Ringvorlesung Forschungslinie

Beyond Objects

From Object Modeling
Via Role Modeling
To Satellite Modeling

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

Softwaretechnologie

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

Softwaretechnologie

- How to extend software?
The Substitutability Problem

• How to substitute a component?
The Varability Problem

- How to vary many components or a subsystem?
The Wrapping Problem

- How to wrap software with code, e.g., for protection?
The Parallelism Problem (Inheritance Anomaly)

1980s: Parallel object-oriented languages (POOL, COOL)

1991: Inheritance Anomaly

1988: Composition Technology (Aksit)
1994: Generic Synchronization Policies (McHale)
1988-94: Composition Filters (Aksit)

1996: Aspect-oriented Programming (AOP)
1996: D+RIDL (Lopes), 1999: Aspect/J (Kiczales)
2003: Transaction MDA (Löcher)
In a parallel program or library, where should synchronization code be inserted?

- Stack?
- Queue?
- OrderedCollection?
- Collection?
- Object?

**Diagram:**

- `OrderedCollection`:
  - `add()`

- `Stack`:
  - `pop()`

- `LockedStack`:
  - `s: semaphor;`
  - `pop()`
  - `wait(s);`
  - `super.pop();`
  - `free(s);`

- `Queue`:
  - `enter()`

- `PlainStack`

- `PriorityQueue`
At the beginning of the 90s, parallel object-oriented languages failed, due to the inheritance anomaly

- **Inheritance anomaly**: In inheritance hierarchies, synchronization code is intermingled with the algorithm and cannot be easily exchanged

- **Synchronization tangling**: Because synchronization code *braces* code, it is *tangling*

- **Synchronization crosscut**: Because synchronization code *is reused* code, it is *crosscutting*
Algorithm and Synchronization are Core and Aspect

Softwaretechnologie

- Composition fixes crosscut between core and aspect

OrderedCollection
  \textit{add()}

Stack
  \textit{pop()}

Queue
  \textit{enter()}

PriorityQueue

Crosscut

LockProtocol

Semaphore
  \textit{wait()}
  \textit{free()}

Monitor
  \textit{enter()}

HoareMonitor

HansenMonitor

BinarySemaphore

CountingSemaphore
What are Crosscut Relations?

- Relations between “things” in a core and an aspect
What is a concern *(Sicht, Belang)*?

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

Softwaretechnologie

1967-1995
Object-oriented development
(OOA, OOD, OOP, POOL)

1995-today
Role-oriented development
(ROD, Objects with roles)

AOSD
Step 1
Beyond Objects - Role Modeling and the Steimann Factorization

Splitting a type into a tuple of natural and founded parts
Rigid and Founded Types

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

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

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

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

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

Natural types are non-founded and rigid.

A natural type is independent of a relationship. The objects cannot leave it.
Roles in the Literature

Softwaretechnologie

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

**Softwaretechnologie**

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

```
FullPerson

- Buyer
- Grand Father
- Reader

Person

- Grand Child

FullPerson

- FullSausage
- Bought
- Sausage

FullPerson

- FullNewsPaper
- Read
- Newspaper

FullPerson

- Person
```
What is a reading buying grandfather person?

- Natural (entity)
  - Role 1
    - Ancestor
      - Father
      - Mother
      - Grand Father
      - Grand Mother
    - Acquainted
      - Reader
      - Writer
    - Negotiator
      - Buyer
      - Seller
  - Role 2
    - Person
      - Mammal
      - Dinosaurs
    - Chicken
  - Role 3
    - LivingBeing
    - Thing
      - Animal
        - Mammal
        - Dinosaurs
      - Animal
        - Person
        - Chicken
Divide a *type* into a *tuple type* over a product lattice of a core dimension and n-1 role dimensions (Core, Role_1, ..., Role_n)
Simplified Representation of Object Nets

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

Diagram:
- **Person**
  - Father
  - Child
- **Buyer**
- **Sausage**
- **Newspaper**
- **Read**
  - Reader
- **Grand Father**
- **Grand Child**
  - Bought
Advantages of the Steimann Factorization for System Construction

- Simplicity
- Representation of concerns and aspects
- Extensibility
- Substitutability (of roles and role models)
- Variability (delayed role embedding decisions)
Advantage 1: Extensibility
Simplified Extension with Connectors (Role Model, Collaboration)

- Object-role nets can be *extended by* new role models (connectors, collaborations)
A Connector is a Relational Module (Collaboration)

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

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

```
P
  GrandFather

GrandFatherShip
```

```
  GrandChild
```
Newspaper-Reading
GrandpaShip

Softwaretechnologie

Diagram showing relationships between Grandfather, Grandchild, Reader, and Reading GrandFathership.
Newspaper-Reading Buying GrandpaShip

**Softwaretechnologie**

- **Bought**
- **Buyer**
- **Grand Father**
- **Grand Child**
- **Reader**
- **Read**

**P**

- **ReadingBuying GrandFathership**
- **Reading**
- **Buy**
- **Grand Father**
- **Grand Child**

**S**

**N**
Connector superimposition extends designs

- Prof. Dr. U. Aßmann
- TU Dresden, Institut für Software- und Multimediatechnik
- Softwaretechnologie
Connectors can be Refined

Softwaretechnologie

Person

Buyer

net

Bought

Sausage

Shop

seller

Buyer

outgoing

Person

incoming
A new role relationship extends the product lattice by another dimension.
Connector Superimposition extends the Steimann Lattices of all involved Classes
Superimposition of connectors to objects in Steimann-factored form retains all inheritance structures

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

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

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

Diagram showing the relationship between Person, Father, FatherMixin, Reader, ReaderMixin, and ReadingFather.
Connectors (Collaborations, role models) are type functors
- Functions on types
- Model concerns and aspects

Direct implementation
- Mixin layers [Batory]
- Role Object Pattern
- Semantic macros
- Generic templates (BETA, Invasive Composition Aßmann)
- Aspects
- Ceasar/J [M. Mezini, Darmstadt]
- www.objectteams.org [S. Herrmann, Berlin]

Rewriting to standard languages
- Mapping (MDD process), e.g., with graph rewriting systems
Advantage 2) Scalable Refinement and Variability: Anchoring Roles into the Model-Driven Software Development (MDSD)
Variability by Role Embedding

- Mapping Roles to Implementation-Records is an *Embedding Decision*
- The plays-a relationship leaves the embedding open, defers the decision to implementation
Families and Banks (in a role model)
Families and Banks
(Delegation, Split Design)
Families and Banks
(Roles Embedded in Players)
Families and Banks
(Roles embedded in Relations)
The question “Where is a role embedded?” is a platform decision
- A role model is more platform independent than a class model
- In MDA, role models are found on a more platform independent level than class models

Role embedding is a task in MDA
The role embedding
- determines, which physical objects inherit from which roles
- computes the physical objects from maximal splits of the complex objects
Role Embedding Yields Scalability

Softwaretechnologie

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

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

RoSI

Concept of Roles as General Modeling Paradigm

- Meta Modeling
- Object Modeling
- Language Modeling
- Data Modeling
- Application Modeling
- Schema Design
- System Modeling

TB1
Roles in Conceptual and Language Modeling

TB2
Roles in Software Engineering

TB3
Roles at Runtime

Foundation
Data Modeling, Logics, Programming Systems, Software Engineering

RoSI - Synchronisation Meeting
Background
The RoSI Project

Areas of Research

- Database
- Business Informatics
- Automata Theory
- Role-based Design
- Algebraic and Logical Foundations
- Software Engineering of Ubiquitous Systems
- Computer Networks
- Software Technology

RoSI - Synchronisation Meeting
CROM - A Metamodel for RoSI

Models based on a common Metamodel

Graphical Notation → created from → Conceptual Model

Textual Syntax → created from → Database Schema

Compartment Role Object Metamodel

represents

abstracts to

Formal Model
Ladder of Paradigms

1967-1995
- Object-oriented development (OOA, OOD, OOP)

1995-today
- Role-oriented development (ROD, Objects with roles)

Key concepts:
- Scalability
- Extensibility
- Variability
- Views, aspects
- Fine-grained modeling
- Product-line engineering

Prof. Dr. U. Aßmann
TU Dresden, Institut für Software- und Multimediatechnik
Step 2
Beyond Role Modeling - Satellite-Based Modeling

(preliminary; unpublished; ongoing work)
Ladder of Paradigms (2)

- **1967-1995**: Object-oriented development (OOA, OOD, OOP)
- **1995-today**: Role-oriented development (ROD, Objects with roles)
- **today-**: Satellite-oriented development (Objects with Orbit)

Rational Grammar
Satellites (Mixins) as Predicates about Objects

as contrast to natural types (standalone things)
Satellite Objects and Types

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

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

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

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

A satellite (mixin object) is a subobject representing a predicate about a core object.
What are Facets?

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

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

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

- non-founded; rigid
Satellite Holders (Complex Objects)

- complex object (full object)
- Core
- Facet
- Role Holder
- Facet Holder
- Role
What are Phases?

- non-founded, non-rigid [Guizzardi]

States in a lifecycle
Satellite Holder

Facet, Phase, Role holder

complex object (full object)

phase

Facet

core

role holder

role

role

Facet, Phase, Role holder
What are Integral Parts?

- non-founded; rigid; non-shared
Satellite-Core Relationships (Elementary Copula)

Softwaretechnologie

- A *satellite relationship* is a “primary key” relationship between a core and a satellite,
- .. an elementary copula
  - thing plays-a role
  - thing has-a phase
  - thing has-a-classification-dimension
  - thing owns-an-integral part
  - thing is unary-predicator
Core Objects with Different Satellites
(Join by Elementary Triples)

- Person
  - teen
  - Non-Smoker
- Child
  - air-filled
- Ball
  - Playable
- Father
- Player
  - grown-up
- Sausage
  - Eatable
  - Male
  - Eating
  - Smoker
  - Stomach
  - Ship
  - Heart
Satellite-Based Development

Softwaretechnologie

Distinguish beyond roles facets, phases, integral parts

Refine the Steimann factorization with more satellite types
• The Steimann factorization works for other satellites
The Satellite Factorization

Consists of an extended Steimann Factorization

Divide a \textit{type} into a \textit{tuple type} over a product lattice of
a core dimension and
\( f \) facet dimensions
\( p \) phase dimensions
\( r \) role dimensions
\( i \) integral part dimensions
(\( \text{Core, Facet}_1, \ldots, \text{Facet}_f, \text{Phase}_1, \ldots, \text{Phase}_p, \text{Role}_1, \ldots, \text{Role}_r, \text{IntegralPart}_1, \ldots, \text{IntegralPart}_i \))
More Scalability with Satellites

- Goal: Deferring the satellite embedding decision to implementation for all satellite relationships
- All advantages of role modeling, but even more
  - Variability
  - Extensible logical designs
  - Scalable embedding decisions
The Satellite-Mapping Process and Model-Driven Architecture

The question "Where is satellite embedded?" is a platform decision.

Satellite embedding is a task in MDA.

- Business model
- Satellite Models
- Satellite Model merging
- Merged Satellite Models
- Satellite Model mapping
- Satellite embedding
- Class Model (roles embedded)
- Code
The Object-Satellite Model (The ORBIT Object Model)

(preliminary; unpublished; ongoing work)
• Roles are classification dimensions, founded
• Facets are classification dimensions, non-founded
Satellites on Orbits of Cores
Predicator Orbits
Ladder of Paradigms (2)

1967-1995
Object-oriented development (OOA, OOD, OOP)

1995-today
Role-oriented development (ROD, Objects with roles)

today-
Satellite-oriented development (Objects with orbits, ORBIT model)

Dissertation Wende 2012
Roles in Language Composition

Dissertation Richly 2011
Roles in Workflows

Dissertation Johannes 2010
Roles in Metacomposition

Chair Software-technologie
Future

Softwaretechnologie

• Why will role modeling prevail?
  – Better typing, better polymorphism
  – Flexible product lines
    • for resource-aware systems
    • for run-time-flexible systems

• Why will satellite modeling prevail?
  – Because roles are not the only types of unary predicators
  – Because “plays-a” is not the only elementary copula
Relations to the Courses of Chair Softwaretechnologie
Softwaretechnologie

Component-Based Software Engineering (CBSE)
- Role models as standard services of components
- Role models as grey-box composition system
- Role models for generic composition

Design Patterns and Frameworks (DPF)
- Role models for design patterns
- Composed design patterns
- Role object pattern
- Roles vs mixins

Diplom- und Belegarbeiten

Software-Management (SWM)
- Roles in the software process
- Roles in process models

Model-Driven Software Development in Technical Spaces (MOST)
- Roles in model-driven software development

Softwaretechnologie II (MDA)
Important References

Softwaretechnologie

  – and many more, see his home page at U Hagen
  www.objectteams.org: a programming lanugage with roles
Important References

Softwaretechnologie

### Works at SMT so far

**Softwaretechnologie**

The End