

# 4 Das UML-Metamodell

---

## 4.1 Einleitung

- 4-Schichten Modell, Object Constraint Language (OCL)

## 4.2 Das Modell

- exemplarische Auszüge aus dem Modell
- Wiederholung der UML-Konstrukte

## 4.1 Einleitung

---

- Zur Beschreibung der Semantik von UML wird ein Metamodell benutzt
- Dieses ist Bestandteil einer allgemeinen 4-Schichten Architektur
  - Meta-Metamodell
  - Metamodell
  - Modell
  - Benutzer Objekte

# 4-Schichten Architektur

---

<i>Schicht</i>	<i>Beschreibung</i>	<i>Beispiel</i>
Meta-Metamodell	Infrastruktur für eine Architektur zur Metamodellierung. Definiert die Sprache zur Spezifikation des Metamodells	Meta-Klasse, Meta-Attribut, Meta-Operation, Meta-Beziehung
Metamodell	Instanz eines Meta-Metamodells. Definiert die Sprache zur Definition eines Modells	Klasse, Attribut, Operation, Assoziation
Model	Instanz eines Metamodells. Spezifiziert Konzepte zur Beschreibung eines Anwendungsbereichs	Teilnehmer, Tagung, Datum, Vortrag
Benutzer-Objekte	Instanz eines Modells. Definiert einen speziellen Fall des Anwendungsbereichs	Ernst Meier, UML2000, 28.06.00

---

# UML-Metamodell

---

- Vorteile eines Metamodells:
  - exakte Definition der Semantik möglich
  - Modell leicht erweiterbar
  - Anpassung an andere Standards, hier z.B. die MOF (Meta Object Facility) der OMG als Meta-Metamodell
- Beschreibung der UML
  - *Syntax* mittels UML-Diagrammen (meta-zirkulär)
  - *Semantik* mittels OCL und natürlicher Sprache

# OCL (Object Constraint Language)

---

- **Formale Sprache zur Beschreibung von Einschränkungen**
- **Motivation:**
  - umgangssprachliche Beschreibung von Einschränkungen oft ungenau
  - traditionelle formale Sprachen oft zu kompliziert
- **Entwurf von OCL als Kompromiss**
  - Entwicklung bei IBM als Geschäftsmodellensprache

# OCL (Forts.)

---

- **Eigenschaften:**
  - reine Ausdruckssprache
  - kann das Modell nicht modifizieren
  - keine Programmiersprache (kein Kontrollfluss)
  - typisiert, d.h. jeder OCL-Ausdruck hat einen Typ

# OCL (Forts.)

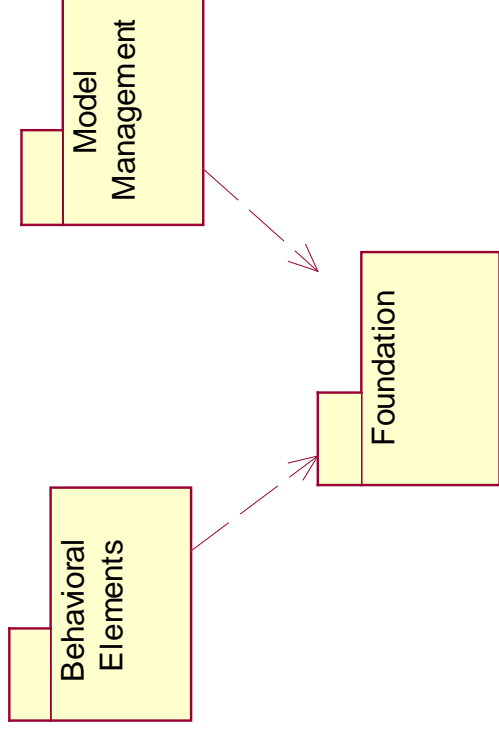
---

- **Einsatz in UML:** Spezifikation von
  - Invarianten auf Klassen und Typen
  - Vor- und Nachbedingungen für Operationen
  - Einschränkungen auf Operationen
  - Zusatzbedingungen (in [ ])

## 4.2 Das Modell

---

- UML-Metamodell ist in Pakete aufgeteilt
- Auf oberster Ebene gibt es die Pakete
  - Foundation
  - Behavioral Elements
  - Model Management





## 4.2.1 Foundation

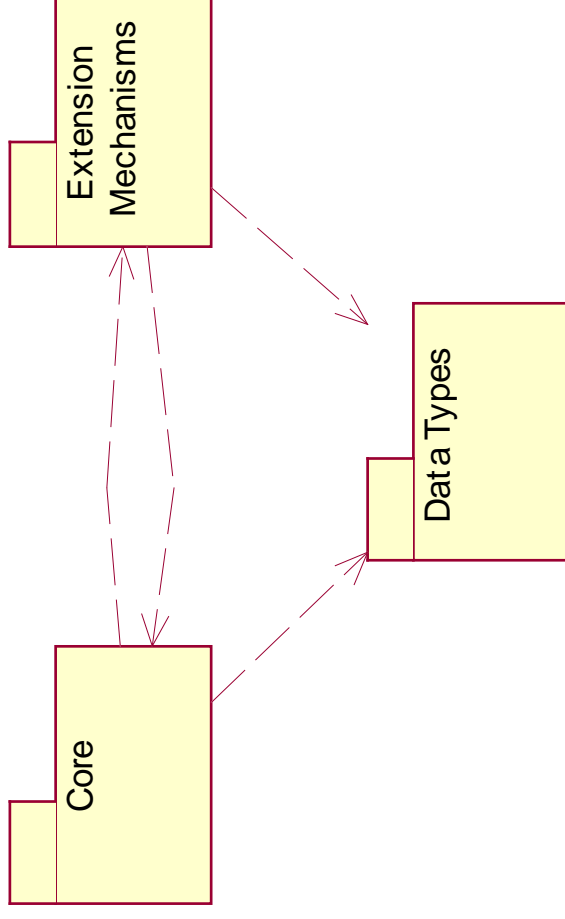
---

- Unterteilung in drei Pakete
  - Paket **Core**  
grundlegende Konzepte (Elemente, Assoziation, Generalisierung, Abhängigkeit)
  - Paket **Extension Mechanisms**  
Benutzung und Erweiterung von Elementen
  - Paket **Data Types**  
grundlegende primitive Datenstrukturen

# Foundation (Forts.)

---

## Unterpakete im Foundation Paket:

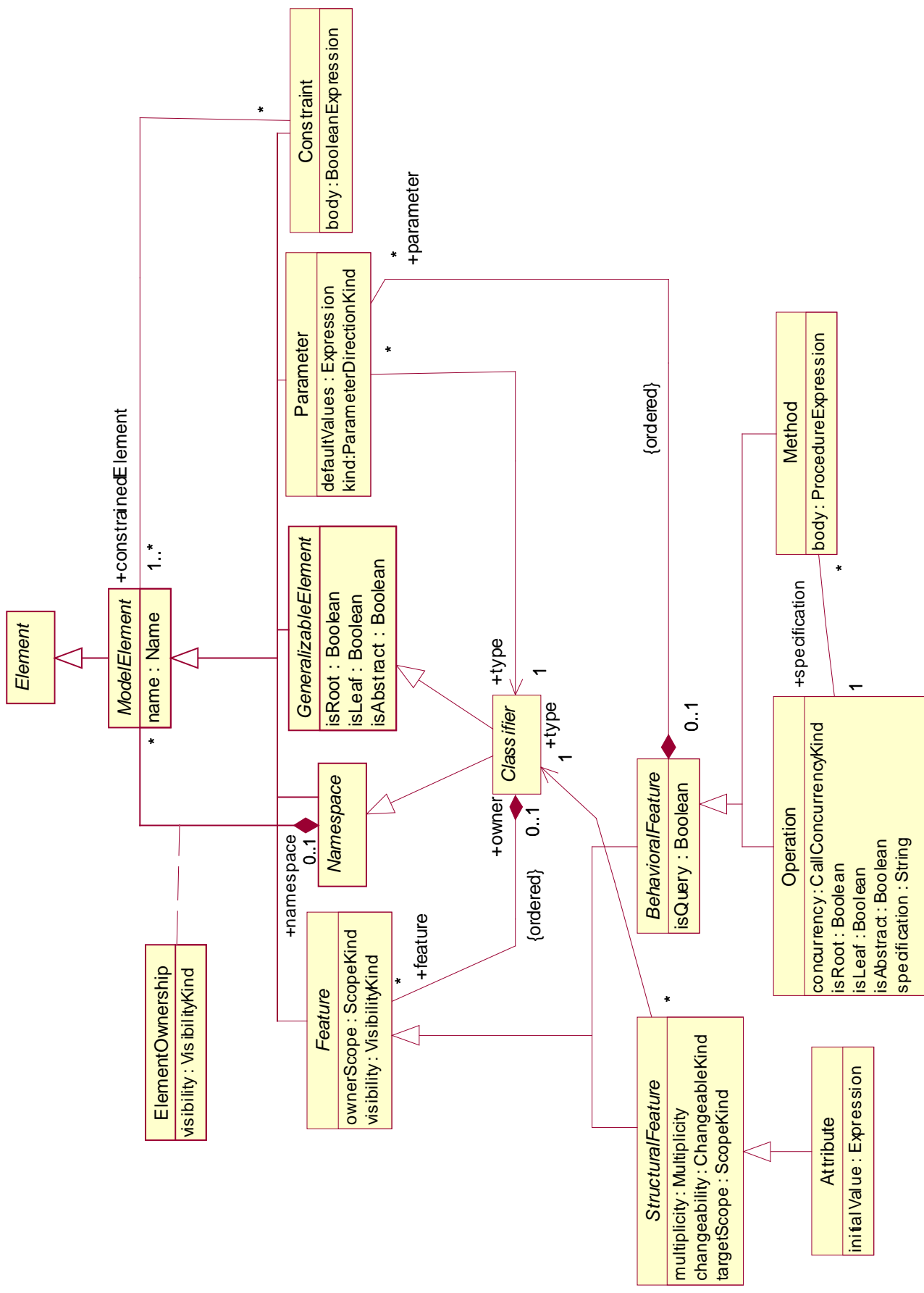


# Foundation - Core

---

- Klassendiagramm *Core - Backbone*:
  - Basisklassen **Element**, **ModelElement**
  - Elemente werden in Namensräumen (**Namespace**) zusammengefasst
  - abstrakte Klasse **Classifier** repräsentiert alle klassenähnlichen Elemente
  - Bestandteile eines Classifiers sind durch Klasse **Feature** abstrahiert

# Klassendiagramm Core - Backbone

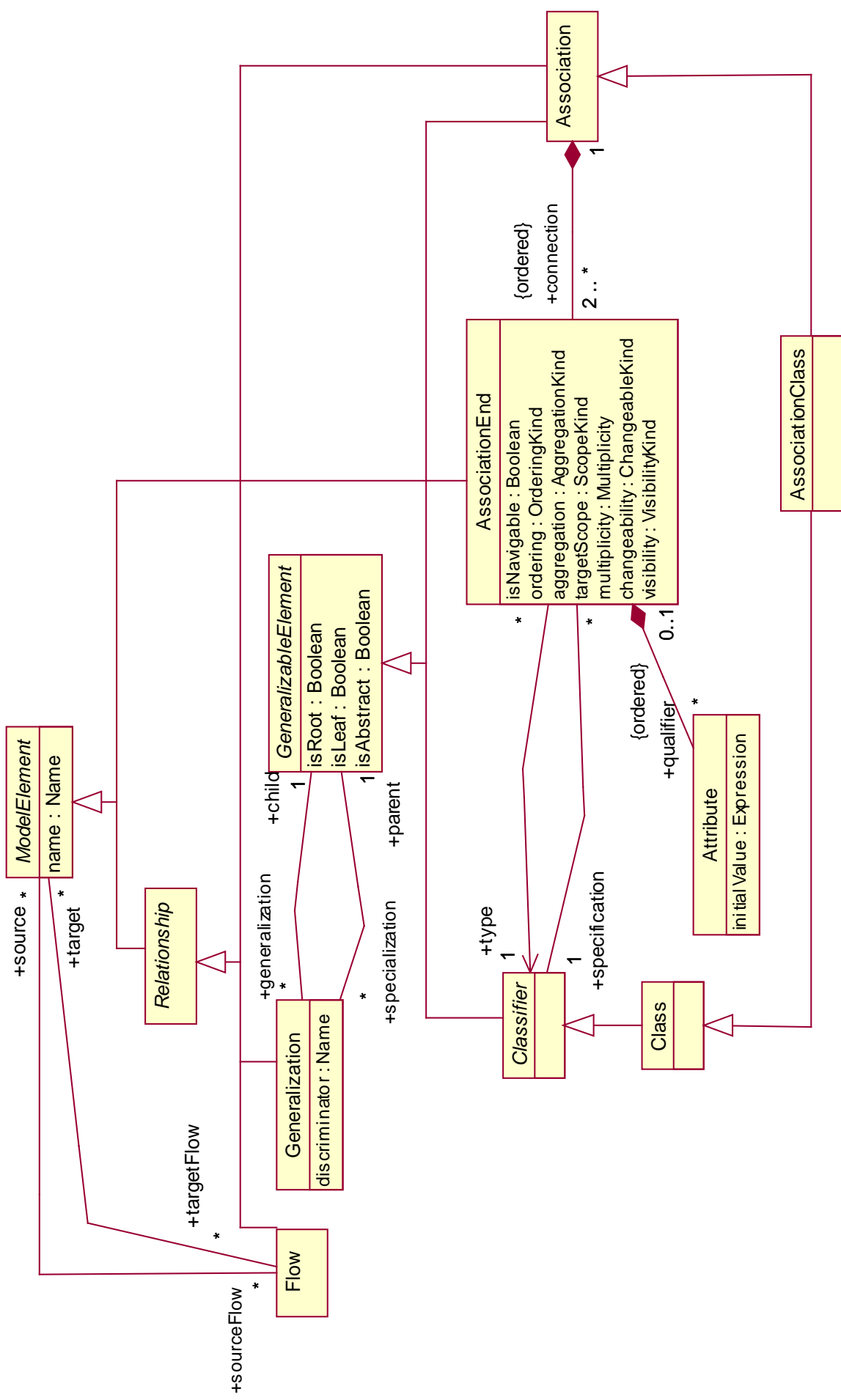


# Foundation - Core (Forts.)

---

- Klassendiagramm *Core - Relationships*:
  - abstrakte Klasse *Relation* ist Oberklasse von *Generalization*, *Association*, *Flow*
  - Assoziationsende als Klasse modelliert
  - Assoziation mit Qualifier, Assoziationsklasse

# Klassendiagramm Core - Relationships



# Foundation - Core (Forts.)

---

- Semantik für **AssociationEnd**:
  - Ist das eine Ende ein Interface oder ein DataType, dann ist das andere Ende nicht navigierbar
  - In OCL:

```
(self.type.ocIsKindOf (Interface) or  
self.type.ocIsKindOf (DataType)) implies  
self.association.connection->select  
(ae | ae <> self)->forall(ae | ae.isNavigable  
= #false)
```

# Foundation - Core (Forts.)

---

– Ein Element darf höchstens zu einer Komposition gehören

– In OCL:

```
self.aggregation = #composite implies  
self.multiplicity.max <= 1
```

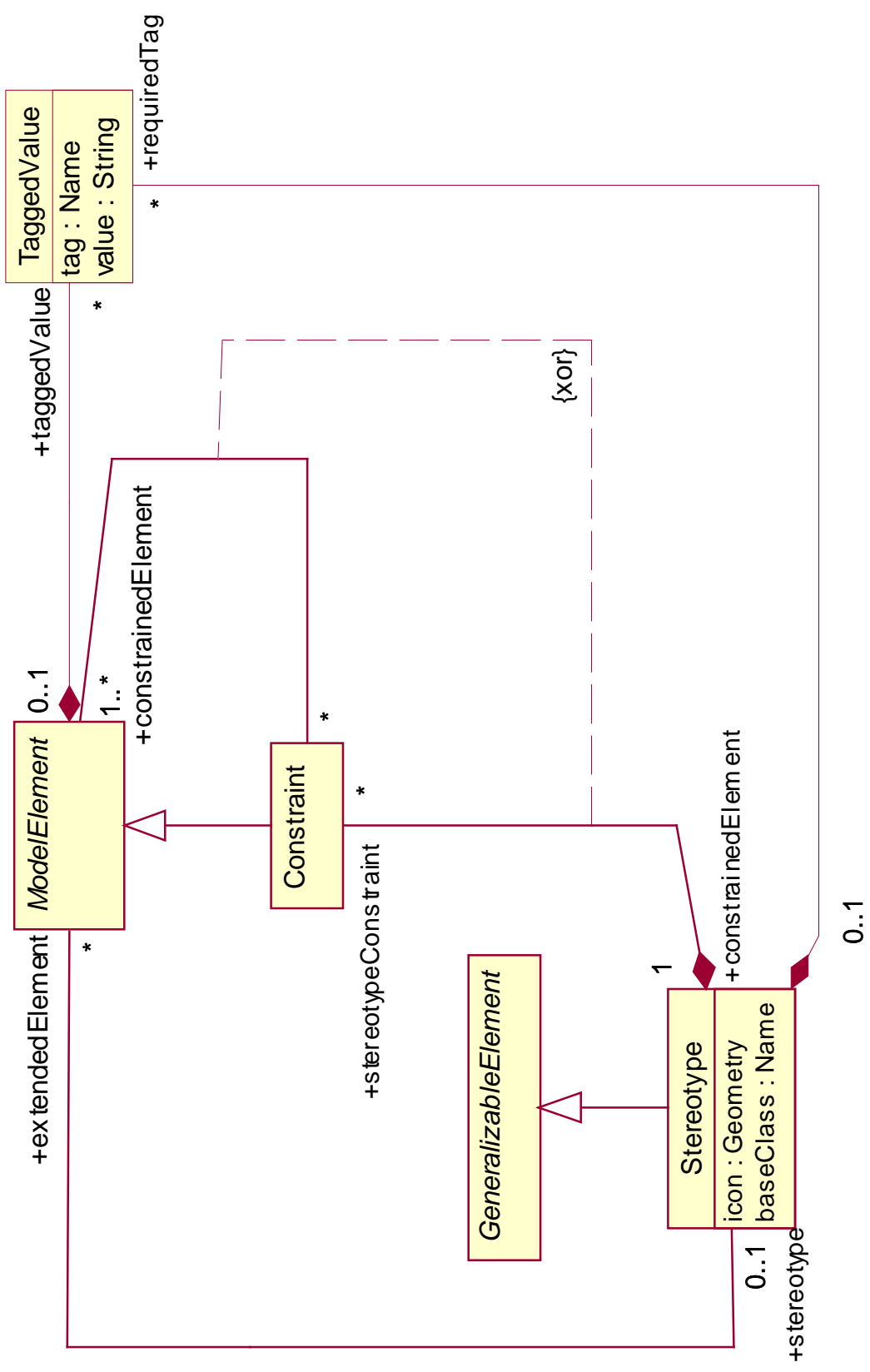


# Foundation - Extension Mechanisms

---

- Klassendiagramm *Extension Mechanisms*:
  - **Stereotyp**
    - enthält Einschränkungen und Eigenschaftswerte
    - ist generalisierbar
  - **Einschränkung**
    - gehört entweder zu einem stereotypisierten Element oder mind. einem ModelElement
  - **Eigenschaftswert**
    - gehört entweder zu ModelElement oder Stereotyp

# Klassendiagramm Core - Extension Mechanisms

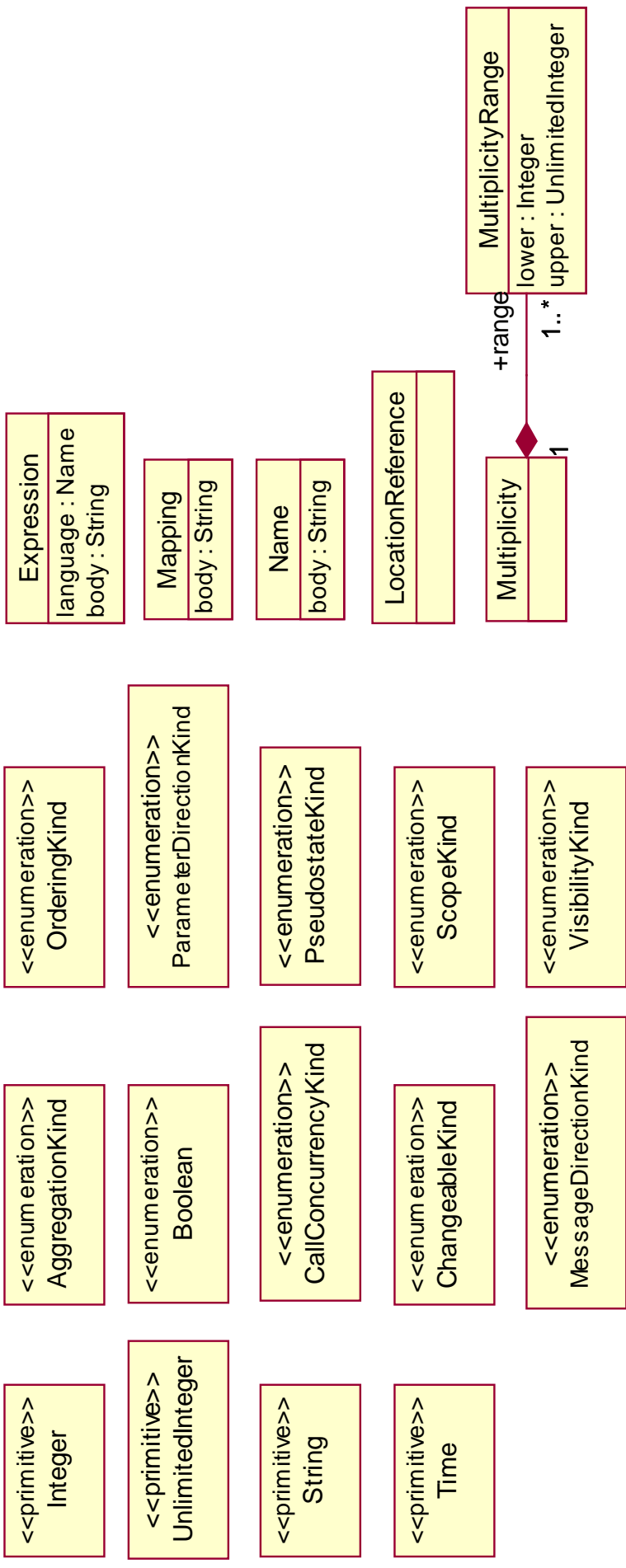


# Foundation - Data Types

---

- Klassendiagramm *Data Types - Main*:
  - enthält alle einfachen Daten- und Aufzählungstypen, die zur Definition von UML benutzt werden

# Klassendiagramm Data Types - Main



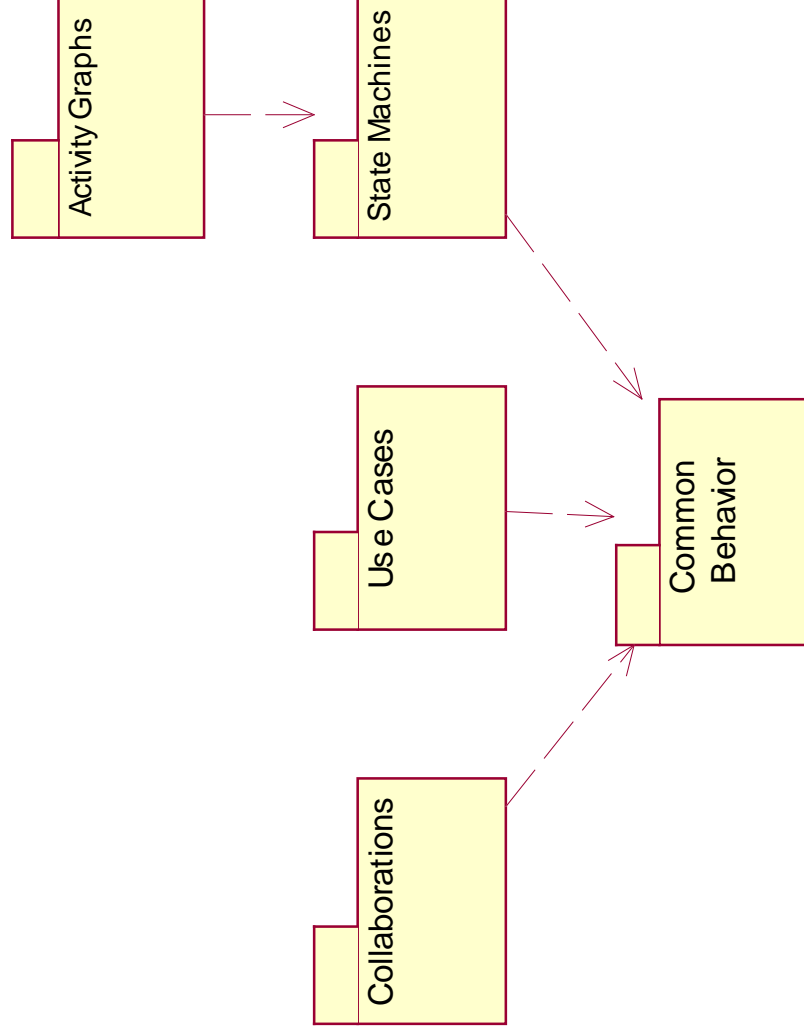
## 4.2.2 Behavioral Elements

---

- Unterteilung in fünf Pakete
  - Paket **Common Behavior**  
grundlegende Konzepte für die anderen Pakete
  - Paket **Collaborations**
  - Paket **Use Cases**
  - Paket **State Machines**
  - Paket **Activity Graphs**

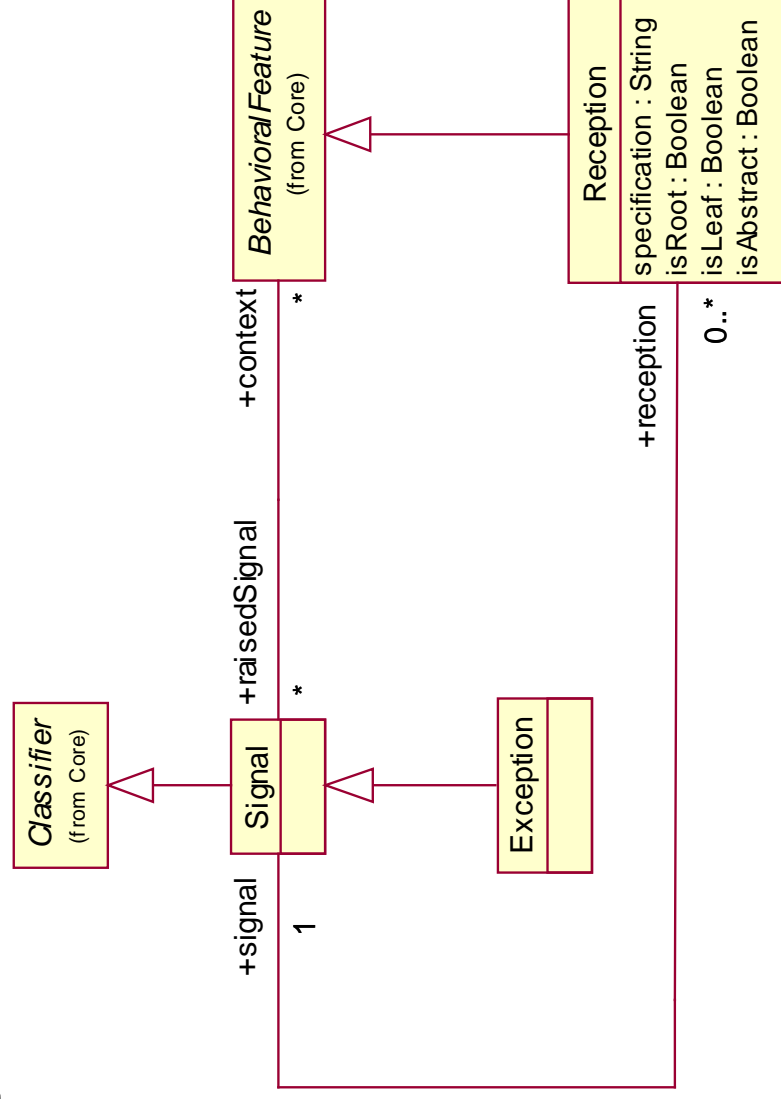
# Behavioral Elements - Unterpakete

---



# Behavioral Elements - Common Behavior

## Klassendiagramm *Common Behavior - Signals*:



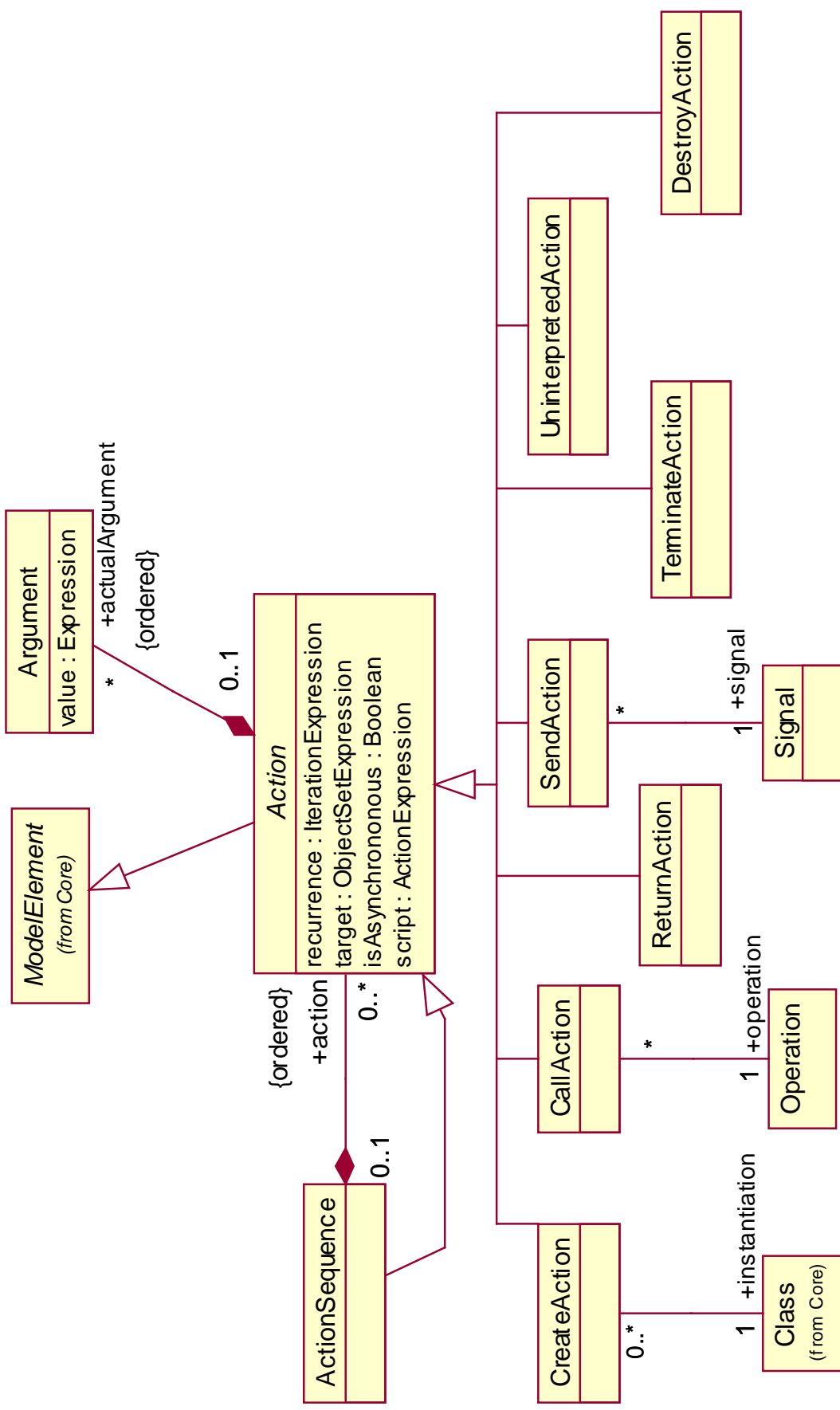
# Behavioral Elements - Common Behavior

---

- Klassendiagramm *Common Behavior - Actions:*
  - zeigt die verschiedenen Aktionsarten
  - Zusammenhang zwischen Call und Operation, Send und Signal



# Klassendiagramm Common Behavior - Actions

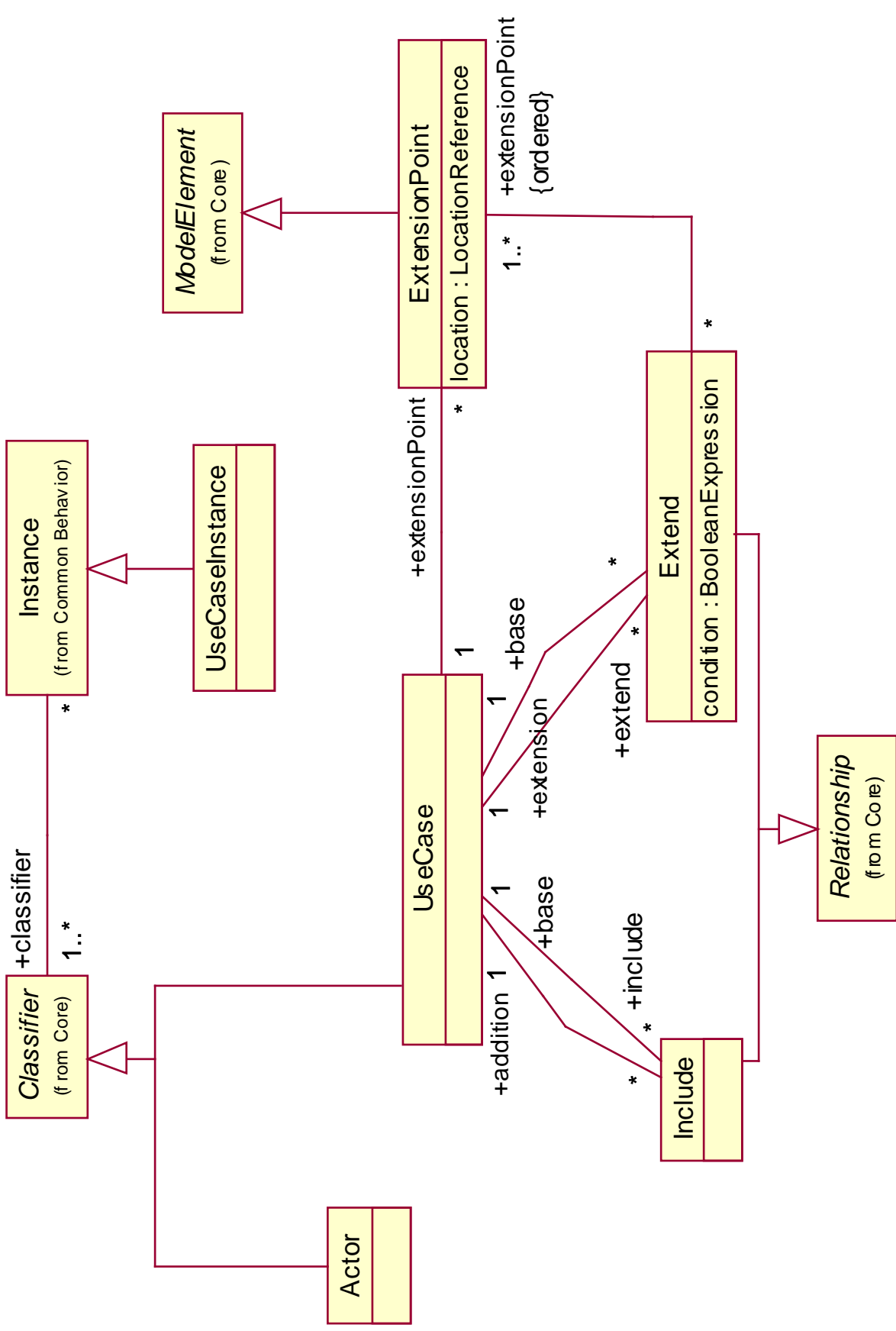


# Behavioral Elements - Use Cases

---

- Klassendiagramm *Use Cases*:
  - UseCase und Actor sind Unterklassen von Classifier
  - Include und Extend Beziehungen

# Klassendiagramm Use Cases

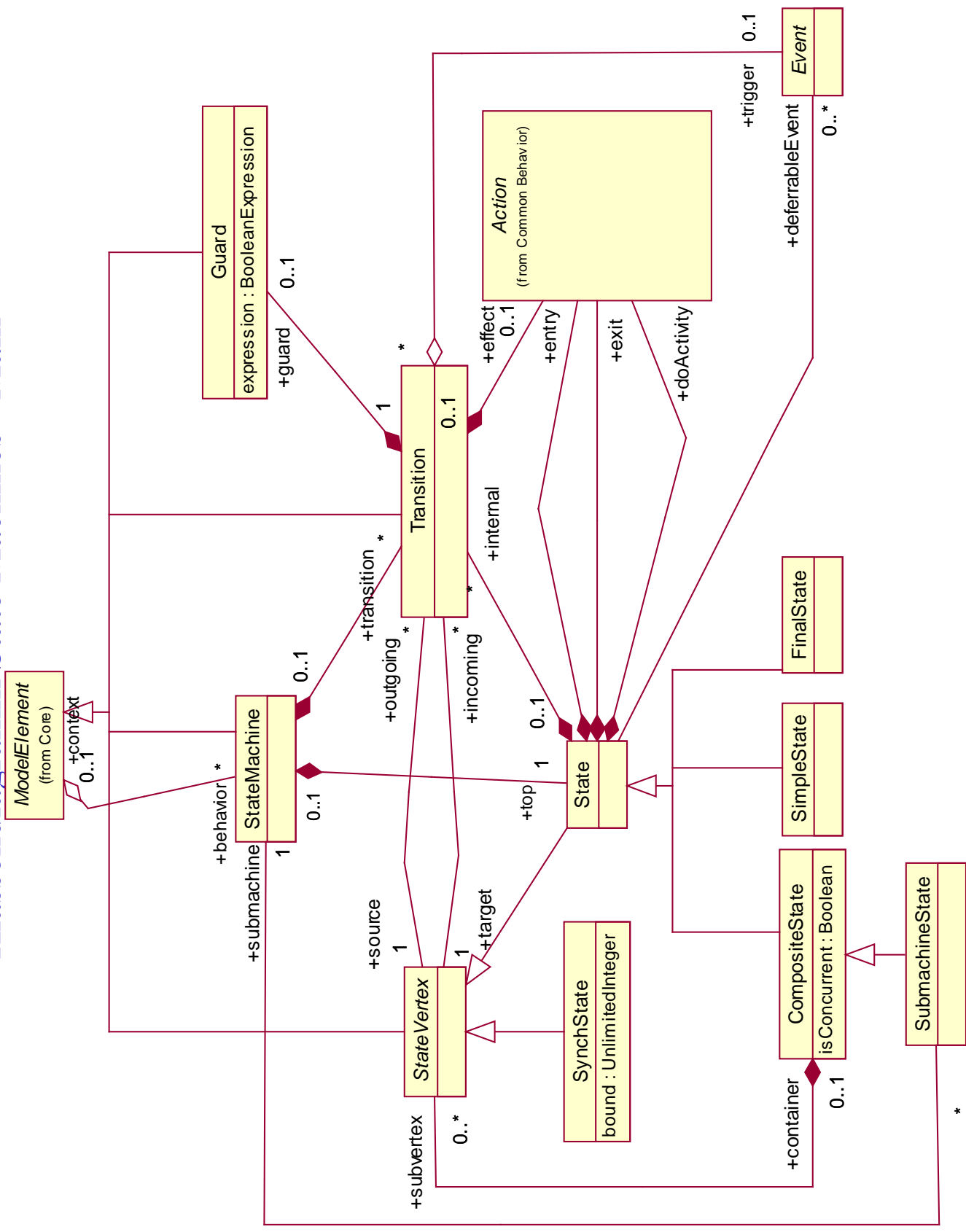


# Behavioral Elements - State Machines

---

- Klassendiagramm *State Machines - Main:*
  - Bestandteile einer Zustandsmaschine
  - Zustandsarten
  - Komponenten eines Zustandes
  - Komponenten einer Transition

# Klassendiagramm State Machines - Main

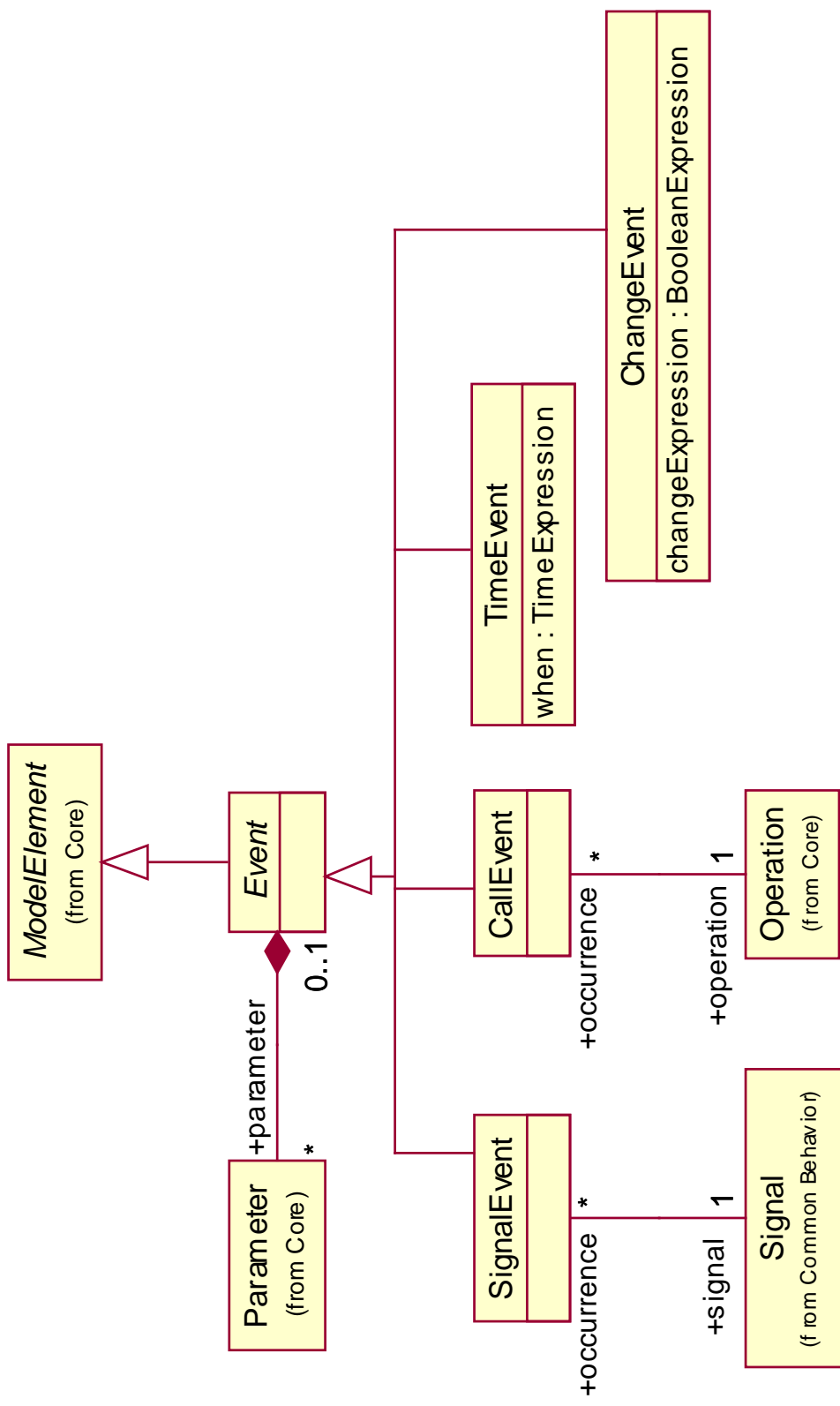


# Behavioral Elements - State Machines

---

- Klassendiagramm *State Machines - Events*:
  - Event Arten
  - Zusammenhänge zu `Signal` und `Operation`

# Klassendiagramm State Machines - Events



## 4.2.3 Model Management

---

- Klassendiagramm *Model Management - Main:*
  - Zuordnung von ModelElements
  - Importieren



# Klassendiagramm Model Management - Main

