
# Modelloperationen

Vincent Wienecke

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

Das Forschungs- und Entwicklungsprojekt OrViA wird mit Mitteln des Bundesministeriums für Bildung und Forschung (BMBF) gefördert, die innerhalb der zweiten Auswahlrunde der Forschungsoffensive „Softwareengineering 2006“ vergeben wurden, und vom Deutschen Zentrum für Luft- und Raumfahrt (DLR), Projektträger Informationstechnik/ Softwaresysteme betreut.

# Übersicht

- Wozu braucht man Modelloperatoren ?
- Begriff Operator / Operation
- Erklärung der Operatoren
- Klassifikation
- Quellen

# Wozu braucht man Modelloperatoren ?

- Ein Modell muss angepasst werden:
  - z. B.: Einstieg in neue Brance, Veränderung der Produktleistungen, ...
  - oder aus alten Modellen neu erstellt werden
- Lösung 1:
  - Manuelles Anpassen jeden einzelnen Elementes im Modell (auch mit hilfe anderer Modell manuell)
- Lösung 2:
  - Einsatz von Modelloperatoren
  - Änderungs-/Erstellungsvorschrift auf höherer Ebene als das Eintragen der Elemente in ein Modell
  - Aktionen im Modell werden von den Modelloperatoren durchgeführt
  - Nur im Konfliktfall ist Eingreifen nötig

# Operator / Operation

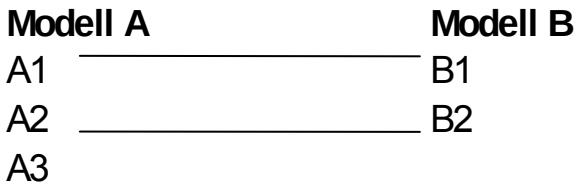
- Operator ist die allgemeine Form der Handlung
  - Beschreibt eine Anweisung, wie auf „etwas“ einzuwirken ist
  - Gleiche Operatoren unterscheiden sich nur in der Notation (Bsp.: Negation)
- Operation ist die konkrete Durchführung eines Operators
  - Für einen Operator kann es verschiedene Realisierungen geben, damit können verschiedene unterschiedliche Operationen für den gleichen Operator stehen

# Beispiel Szenario

- Metamodell map:
  - Modell A:
    - Speicherung im Lokalensystem
  - Modell B:
    - Austausch der Daten mit einem Data-Warehousesystem
  - Einige Elemente beider Modelle stehen miteinander in Relation, (z. B. da sie die ähnliche/gleiche Inhalte beschreiben)

# Domain

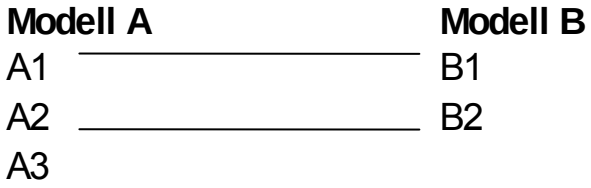
- Domain:
  - Gibt die Menge der Elemente eines Modells (Modell A) zurück, die zu Elementen des anderen Modells (Modell B) in Relation stehen.
  - Bsp – Metamodell map:




s = Domain (map), s = {A1, A2}

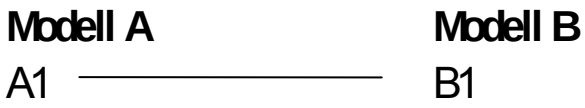
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- RestrictDomain:
  - Gibt ein Metamodell zurück, welches eine Menge übergebener Elemente und deren in Relation stehender Elemente beinhaltet
  - Bsp – Metamodell map:



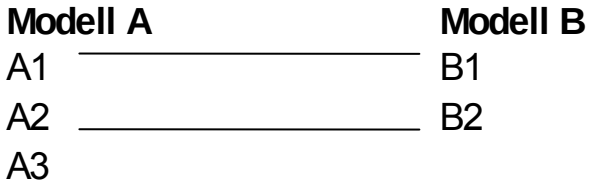

map_2 = RetrictDomain (map, {A1})

map_2 :



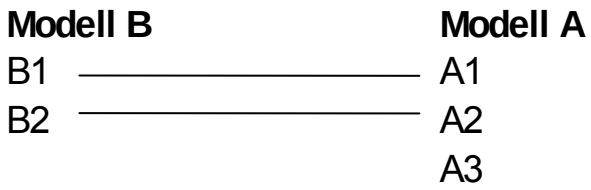
# Invert

- Invert:
  - Gibt ein Metamodell zurück, in dem die Relation des übergebenen Metamodells umgekehrt ist.
  - Bsp – Metamodell map:




map_2 = Invert (map)

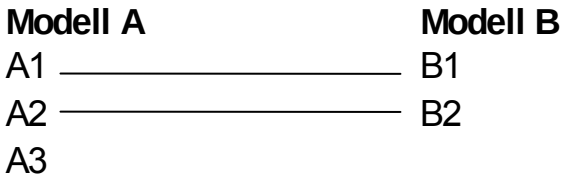
map_2 :



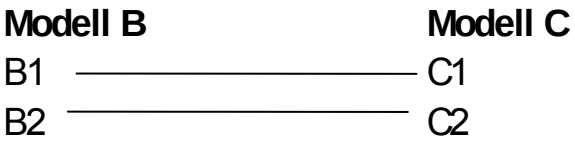

  - map = Invert(Invert(map))

# Compose

- Compose:
  - Kombiniert transitiv die Relation zweier Metamodelle, wobei das Ergebnis auch ein Metamodell ist.
  - Bsp. Metamodell map:

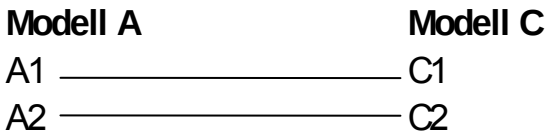



Metamodell map_2



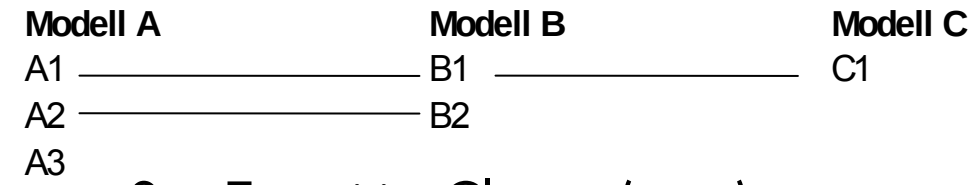

map_3 = Compose(map, map_2)

map_3:

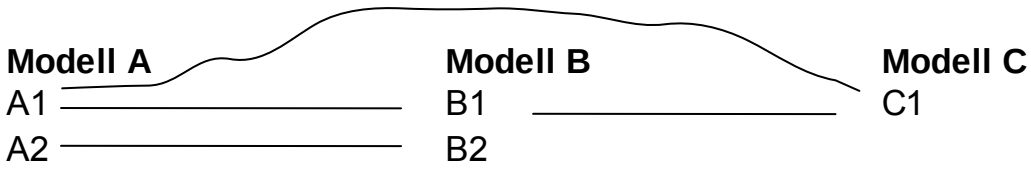



  - Compose(map_1, Compose(map_2, map_3)) = Compose(Compose(map_1, map_2), map_3)

- TransitiveClosure:
  - Gibt alle Relationen und transitiv entstehenden Relationen in einem Metamodell an, Resultat ist ein Metamodell.
  - Bsp. Metamodell map, wobei eine Relation von B1 zu A2 besteht:



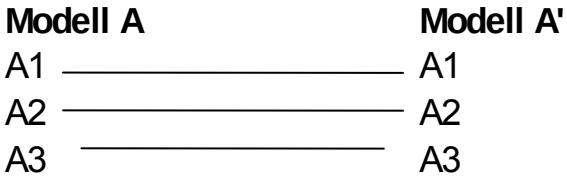

map_2 = TransitiveClosure(map)



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- Id:
  - Erstellt ein Metamodell, in dem eine Identitätsrelation für eine übergebene Menge an Elementen besteht
  - Bsp.:

map = Id({A1, A2, A3})



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- All:
  - Gibt die Menge aller Elemente des übergeben Modells zurück
  - Bsp. Modell A

**Modell A**
A1
A2
A3

s = All(Modell A), s = {A1, A2, A3}

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- Erstellt eine Copy eines Modells mit einer Übergebenen Menge an Knoten (Elemente), welche einer neue ID bekommen.

  Außerdem wird ein Metamodell erstellt, welches das Original und die Kopie mit einer Relation verbindet
  - Bsp. Modell A:

**Modell A**
A1
A2
A3

< m', m_m' > = Copy(Modell A, {A1, A2, A3})

m':

**M'**
A4
A5
A6

m_m':

| **Modell A** | | **M'** |
|---|---|---|
| A1 | ——— | A4 |
| A2 | ——— | A5 |
| A3 | ——— | A6 |

# Mengentheoretische Operatoren

- Union (+), Difference (-) , Intersection (∩)
- können auf Modell und Mengen von Elementen angewendet werden, wo bei das Verhalten so wie bei der Mengentheorie ist.
- x + y : Union(x, y)
- x – y : Difference(x, y)
- x ∩ y : Intersection(x, y)
- Die Mengenoperatoren angewendet auf ein wohlgeformtes Modell garantieren nicht, dass ein wohlgeformtes Modell entsteht

# Zusammengesetzte Operatoren

- Aus der Kombination der einfachen Operatoren lassen sich weitere Operatoren zusammensetzen:
  - Operator Range(map):

    return Domain(Invert(map))
  - Operator RestrictRange(map, {A1}):

    return Invert(RetrictDomain(Invert(map), {A1}))
  - Operator Traverse({A1}, map):

    return Range(RestrictDomain(map, {A1})
  - Operator Retrict(map, Modell A, Modell B)

    return RestrictRange(

    RestrictDomain(map, All(ModellA)), All(Modell B))

# Extract, Delete, Difference

- Extract:
  - Gibt den Teil eines übergebenen Models zurück, welche die übergebene Menge an Elementen des Modells beinhaltet
  - Bei Realisierung transitive Beziehungen beachten
  - Ergebnis ist ein wohlgeformtes Modell, und Metamodell
- Delete
  - „Gegenteil“ zu Extract
  - Gibt den Teil eines übergebenen Models zurück, in dem keines der Elemente der übergebenen Menge vorkommen
  - Realisierung kann sich auf Operator Extract beziehen
- Differenz
  - wie Delete, nur dass statt einer Menge von Elementen, ein Metamodell übergeben wird
  - Operator Diff:

    return Delete(Model A, Domain(Metamodel map))

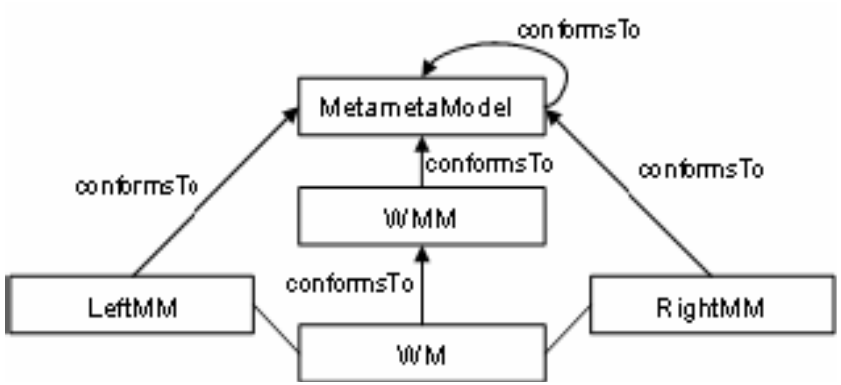
- Aus 2 übergebenen Modellen wird ein Metamodell erstellt
  - Metamodell setzt beide Modelle in eine (Ähnlichkeits)Relation
  - Die Relation der Elemente beider Modell im Metamodell beschreibt ein relative Geleichheit (kann fürs Mergen verwendet werden)

# Merge

- Merge erstellt aus 2 Modellen ein Modell, in dem die Elemente beider Modelle zusammengefügt werden.
- Zum Mergen benötigt man beide Modell und ein Metamodell, welches die Relationen (Ähnlichkeit) der Elemente beider Modelle beinhaltet
- Als Rückgabe bekommt man:
  - das „gemergte“ Modell
  - 2 Metamodelle, welches zu jedem Element der übergebenen Modelle die Relation vom „gemergten“ Modell beinhaltet

# Model weaving

- Fügt mehrere Modelle zu einen weaving Modell zusammen.
- Man übergibt die zu weavende Modelle und ein Metaweavingmodell, welches die Struktur des weaving beinhaltet
  - Über das Metaweavingmodell wird die Struktur der entstehenden Weaving Modells bestimmt
- Zurück kommt das fertige weaving Modell



# Klassifikation der Operatoren

- Einfache Operatoren
  - Domain, Id, All, Compose, Mengentheor. Op., ....
  - führen nur „einfache“ Umformungen oder Aufgaben durch
- Zusammengesetzte Operatoren
  - Range, Traverse, ...
  - nutzt einfache Operatoren um komplexere Funktionen zu erfüllen
- Komplexe Operatoren
  - Extract, Match, Merge, Weaving

- Prototyp Rondo:
  - S. Melnik
  - Alle Operatoren außer Model Weaving
- ATLAS Model Weaver:
  - Model Weaving

# Quellen

- Quellen:
  - Operator / Operation: http://web.uni-bamberg.de/ppp/insttheopsy/glossar/Operator.html
  - Sergey Melnik - Generic Model Management: http://research.microsoft.com/~melnik/
  - Model Weaving http://www.sciences.univ-nantes.fr/lina/atl/www/papers/http://www.sciences.univ-nantes.fr/lina/atl/www/papers/IDM_2005_weaver.pdf