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
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
Recap Role-Based (Meta-)Modeling
Formal Foundation of CROM

CROM EMOF (Ecore) Metamodel
Recap Role-Based (Meta-)Modeling

Tool Support Surrounding CROM

Graphical Notation: FRaMED [Kühn2015]

Conceptual Model

Textual Syntax: TRoML

Operational Model: ProRoles

Compartment Role Object Meta-Model

role-based Language: SCROLL

Formal Model: ConDL

Database Schema: RSQl

Formal Model: FormalCROM
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, 27.03.18
Literature

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

[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

Graphical Notation
FRaMED [Kühn2015]

Conceptual Model
is a
created from

Textual Syntax
TRoML

Operational Model
ProRoles

Compartiment Role Object Meta-Model

Role-based Language
SCROLL

translated to

Formal Model
ConDL

Formal Model
FormalCROM

Database Schema
RSQ
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

---

Model-Driven Software Development in Technical Spaces (MOST)

© Prof. U. Aßmann

Slides prepared by Max Leuthäuser
Roles in Other Technical Spaces
Role-oriented Programming with SCROLL

Four Dimensional Dispatch \([\text{Hirschfeld2008}]\)

- **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

by context (sender, receiver context, relations)

by sender

- \(S_A\)
- \(S_B\)

by receiver

- \(O_A\)
- \(O_B\)

by name

- \(m_1: * : *\)
- \(m_1: * : C_\beta\)
- \(m_1: S_B : C_\alpha\)
Roles in Other Technical Spaces
Role-oriented Programming with SCROLL

SCala RoLes Library (SCROLL) [Leuthäuser2014]

- Lightweight Library for role-oriented programming¹
- 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
Roles in Other Technical Spaces
Role-based Programming with SCROLL

Example Banking Application

```scala
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
    }
  }
}
```

```scala
// 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))
    }
  }
}
```
Roles in Other Technical Spaces
Role-oriented Programming with SCROLL

Declarative Dispatch Description

- Based on graph traversal operators

// implicit values are passed as additional, hidden argument to method invocations

```scala
implicit val dd =
  From ( _ is[Account] ).
To ( _ is[Source] ).
Through ( _ => checkSomeRestriction() ).
Bypassing ( _ => checkSomeOtherRestriction() )
```

Slides prepared by Max Leuthäuser
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 library¹ → 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

- Graphical Notation: FRaMED [Kühn2015]
- Textual Syntax: TRoML
- Operational Model: ProRoles
- Role-based Language: SCROLL
- Conceptual Model
- Formal Model: ConDL
- Formal Model: FormalCROM
- Database Schema: RSQL
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

Slides prepared by Tobias Jäkel
Roles in Other Technical Spaces
Role-based Data Management with RSQl

Technology Stack

Solution

Role-Based Applications

RSQl QL

Result Net

Role-Based Database Model (RSQl Database Model)

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

Slides prepared by Tobias Jäkel
Roles in Other Technical Spaces
Role-based Data Management with RSQl

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

Conceptual Model (Compartment Role Object Model)

RSQL Database Model

Dynamic Data Type → Transaction

Transaction

{MT}

{S, Ta}

Core

DDL

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

Source

Target

Money Transfer

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

RSQQL 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
PLAYING MoneyTransfer mt
FEATURING Source s AND Target ta
```

General syntax single Config-Expression

```
SELECT <projection> FROM 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.
$\text{CONSULTANT} \sqsubseteq \exists \text{advises.} (\text{CUSTOMER} \sqcap \exists \text{own_ca.CHECKINGACCOUNT})$

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

$N_C$ ... concept names
$N_R$ ... DL role names
$N_I$ ... individual names

$\text{CONSULTANT, CUSTOMER, CHECKINGACCOUNT}$
$\text{advises, own_ca}$
$\text{Peter}$

Concept constructors:

Set of $\mathcal{ALC}$ concepts:

General concept inclusion (GCI):

assertion:

$\mathcal{ALC}$-axiom:

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

$C \sqsubseteq D$

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

a GCI or an assertion
Syntax and **Semantics** of the DL $\mathcal{ALC}$

Every consultant advises customers who own a checking account.

$$\text{CONSULTANT} \sqsubseteq \exists \text{advises.}(\text{CUSTOMER} \sqcap \exists \text{own_c. CHECKING ACCOUNT})$$

Peter is a consultant.  

$$\text{CONSULTANT}(Peter)$$

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
- $(\exists r.C)^\mathcal{I} := \{ d \in \Delta^\mathcal{I} \mid \exists e.(d, e) \in r^\mathcal{I} \land e \in C^\mathcal{I} \}$

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 $a^\mathcal{I} \in C^\mathcal{I} \ ((a^\mathcal{I}, b^\mathcal{I}) \in r^\mathcal{I})$.  

Slides prepared by Stephan Böhme
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 \(\simeq\) possible worlds
- Concepts/axioms of object logic are usual \(\mathcal{L}_O\) concepts/axioms
- Object axioms used as meta concepts

\[ [C \sqsubseteq D] \] describes set of worlds where \(C \sqsubseteq D\) holds

meta concept

object axiom
Roles in Other Technical Spaces
Role Model Consistency Checking

Mapping CROM to ConDL

\[ T \subseteq \text{CONSULTANT} \sqcup \text{CUSTOMER} \sqsubseteq =_{1}\text{counting}^{-} \cdot \{\delta\} \]

\[ \text{BANK} \subseteq \text{\texttt{(\geq 1}\text{counting.}\text{CONSULTANT})}(\delta) \]

\[ \text{BANK} \subseteq \text{\texttt{(T \sqsubseteq \forall.}\text{advises.}\text{CUSTOMER})} \]

\[ \text{BANK} \subseteq \text{\texttt{(CONSULTANT \sqsubseteq \geq 1}\text{advises.}\text{T})} \]
Roles in Other Technical Spaces
Role Model Consistency Checking

Mapping CROM to ConDL

\[ T \subseteq [\text{CONSULTANT} \cup \text{CUSTOMER} \subseteq =_1 \text{counting} \cdot \{\delta\}] \]
\[ \text{BANK} \subseteq [(\geq_1 \text{counting}.\text{CONSULTANT})(\delta)] \]
\[ \text{BANK} \subseteq [T \subseteq \forall.\text{advises}.\text{CUSTOMER}] \]
\[ \text{BANK} \subseteq [\text{CONSULTANT} \subseteq \geq_1 \text{advises}.T] \]
\[ \text{Bank} \subseteq [(\geq_1 \text{counting}.\text{CUSTOMER})(\delta)] \]

\[ c_1, \text{BANK} \]

- counting
- advises
- counting

\[ \text{CONSULTANT} \quad \delta \quad \text{CUSTOMER} \]
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 \(\text{S} \text{H} \text{O} \text{I} \text{I} \text{Q} \left[ \text{S} \text{H} \text{O} \text{I} \text{I} \text{Q} \right]\)

\(^3\) 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, 27.03.18
Literature

[Kühn2014] A Metamodel Family for Role-based Modelling and Programming Languages
Software Language Engineering SLE'14, Springer (2014)

[Kühn2017] A Family of Role-Based Languages
T. Kühn
References


Friedrich Steimann

*Data & Knowledge Engineering, Elsevier, (2000)*
Family of Role-based Languages

Motivation

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

- Friedrich Steimann [Steimann2000]

How to harmonize and reconcile the research field?
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)
Family of Role-based Languages
Common Graphical Notation

<table>
<thead>
<tr>
<th>Entities</th>
<th>Relations</th>
<th>Rigid Type Inheritance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Types</td>
<td>Participation (participates-Relation)</td>
<td>RigidType → SubType</td>
</tr>
<tr>
<td>Natural Types</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fields</td>
<td></td>
<td></td>
</tr>
<tr>
<td>methods()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compartment Types</td>
<td>Role Types</td>
<td>Binary Relationship</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RigidType</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A cardA cardB B</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Failure (fills-Relation)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RoleType → RigidType</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A cardA cardB B</td>
</tr>
<tr>
<td>Local Role Constraints</td>
<td>Relationship Constraints</td>
<td>Global Role Constraints</td>
</tr>
<tr>
<td>Role Groups</td>
<td>Intra-Relationship Constraints</td>
<td>Universal</td>
</tr>
<tr>
<td>RoleGroup (n..m)</td>
<td>irreflexive, acyclic, total, ...</td>
<td></td>
</tr>
<tr>
<td>RG_1</td>
<td></td>
<td>RG</td>
</tr>
</tbody>
</table>
| ... | | ... | ...
| Role Constraints | Inter-Relationship Constraints | Global Implications / Prohibition |
| Role Implication | | CT_A | |
| | | ∀CT_1 ...
| Role Equivalence | | CT_B | |
| | | ∃CT_1 ...
| Role Prohibition | | CT_A | |
| | | ∀CT_1 ...
| Occurrence Constraints | | CT_B | |
| Compartment Type | | ∀CT_1 ...
| | | ∀CT_1 ...
| card1 cardk | | ∀CT_1 ...
| RG_1 | | ∀CT_1 ...
| ... | | ∀CT_1 ...
| RG_k | | ∀CT_1 ...

Relationship Constraints

Card = (n..m)
where n is lower and m upper bound
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]³

- 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
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
Family of Role-based Languages

Tool Support

Tools Applicable within FRaMED SPL

- Conceptual Model
  - Created from Graphical Notation FRaMED SPL [Kühn2017]
  - Created from Textual Syntax TRoML

- Compartments Role Object Model Variants
  - Translated to Role-based Language SCROLL [Leuthäuser2015]
  - Translated to Formal Model ConDL [Böhme2016]

- Role Object
  - Created from Operational Model ProRoles

- Database Schema
  - Created from Formal Model FormalCROM [Kühn2014]
  - Translated to Formal Syntax

- Formal Model
  - Created from Textual Syntax

- Textual Syntax
  - Created from Graphical Notation

- Graphical Notation
  - Created from Operational Model

- Operational Model
  - Created from Conceptual Model

- Conceptual Model
  - Created from Role-based Language

- Role-based Language
  - Created from Graphical Notation
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

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

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

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

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

- 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