

COPE – COuPled Evolution of metamodels and models

Diplomarbeit in Zusammenarbeit mit der BMW Car IT
(Betreuer: Elmar Jürgens, Sebastian Benz)

Markus Herrmannsdörfer

7. November 2007

Perlen der Informatik

Einführung – Domänenspezifische Sprache (DSL)

Definition

spezialisierte Sprache für eine bestimmte Problemdomäne

Vorteile

- + höhere Konzentration auf das Wesentliche
- + Steigerung der Produktivität
- + bessere Analysierbarkeit

Nachteile

- Kosten für Entwicklung und Wartung einer DSL
- anfällig gegenüber Änderungen der Domäne

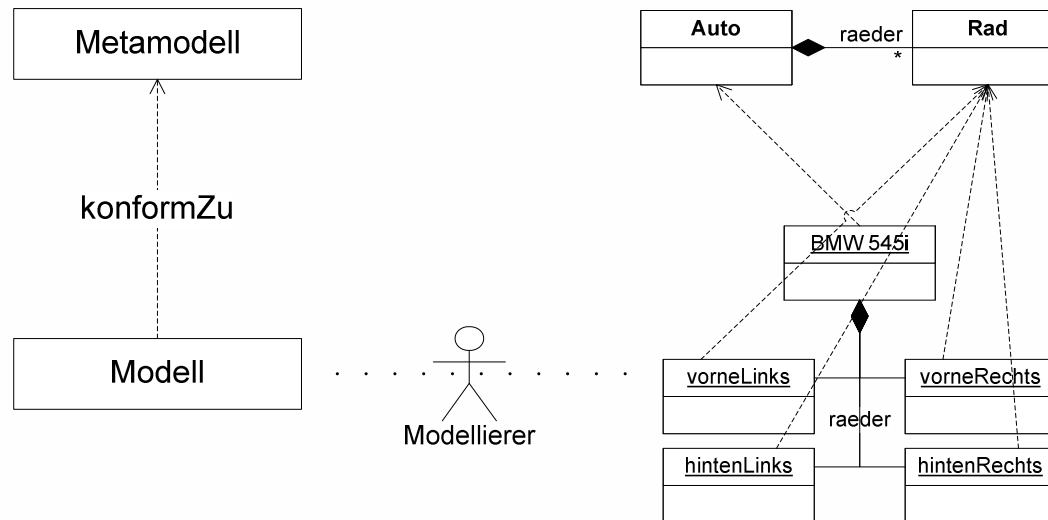
Entwicklung einer DSL

- Metamodellierung
- ...

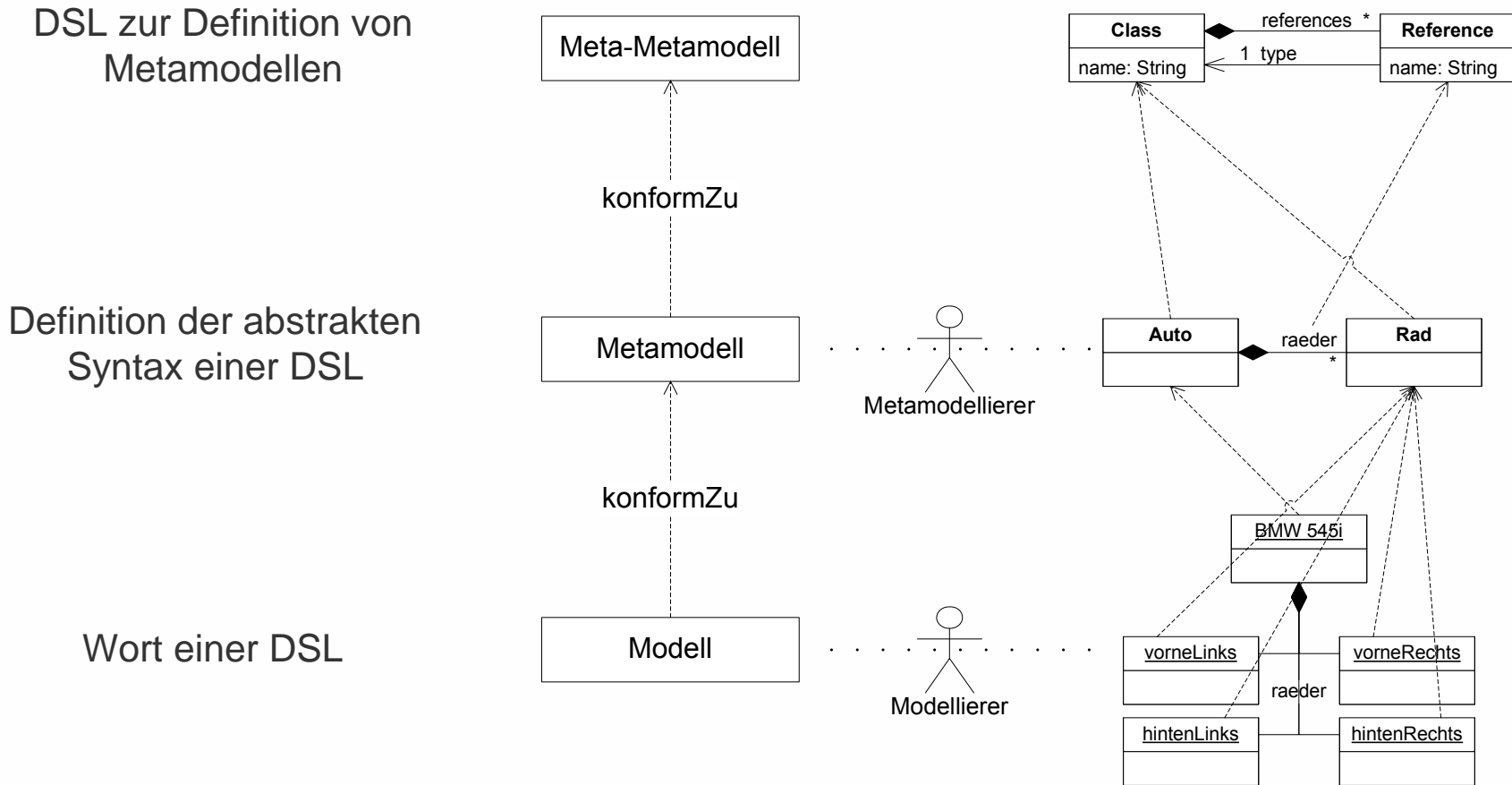
Einführung – Metamodellierung

Definition der abstrakten
Syntax einer DSL

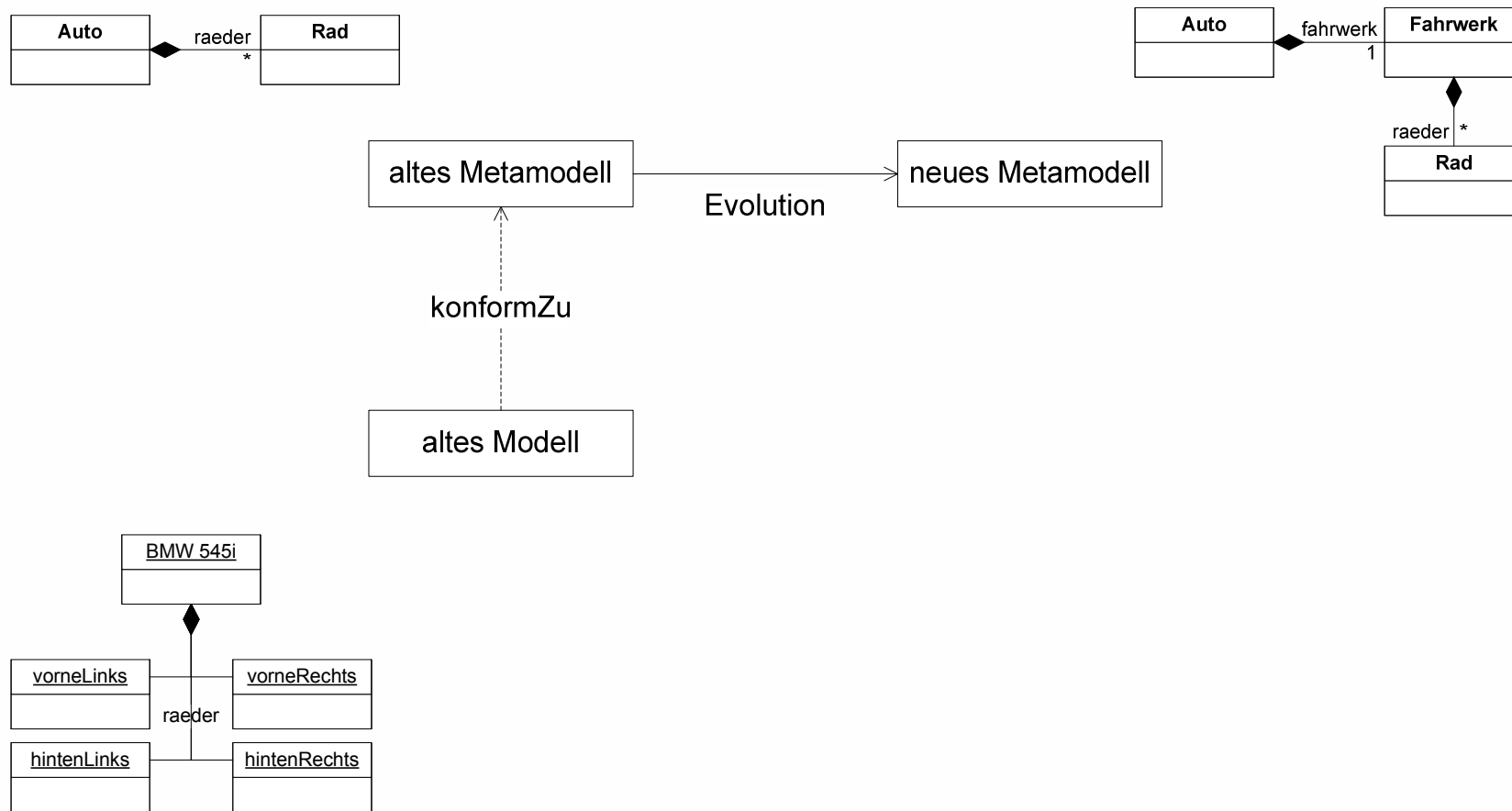
Wort einer DSL



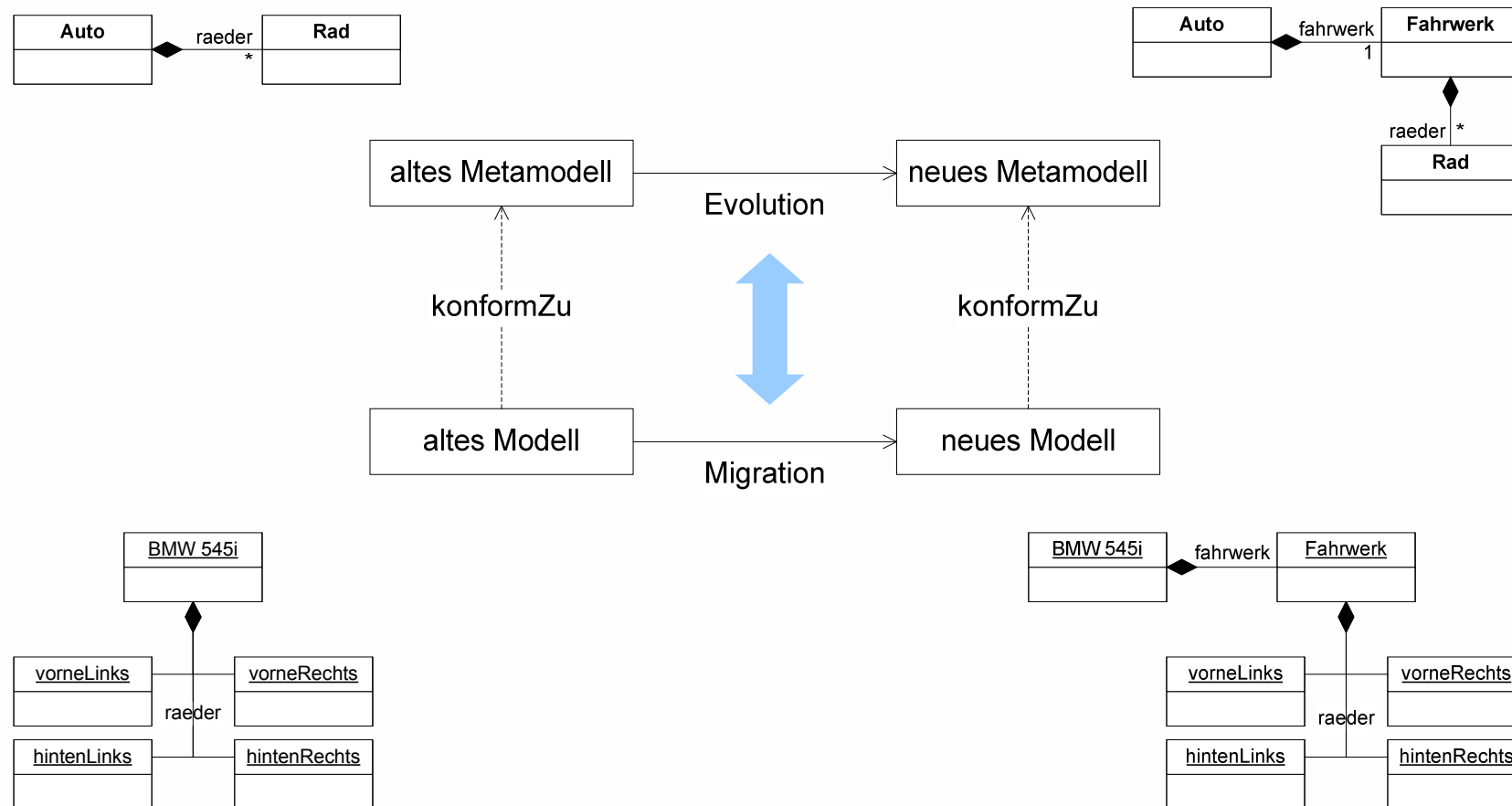
Einführung – Metamodellierung



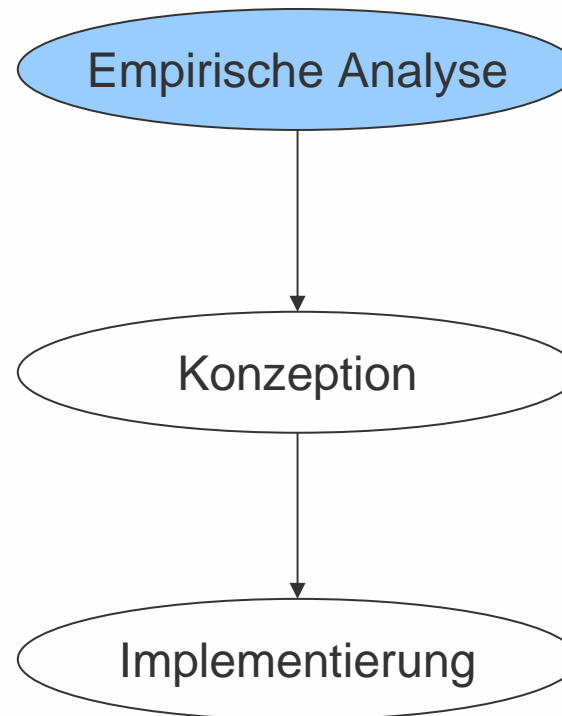
Problem – Gekoppelte Evolution



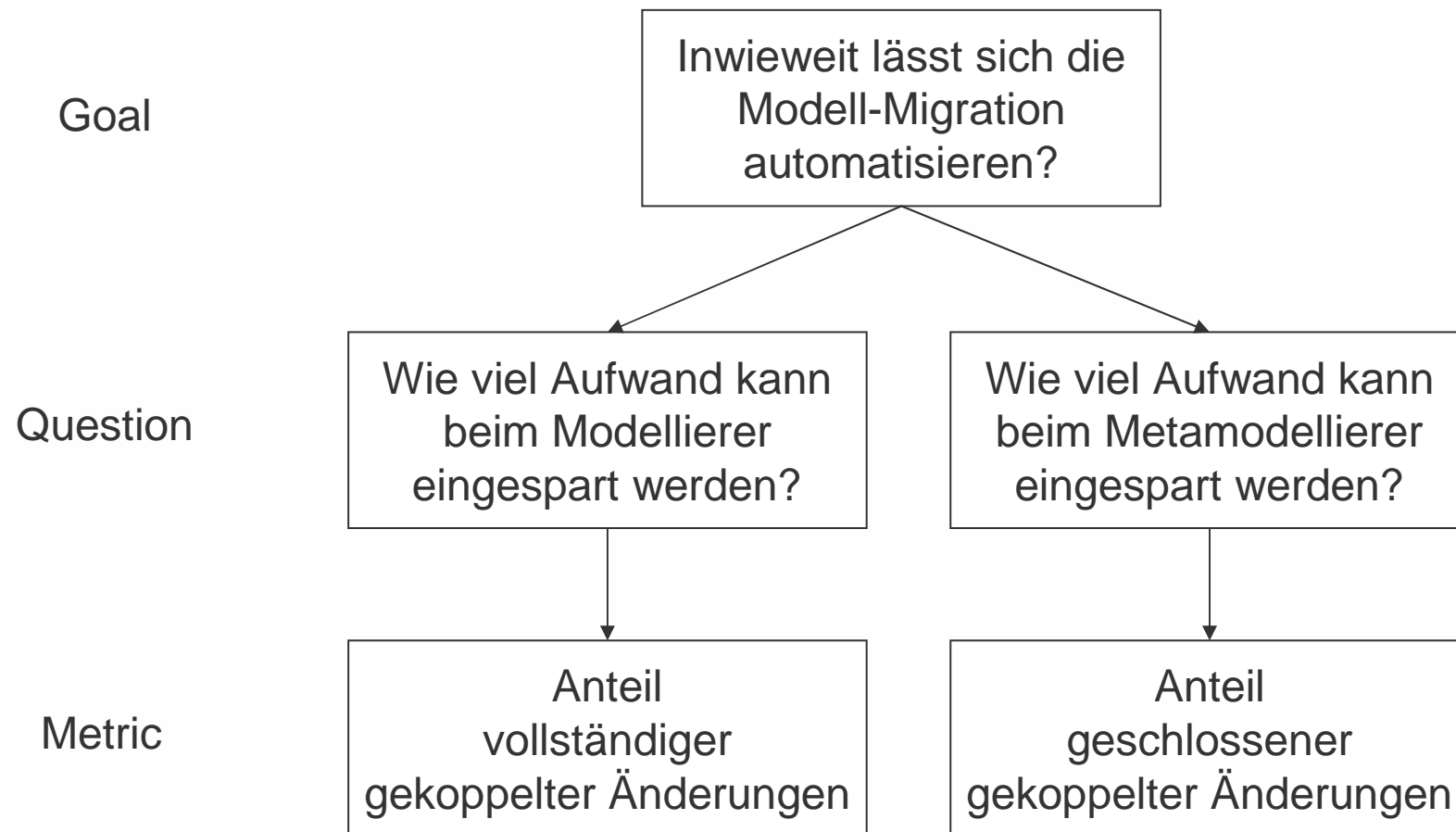
Problem – Gekoppelte Evolution



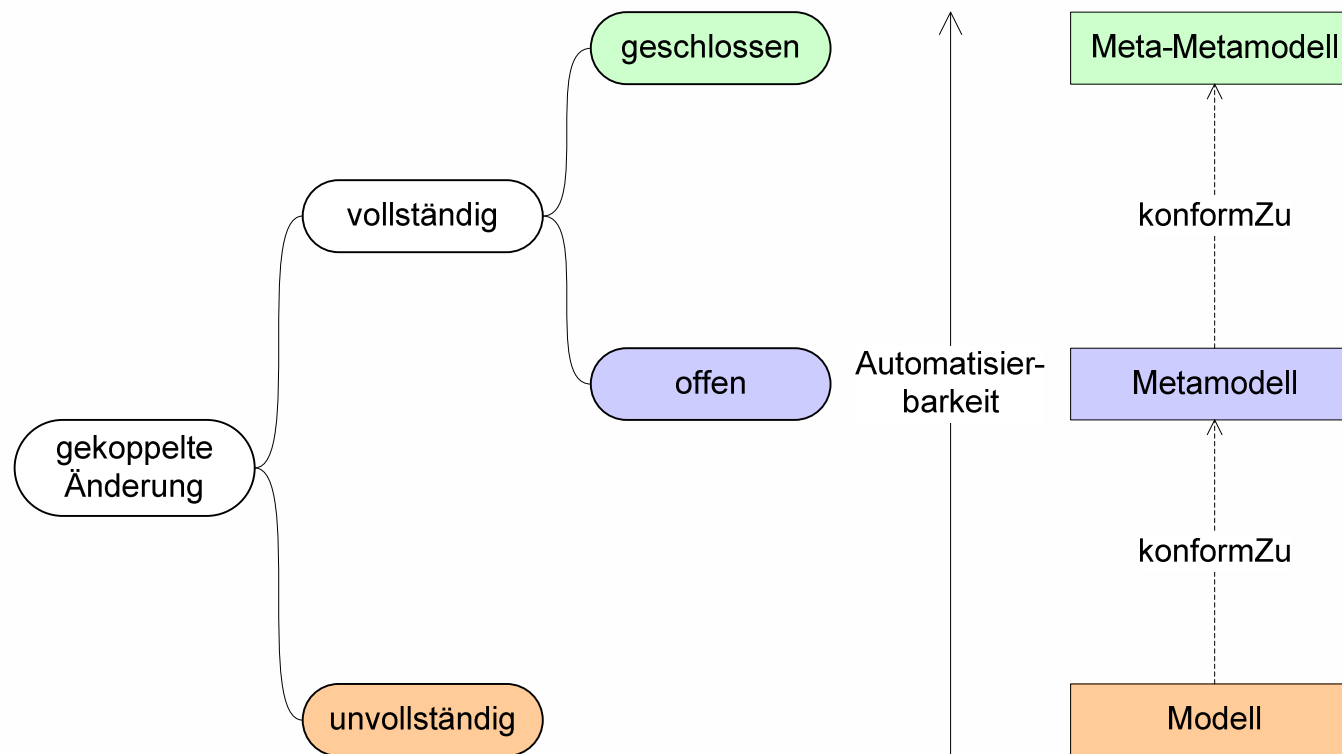
Vorgehensweise



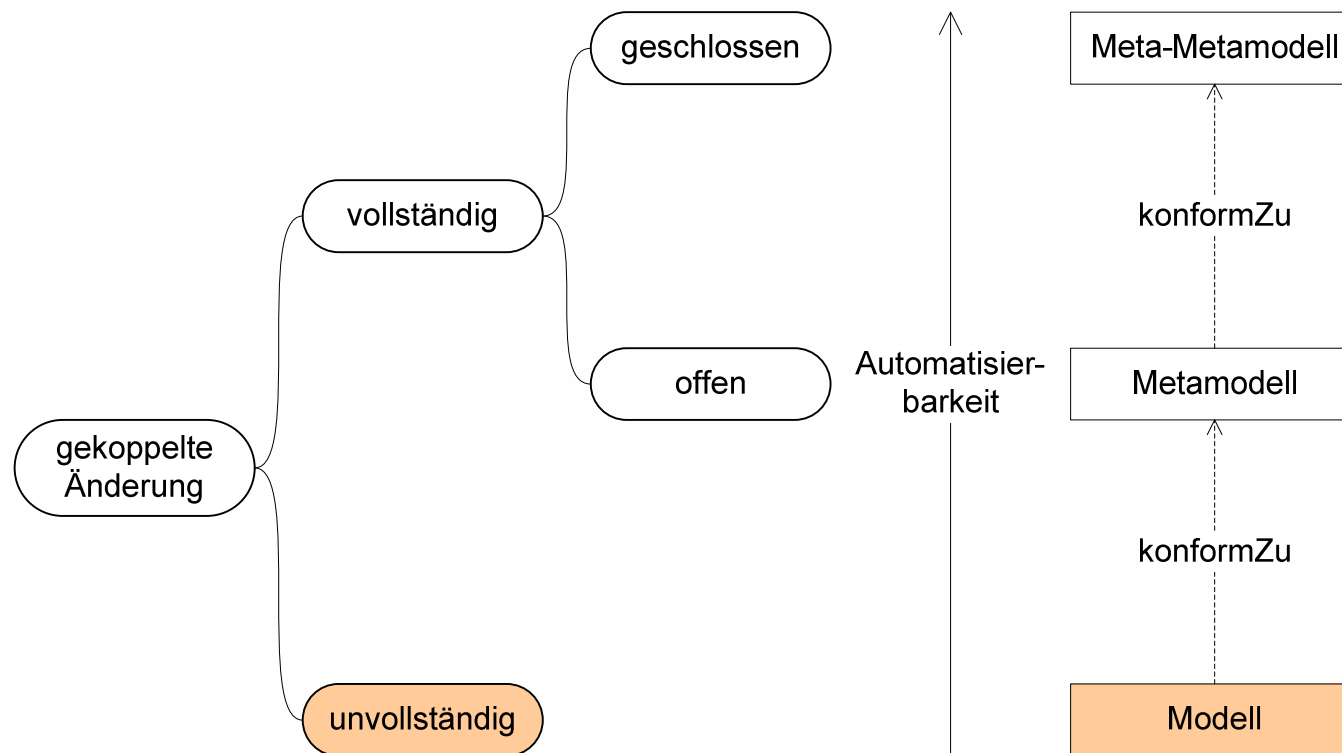
Analyse – Ziele



Analyse – Klassifikation von gekoppelten Änderungen



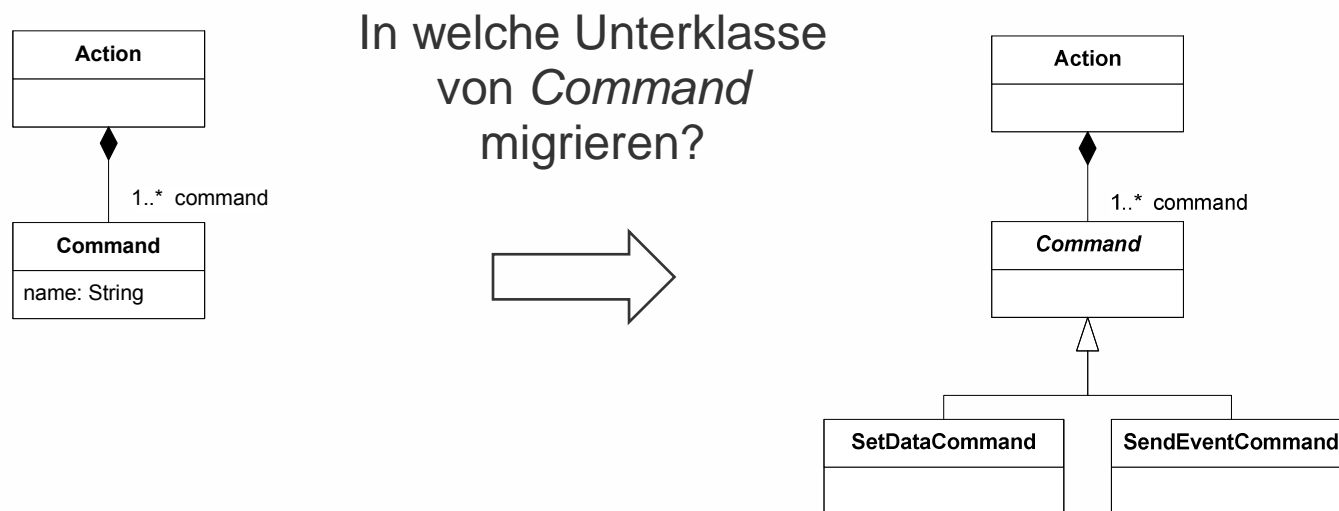
Analyse – Klassifikation von gekoppelten Änderungen



Analyse – Unvollständige gekoppelte Änderung

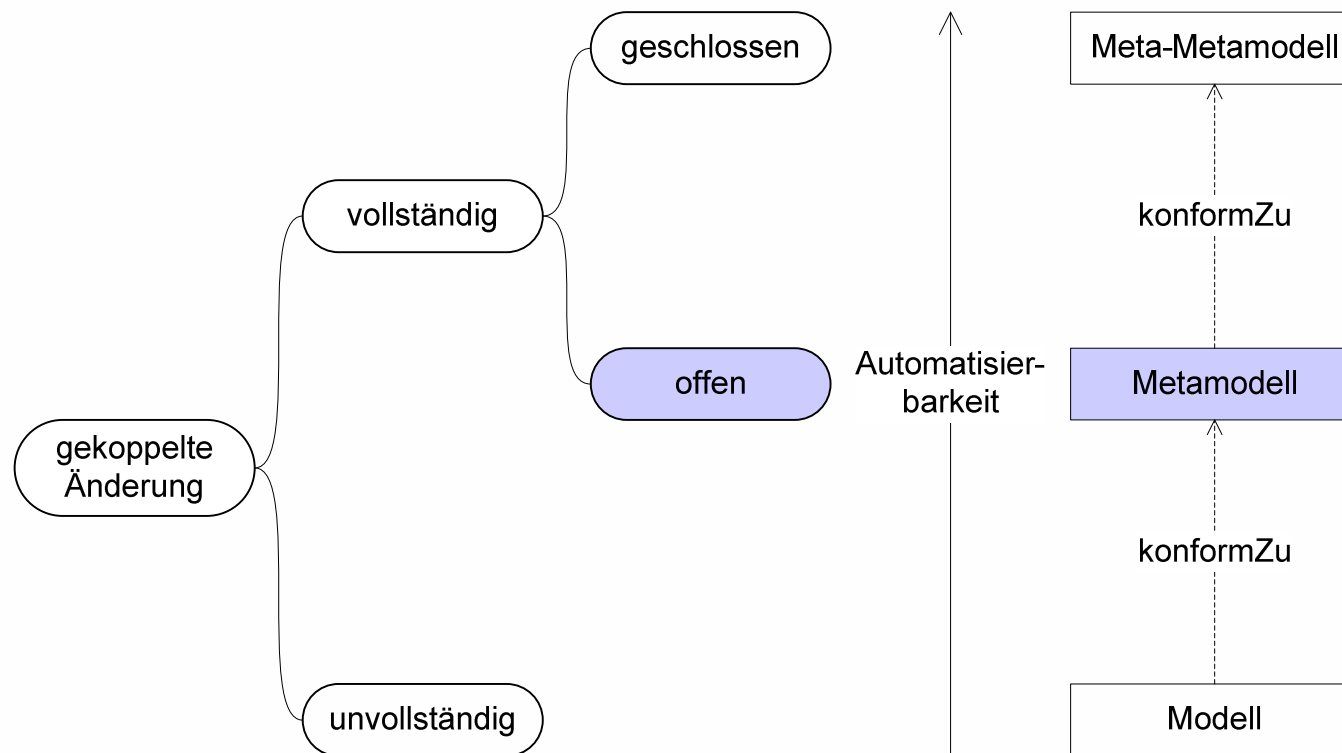
Merkmal: Modelliererwissen erforderlich zur Durchführung der Migration

Beispiel: Verfeinerung von *Command*



Auswirkung: Modellierer-Interaktion bei Migration notwendig

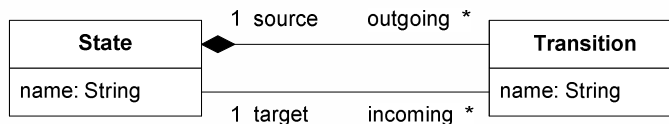
Analyse – Klassifikation von gekoppelten Änderungen



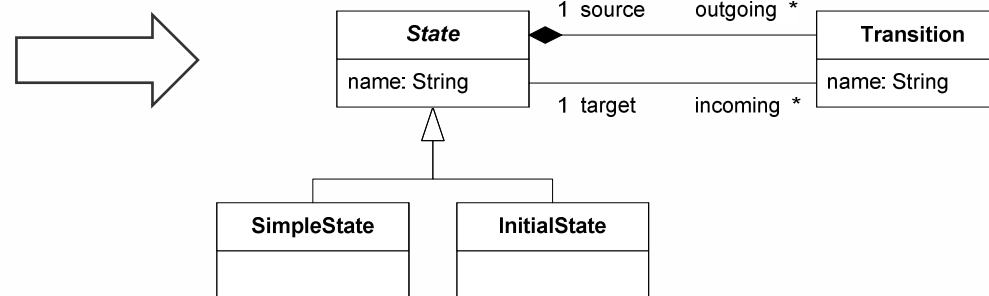
Analyse – Offene gekoppelte Änderung

Merkmal: Domänenwissen erforderlich zur Spezifikation der Migration

Beispiel: Verfeinerung von *State*

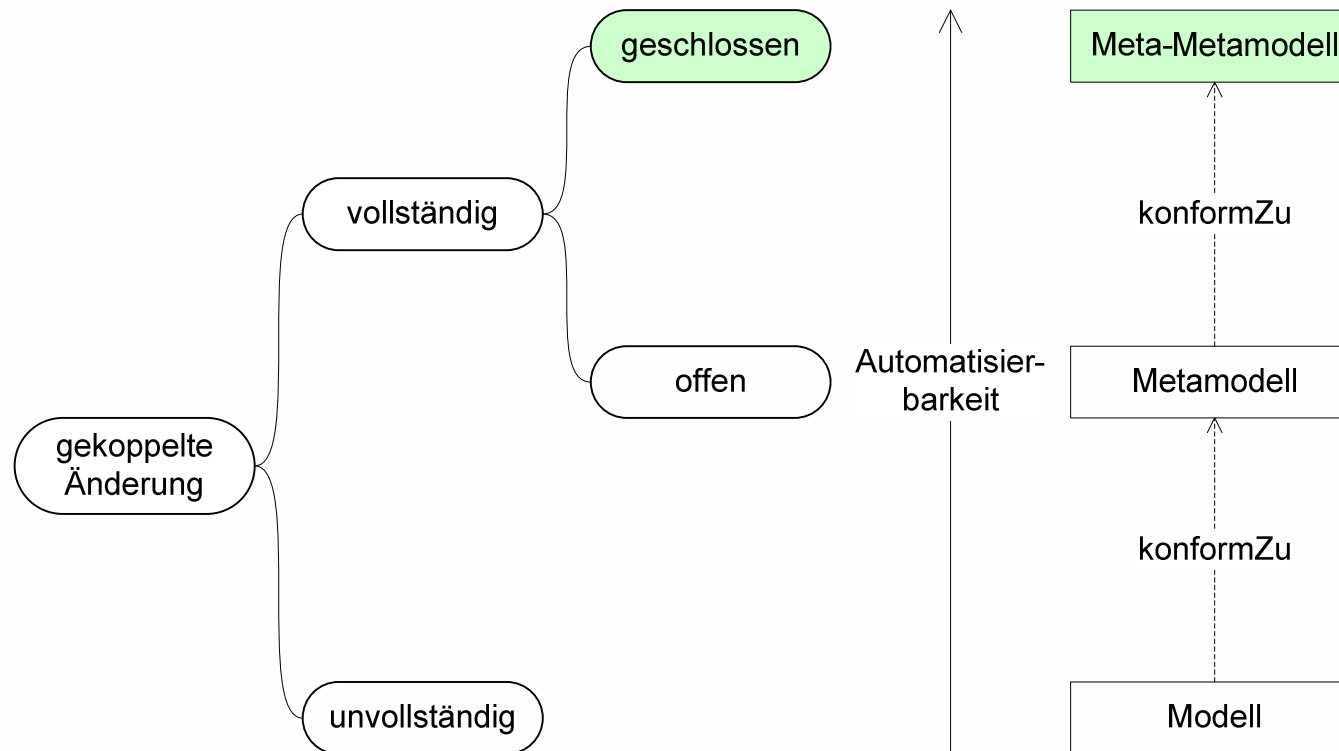


Zustände ohne eingehende
Transition nach *InitialState*
migrieren



Auswirkung: Anwendbarkeit nur innerhalb der Domäne

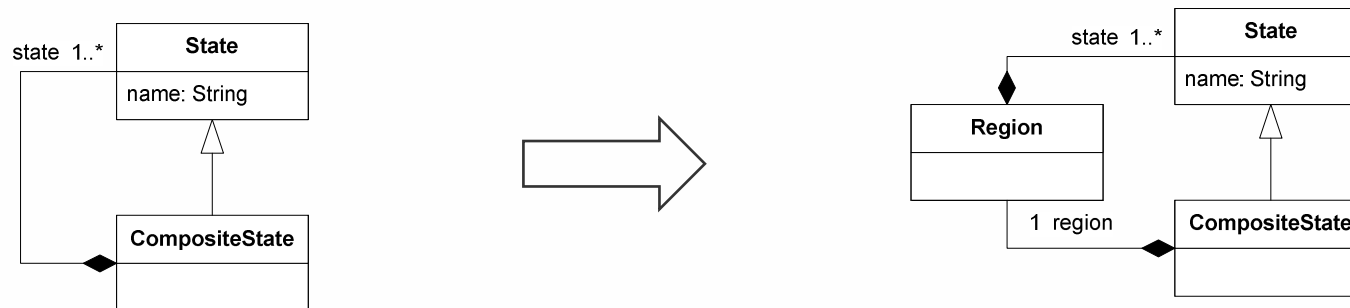
Analyse – Klassifikation von gekoppelten Änderungen



Analyse – Geschlossene gekoppelte Änderung

Merkmal: kein Domänenwissen erforderlich zur Spezifikation der Migration

Beispiel: Einführung von *Region*



Referenz in Teilklasse
extrahieren

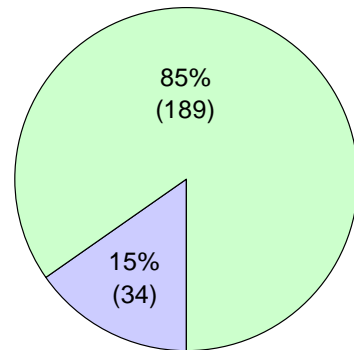
Auswirkung: Wiederverwendbarkeit über Domänen hinweg

Analyse - Ansatz

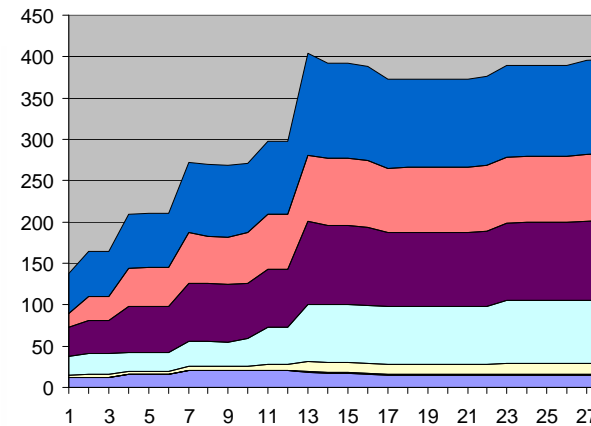
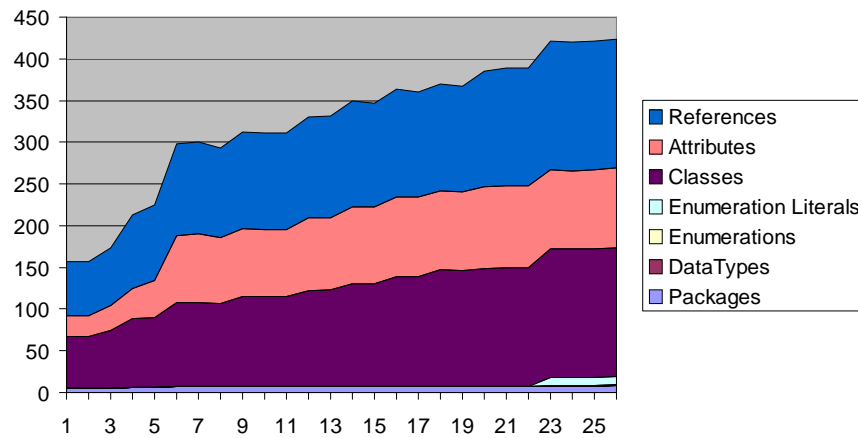
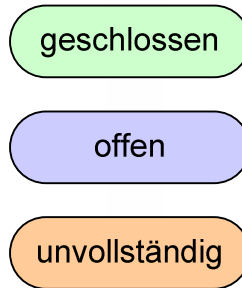
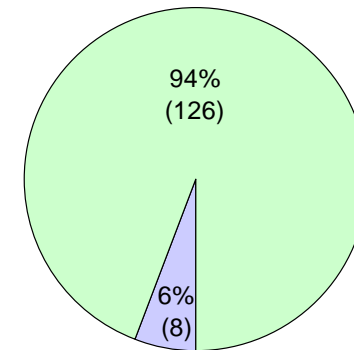
1. Entnahme von Metamodell-Versionen aus dem Versionskontrollsystem
→ Schnappschüsse
2. Vergleich aufeinander folgender Metamodell-Versionen
→ Menge von primitiven Änderungen
3. Zusammenfassung von primitiven Änderungen basierend auf Änderungen entsprechender Modelle
→ gekoppelte Änderung
4. Klassifikation der gekoppelten Änderungen

Analyse – Ergebnis

FLUID



MMITest



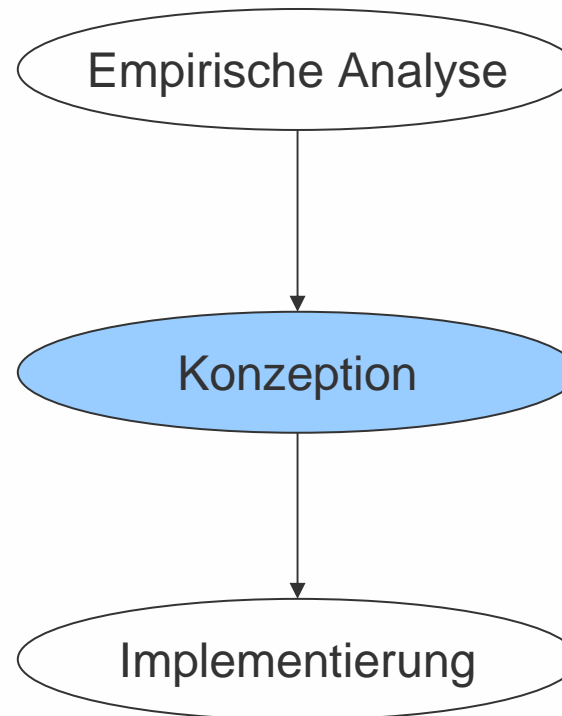
Analyse – Konsequenzen

- Keine unvollständigen gekoppelten Änderungen
 - Vernachlässigung
(manuelle Anpassung des Modells auch nach Migration möglich)
 - Geringer Anteil offener gekoppelter Änderungen
 - Spezifikation beliebiger Modell-Migrationen für Änderungen am Metamodell
 - Hoher Anteil geschlossener gekoppelter Änderungen
 - Wiederverwendung der Spezifikation von gekoppelten Änderungen über Metamodelle hinweg
- Möglichkeit der beliebigen Kombination

Analyse – Bedrohung der Gültigkeit

1. Entnahme von Metamodell-Versionen aus dem Versionskontrollsystem
 - Annahme, dass mit einem Commit eine neue Version entsteht
2. Vergleich aufeinander folgender Metamodell-Versionen
 - Problem der Element-Gleichheit
 - Intermediäre Änderungen nicht berücksichtigt
3. Zusammenfassung von primitiven Änderungen basierend auf Änderungen entsprechender Modelle
 - Modelle nicht verfügbar für alle Metamodell-Versionen
4. Klassifikation der gekoppelten Änderungen

Vorgehensweise



COPE – Offene Kopplung

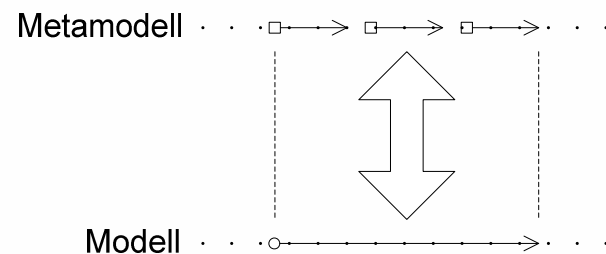
offen

Formalisierung der

- Evolution: Sequenz von primitiven Änderungen
- Migration: Modell-Transformation

Offene Kopplung

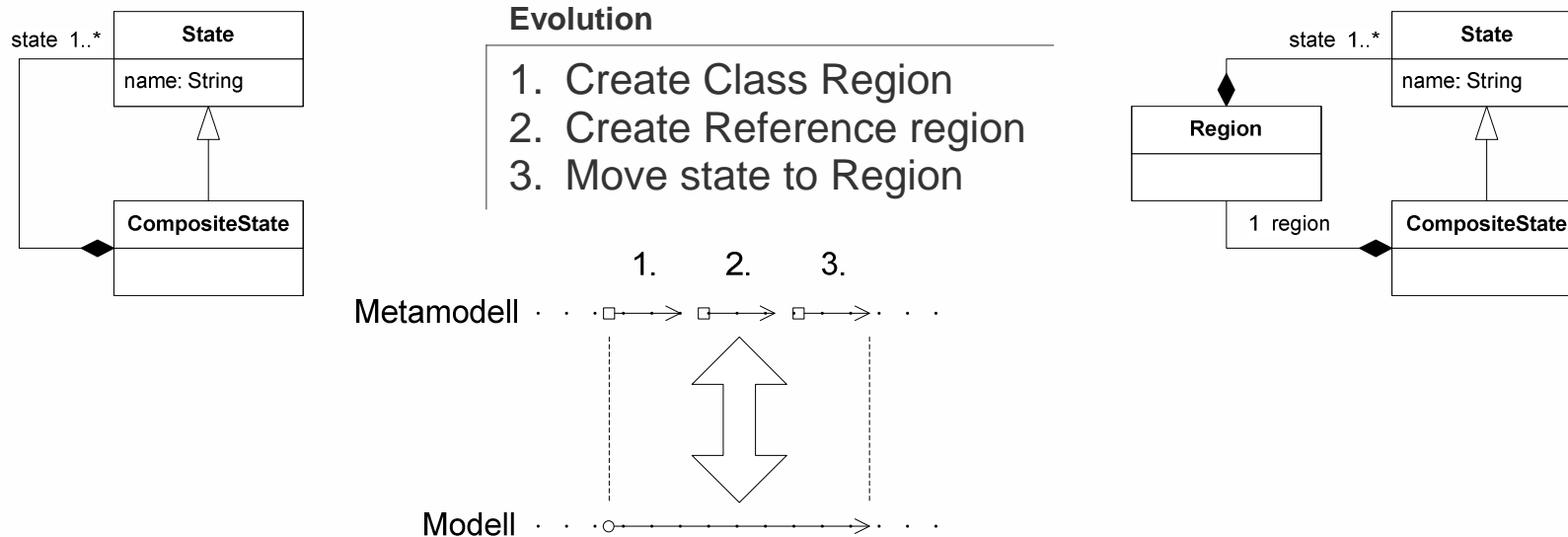
- manuelle Zuordnung von Migration zu Evolution
 - automatische Migration von alten Modellen
- Beschreibung aller vollständigen gekoppelten Änderungen



COPE – Offene Kopplung

offen

Beispiel: Einführung von *Region*



Evolution

1. Create Class Region
2. Create Reference region
3. Move state to Region

Migration

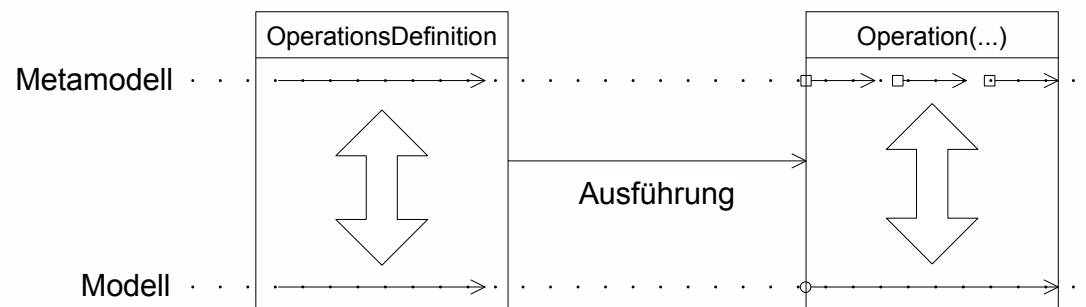
```

For each compositeState instanceof CompositeState
  Create Region region
  Set compositeState.region to region
  Move compositeState.state to region.state
    
```

COPE – Geschlossene Kopplung geschlossen

Wiederverwendung durch gekoppelte Operation

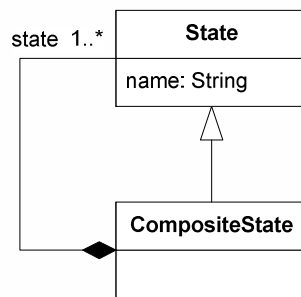
- Definition einer gekoppelten Operation unabhängig vom Metamodell
 - Ausführung der Operation auf konkretem Metamodell
- Beschreibung von geschlossenen gekoppelten Änderungen



COPE – Geschlossene Kopplung

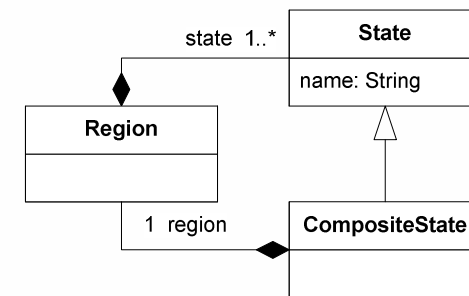
geschlossen

Beispiel: Einführung von *Region*

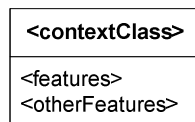


Evolution

1. Create Class *Region*
2. Create Reference *region*
3. Move *state* to *Region*

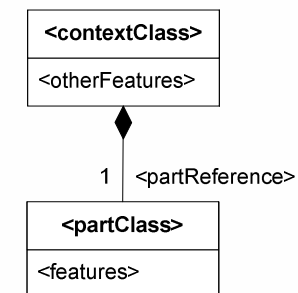


↓ Parametrisierung



Evolution

- Create Class *<partClass>*
 Create Reference *<partReference>*
 For each *<feature>* in *<features>*
 Move *<feature>* to *<partClass>*



COPE – Geschlossene Kopplung

geschlossen

Beispiel: Einführung von *Region*

Migration

```
For each compositeState instanceOf CompositeState
  Create Region region
  Set compositeState.region to region
  Move compositeState.state to region.state
```

↓
Parametrisierung

Migration

```
For each contextObject instanceOf <contextClass>
  Create <partClass> partObject
  Set contextObject.<partReference> to partObject
  For each <feature> in <features>
    Move contextObject.<feature> to partObject.<feature>
```

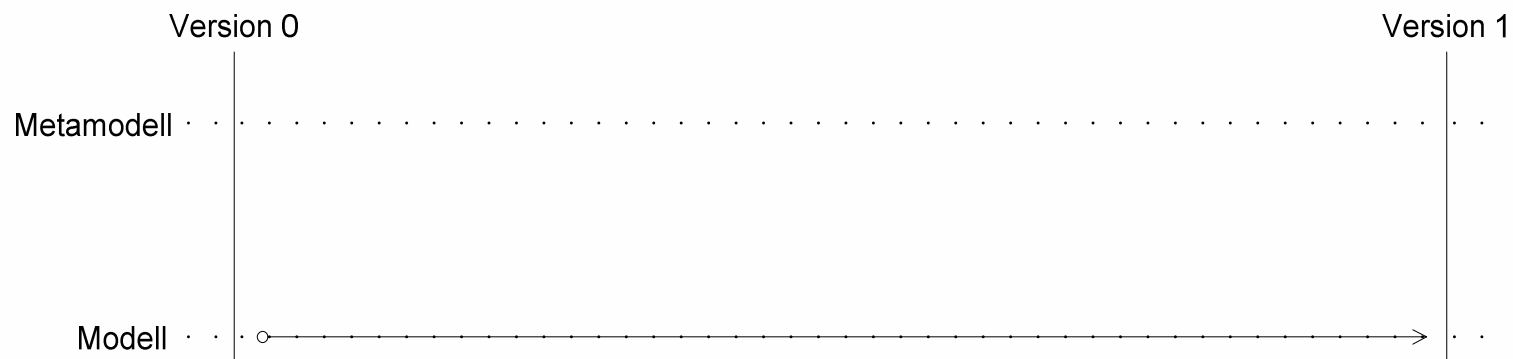
COPE – Gekoppelte Evolution

geschlossen

offen

ohne COPE

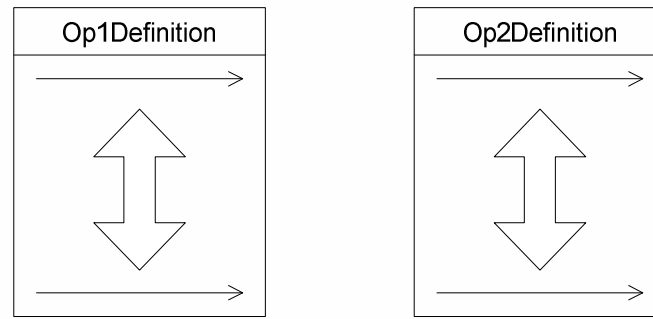
- alte und neue Version des Metamodells
- Spezifikation der kompletten Migration



COPE – Gekoppelte Evolution

geschlossen

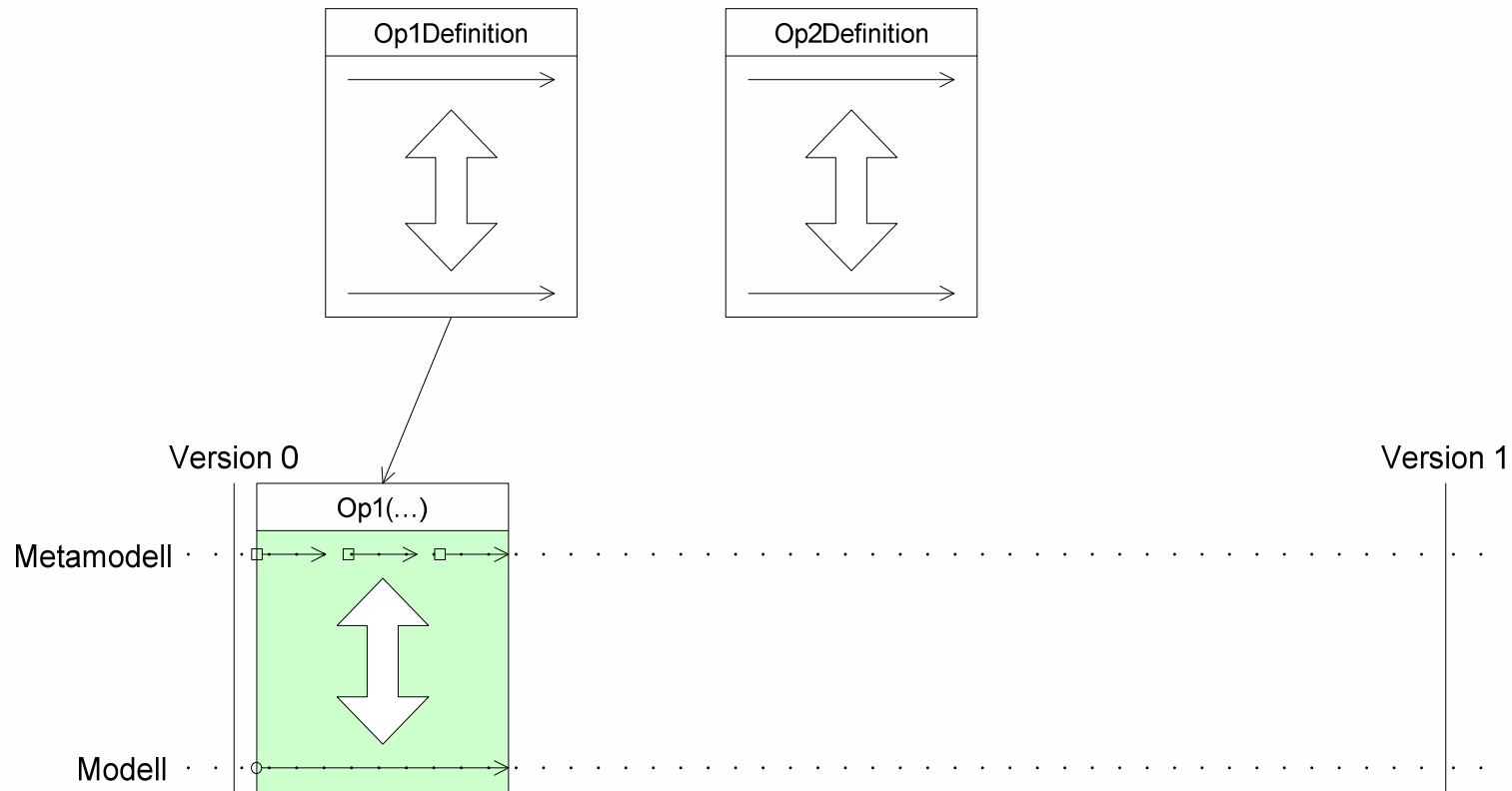
offen



COPE – Gekoppelte Evolution

geschlossen

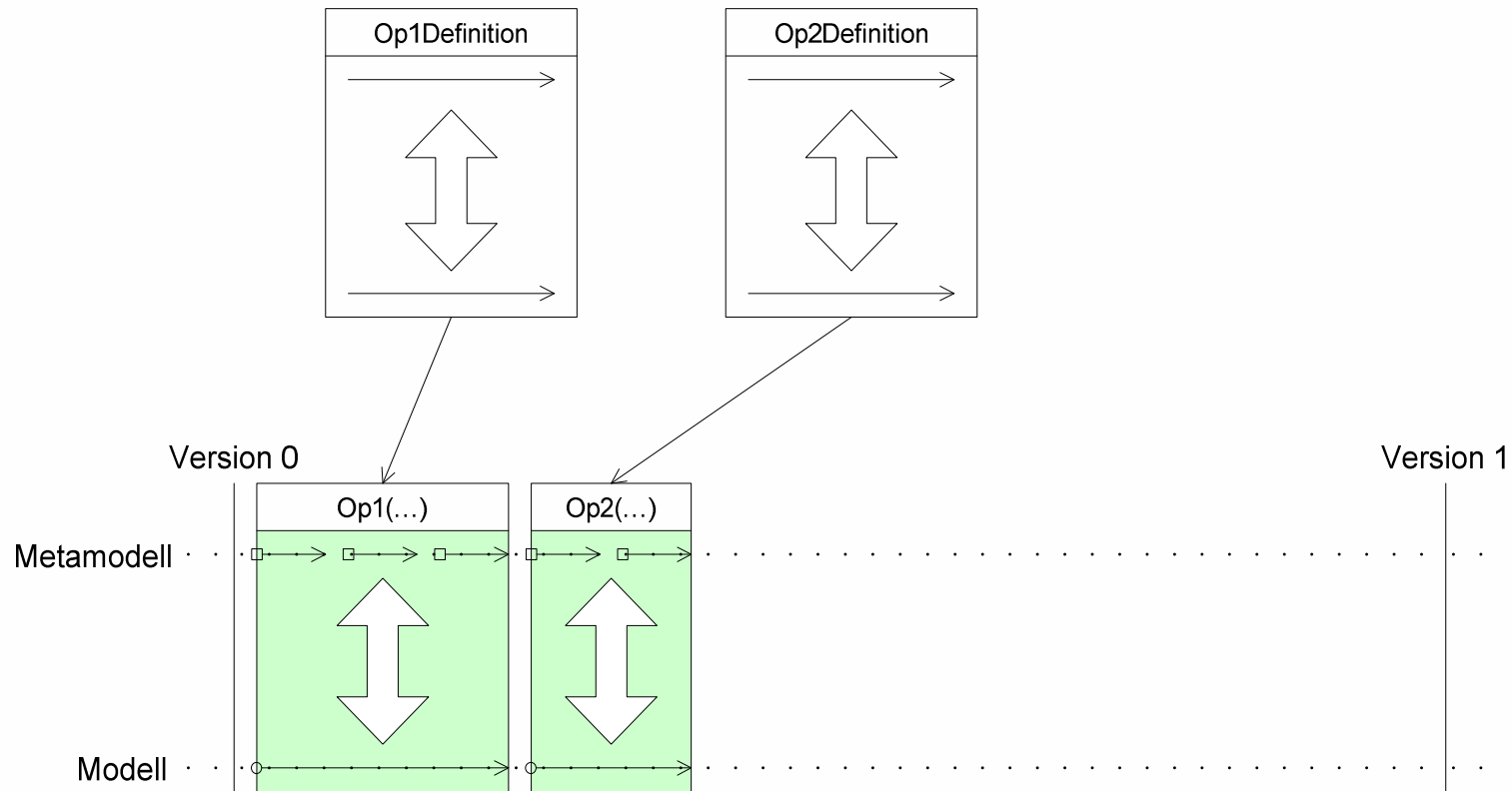
offen



COPE – Gekoppelte Evolution

geschlossen

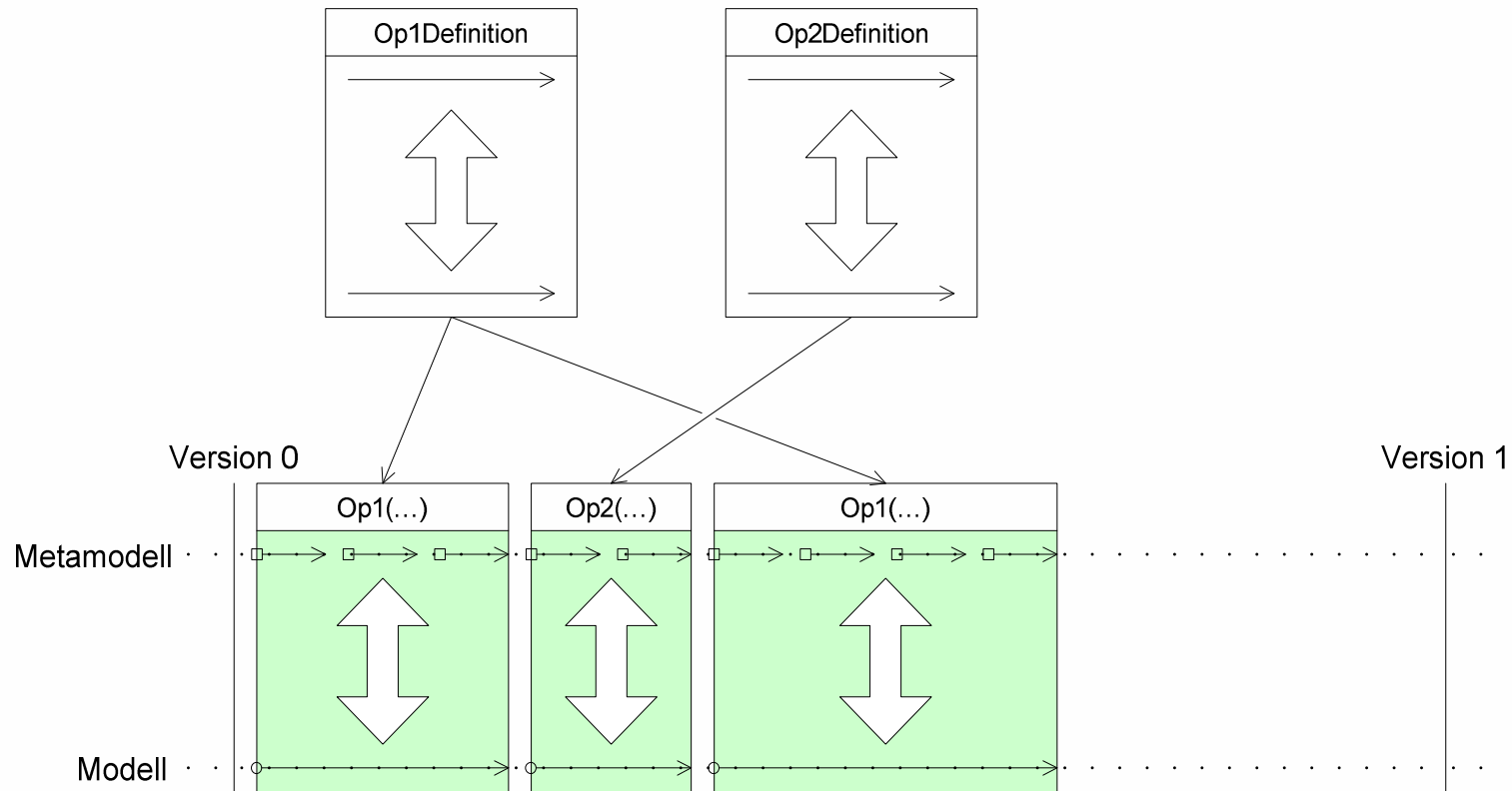
offen



COPE – Gekoppelte Evolution

geschlossen

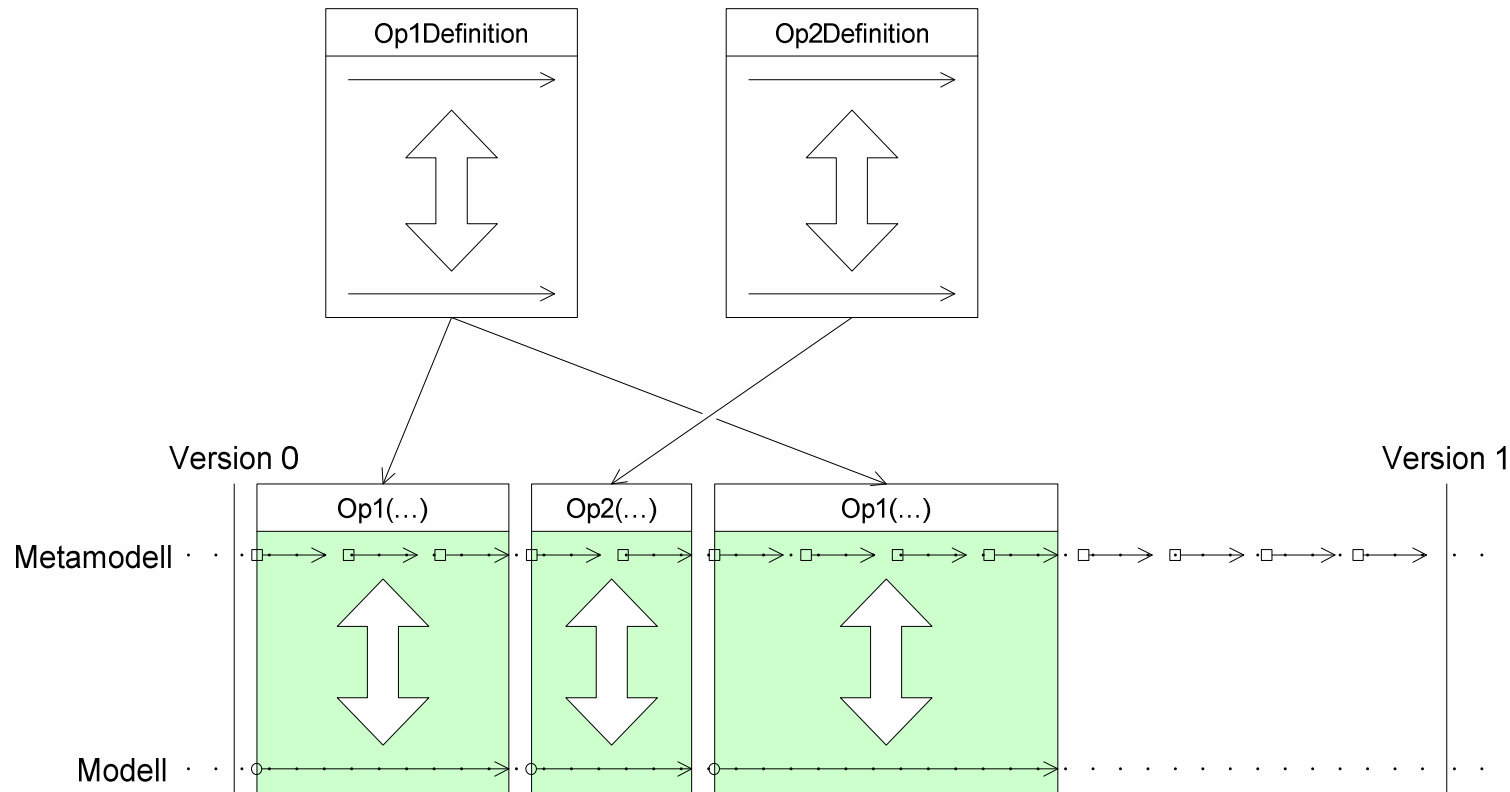
offen



COPE – Gekoppelte Evolution

geschlossen

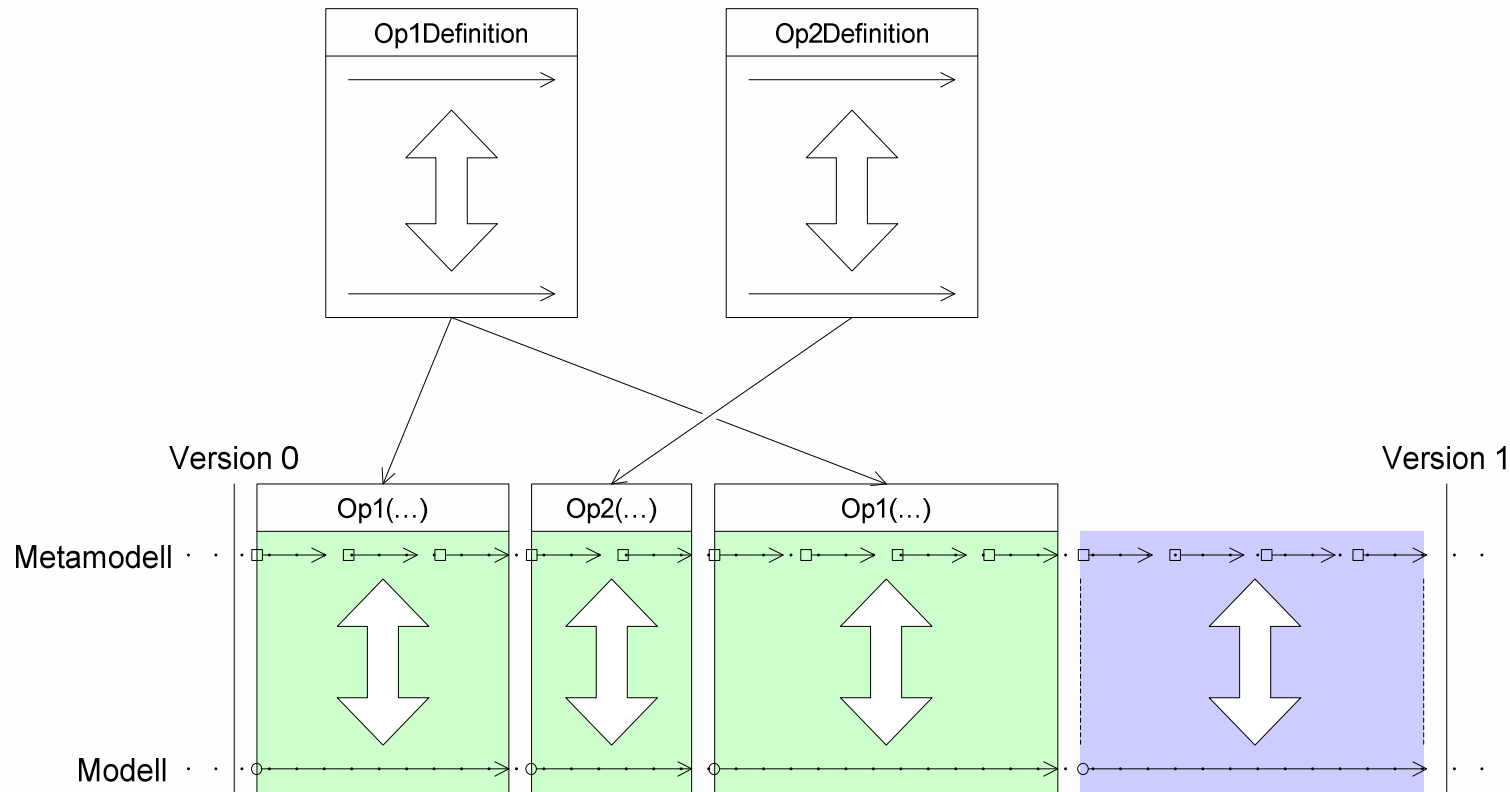
offen



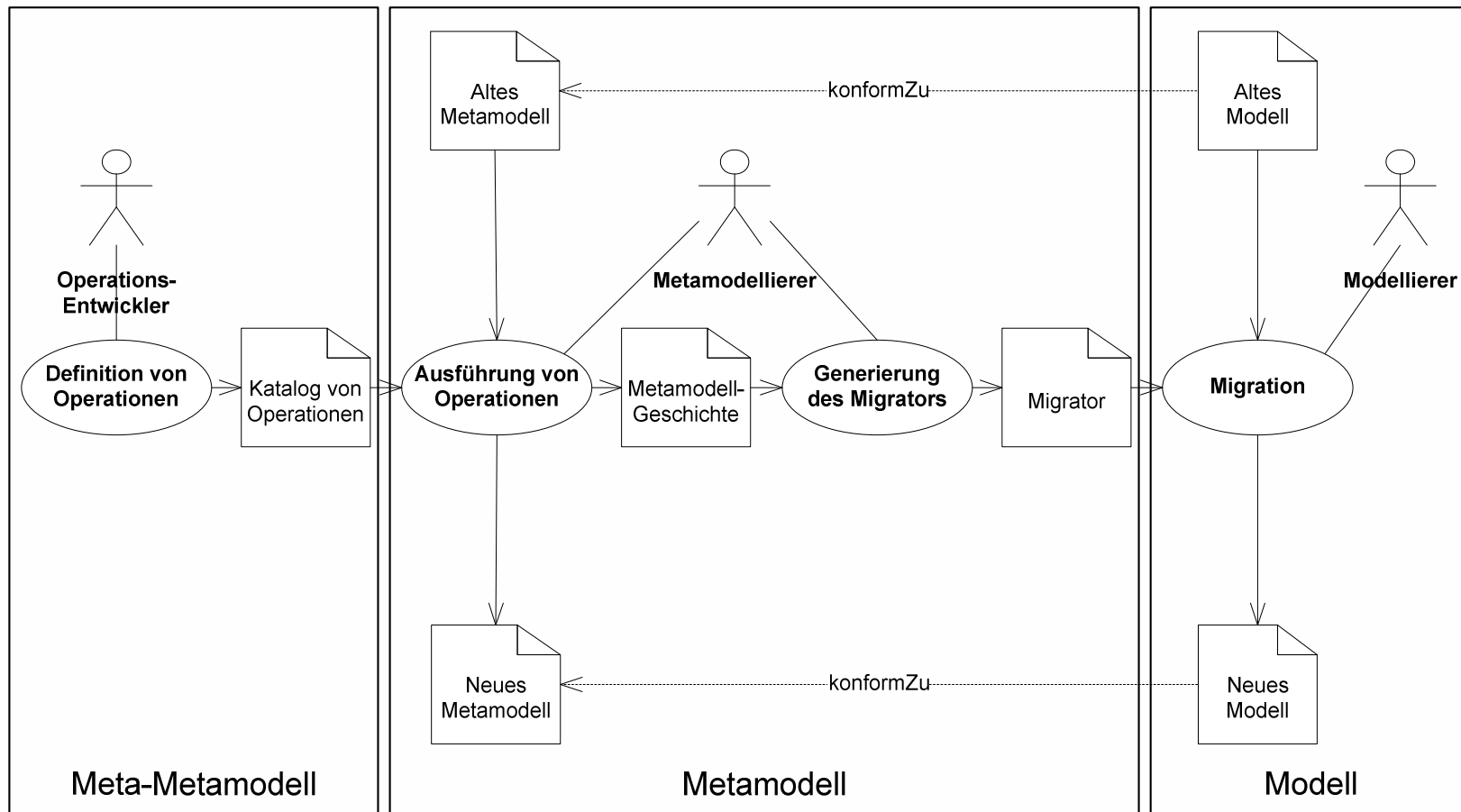
COPE – Gekoppelte Evolution

geschlossen

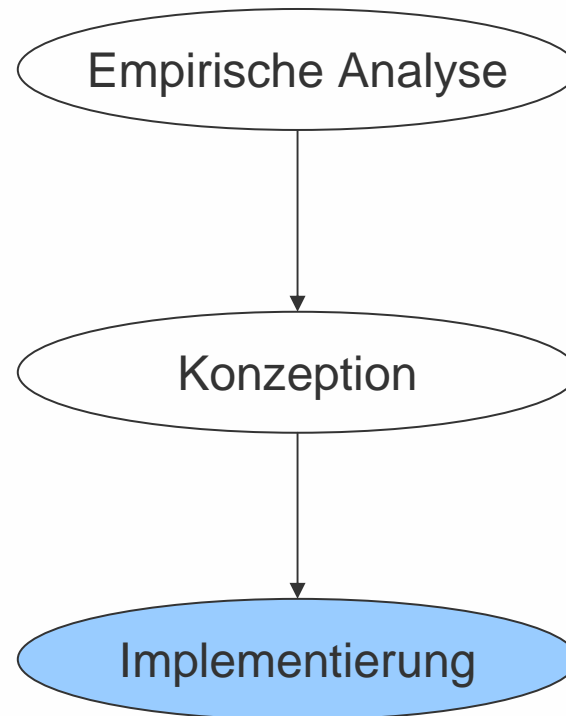
offen



COPE – Prozess



Vorgehensweise



Implementierung – Verwendete Technologien

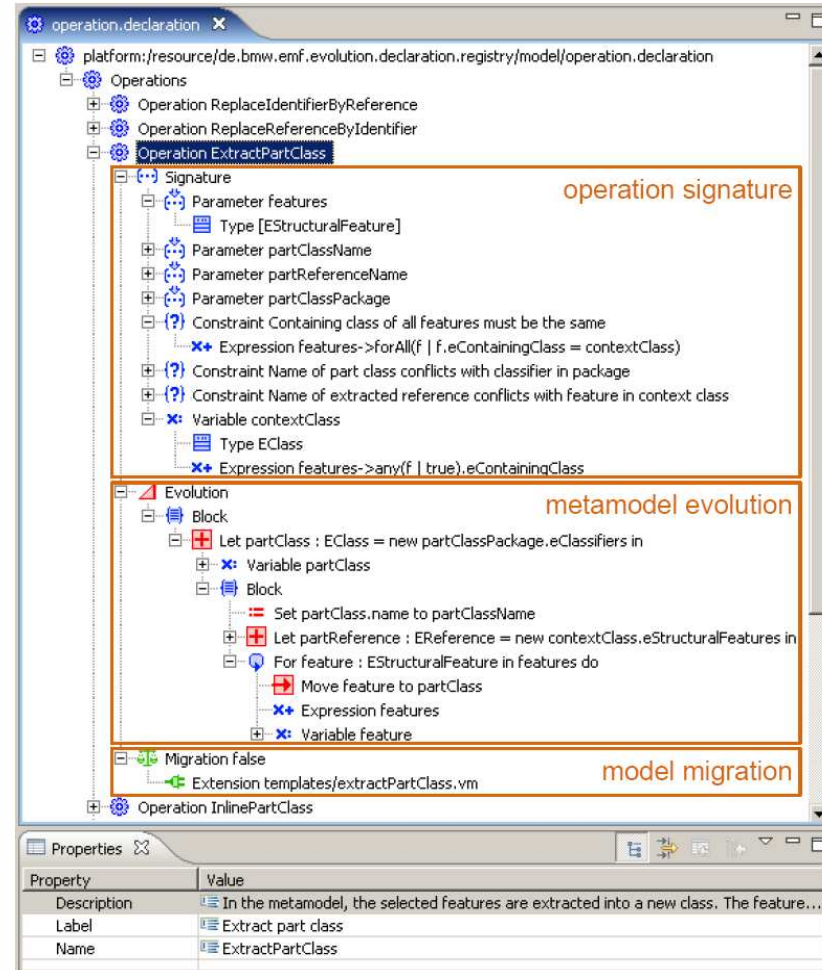
- Eclipse Modeling Framework (EMF)
 - Metamodellierungs-Umgebung
- Model Development Tools OCL (OCL)
 - Ausdrücke zur Deklaration von gekoppelten Operationen
- Atlas Transformation Language (ATL)
 - Spezifikation der Modell-Migration
- Velocity Template Language (VTL)
 - Spezifikation der Modell-Migration einer gekoppelten Operation unabhängig vom Metamodell



Implementierung – Gekoppelte Operation

Definition von Operationen

- Signatur
 - Parameter
 - Constraint (OCL)
- Metamodell-Evolution
 - Kontrollstrukturen
 - Primitive
 - Änderungsoperationen
- Modell-Migration
 - Template über ATL (VTL)



The screenshot displays the 'operation.declaration' window in an IDE, showing the definition of the 'Operation ExtractPartClass'. The window is divided into three main sections, each highlighted with an orange box and a label:

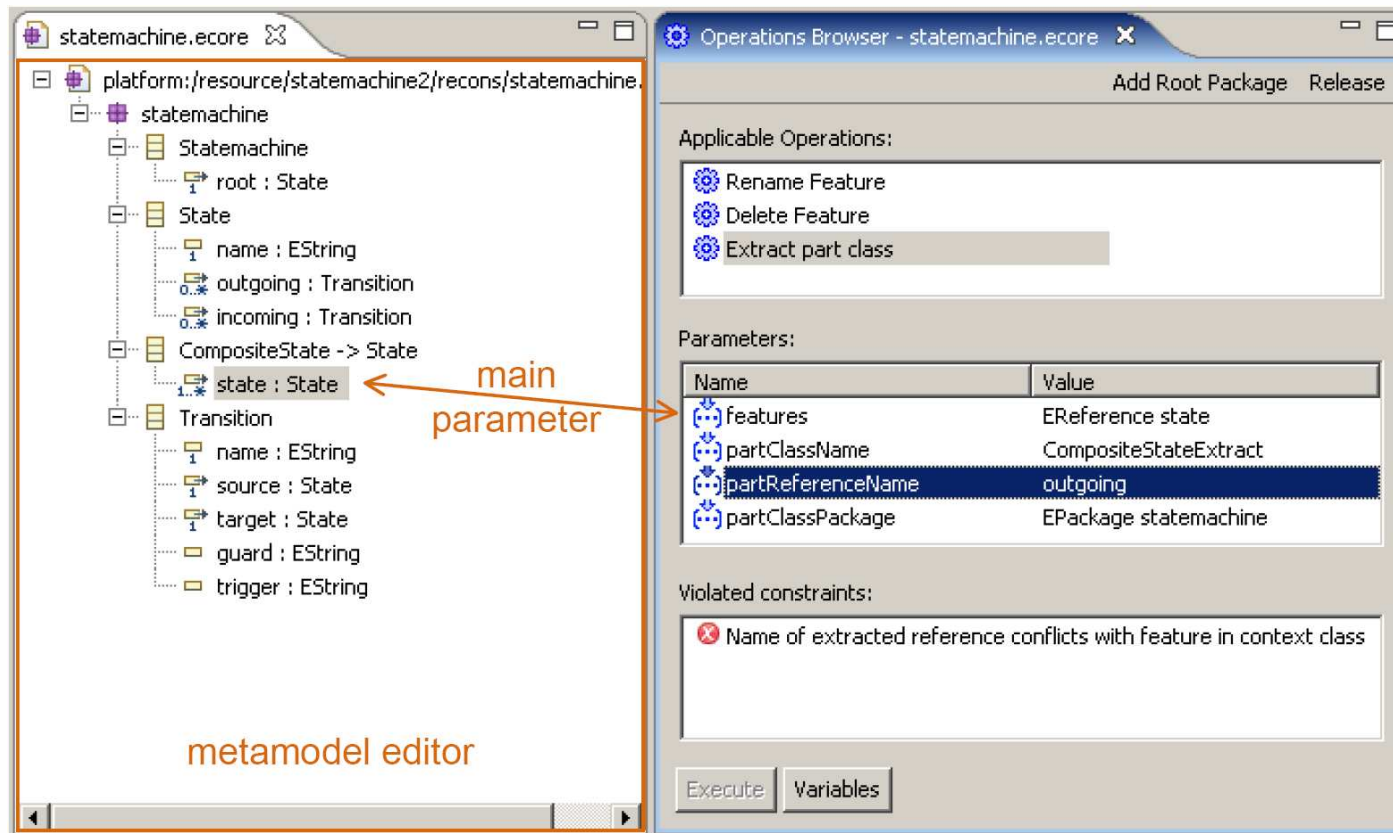
- operation signature:** This section defines the parameters and constraints for the operation. It includes:
 - Parameter features: Type [EStructuralFeature]
 - Parameter partClassName
 - Parameter partReferenceName
 - Parameter partClassPackage
 - Constraint: Containing class of all features must be the same
 - Expression: features->forall(f | f.eContainingClass = contextClass)
 - Constraint: Name of part class conflicts with classifier in package
 - Constraint: Name of extracted reference conflicts with feature in context class
 - Variable: contextClass
 - Type: EClass
 - Expression: features->any(f | true).eContainingClass
- metamodel evolution:** This section defines the evolution of the metamodel. It includes:
 - Block: Let partClass : EClass = new partClassPackage.eClassifiers in
 - Variable: partClass
 - Block: Set partClass.name to partClassName
 - Let partReference : EReference = new contextClass.eStructuralFeatures in
 - For feature : EStructuralFeature in features do
 - Move feature to partClass
 - Expression features
 - Variable feature
- model migration:** This section defines the migration of the model. It includes:
 - Migration: false
 - Extension: templates/extractPartClass.vm

At the bottom of the window, there is a 'Properties' panel with the following information:

Property	Value
Description	In the metamodel, the selected features are extracted into a new class. The feature...
Label	Extract part class
Name	ExtractPartClass

Implementierung – Gekoppelte Operation

Kontextsensitives Anbieten von Operation



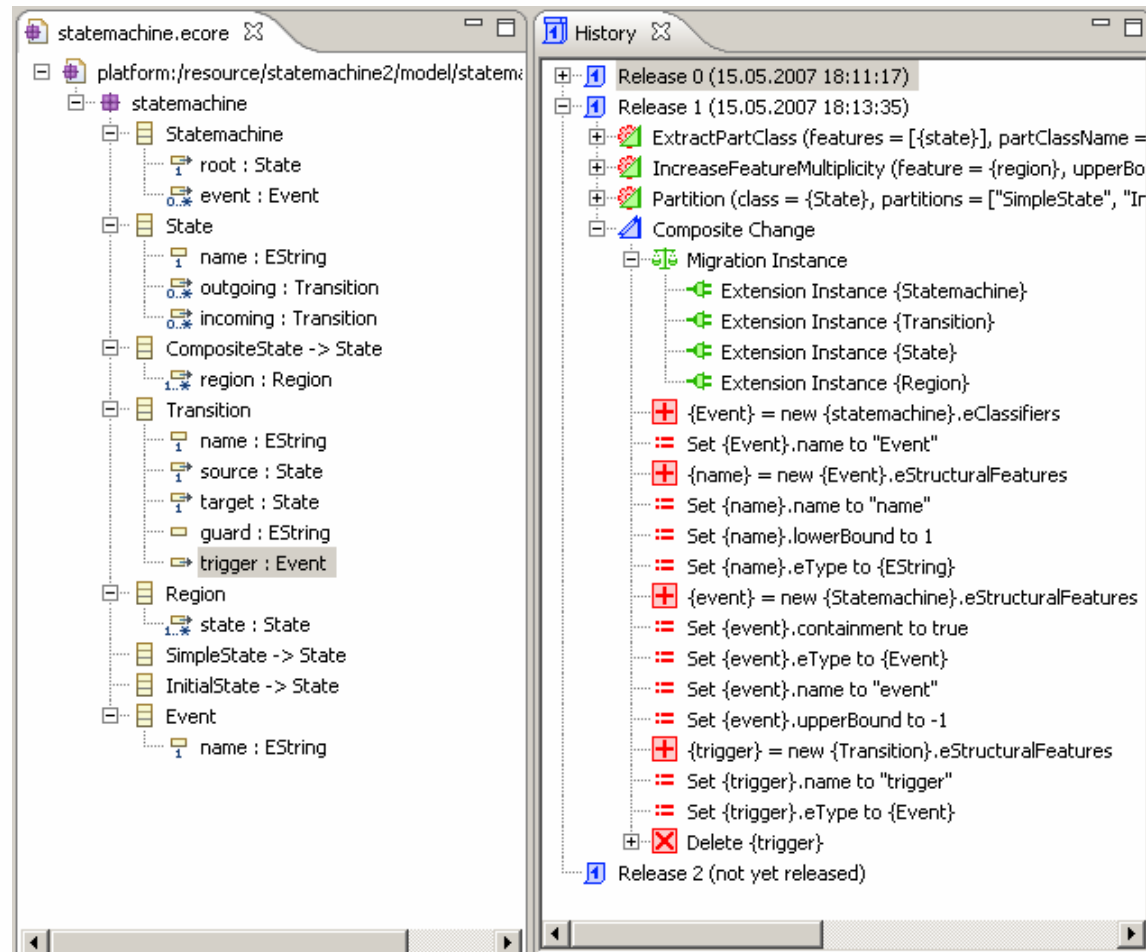
Implementierung – Metamodell-Geschichte

Automatisches Mitspeichern
der Änderungen

- Instanz einer Operation
- Sequenz von primitiven
Änderungen

Offene Kopplung

- Zusammenfassung von
primitiven Änderungen
- Zuordnung einer
Migration (ATL)



Implementierung – Offene Kopplung

Edit code of extension instance

```

rule CompositeState_statemachine
extends State_statemachine {
  from
  old: oldVersion!statemachine::CompositeState
  to
  new: newVersion!statemachine::CompositeState (
    region <- region.asSequence()
  ),
  region: newVersion!statemachine::Region (
    state <- old.state
  )
}
    
```

overwritten transformation definition

platform:/resource/statemachine2/non_coupled/.reconstr

- statemachine
 - Statemachine
 - State
 - CompositeState -> State
 - state : State
 - Transition

metamodel before evolution

Property	Value
Changeable	true
Container	false
Containment	true
Default Value	
Default Value Literal	
Derived	false
EContaining Class	CompositeState -> State

platform:/resource/statemachine2/non_coupled/.reconstr

- statemachine
 - Statemachine
 - State
 - CompositeState -> State
 - Transition
 - Region
 - state : State

metamodel after evolution

Property	Value
Changeable	true
Container	false
Containment	true
Default Value	
Default Value Literal	
Derived	false
EContaining Class	Region

mapping

OK Cancel

Zusammenfassung

Problem

- Kosten für Modell-Migration bei Metamodell-Evolution

Lösung: COPE

- **Geschlossene Kopplung:** Wiederverwendung von Migrationswissen durch gekoppelte Operation
- **Offene Kopplung:** Unterstützung beliebiger Migrationen durch manuelle Spezifikation der Evolution auf beiden Ebenen
- **Geschichte:** Vereinigung von offener und geschlossener Kopplung

→ Werkzeug basierend auf Eclipse Modeling Framework

Vielen Dank für die Aufmerksamkeit

- Fragen?
- Kritik?
- Anregungen?

herrmama@in.tum.de