41. Family of Role-Based (Meta-)Models

in the Research Training School on
Role-oriented Software Infrastructures (RoSI)

4. Roles in Other Technical Spaces
5. Family of Role-based Languages

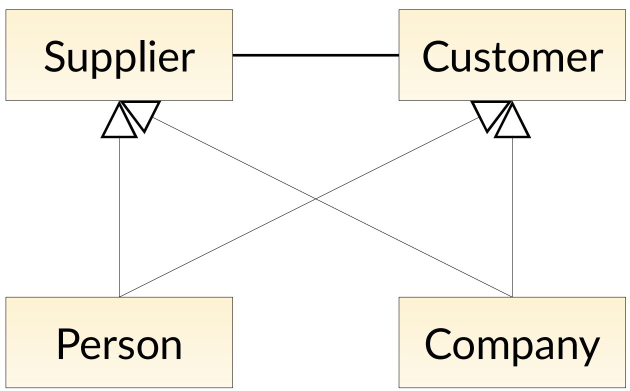
DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

Recap Role-Based (Meta-)Modeling

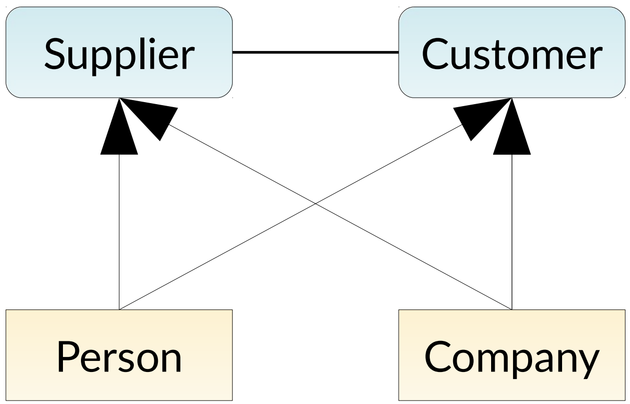
Limitations of Object-Oriented Design

Supplier/Customer Problem

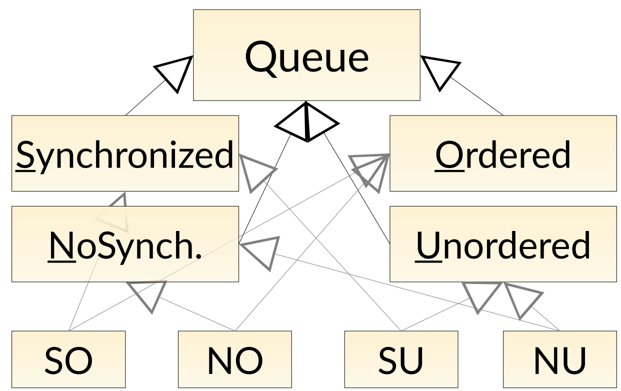
[Steimann2000]



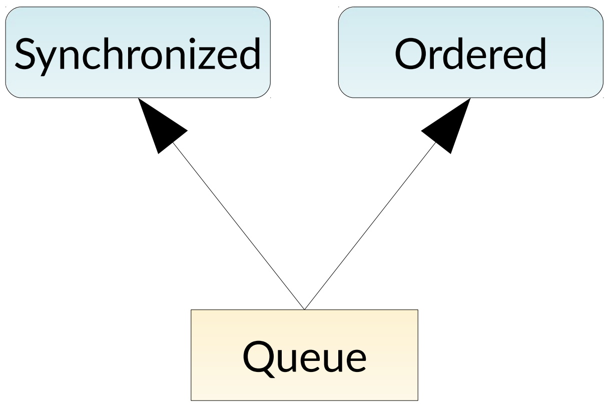
- Multiple entities fulfill the same roles



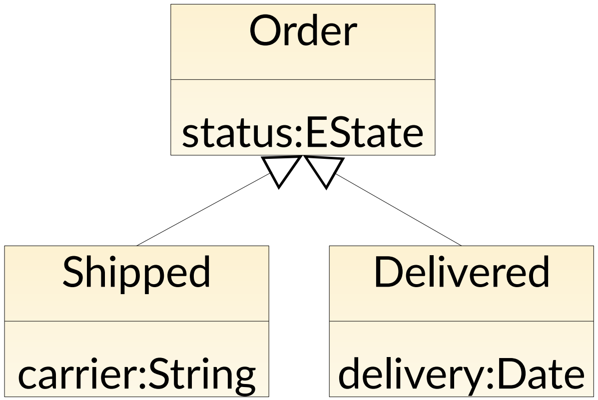
Multiple Classification



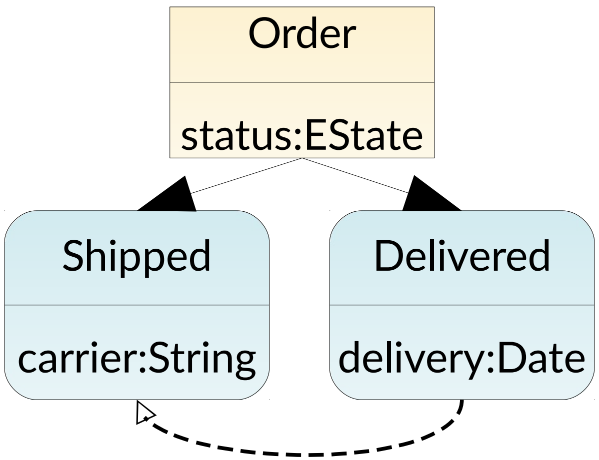
- Entity subject to multiple classifying features



State-Dependence

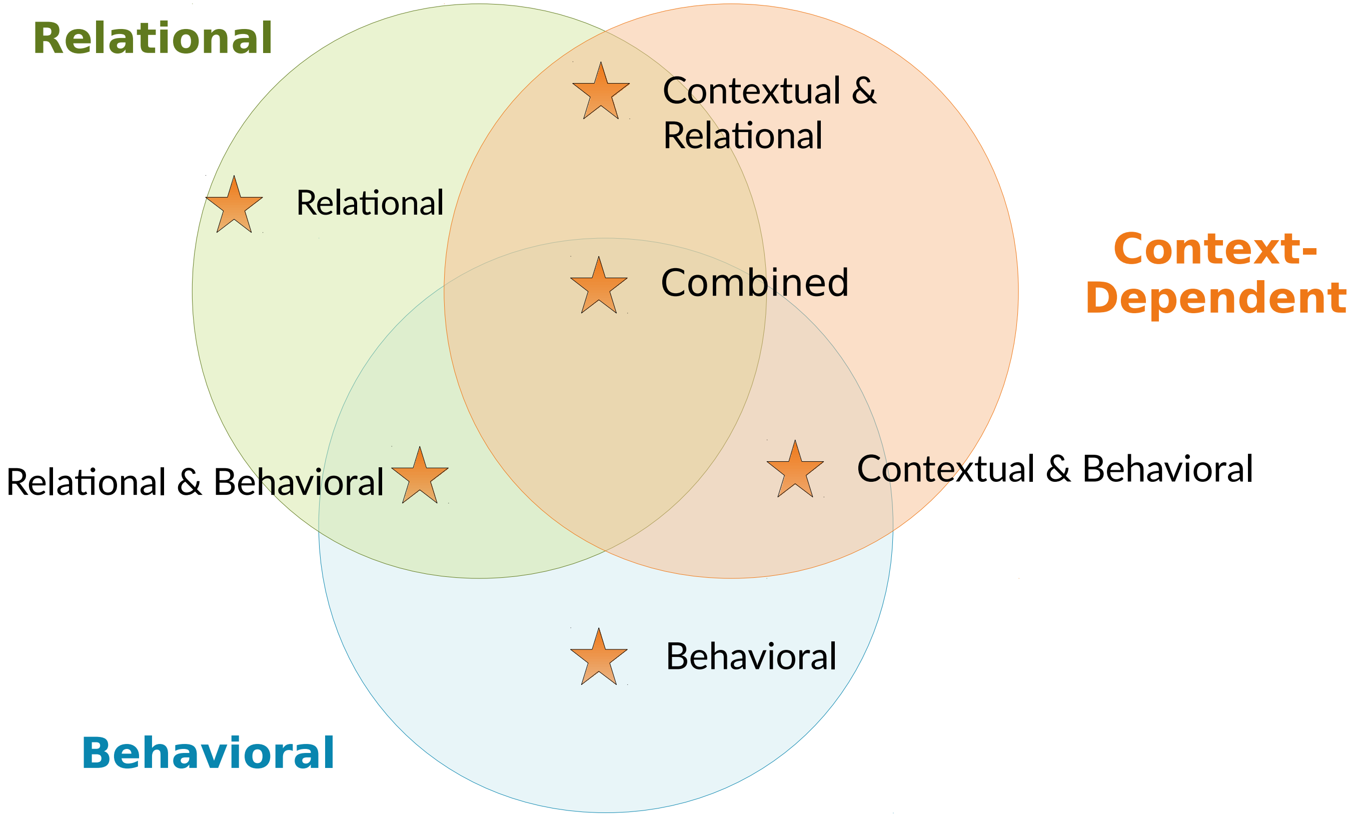


- Specialization of entity depends on state



Recap Role-Based (Meta-)Modeling
Roles in Modeling and Programming Languages

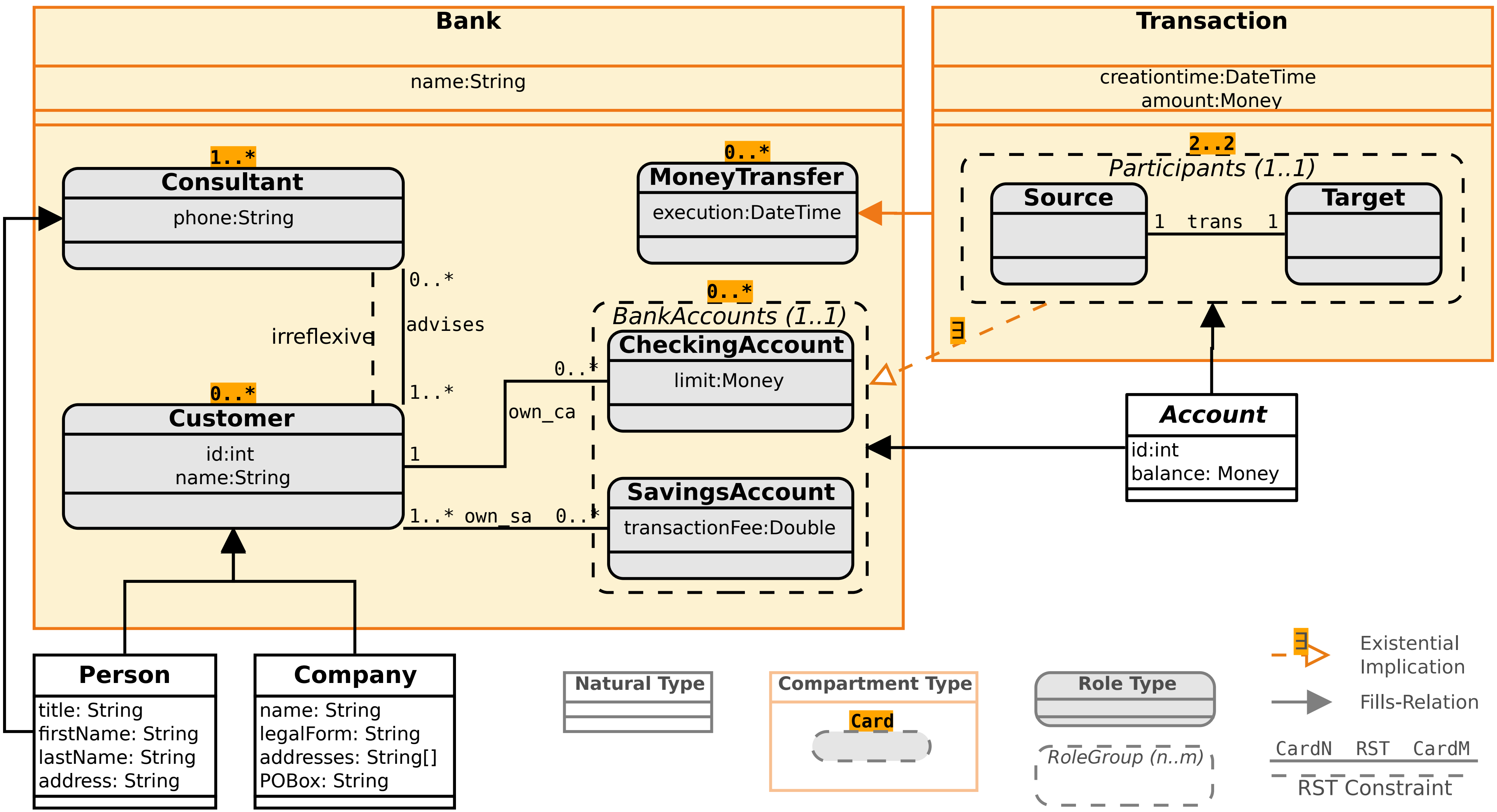
- *Structured Literature Review* of publications since 2000
- Published by the big four (i.e., *Springer, IEEE, ACM, Science Direct)*



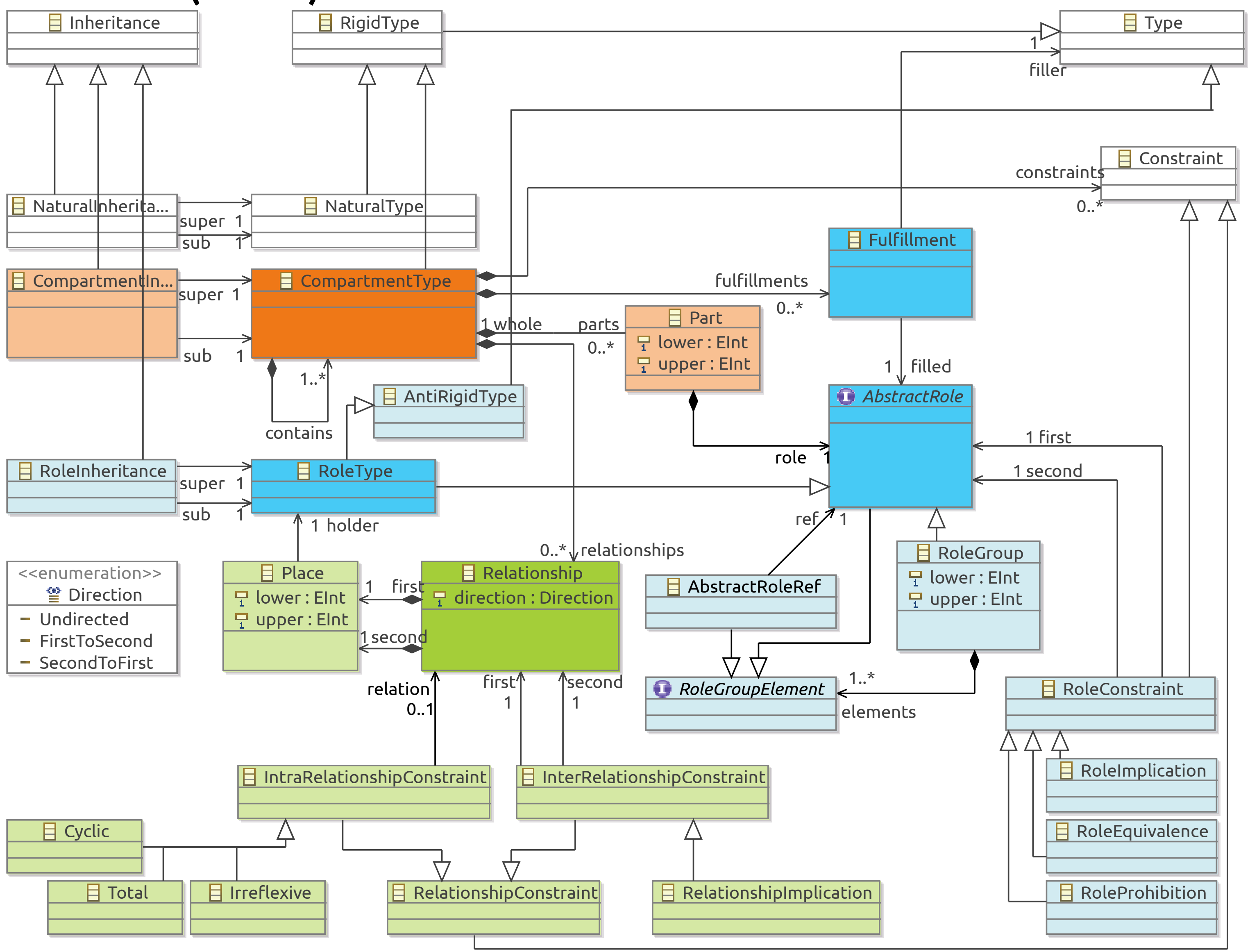
Research Field suffers from *fragmentation* and *discontinuity*

Recap Role-Based (Meta-)Modeling
The Compartment Role Object Model (CROM)

Example: Banking Application

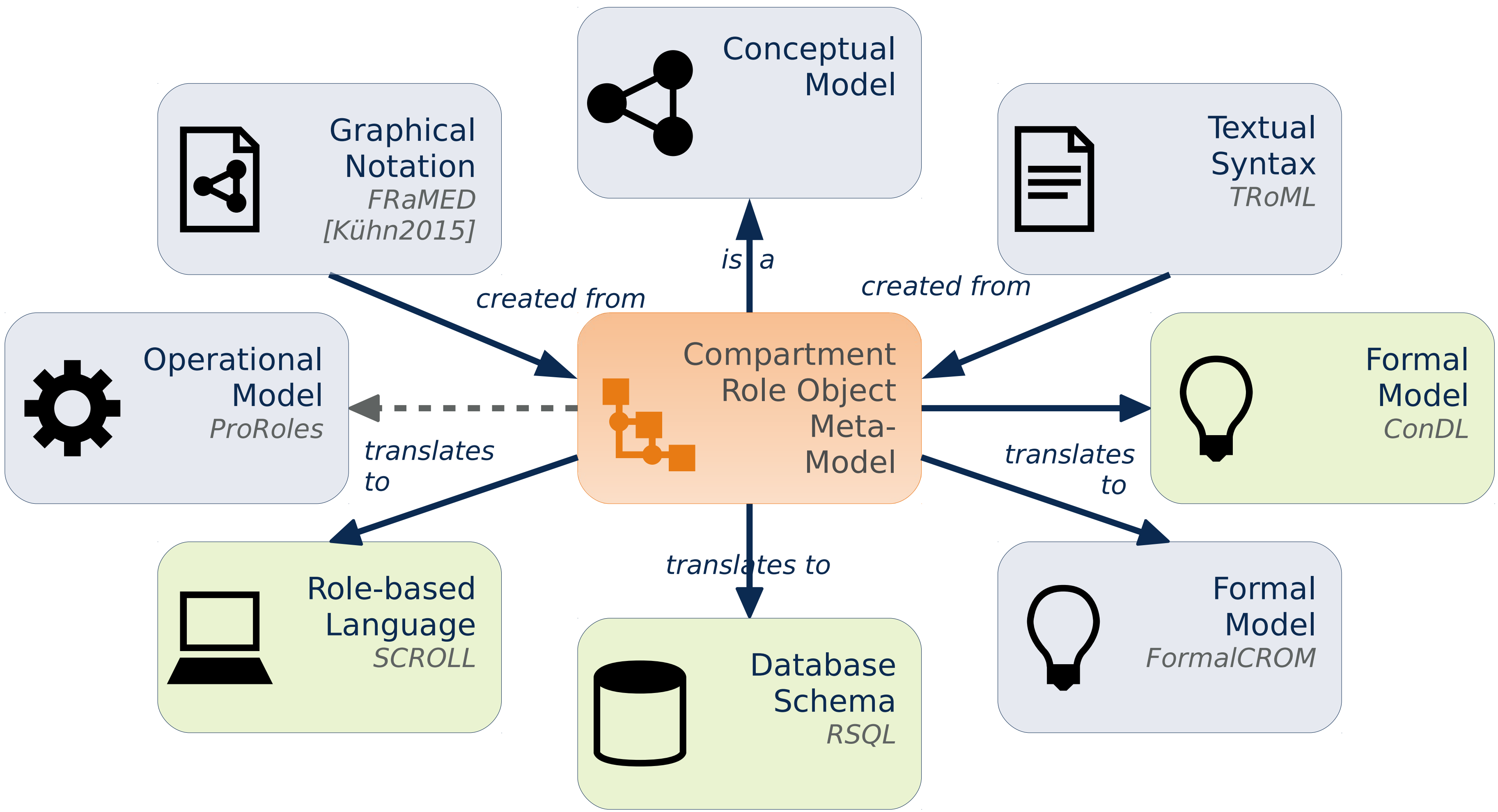


CROM EMOF (Ecore) Metamodel



Recap Role-Based (Meta-)Modeling

Tool Support Surrounding CROM



41.4. Roles in Other Technical Spaces

Prof. Dr. Uwe Aßmann

Dr.-Ing. Thomas Kühn

Technische Universität Dresden

Institut für Software- und Multimediatechnik

http://st.inf.tu-dresden.de/teaching/most

Version 16-1.0, 18.12.17

Literature

[Leuthäuser2015] Enabling View-based Programming with SCROLL: Using Roles and Dynamic Dispatch for Establishing View-based Programming
Max Leuthäuser and Uwe Aßmann
MORSE/VAO '15, ACM (2015)

[Jäkel2016] Towards a Contextual Database
T. Jäkel, T. Kühn, H. Voigt, and W. Lehner
ADBIS (2016)

[Böhme2017] Reasoning on Context-Dependent Domain Models
S Böhme, T. Kühn
Proceedings of the JIST (2017)

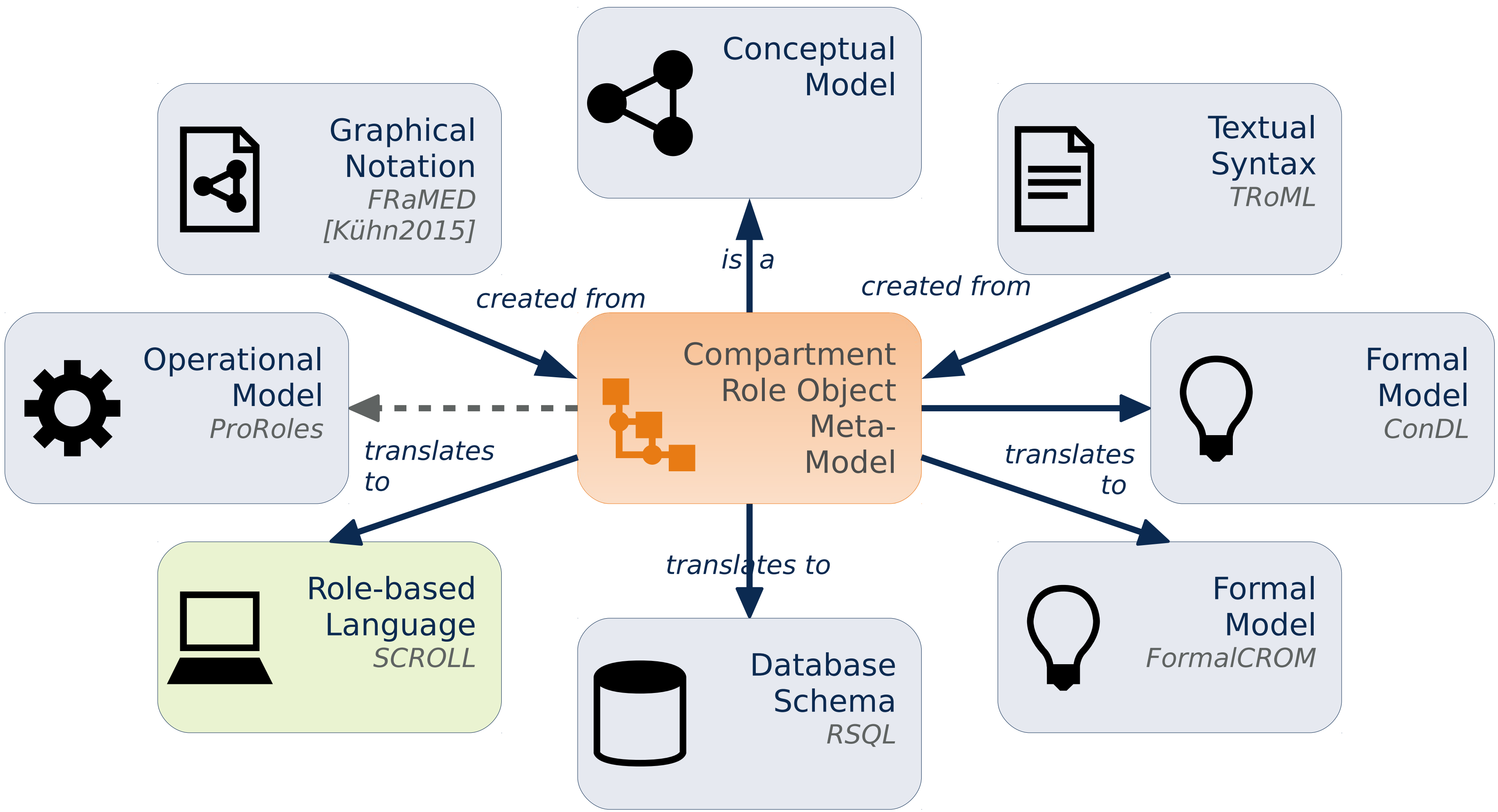
References

[Hirschfeld2008] Context-oriented programming
R. Hirschfeld, P. Costanza, and O. Nierstrasz
Journal of Object technology 7:3 (2008).

[Meijer2004] Static typing where possible, dynamic typing when needed: The end of the cold war between programming languages
E. Meijer and P. Drayton
OOPSLA (2004)

Roles in Other Technical Spaces

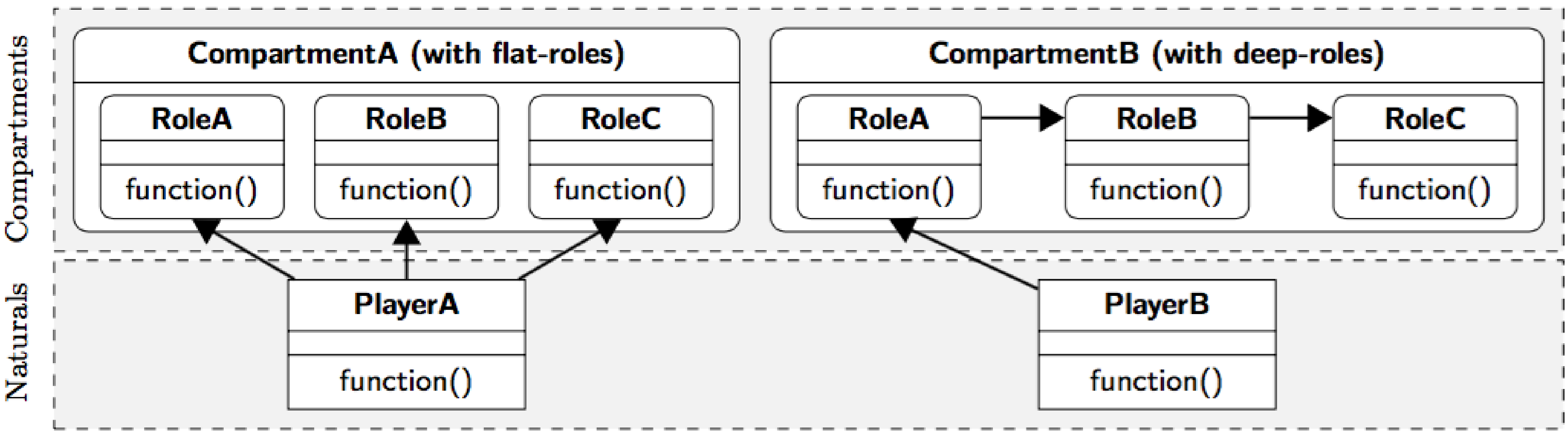
Overview



Roles in Other Technical Spaces
Role-based Programming with SCROLL

Issue of Role-based Software Systems

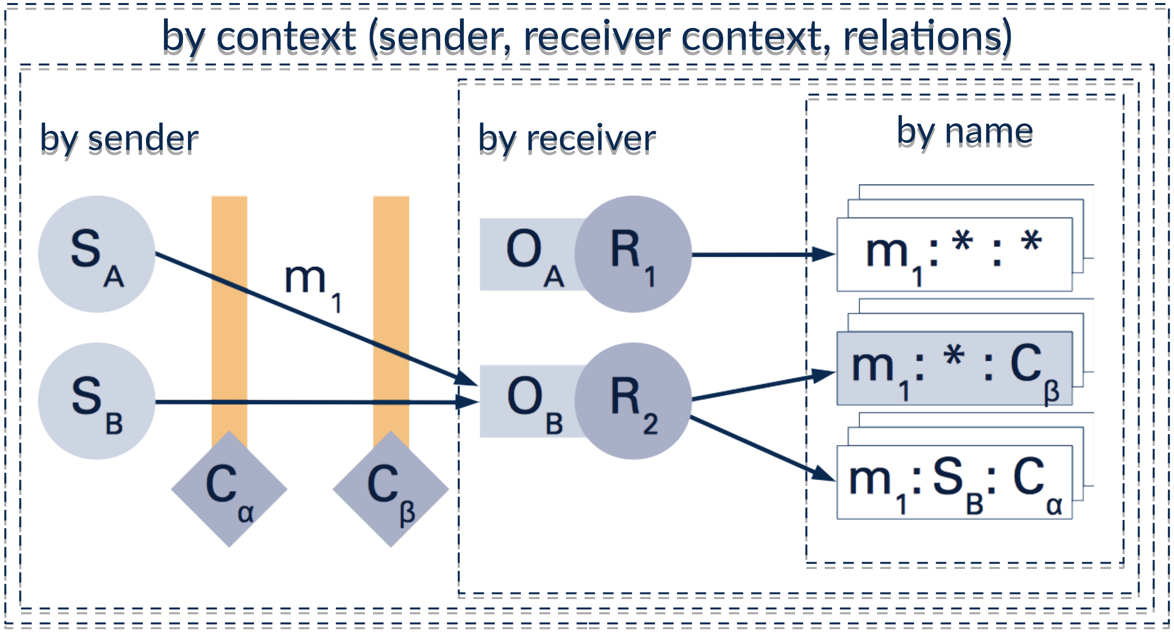
- Ambiguity of object's behavior and role's behavior
    - Object playing multiple roles adapting the same behavior
    - Object playing instances of the same role type in different compartments



Slides prepared by Max Leuthäuser

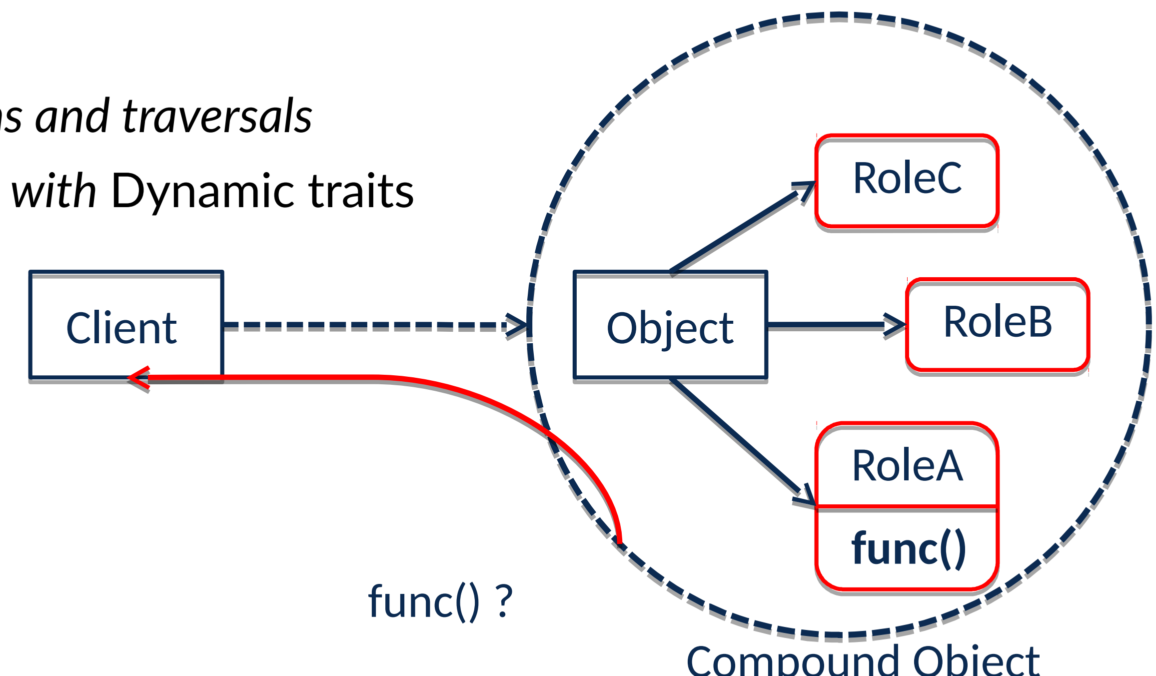
Four Dimensional Dispatch *[Hirschfeld2008]*

- Dispatch: Discover the correct computational unit utilizing the *type system* and *relationship information*
  - 1D address computational unit with a name
  - 2D 1D + receiver
  - 3D 2D + sender
  - 4D 3D + context



SCala RoLes Library (SCROLL) *[Leuthäuser2014]*

- Lightweight Library for role-oriented programming[1]
- Embedded DSL for
  - *Compartment* and *Role Type* declaration
  - Definition of *role constraints*
  - *Role Playing Automaton* defining a role's life cycle
- Customizable 4D-dispatch based on declarative description
- Based on SCALA and utilizing:
  - *Directed acyclic graphs and traversals*
  - *Compiler rewrite rules with* Dynamic traits
  - *Implicit conversions*



1) https://github.com/max-leuthaeuser/SCROLL

Example Banking Application

```
object BankExample extends App
{
  // Naturals
  case class Person(name: String)
  case class Company(name: String)
  class Account(var balance: Double = 0)
  {
    def increase(amount: Double) {
        balance = balance + amount
      }
    def decrease(amount: Double) {
        balance = balance – amount
      }
  }
}
```

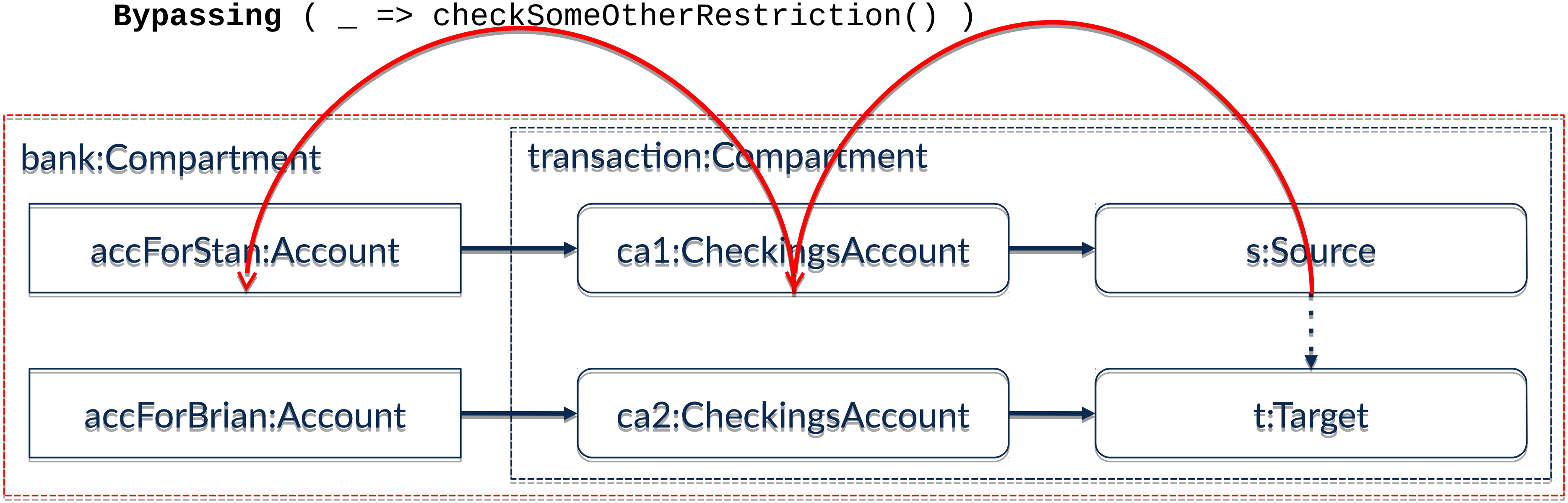
```
// Compartment and Roles
class Bank extends Compartment
{
  @Role case class Customer()
  @Role class CheckingsAccount() {
    def decrease(amount: Double) {
       (-this).decrease(amount)
     }
  }
  @Role class SavingsAccount() {
    private def transactionFee
      (amount: Double) = amount * 0.1

    def decrease(amount: Double) {
      (-this).decrease(amount -
                transactionFee(amount))
      }
  }
}
```

Declarative Dispatch Description

- Based on graph traversal operators

```
// implicits are passed as additional, hidden argument to method invocations
implicit val dd =
    From ( _ is[Account] ).
    To ( _ is[Source] ).
    Through ( _ => checkSomeRestriction() ).
    Bypassing ( _ => checkSomeOtherRestriction() )
```



Roles in Other Technical Spaces
Role-oriented Programming with SCROLL

"Static typing where possible, dynamic typing when needed!"

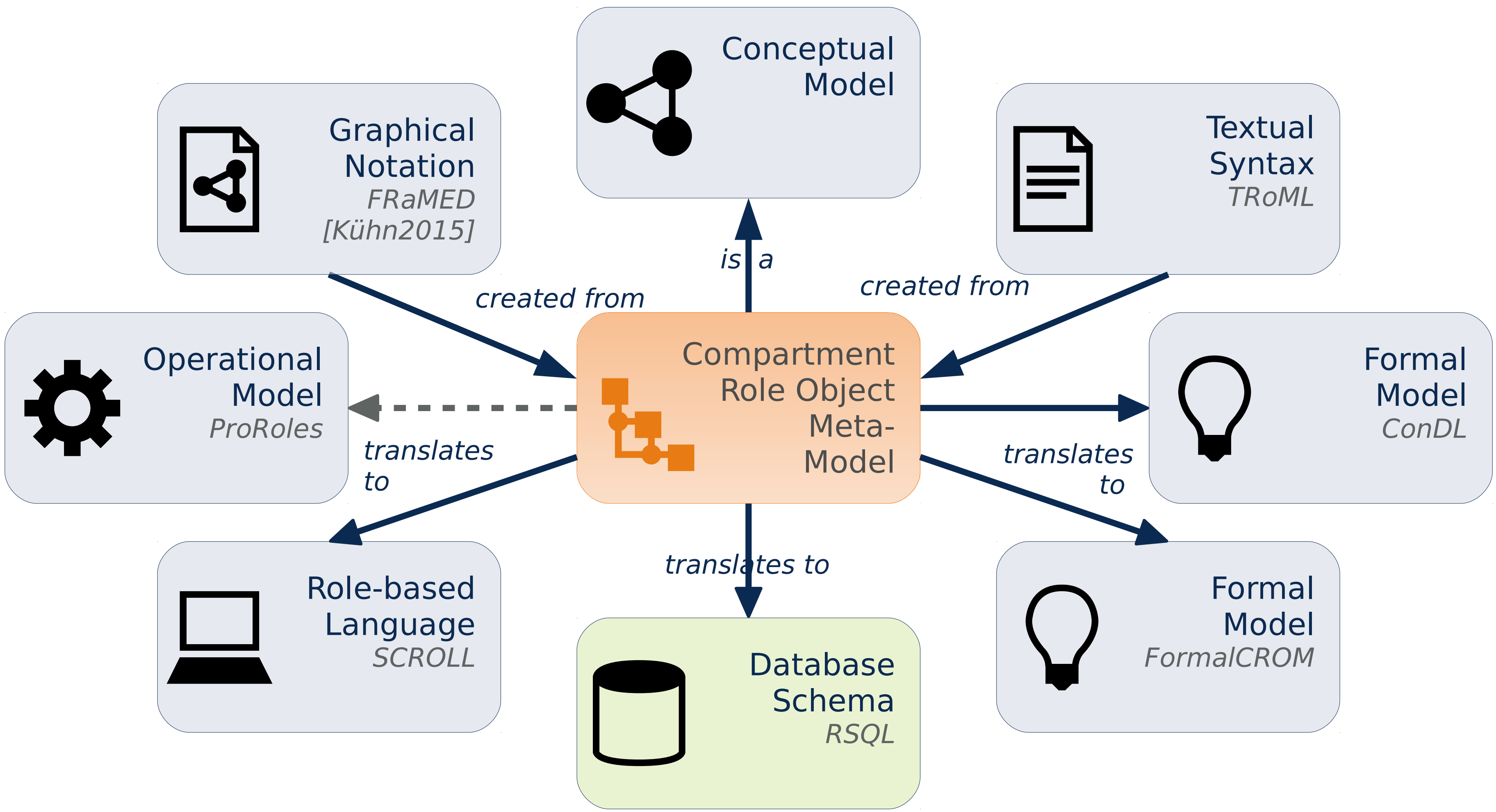
– Eric Meijer *[Meijer2004]*

Summary

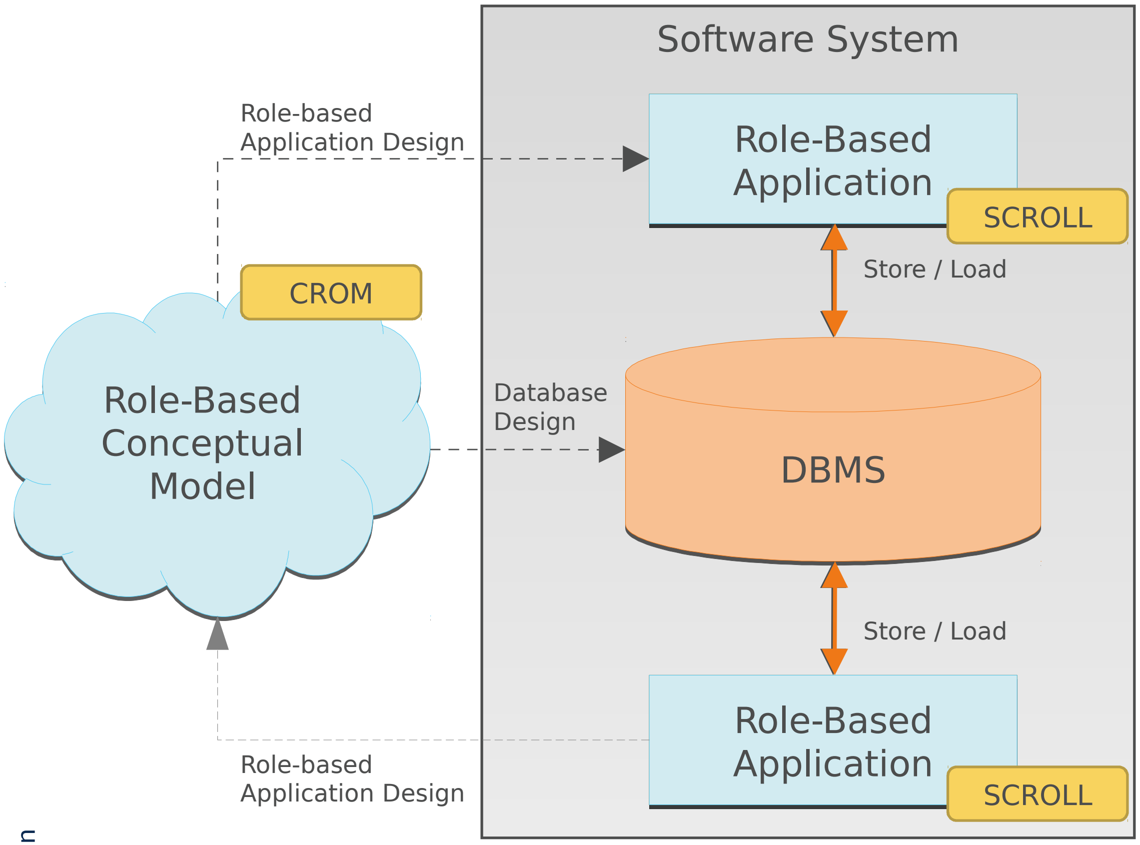
- SCROLL: Scala-based library approach for *role-oriented programming*
- **no** additional tools, compilers, or translation step needed
- Scala ensures *type safety* for static code, but roles enable *dynamic evolution*
- Open source, lightweight library[1] → easy to extend and/or change
- Fully configurable *declarative dispatch*
- Graphs and traversals represent powerful tool for *4D dispatch*

1) https://github.com/max-leuthaeuser/SCROLL

Roles in Other Technical Spaces
Overview



Roles in Other Technical Spaces
Role-based Data Management with RSQL



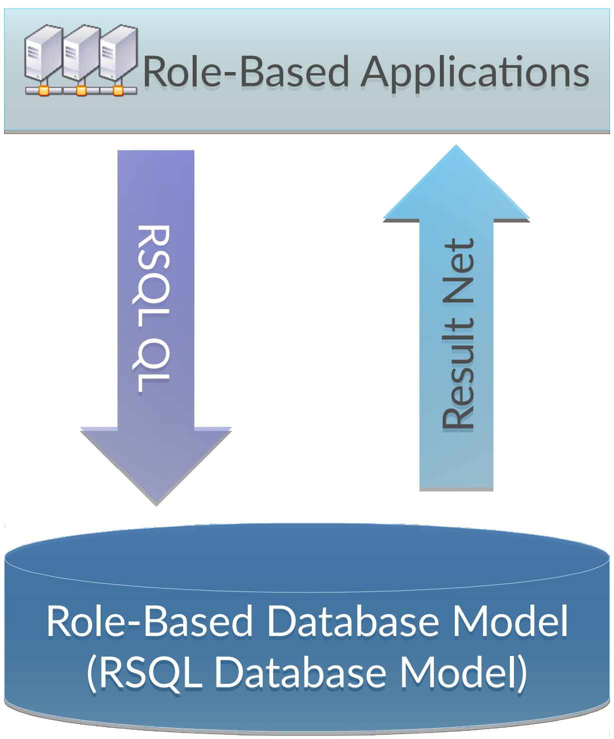
Role-Relational Impedance Mismatch

- Issues for Apps and Developers
  - Mapping overhead
  - Redundant code implementation
- Issues for the DBMS
  - No *“single point of truth”*
  - Reconstruction overhead
- Issues for the Software System
  - Huge semantic gap
  - Unstructured design

Roles in Other Technical Spaces

Role-based Data Management with RSQL

Technology Stack



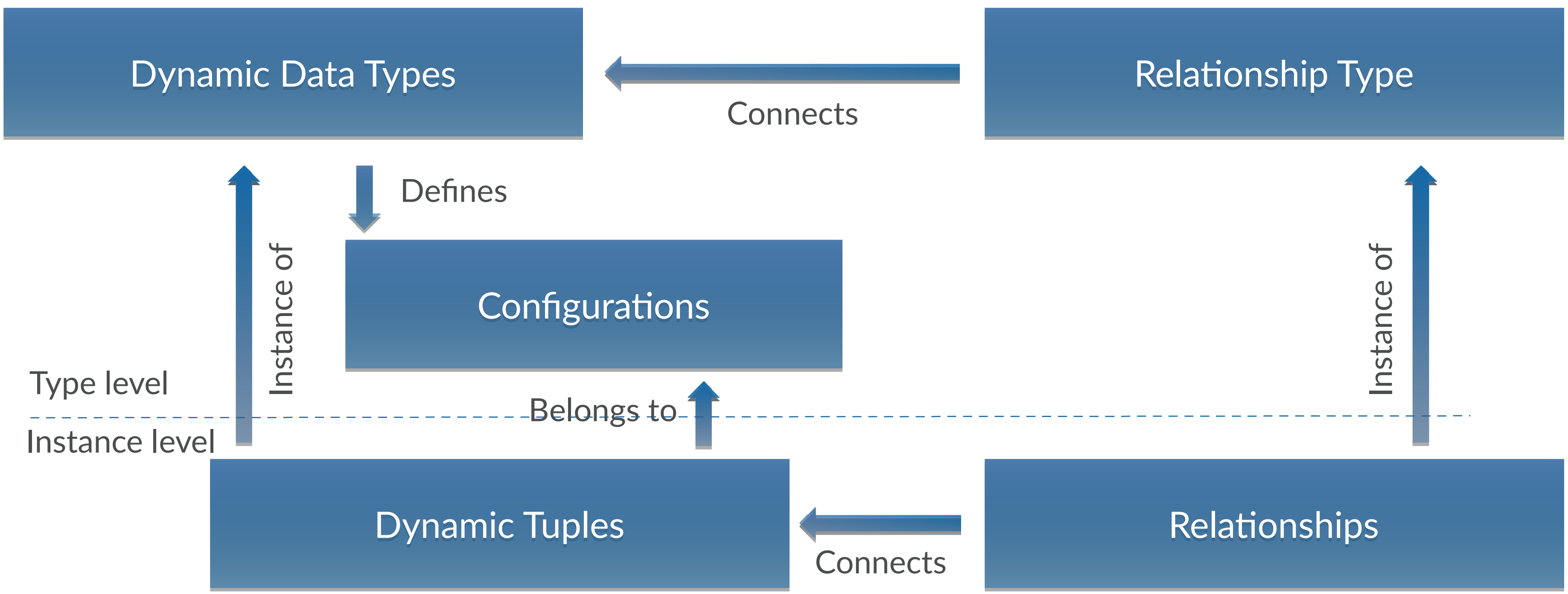
Solution

Result Representation (**RSQL Result Net**)
- Accessing role-based data structures
- Client-side support

Query Language (**RSQL Query Language**)
- Role-based communication interface
- Revised DDL – DML – DQL

Database Model (**RSQL Database Model**)
- Explicit metatype distinction
- Operators

RSQL Database Model *[Jäkel2016]*



- *Dynamic Data Types* represent complex entities *filling* and *containing* role types
- *Configuration* denotes the currently *filled* and *participating* role types
- *Relationship Types* connect two distinct role types

2) https://github.com/Eden-06/CROM

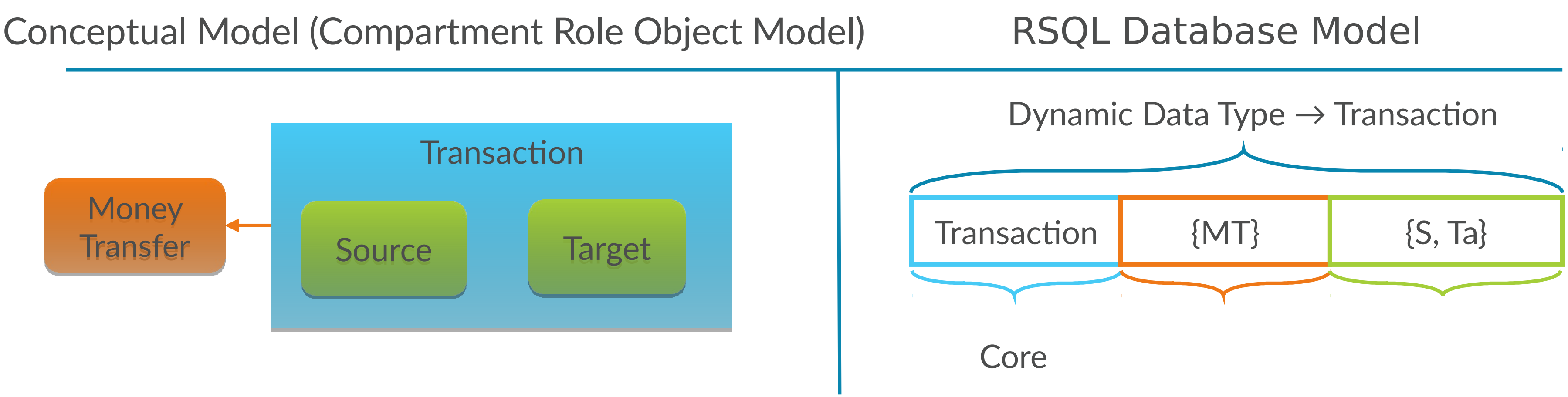
Dynamic Data Types

Logical data structure that encapsulates role-based semantics

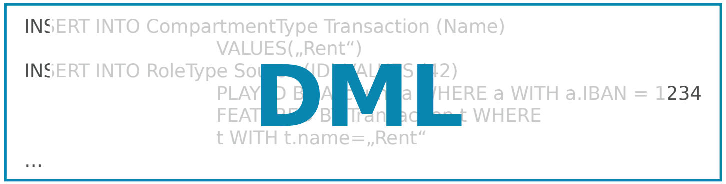
- Describes the expansion possibilities of instances
- Consists of a core type and role types in the two dimensions
  - Filling and Participating

DDL

```
CREATE CompartmentType Transaction
CREATE RoleType Source PLAYED BY (Account)
        PART OF Transaction
CREATE RoleType Target PLAYED BY (ACCOUNT)
        PART OF Transaction
CREATE RoleType MoneyTransfer PLAYED BY (Transaction)
        PART OF Bank
```



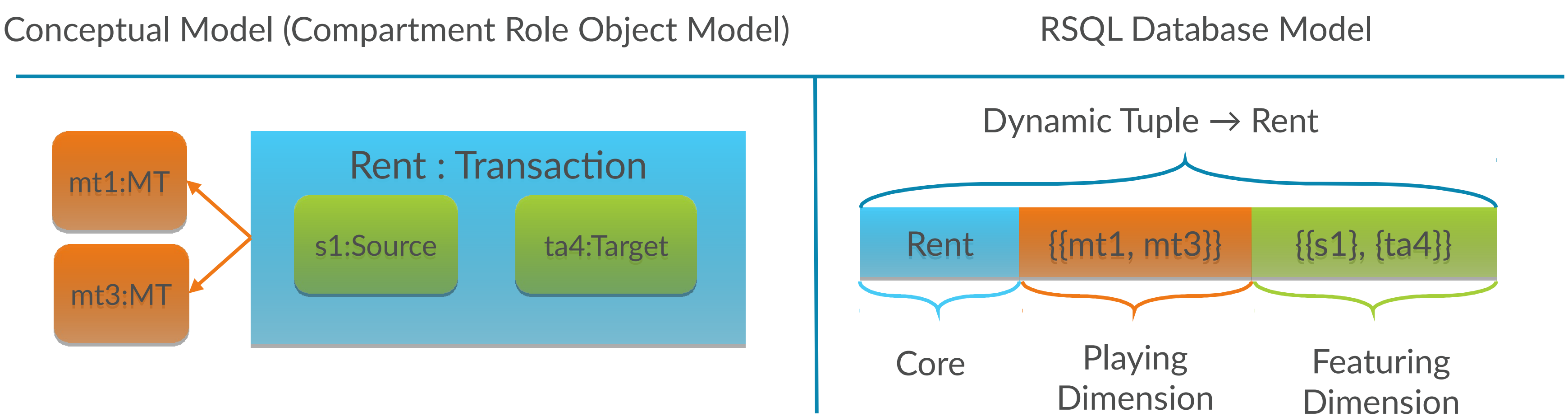
Roles in Other Technical Spaces
Role-based Data Management with RSQL



Dynamic Tuples

Logical data structure encapsulating role semantics

- Describes the current structure of an instance
- Consists of a core and roles in two dimensions
    - Playing and featuring
- Roles are grouped by their respective role type



Roles in Other Technical Spaces

Role-based Data Management with RSQL

RSQL Data Query Language *[Jäkel2016}*

- Dynamic Tuple focused querying
  - For each targeted Dynamic Data Type
  - Dynamic Tuples have to match given Configuration

SELECT * FROM Transaction t ← Core
PLAYING MoneyTransfer mt ← Filling Dimension
FEATURING Source s AND Target ta ← Participating Dimension

Transaction	{MT}	{S, Ta}

General syntax single Config-Expression

SELECT <projection> FROM Core PLAYING Role Types FEATURING Role Types

(Core PLAYING Role Types FEATURING Role Types: Configuration-Expression)

RSQL Result Net *[Jäkel2016]*

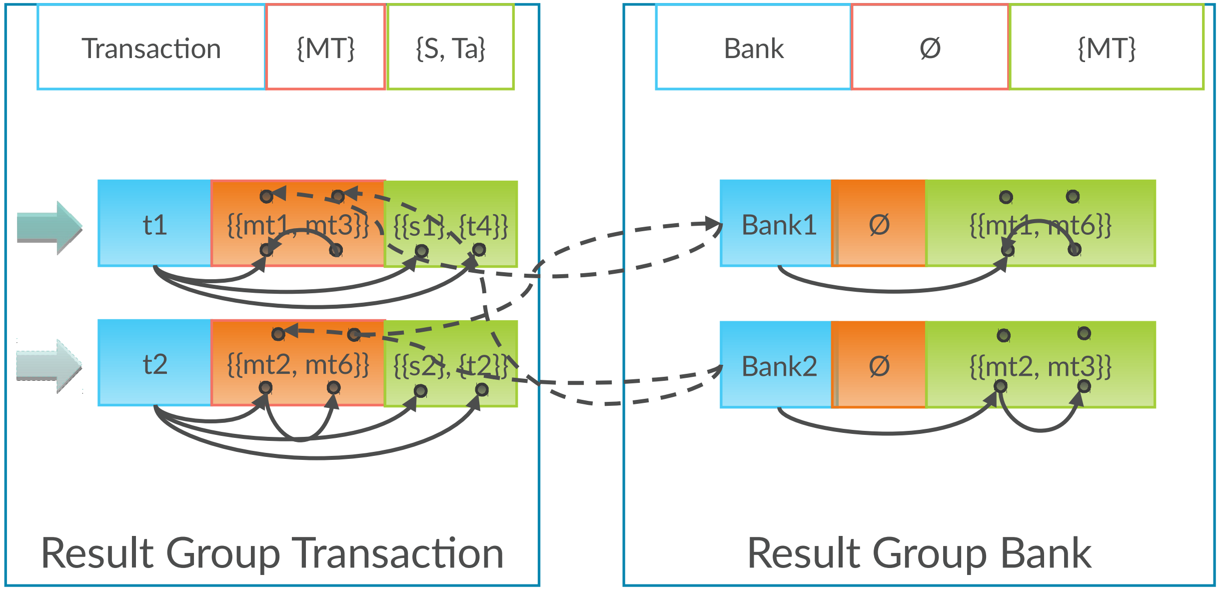
- Sets of Dynamic Tuple as query result
  - Initial pointer to a Dynamic Tuple
  - Navigation path between Dynamic Tuples

Internal navigation (solid)

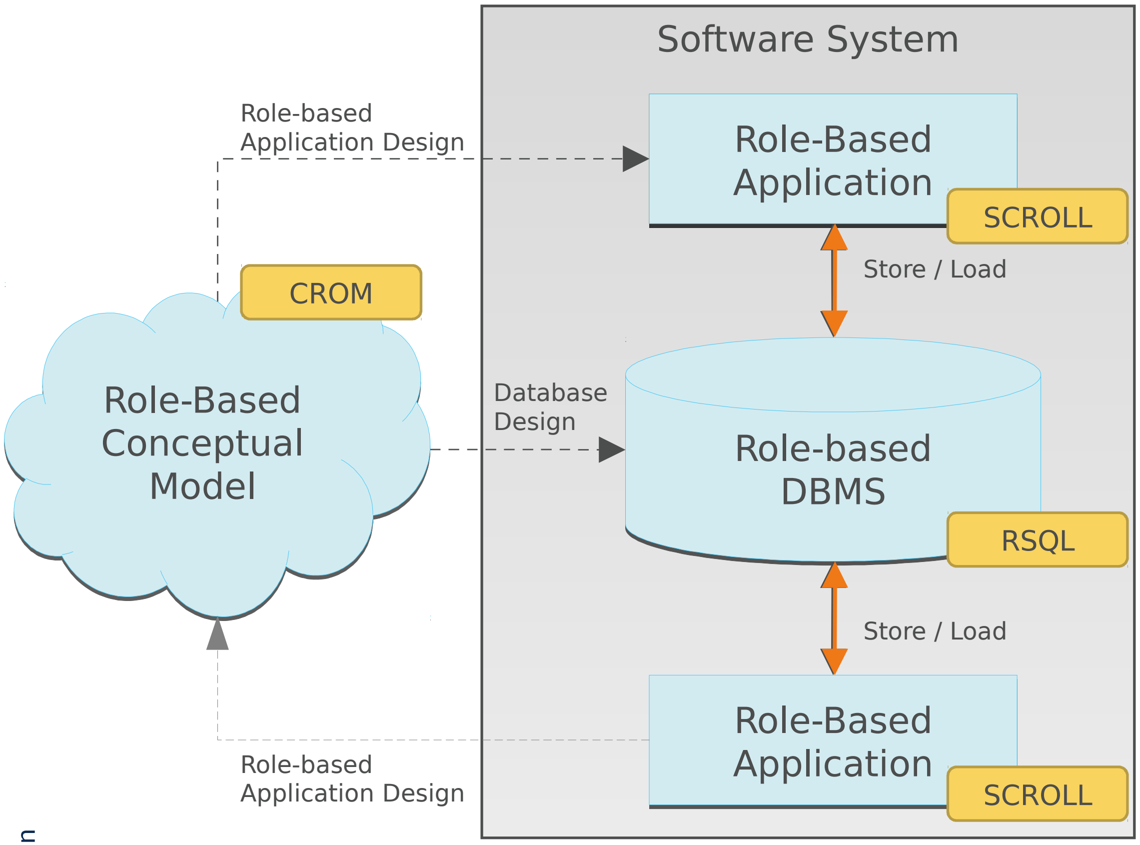
- Dynamic Tuple intern
- Accessing roles

External navigation (dashed)

- From roles to Dynamic Tuples
- Leveraging overlapping information



Roles in Other Technical Spaces
Role-based Data Management with RSQL



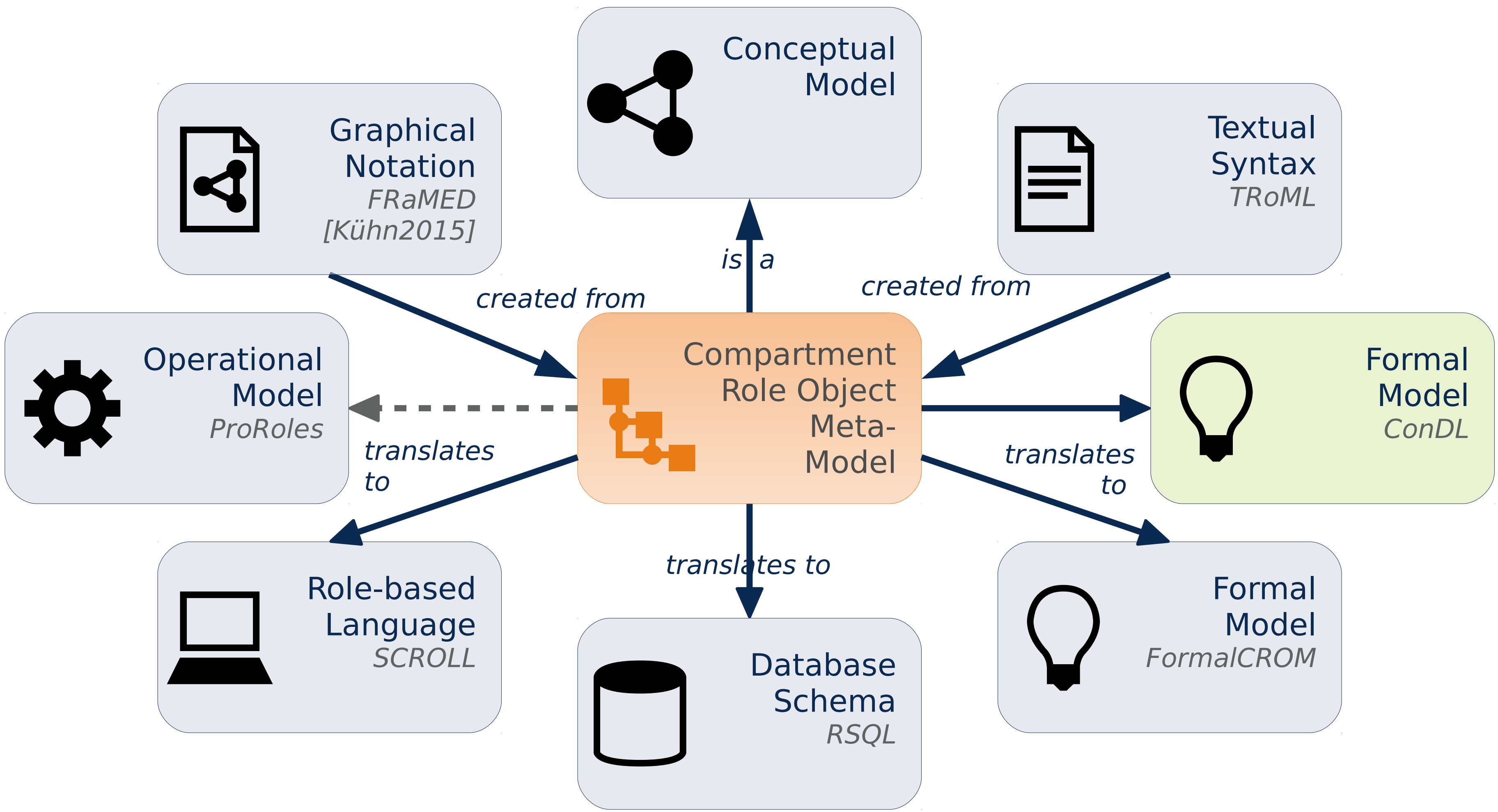
RSQL Approach

- Standard role abstraction for DBMS
  - Data Model
  - Query Language
  - Result Net
- *Dynamic Data Types* and *Dynamic Tuple* as logical structuring unit
- Independent of the underlying store

Consequences

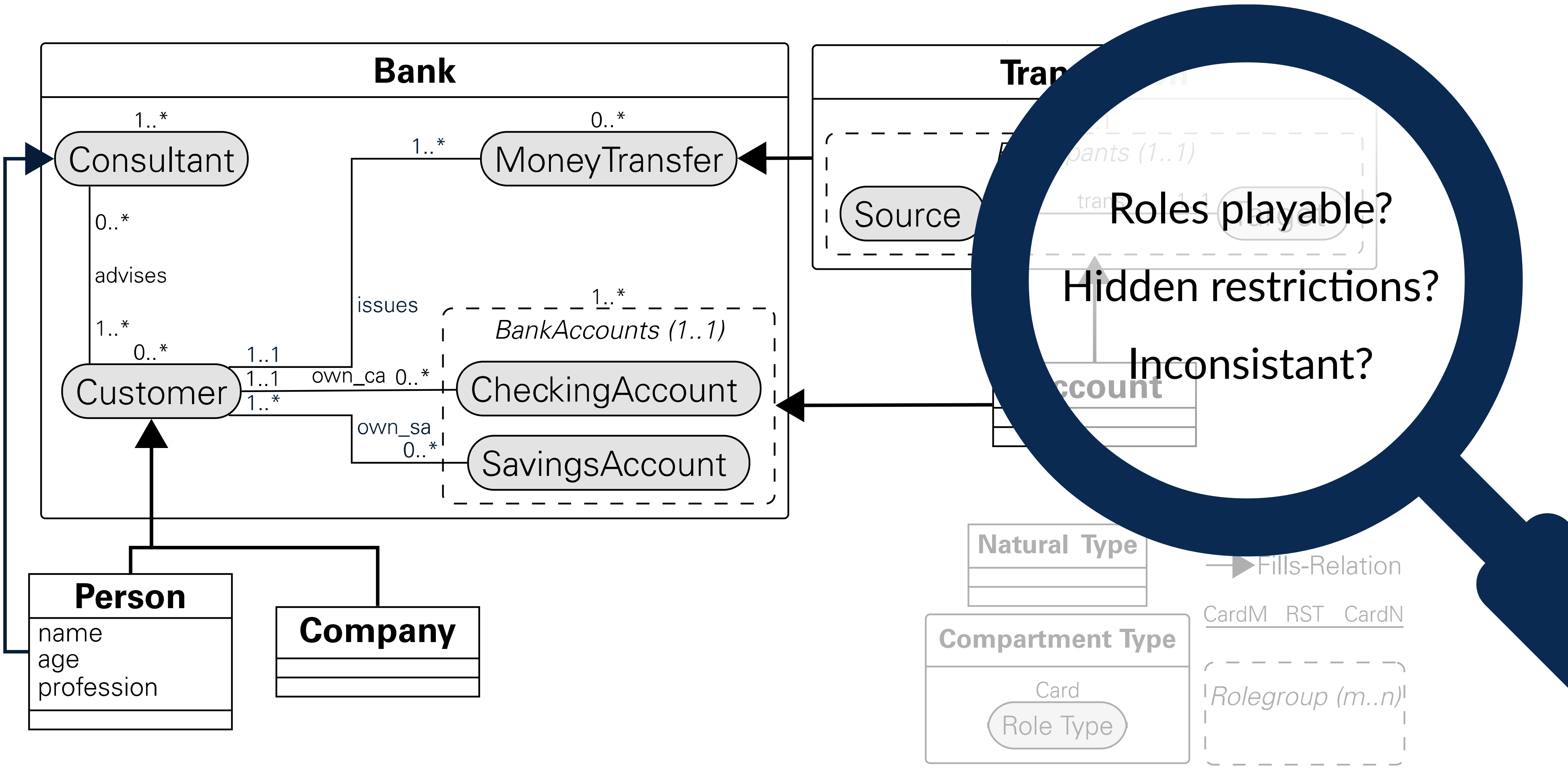
- Better *interoperability* between multiple role-based applications
- Role-based *consistency* enforceable by DBMS
- More *stable* DB schemata

Roles in Other Technical Spaces
Overview



Roles in Other Technical Spaces

Role Model Consistency Checking



- Increased complexity of CROM domain models
- Context-dependence and various constraints are hard to comprehend
- Easily leading to *inconsistent model* or *unintended restrictions*

Roles in Other Technical Spaces
Role Model Consistency Checking

Verify consistency of CROM domain models

- Utilize *Description Logic* (DL) as technical space with highly optimized reasoners
- Express *compartments*, *"players"* and *roles* as DL concepts
- Model *compartments* and ternary *role-playing* relation with binary DL roles
- Permit handling rigid, i.e., context-independent, knowledge
- Decidable reasoning on model consistency

Syntax and Semantics of the DL $\mathcal{ALC}$

Every consultant advises customers who own an checking account.
$\text{Consultant} \sqsubseteq \exists \text{advises}.(\text{Customer} \sqcap \exists \text{own\_ca}.\text{CheckingAccount})$

Peter is a consultant. $\text{Consultant}(\textit{Peter})$

N_C ... concept names — Consultant, Customer, CheckingAccount
N_R ... DL role names — advises, own_ca
N_I ... individual names — *Peter*

concept constructors: $C_1 \sqcap C_2$, $C_1 \sqcup C_2$, $\neg C_1$, $\exists r.C$, $\forall r.C$

set of $\mathcal{ALC}$ concepts: smallest set that is closed under N_C and the concept constructors of $\mathcal{ALC}$

General concept inclusion (GCI): $C \sqsubseteq D$

assertion: $C(a)$, $r(a, b)$

$\mathcal{ALC}$-axiom: a GCI or an assertion

Syntax and Semantics of the DL $\mathcal{ALC}$

Every consultant advises customers who own an checking account.
$\text{CONSULTANT} \sqsubseteq \exists \text{advises}.(\text{CUSTOMER} \sqcap \exists \text{own\_ca}.\text{CHECKINGACCOUNT})$

Peter is a consultant. $\text{CONSULTANT}(\mathit{Peter})$

A DL interpretation $\mathcal{I}$ has a domain $\Delta^{\mathcal{I}}$ and maps

- concept names A to sets $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$,
- DL role names r to binary relations $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and
- individual names a to elements $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$.

The semantics of the constructors is defined as

- $(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}$,
- $(\neg C)^{\mathcal{I}} := \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$, and
- $(\exists r.C)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \exists e.(d, e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\}$

Interpretation $\mathcal{I}$ is a model of

- the GCI $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and
- the assertion $C(a)$ ($r(a, b)$) iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ ($(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$).

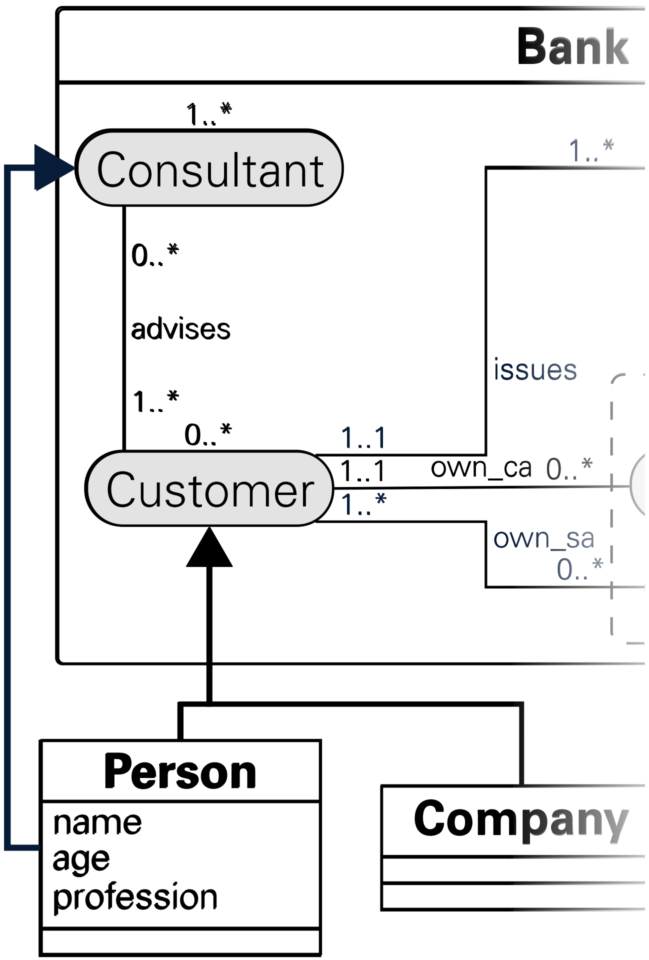
Roles in Other Technical Spaces
Role Model Consistency Checking

Contextualized Description Logic (ConDL) *[Böhme2015]*

- Two-dimensional, two-sorted description logic $\mathcal{L}_M[\![\mathcal{L}_O]\!]$
- $\mathcal{L}_M$ to describe knowledge about contexts (meta level)
- $\mathcal{L}_O$ to describe knowledge within contexts (object level)
- Contexts $\approx$ possible worlds
- Concepts/axioms of object logic are usual $\mathcal{L}_O$ concepts/axioms
- Object axioms used as meta concepts

  → $\underbrace{[\![C \sqsubseteq D]\!]}_{\text{meta concept}}$ describes set of worlds where $\underbrace{C \sqsubseteq D}_{\text{object axiom}}$ holds

Mapping CROM to ConDL

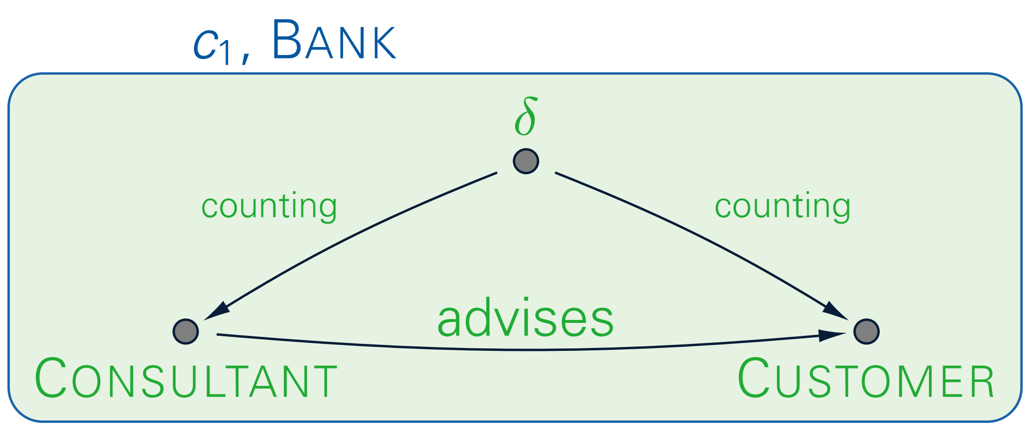


$$\top \sqsubseteq [\![\textsc{Consultant} \sqcup \textsc{Customer} \sqsubseteq =_1 \text{counting}^-.\{\delta\}]\!]$$

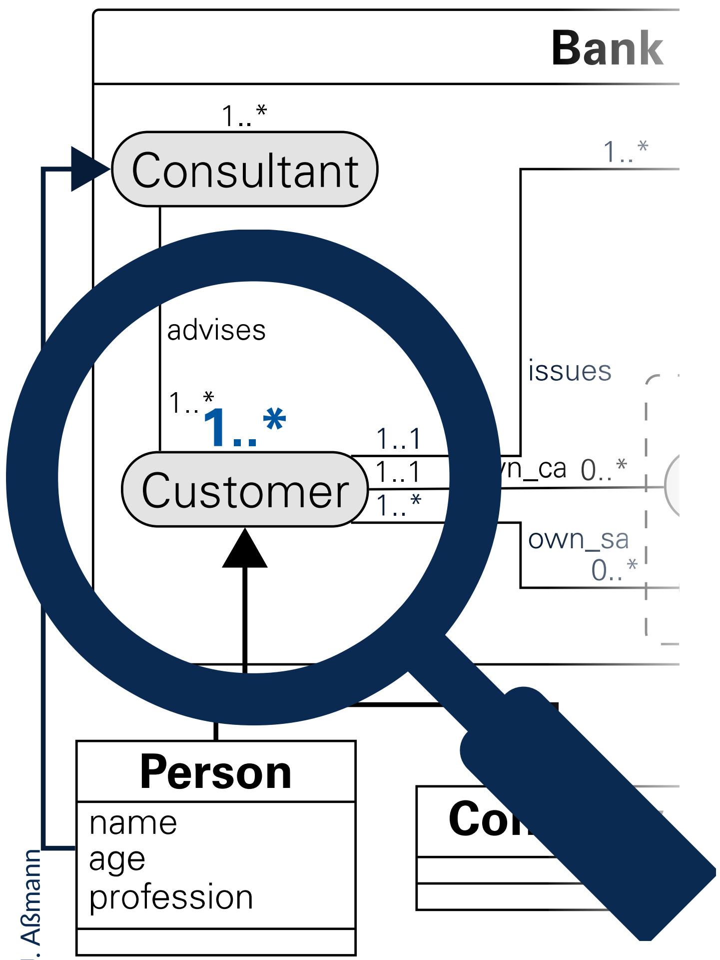
$$\textsc{Bank} \sqsubseteq [\![(\geqslant_1 \text{counting}.\textsc{Consultant})(\delta)]\!]$$

$$\textsc{Bank} \sqsubseteq [\![\top \sqsubseteq \forall.\mathit{advises}.\textsc{Customer}]\!]$$

$$\textsc{Bank} \sqsubseteq [\![\textsc{Consultant} \sqsubseteq \geqslant_1 \text{advises}.\top]\!]$$



Mapping CROM to ConDL



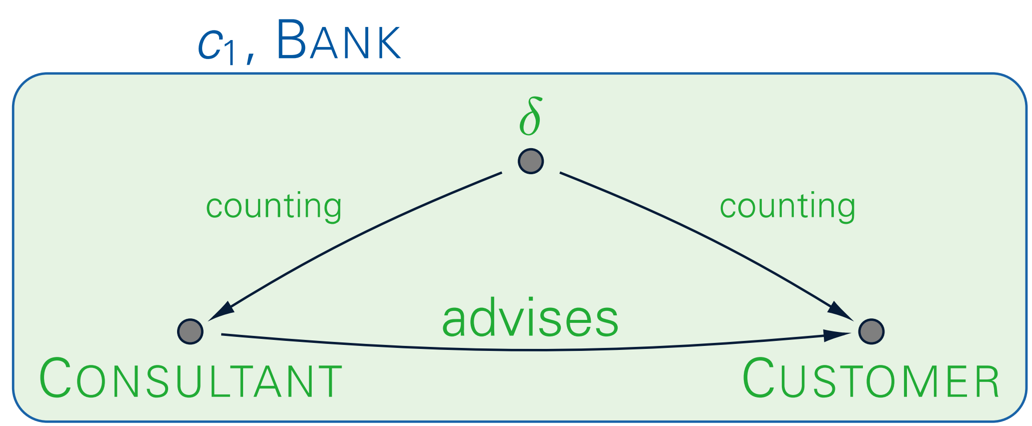
$$\top \sqsubseteq [\![\textsc{Consultant} \sqcup \textsc{Customer} \sqsubseteq =_1 \text{counting}^-.\{\delta\}]\!]$$

$$\textsc{Bank} \sqsubseteq [\![(\geqslant_1 \text{counting}.\textsc{Consultant})(\delta)]\!]$$

$$\textsc{Bank} \sqsubseteq [\![\top \sqsubseteq \forall.\mathit{advises}.\textsc{Customer}]\!]$$

$$\textsc{Bank} \sqsubseteq [\![\textsc{Consultant} \sqsubseteq \geqslant_1 \text{advises}.\top]\!]$$

$$\mathit{Bank} \sqsubseteq [\![(\geqslant_1 \text{counting}.\textsc{Customer})(\delta)]\!]$$



Roles in Other Technical Spaces
Role Model Consistency Checking

Limitations (so far)

- CROM does not support *attribute-based* constraints, while ConDL does
- *Global role constraints* of CROM not supported, yet

Verifying consistency of CROM models

- ConDL naturally captures semantics of *compartments*, *“players”* and *roles*
- Dedicated reasoner JConHT[3] supports efficient reasoning on ConDLs
  - *2EXPTIME-hard* complexity
  - Improved, if no rigid names occur
  - Reduced, if nested contexts (compartments) occur
- Decidable reasoning on $\mathcal{SHOIQ}[\![\mathcal{SHOIQ}]\!]$

3) https://github.com/ElCattivo13/JConHT

41.5. Family of Role-based Languages

Prof. Dr. Uwe Aßmann

Dr.-Ing. Thomas Kühn

Technische Universität Dresden

Institut für Software- und Multimediatechnik

http://st.inf.tu-dresden.de/teaching/most

Version 16-1.0, 18.12.17

DRESDEN concept Exzellenz aus Wissenschaft und Kultur

Literature

[Kühn2014] A Metamodel Family for Role-based Modelling and Programming Languages

T. Kühn, M. Leuthäuser, S. Götz, C. Seidl, and U. Aßmann

Software Language Engineering SLE'14, Springer (2014)

[Kühn2017] A Family of Role-Based Languages

T. Kühn

Dissertation, Technische Universität Dresden, Fakultät Informatik (2017)

References

[Steimann2000] On the Representation of Roles in Object-Oriented and Conceptual Modelling.

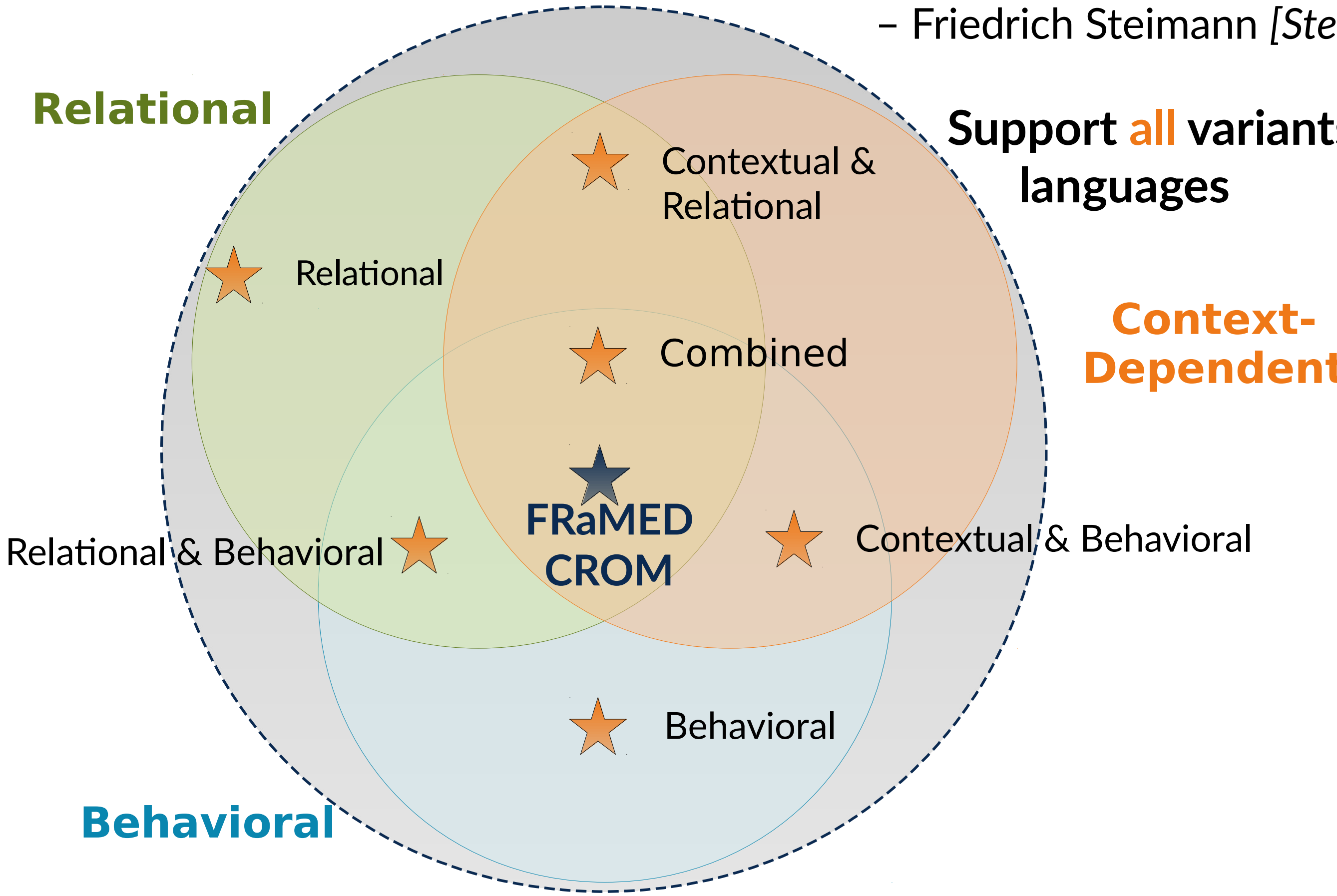
Friedrich Steimann

Data & Knowledge Engineering, Elsevier, (2000)

„[...] there is not one ideal way of defining [the role concept], but a number of competing approaches.“

– Friedrich Steimann *[Steimann2000]*

Support all variants of role-based languages

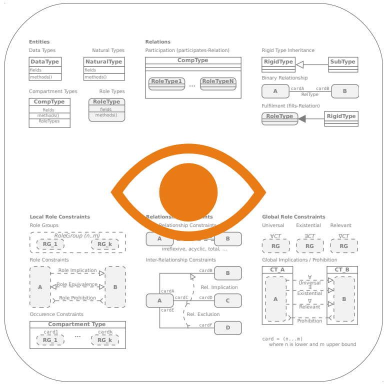


- How to *harmonize* and *reconcile* the research field?

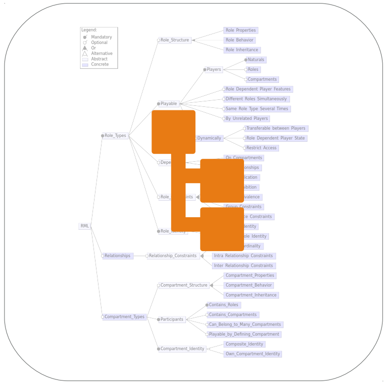
Family of Role-based Languages
Feature Modeling Approach

Design a family of role-based modeling languages

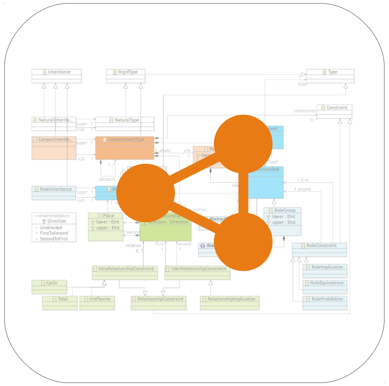
- Reuse graphical notation of CROM as *common notation*
- Design *feature model* for role-based languages
- Provide a *family of metamodels* for language variants
- Extend FRaMED to *software product line* (SPL)



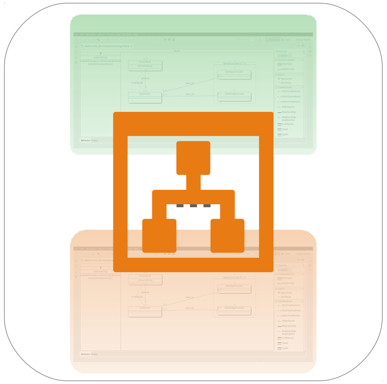
Common Notation



Feature Model



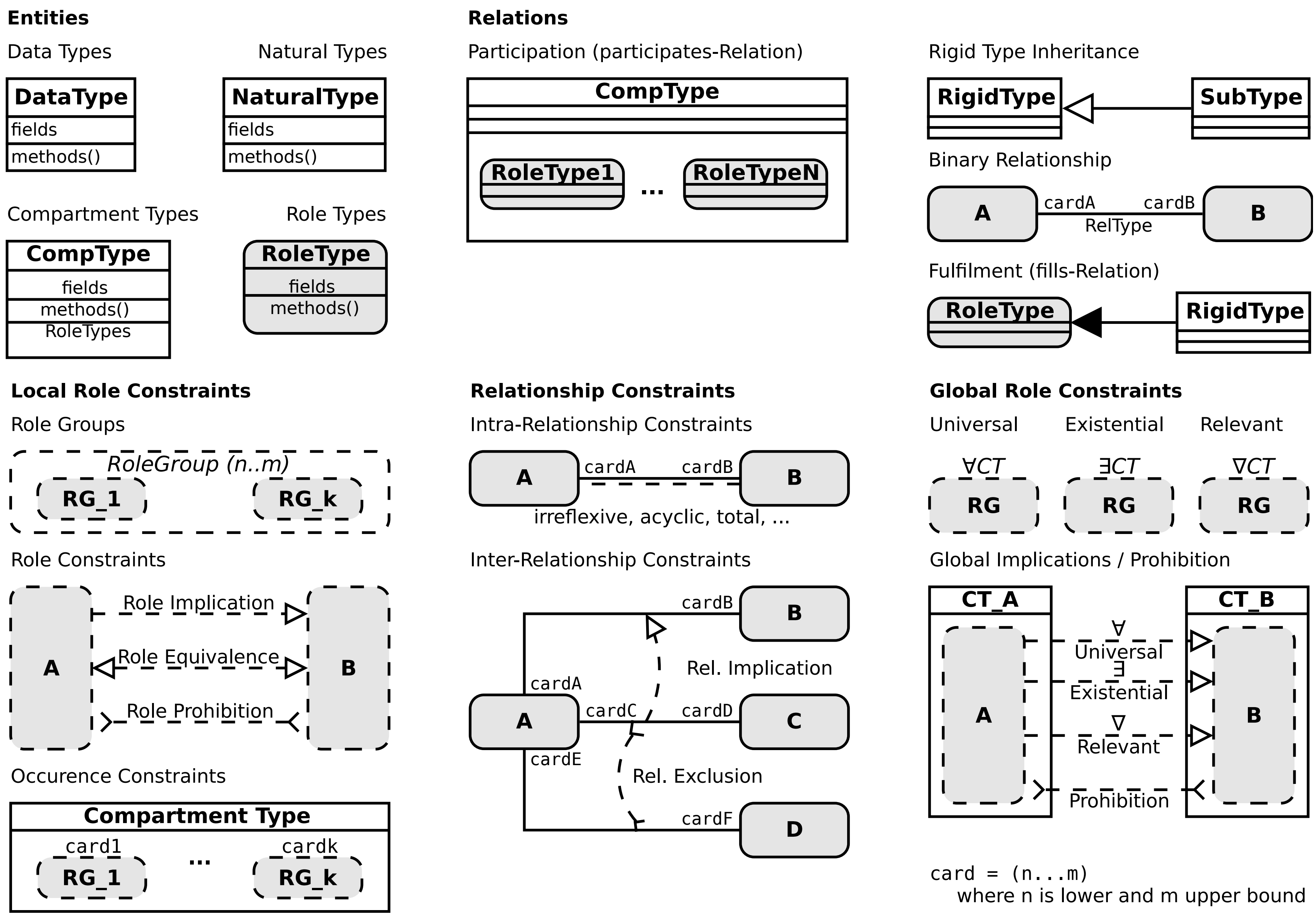
Family of Metamodels



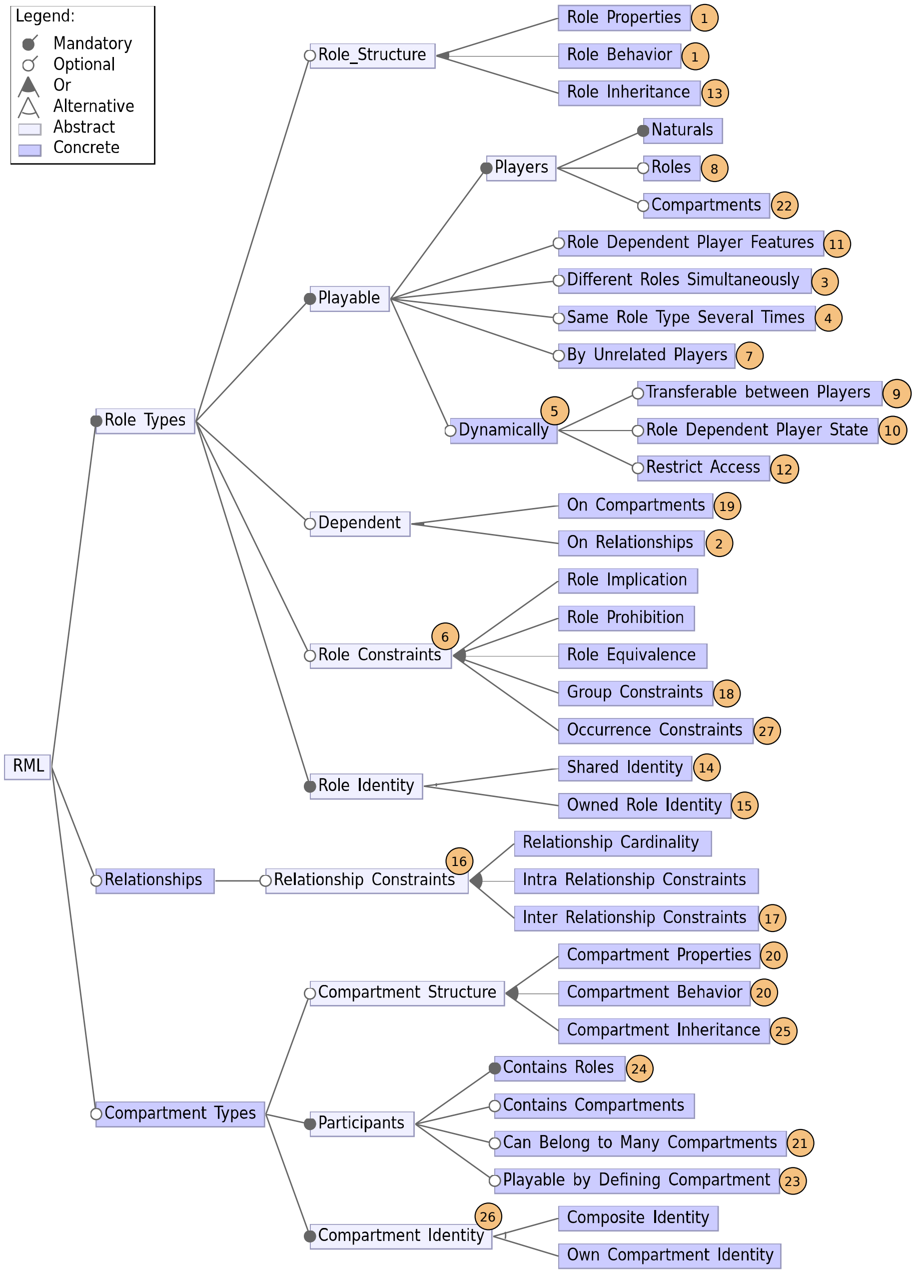
Family of Role Model Editors

Family of Role-based Languages

Common Graphical Notation



Family of Role-based Languages
Feature Modeling Approach



Feature Model *[Kühn2014]*

- Collects all 27 features of roles
- Captures implicit dependencies among features
- 6 cross tree constraints enforce consistency

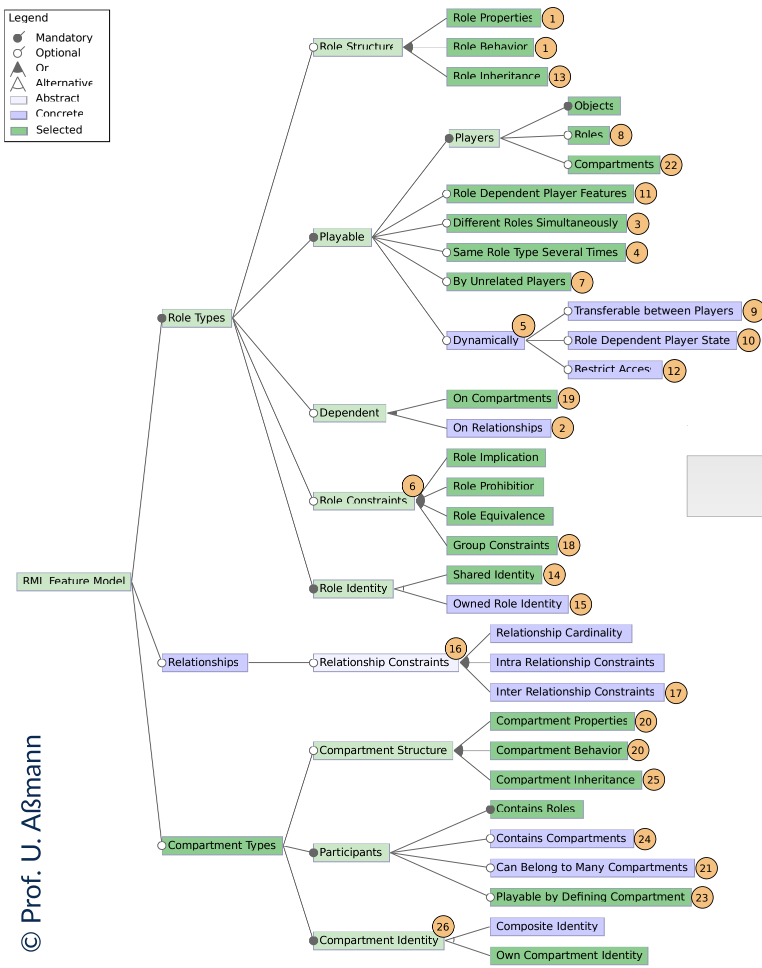
Usage

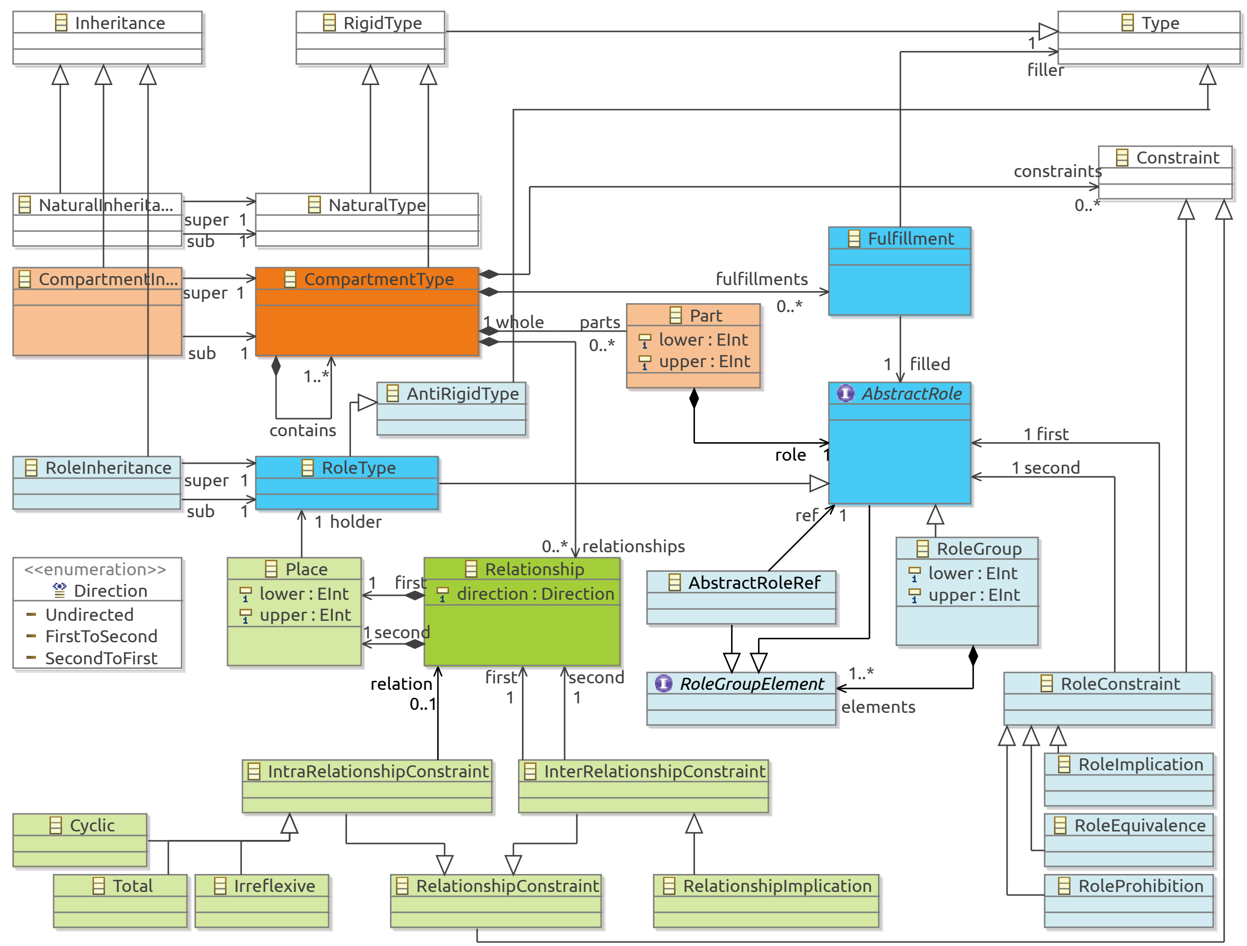
- Configuration of language variant
- Automatic generation of corresponding
  - *Metamodel* and
  - *Role model editor*

Family of Role-based Languages

Software Product Line of CROM Metamodels *[Kühn2014]*[4]

- Eclipse-based metamodel generator to create Ecore model variant
- *Delta Modeling Approach* refines a common base wrt. each selected feature

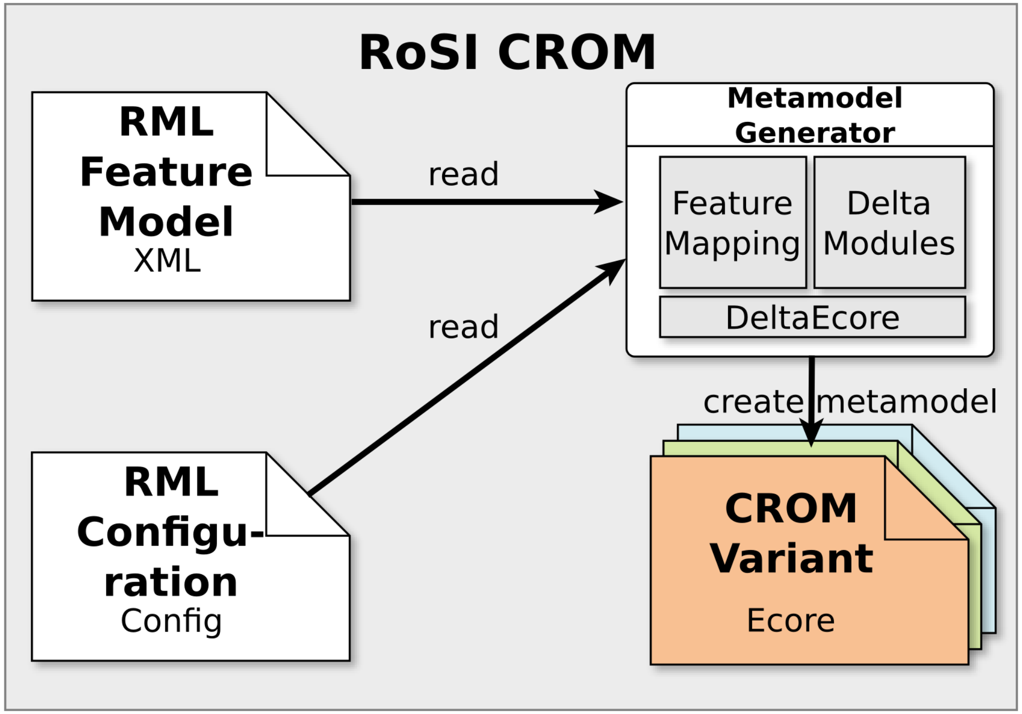




4) https://github.com/Eden-06/RoSI_CROM

Family of Role-based Languages

Software Product Line of CROM Metamodels *[Kühn2014]*[4]

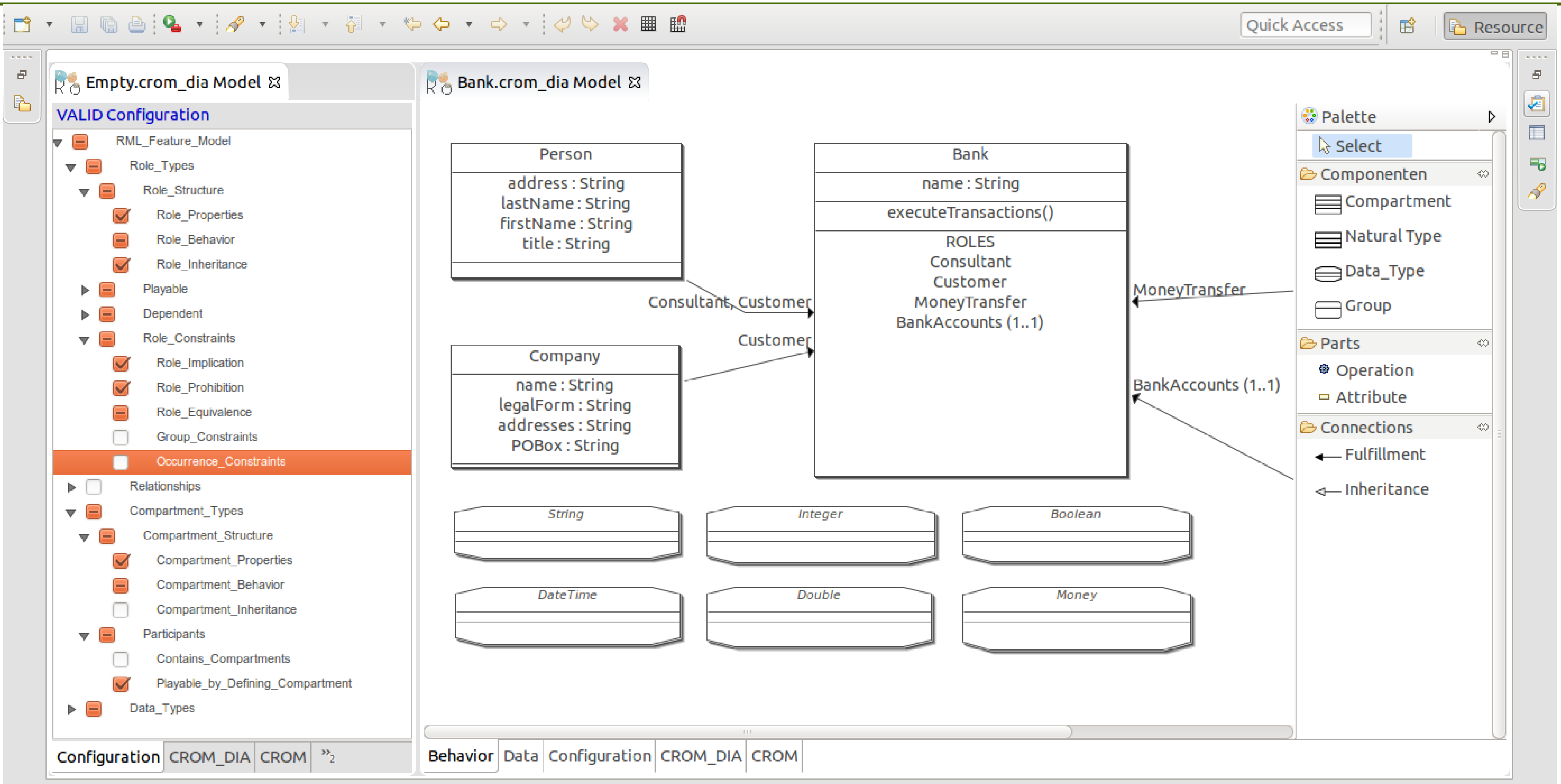


- Based on *Eclipse Modeling Framework* (EMF), *FeatureIDE [Thüm2014]*, and *DeltaEcore [Seidl2014]*
- *Feature minimal metamodel* as common base
- *Feature Mapping* maps configuration to *delta modules*
- *Delta modules* add or refine model elements

4) https://github.com/Eden-06/RoSI_CROM

Family of Role-based Languages

Software Product Line of Role Model Editors *[Kühn2017]*[5]

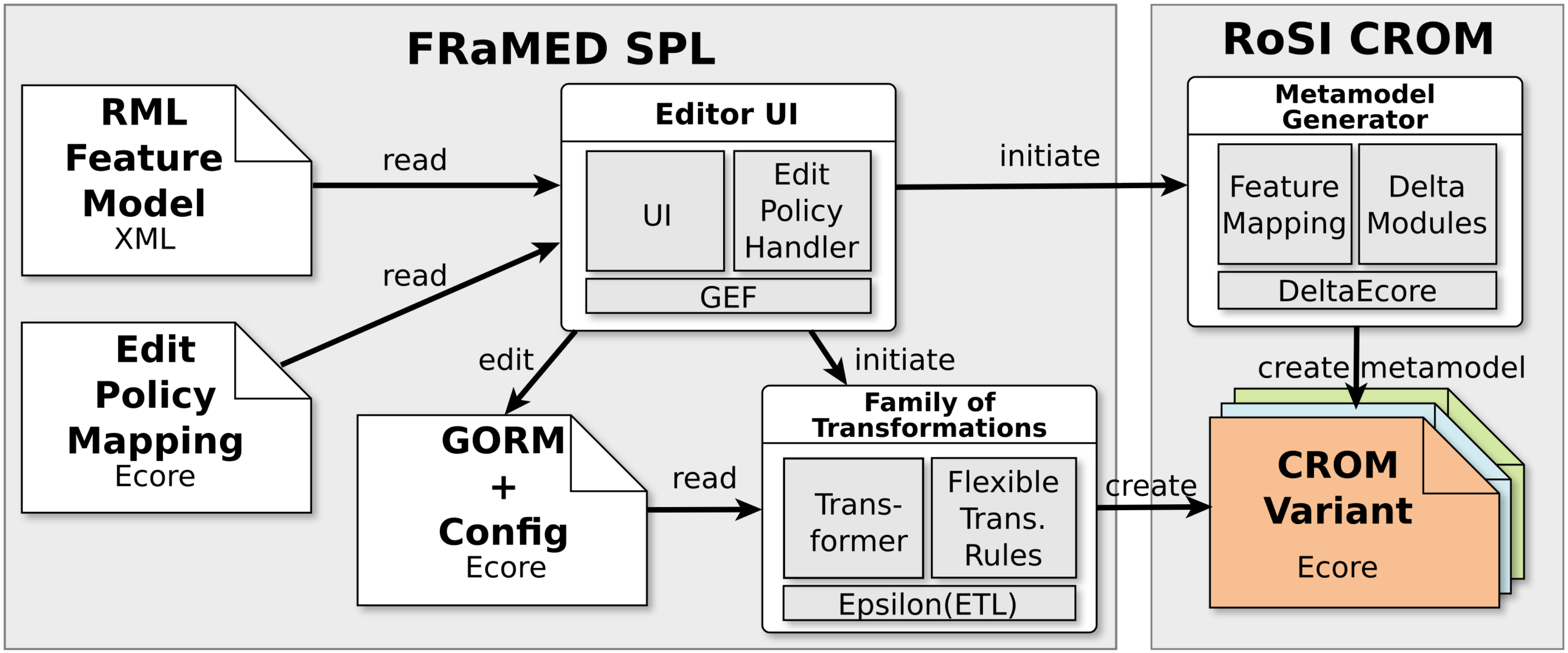


- Support easy runtime *reconfiguration* of modeling language variants
- *Feature configuration* maintained for each graphical model (GORM)
- CROM variant is updated upon saving

5) https://github.com/Eden-06/FRaMED-2.0

Family of Role-based Languages

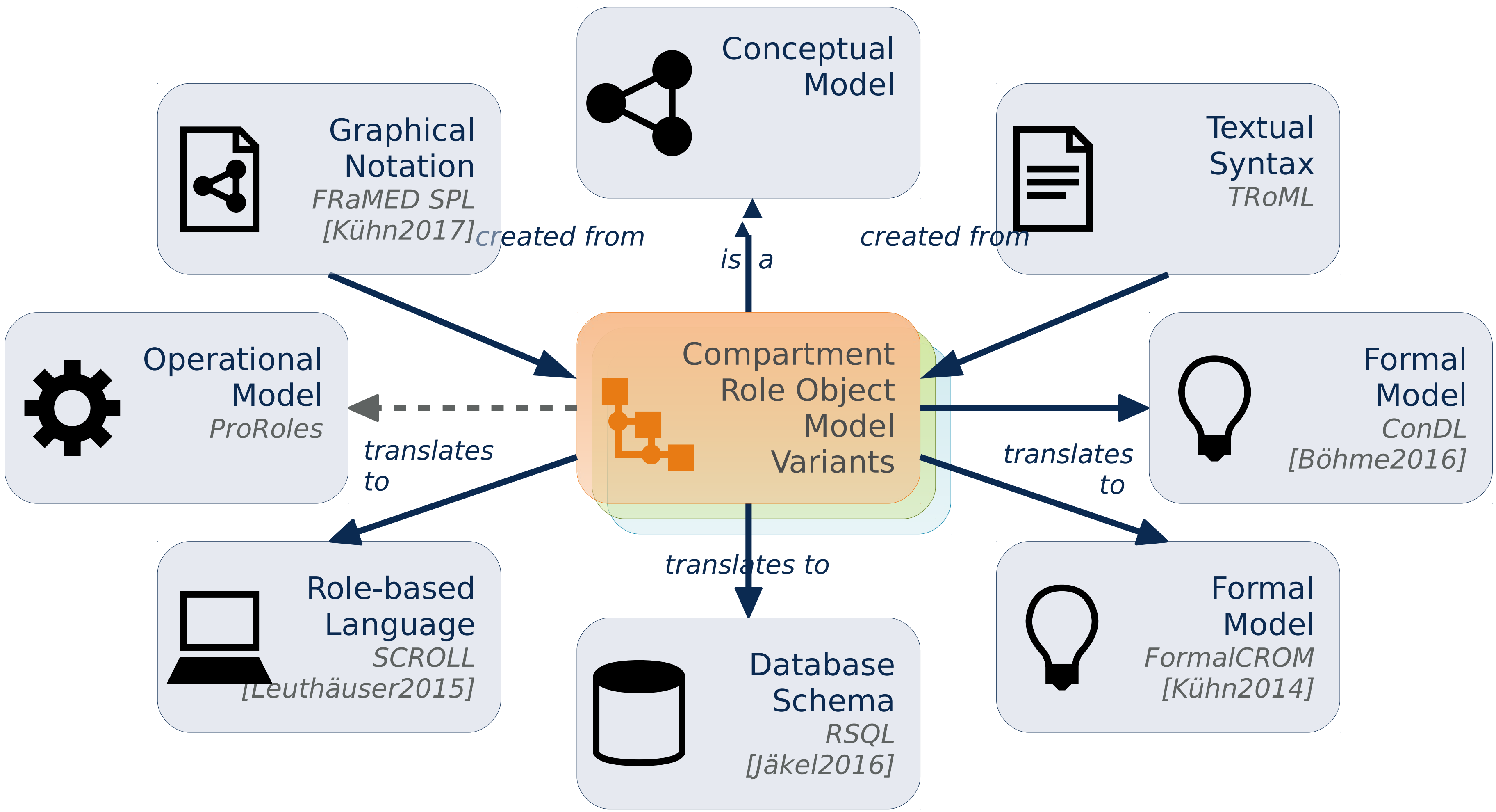
Software Product Line of Role Model Editors *[Kühn2017]*[5]



- Extension of FRaMED to fully dynamic feature-oriented product line
  - Feature-aware *Palette*
  - Family of *Edit Policies* to adapt editor behavior
  - Family of *Model Transformations* to save selected CROM variant
- Extensible due to family of *Metamodels*, *Edit Policies* and *Model Transformations*

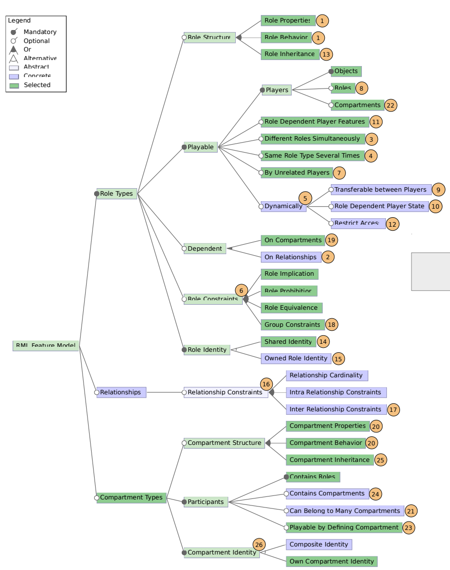
5) https://github.com/Eden-06/FRaMED-2.0

Tools Applicable within FRaMED SPL



Family of Role-based Languages
Summary

Feature Configuration — **Metamodel Generation** — **FRaMED Configuration** — **Modeling** — **Artifact Generation**



Behavioral Metamodel
- Features **roles** played by objects
- Supports role- and group-constraints

Behavioral Role-Based Modeling Language
- For **behavioral** models
- Design simple role models

Behavioral Role Model

Relational Metamodel
- **Relationships** with role ends
- Adds inter- and intra-relationship constraints

Relational Role-Based Modeling Language
- Generates **relational** models
- Declare role relational models

Relational Role Model

• • •

Contextual Metamodel
- **Compartments** containing roles and relationships
- Adds occurrence constraints

Contextual Role-Based Modeling Language
- **Context-dependent** models
- For contextual role models

Contextual Role Model

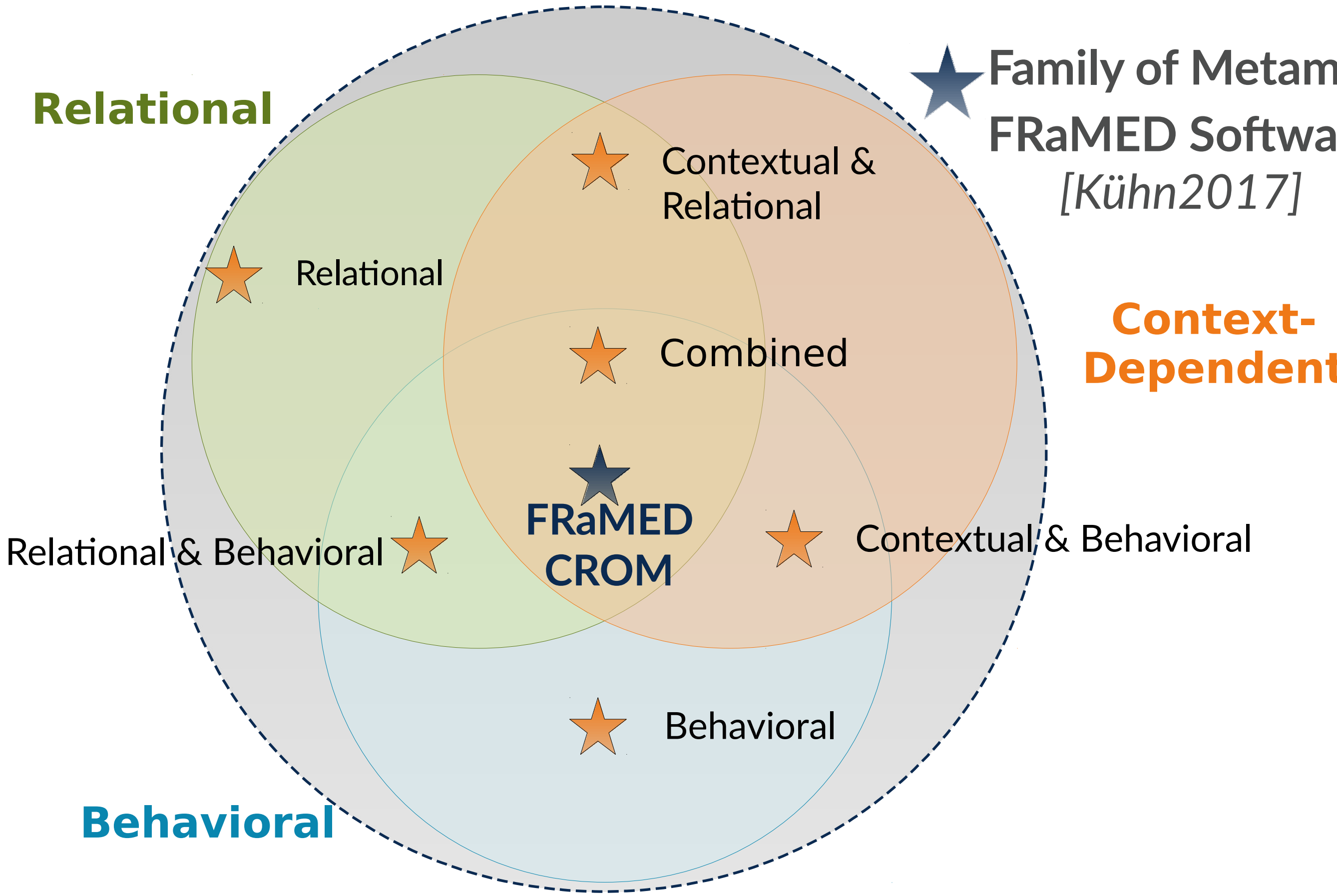
formalCROM
- Formal model for roles
- Validation of **well-formedness**

ConDL
- Contextual ontology
- Validation of **consistency**

SCROLL
- Role-oriented programming
- Generation of **program stub**

RSQL
- Role-based database
- Generation of **database schema**

Family of Role-based Languages
Conclusion



Family of Metamodels *[Kühn2014]*
FRaMED Software Product Line *[Kühn2017]*

- Metamodeling approach to *reconcile* and *harmonize* a research field
- Applicable for other domains: *Context-Oriented Programming (COP)*

The End

- Why is it hard to reconcile and harmonize a research field?
- What role does a metamodel play in a language?
- Why is the generator of metamodels beneficial for RoSI?
- How does one typically bridge the gap between technical spaces