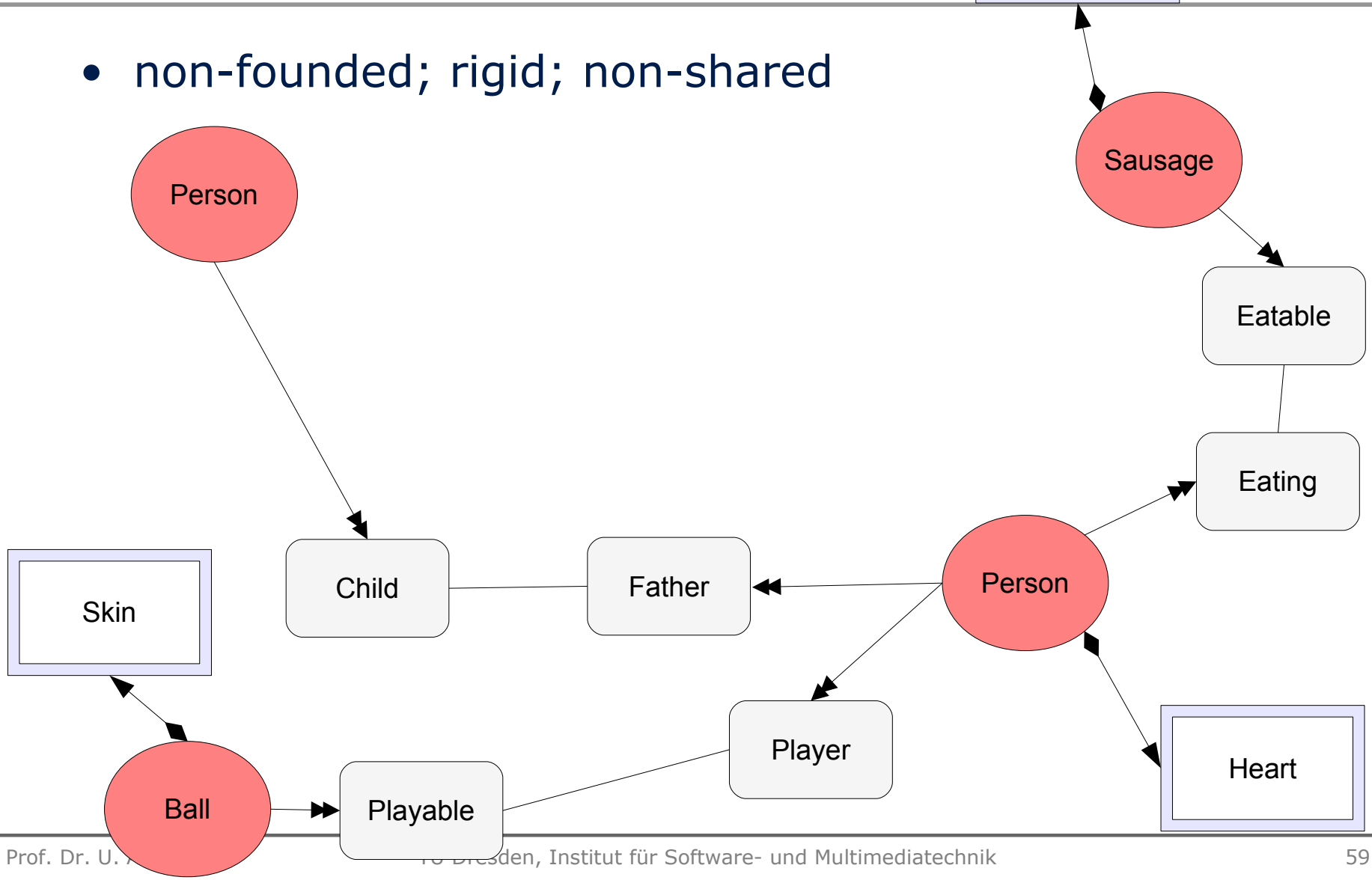


Satellite Holder



What are Integral Parts?

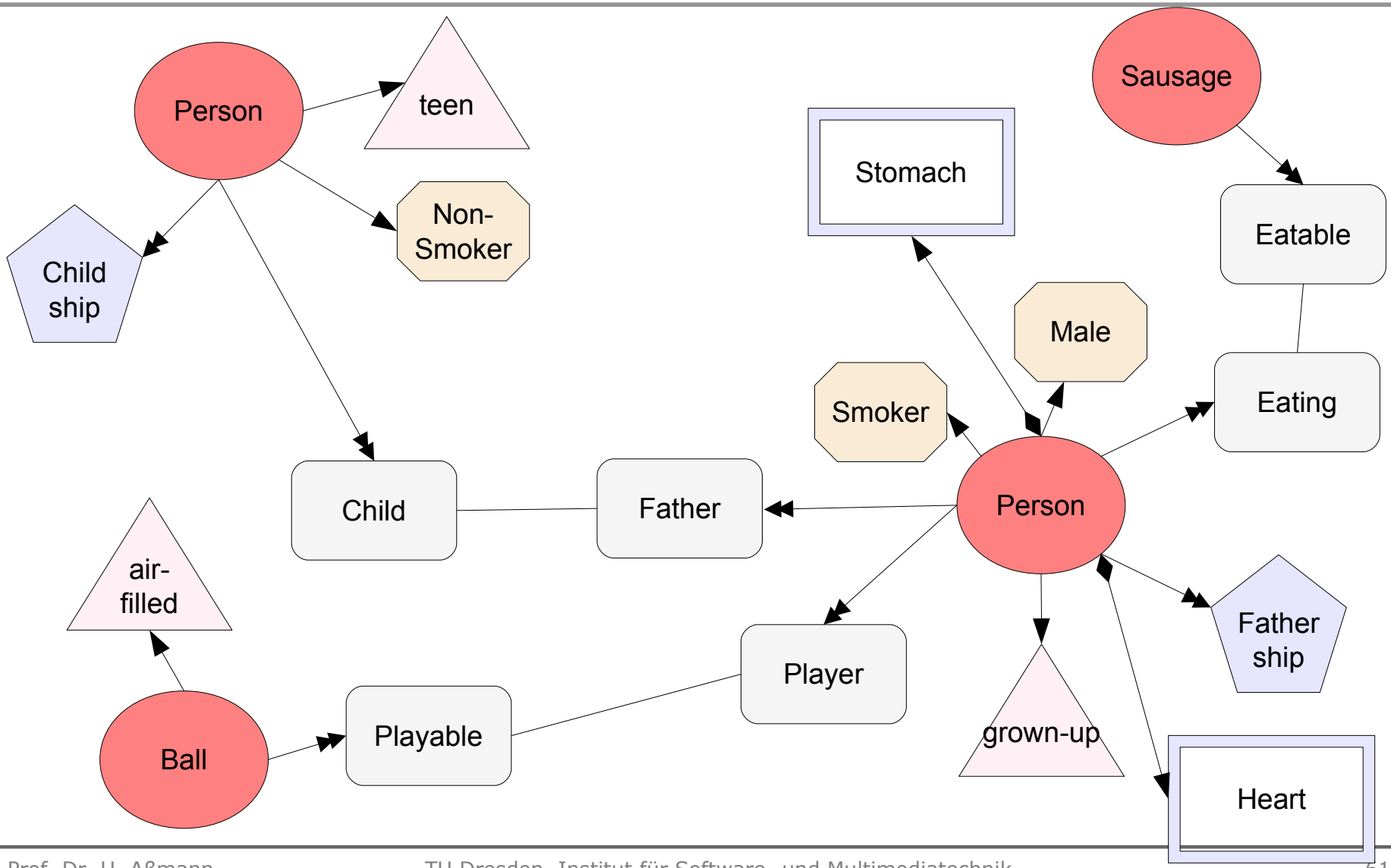
- non-founded; rigid; non-shared



Satellite-Core Relationships (Elementary Copula)

- A *satellite relationship* is a "primary key" relationship between a core and a satellite,
- .. an elementary copula
  - thing plays-a role
  - thing has-a phase
  - thing has-a-classification-dimension
  - thing owns-an-integral part
  - thing is unary-predicator

Core Objects with Different Satellites (Joined by Elementary Triples)

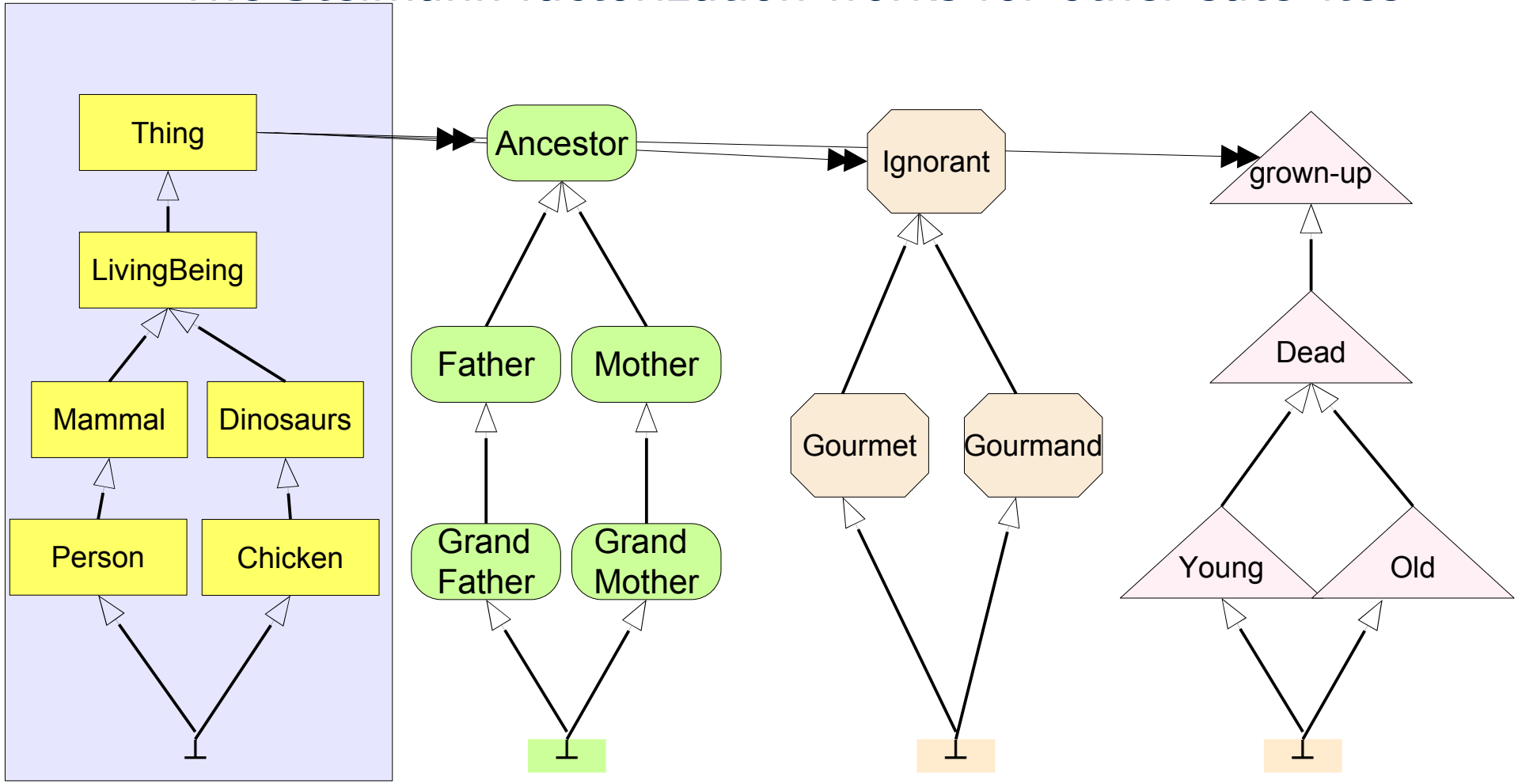


Distinguish beyond roles
facets, phases, integral parts

Refine the Steimann factorization
with more satellite types

The Satellite Factorization

- The Steimann factorization works for other satellites

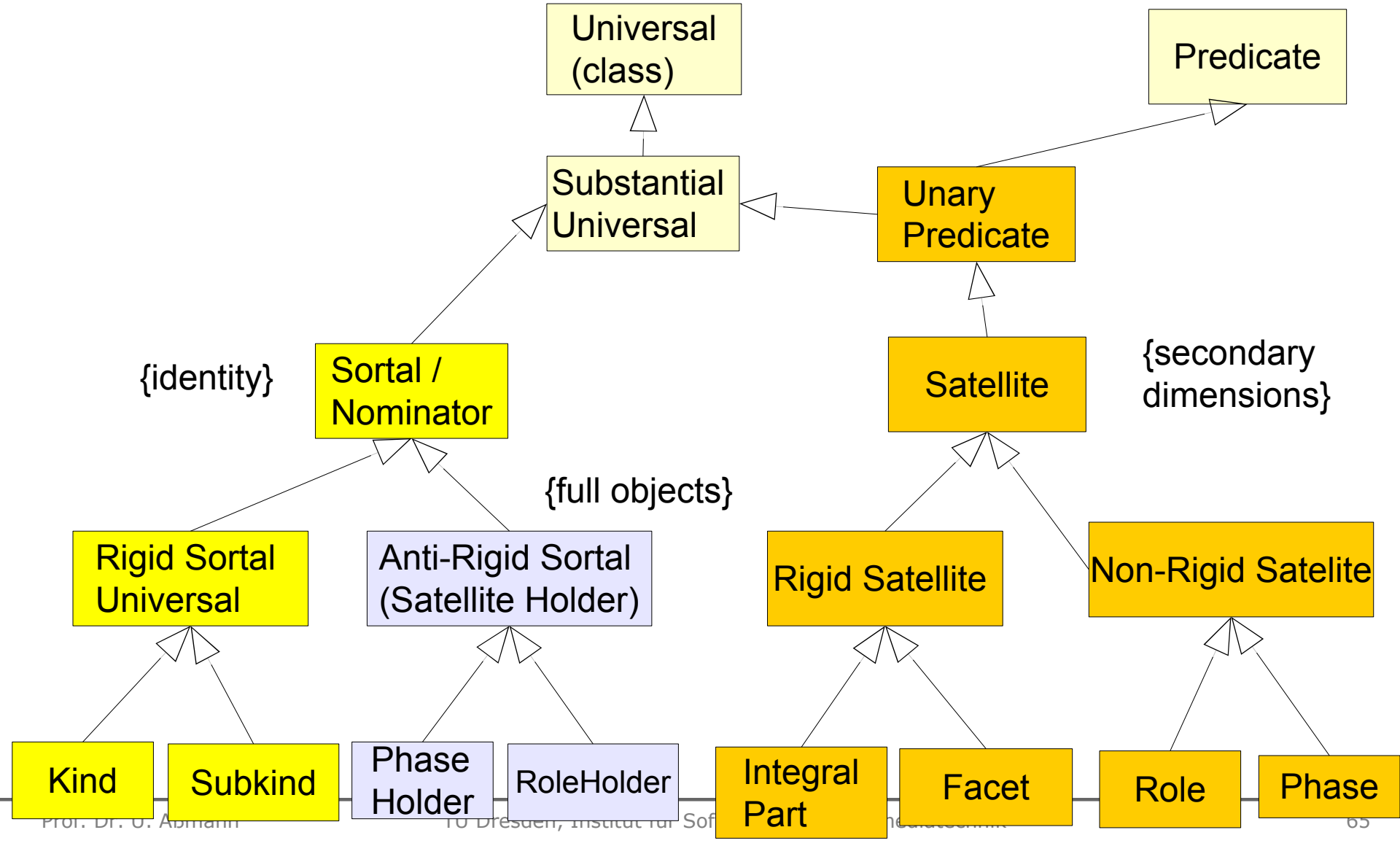


The Satellite Factorization

Consists of an extended Steimann Factorization

Divide a *type* into a *tuple type* over a product lattice of
a core dimension and
f facet dimensions
p phase dimensions
r role dimensions
i integral part dimensions
(Core,
Facet_1,..,Facet_f,
Phase_1, .., Phase_p
Role_1, ..., Role_r,
IntegralPart_1,..,IntegralPart_i)

The Satellite Hierarchy

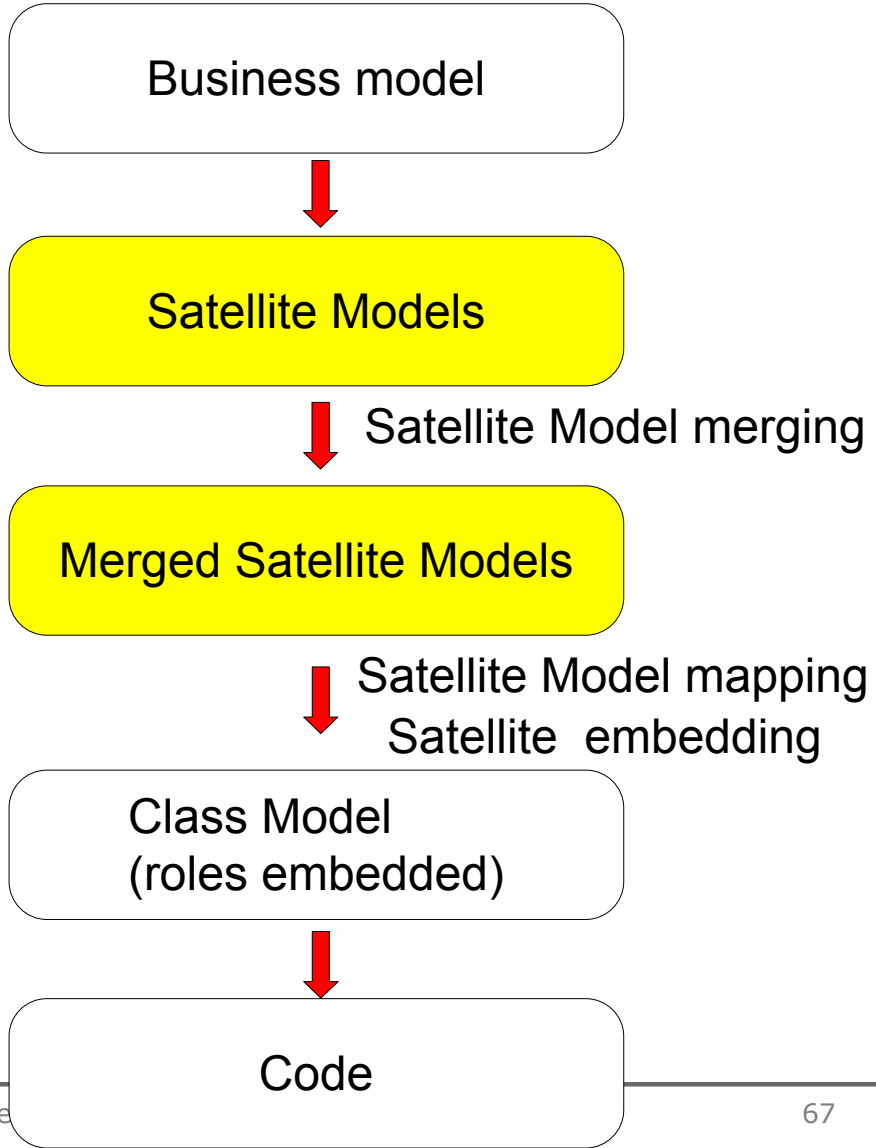


More Scalability with Satellites

- Goal: Defering the *satellite embedding decision* to implementation for *all* satellite relationships
- All advantages of role modeling, but even more
  - Variability
  - Extensible logical designs
  - Scalable embedding decisions

The Satellite-Mapping Process and Model-Driven Architecture

- The question "*Where is satellite embedded?*" is a *platform decision*
- Satellite embedding is a task in MDA

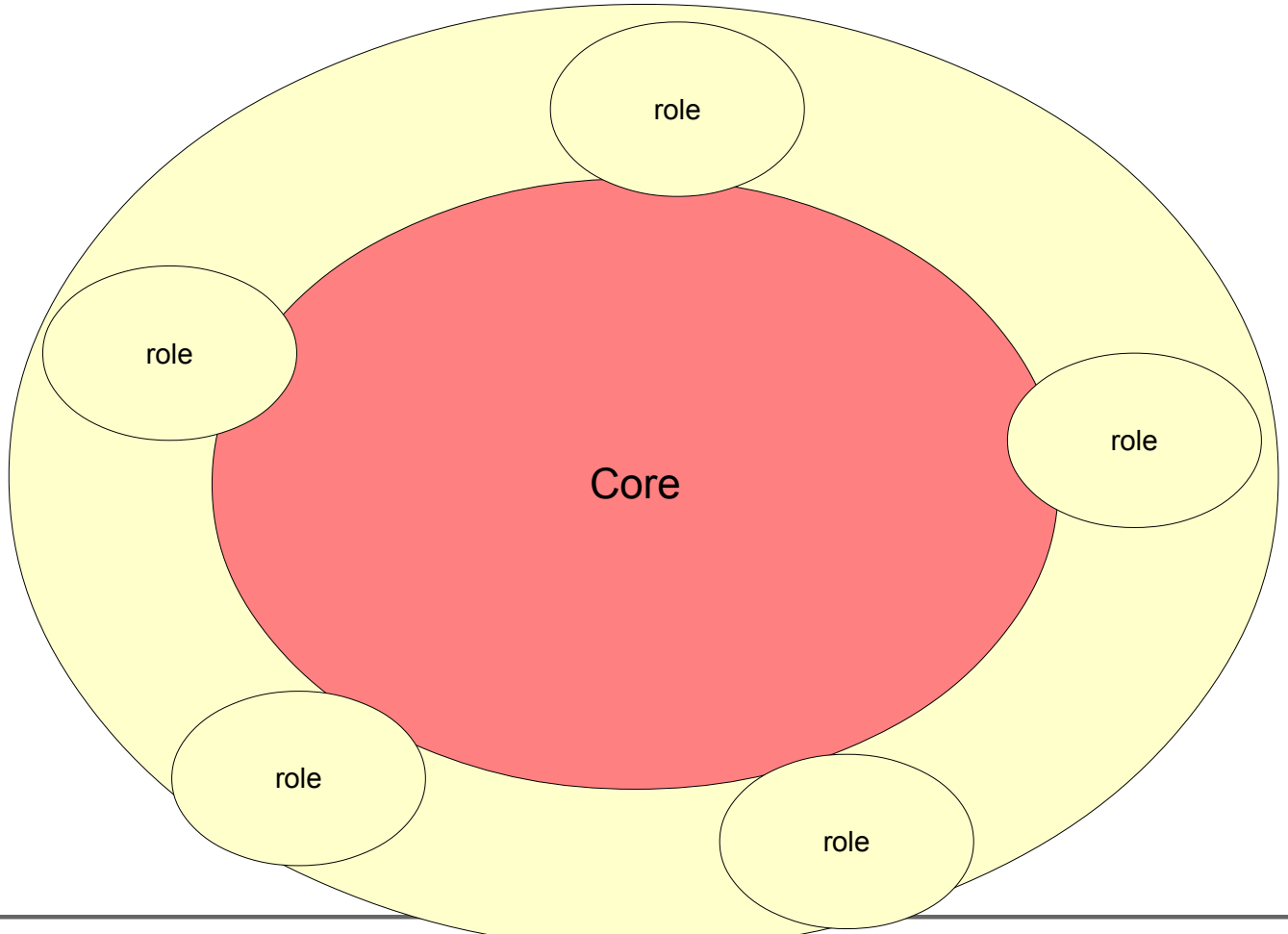


The Object-Satellite Model (The ORBIT Object Model)

(preliminary; unpublished; ongoing work)

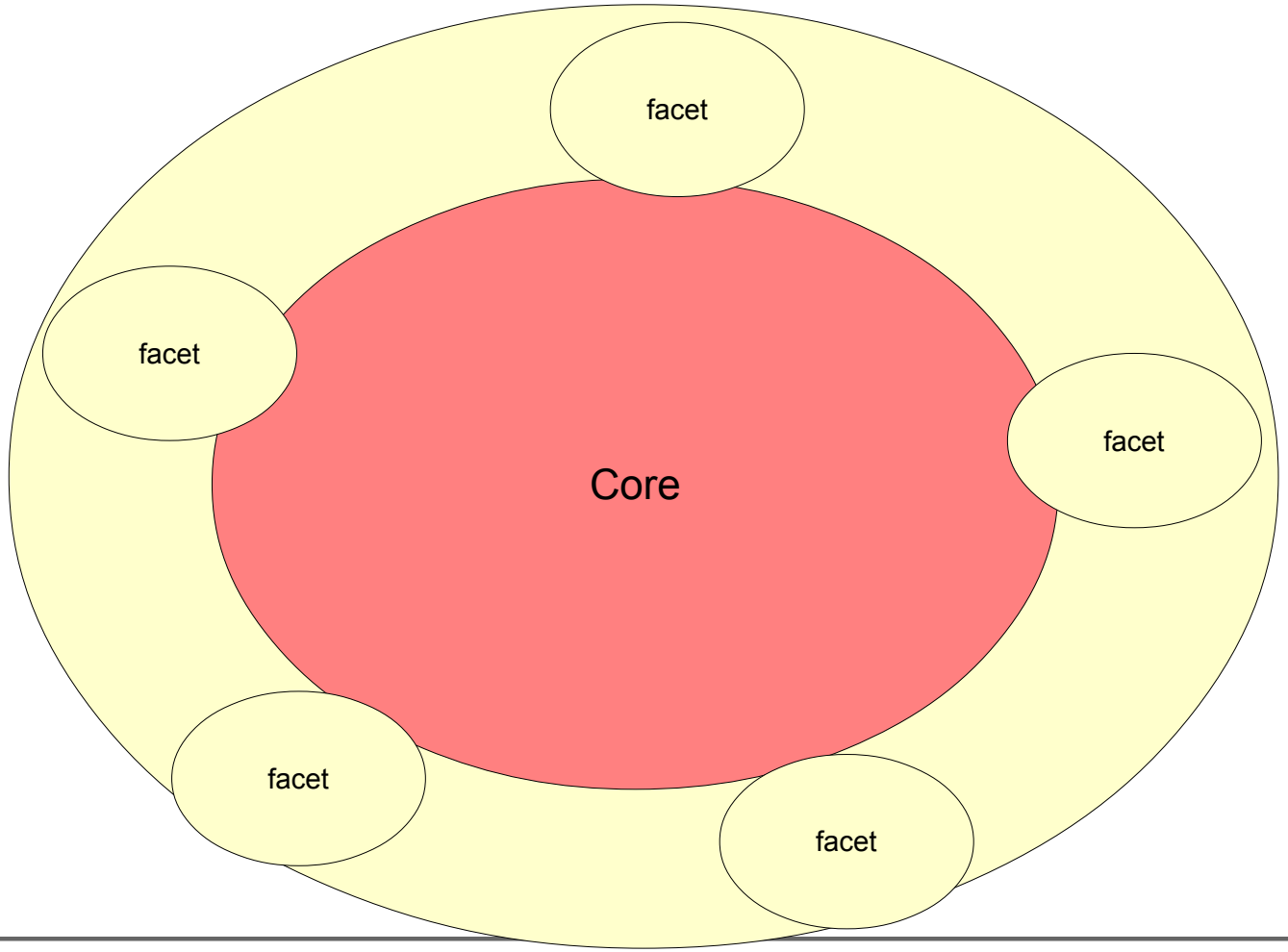
Core and Role Orbit

- Roles are classification dimensions, founded

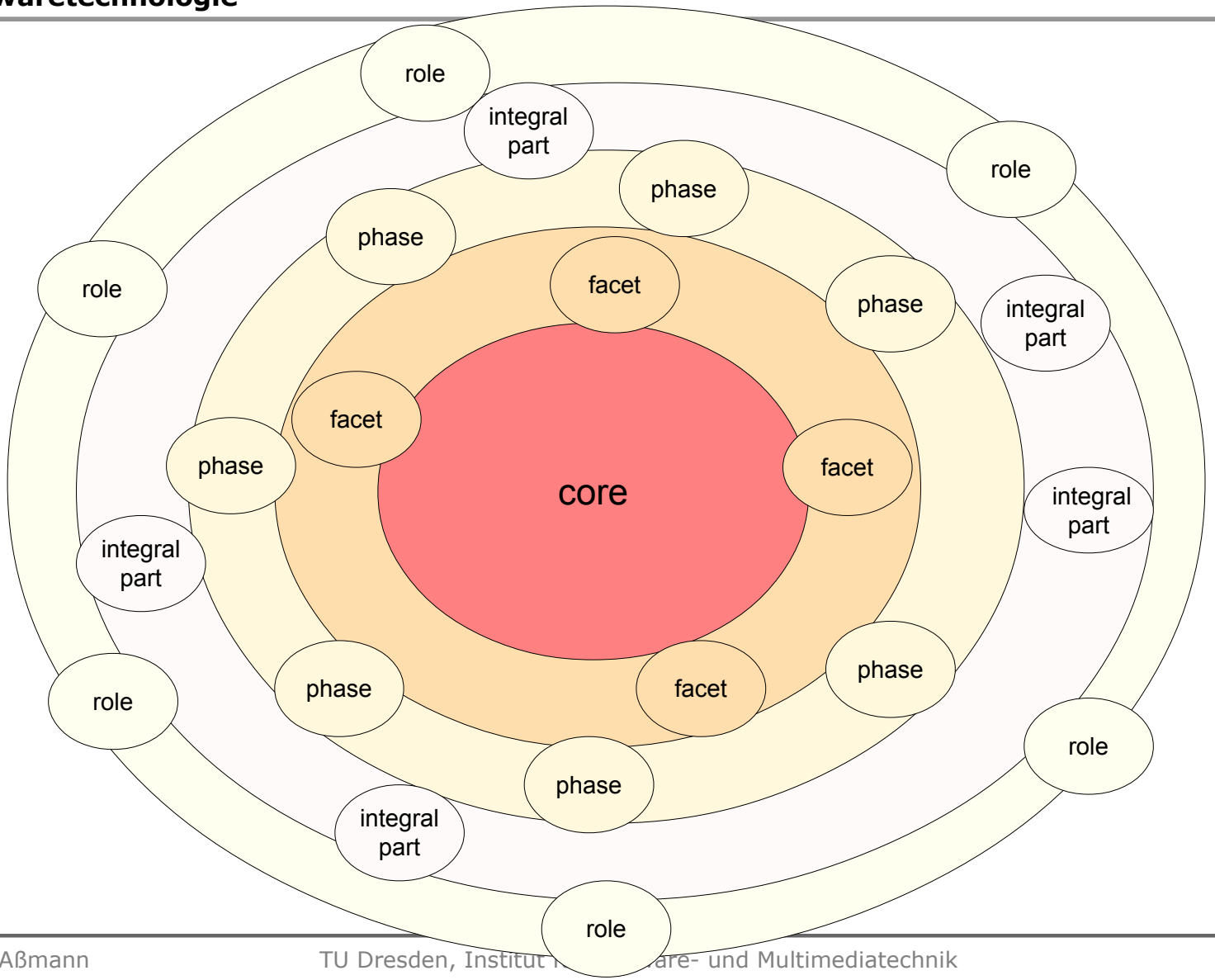


Core and Facet Orbit

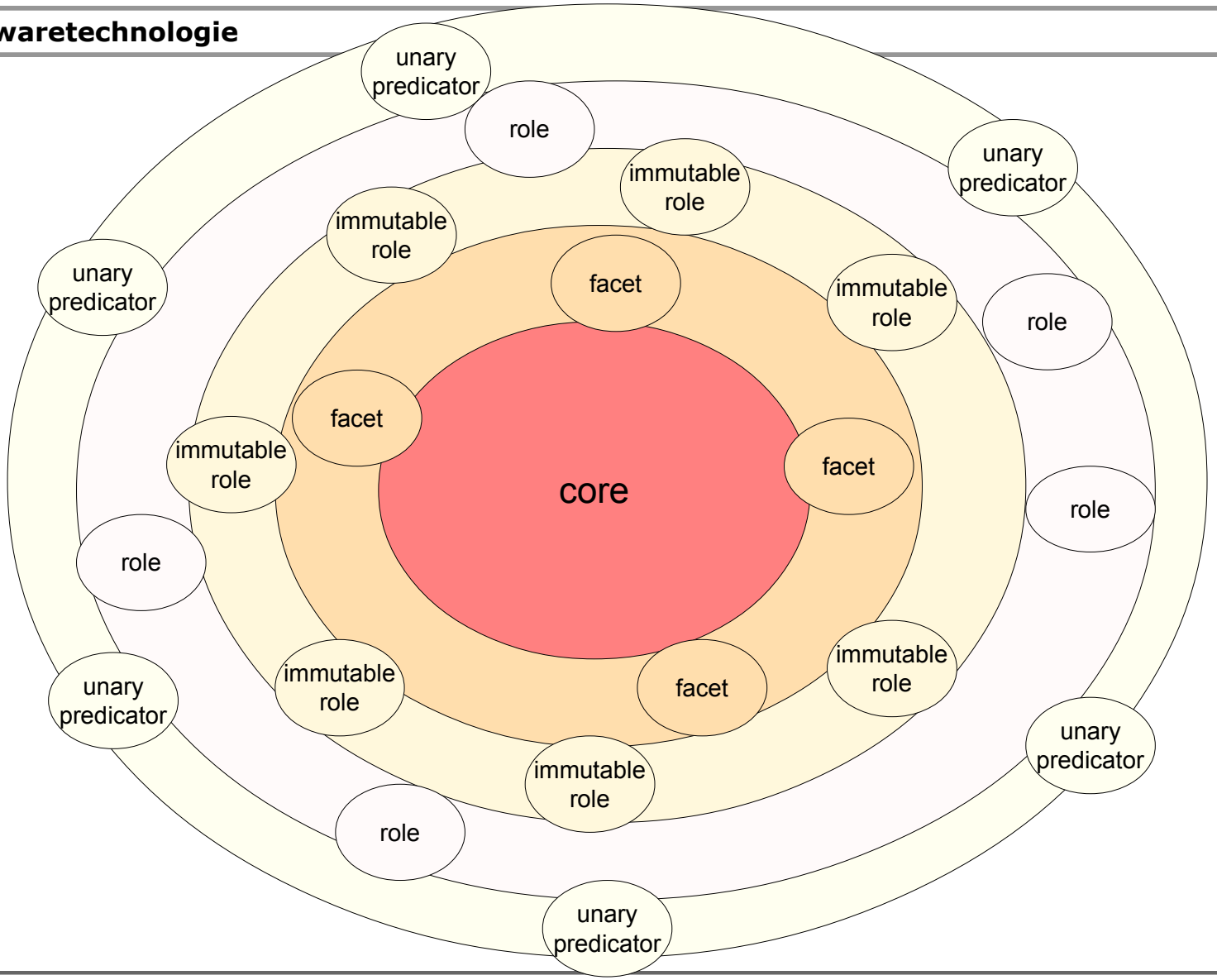
- Facets are classification dimensions, non-founded



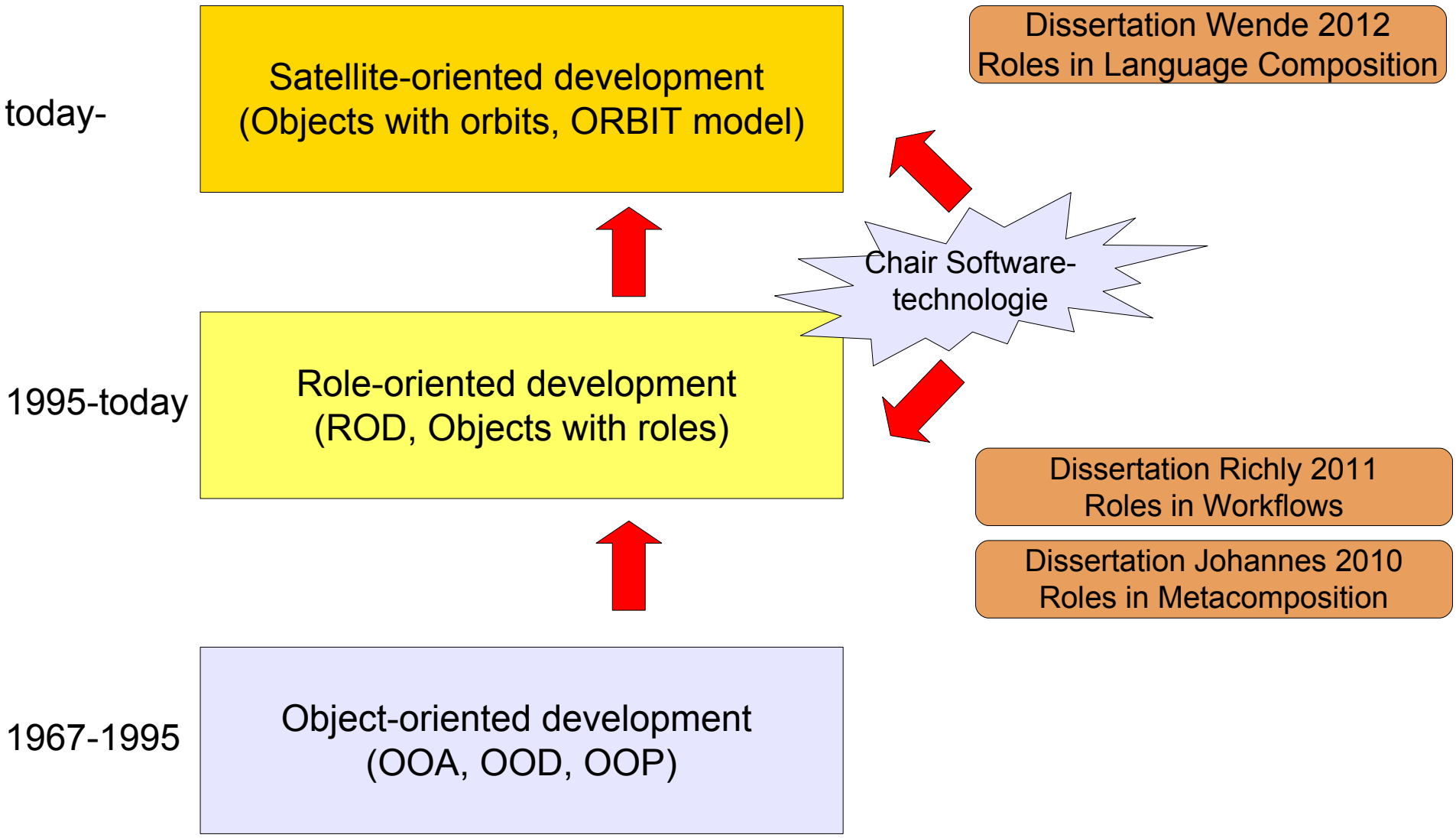
Satellites on Orbits of Cores



Predicator Orbits



Ladder of Paradigms (2)



Future

- Why will role modeling prevail?
  - Better typing, better polymorphism
  - Flexible product lines
    - for resource-aware systems
    - for run-time-flexible systems
- Why will satellite modeling prevail?
  - Because roles are not the only types of unary predicators
  - Because “plays-a” is not the only elementary copula

Relations to the Courses of Chair Softwaretechnologie

Diplom- und Belegarbeiten

Component-Based Software Engineering (CBSE)

Role models as standard services of components
Role models as grey-box composition system
Role models for generic composition

Software-Management (SWM)

Roles in the software process
Roles in process models

Model-Driven Software Development in Technical Spaces (MOST)

Roles in model-driven software development

Design Patterns and Frameworks (DPF)

Role models for design patterns
Composed design patterns
Role object pattern
Roles vs mixins

Softwaretechnologie II

(MDA)

Important References

- T. Reenskaug, P. Wold, and O. Lehne. Working with Objects, The OOram Software Engineering Method. Manning Publications, 1996.
- Friedrich Steimann. On the representation of roles in object-oriented and conceptual modelling. Data Knowl. Eng, 35(1):83-106, 2000.
- Friedrich Steimann. A radical revision of UML's role concept". UML 2000, 3rd International Conference, Springer LNCS, 194–209.
  - and many more, see his home page at U Hagen
- Charles W. Bachman and Manilal Daya. The role concept in data models. In VLDB '1977: Proceedings of the third int.l conf. on Very large data bases, pages 464–476. VLDB Endowment, 1977.
- Giancarlo Guizzardi, Heinrich Herre, and Gerd Wagner. On the general ontological foundations of conceptual modeling. 21st Int. Conf. on Conceptual Modeling (ER 2002), LNCS 2503, pages 65-78, 2002.
- Guizzardi, G. (2005). Ontological Foundations for Structural Conceptual Models. PhD thesis, University of Twente.
- S. Herrmann. Object teams: Improving modularity for crosscutting collaborations. In Proc. Net Object Days 2002, 2002.
- S. Herrmann. A precise model for contextual roles: The programming language objectteams/java. Applied Onthology, (to appear), 2007.
- www.objectteams.org: a programming lanugage with roles

Important References

- D. Bäumer, D. Riehle, W. Silberski, and M. Wulf. Role object. In Conf. On Pattern Languages of Programming (PLOP), 1997.
- Dirk Riehle and Thomas Gross. Role model based framework design and integration. ACM SIGPLAN Notices, 33(10):117-133, October 1998.
- Dirk Riehle. Framework Design - A Role Modelling Approach. PhD thesis, ETH Zürich, 2000. No. 13509. www.riehle.org.
- Y. Smaragdakis and D. Batory. Mixin layers: an object-oriented implementation technique for refinements and collaboration-based designs. ACM Transactions on Software Engineering and Methodology, 11(2):215–255, 2002.
- Paul Lorenzen, Oswald Schwemmer. Konstruktive Logi, Ethik und Wissenschaftstheorie. BI Hochschultaschenbücher, Band 700, 1973.
- Paul Lorenzen. Lehrbuch der konstruktiven Wissenschaftstheorie, Metzler Reprint, 2000.
- H. Wedekind, E. Ortner, R. Inhetveen. Informatik als Grundbildung. Informatik Spektrum, Springer, April 2004
- H. v. Braun, MSP München; W. Hesse, Univ. Marburg; H.B. Kittlaus, SIZ Bonn; G. Scheschonk, C.I.T. Berlin. Ist die Welt objektorientiert? Von der natürlich-sprachlichen Weltsicht zum OO-Modell. Uni Marburg.

Works at SMT so far

TECHNISCHE UNIVERSITÄT DRESDEN

Softwaretechnologie

- U Aßmann, J Johannes, J Henriksson, and Ilie Savga. Composition of rule sets and ontologies. In F. Bry, editor, Reasoning Web, Second Int. Summer School 2006, number 4126 in LNCS, pages 68-92, Sept 2006. Springer.
- M. Pradel, J. Henriksson, and U. Aßmann. A good role model for ontologies: Collaborations. Int. Workshop on Semantic-Based Software Development. at OOPSLA'07, Montreal, Oct 22, 2007.
- Mirko Seifert, Christian Wende, and Uwe Aßmann. Anticipating Unanticipated Tool Interoperability using Role Models. Proceedings of the First Workshop on Model Driven Interoperability (MDI'2010), Oslo, Norway, 2010.
- Sebastian Götz, Max Leuthäuser, Jan Reimann, Julia Schroeter, Christian Wende, Claas Wilke, and Uwe Aßmann. Naotext: A role-based language for collaborative robot applications. In Proceedings of the 1st International ISoLA Workshop on Software Aspects of Robotic Systems, 2011.
- Christian Piechnick, Sebastian Richly, Sebastian Götz, Claas Wilke, and Uwe Aßmann. Using Role-Based Composition to Support Unanticipated, Dynamic Adaptation -- Smart Application Grids. In Proceedings of ADAPTIVE 2012, The Fourth International Conference on Adaptive and Self-adaptive Systems and Applications, pages 93--102, 2012.
- Thomas Kühn, Max Leuthäuser, Sebastian Götz, Christoph Seidl, and Uwe Aßmann. A metamodel family for role-based modeling and programming languages. In Benoit Combemale, David J. Pearce, Olivier Barais, and Jurgen J. Vinju, editors, SLE, volume 8706 of Lecture Notes in Computer Science, pages 141--160. Springer, 2014.
- Thomas Kühn, Stephan Böhme, Sebastian Götz, and Uwe Aßmann. A combined formal model for relational context-dependent roles. In Richard F. Paige, Davide Di Ruscio, and Markus Völter, editors, SLE, pages 113--124. ACM, 2015.
- Matthias Bräuer and Henrik Lochmann. Towards Semantic Integration of Multiple Domain-Specific Languages Using Ontological Foundations.

The End