MOST and
Role-based Context-Aware Software Infrastructures (RoSI)

4. Context- and Role-Oriented Modeling and Development

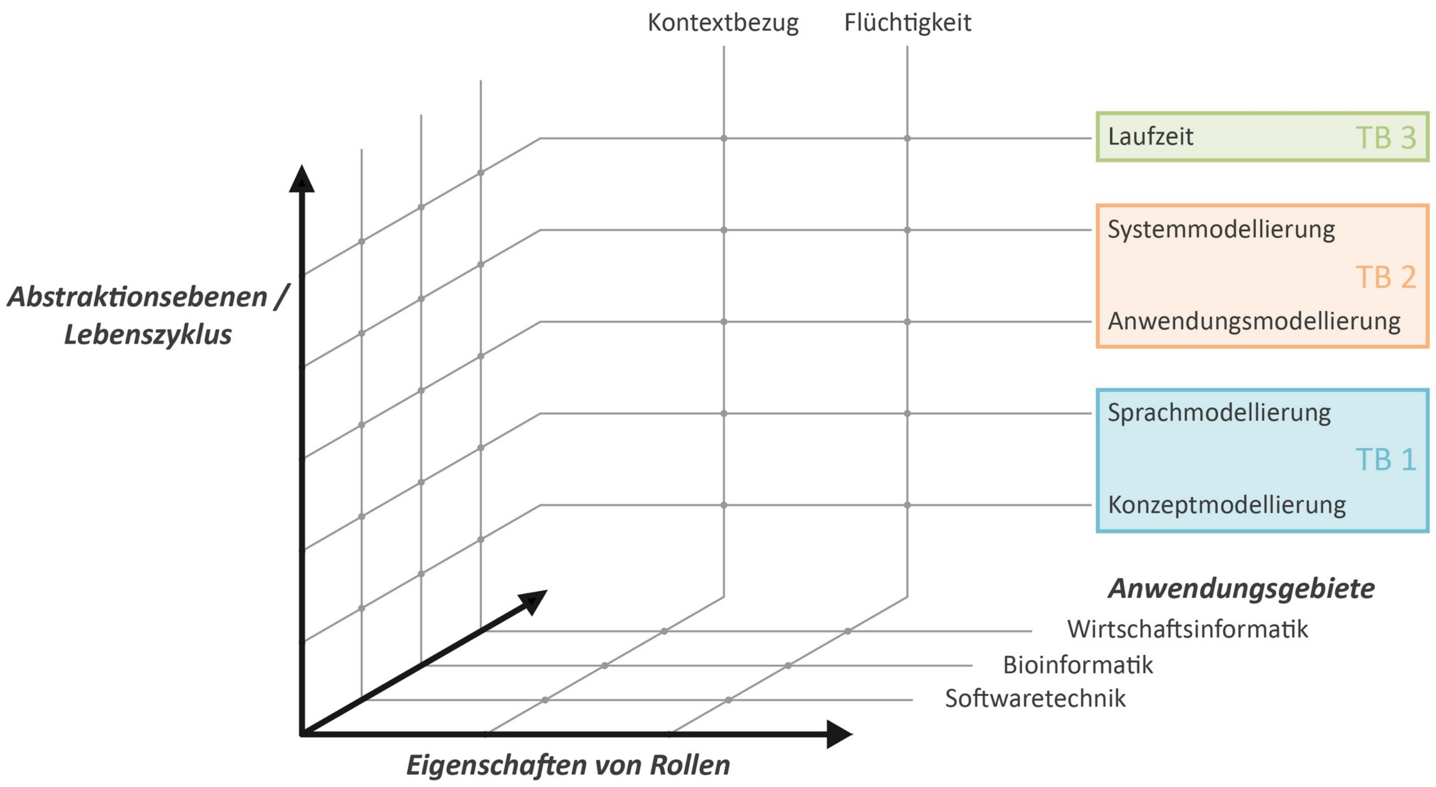
Prof. Uwe Aßmann

Version 20-0.1, 9/27/21

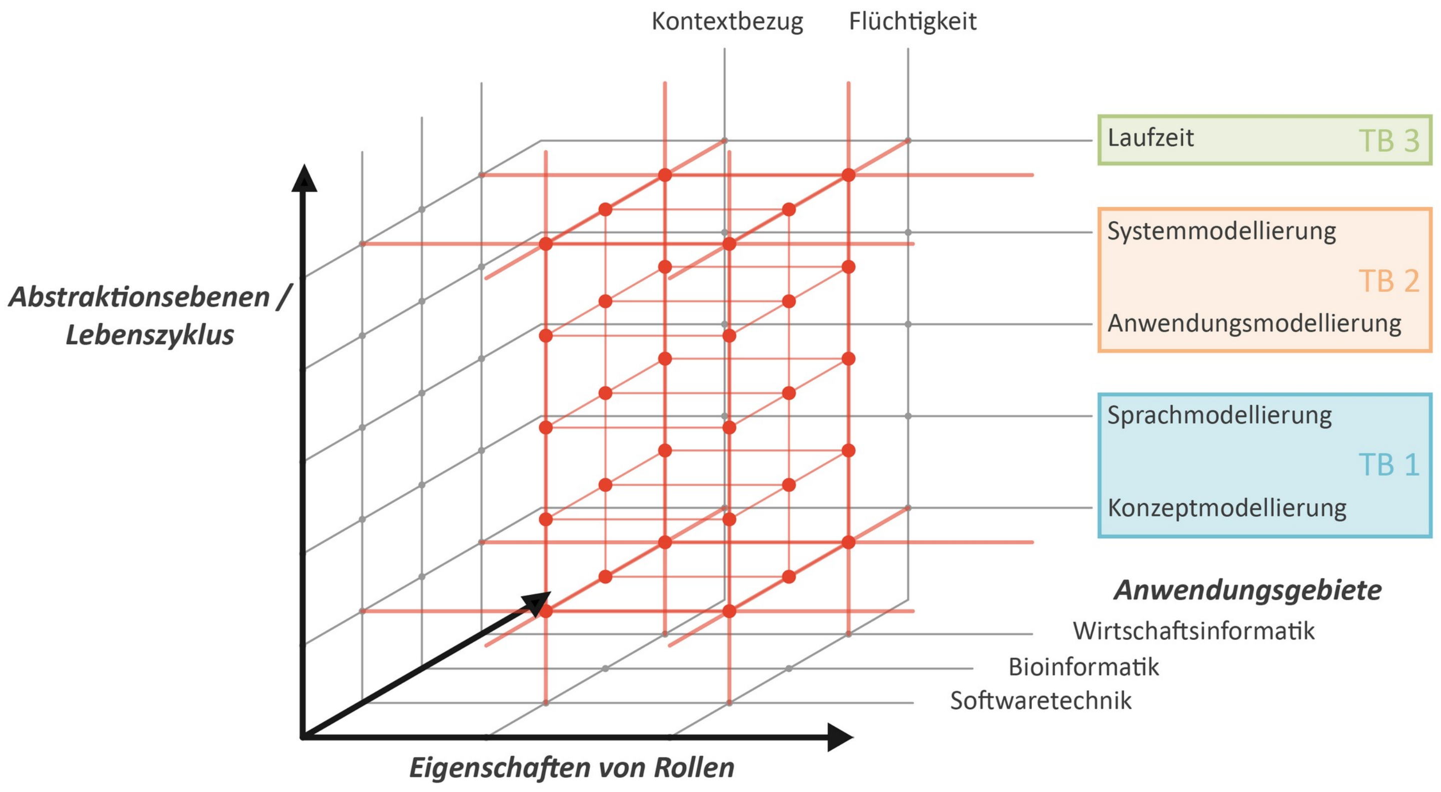
The RoSI Cube

4.1 Roles are a Core Concept in Software Development

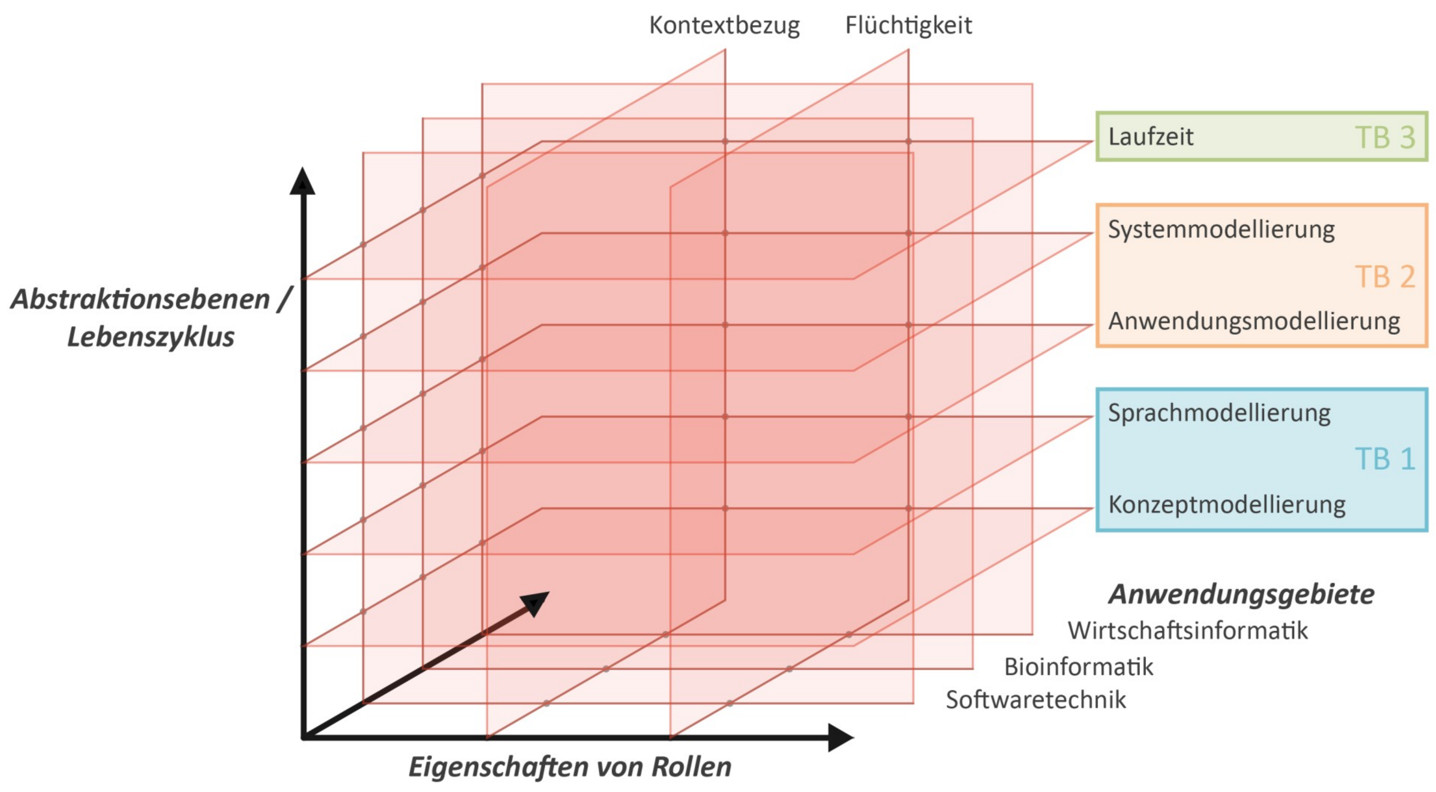
Hypothesis: Roles are a Core Concept of Software Development



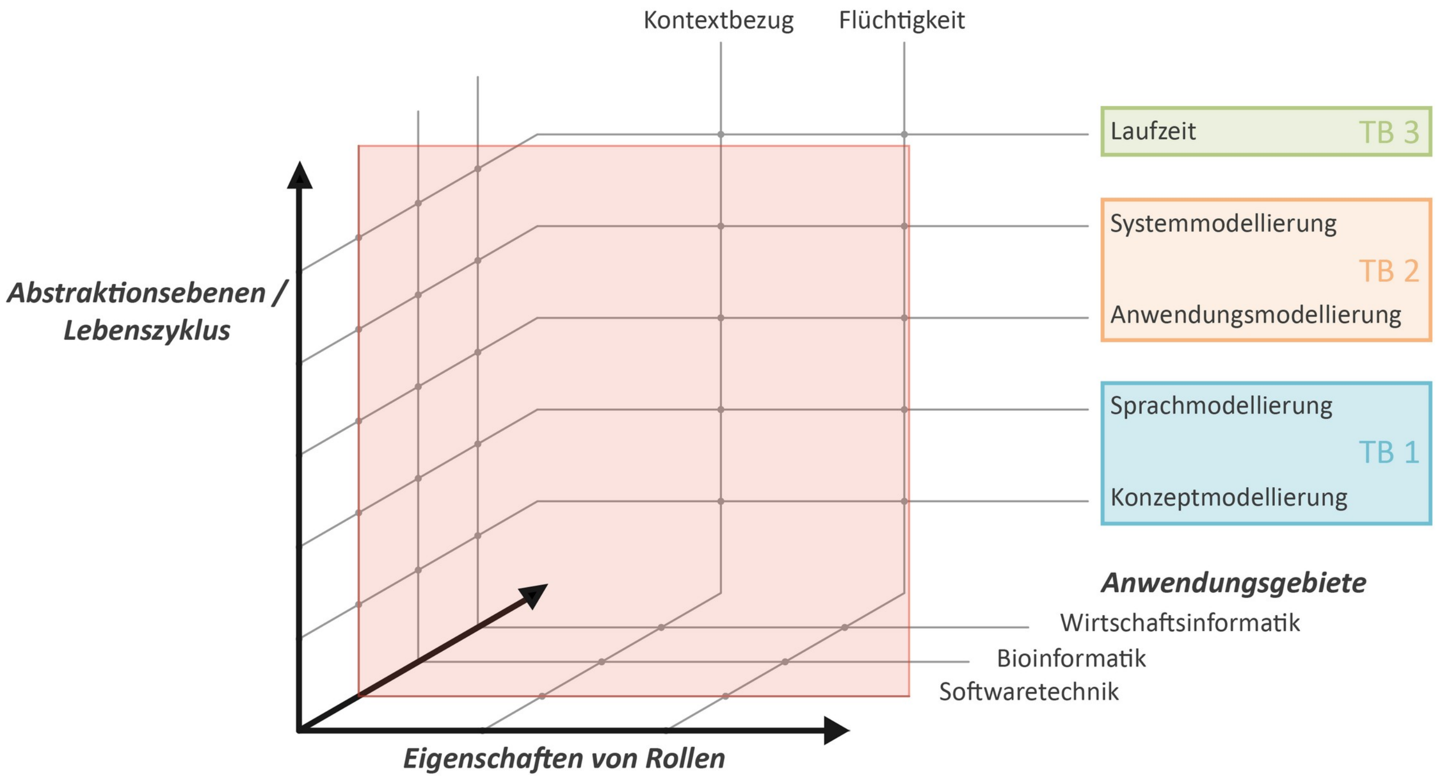
Hypothesis: Roles are a Core Concept of Software Development - *Universality*



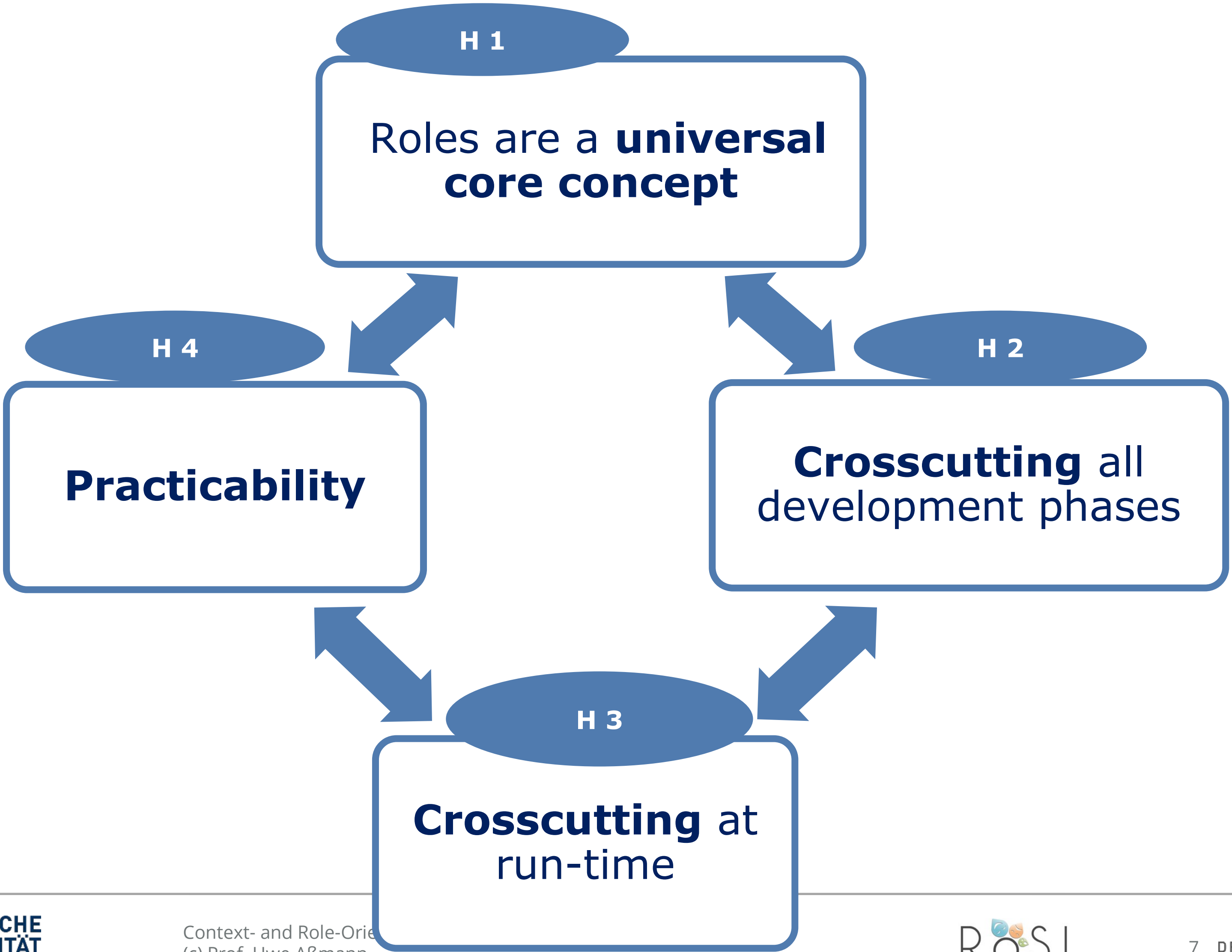
Hypothesis: Roles are a Core Concept of Software Development - *Crosscutting*



Hypothesis: Roles are a Core Concept of Software Development - *Practicality*



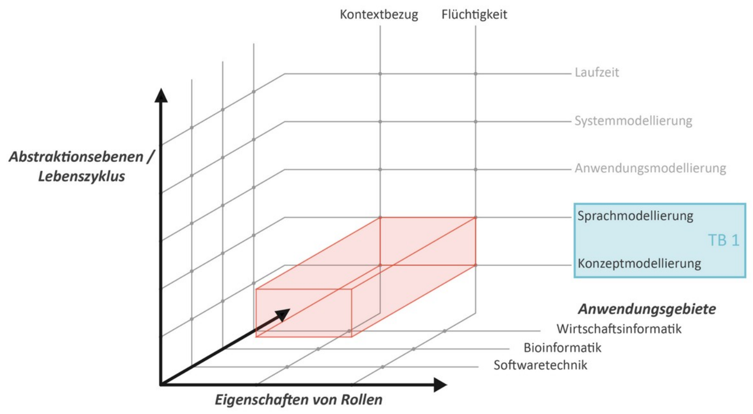
Hypotheses of Role-Oriented Software Infrastructures


H 1
Roles are a **universal core concept**
H 2
Crosscutting all development phases
H 3
Crosscutting at run-time
H 4
Practicability

The RoSI Cube

4.2 Roles as a Universal Core Concept in Software Development

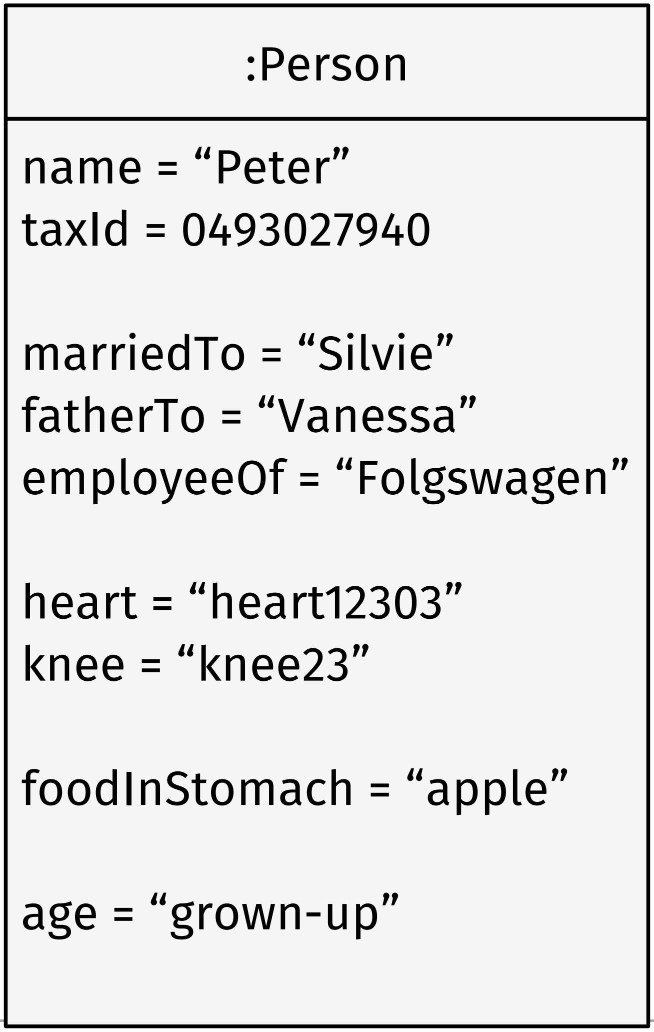
Objective 1: Roles are a Core Concept of Software Development - *Universality*



- Fine-grain information for better analysis of life times
- Behavior abstraction for better provability
- Better extensibility
- Better substitutability

4.2.1. Fine-Grained Information for Separation of Concerns

Different Attributes



:Person

name = “Peter”
taxId = 0493027940

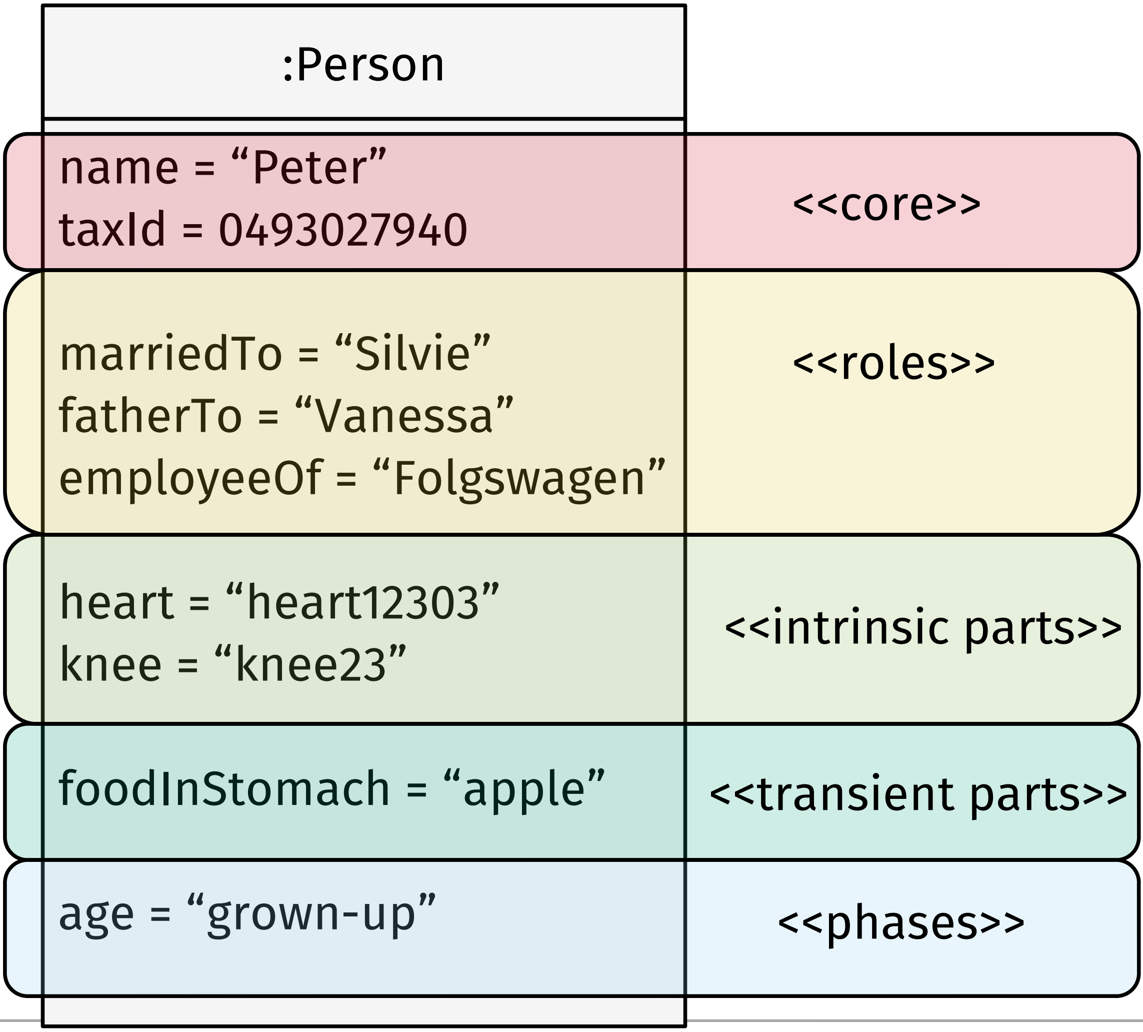
marriedTo = “Silvie”
fatherTo = “Vanessa”
employeeOf = “Folgswagen”

heart = “heart12303”
knee = “knee23”

foodInStomach = “apple”

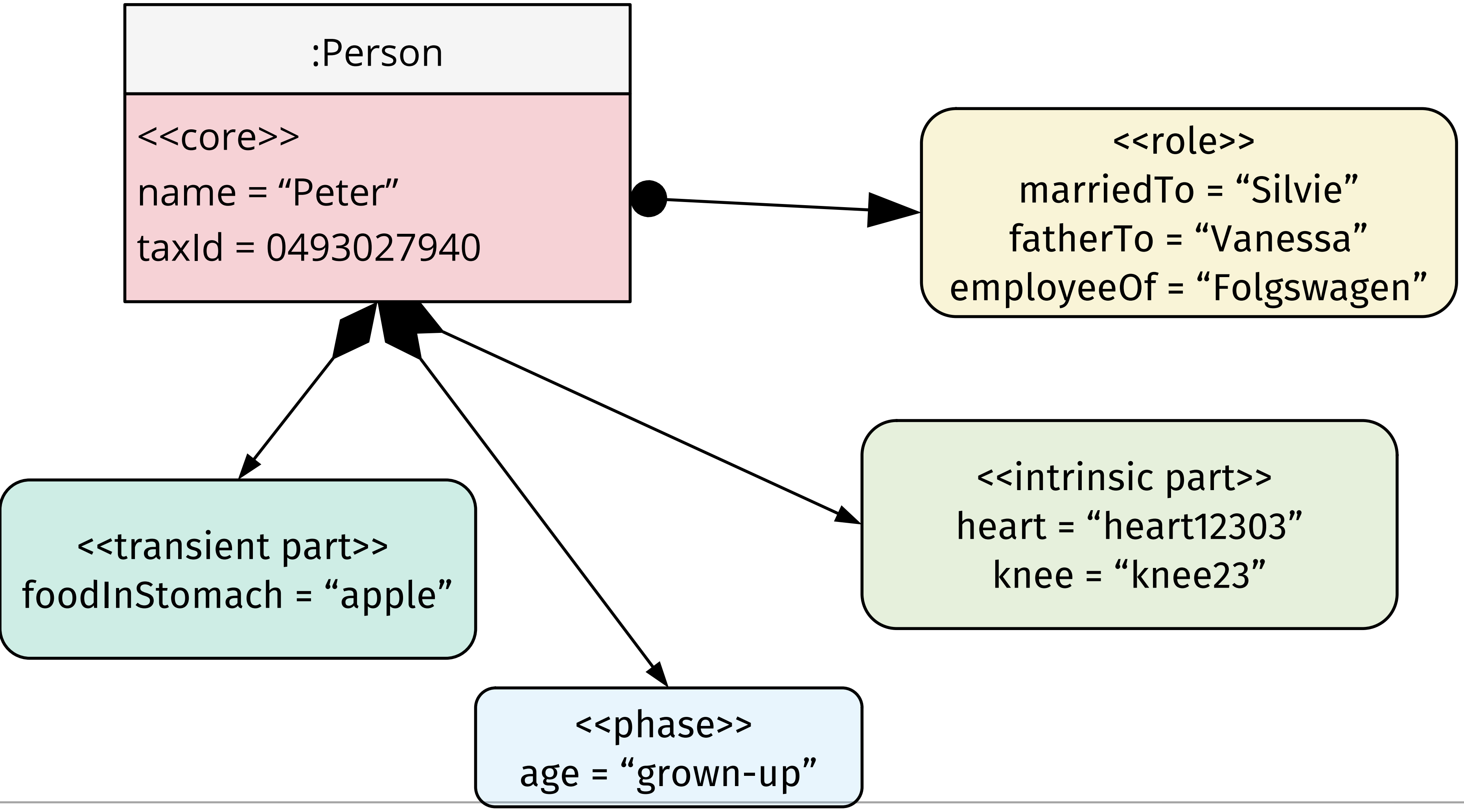
age = “grown-up”

Different Attributes



Cores and Mixins (“Subobjects”, “Satellites”)

Role arrows are drawn with Rounded source



Separation of Concerns with Roles: Distinguishing Life-Times

- Roles are contextually dependent (founded), and have a different life-time as the core
  - → Memory allocation must be different
- Distinguish core-local, role-local, role-alternative, role-shared memory between core and roles
  - natural memory (core-local memory)
  - founded memory (context-dependent memory)
- Roles-of-roles (deep roles) are stacked upon roles;
  - Obstack allocation possible (mark-release heaps)
-

Roles can improve knowledge about life-time and co-life-time of memory

Separation of Concerns with Roles: Alias Freedom and Data Independence

- Natural and role-local memory are alias free
- Shared memory is still problematic (competitive writes)

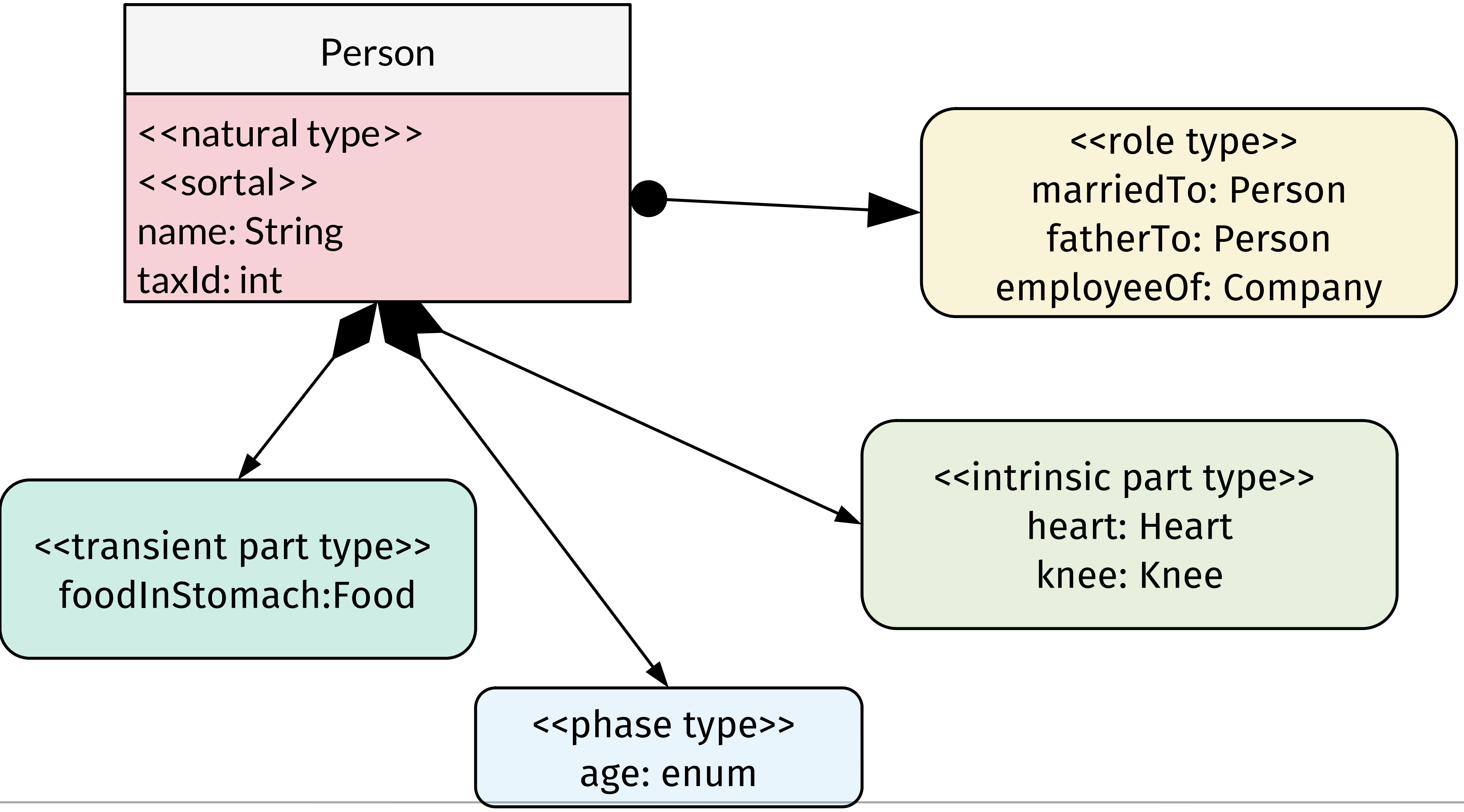
Roles can improve life-time and independence knowledge

Role Types are Metatypes (Mixin Types)

- A **metatype** describes a type (is a type of a type) [Guarino:OntoClean]
  - Natural Type
  - Part Type (intrinsic, shared, owned,..)
  - Role Type
  - Facet Type
  - Phase Type

Hypothesis:
The distinction of metatypes promotes Separations of Concerns.

Distinguishing Mixin Types ("Colors", "Metatypes", "Satellite Types")



Separation of Concerns Helps

- The distinction of **metatypes** enables us to separate more concerns (SoC)
  - And bring it to run-time: Life-time, independence, ….
  - Cross-cutting: traceability, certification,...

Roles can improve modeling and programming.

Role-Oriented Context-Aware Software Infrastructures (ROSI)

4.2.2. Abstraction of Object Behavior - Compartments and Role Playing

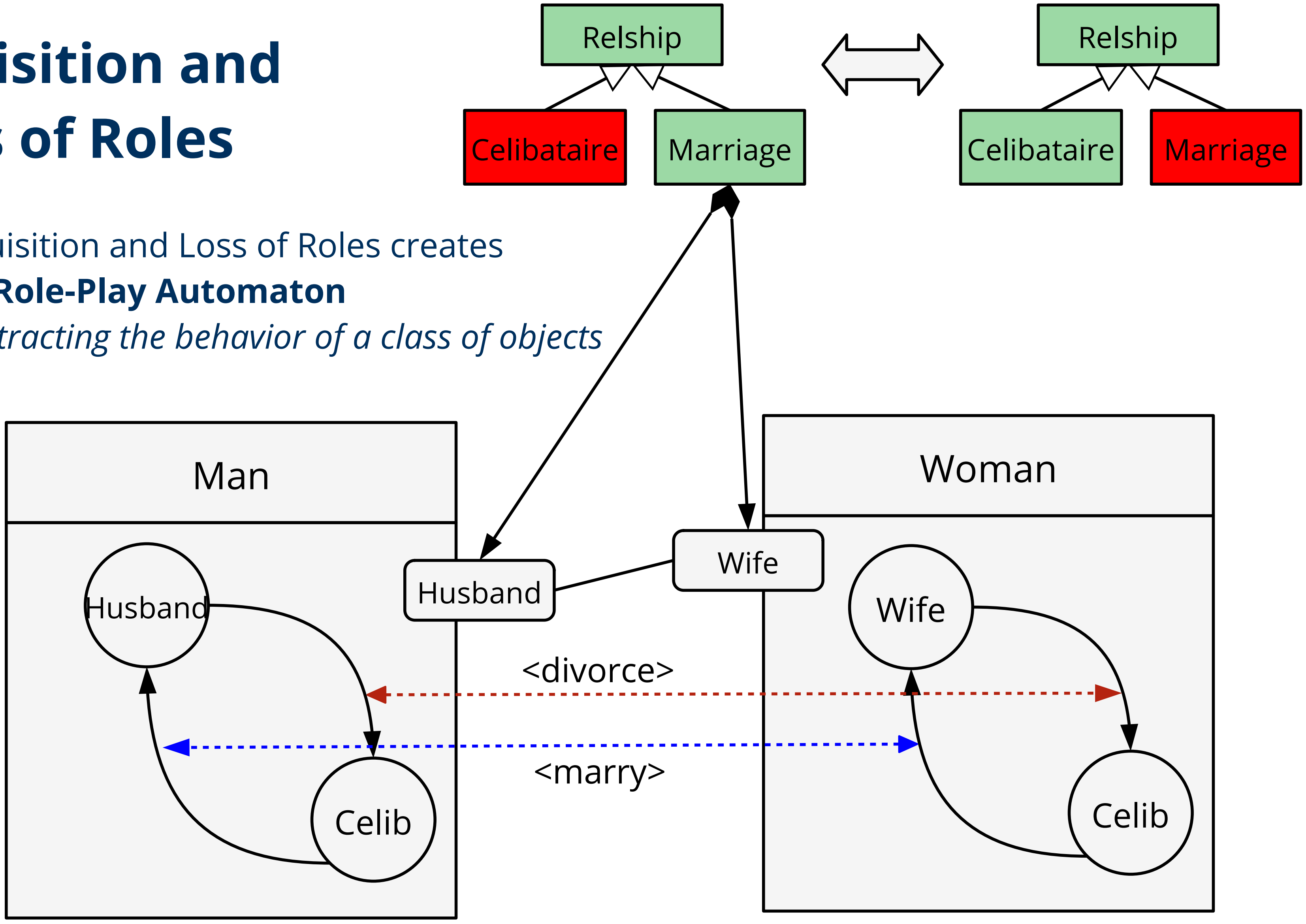
Roles are a Core Concept
Advantages of Roles:
The Role-Play Automaton
The Role-Play Petri Net

Role-Play Nets

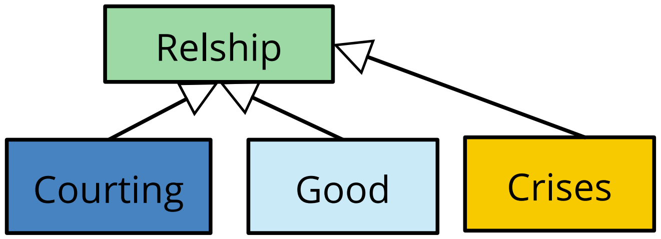
- The **role-play (petri) net of an object** switches in and off the object‘s roles
  - Specifies constraints on the order of the role play
  - Thereby constraints on the compartment activation
- Roles are specific states indicating
  - There is a compartment active to which the role belongs
  - There is a partner role within the compartment that can be called or notified or streamed
- Two forms:
  - Role-Play automaton (sequential)
  - Role-play net (parallel)

Aquisition and Loss of Roles

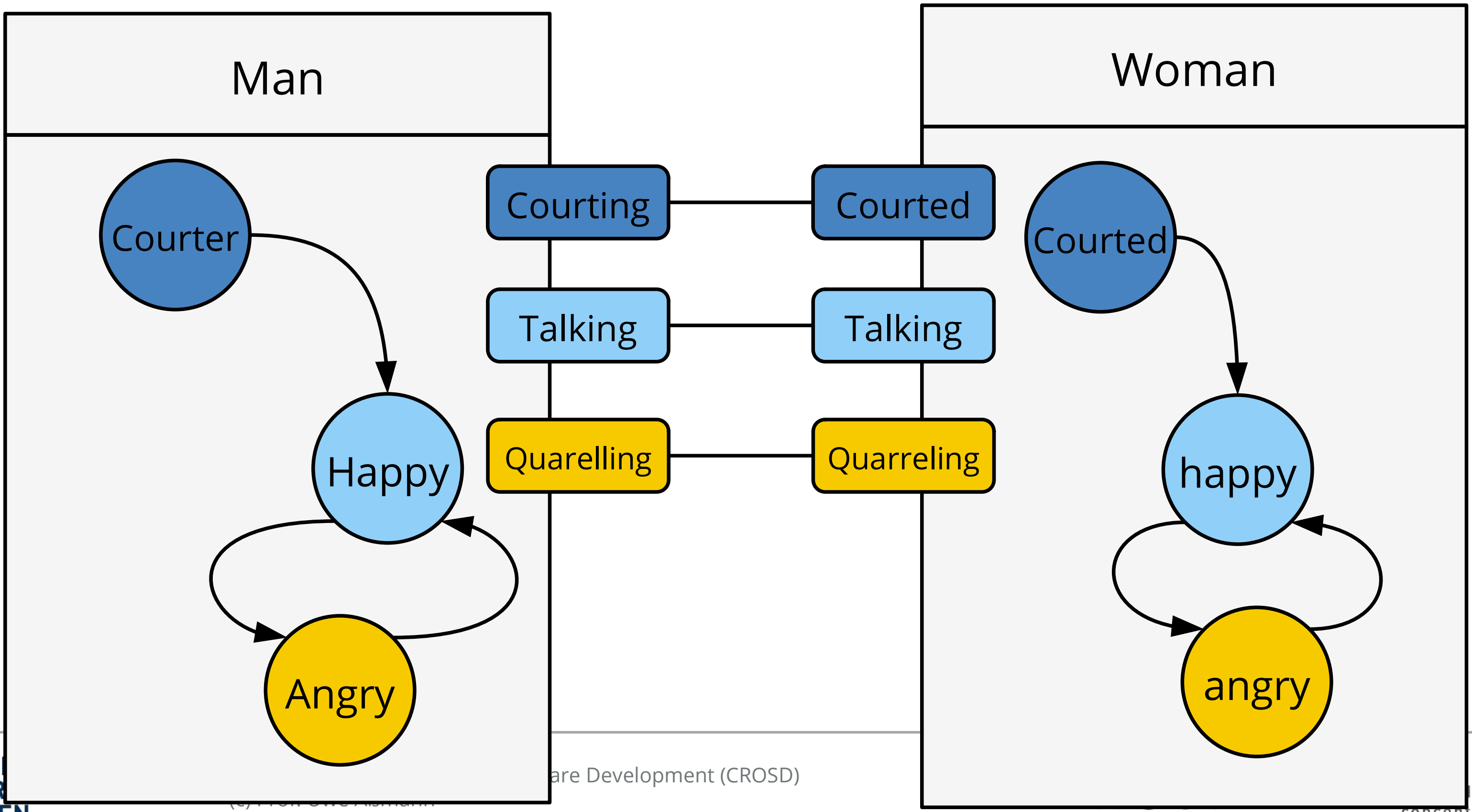
- Aquisition and Loss of Roles creates an **Role-Play Automaton** *abstracting the behavior of a class of objects*



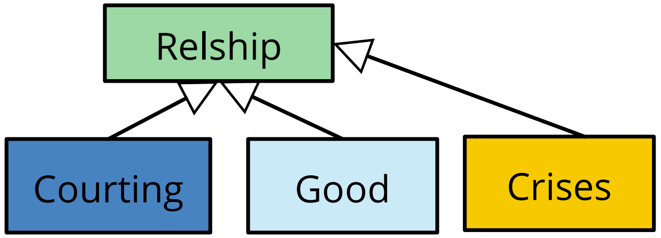
Aquisition and Loss of Roles with Role-Play Automata



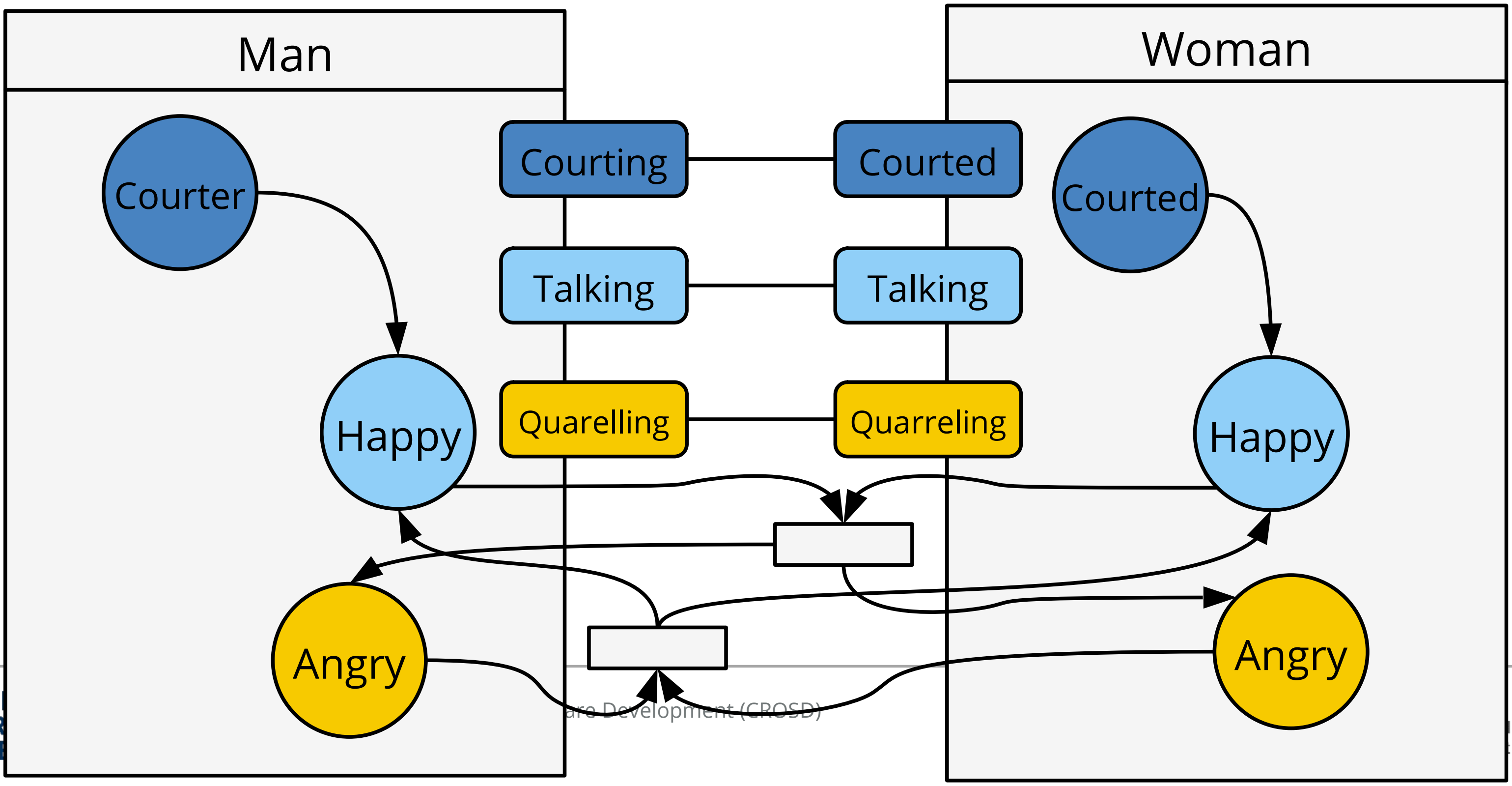
- Aquisition and Loss of Roles creates an **Role-Play Automaton** *abstracting the behavior of a class of objects*
- *Here:* some states with the same color are coupled



Aquisition and Loss of Roles with Role Nets



- Aquisition and Loss of Roles of parallel objects and their state transitions creates a **Role-Play Net** indicating parallel transitions
- **Here:** exclusive compartments, exclusive roles
- Coupling via synchronizing transitions



A Fancy Observation

Humans think and argue based on Role-Play Nets

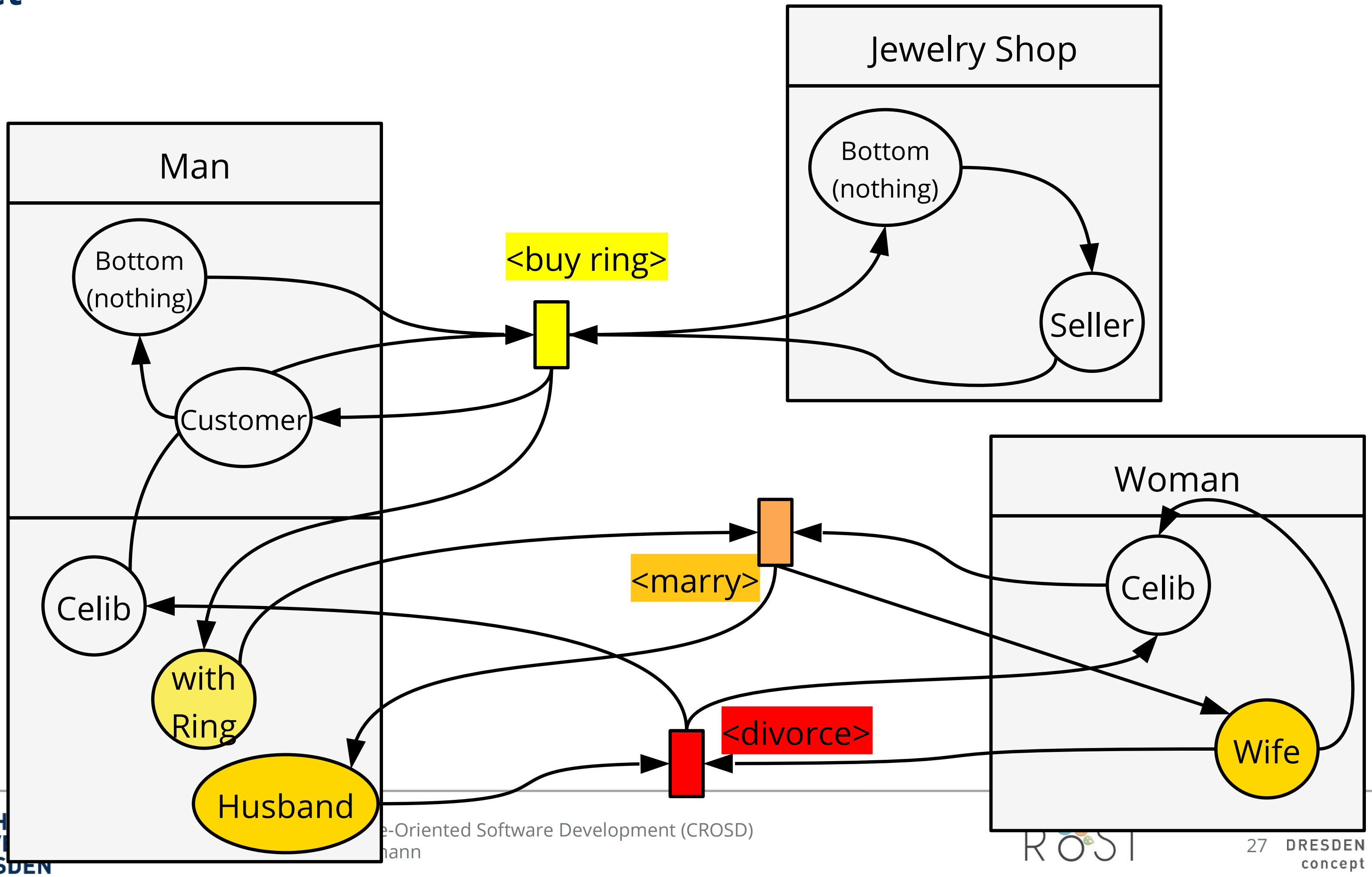
- “become a father”
- „if you are a husband, you should care about your wife“
- “become a driver”, „drivers, watch out for pedestrians“
- “cease to be an employee”
- “cease to be student”

Role-Play Net of a Compartment

- The **role-play net of a compartment** is the view on all role-play nets comprising all roles places of the compartment.
- When a compartment is activated there is the constraint that
  - all the compartment‘s roles in all their players are activated (firable)
  - Otherwise the net is inconsistent.
- When a compartment is deactivated there is the constraint that
  - all the compartment‘s roles in all their players are deactivated (non-firable)

Parallel Aquisition and Loss of Roles

- Parallel Aquisition and Loss of Roles in a parallel class creates an **Role-Play (Petri) net**



Regular Adaptability and Variability

- Many applications have a restricted form of adapability (variability)
- A **regularly adaptable class** has a finite role-play automaton with n compartments as states
  - Infinitely many adaptations, but regularly many

The role-play petrinet of a regularly adaptable class is k-bounded.

Roles are a Core Concept

4.2.3. Advantages of Roles: Behavioral Extensibility

Extensibility as a Universal Feature of Role-based Infrastructures

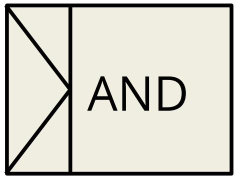
- New compartments with their roles can easily be integrated into an application → extensibility (see lecture 01)
- Roles may have different implementation paradigms (groundings):
  - Functional programs
  - Workflow nets
  - Data-flow nets (see MOST)
  - Attributed trees (see MOST)
- All of them have the extensibility feature, but use different „open operators“ for extensions.

Example: Extending Role-based Systems Grounded by Workflow Nets (Petri Nets)

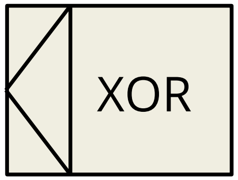
- With an appropriate behavioral specification language, role classes and natural classes can be extended with regard to behavior
- Example: Workflow Nets are a specific form of Petri Nets
  - **Place workflow nets** have one single input place and a single output place
  - **Transition workflow nets** have one single input transition and a single output transition
- For extension (and variation) of behavior of classes, we use the extension of AND, OR, XOR split and join *open transition operators*

Complex Transition Operators in Workflow Nets: Join and Split „Open“ Transitions (of YAWL)

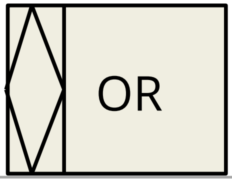
- All incoming places are ready (conjunctive input, AND-join)



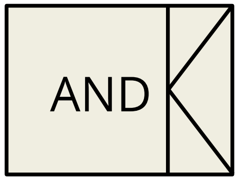
- One out of n incoming places are ready (disjunctive input)



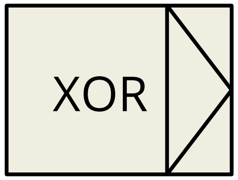
- Some out of n incoming places are ready (selective input)



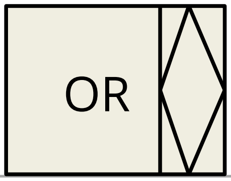
- All outgoing places are filled (conjunctive output, AND-split)



- One out of n outgoing places are filled (disjunctive output, XOR split)

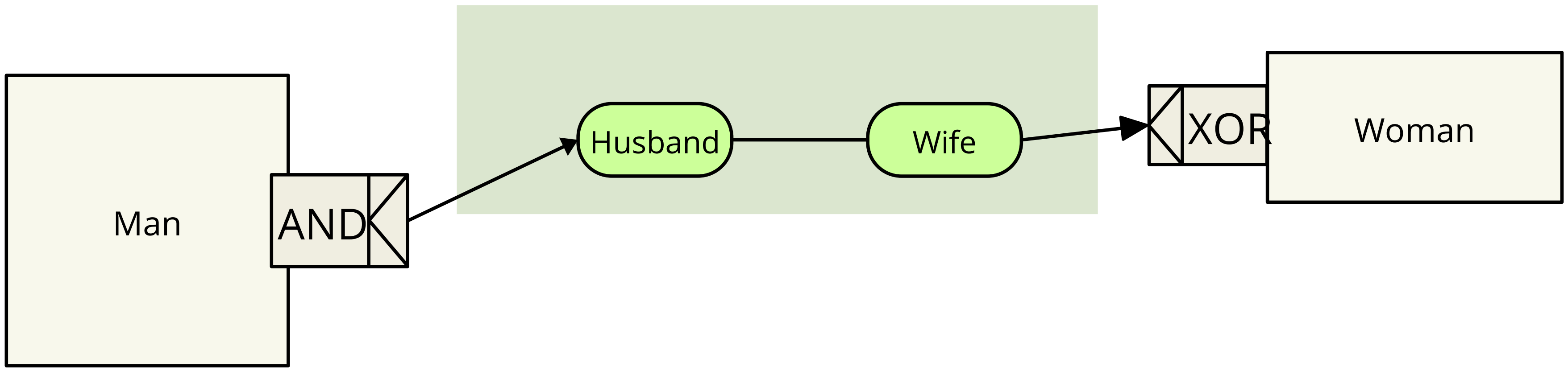


- Some out of n outgoing places are filled (selective output, OR-split)



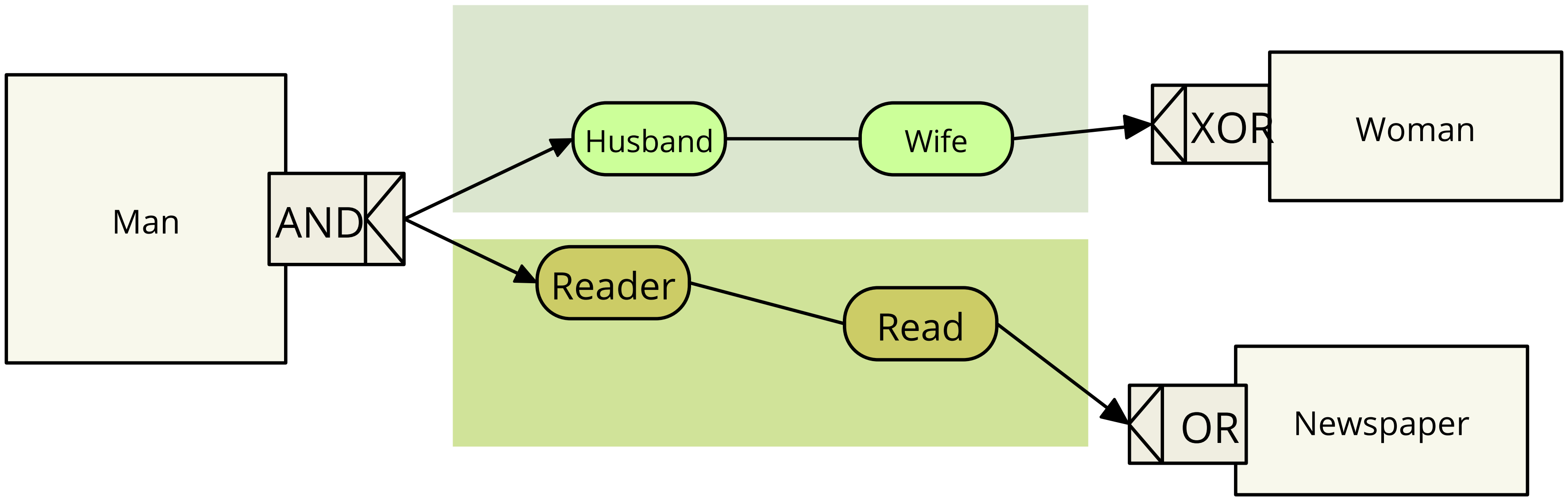
Extension of Workflows with new Place Workflow Nets

- Behavior can be added in *slices* to *open* split and join operators



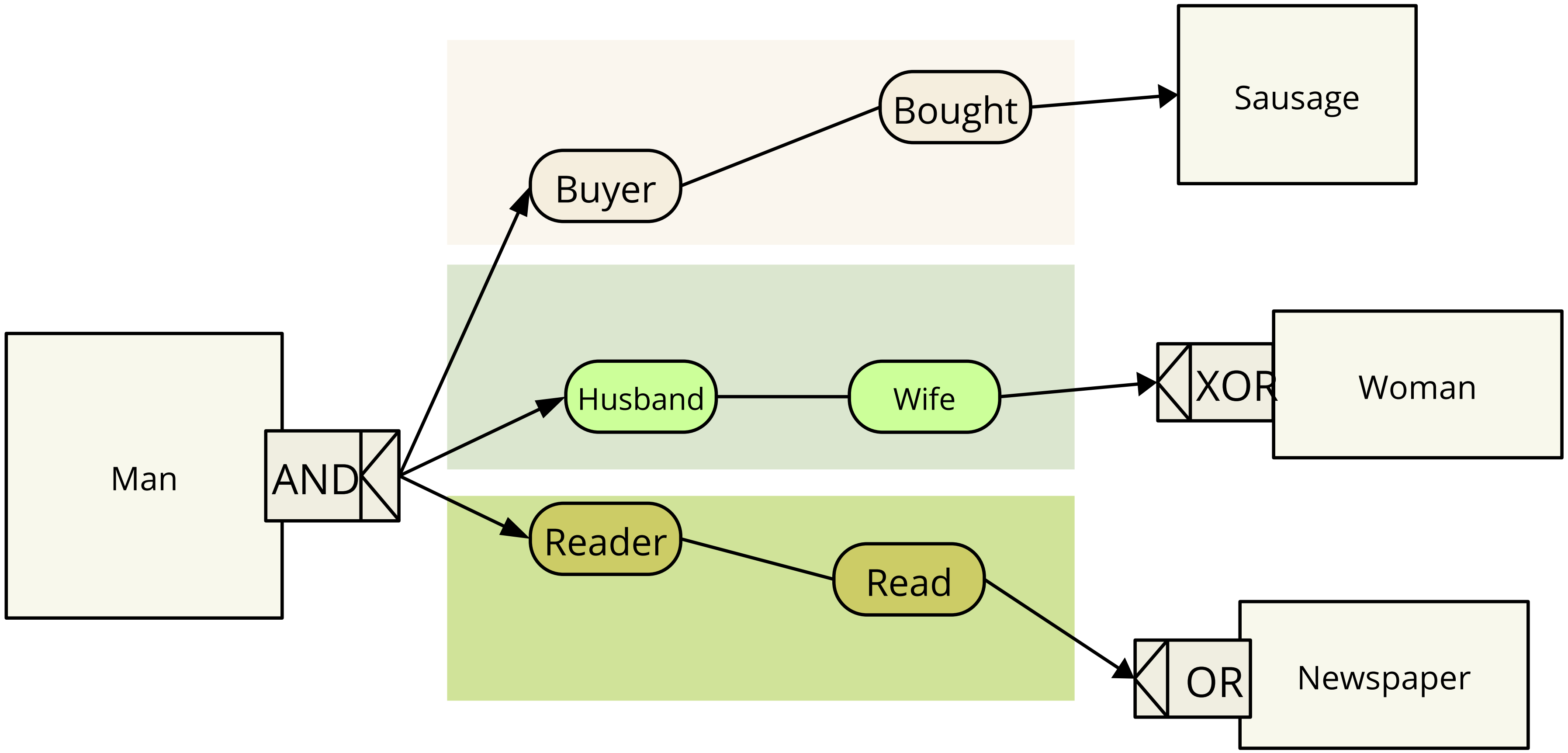
Extension of Workflows with new Place Workflow Nets

- Behavior can be added in *slices* to *open* split and join operators



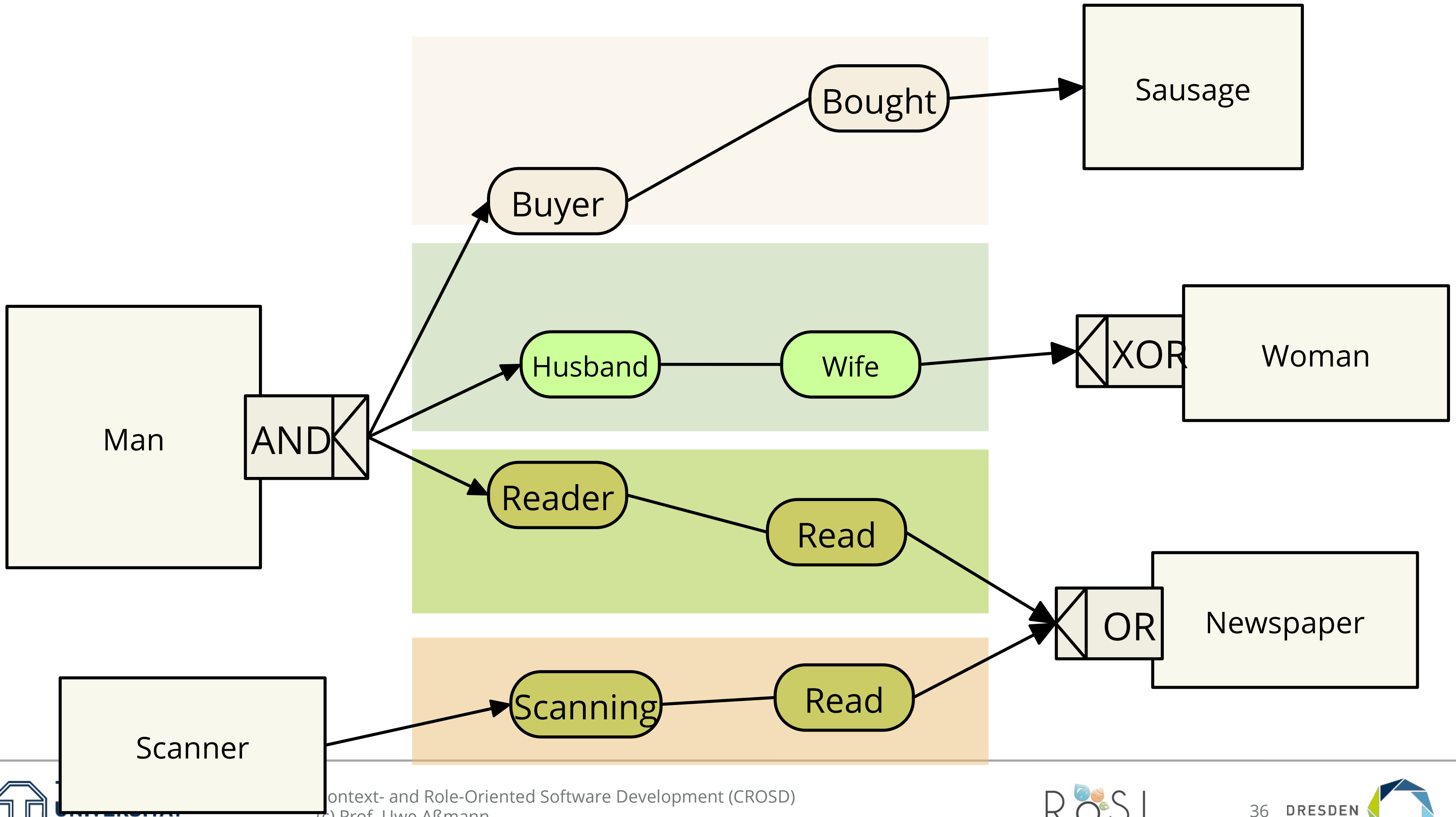
Extension of Workflows with new Place Workflow Nets

- with AND semantics



Extension of Workflows with new Place Workflow Nets

- with OR semantics



Roles as a Core Concept in Software Development

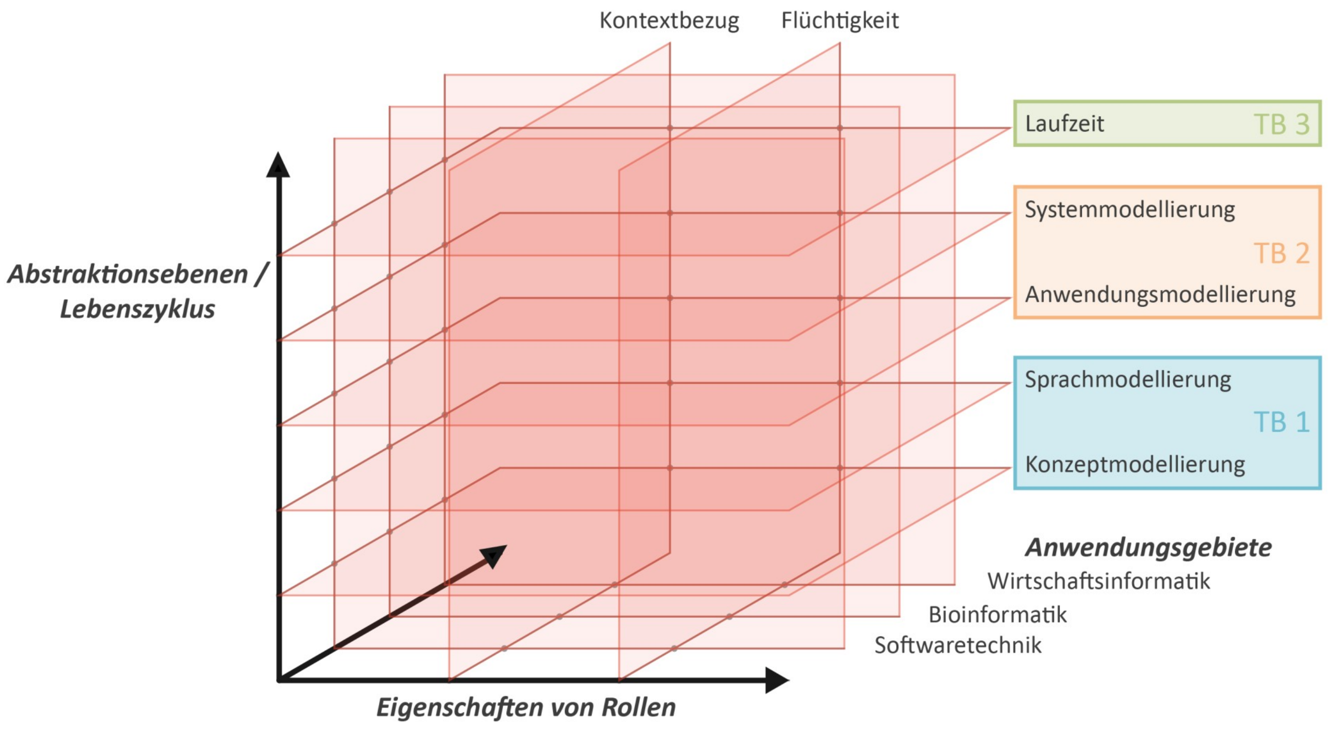
4.2.4 Better Substitutability: Role-Specific Contracts

Separation of Concerns with Roles: Role-Based Contracts are Context-Based

- Contracts describe conditions for *substitutability*
- A **contract** is a constraint on inputs (precondition), outputs (postcondition) and invariants of a component (see courses CBSE, ST)
- Life-time and Alias Independence enable simpler proof of contracts
- The Role-Play Automaton determines which contracts are active
  - in which context

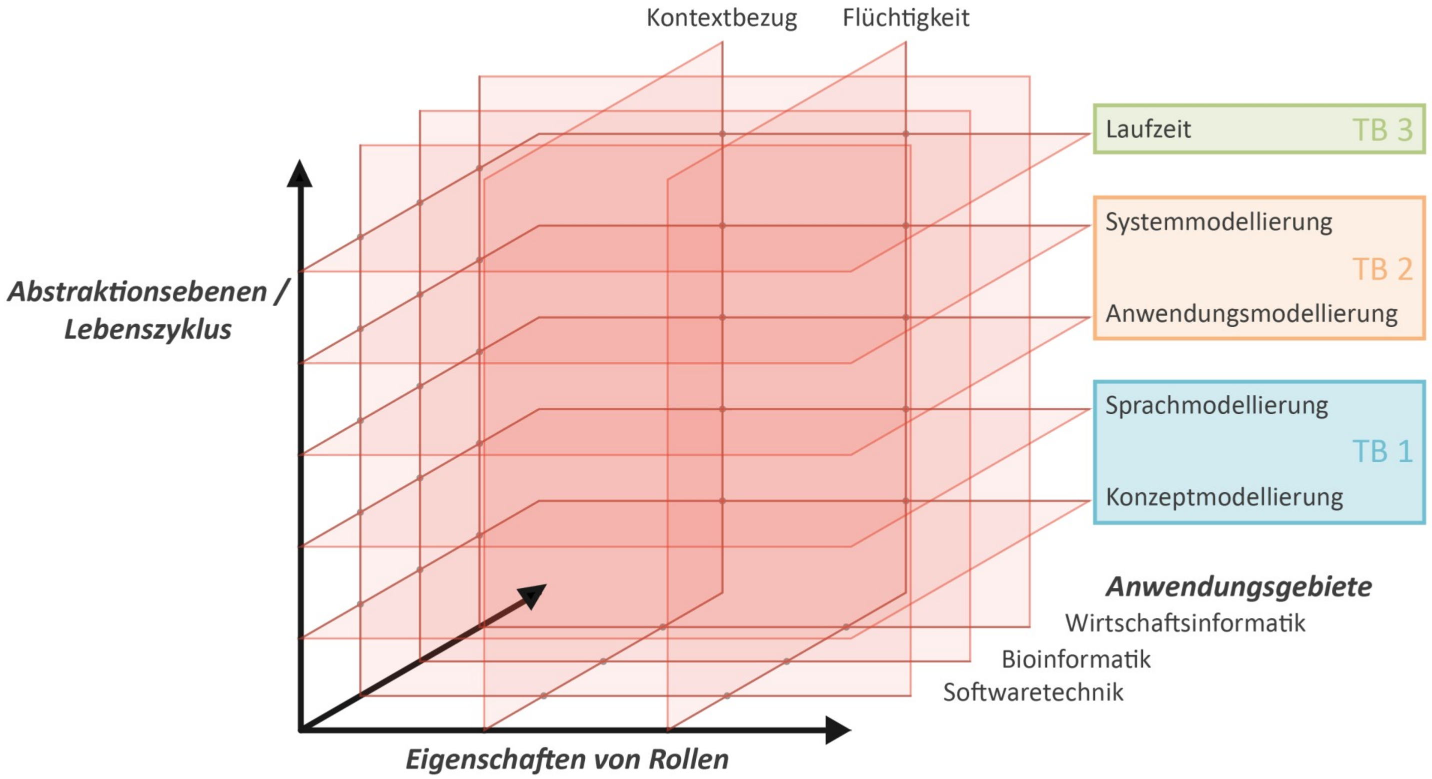
Roles can improve contract theory for sequential and parallel classes

Summary: Roles are a Core Concept of Software Development



4.3. Roles are a Concept Crosscutting all Phases

Objective 2: Roles Crosscut all Development Phases



Roles as a Concept Crosscutting all Phases

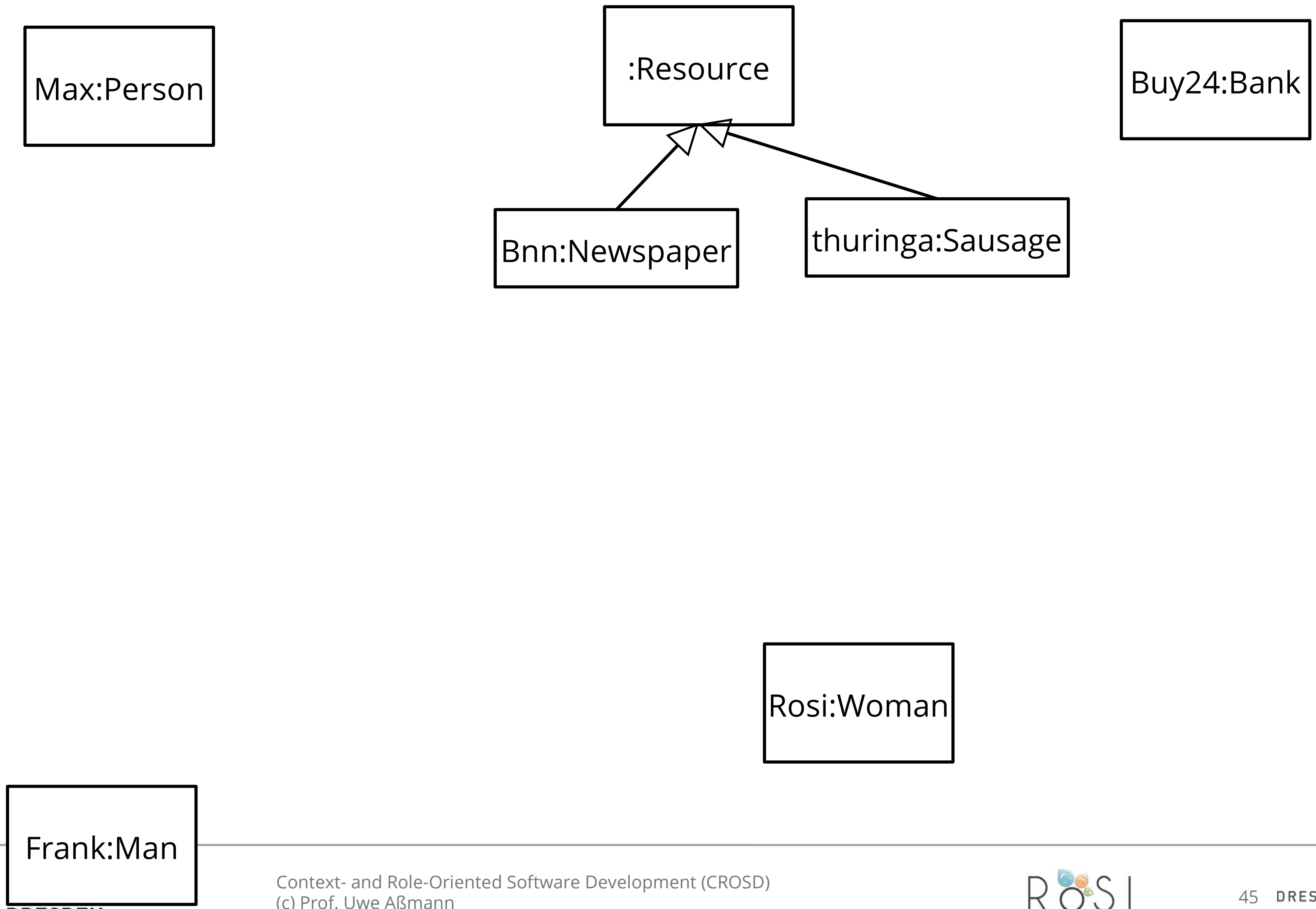
4.3.1 Roles in Software Modeling

4.3.1.1. How to Do Object-Oriented Analysis with ROSI

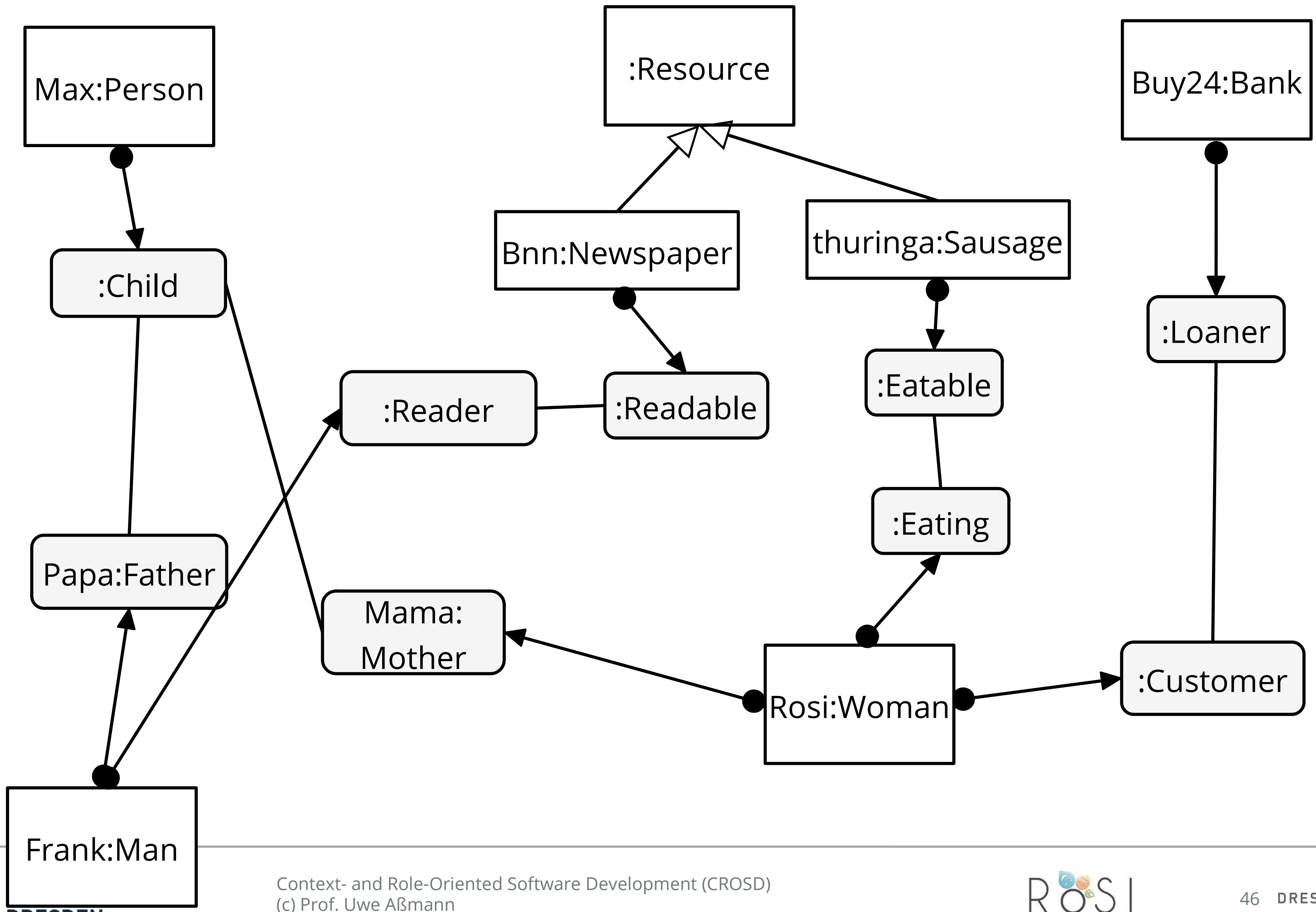
RoSI Object Models
RoSI Component Models

- An **Object Model** describes a structure and behavior for all objects in all phases of the life cycle
  - It forms type systems
  - specification languages
  - the parallelism available
- Roles and Contexts can be used in Object-oriented Analysis (OOA), offering a very flexible object model

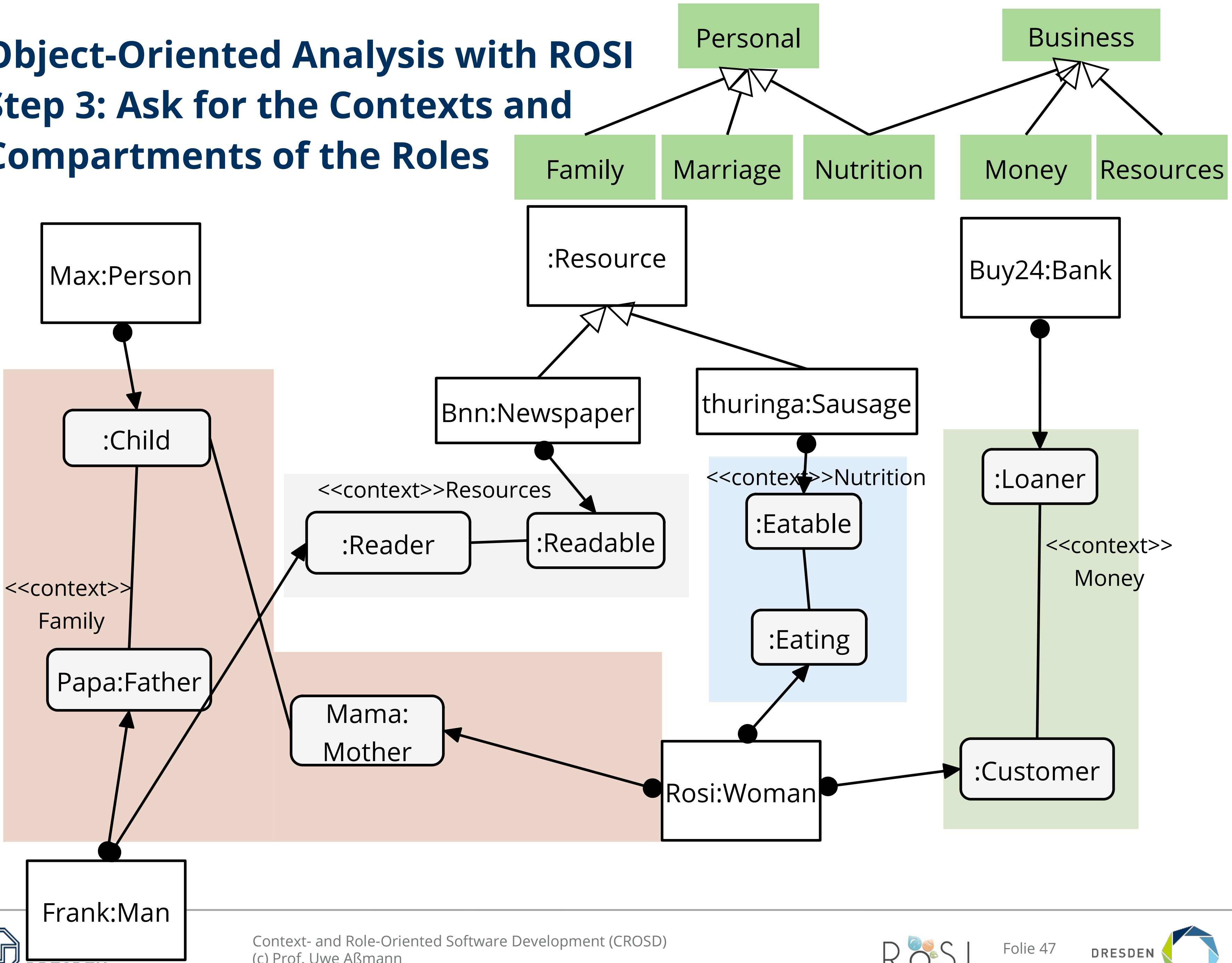
Object-Oriented Analysis with ROSI Step 1: Ask for the Core Objects with Natural Types



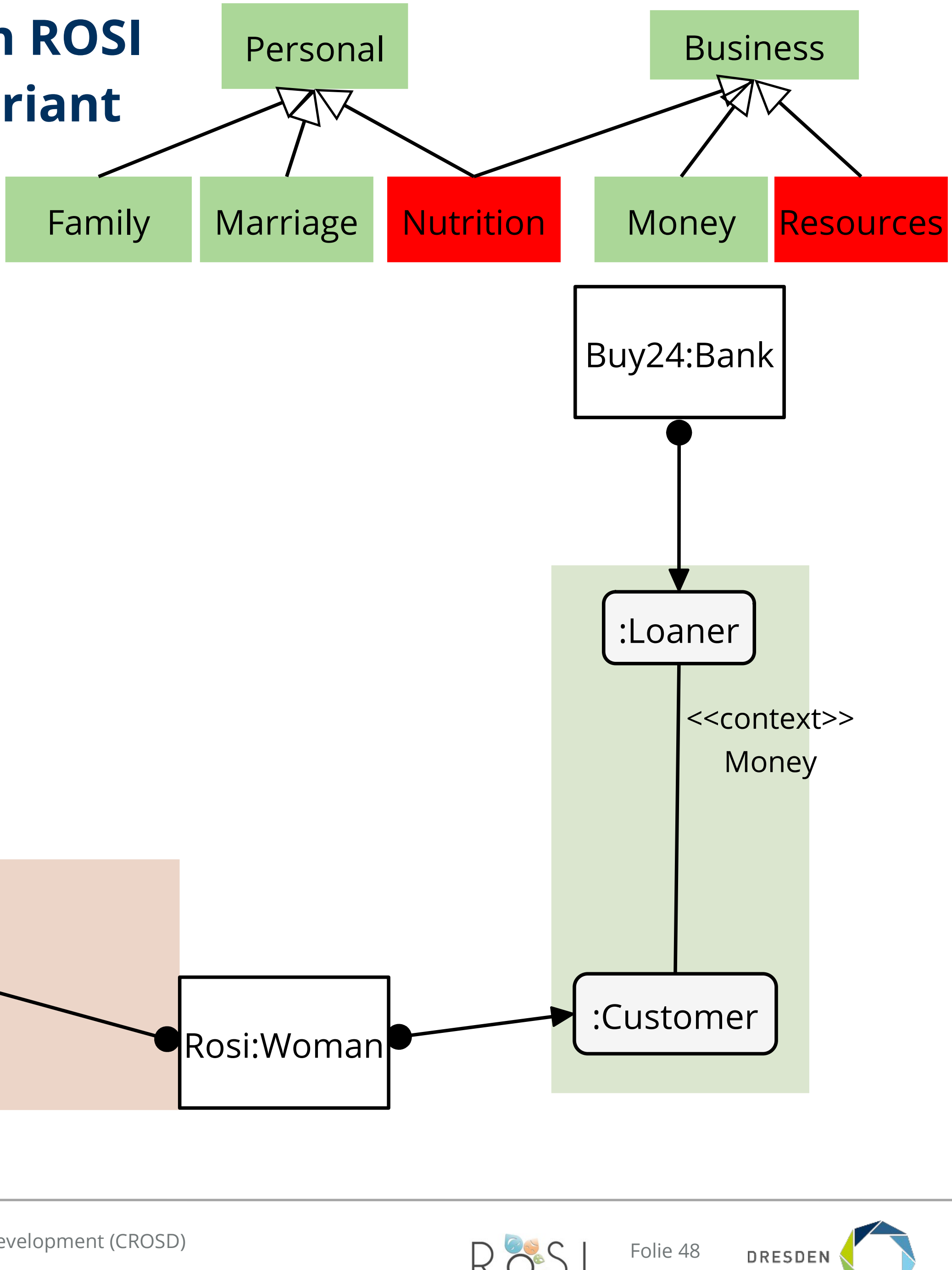
Object-Oriented Analysis with ROSI
Step 2: Ask for the Roles with Founded Types



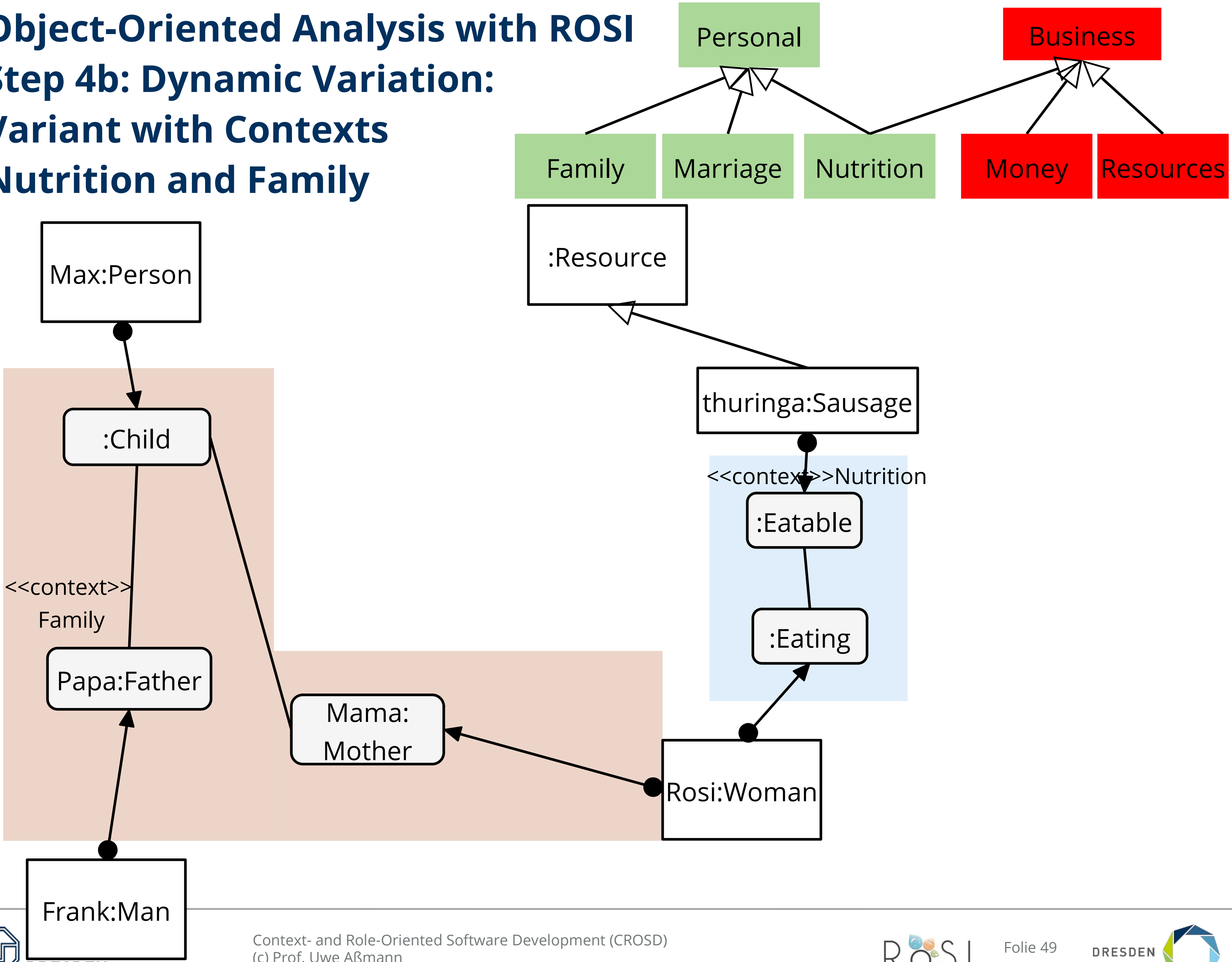
Object-Oriented Analysis with ROSI Step 3: Ask for the Contexts and Compartments of the Roles



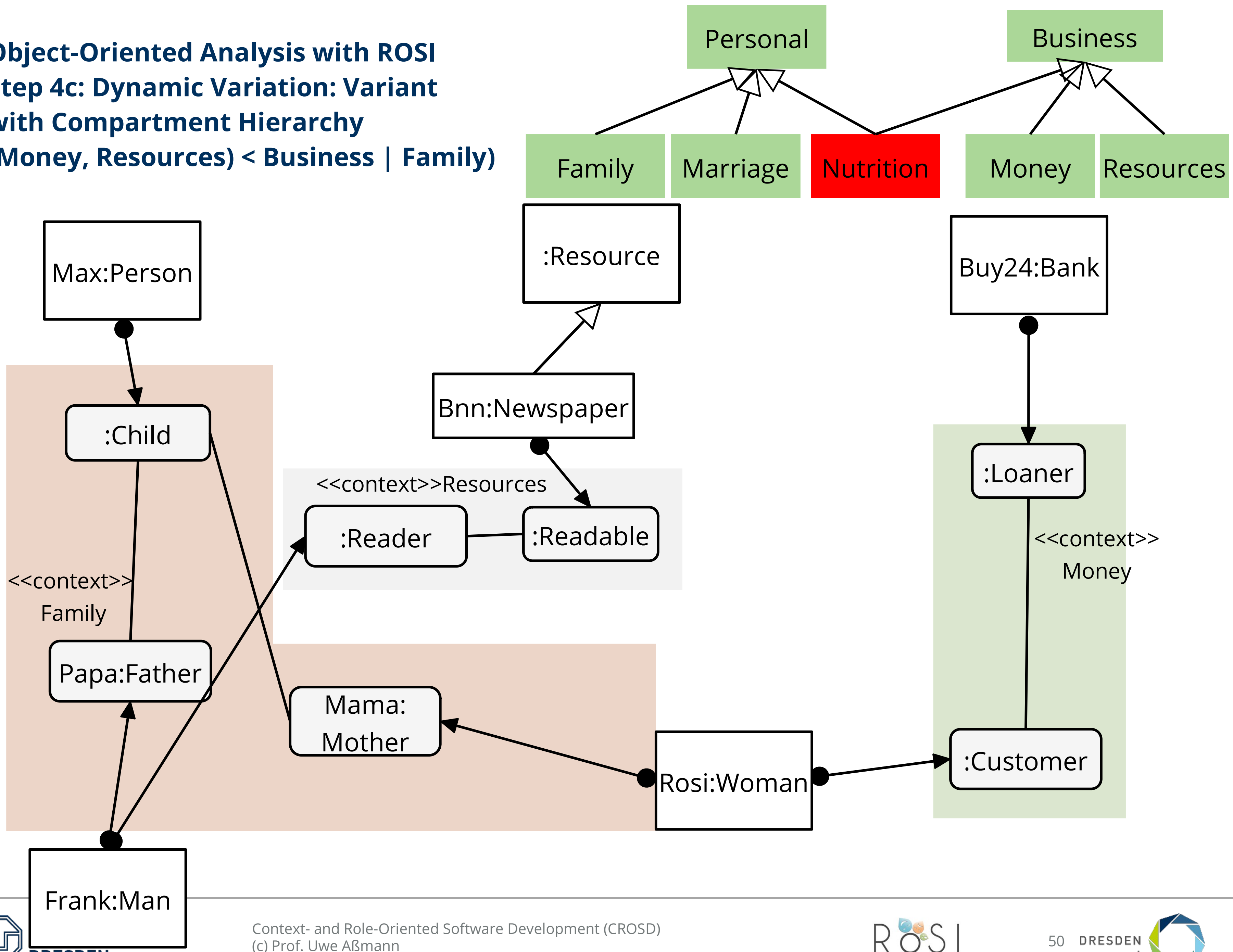
Object-Oriented Analysis with ROSI Step 4: Dynamic Variation: Variant with Contexts Family and Money



Object-Oriented Analysis with ROSI Step 4b: Dynamic Variation: Variant with Contexts Nutrition and Family

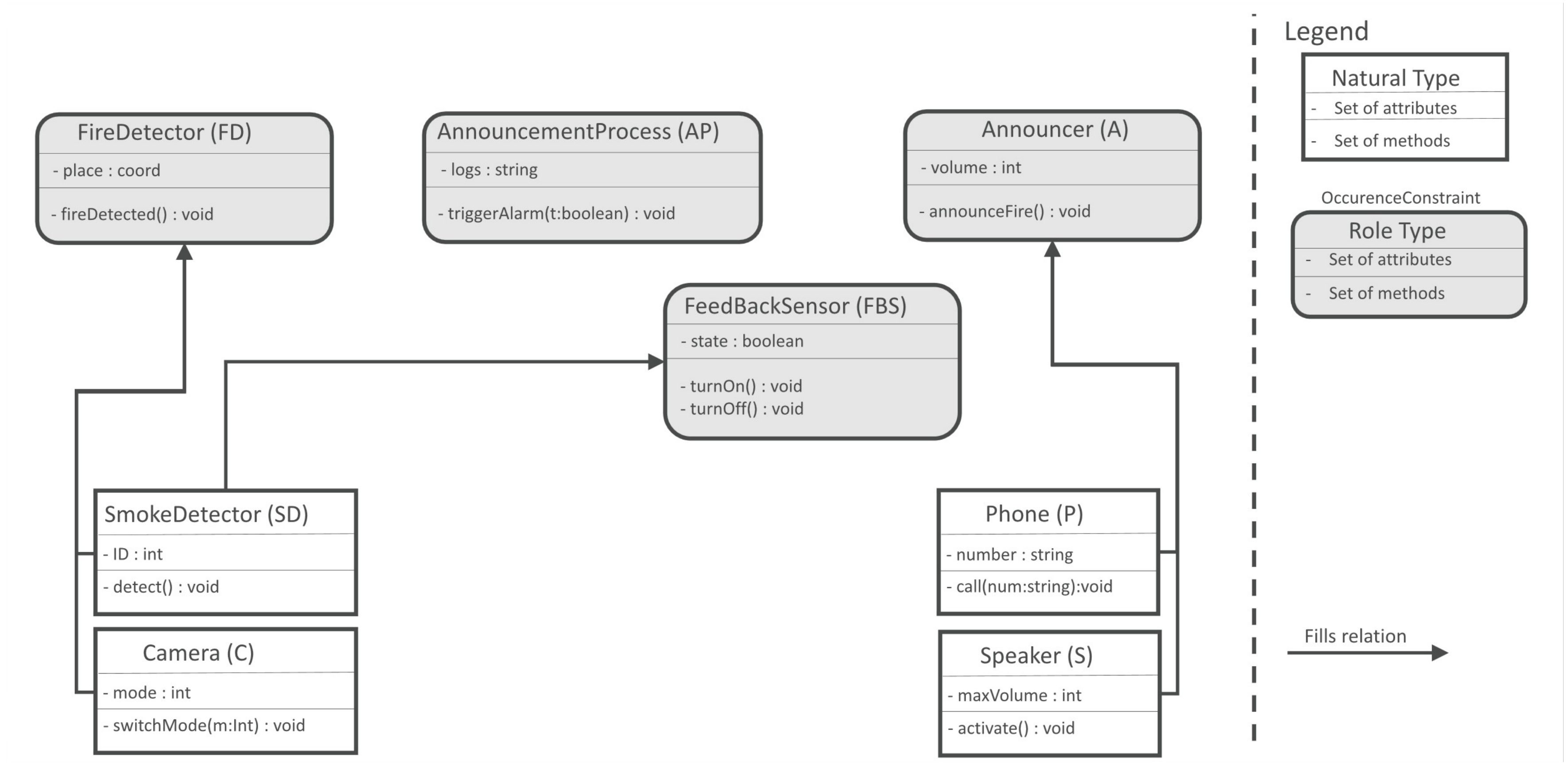


Object-Oriented Analysis with ROSI Step 4c: Dynamic Variation: Variant with Compartment Hierarchy (Money, Resources) < Business | Family)



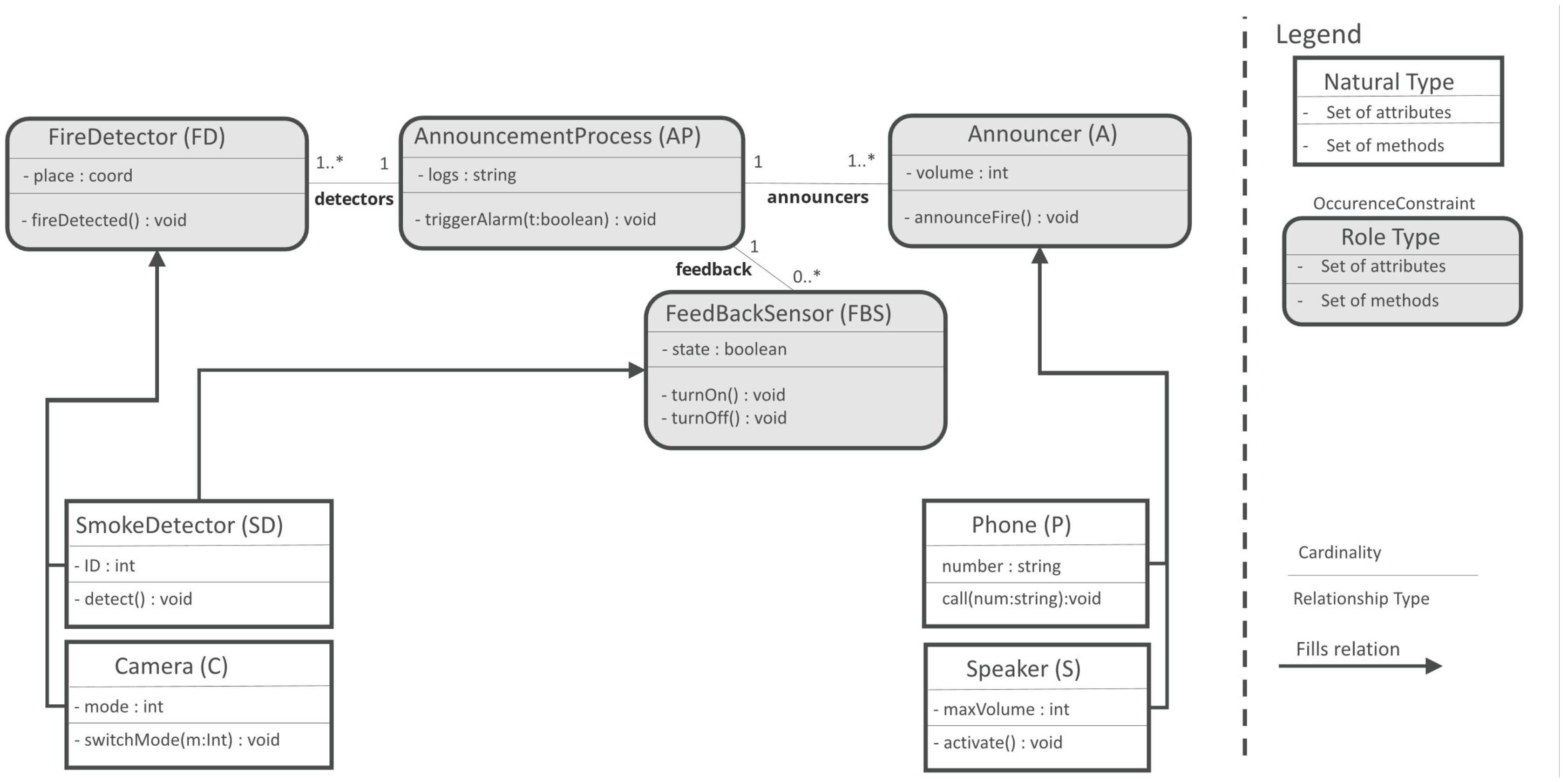
4.2. Scenario Fire Alarm – in the CROM Modeling Language

Context-Dependent Runtime Models
Compartment Role Object Model (CROM) *[Kühn2015]*



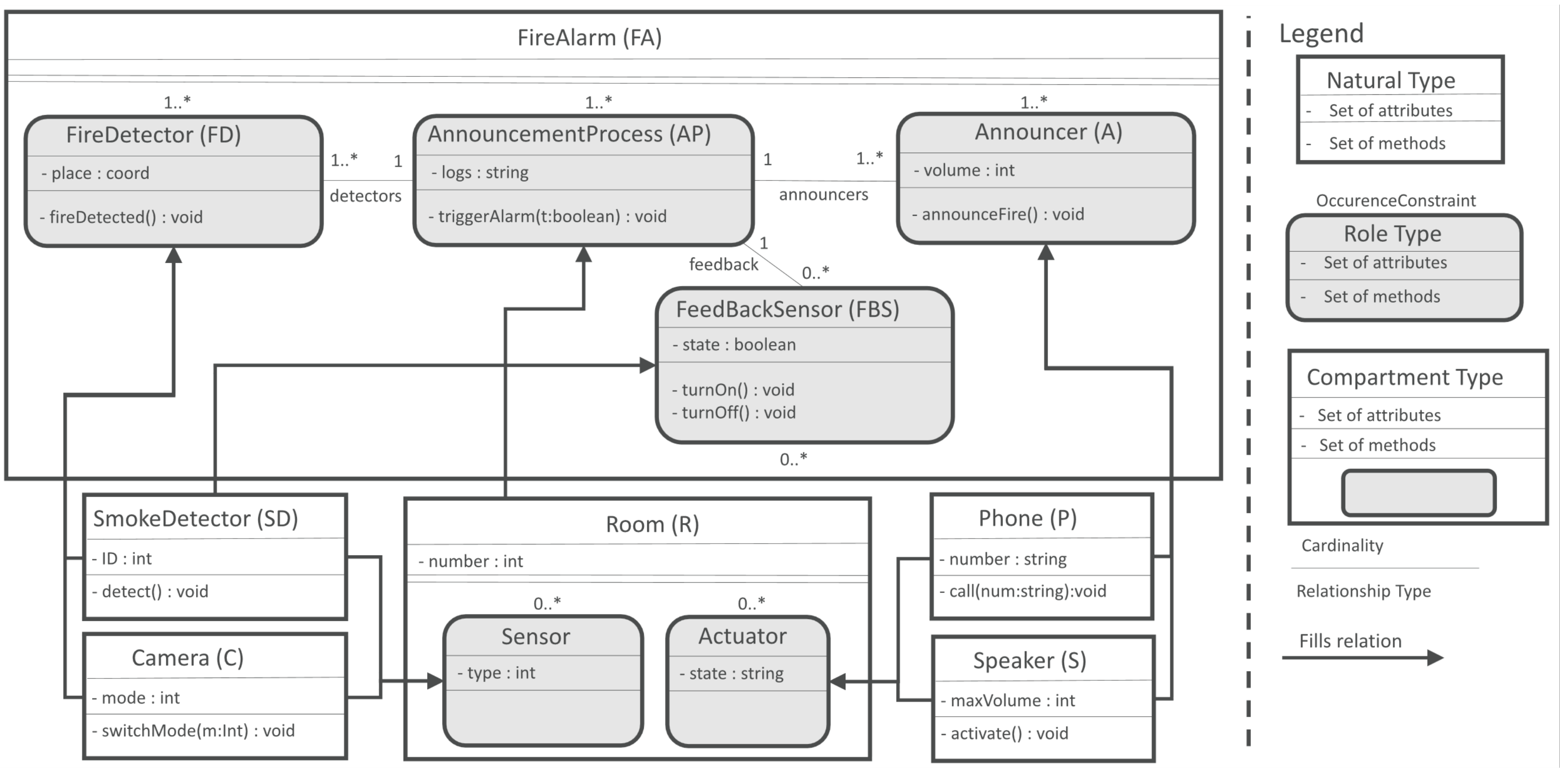
Context-Dependent Runtime Models

Compartment Role Object Model (CROM) *[Kühn2015]*



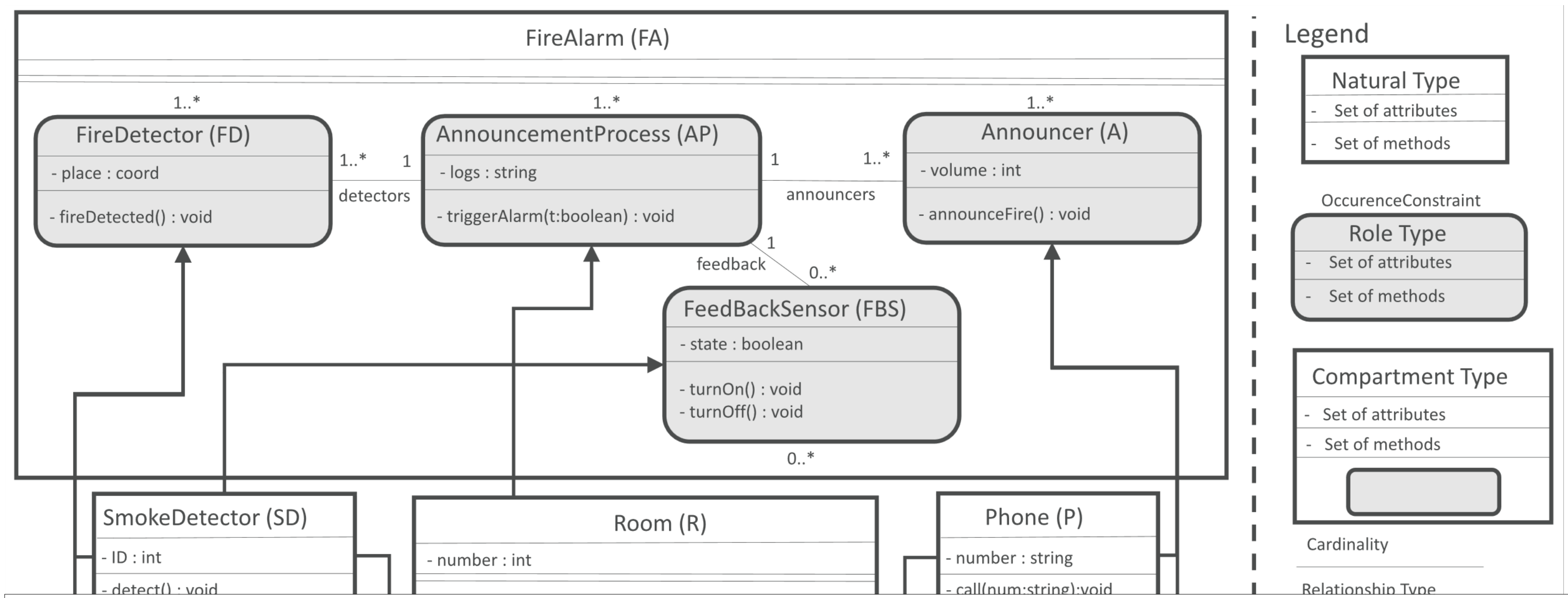
Context-Dependent Runtime Models

Compartment Role Object Model (CROM) *[Kühn2015]*



Context-Dependent Runtime Models

Compartment Role Object Model (CROM) *[Kühn2015]*

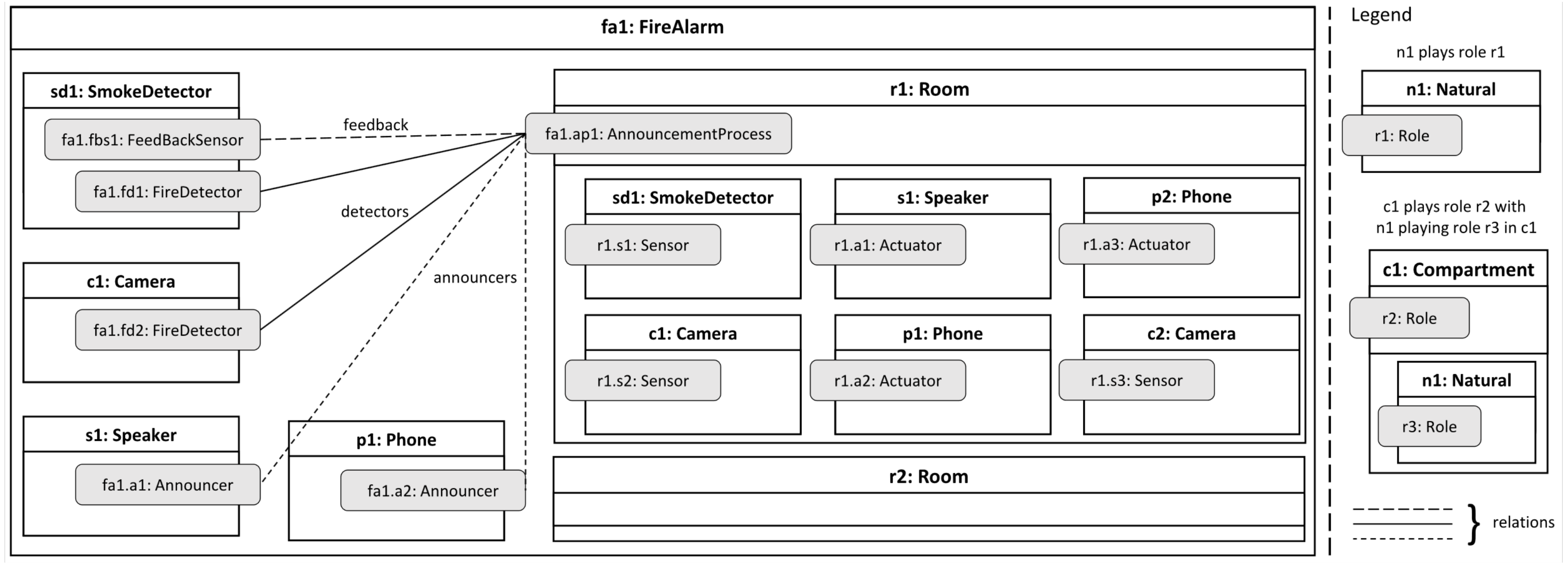


Key properties

- Roles and Relationships depend on the compartments (contexts)
- Roles change over time
- Compartments, “players” and roles have their own identity
- Formal definition of *well-formedness*, *compliance*, and *validity*

Context-Dependent Runtime Models

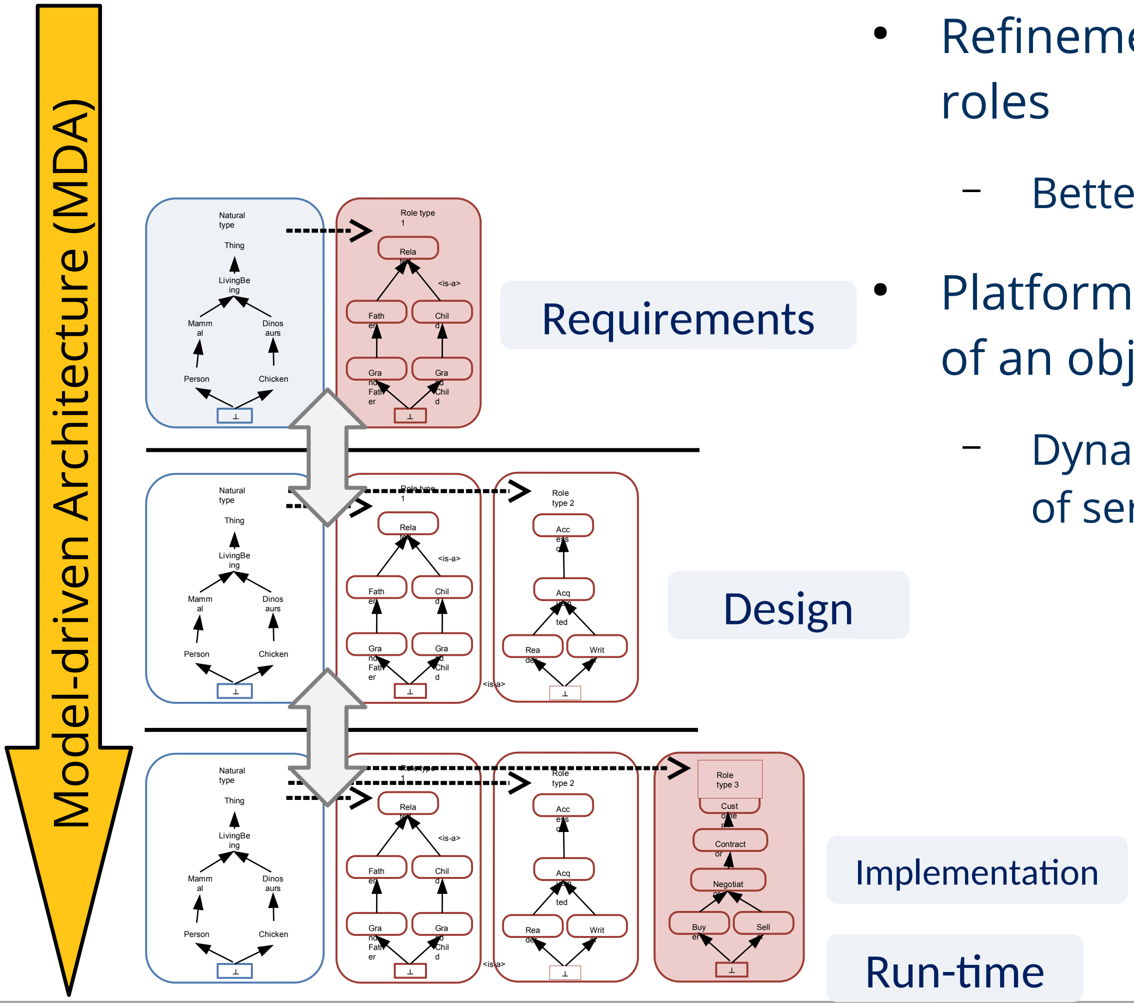
Compartment Role Object Instance (CROI) *[Kühn2015]*



Roles as a Concept Crosscutting all Phases

4.3.3 Role Refinement in Model-Driven Software Development (MDSD) and Model-Driven Architecture (MDA)

Role-based Refinement in the MDSD- and MDA-Process



- Refinement by allocation of additional roles
  - Better traceability
- Platform-features are „technical“ Roles of an object
  - Dynamic contexts (space, time, quality of service)

Causal connection of context-based features and fluidity from requirements level to run time

The Extended MDSD/MDA-Process with Contexts and Roles

Requirements Specification

- Domain models, tests

Platform-Independent Models

- Software architecture

Platform-Specific Models

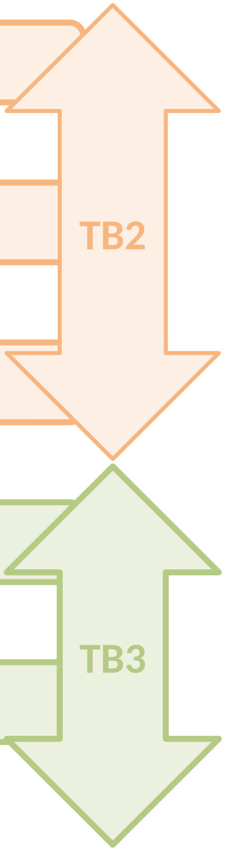
- Platforms correspond to technical contexts

Dynamic Context-Free Models (models at run time)

- Roles belong to run-time contexts

Dynamic Context-Sensitive Models

- Dynamic reconfiguration to new contexts by exchange of roles

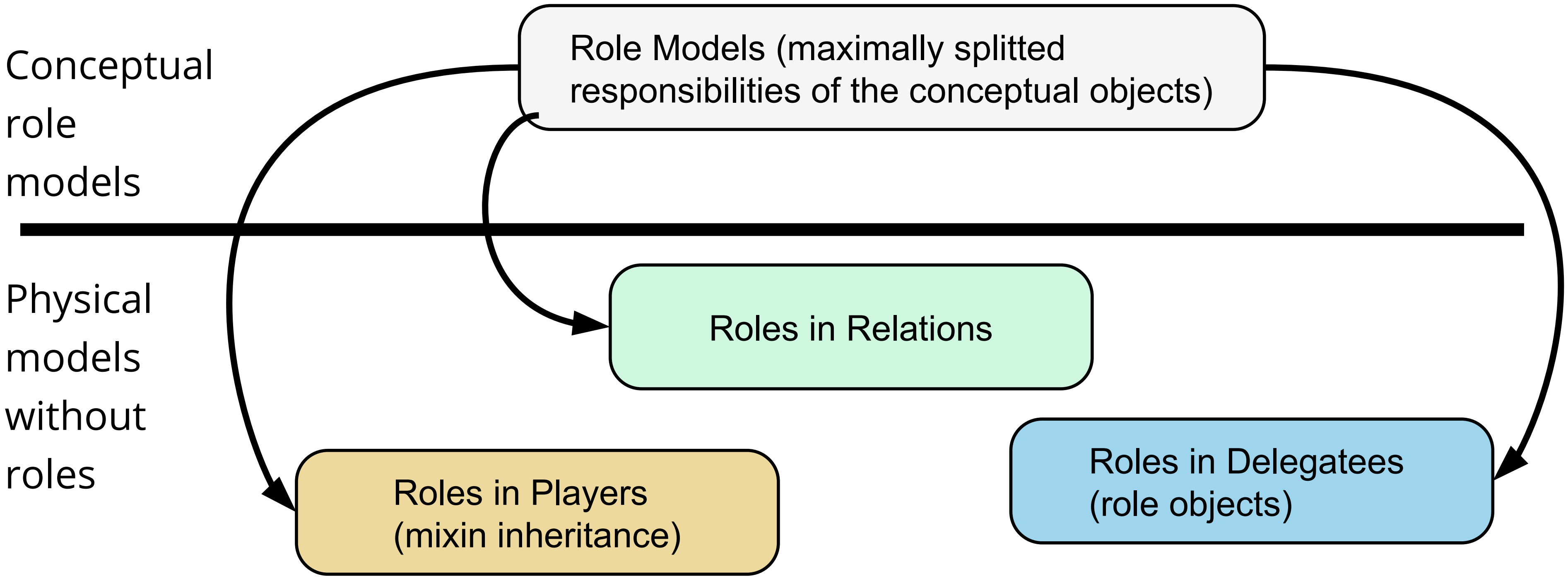


Good Mapping of Conceptual Role Models to Physical Class Models

- Role instances must be
  - embedded into core objects
  - or become physical role objects
- **Role mapping:** Mapping conceptual role types to physical implementation-records is an *Embedding Decision*
- For one conceptual model, many alternative phyisical models

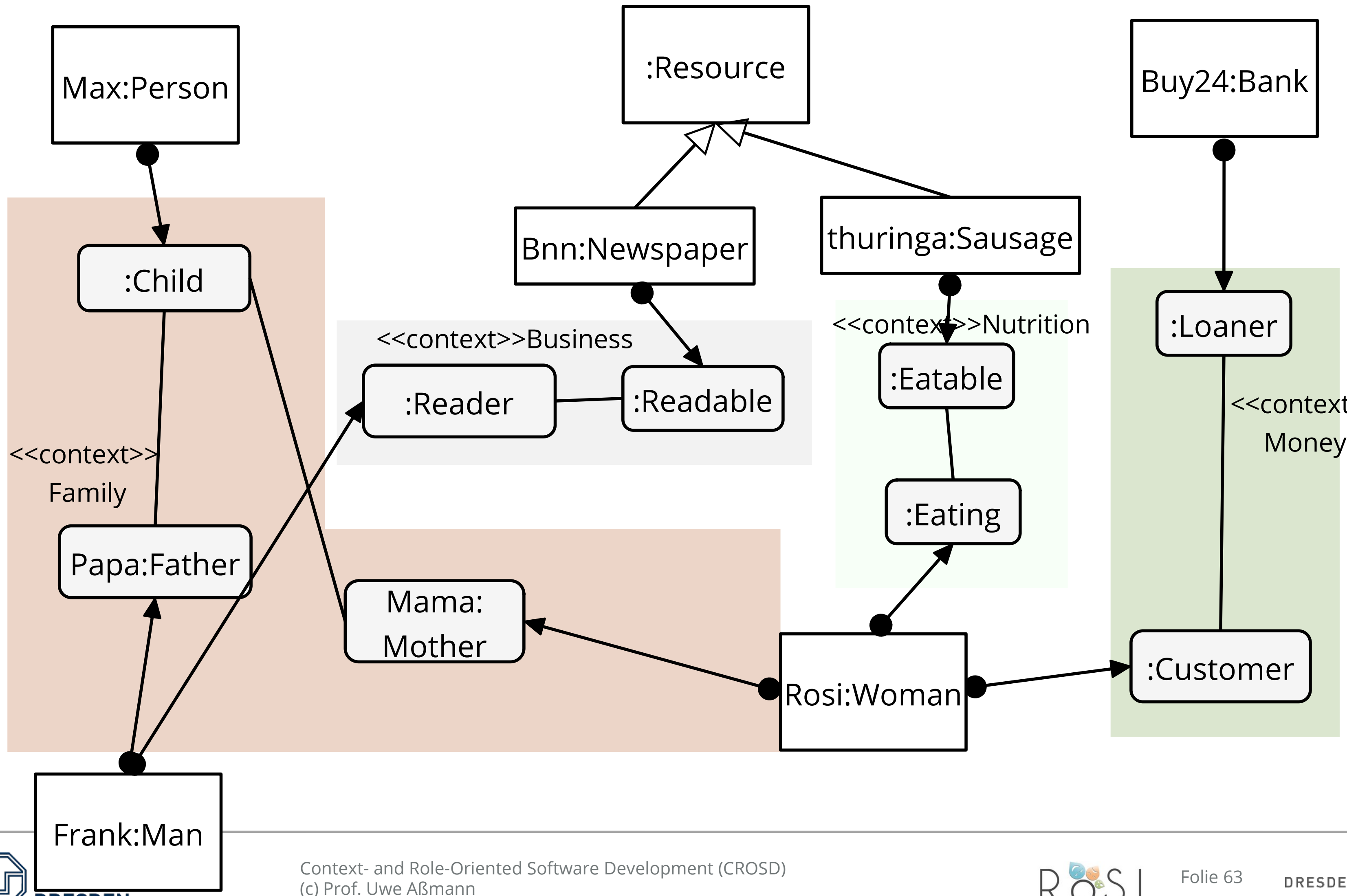
Computing Physical Representation from Conceptual Models

- Role embedding determines, which roles are embedded into which physical objects

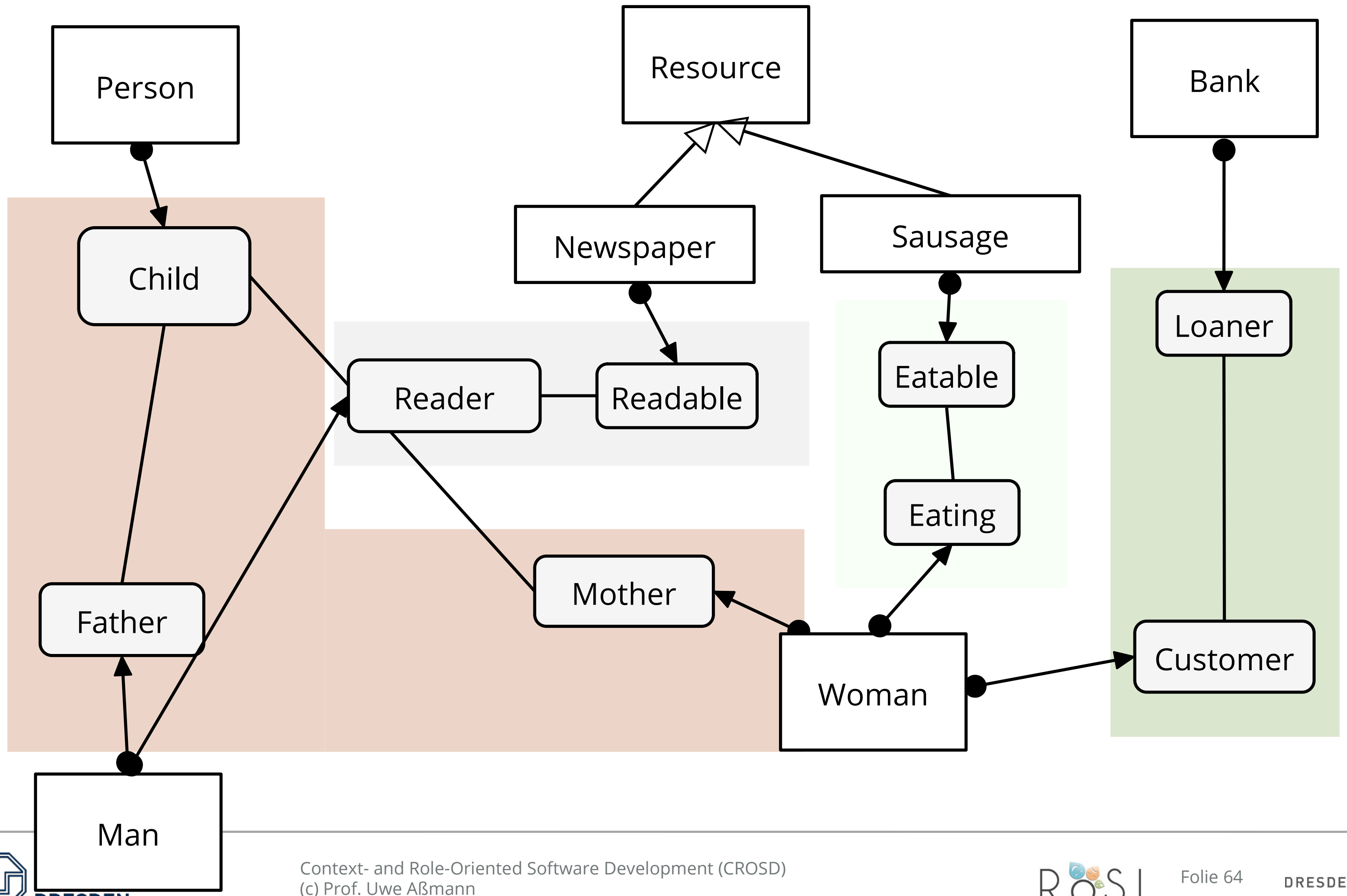

Conceptual role models
Role Models (maximally splitted responsibilities of the conceptual objects)
Physical models without roles
Roles in Relations
Roles in Delegatees (role objects)
Roles in Players (mixin inheritance)

4.5.3 Role-Mapping MDA with Scenario „Families and Banks“

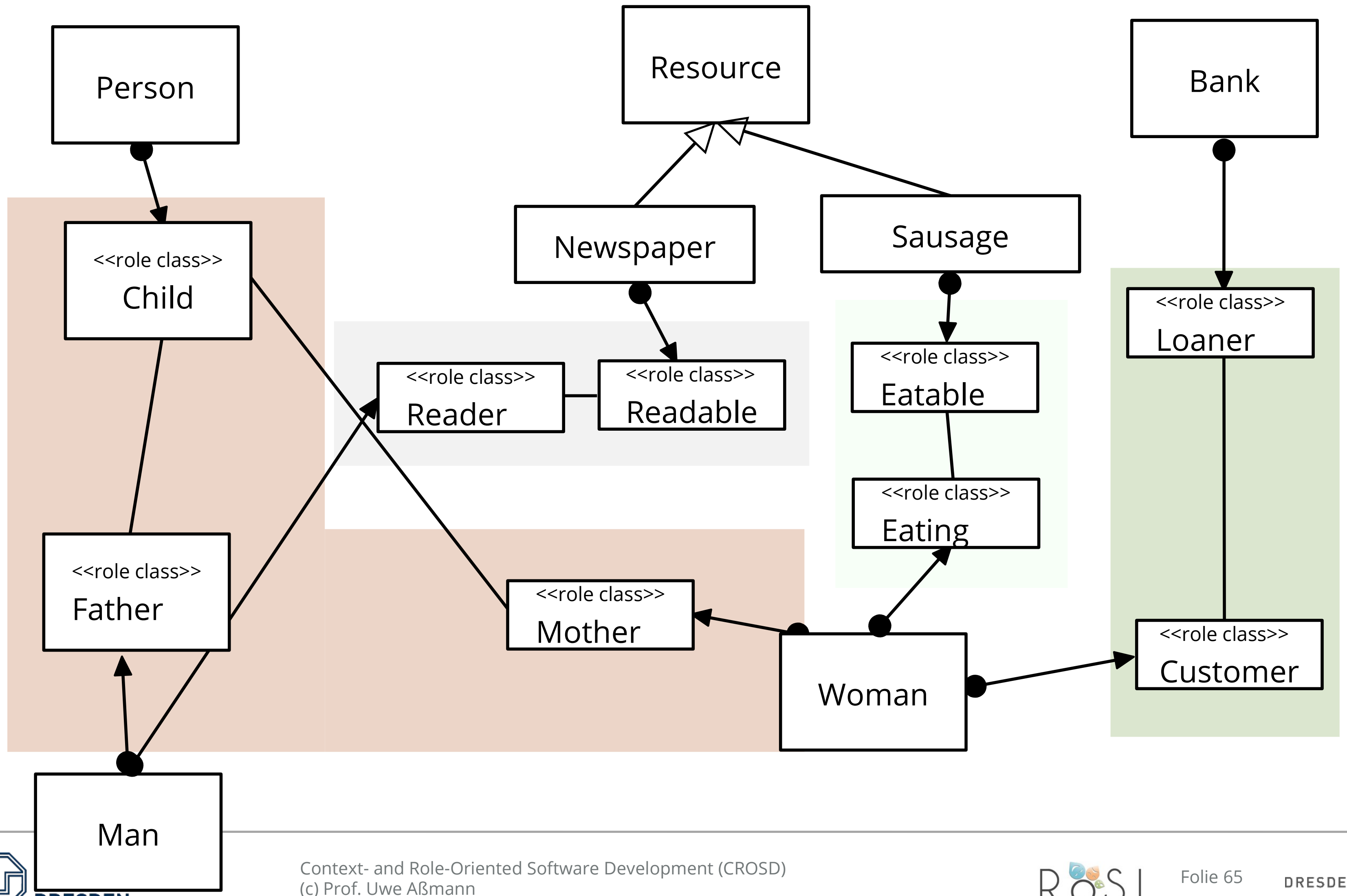
Families, Resources and Banks (Snapshot, Object-Role Model)



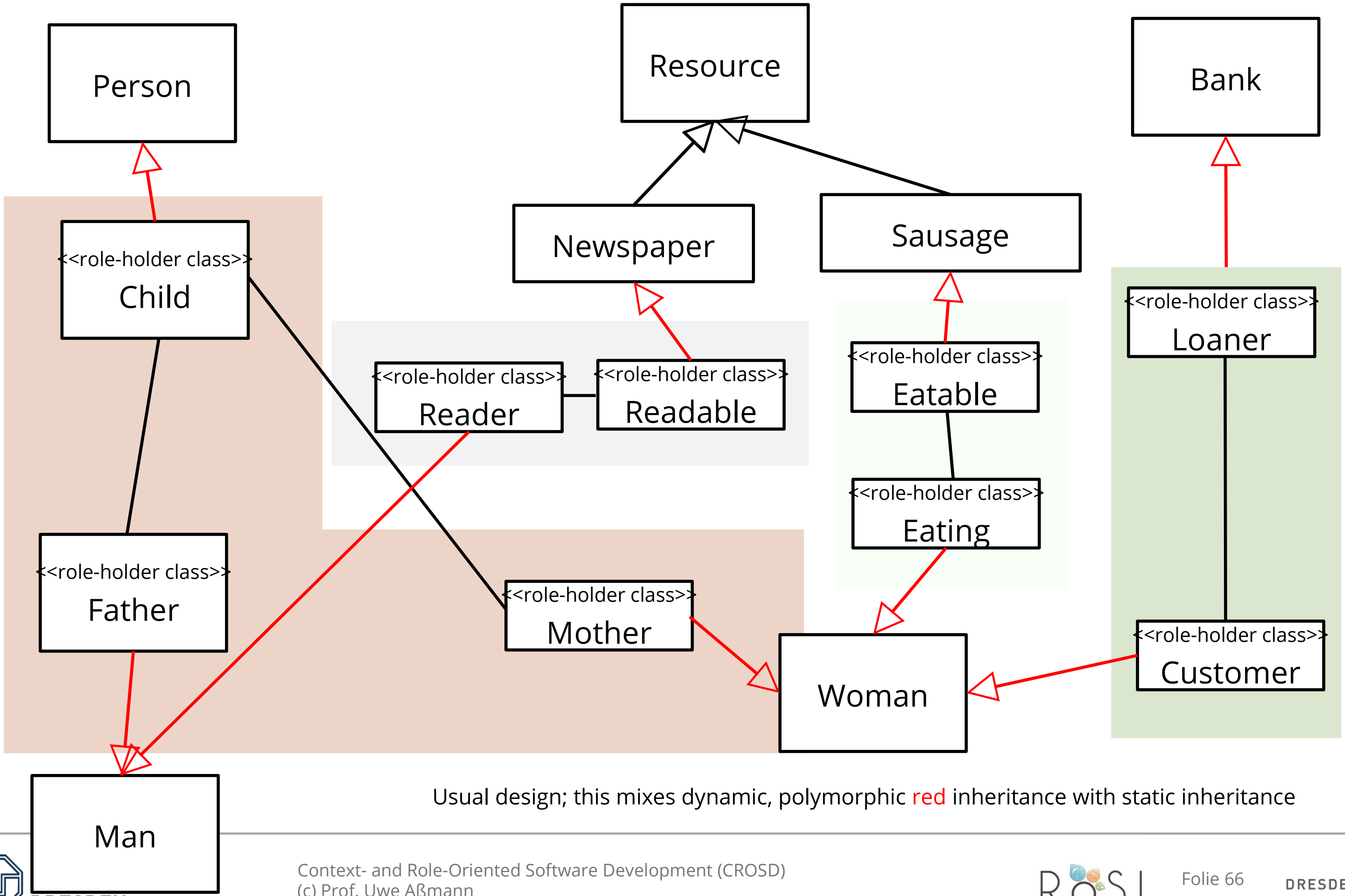
Families and Banks in Natural and Role Types



Implement „Families and Banks“ (Delegation to Role Objects - „Split Design“)



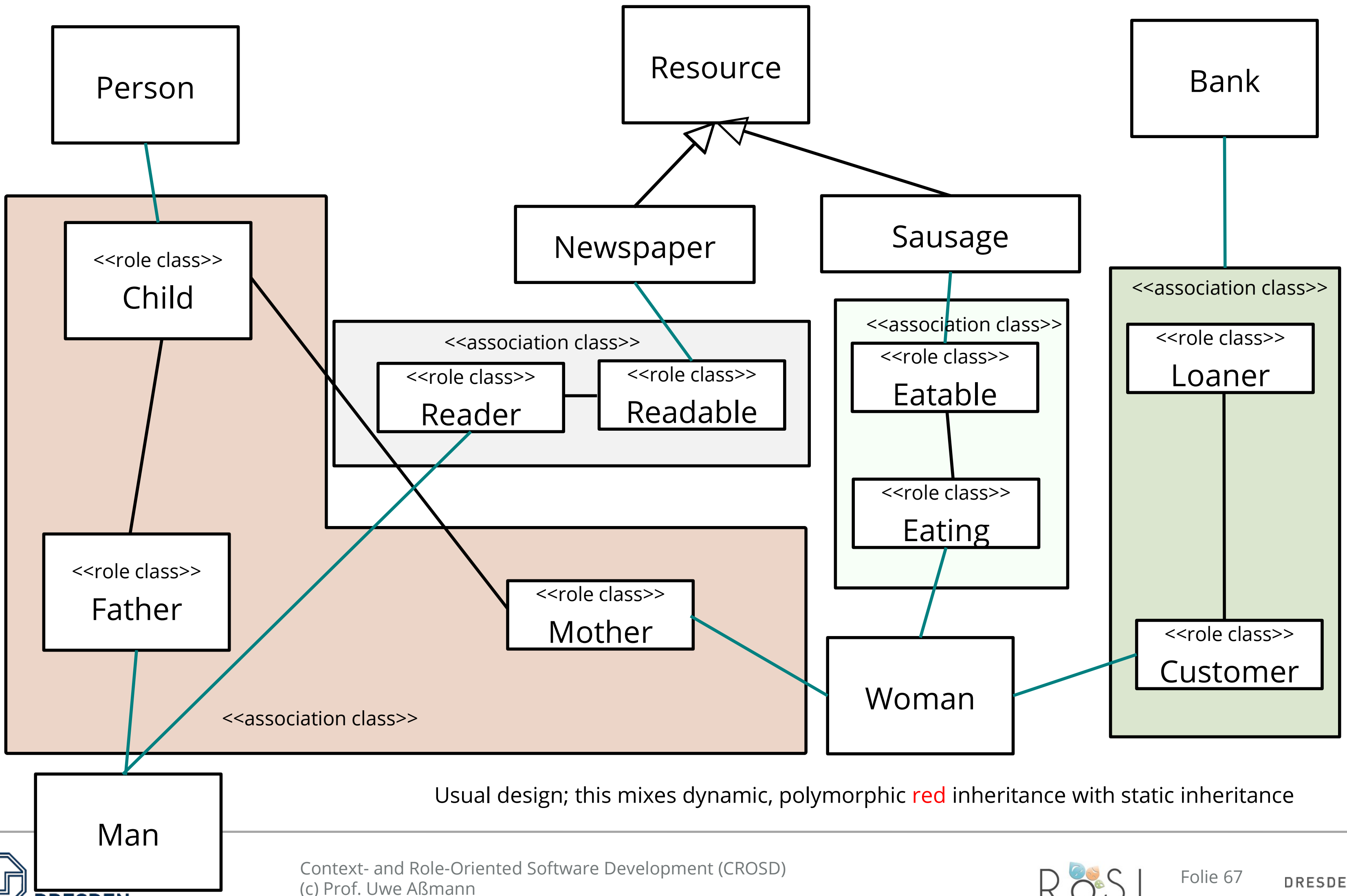
Implement „Families and Banks“ (Delegation to Role Objects – Design „Inheritance Embeds Roles in Players“)



Person
Resource
Bank
Newspaper
Sausage
<<role-holder class>> Child
<<role-holder class>> Reader
<<role-holder class>> Readable
<<role-holder class>> Eatable
<<role-holder class>> Loaner
<<role-holder class>> Eating
<<role-holder class>> Father
<<role-holder class>> Mother
<<role-holder class>> Customer
Woman
Man

Usual design; this mixes dynamic, polymorphic red inheritance with static inheritance

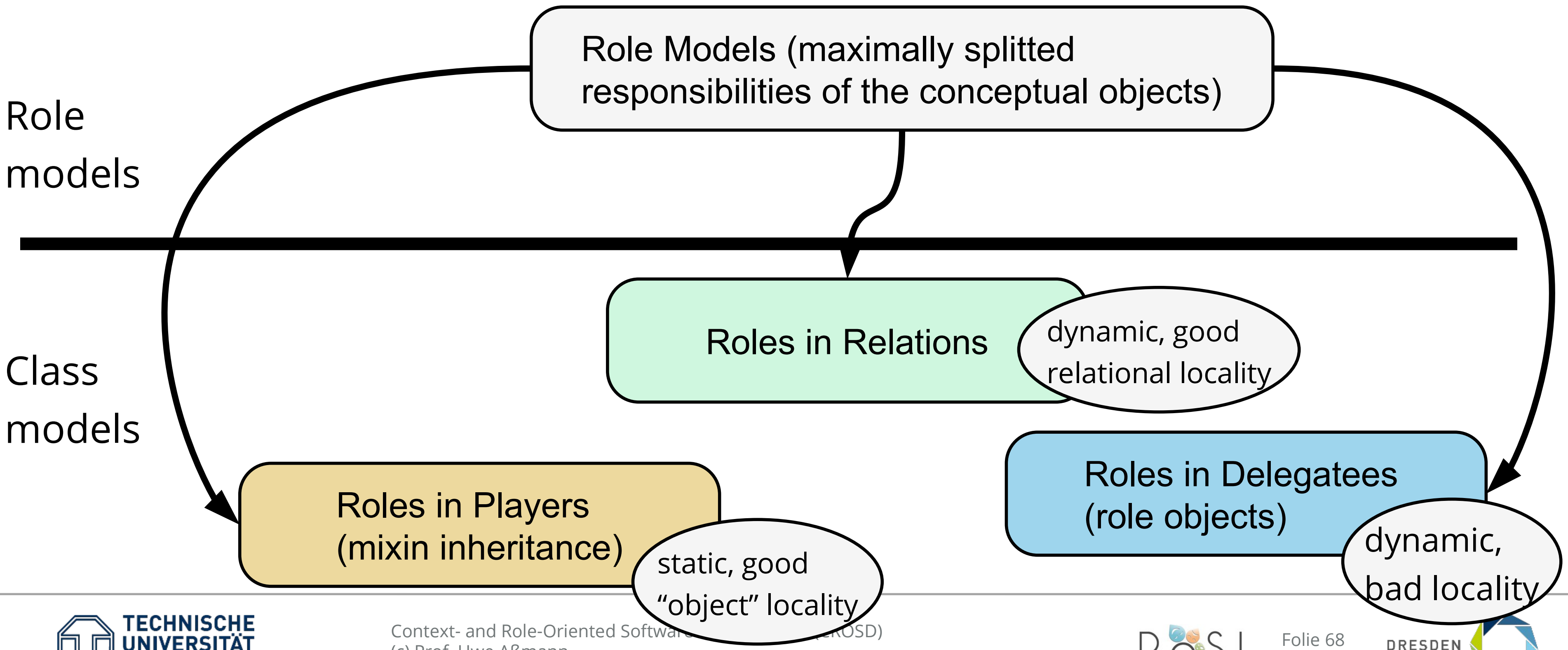
Implement „Families and Banks“ (Delegation to Role Objects – Design „Roles Embedded in Relations“)



Usual design; this mixes dynamic, polymorphic red inheritance with static inheritance

Scalable Binding Times of Contexts

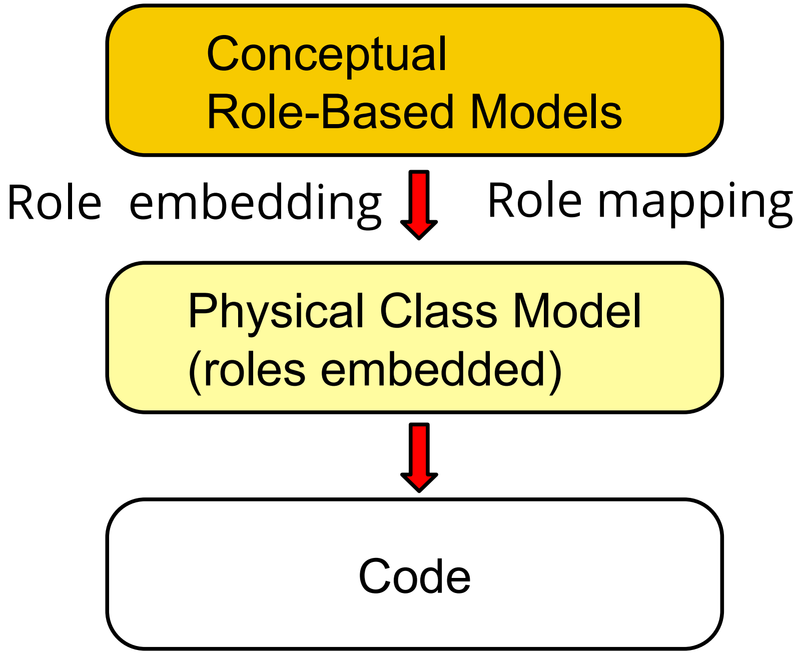
- Problematic: Role mapping fixes binding time


Role models
Role Models (maximally splitted responsibilities of the conceptual objects)
Class models
Roles in Relations
dynamic, good relational locality
Roles in Players (mixin inheritance)
static, good “object” locality
Roles in Delegatees (role objects)
dynamic, bad locality

The Role-Mapping Process and Model-Driven Architecture

- The question “Where is a role embedded?” is a *platform decision* in Model-Driven Architecture (MDA)
  - A role model is more *platform independent* than a class model

- → Role mapping is a task in Model-Driven Architecture (MDA)


Conceptual Role-Based Models
Role embedding
Role mapping
Physical Class Model (roles embedded)
Code

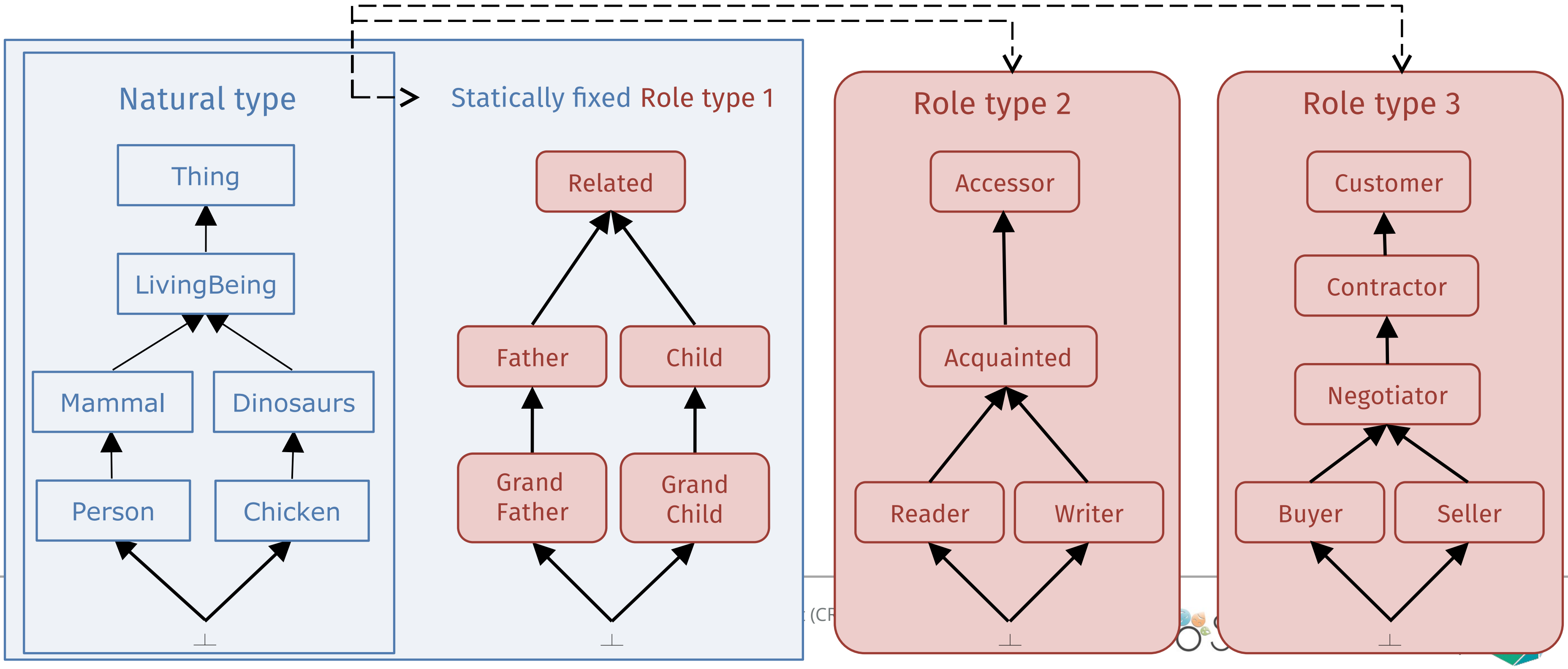
Role Mapping MDA Yields Scalability

- From one conceptual role-based design, derive via Role-MDA:
  - many physical designs
  - many run-time behaviors with different QoS
- When to embed?
  - At compile-time
  - At run-time
- Tuning and optimization possible

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

How to Achieve Scalable Binding Times of Contexts

- **Scalability**: Roles and their contexts can be statically bound
- Effects on Life-time, aliases and dependencies, cohesion, allocation, adaptation, reconfiguration



2.4. Roles are a Concept for Language Modeling and Language Engineering

The IRDS/MOF Metamodelling Hierarchy

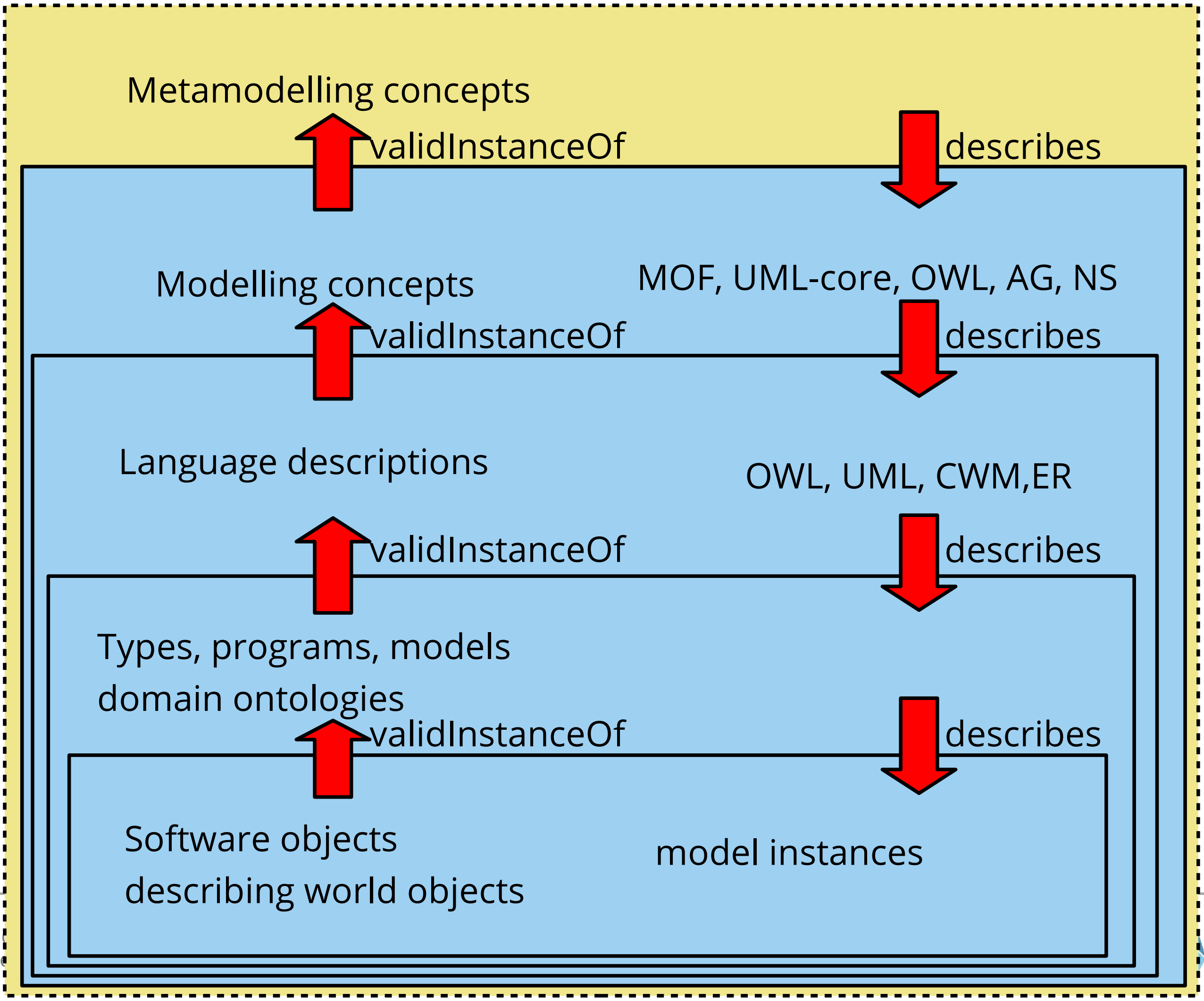
M4 level = M3

M3 metametamodel level

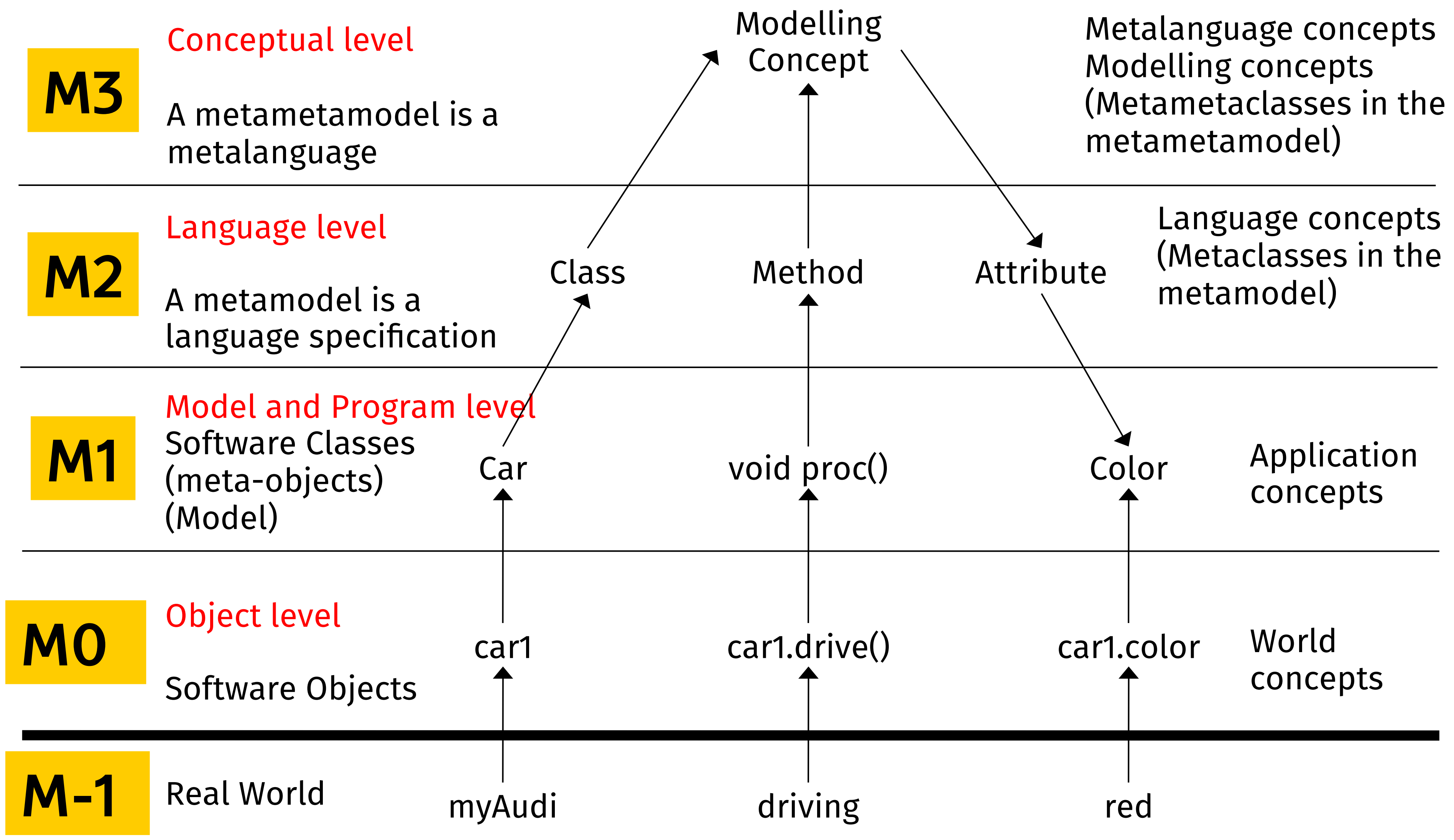
M2 metamodel level

M1 model level

M0 Object level

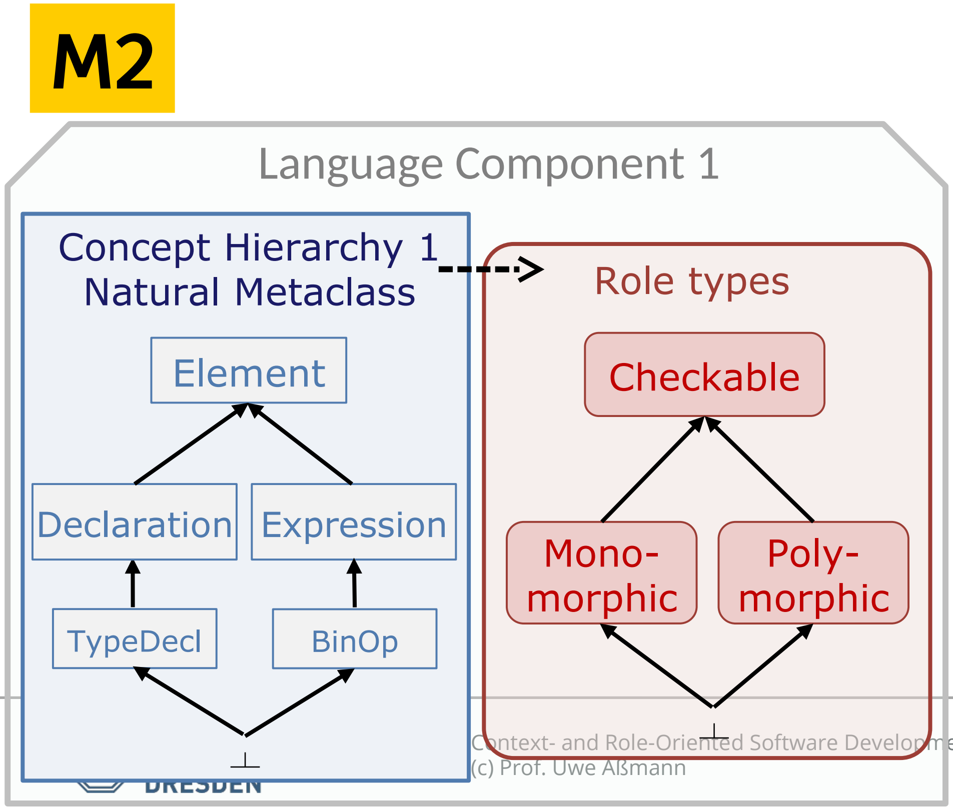


Metalevels in Programming Languages (The Meta-Pyramid)



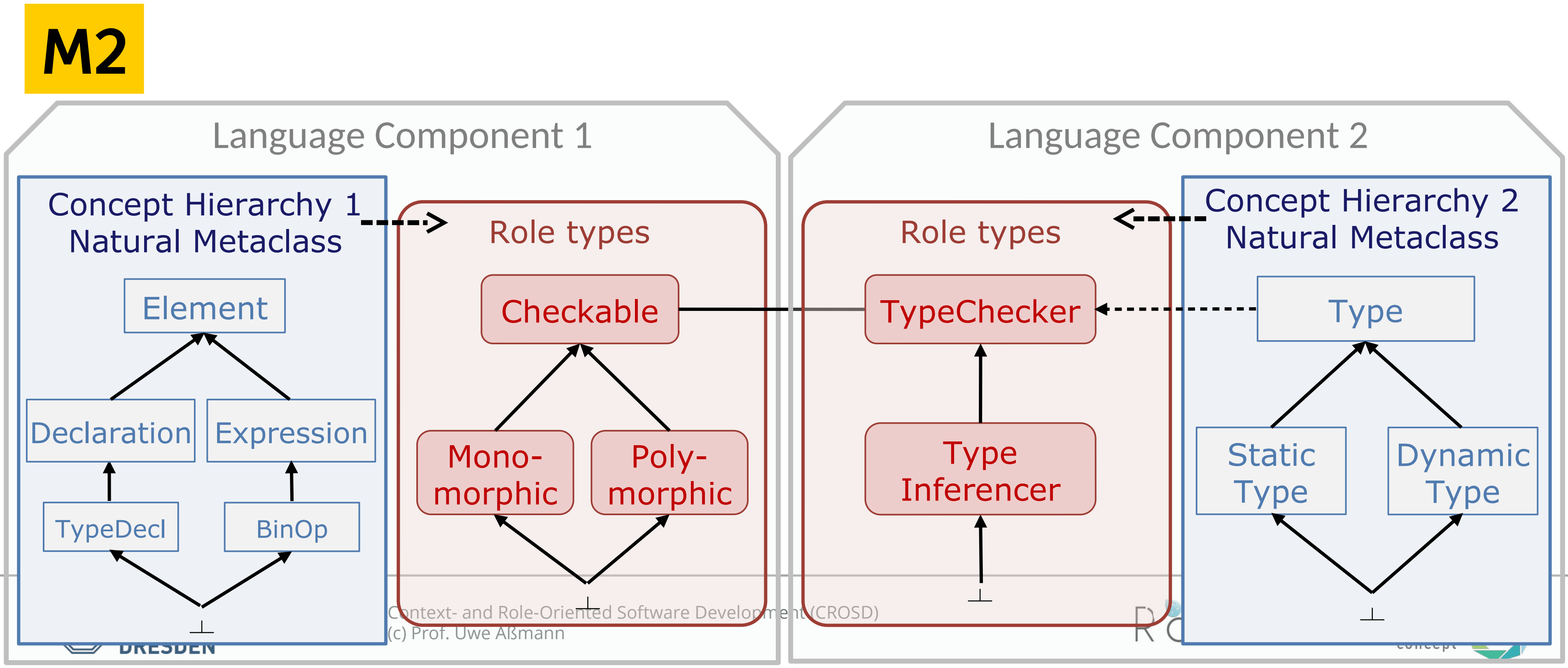
Context-Based Modelling of Languages on M2

- Role-types factor concept hierarchies into context-free and context-dependent features
- Improved separation of concerns
- [Wende] PhD Thesis



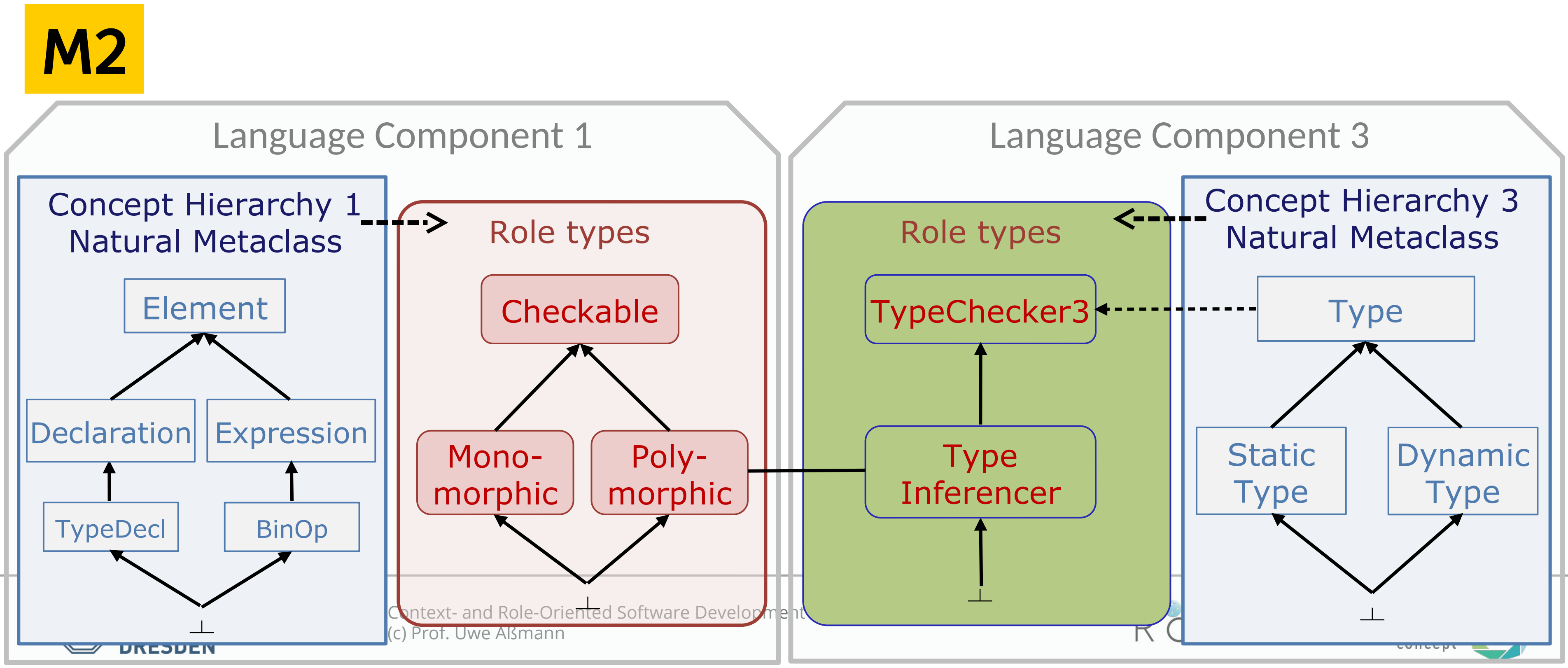
Context-Based Modelling of Languages on M2

- Context-dependent features can easily be exchanged



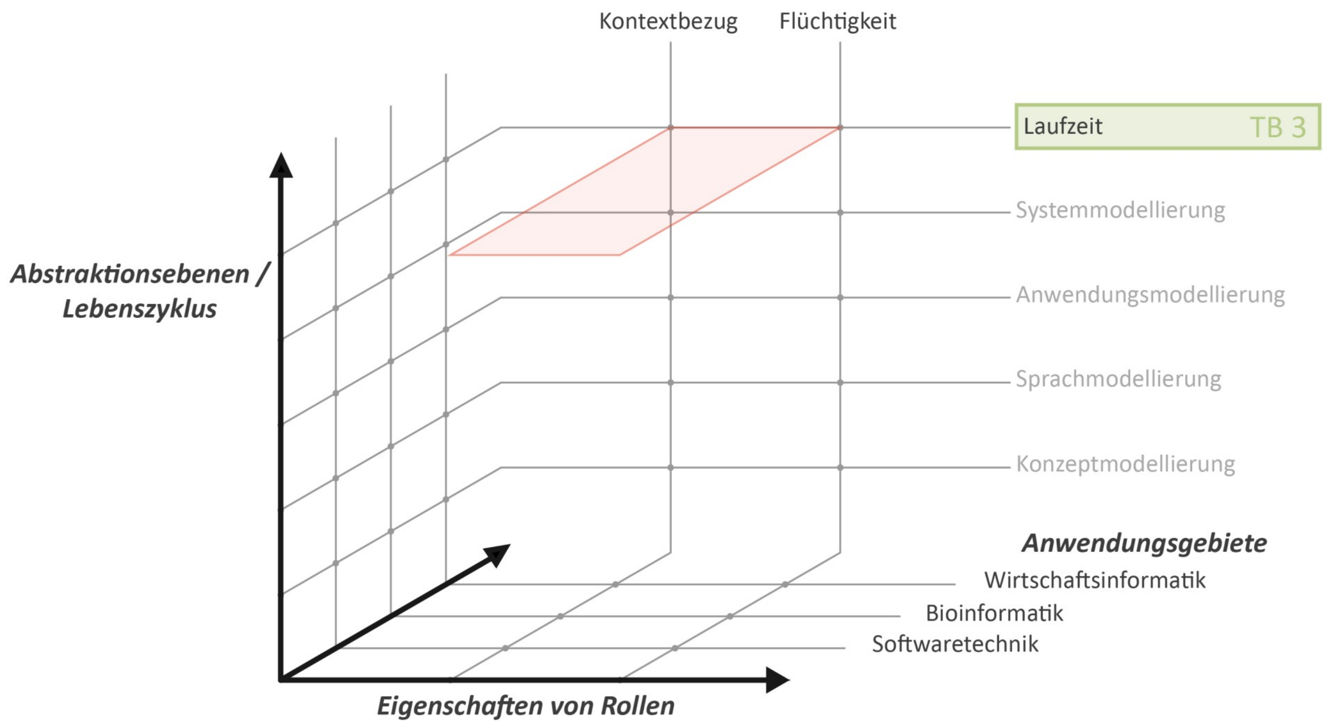
Context-Based Modelling of Languages on M2

- Modular languages
  - Domain-specific languages
  - Ontologies



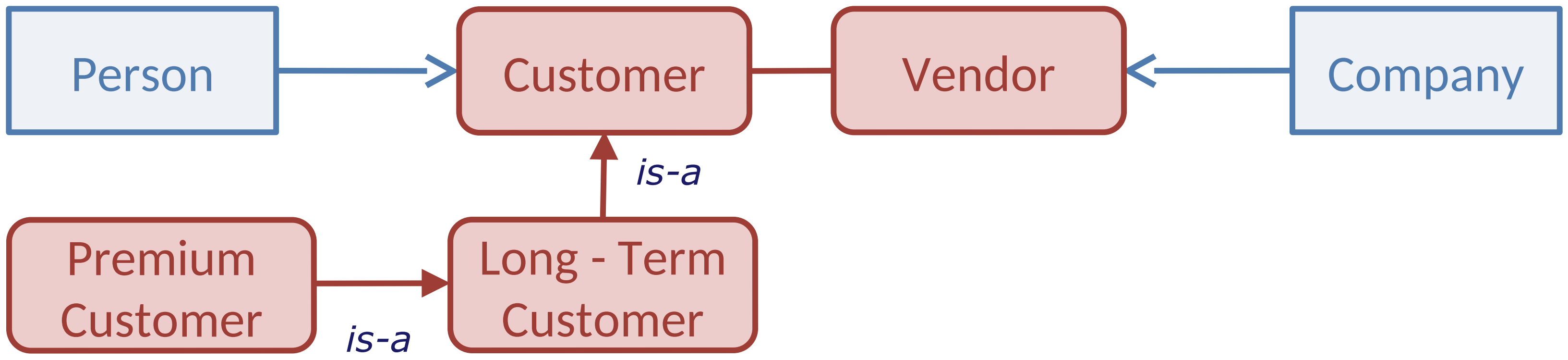
2.3.3 Roles are a Concept for Run-Time Infrastructures

Objective 3: Investigation of Context-Based and Fluid Run-Time-Infrastructures



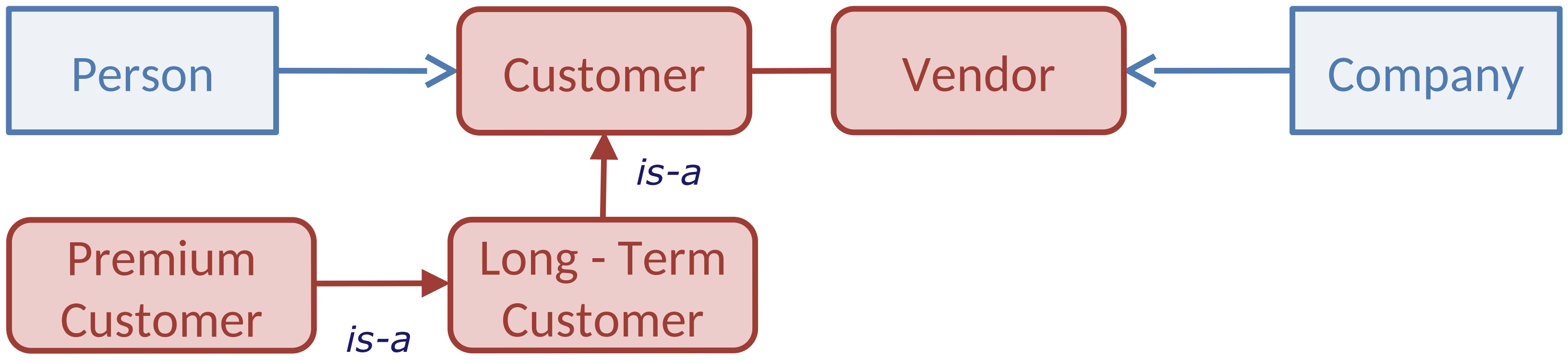
Context-Based and Fluid Run-Time Features

- Fluid complex objects can be dynamically reconfigured
- Context-dependent run-time behavior
- Fine-grained monitoring, persistency, adaption



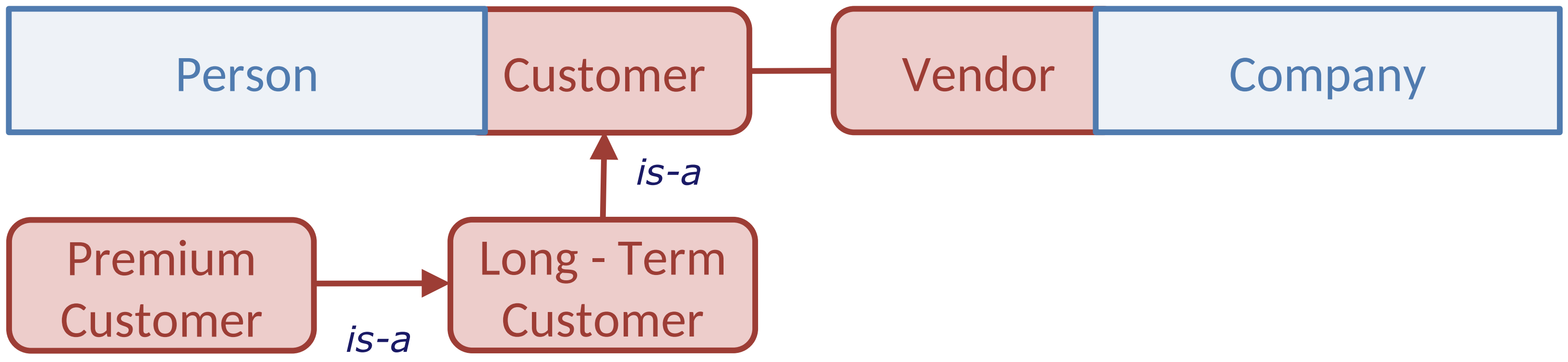
Dynamic Mixins

- Can role types be *mixed into* core types at run-time?



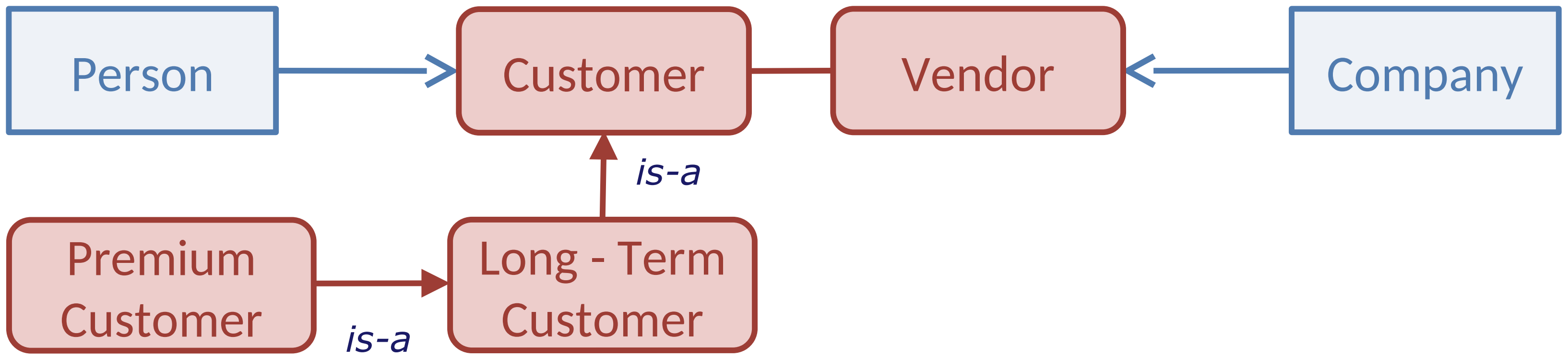
Dynamic Mixins

- Can role objects be *mixed into* core objects at run-time?
- Yes – by memory compaction in JIT recompilation



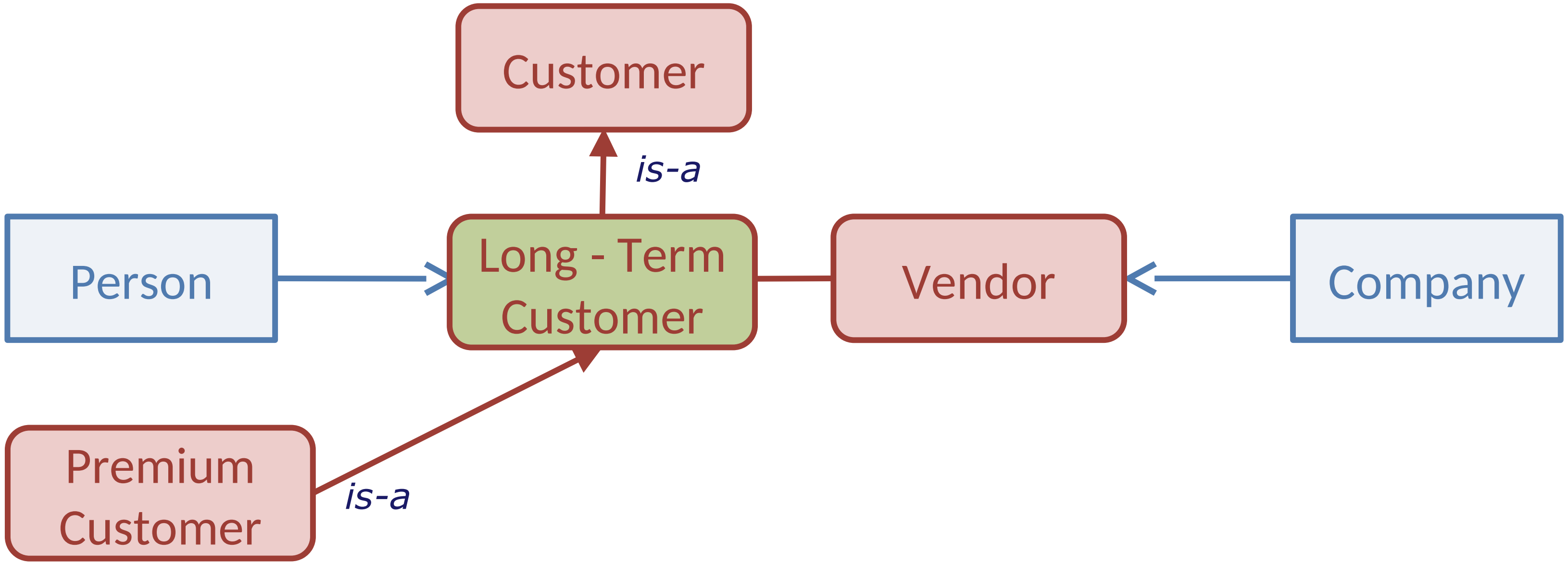
Dynamic Mixins

- But role instances can also be *outlined* again



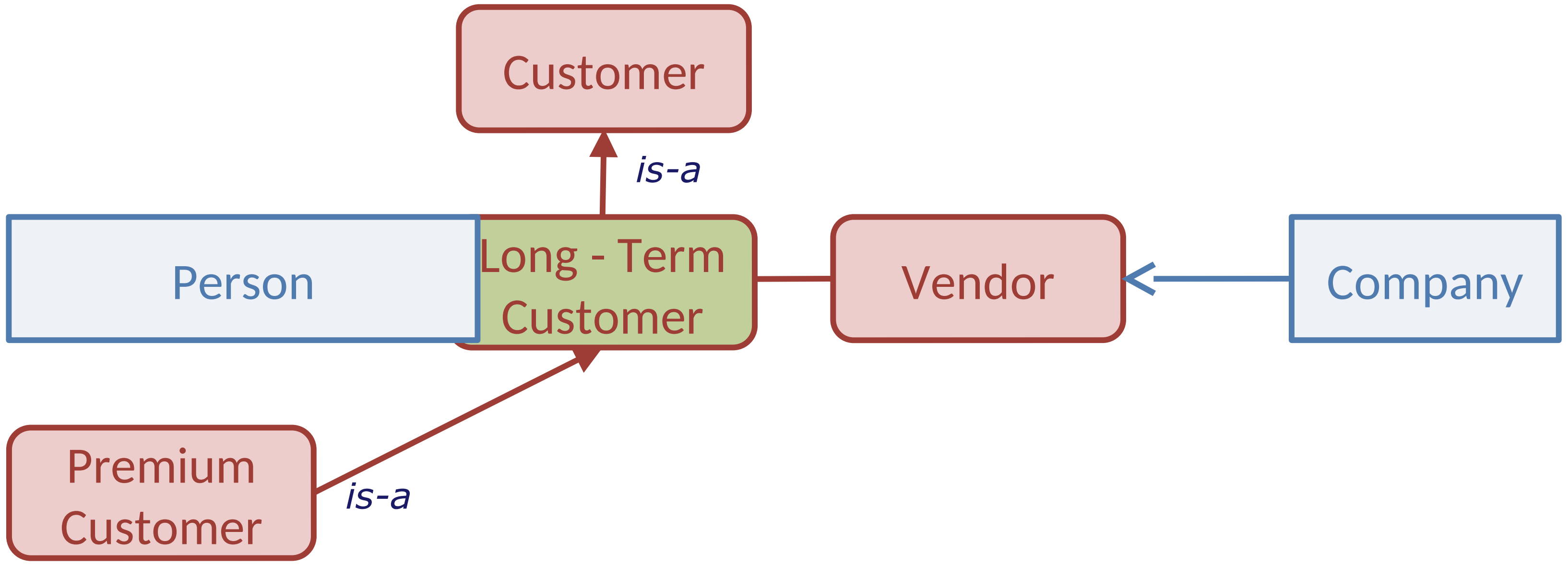
Dynamic Mixins

- But role instances can also be *outlined* again
- To change the role type



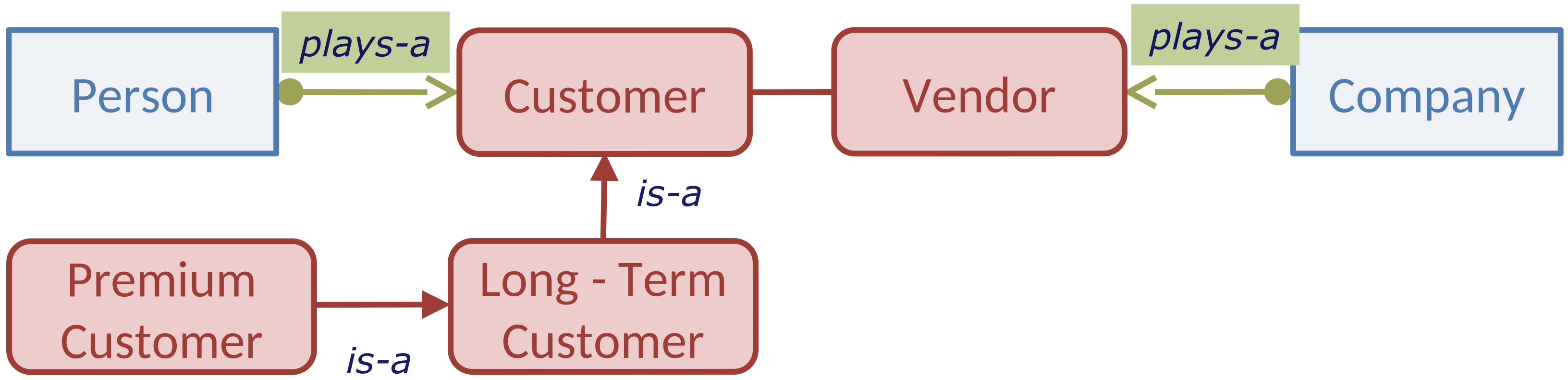
Dynamic Mixins

- And then re-inlined (dynamic mixin)
  - by memory compaction during JIT re-compilation



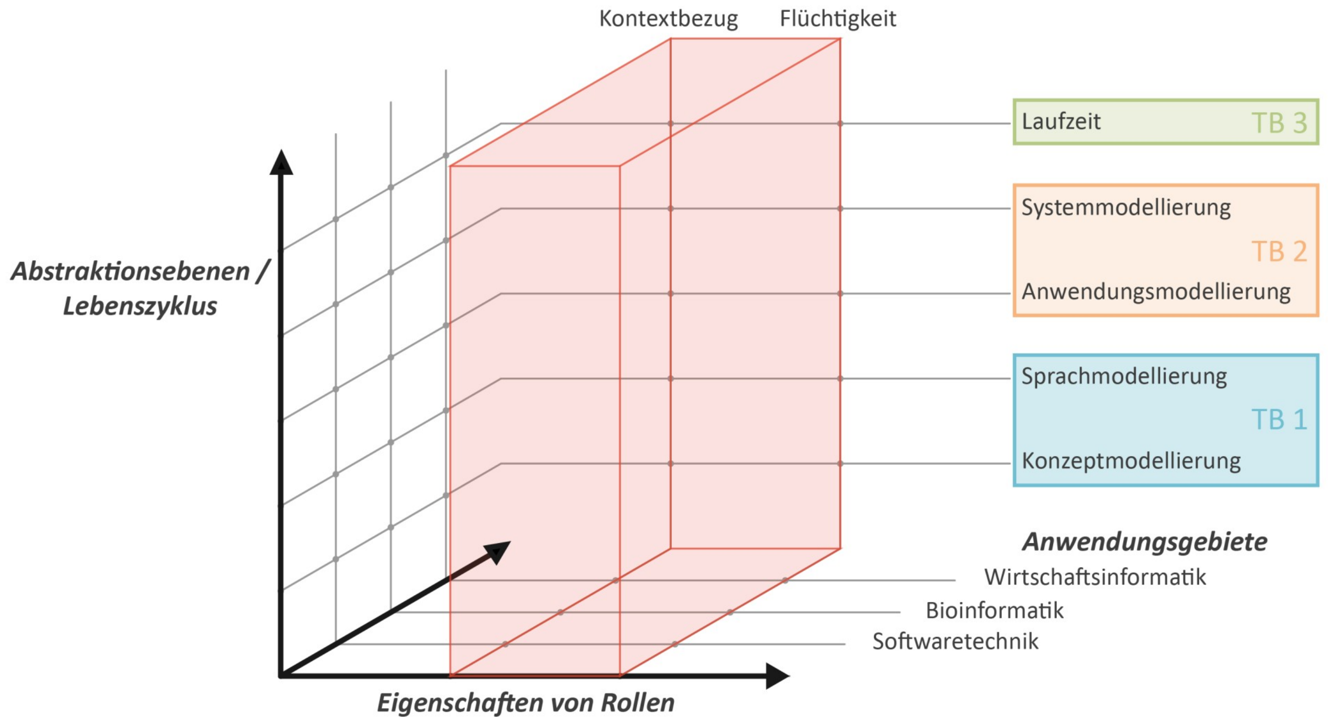
Dynamic Mixins

Role-based run-time infrastructures can optimize locality of roles dynamically
by dynamic mixins and recompilation



2.5. Roles are a Practical Concept

Objective 4: Practicality in Application Areas



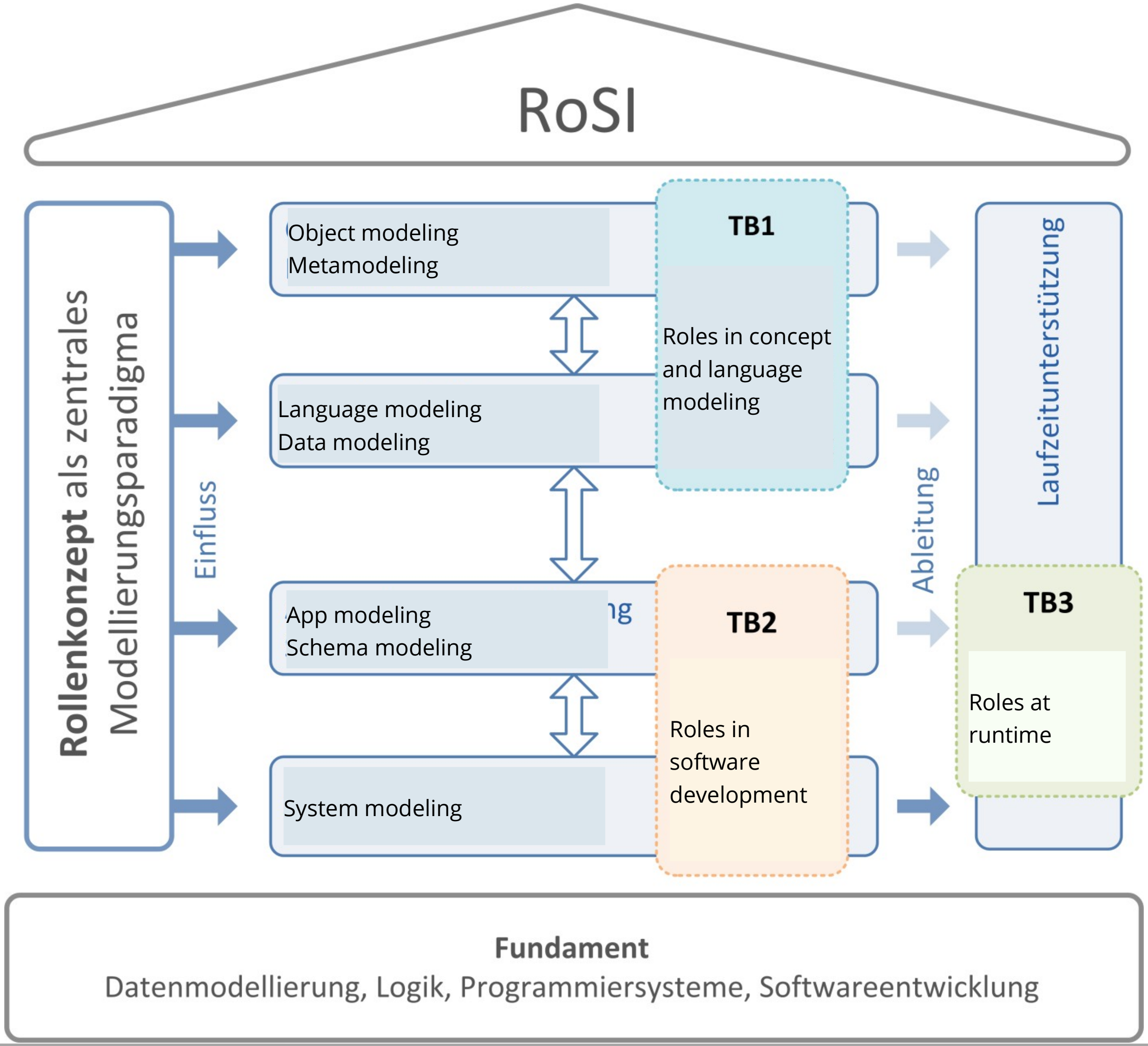
Practicality of Role Modeling

- Business Informatics (Wirtschaftsinformatik)
  - Improved Modeling of business objects and business models in ERP-systems
  - Role-based organisation models
- Bioinformatics (Bioinformatik)
  - Context-based dynamic biological processes
  - Search in context-based ontologies

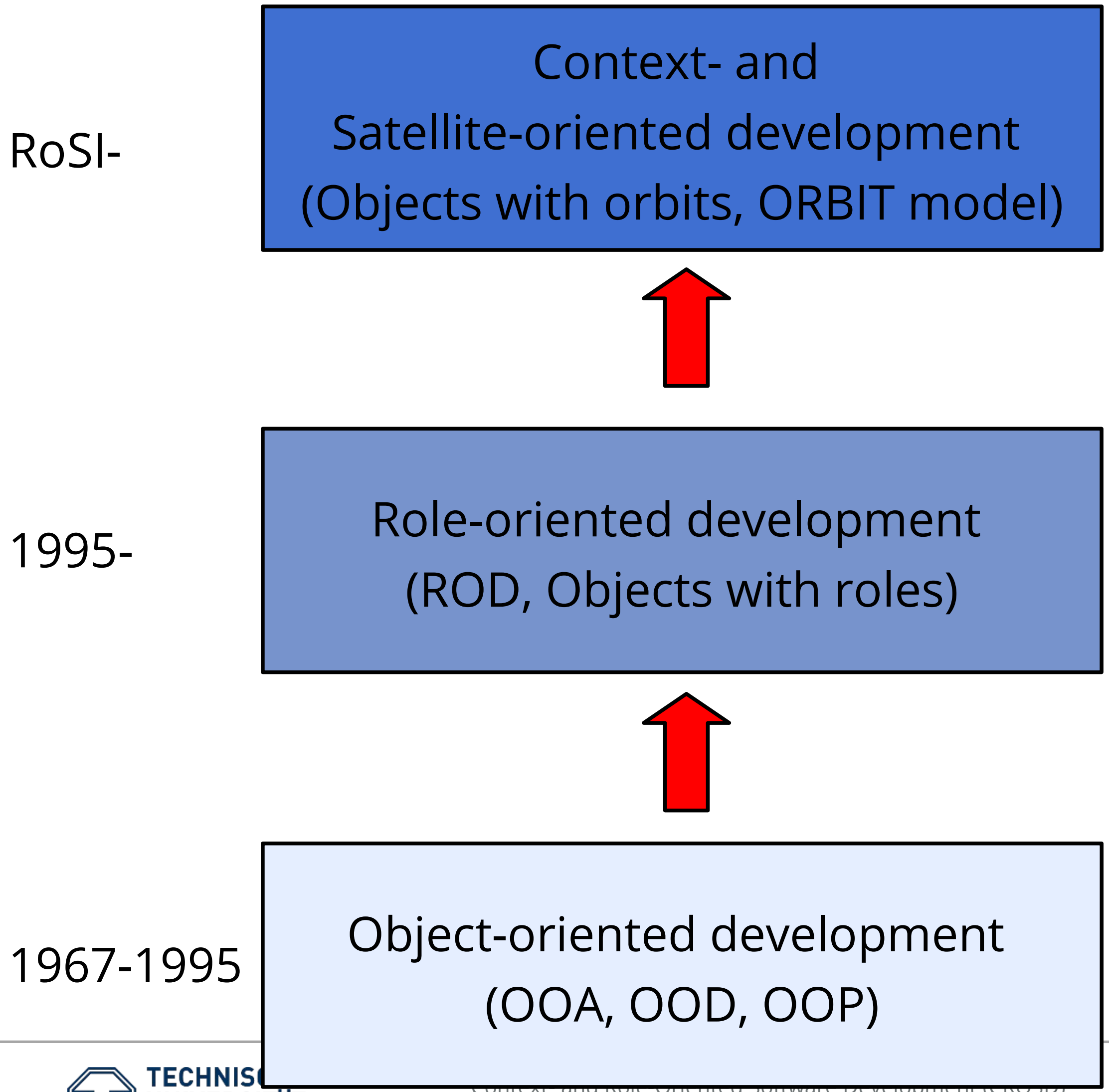
New Application Areas

- Roles for context-sensitive cyber-physical systems (CPS)
  - Hypothesis: Role-contracts for safety and security
- Roles for emergence in Systems-of-Systems (SoS)
  - Hypothesis: Role models for unforeseen emergence
- Roles for Natural Energy Servers
  - Hypothesis: Multi-criteria optimization for energy-adaptive systems

The RoSI House


RoSI
Rollenkonzept als zentrales Modellierungsparadigma
Einfluss
Object modeling
Metamodeling
Language modeling
Data modeling
App modeling
Schema modeling
ng
System modeling
TB1
Roles in concept and language modeling
TB2
Roles in software development
Ableitung
Laufzeitunterstützung
TB3
Roles at runtime
Fundament
Datenmodellierung, Logik, Programmiersysteme, Softwareentwicklung

Ladder of Paradigms (ctd)


RoSI-
Context- and Satellite-oriented development (Objects with orbits, ORBIT model)
1995-
Role-oriented development (ROD, Objects with roles)
1967-1995
Object-oriented development (OOA, OOD, OOP)

E. W. Dijkstra “On the Role of Scientific Thought”, EWD 447 Selected Writings on Computing: A Personal Perspective, pages 60–66, 1982.

"Let me try to explain to you, what to my taste is *characteristic for all intelligent thinking*.

It is, that one is willing to study in depth an aspect of one's subject matter in isolation for the sake of its own consistency, all the time knowing that one is occupying oneself only with one of the aspects.

The End

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.
- 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.
- Nicola Guarino Chris Welty. Supporting ontological analysis of taxonomic relationships. Data and Knowledge Engineering, 39:51-74, 2001.
- 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

Important References for Role-Based Modeling

- 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.
- 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.

Role-Based Programming

- 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, 2007.
- www.objectteams.org: a Java-based programming language with roles

Works at SMT

AOSD, MDD:

- U. Aßmann, S. Zschaler, and G. Wagner. Ontologies, Meta-Models, and the Model-Driven Paradigm, Handbook on Ontologies and Software Engineering. pages 249–273. Springer, 2006.
- J. Henriksson, J. Johannes, S. Zschaler, U. Aßmann. Reuseware – adding modularity to your language of choice. Proc. of TOOLS EUROPE 2007: Spec Iss Journal of Object Technology, 2007.

Roles and aspects in ontologies and metamodeling:

- 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.
- Matthias Bräuer and Henrik Lochmann. Towards Semantic Integration of Multiple Domain-Specific Languages Using Ontological Foundations.

Works at PhD Theses ST (all available via www.qucosa.de)

- Mirko Seifert. Designing Round-Trip Systems by Model Partitioning and Change Propagation. PhD thesis, Dresden University of Technology, June 2011.
  - Shows how roles simplify round-trip engineering by partitioning data
- Sebastian Richly. Autonom rekonfigurierbare Workflows. PhD thesis, Dresden University of Technology, December 2011.
  - Shows how roles can be used to provide an extensible tool platform
- Christian Wende. Language Family Engineering. PhD thesis, Dresden University of Technology, March 2012.
  - Shows how roles can be used to do context-based language composition
- Max Leuthäuser. A Pure Embedding of Roles - Exploring 4-dimensional Dispatch for Roles in Structured Contexts. PhD thesis, Technische Universität Dresden, August 2017.
  - This PhD thesis developes a programming language for contexts and roles, based on some implementation patterns and the base language Scala.
- Thomas Kühn. A Family of Role-Based Languages. PhD thesis, Technische Universität Dresden, March 2017.
  - This PhD develops language design with contexts and roles in CROM
- Georg Püschel. Testing Self-Adaptive Systems - A Model-based Approach to Resilience. PhD thesis, Technische Universität Dresden, June 2018.
  - Contexts for testing robots

- Matthias Schmidt, Jan Polowinski, Jendrik Johannes, and Miguel A. Fernández. An integrated facet-based library for arbitrary software components. In Thomas Kühne, Bran Selic, Marie-Pierre Gervais, and Francois Terrier, editors, ECMFA, volume 6138 of Lecture Notes in Computer Science, pages 261-276. Springer, 2010.

Best paper awards:

- C. Piechnick, S. Richly. Using Role-Based Composition to Support Unanticipated, Dynamic Adaptation, ADAPTIVE 2012
- J. Reimann, M. Seifert, U. Aßmann. Role-based generic model refactoring. MODELS Okt. 2010

TECHNISCHE
UNIVERSITÄT
DRESDEN

RoSI

MOST and
Role-based Context-Aware Software Infrastructures (RoSI)

4. Context- and Role-Oriented Modeling and Development

Prof. Uwe Aßmann

Version 20-0.1, 9/27/21

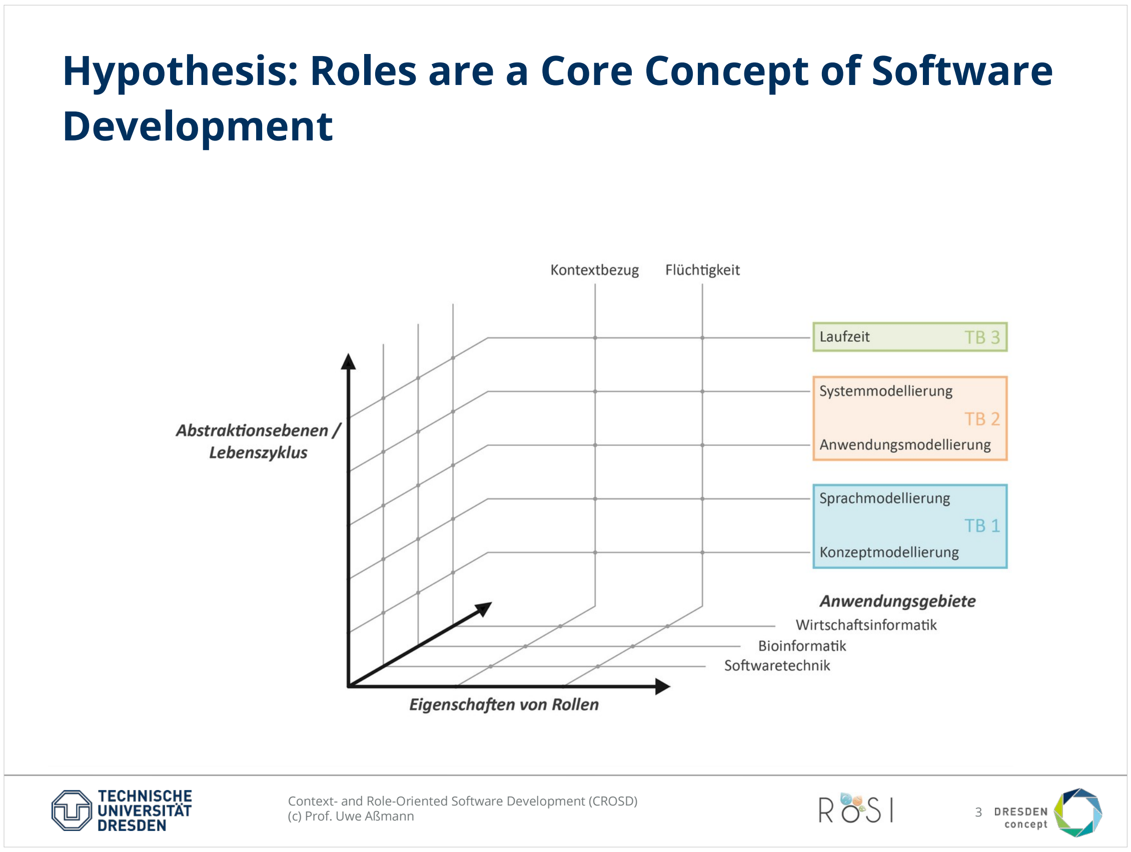
TECHNISCHE UNIVERSITÄT DRESDEN

RoSI

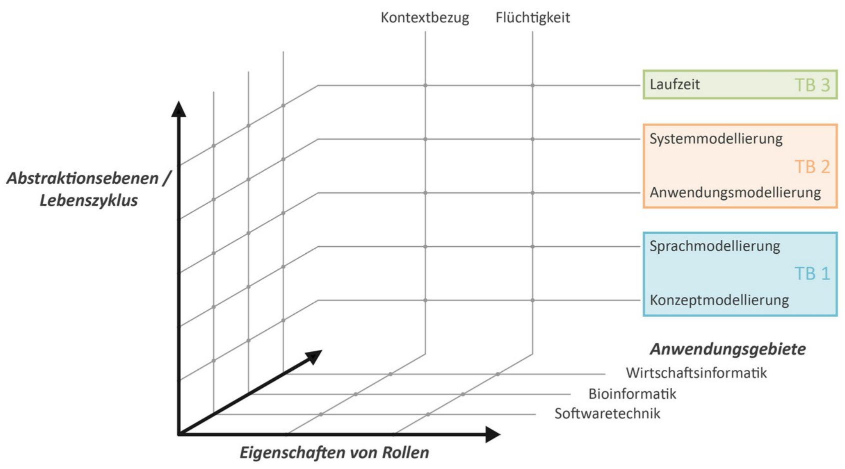
The RoSI Cube

4.1 Roles are a Core Concept in Software Development

Working still on locality and role mapping.



Hypothesis: Roles are a Core Concept of Software Development



Die Hypothese des GK spannt **einen 3-dimensionalen Raum auf:**

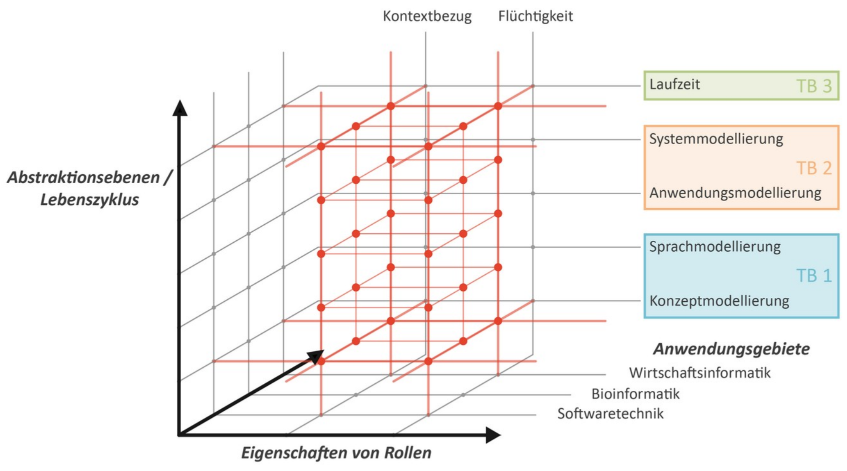
Themenbereiche erklären!!

Rollen sind ein Kernkonzept der Software-Entwicklung für Kontextbezug, aber auch für andere Eigenschaften (Dimension 1).

Um das nachzuweisen, muss man die Rollen in allen Abstraktionsebenen und Phasen des Lebenszyklusses untersuchen (Dimension 2).

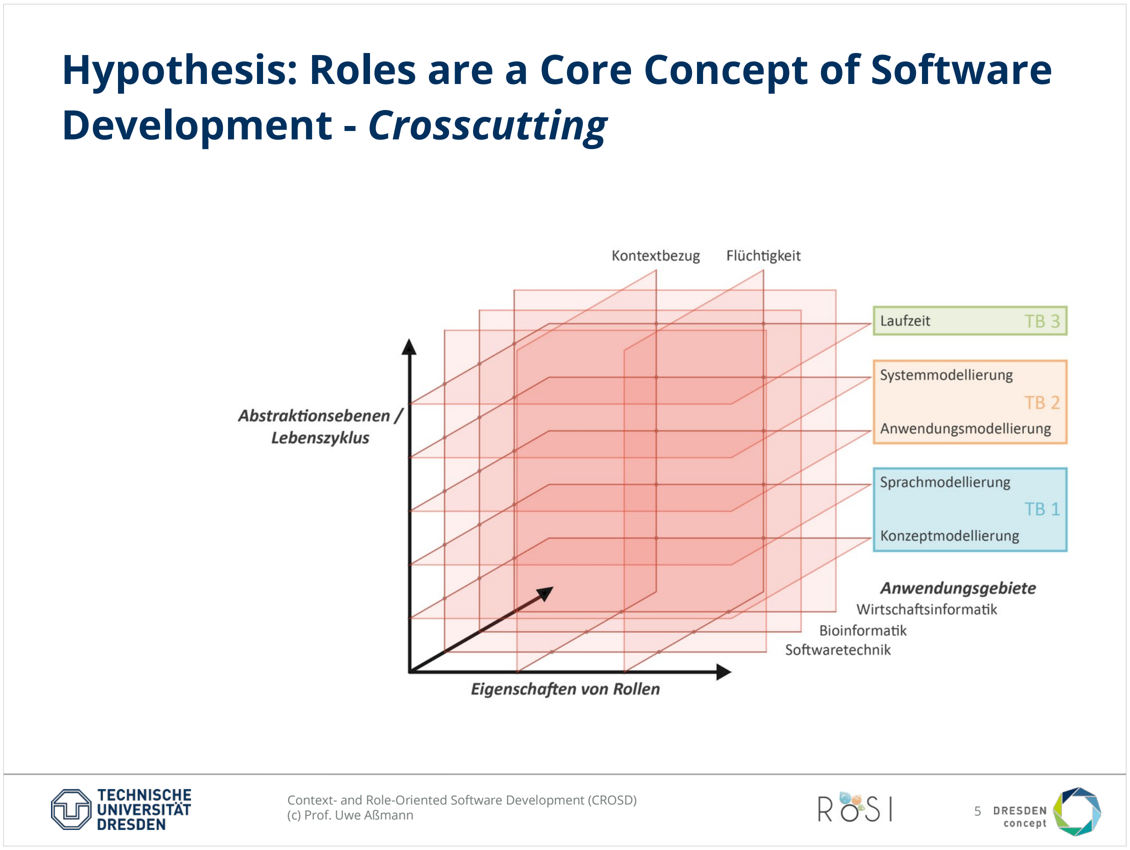
Daneben muss man Anwendungsgebiete untersuchen (Dimension 3).

Hypothesis: Roles are a Core Concept of Software Development - *Universality*

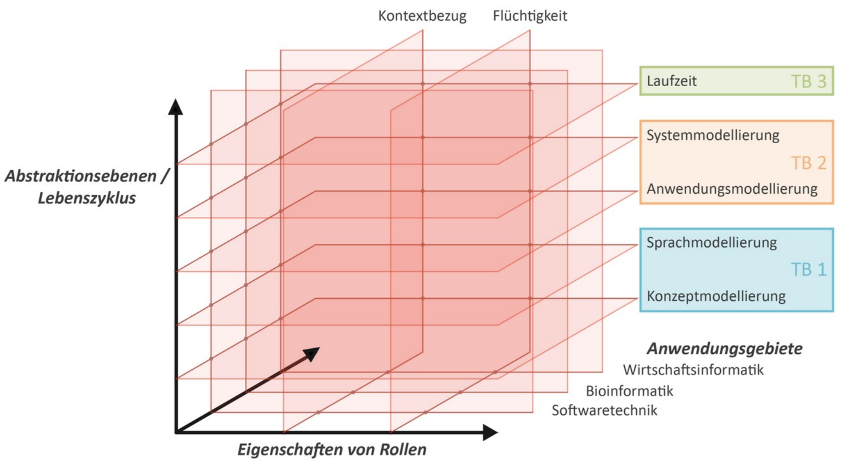


Dabei ist nicht nur singulär jeder Punkt in diesem Raum zu untersuchen (Universalität),

Universalität: für alle Zeitpunkte im Lebenszyklus

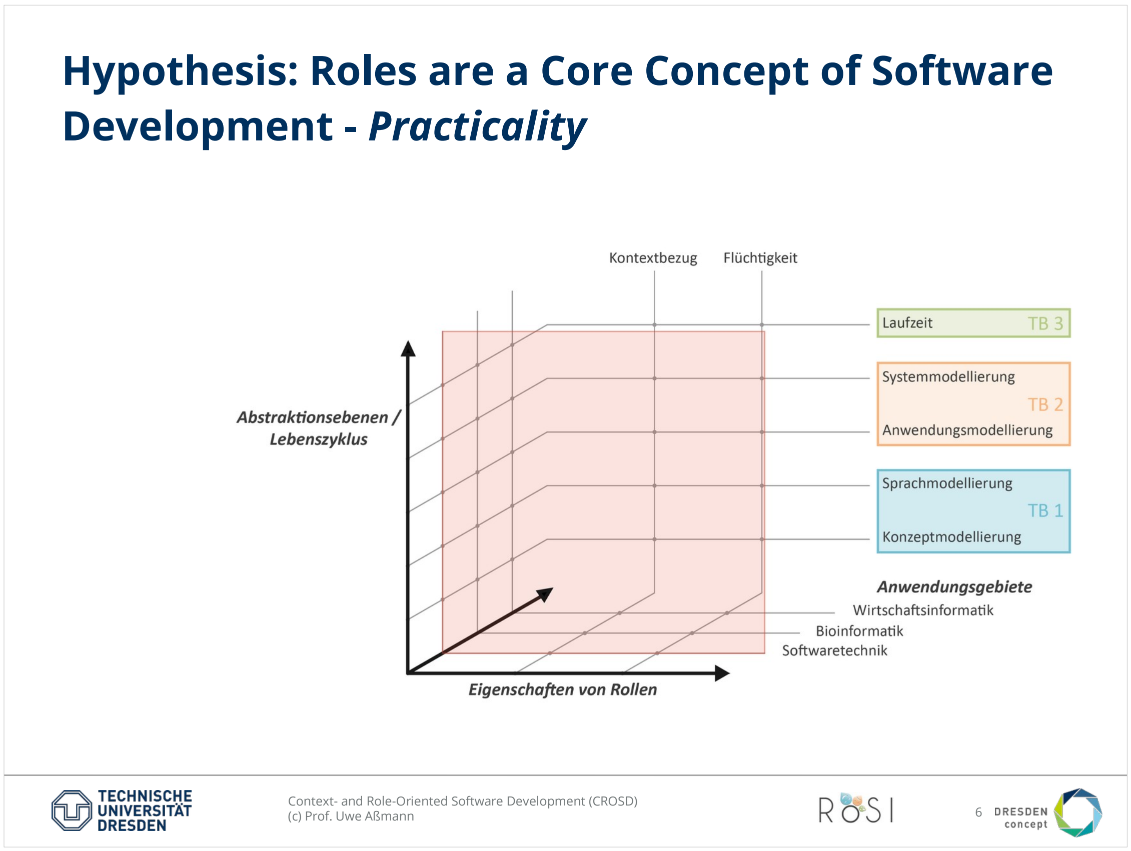


Hypothesis: Roles are a Core Concept of Software Development - *Crosscutting*

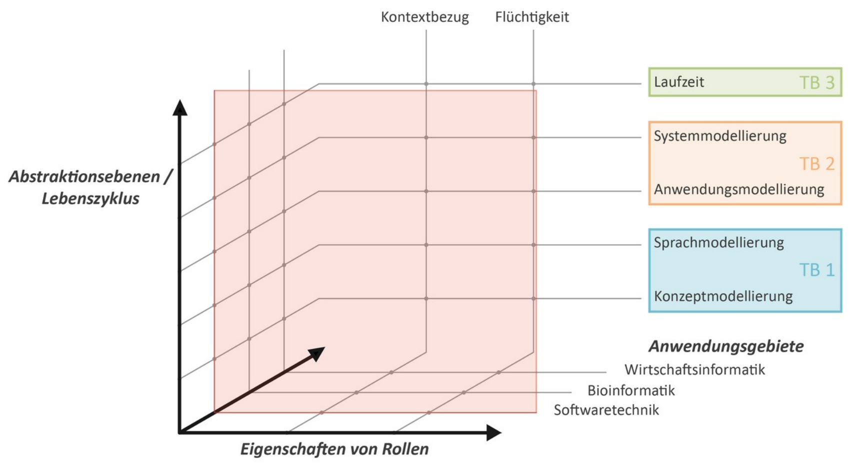


sondern auch die Durchgängigkeit (Verbindung und Interation von Punktmengen, Dimensionen oder Scheiben/slices)

Skalierbarkeit



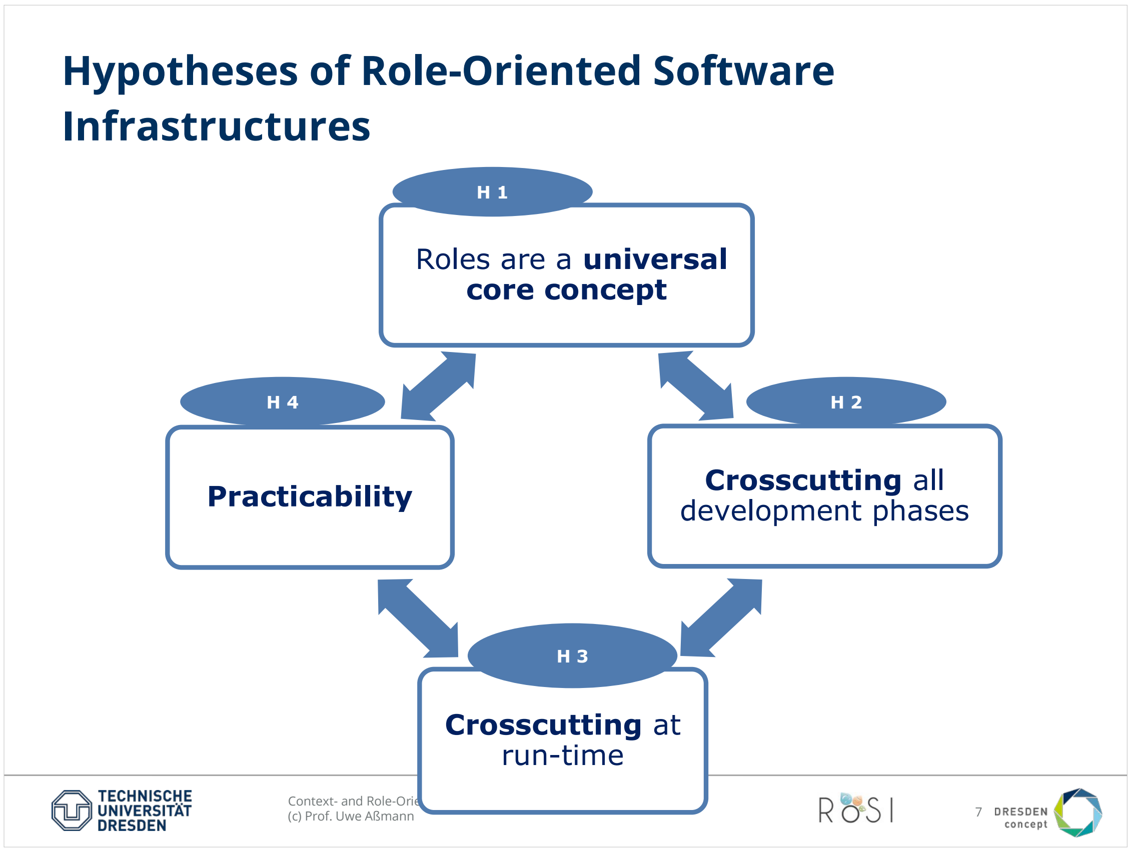
Hypothesis: Roles are a Core Concept of Software Development - *Practicality*



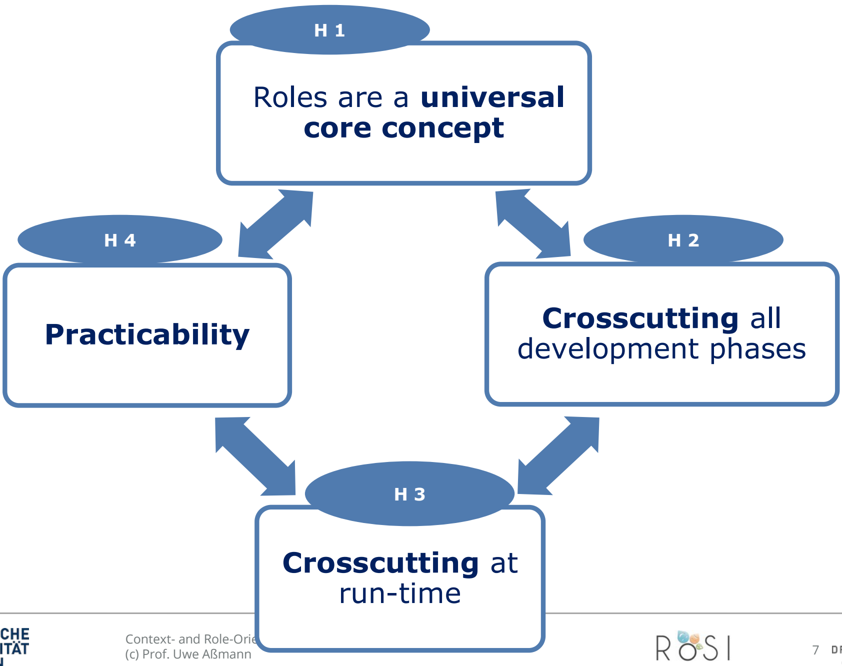
und die Praktikabilität (Nachweis in Anwendungsgebieten).

Dimension 3 ist unterspezifiziert, d.h. die Hypothese muss für weitere Anwendungsgebiete untersucht werden.

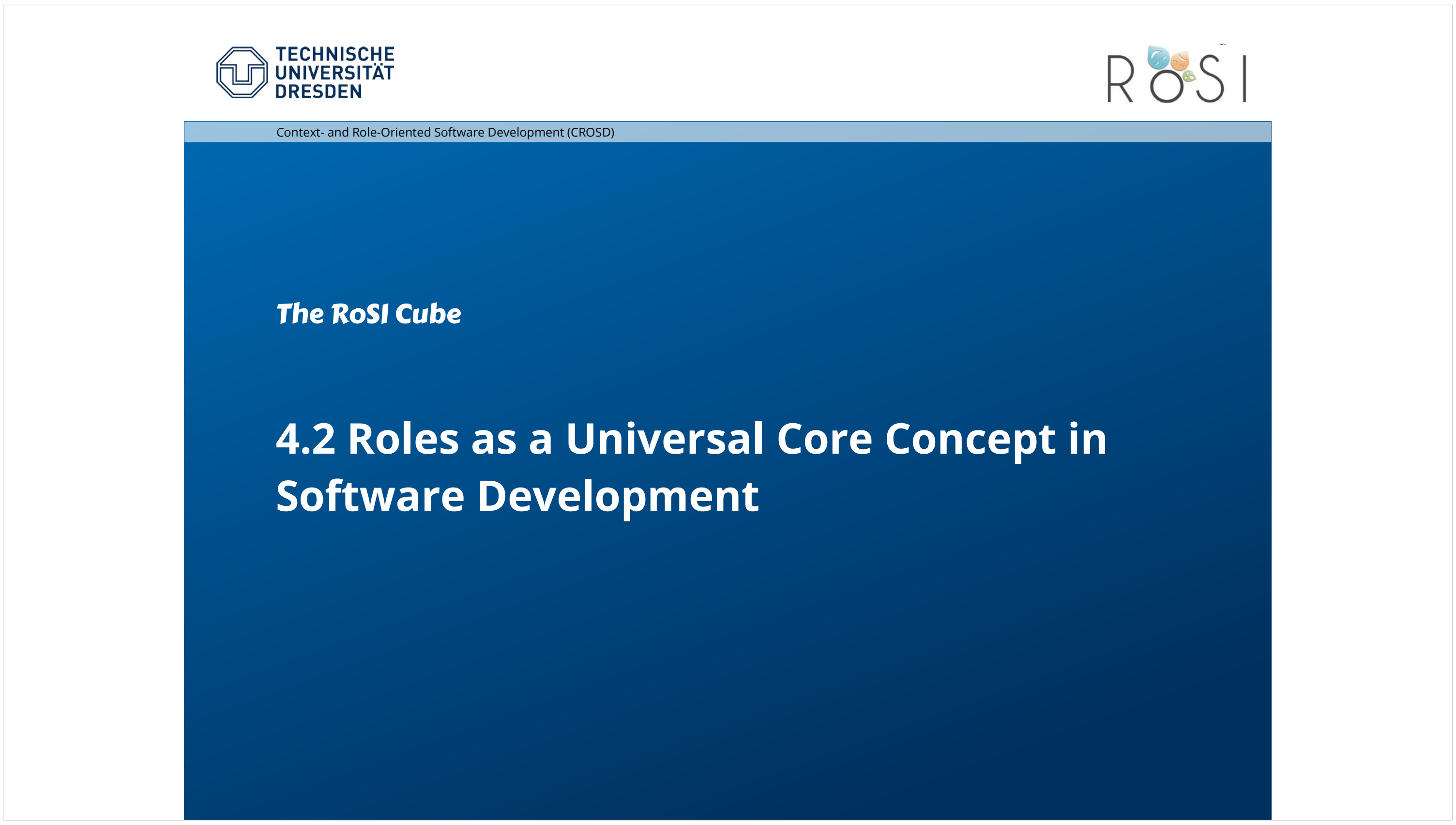
Die Hoffnung ist, dass mit der exemplarischen Nachweis für die untersuchten Gebiete dies einfacher ergibt bzw. Randbedingungen
Für weitere Untersuchungen bestimmt werden können.



Hypotheses of Role-Oriented Software Infrastructures


H 1
Roles are a **universal core concept**
H 4
Practicability
H 2
Crosscutting all development phases
H 3
Crosscutting at run-time

Animation weg



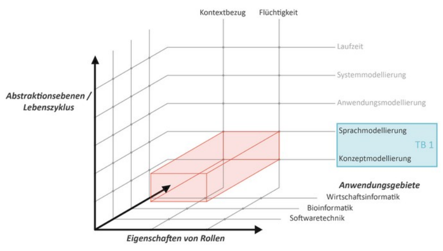
Context- and Role-Oriented Software Development (CROSD)

The RoSI Cube

4.2 Roles as a Universal Core Concept in Software Development

Working still on locality and role mapping.

Objective 1: Roles are a Core Concept of Software Development - *Universality*

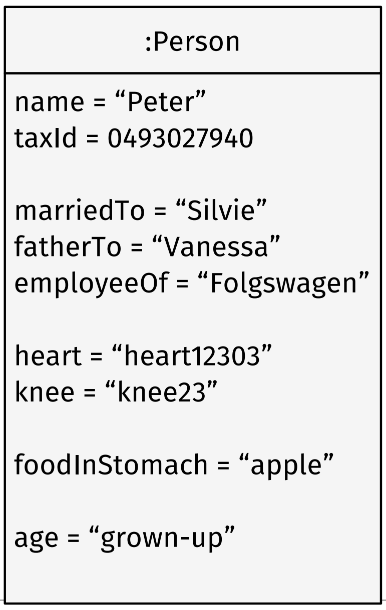


- Fine-grain information for better analysis of life times
- Behavior abstraction for better provability
- Better extensibility
- Better substitutability

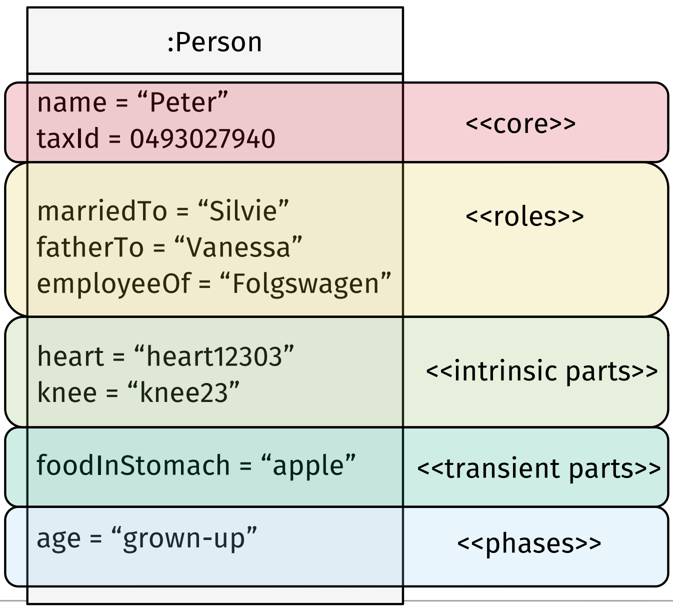
Context- and Role-Oriented Software Development (CROSD)

4.2.1. Fine-Grained Information for Separation of Concerns

Different Attributes

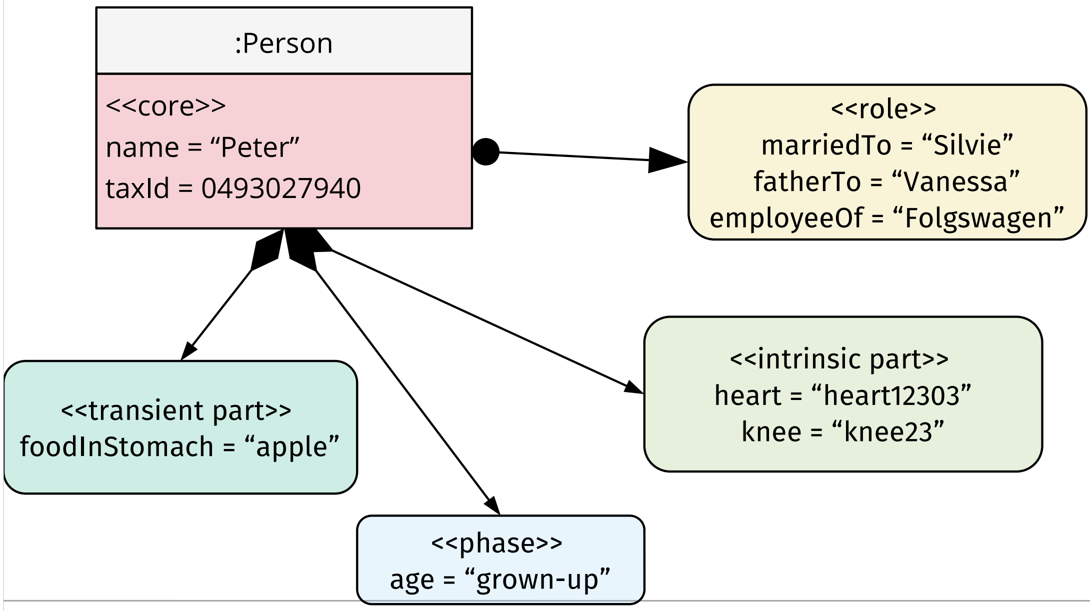


Different Attributes



Cores and Mixins (“Subobjects”, “Satellites”)

Role arrows are drawn with Rounded source



Separation of Concerns with Roles: Distinguishing Life-Times

- Roles are contextually dependent (founded), and have a different life-time as the core
  - → Memory allocation must be different
- Distinguish core-local, role-local, role-alternative, role-shared memory between core and roles
  - natural memory (core-local memory)
  - founded memory (context-dependent memory)
- Roles-of-roles (deep roles) are stacked upon roles;
  - Obstack allocation possible (mark-release heaps)
-

Roles can improve knowledge about life-time and co-life-time of memory

Separation of Concerns with Roles: Alias Freedom and Data Independence

- Natural and role-local memory are alias free
- Shared memory is still problematic (competitive writes)

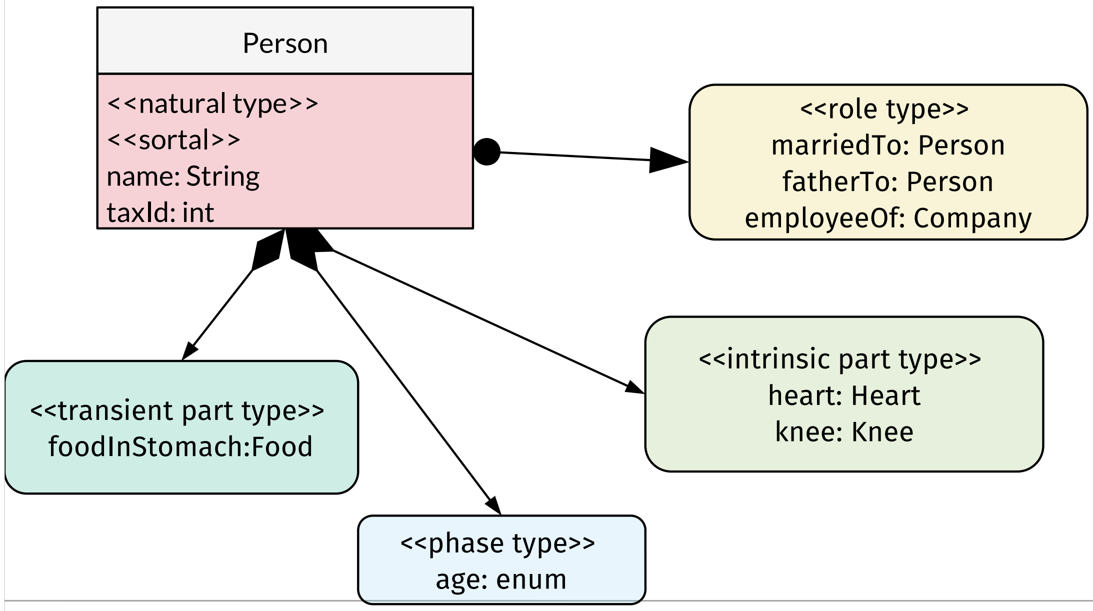
Roles can improve life-time and independence knowledge

Role Types are Metatypes (Mixin Types)

- A **metatype** describes a type (is a type of a type) [Guarino:OntoClean]
  - Natural Type
  - Part Type (intrinsic, shared, owned,..)
  - Role Type
  - Facet Type
  - Phase Type

Hypothesis:
The distinction of metatypes promotes
Separations of Concerns.

Distinguishing Mixin Types ("Colors", "Metatypes", "Satellite Types")



Separation of Concerns Helps

- The distinction of **metatypes** enables us to separate more concerns (SoC)
  - And bring it to run-time: Life-time, independence, ...
  - Cross-cutting: traceability, certification,...

Roles can improve modeling and programming.

TECHNISCHE UNIVERSITÄT DRESDEN

ROSI

Role-Oriented Context-Aware Software Infrastructures (ROSI)

4.2.2. Abstraction of Object Behavior - Compartments and Role Playing

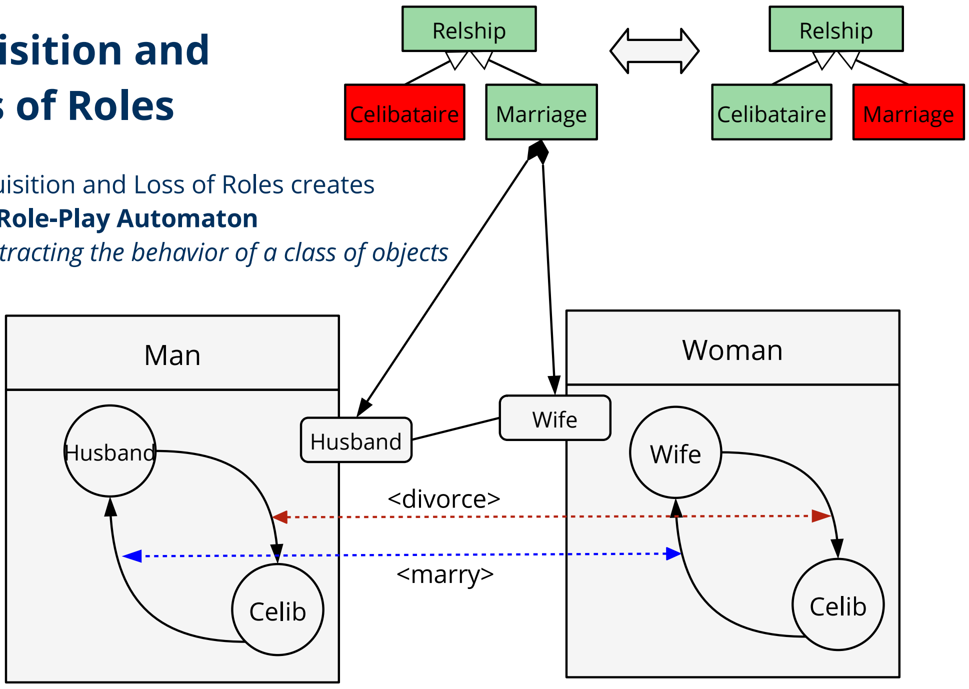
Roles are a Core Concept
Advantages of Roles:
The Role-Play Automaton
The Role-Play Petri Net

Role-Play Nets

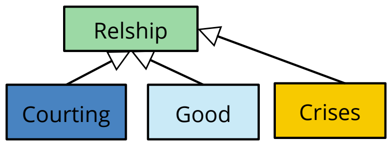
- The **role-play (petri) net of an object** switches in and off the object's roles
  - Specifies constraints on the order of the role play
  - Thereby constraints on the compartment activation
- Roles are specific states indicating
  - There is a compartment active to which the role belongs
  - There is a partner role within the compartment that can be called or notified or streamed
- Two forms:
  - Role-Play automaton (sequential)
  - Role-play net (parallel)

Aquisition and Loss of Roles

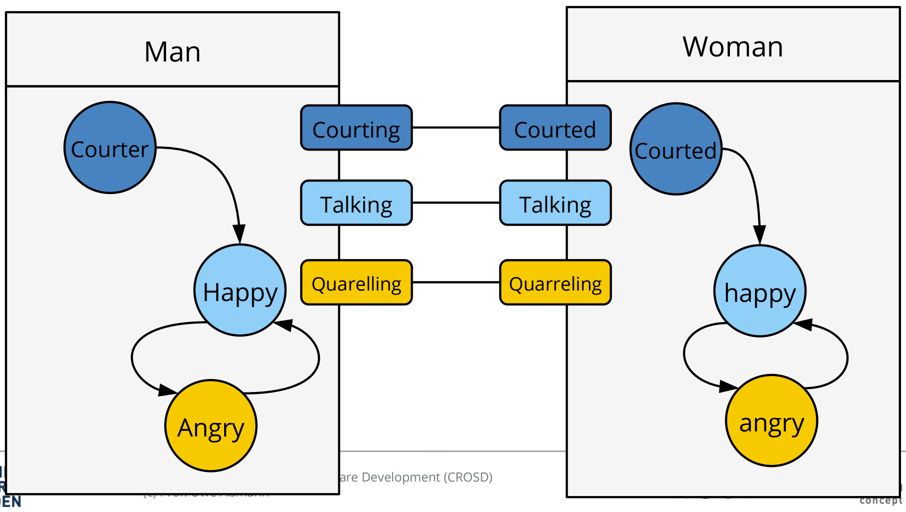
- Aquisition and Loss of Roles creates an **Role-Play Automaton** *abstracting the behavior of a class of objects*



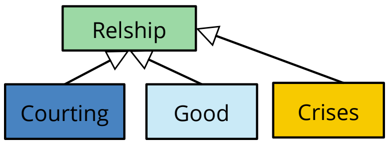
Aquisition and Loss of Roles with Role-Play Automata



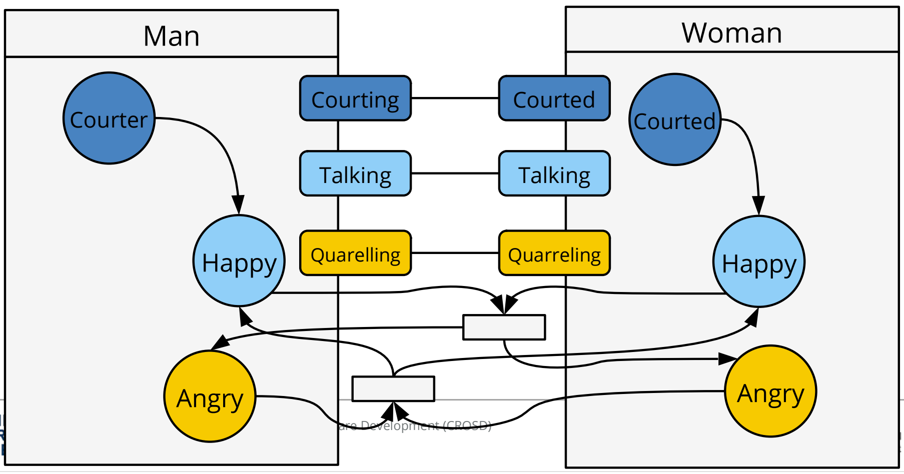
- Aquisition and Loss of Roles creates an **Role-Play Automaton** *abstracting the behavior of a class of objects*
- *Here:* some states with the same color are coupled



Aquisition and Loss of Roles with Role Nets



- Aquisition and Loss of Roles of parallel objects and their state transitions creates a **Role-Play Net** indicating parallel transitions
- **Here:** exclusive compartments, exclusive roles
- Coupling via synchronizing transitions



A Fancy Observation

Humans think and argue based on Role-Play Nets

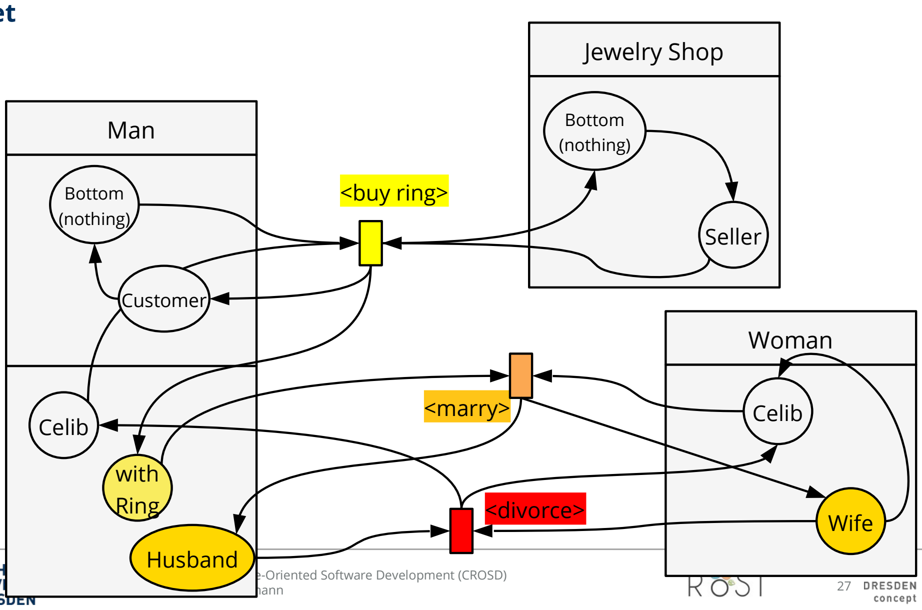
- “become a father”
- „if you are a husband, you should care about your wife“
- “become a driver”, „drivers, watch out for pedestrians“
- “cease to be an employee”
- “cease to be student”

Role-Play Net of a Compartment

- The **role-play net of a compartment** is the view on all role-play nets comprising all roles places of the compartment.
- When a compartment is activated there is the constraint that
  - all the compartment‘s roles in all their players are activated (firable)
  - Otherwise the net is inconsistent.
- When a compartment is deactivated there is the constraint that
  - all the compartment‘s roles in all their players are deactivated (non-firable)

Parallel Aquisition and Loss of Roles

- Parallel Aquisition and Loss of Roles in a parallel class creates an **Role-Play (Petri) net**



Regular Adaptability and Variability

- Many applications have a restricted form of adapability (variability)
- A **regularly adaptable class** has a finite role-play automaton with n compartments as states
  - Infinitely many adaptations, but regularly many

The role-play petrinet of a regularly adaptable class is k-bounded.

Roles are a Core Concept

4.2.3. Advantages of Roles: Behavioral Extensibility

Extensibility as a Universal Feature of Role-based Infrastructures

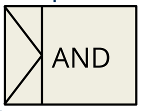
- New compartments with their roles can easily be integrated into an application → extensibility (see lecture 01)
- Roles may have different implementation paradigms (groundings):
  - Functional programs
  - Workflow nets
  - Data-flow nets (see MOST)
  - Attributed trees (see MOST)
- All of them have the extensibility feature, but use different „open operators“ for extensions.

Example: Extending Role-based Systems Grounded by Workflow Nets (Petri Nets)

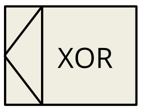
- With an appropriate behavioral specification language, role classes and natural classes can be extended with regard to behavior
- Example: Workflow Nets are a specific form of Petri Nets
  - **Place workflow nets** have one single input place and a single output place
  - **Transition workflow nets** have one single input transition and a single output transition
- For extension (and variation) of behavior of classes, we use the extension of AND, OR, XOR split and join *open transition operators*

Complex Transition Operators in Workflow Nets: Join and Split „Open“ Transitions (of YAWL)

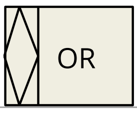
- All incoming places are ready (conjunctive input, AND-join)



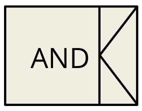
- One out of n incoming places are ready (disjunctive input)



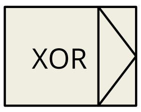
- Some out of n incoming places are ready (selective input)



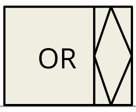
- All outgoing places are filled (conjunctive output, AND-split)



- One out of n outgoing places are filled (disjunctive output, XOR split)

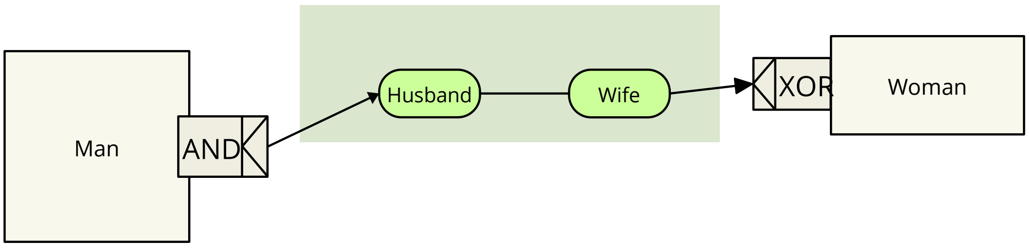


- Some out of n outgoing places are filled (selective output, OR-split)



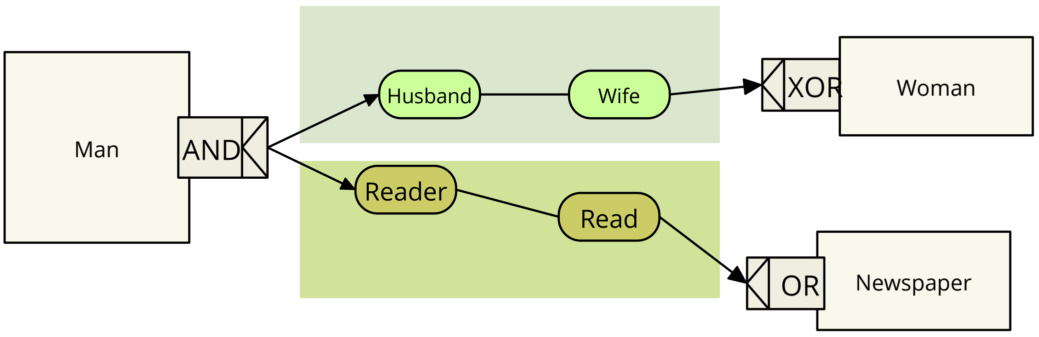
Extension of Workflows with new Place Workflow Nets

- Behavior can be added in *slices* to *open* split and join operators



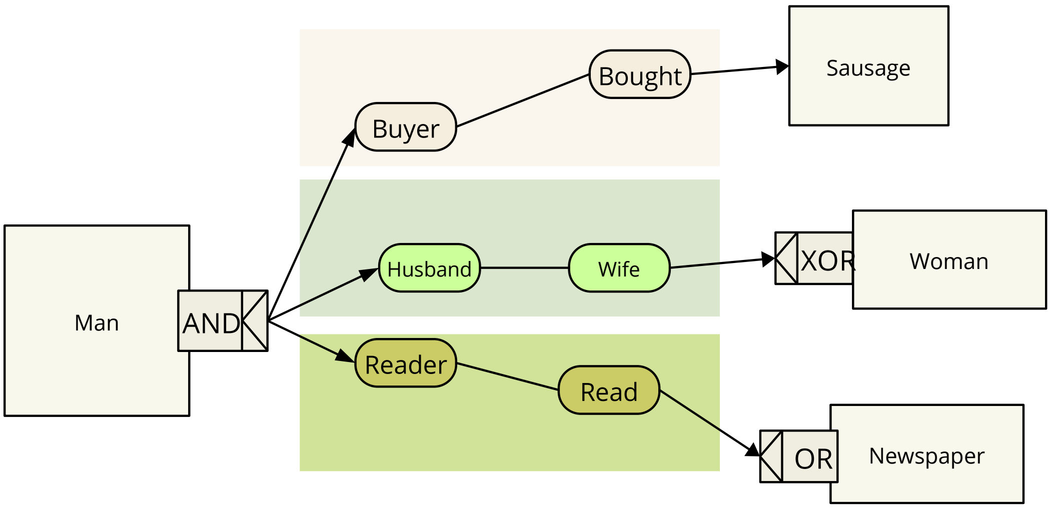
Extension of Workflows with new Place Workflow Nets

- Behavior can be added in *slices* to *open* split and join operators



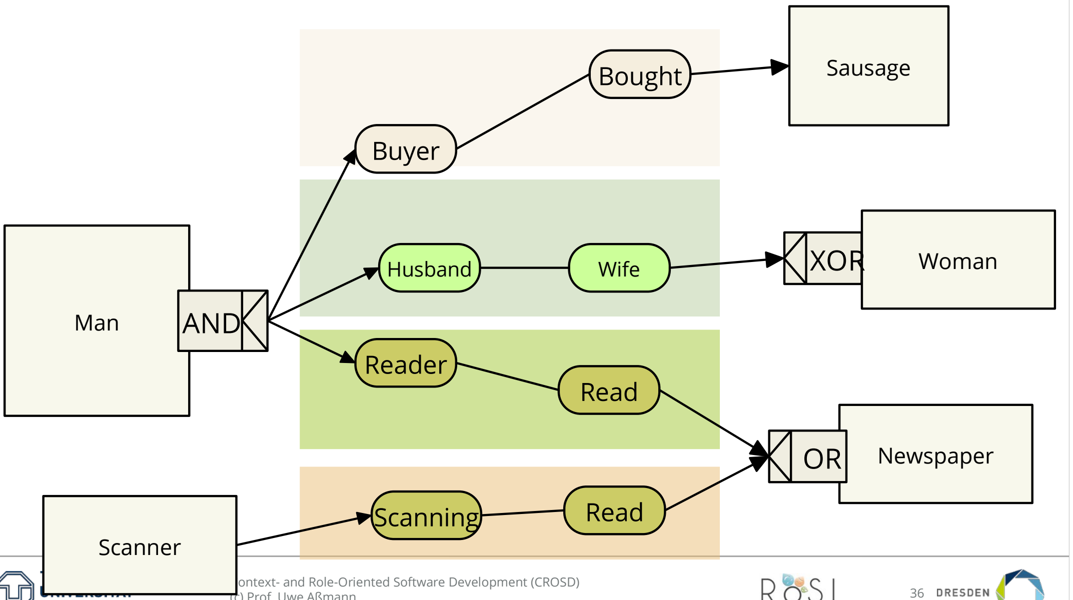
Extension of Workflows with new Place Workflow Nets

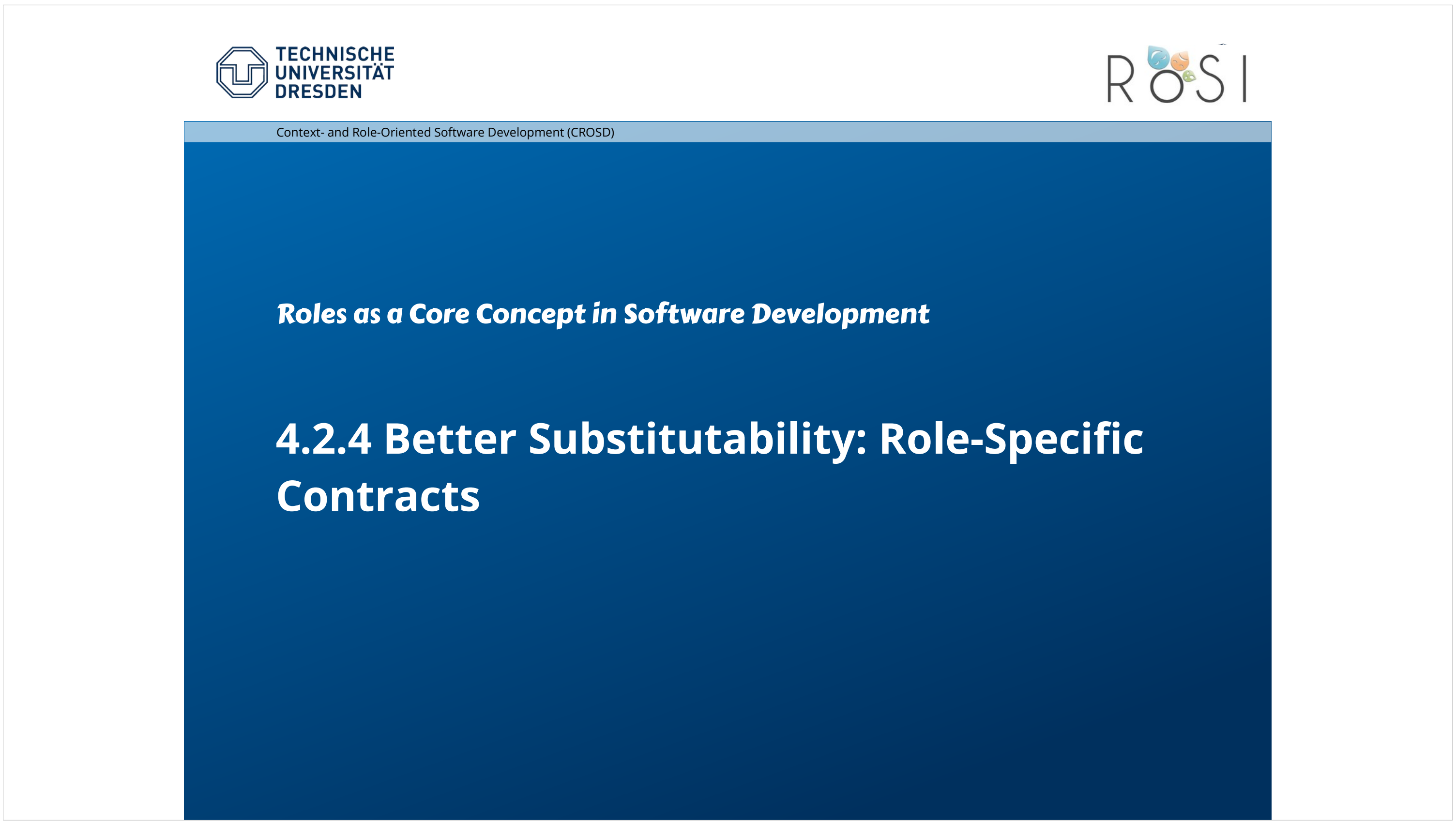
- with AND semantics



Extension of Workflows with new Place Workflow Nets

- with OR semantics




TECHNISCHE UNIVERSITÄT DRESDEN

RoSI

Context- and Role-Oriented Software Development (CROSD)

Roles as a Core Concept in Software Development

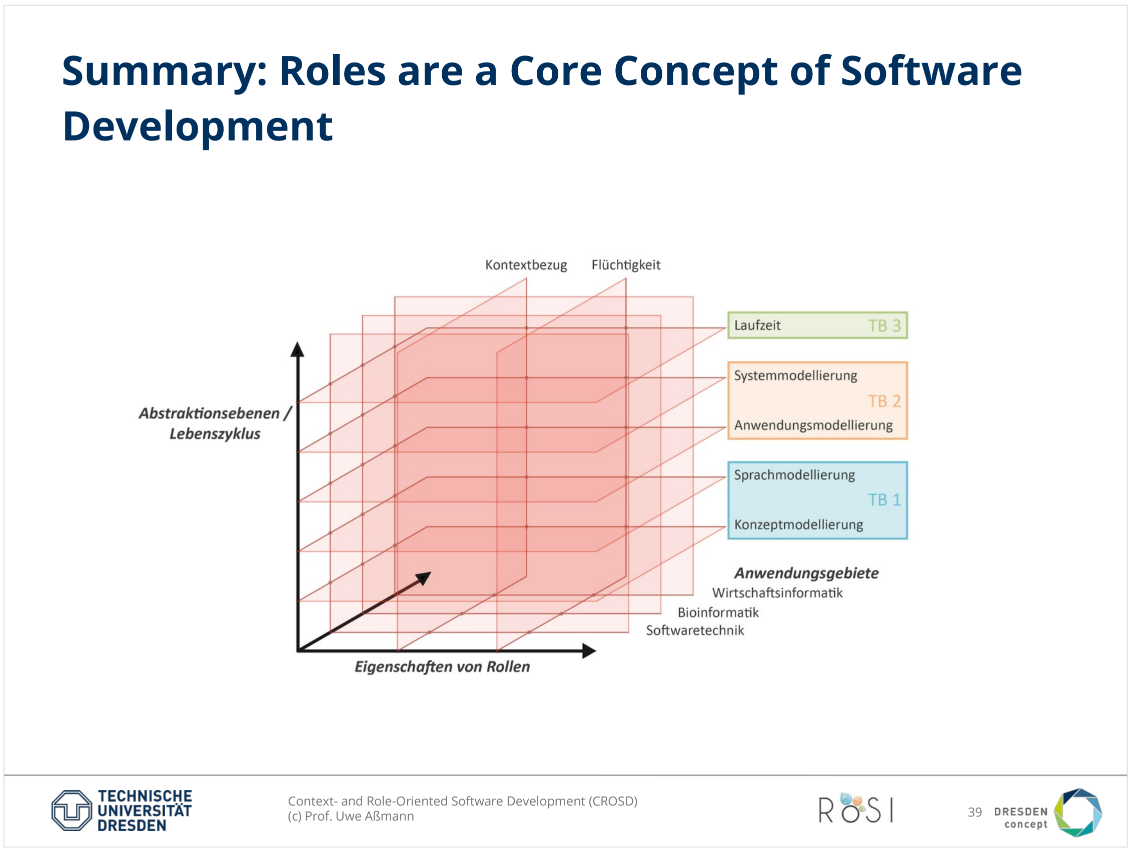
4.2.4 Better Substitutability: Role-Specific Contracts

Working still on locality and role mapping.

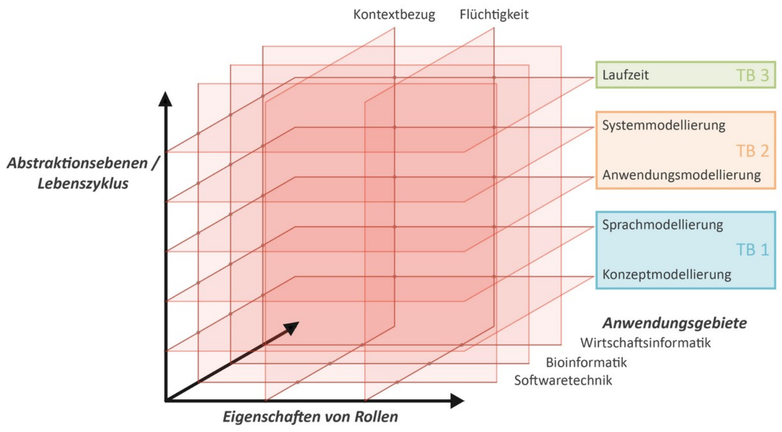
Separation of Concerns with Roles: Role-Based Contracts are Context-Based

- Contracts describe conditions for *substitutability*
- A **contract** is a constraint on inputs (precondition), outputs (postcondition) and invariants of a component (see courses CBSE, ST)
- Life-time and Alias Independence enable simpler proof of contracts
- The Role-Play Automaton determines which contracts are active
  - in which context

Roles can improve contract theory for sequential and parallel classes



Summary: Roles are a Core Concept of Software Development

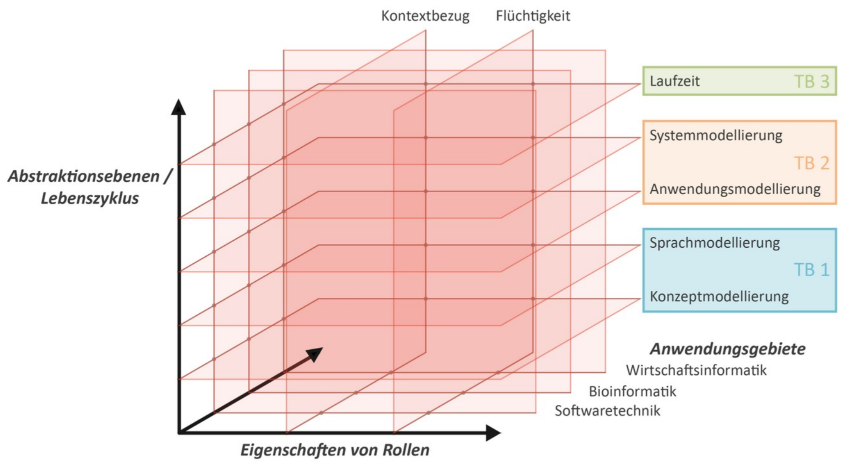


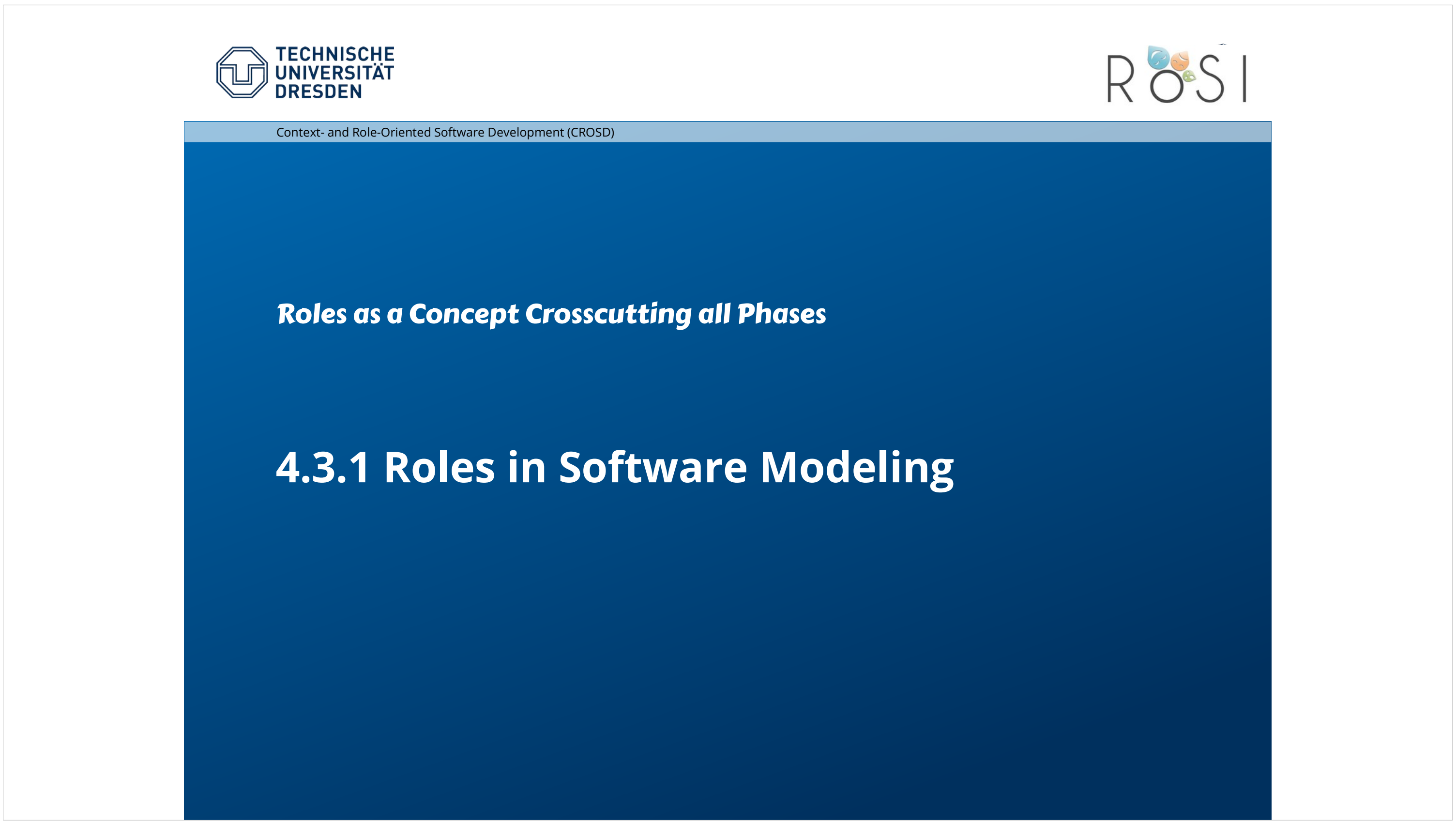
sondern auch die Durchgängigkeit (Verbindung und Interation von Punktmengen, Dimensionen oder Scheiben/slices)

Skalierbarkeit

4.3. Roles are a Concept Crosscutting all Phases

Objective 2: Roles Crosscut all Development Phases




TECHNISCHE UNIVERSITÄT DRESDEN

RoSI

Context- and Role-Oriented Software Development (CROSD)

Roles as a Concept Crosscutting all Phases

4.3.1 Roles in Software Modeling

Working still on locality and role mapping.

TECHNISCHE UNIVERSITÄT DRESDEN

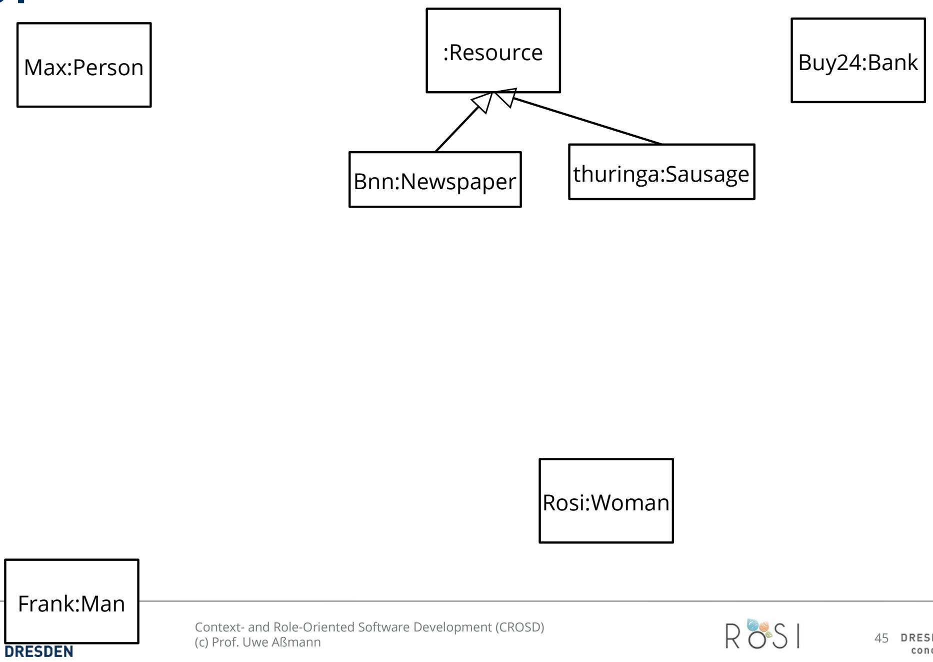
ROSI

4.3.1.1. How to Do Object-Oriented Analysis with ROSI

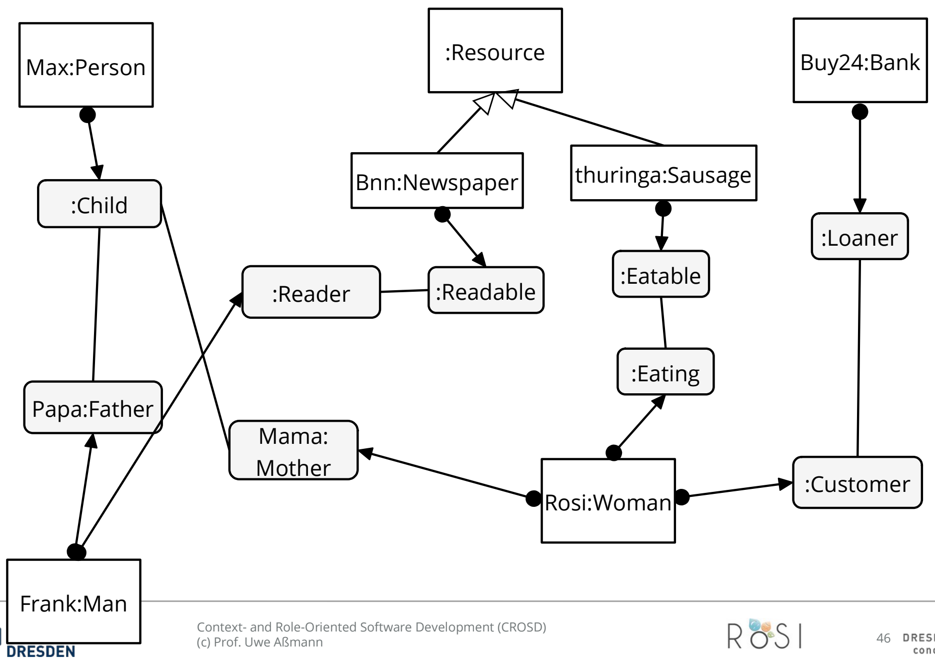
RoSI Object Models
RoSI Component Models

- An **Object Model** describes a structure and behavior for all objects in all phases of the life cycle
  - It forms type systems
  - specification languages
  - the parallelism available
- Roles and Contexts can be used in Object-oriented Analysis (OOA), offering a very flexible object model

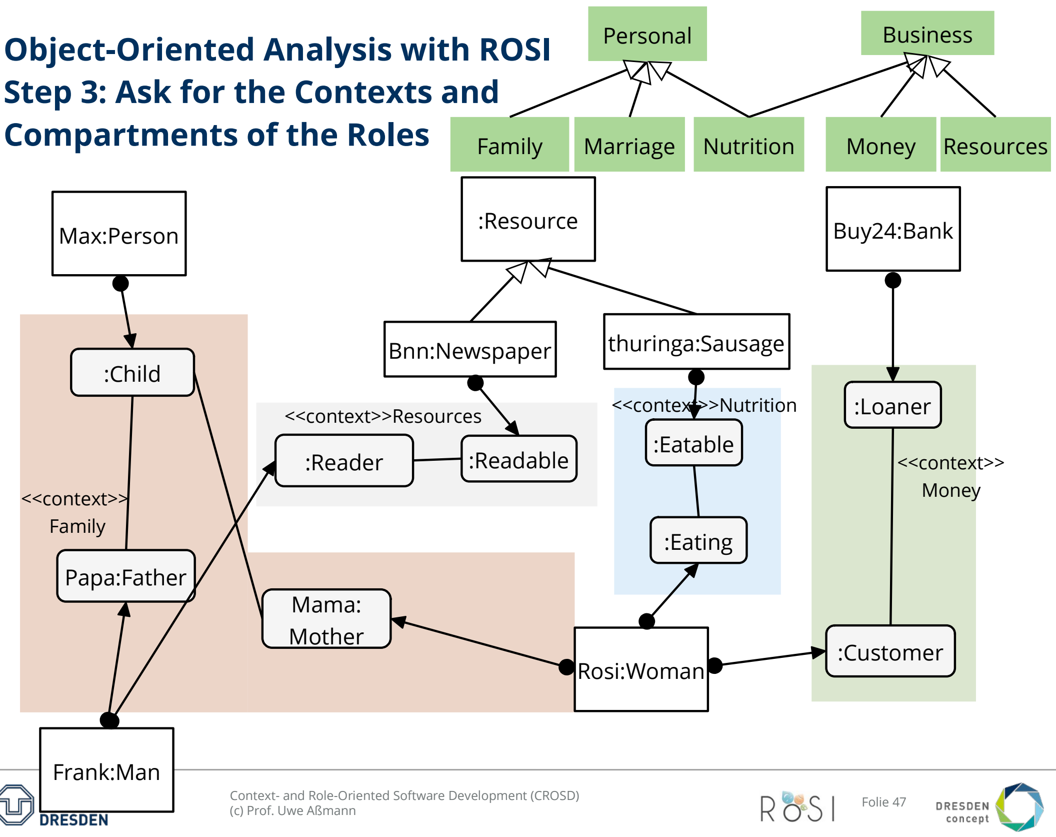
Object-Oriented Analysis with ROSI
Step 1: Ask for the Core Objects with Natural Types



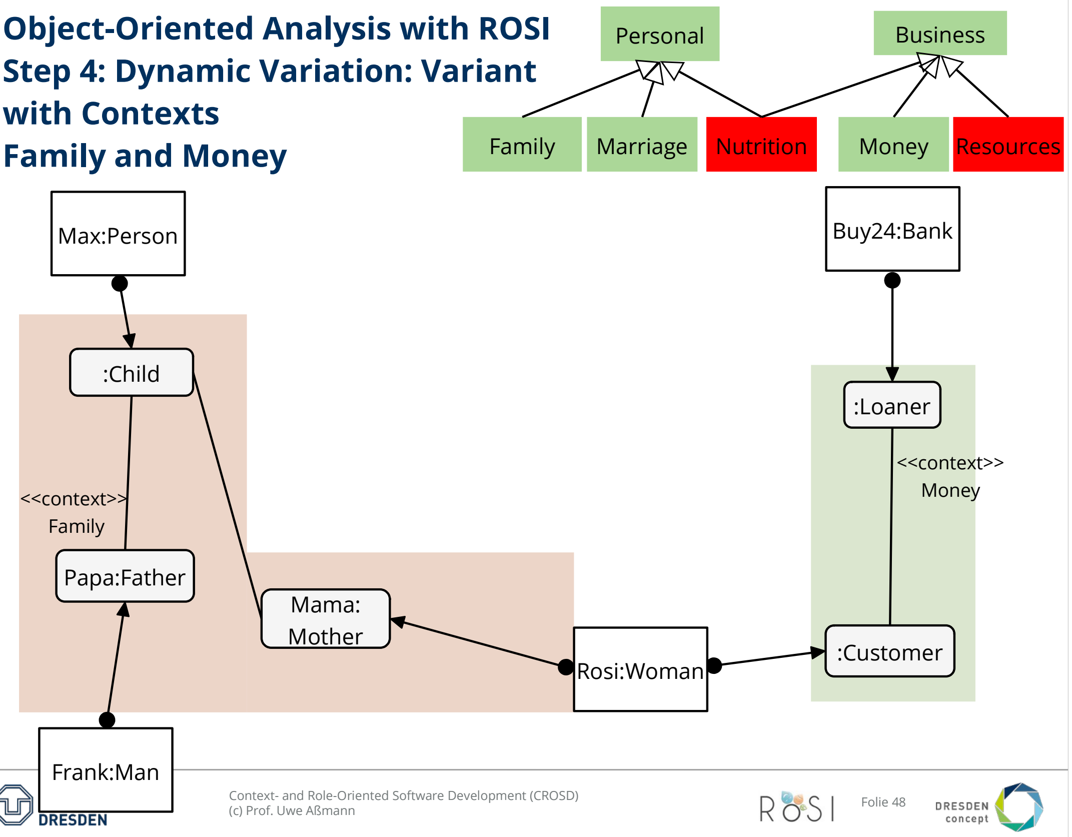
Object-Oriented Analysis with ROSI
Step 2: Ask for the Roles with Founded Types



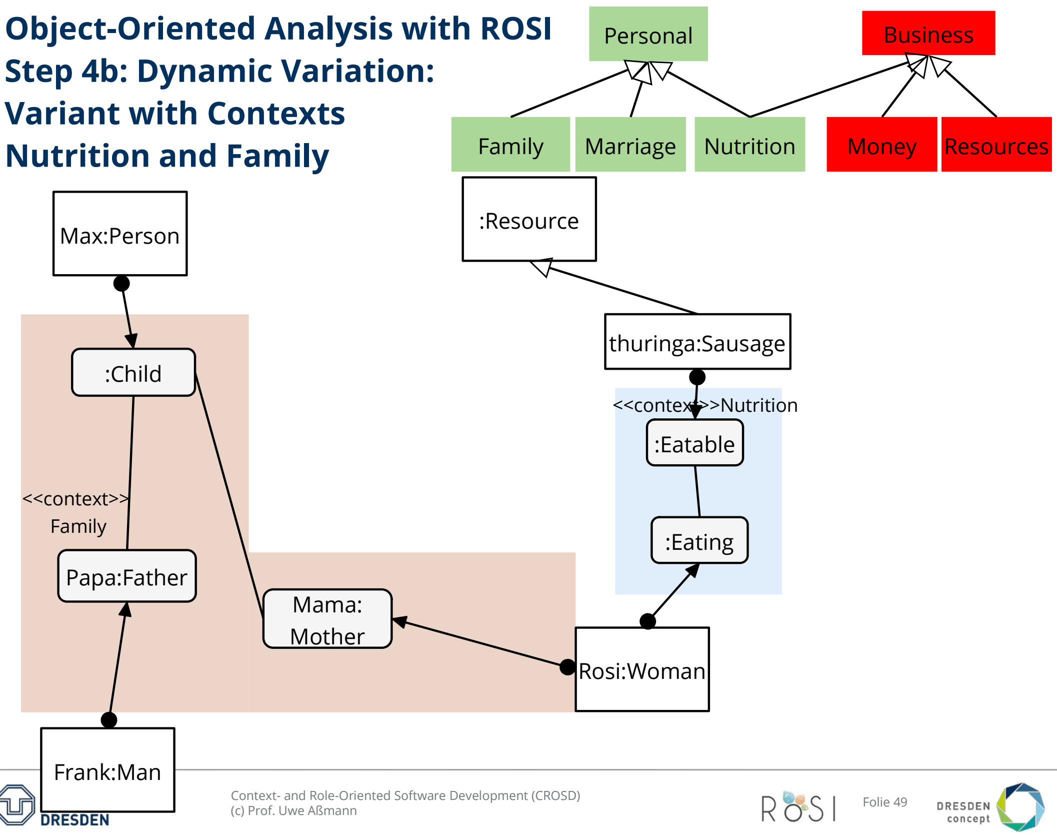
Object-Oriented Analysis with ROSI Step 3: Ask for the Contexts and Compartments of the Roles



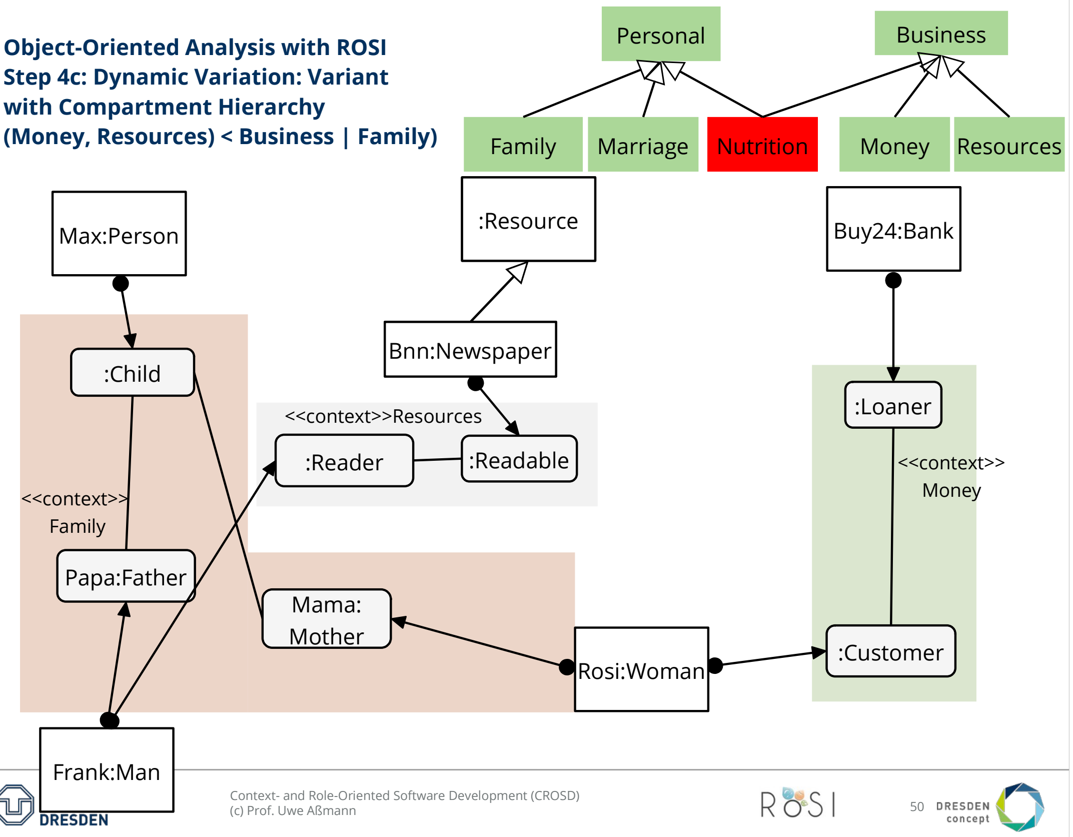
Object-Oriented Analysis with ROSI
Step 4: Dynamic Variation: Variant with Contexts
Family and Money



Object-Oriented Analysis with ROSI Step 4b: Dynamic Variation: Variant with Contexts Nutrition and Family



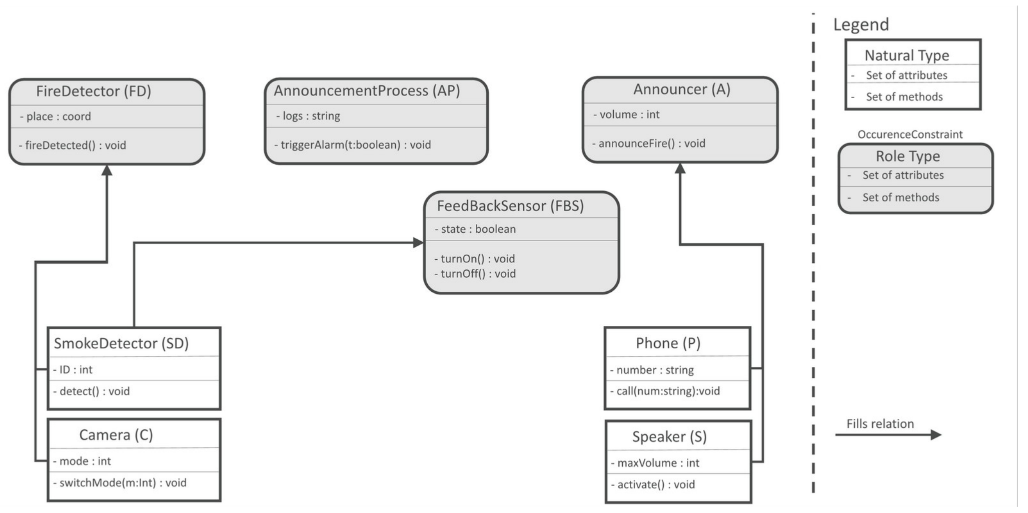
Object-Oriented Analysis with ROSI Step 4c: Dynamic Variation: Variant with Compartment Hierarchy (Money, Resources) < Business | Family)



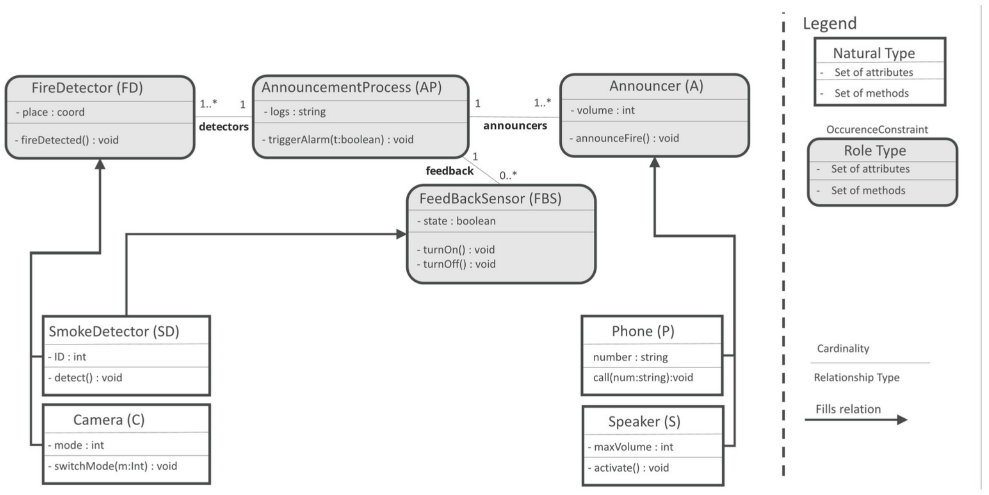
4.2. Scenario Fire Alarm – in the CROM Modeling Language

Context-Dependent Runtime Models

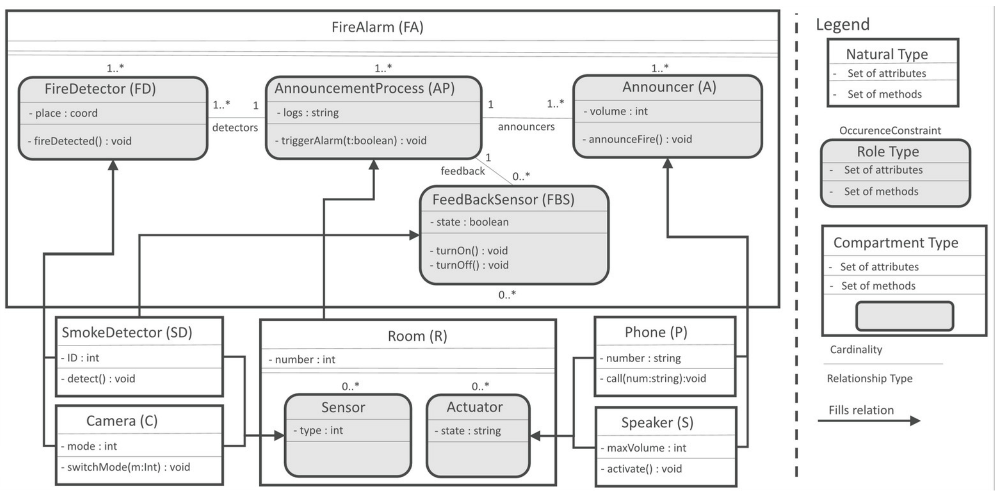
Compartment Role Object Model (CROM) *[Kühn2015]*



Context-Dependent Runtime Models
Compartment Role Object Model (CROM) *[Kühn2015]*

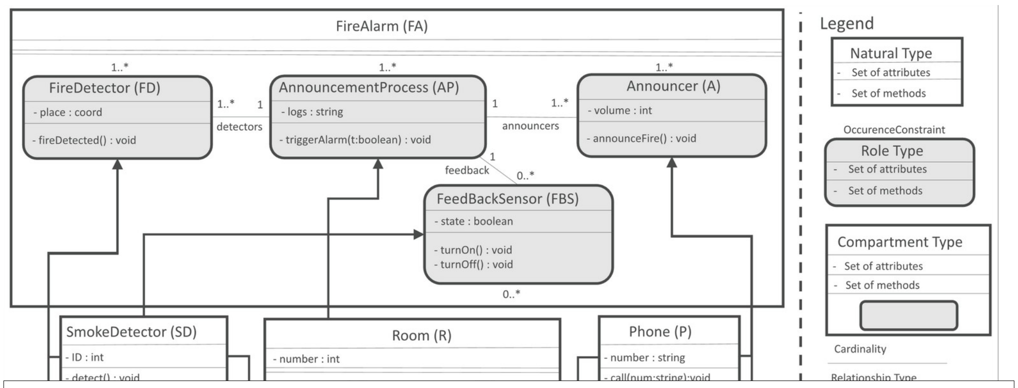


Context-Dependent Runtime Models
Compartment Role Object Model (CROM) *[Kühn2015]*



Context-Dependent Runtime Models

Compartment Role Object Model (CROM) *[Kühn2015]*



Key properties

•Roles and Relationships depend on the compartments (contexts)

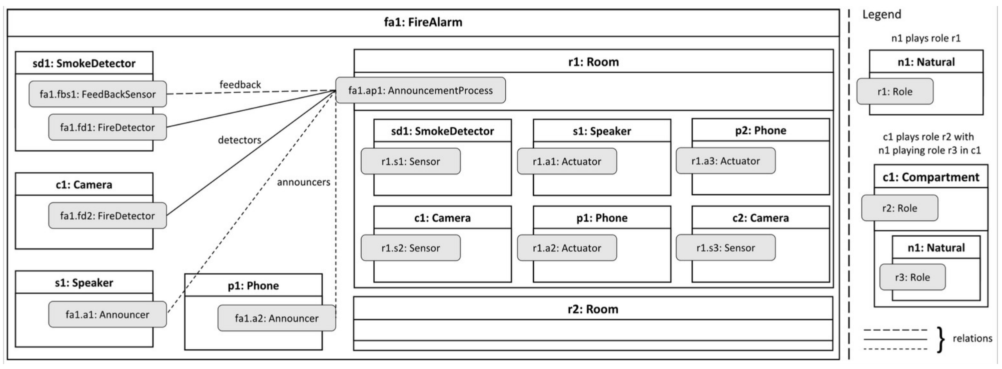
•Roles change over time

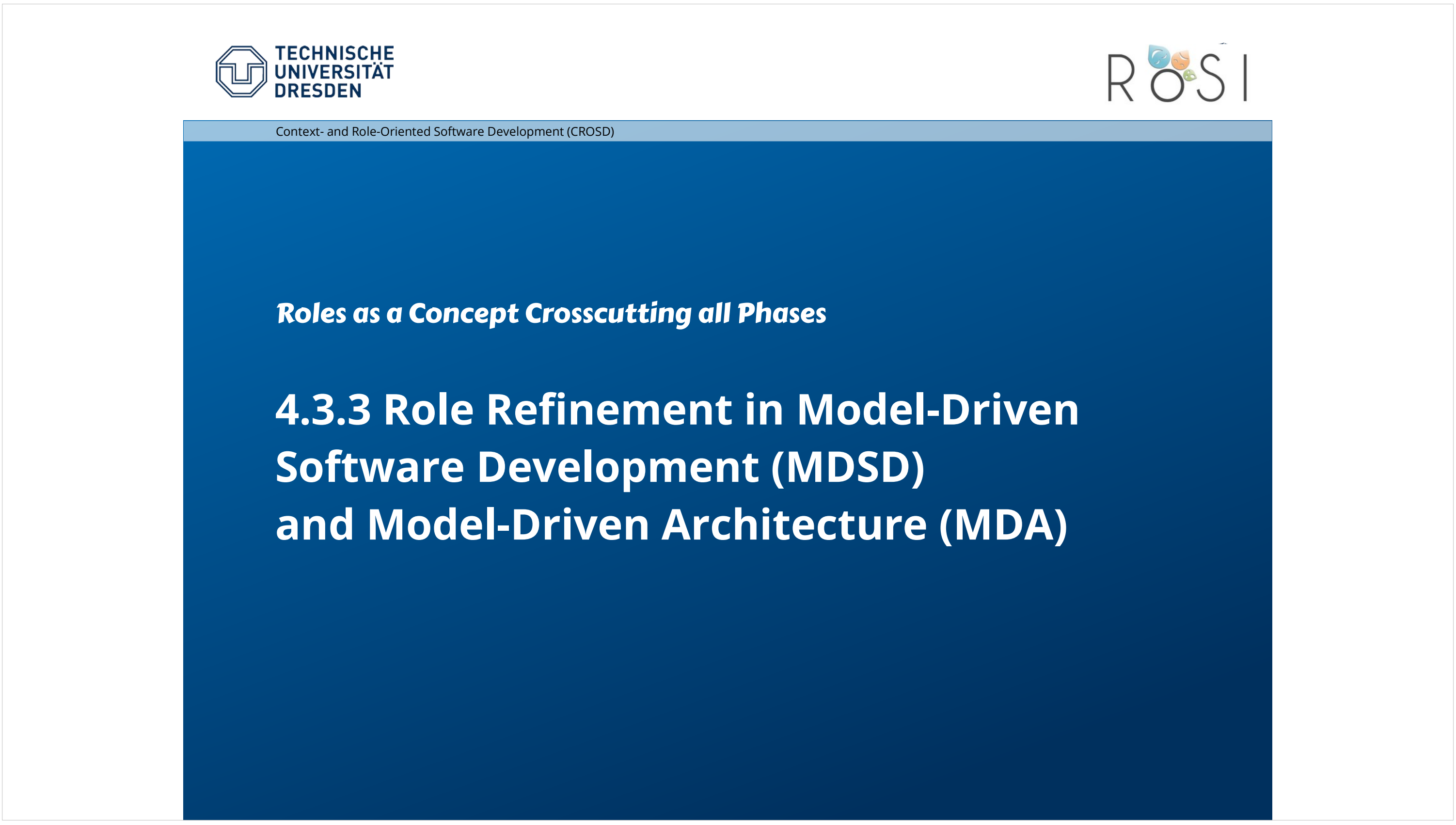
•Compartments, “players” and roles have their own identity

•Formal definition of *well-formedness*, *compliance*, and *validity*

Context-Dependent Runtime Models

Compartment Role Object Instance (CROI) *[Kühn2015]*

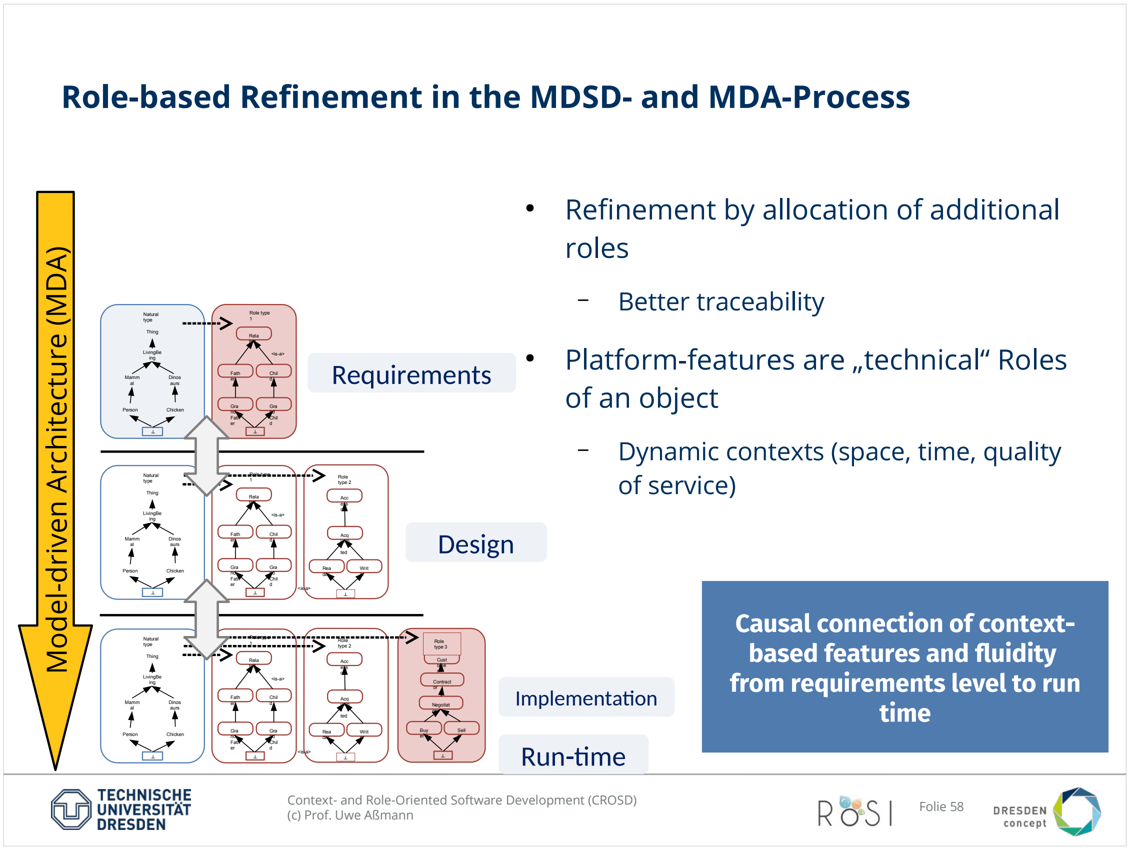




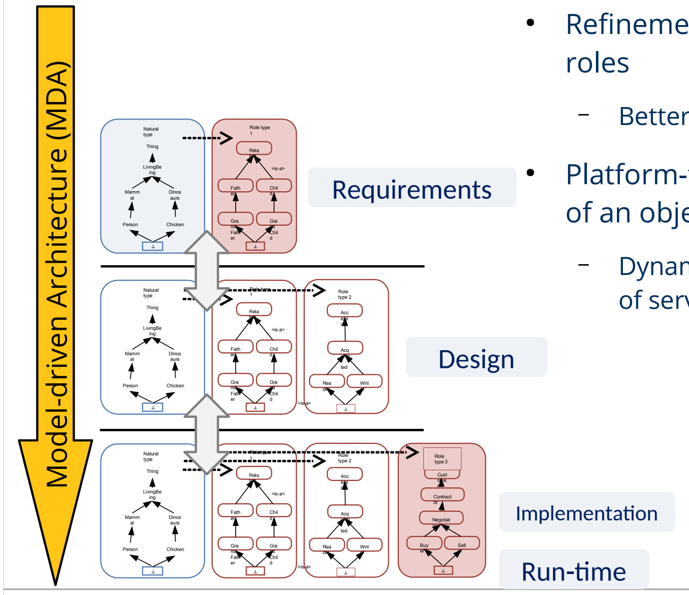
Roles as a Concept Crosscutting all Phases

4.3.3 Role Refinement in Model-Driven Software Development (MDSD) and Model-Driven Architecture (MDA)

Working still on locality and role mapping.



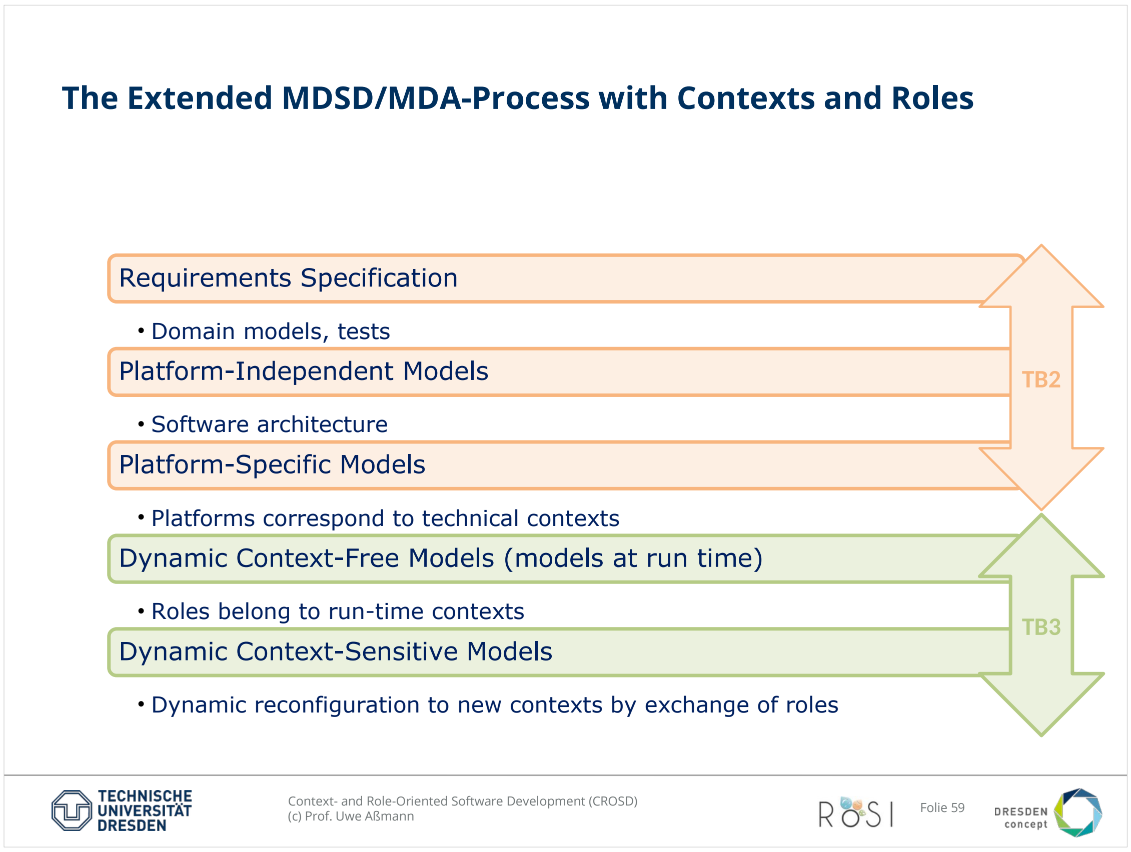
Role-based Refinement in the MDSD- and MDA-Process



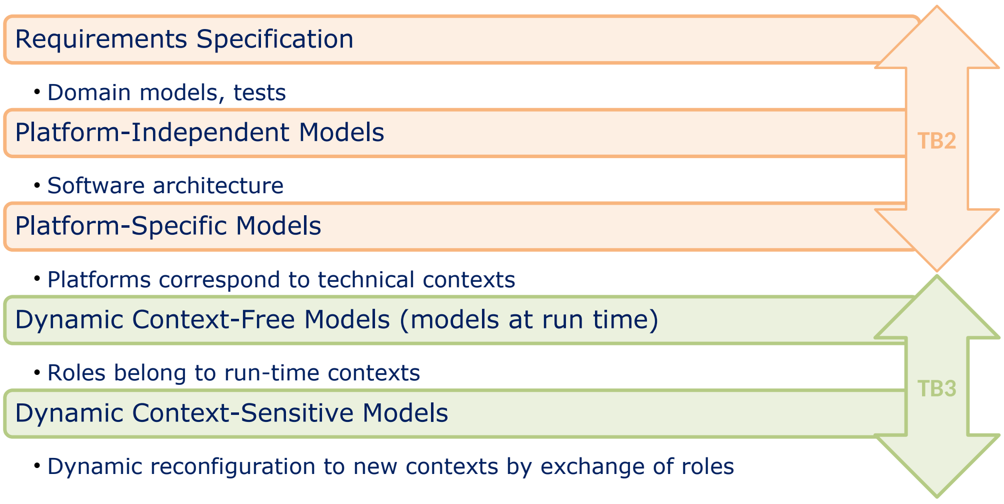
- Refinement by allocation of additional roles
  - Better traceability
- Platform-features are „technical“ Roles of an object
  - Dynamic contexts (space, time, quality of service)

Causal connection of context-based features and fluidity from requirements level to run time

Die Faktorisierung hilft, die Traceability von natürlichen Objekten zu verbessern, denn sie können nun von Rollen unterschieden werden



The Extended MDSD/MDA-Process with Contexts and Roles



Requirements Specification
Platform-Independent Models
Platform-Specific Models
Dynamic Context-Free Models (models at run time)
Dynamic Context-Sensitive Models
TB2
TB3

- Domain models, tests
- Software architecture
- Platforms correspond to technical contexts
- Roles belong to run-time contexts
- Dynamic reconfiguration to new contexts by exchange of roles

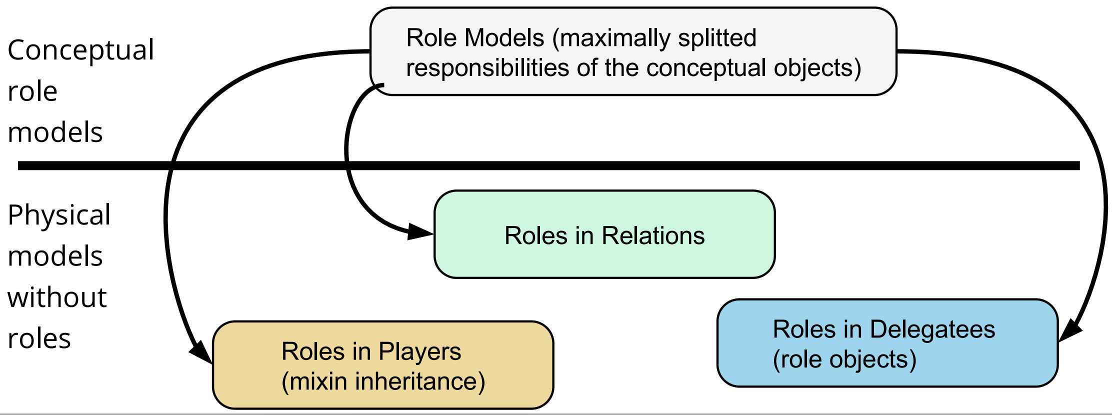
Referenzen weg

Good Mapping of Conceptual Role Models to Physical Class Models

- Role instances must be
  - embedded into core objects
  - or become physical role objects
- **Role mapping:** Mapping conceptual role types to physical implementation-records is an *Embedding Decision*
- For one conceptual model, many alternative phyisical models

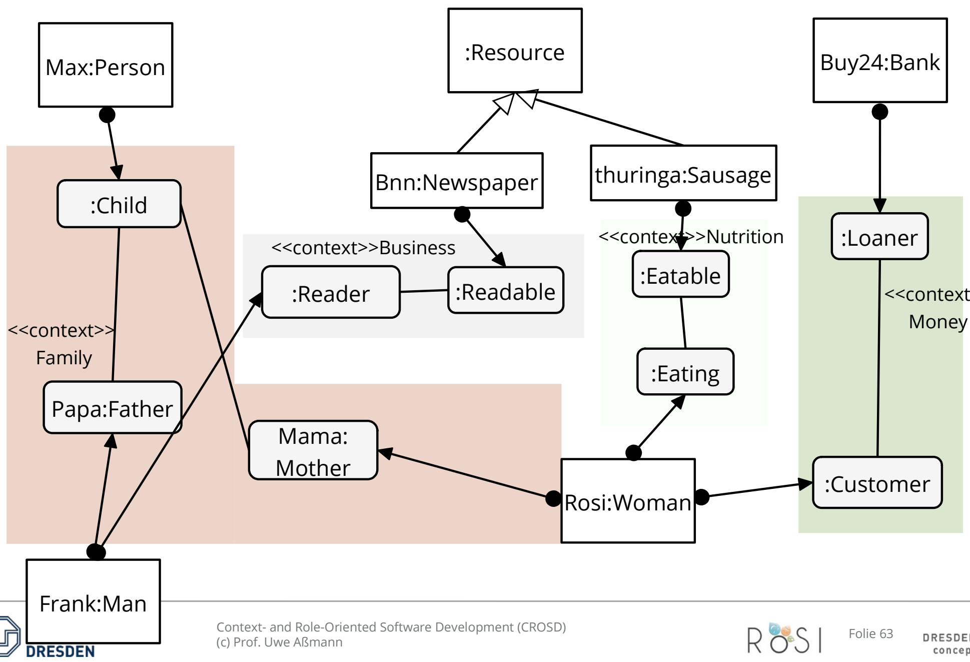
Computing Physical Representation from Conceptual Models

- Role embedding determines, which roles are embedded into which physical objects

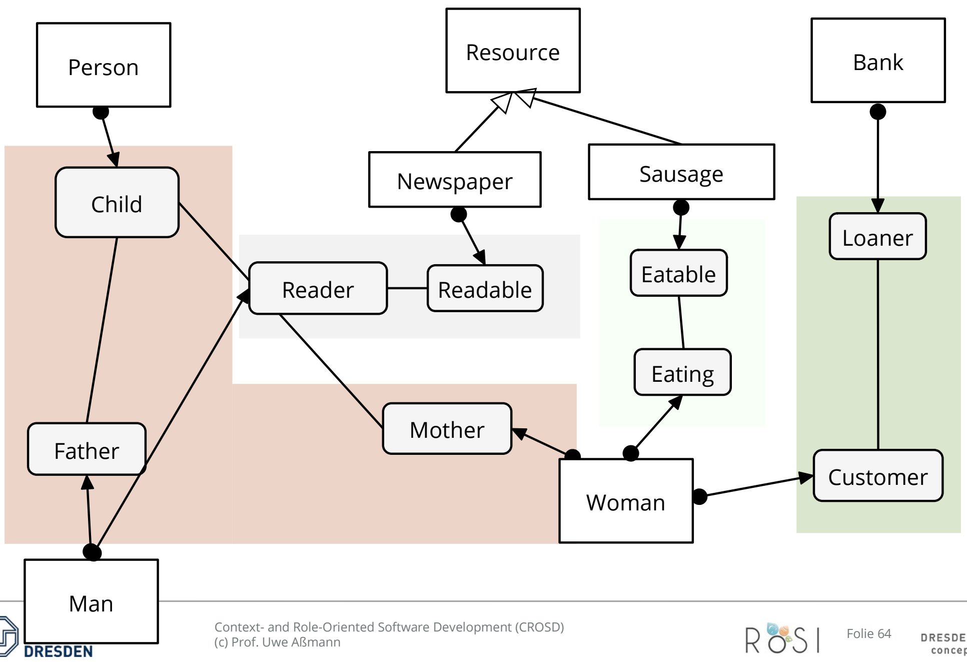

Conceptual role models
Physical models without roles
Role Models (maximally splitted responsibilities of the conceptual objects)
Roles in Relations
Roles in Players (mixin inheritance)
Roles in Delegatees (role objects)

4.5.3 Role-Mapping MDA with Scenario „Families and Banks“

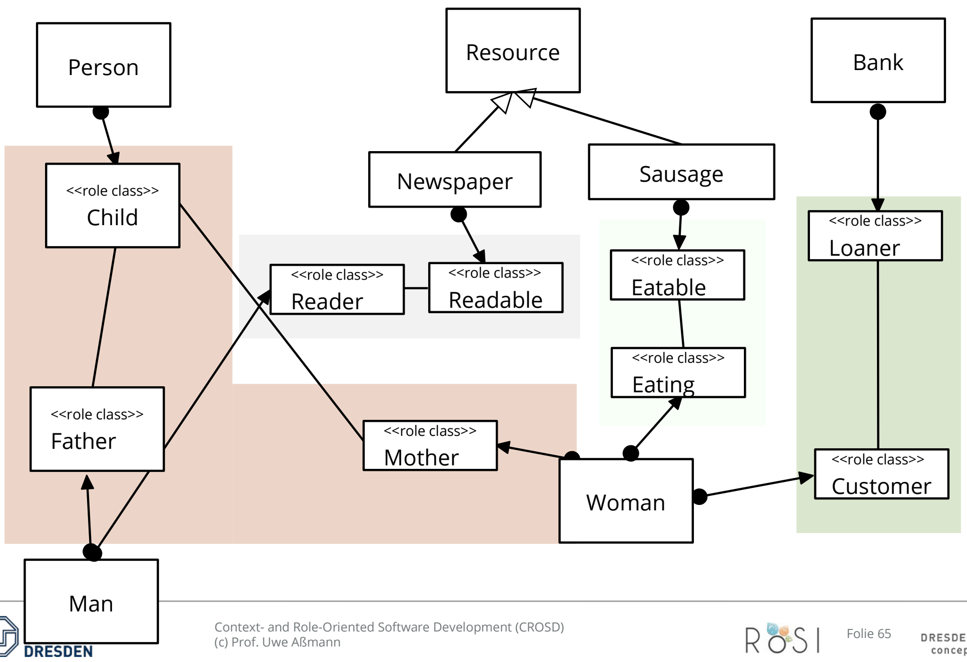
Families, Resources and Banks (Snapshot, Object-Role Model)



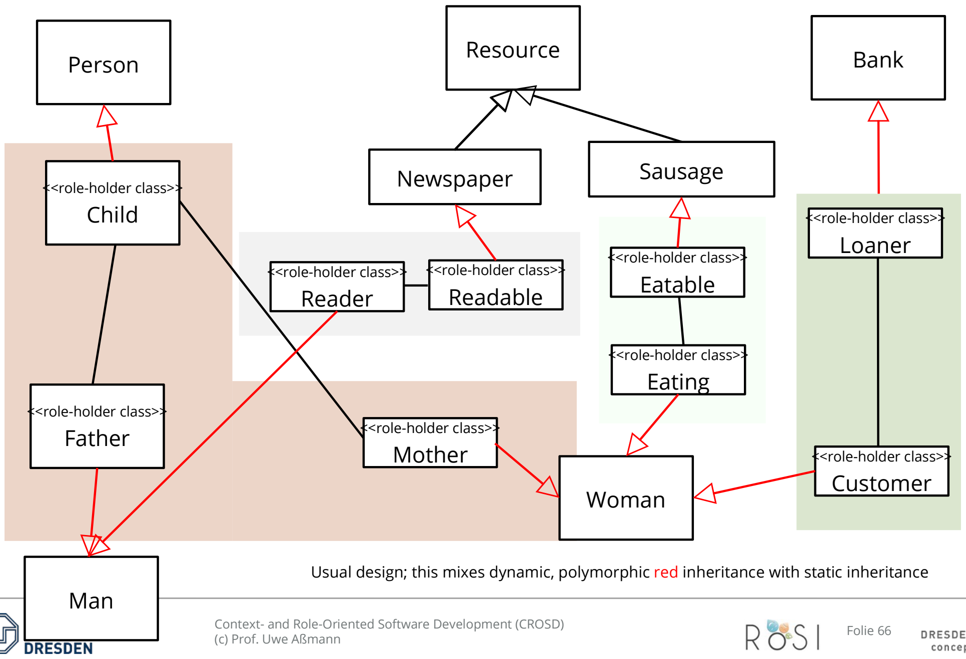
Families and Banks in Natural and Role Types



Implement „Families and Banks“ (Delegation to Role Objects - „Split Design“)

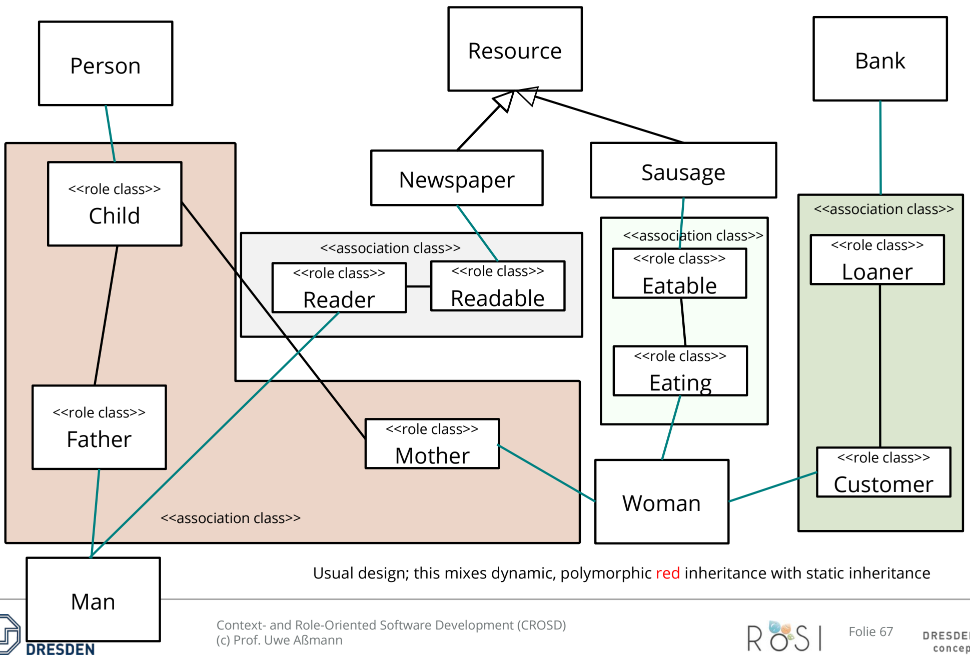


Implement „Families and Banks“ (Delegation to Role Objects – Design „Inheritance Embeds Roles in Players“)


Person
Resource
Bank
Newspaper
Sausage
<<role-holder class>> Child
<<role-holder class>> Reader
<<role-holder class>> Readable
<<role-holder class>> Eatable
<<role-holder class>> Loaner
<<role-holder class>> Eating
<<role-holder class>> Father
<<role-holder class>> Mother
Woman
<<role-holder class>> Customer
Man

Usual design; this mixes dynamic, polymorphic red inheritance with static inheritance

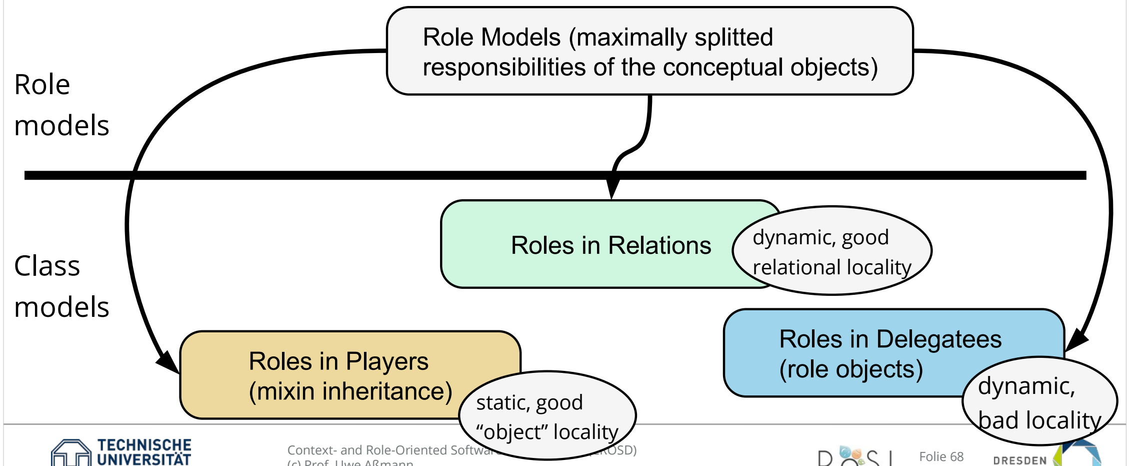
Implement „Families and Banks“ (Delegation to Role Objects – Design „Roles Embedded in Relations“)



Usual design; this mixes dynamic, polymorphic red inheritance with static inheritance

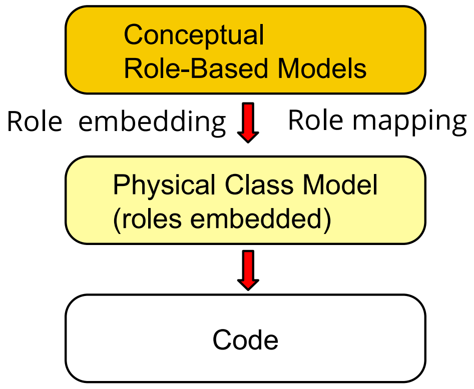
Scalable Binding Times of Contexts

- Problematic: Role mapping fixes binding time


Role Models (maximally splitted responsibilities of the conceptual objects)
Role models
Class models
Roles in Relations
dynamic, good relational locality
Roles in Players (mixin inheritance)
static, good "object" locality
Roles in Delegatees (role objects)
dynamic, bad locality

The Role-Mapping Process and Model-Driven Architecture

- The question "Where is a role embedded?" is a *platform decision* in Model-Driven Architecture (MDA)
  - A role model is more *platform independent* than a class model
- → Role mapping is a task in Model-Driven Architecture (MDA)


Conceptual Role-Based Models
Role embedding
Role mapping
Physical Class Model (roles embedded)
Code

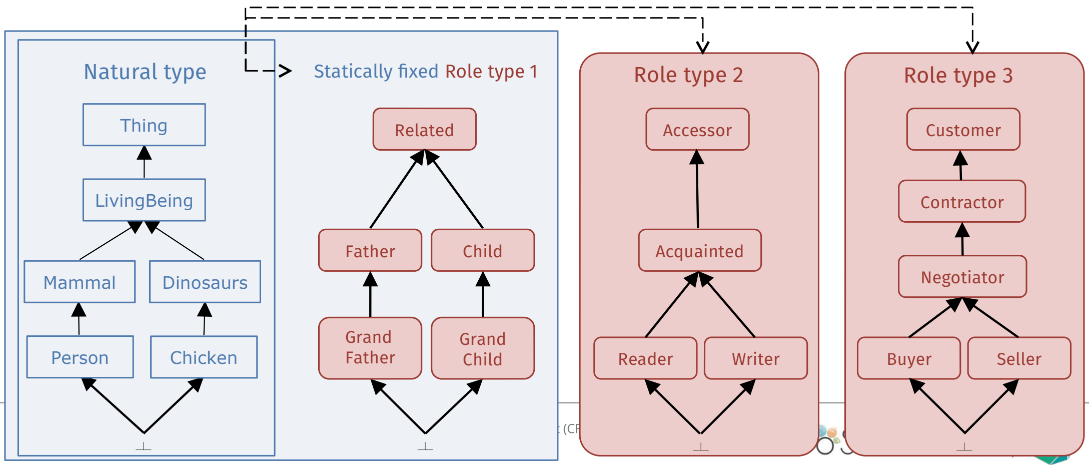
Role Mapping MDA Yields Scalability

- From one conceptual role-based design, derive via Role-MDA:
  - many physical designs
  - many run-time behaviors with different QoS
- When to embed?
  - At compile-time
  - At run-time
- Tuning and optimization possible

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

How to Achieve Scalable Binding Times of Contexts

- **Scalability**: Roles and their contexts can be statically bound
- Effects on Life-time, aliases and dependencies, cohesion, allocation, adaptation, reconfiguration



OPTIONAL

TECHNISCHE UNIVERSITÄT DRESDEN

RoSI

2.4. Roles are a Concept for Language Modeling and Language Engineering

The IRDS/MOF Metamodelling Hierarchy

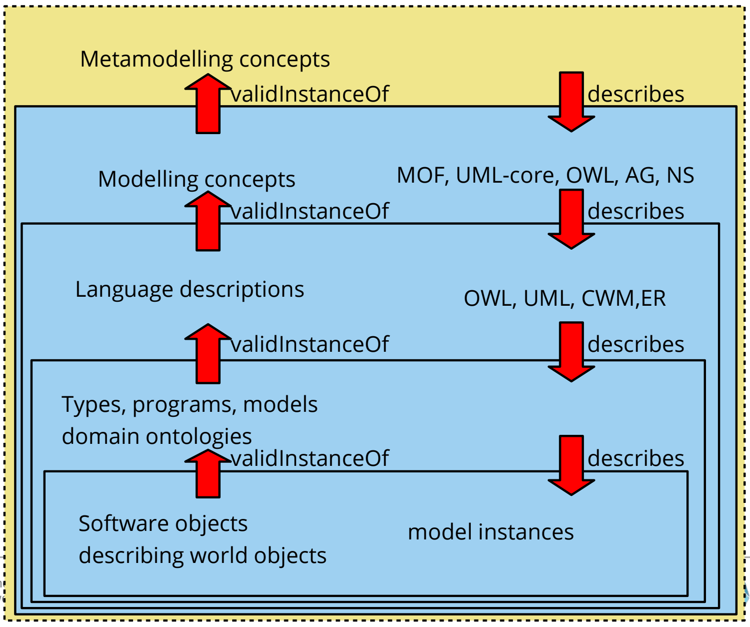
M4 level = M3

M3 metametamodel level

M2 metamodel level

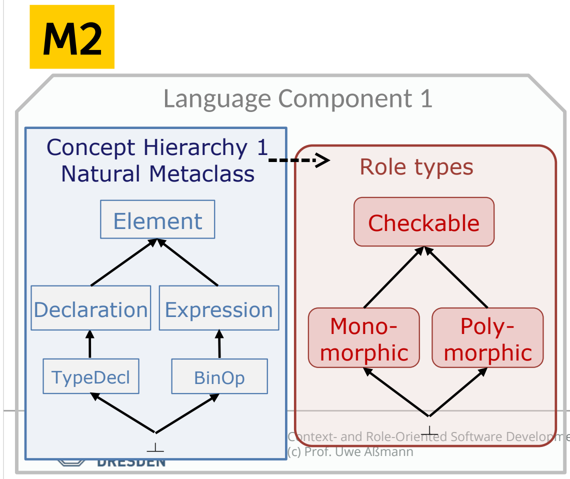
M1 model level

M0 Object level



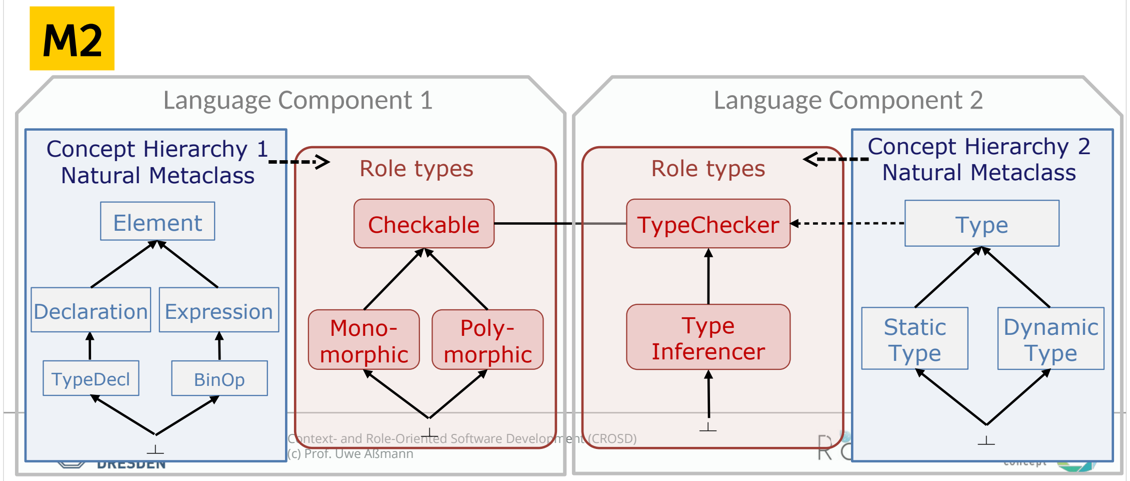
Context-Based Modelling of Languages on M2

- Role-types factor concept hierarchies into context-free and context-dependent features
- Improved separation of concerns
- [Wende] PhD Thesis



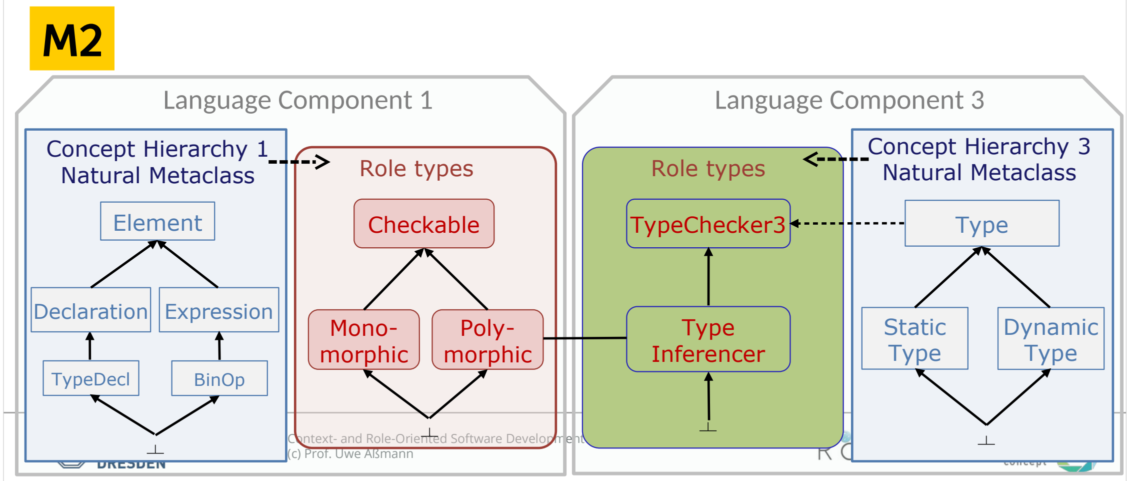
Context-Based Modelling of Languages on M2

- Context-dependent features can easily be exchanged



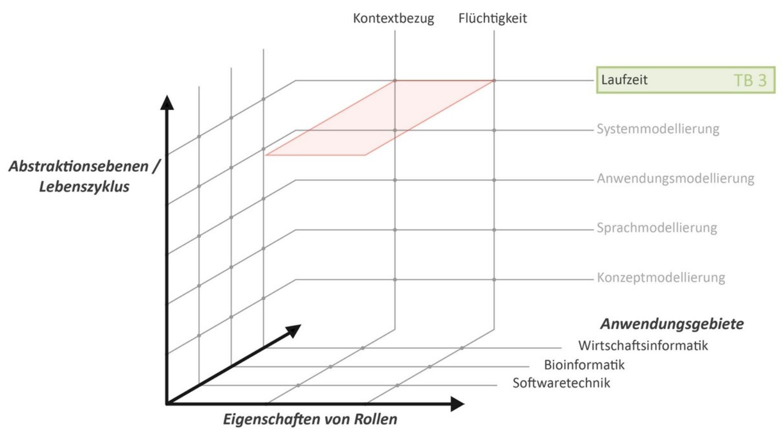
Context-Based Modelling of Languages on M2

- Modular languages
  - Domain-specific languages
  - Ontologies



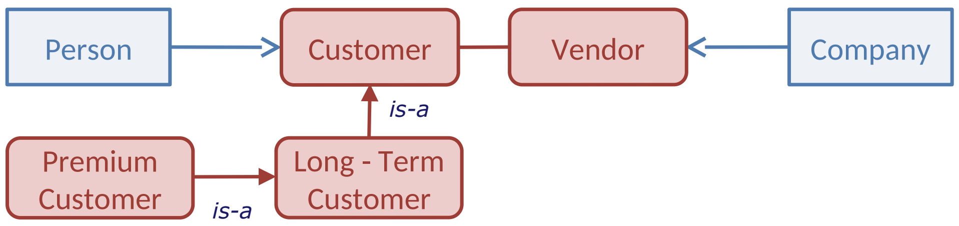
2.3.3 Roles are a Concept for Run-Time Infrastructures

Objective 3: Investigation of Context-Based and Fluid Run-Time-Infrastructures



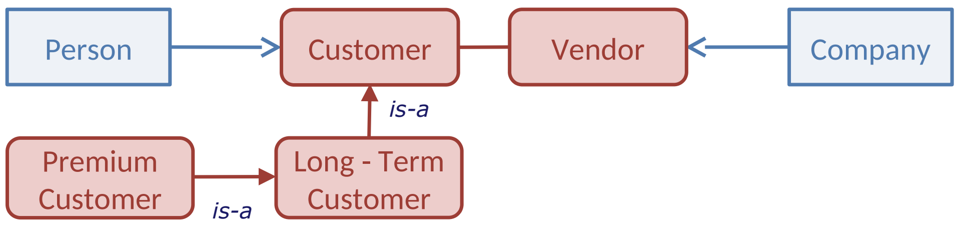
Context-Based and Fluid Run-Time Features

- Fluid complex objects can be dynamically reconfigured
- Context-dependent run-time behavior
- Fine-grained monitoring, persistency, adaption



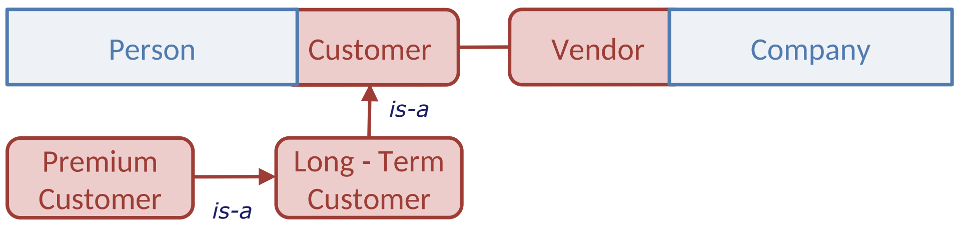
Dynamic Mixins

- Can role types be *mixed into* core types at run-time?



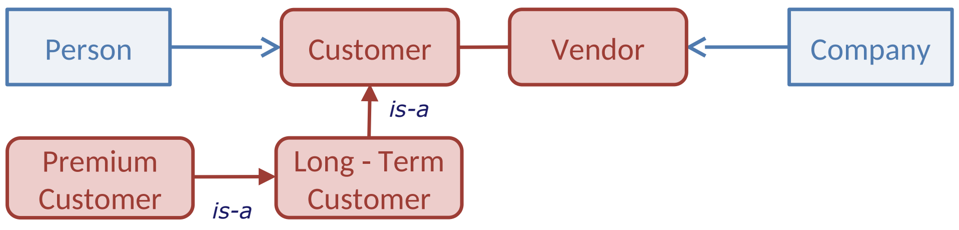
Dynamic Mixins

- Can role objects be *mixed into* core objects at run-time?
- Yes – by memory compaction in JIT recompilation



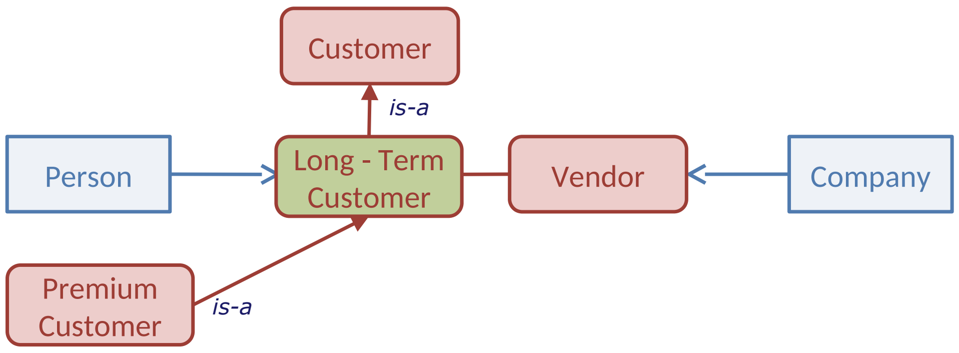
Dynamic Mixins

- But role instances can also be *outlined* again



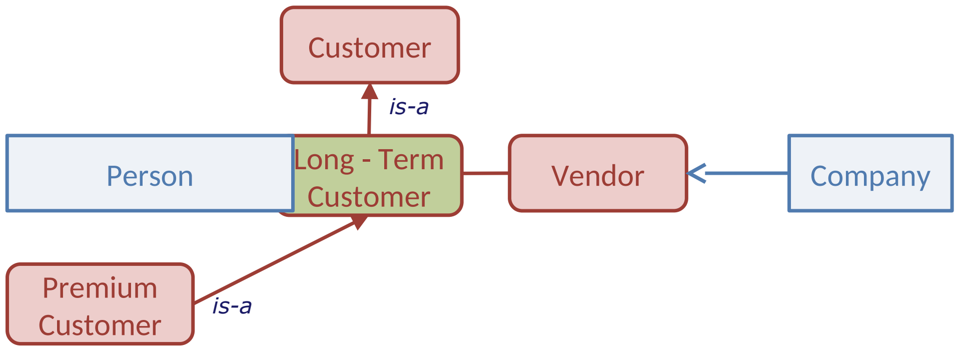
Dynamic Mixins

- But role instances can also be *outlined* again
- To change the role type



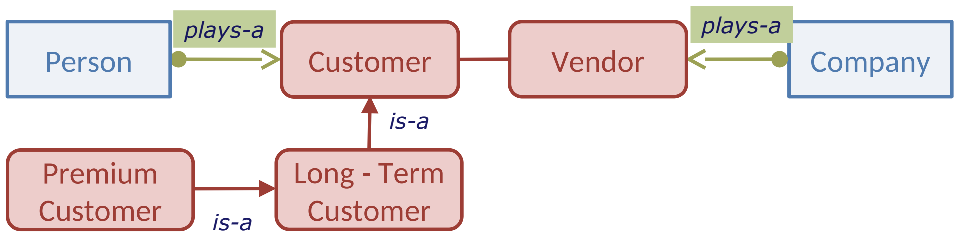
Dynamic Mixins

- And then re-inlined (dynamic mixin)
  - by memory compaction during JIT re-compilation



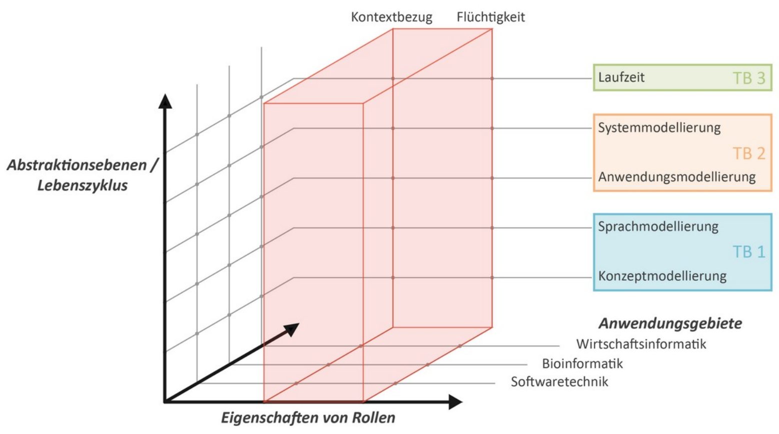
Dynamic Mixins

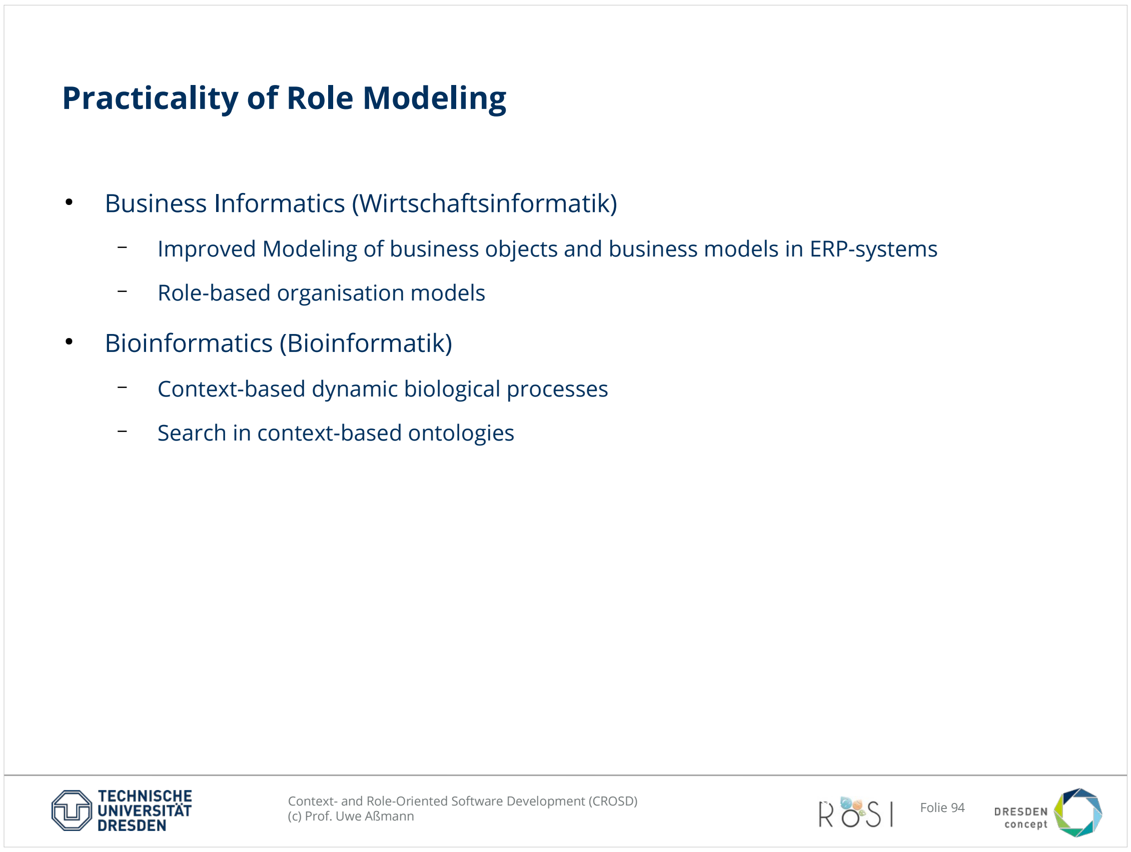
Role-based run-time infrastructures can optimize locality of roles dynamically
by dynamic mixins and recompilation



2.5. Roles are a Practical Concept

Objective 4: Practicality in Application Areas

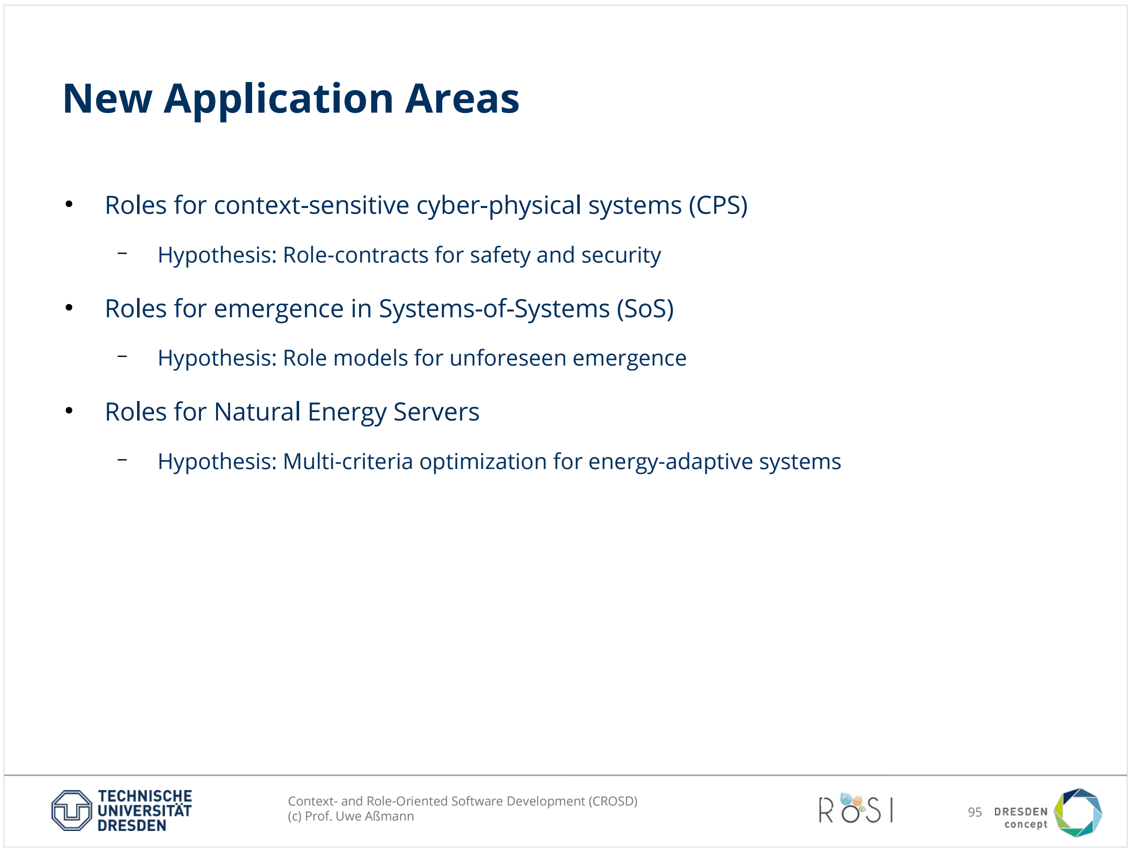




Practicality of Role Modeling

- Business Informatics (Wirtschaftsinformatik)
  - Improved Modeling of business objects and business models in ERP-systems
  - Role-based organisation models
- Bioinformatics (Bioinformatik)
  - Context-based dynamic biological processes
  - Search in context-based ontologies

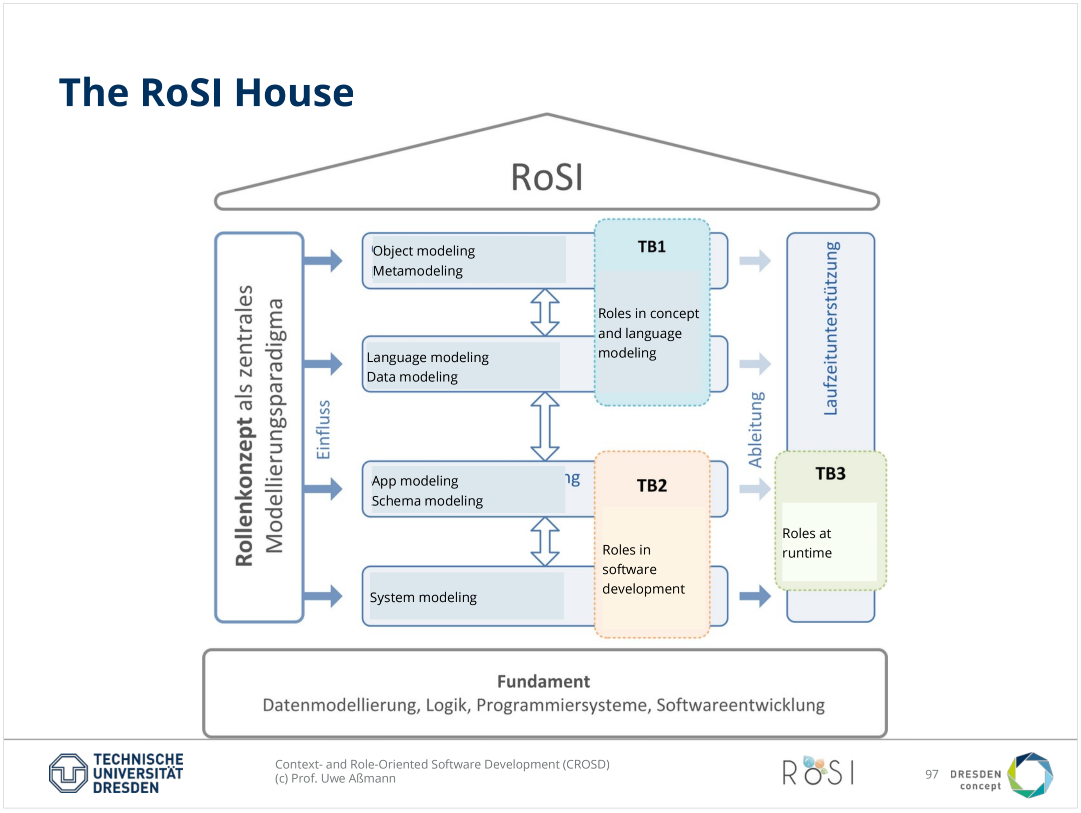
Querschneidende Arbeiten hier gruppieren



New Application Areas

- Roles for context-sensitive cyber-physical systems (CPS)
  - Hypothesis: Role-contracts for safety and security
- Roles for emergence in Systems-of-Systems (SoS)
  - Hypothesis: Role models for unforeseen emergence
- Roles for Natural Energy Servers
  - Hypothesis: Multi-criteria optimization for energy-adaptive systems

Verschmelzen in “neue Herausforderungen”



The RoSI House

RoSI
Rollenkonzept als zentrales Modellierungsparadigma
Einfluss
Object modeling
Metamodeling
Language modeling
Data modeling
App modeling
Schema modeling
System modeling
TB1
Roles in concept and language modeling
TB2
Roles in software development
TB3
Roles at runtime
Ableitung
Laufzeitunterstützung
Fundament
Datenmodellierung, Logik, Programmiersysteme, Softwareentwicklung

Themenbereich 1 (TB1 - Rollen in der Konzept- und Sprachmodellierung) widmet sich den Metaebenen M3 und M2.

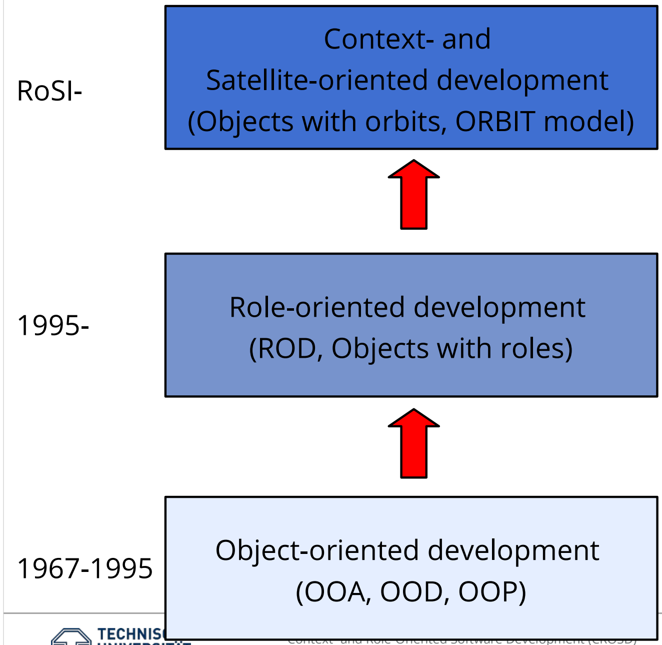
Arbeiten in diesem Themenbereich untersuchen die Begründung und Definition des Rollenbegriffs und seine Einbettung in den verschiedenen Sprachen (Modellierungssprachen, Datendefinitions- und -abfragesprachen, Programmiersprachen) der Softwareentwicklung.

Themenbereich 2 (TB2 - Rollen in der Softwareentwicklung) konzentriert sich auf die Verwendung des Rollenbegriffs auf Objektebene

die Grundlagen der Anwendungsentwicklung (Anwendungsmodellierung, Schemaentwurf, Systemmodellierung) mit Rollen.

Themenbereich 3 (TB3 - Rollen zur Laufzeit) betrachtet die Verwendung des Rollenbegriffs und rollenspezifischer Modelle zur Laufzeit (Instanzebene) und deren Auswirkung.

Ladder of Paradigms (ctd)


RoSI-
Context- and Satellite-oriented development (Objects with orbits, ORBIT model)
1995-
Role-oriented development (ROD, Objects with roles)
1967-1995
Object-oriented development (OOA, OOD, OOP)

E. W. Dijkstra “On the Role of Scientific Thought”, EWD 447 Selected Writings on Computing: A Personal Perspective, pages 60–66, 1982.

"Let me try to explain to you, what to my taste is *characteristic for all intelligent thinking.*

It is, that one is willing to study in depth an aspect of one's subject matter in isolation for the sake of its own consistency, all the time knowing that one is occupying oneself only with one of the aspects.

The End

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.
- 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.
- Nicola Guarino Chris Welty. Supporting ontological analysis of taxonomic relationships. Data and Knowledge Engineering, 39:51-74, 2001.
- 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

Important References for Role-Based Modeling

- 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.
- 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.

Role-Based Programming

- 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, 2007.
- www.objectteams.org: a Java-based programming language with roles

Works at SMT

AOSD, MDD:

- U. Aßmann, S. Zschaler, and G. Wagner. Ontologies, Meta-Models, and the Model-Driven Paradigm, Handbook on Ontologies and Software Engineering. pages 249–273. Springer, 2006.
- J. Henriksson, J. Johannes, S. Zschaler, U. Aßmann. Reuseware – adding modularity to your language of choice. Proc. of TOOLS EUROPE 2007: Spec Iss Journal of Object Technology, 2007.

Roles and aspects in ontologies and metamodeling:

- 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.
- Matthias Bräuer and Henrik Lochmann. Towards Semantic Integration of Multiple Domain-Specific Languages Using Ontological Foundations.

Works at PhD Theses ST (all available via www.qucosa.de)

- Mirko Seifert. Designing Round-Trip Systems by Model Partitioning and Change Propagation. PhD thesis, Dresden University of Technology, June 2011.
  - Shows how roles simplify round-trip engineering by partitioning data
- Sebastian Richly. Autonom rekonfigurierbare Workflows. PhD thesis, Dresden University of Technology, December 2011.
  - Shows how roles can be used to provide an extensible tool platform
- Christian Wende. Language Family Engineering. PhD thesis, Dresden University of Technology, March 2012.
  - Shows how roles can be used to do context-based language composition
- Max Leuthäuser. A Pure Embedding of Roles - Exploring 4-dimensional Dispatch for Roles in Structured Contexts. PhD thesis, Technische Universität Dresden, August 2017.
  - This PhD thesis developes a programming language for contexts and roles, based on some implementation patterns and the base language Scala.
- Thomas Kühn. A Family of Role-Based Languages. PhD thesis, Technische Universität Dresden, March 2017.
  - This PhD develops language design with contexts and roles in CROM
- Georg Püschel. Testing Self-Adaptive Systems - A Model-based Approach to Resilience. PhD thesis, Technische Universität Dresden, June 2018.
  - Contexts for testing robots

- Matthias Schmidt, Jan Polowinski, Jendrik Johannes, and Miguel A. Fernández. An integrated facet-based library for arbitrary software components. In Thomas Kühne, Bran Selic, Marie-Pierre Gervais, and Francois Terrier, editors, ECMFA, volume 6138 of Lecture Notes in Computer Science, pages 261-276. Springer, 2010.

Best paper awards:

- C. Piechnick, S. Richly. Using Role-Based Composition to Support Unanticipated, Dynamic Adaptation, ADAPTIVE 2012
- J. Reimann, M. Seifert, U. Aßmann. Role-based generic model refactoring. MODELS Okt. 2010