

# OMSI-1

## Praktikum

### UML

# Heute

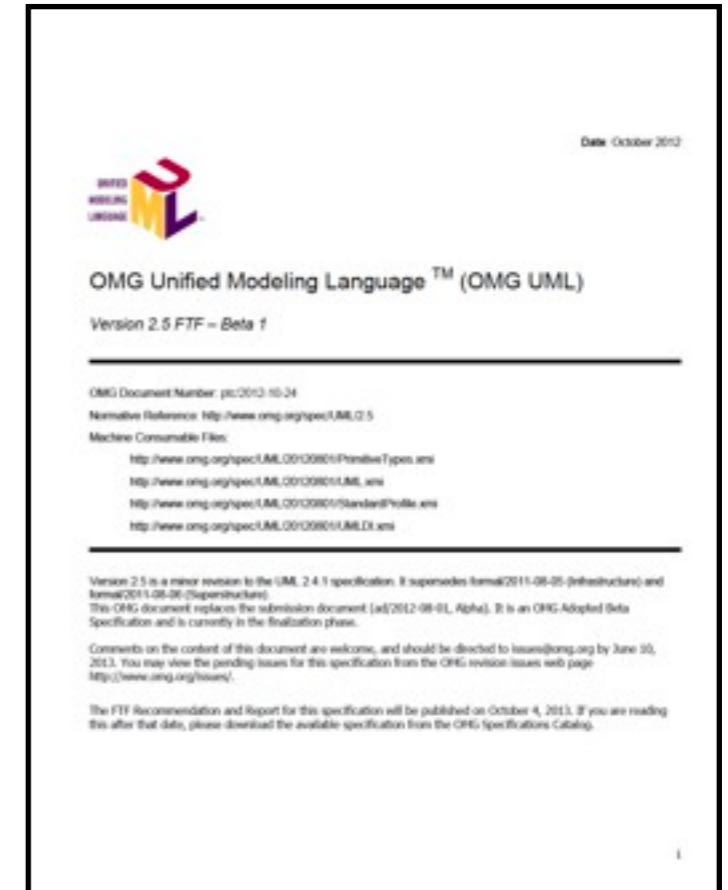
- Literatur zu UML
- Übungsteil
  - Klassen- & Objektdiagramm
  - Zustandsautomaten
- Werkzeug: MagicDraw UML
- Aufgaben zu UML

# Literatur zu UML


Der UML-Sprachstandard als Primärquelle  
(Superstructure)

<http://www.omg.org/spec/index.htm>

<http://www.omg.org/spec/UML/2.5/Beta1/>



# Literatur zu UML



Date: October 2012

## OMG Unified Modeling Language™ (OMG UML)

Version 2.5 FTF – Beta 1

---

OMG Document Number: ptc/2012-10-24  
Normative Reference: <http://www.omg.org/spec/UML/2.5>  
Machine Consumable Files:  
<http://www.omg.org/spec/UML/20120801/PrimitiveTypes.xml>  
<http://www.omg.org/spec/UML/20120801/UML.xml>  
<http://www.omg.org/spec/UML/20120801/StandardProfile.xml>  
<http://www.omg.org/spec/UML/20120801/UMLDI.xml>

---

Version 2.5 is a minor revision to the UML 2.4.1 specification. It supersedes formal/2011-08-05 (Infrastructure) and formal/2011-08-06 (Superstructure). This OMG document replaces the submission document (ad/2012-08-01, Alpha). It is an OMG Adopted Beta Specification and is currently in the finalization phase.

Comments on the content of this document are welcome, and should be directed to [issues@omg.org](mailto:issues@omg.org) by June 10, 2013. You may view the pending issues for this specification from the OMG revision issues web page <http://www.omg.org/issues/>.

The FTF Recommendation and Report for this specification will be published on October 4, 2013. If you are reading this after that date, please download the available specification from the OMG Specifications Catalog.

i

10	Simple Classifiers .....	172
10.1	Summary .....	172
10.2	DataTypes .....	172
10.3	Signals.....	174
10.4	Interfaces.....	176
10.5	Classifier Descriptions .....	180
10.6	Association Descriptions .....	185
11	Structured Classifiers.....	190
11.1	Summary .....	190
11.2	Structured Classifiers .....	190
11.3	Encapsulated Classifiers .....	197
11.4	Classes.....	202
11.5	Associations .....	208
11.6	Components .....	217
11.7	Collaborations.....	224
11.8	Classifier Descriptions .....	229
11.9	Association Descriptions .....	244
12	Packages.....	254
12.1	Summary .....	254
12.2	Packages.....	254
12.3	Profiles.....	266
12.4	Classifier Descriptions .....	282
12.5	Association Descriptions .....	291
13	Common Behavior.....	295
13.1	Summary .....	295
13.2	Behaviors .....	295
13.3	Events.....	300
13.4	Classifier Descriptions .....	305
13.5	Association Descriptions .....	312
14	StateMachines.....	316
14.1	Summary .....	316
14.2	Behavior StateMachines .....	316
14.3	StateMachine Redefinition.....	347
14.4	ProtocolStateMachines .....	352
14.5	Classifier Descriptions .....	358
14.6	Association Descriptions .....	379
15	Activities.....	388
15.1	Summary .....	388
15.2	Activities.....	388
15.3	Control Nodes .....	403
15.4	Object Nodes.....	413
15.5	Executable Nodes.....	420
15.6	Activity Groups.....	423
15.7	Classifier Descriptions .....	431

# UML-Sprachstandard: Classes

- **Summary**
- Abstract Syntax
- Semantics
- Notation
- Examples

## **11.4 Classes**

### **11.4.1 Summary**

Class is the concrete realization of EncapsulatedClassifier and BehavioredClassifier. The purpose of a Class is to specify a classification of objects and to specify the Features that characterize the structure and behavior of those objects.

# UML-Sprachstandard: Classes

- Summary
- Abstract Syntax
- Semantics
- **Notation**
- Examples

## 11.4.4 Notation

A Class is shown using the Classifier symbol. As Class is the most widely used Classifier, the keyword “Class” need not be shown in guillemets above the name.

A Class has four mandatory compartments: attributes, operations, receptions (see [9.2.4](#)) and internal structure (see 11.2.4). A Class may also have optional compartments as described for Classifiers in general (see [9.2.4](#)).



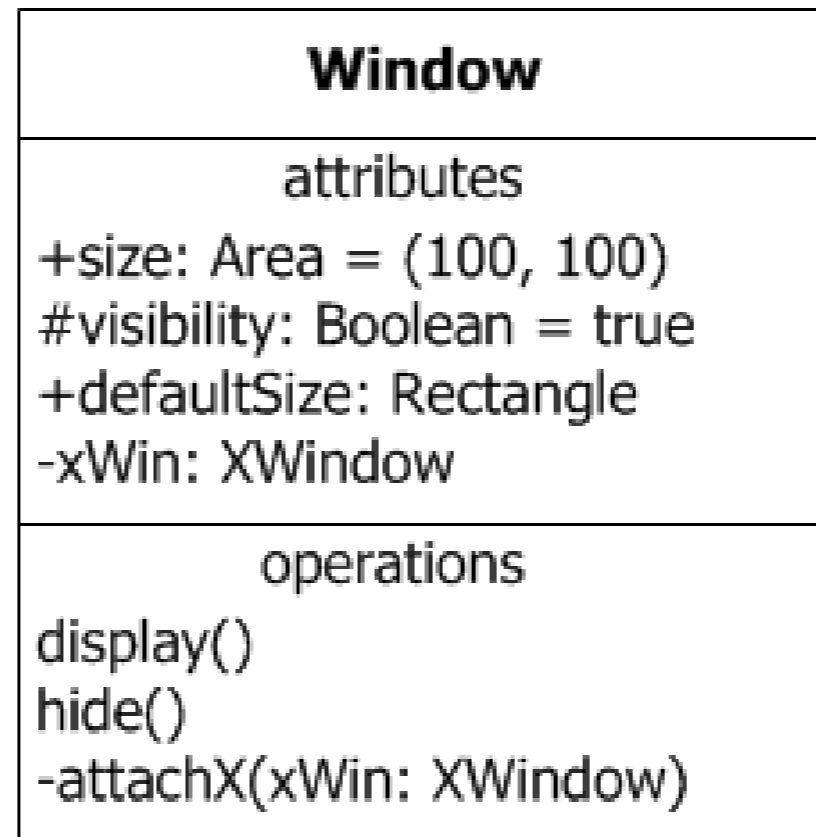
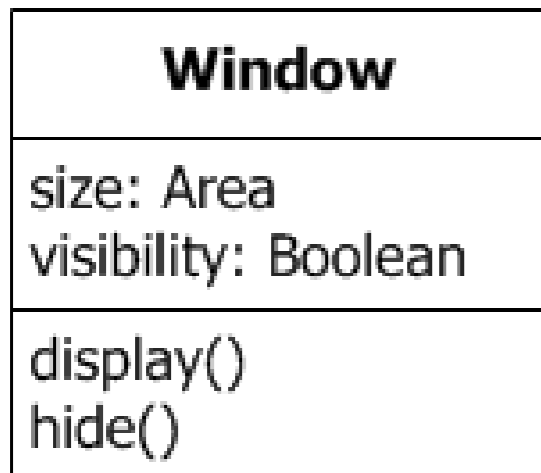
# UML-Sprachstandard: Classes

- Summary
- Abstract Syntax
- Semantics
- Notation
- **Examples**

## 11.4.5 Examples

Figure 11.16 shows three ways of displaying the Class Window, according to the options set out for Classifier notation in [9.2.4](#). The top left symbol shows all compartments suppressed. The lower left symbol shows the attributes and operations compartments, each listing the features but suppressing details such as default values, parameters, and visibility markings. The right symbol shows these details, as well as the optional compartment headers.

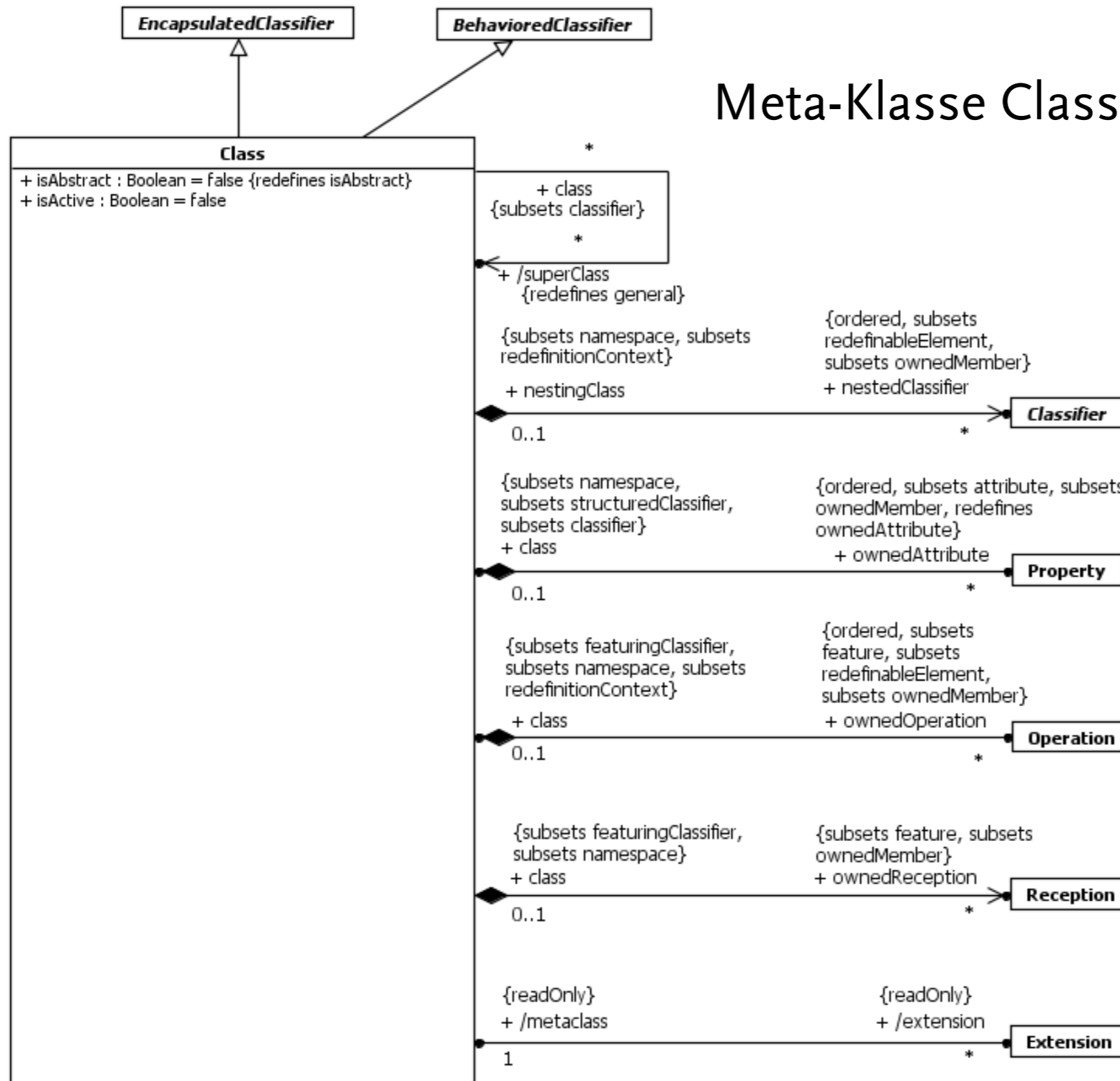
**NOTE.** The `display()` and `hide()` operations have no visibility specified.



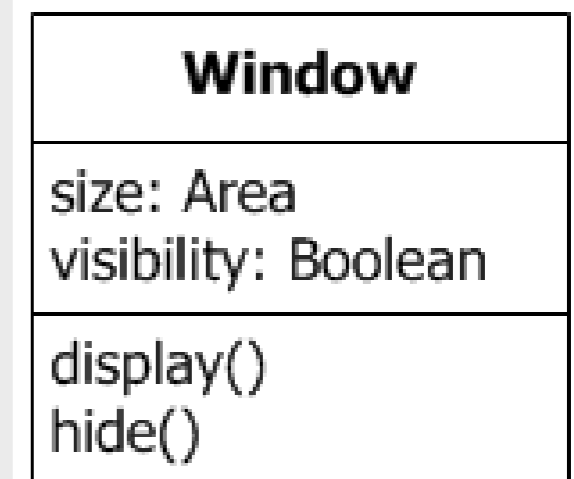
# UML-Sprachstandard: Classes

- Summary
- **Abstract Syntax**
- Semantics
- Notation
- Examples

## Meta-Klasse Class



## Instanz/Objekt von Meta-Klasse Class



## Instanz/Objekt von Klasse Window

w1 : Window



# UML-Sprachstandard: Classes

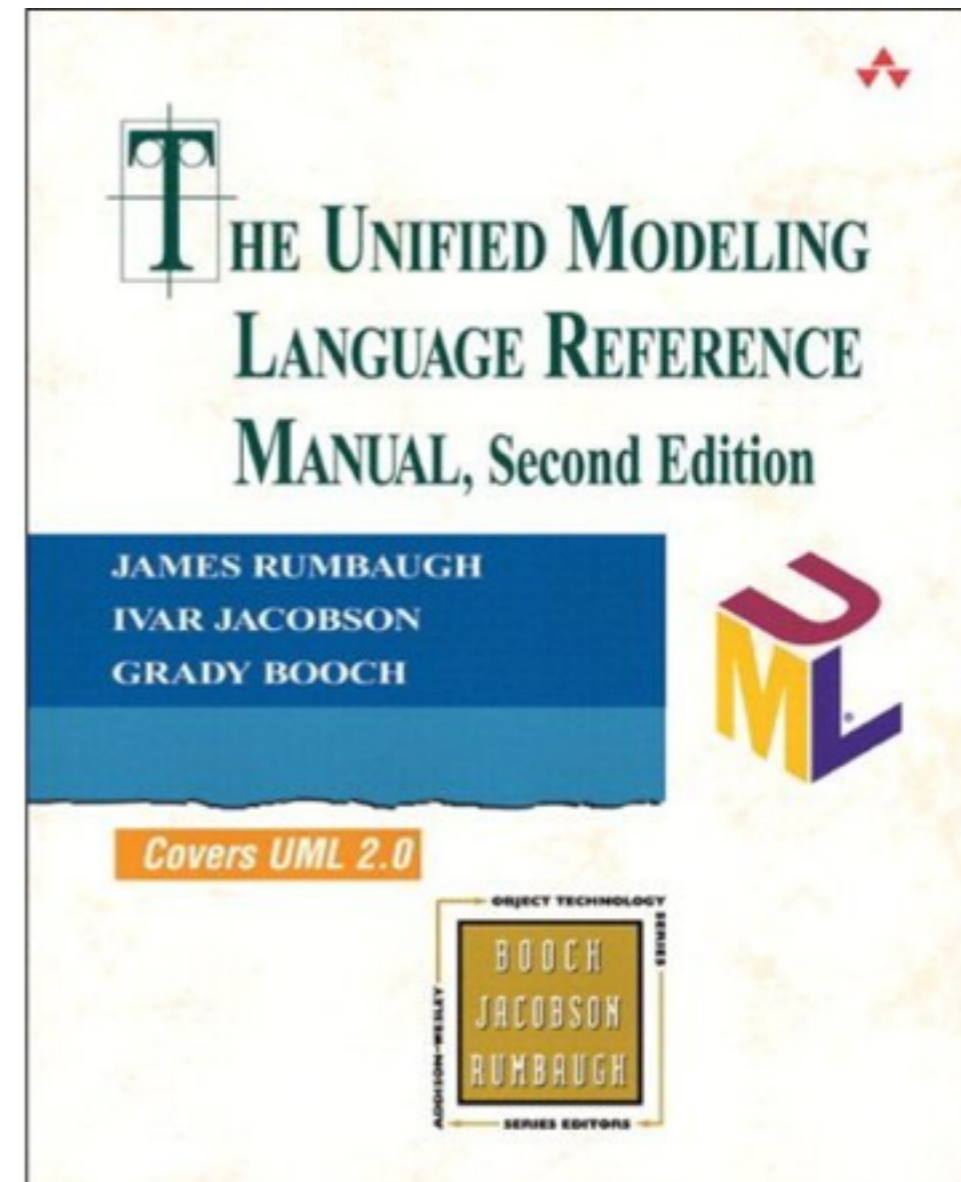
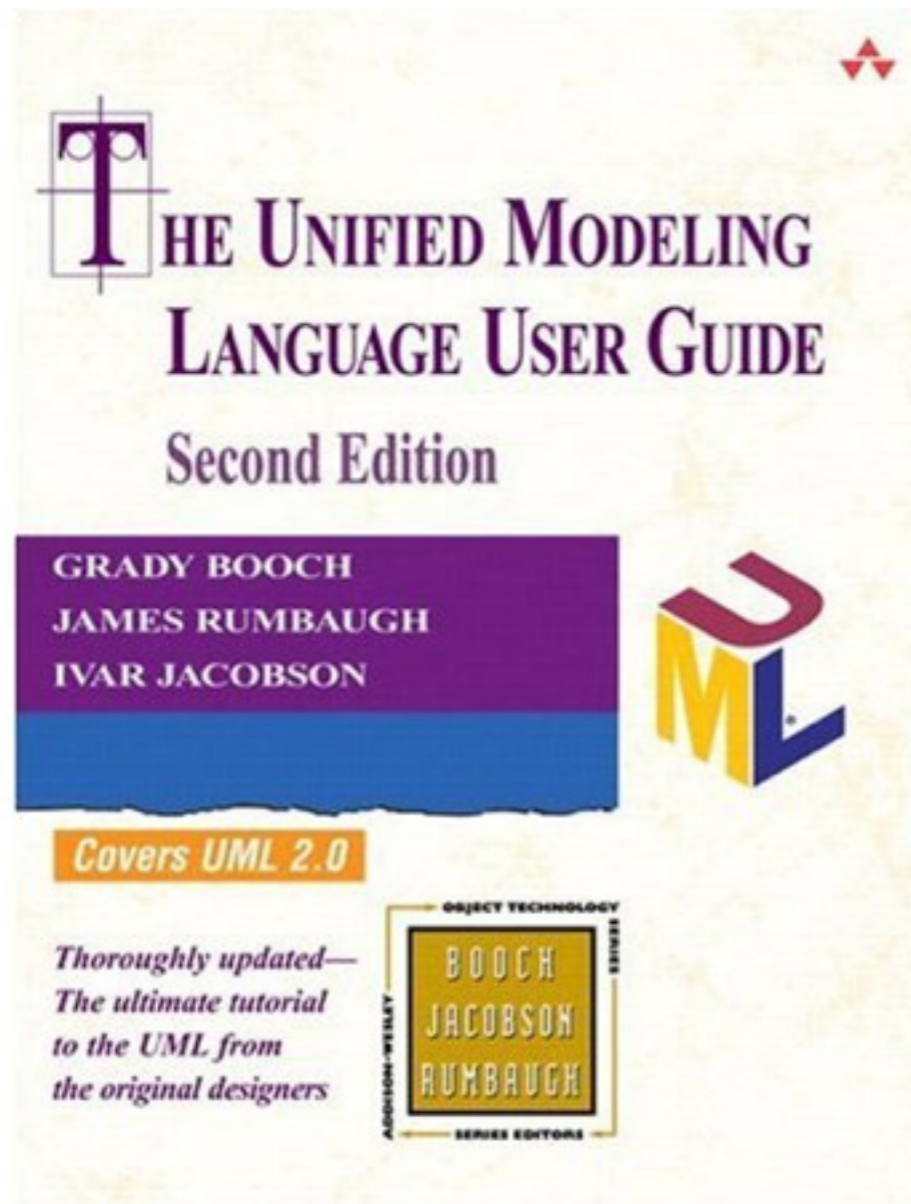
- Summary
- Abstract Syntax
- **Semantics**
- Notation
- Examples

## 11.4.3 Semantics

A Class may be designated by setting `isActive` to true as active (i.e., each of its instances is an active object). When `isActive` is false the Class is passive (i.e., each of its instances executes within the context of some other object).

An active object is an object that, as a direct consequence of its creation, commences to execute its `classifierBehavior`, and does not cease until either the complete Behavior is executed or the object is terminated by some external object. (This is sometimes referred to as “the object having its own thread of control.”) The points at which an active object responds to communications from other objects is determined solely by the Behavior of the active object and not by the invoking object. If the `classifierBehavior` of an active object completes, the object is terminated.

# Literatur zu UML



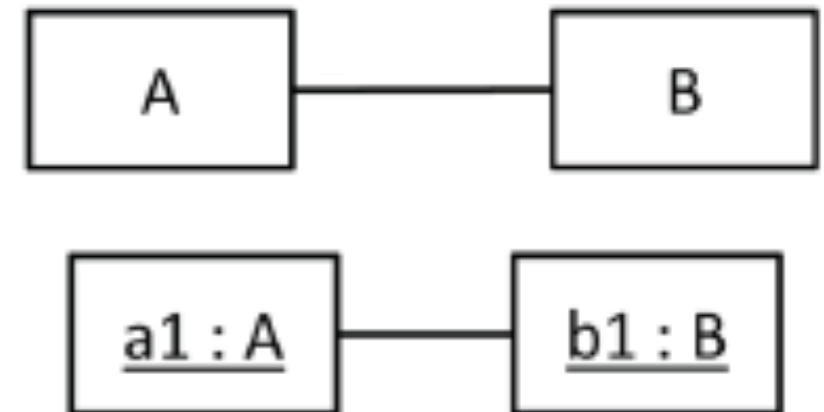
Bücher von den Erschaffern der UML

# Übungsteil

# Quiz: Klassendiagramme

- Was ist die Bedeutung eines Klassendiagramm?
  - Ein Klassendiagramm beschreibt eine Menge von möglichen Objekten mit ihren möglichen Attributbelegungen und möglichen Beziehungen  
... sowie ihren Verhaltensmerkmalen in Form von aufrufbaren Operationen und empfangbaren Signalen

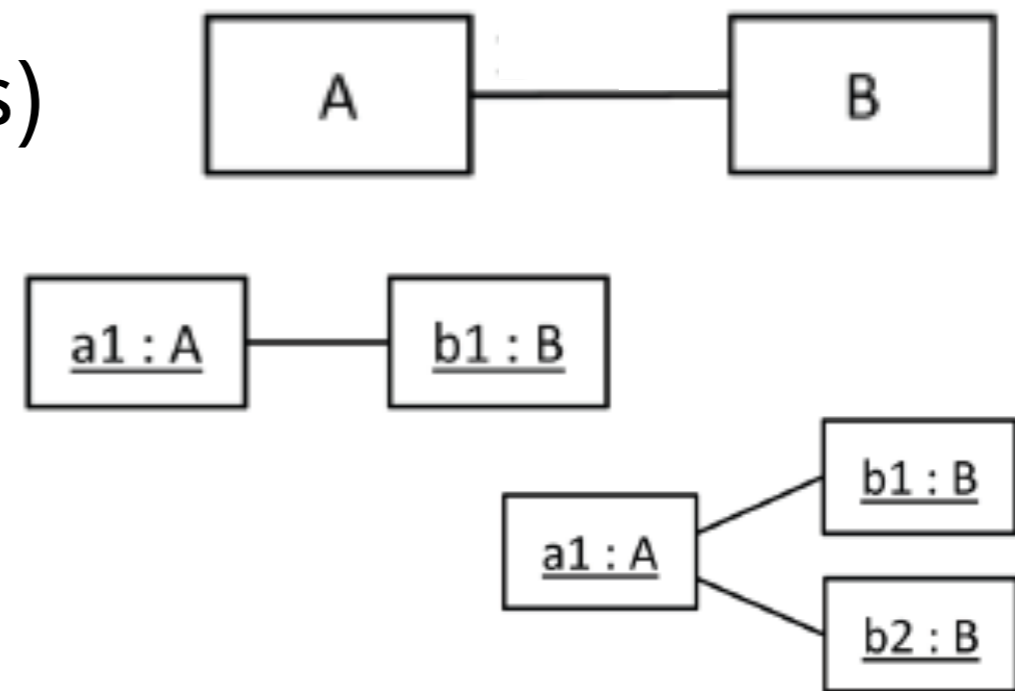
- Welche Bedeutung haben Klassen- und Objektdiagramm für ein modelliertes System?



- Klassendiagramme: mögliche Systemzustände über die Zeit
- Objektdiagramme: ein Zustand zu einem Zeitpunkt, z.B. der Anfangszustand

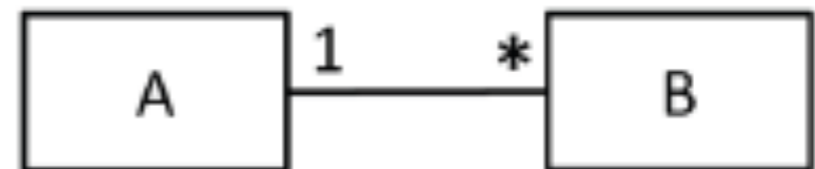
# Quiz: Assoziation

- Was ist die Bedeutung einer Assoziation?  
(z.B. zwischen den Klassen A und B)
  - Eine Assoziation beschreibt eine Menge von Verbindungen (oder Links) zwischen Objekten von Klassen.
  - Wieviele kann es geben?



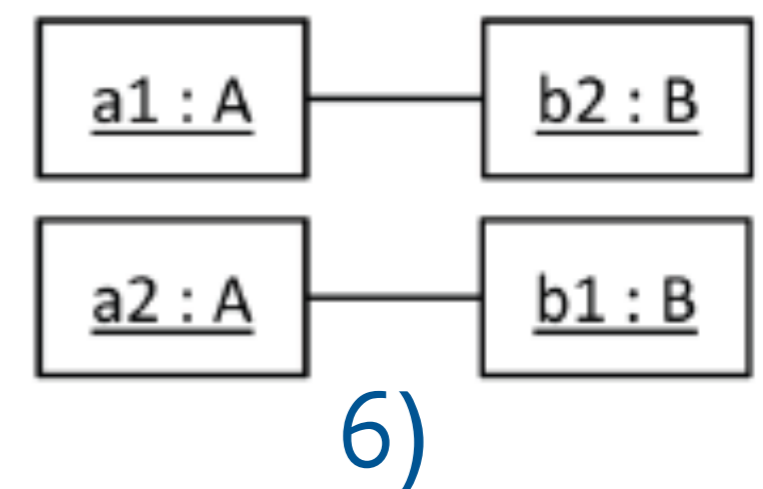
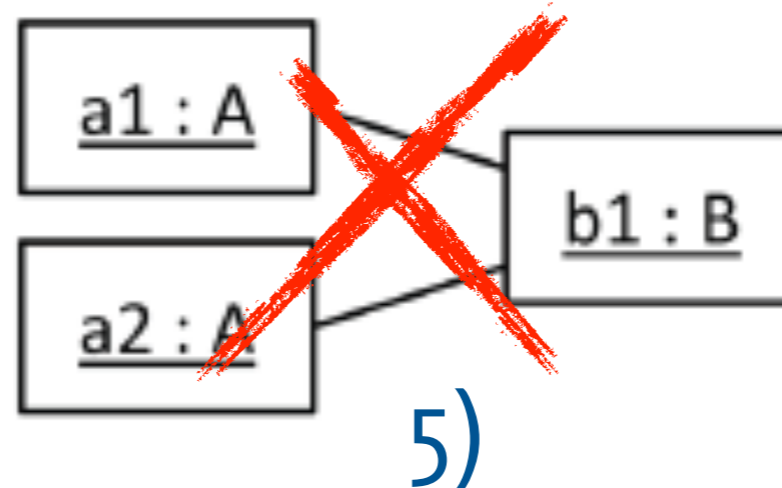
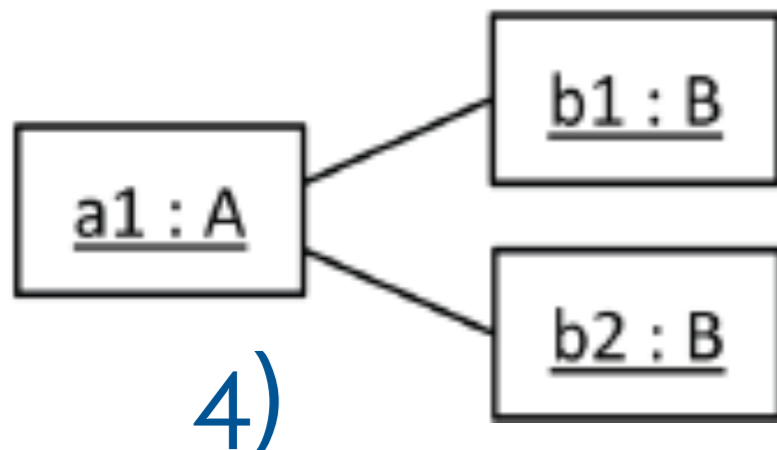
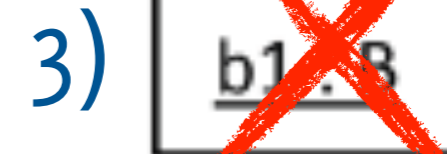
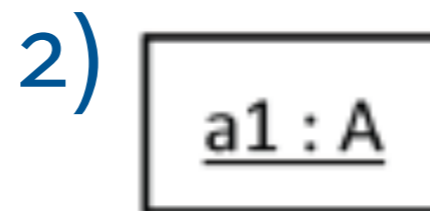
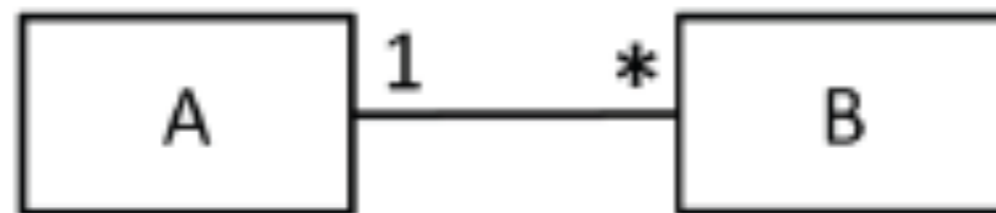
# Quiz: Multiplizitäten

- Anzahl festgelegt durch Multiplizitäten
  - Bedeutung?
- Ausgehend von einem Objekt X wird definiert mit wievielen anderen Objekten Y dieses über Links verbunden sein kann



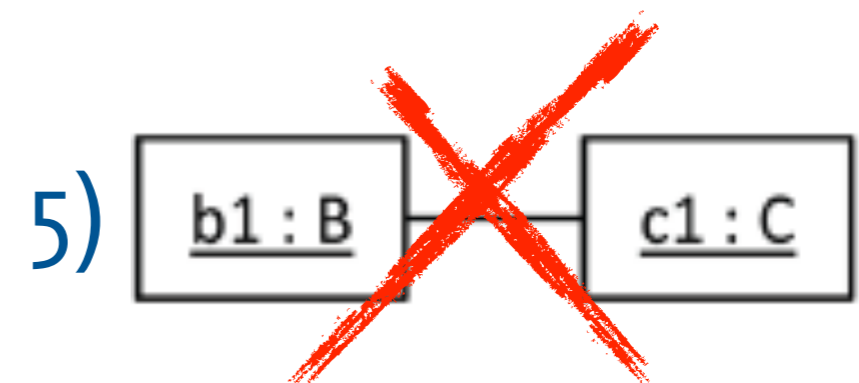
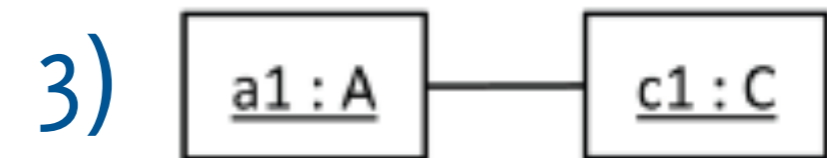
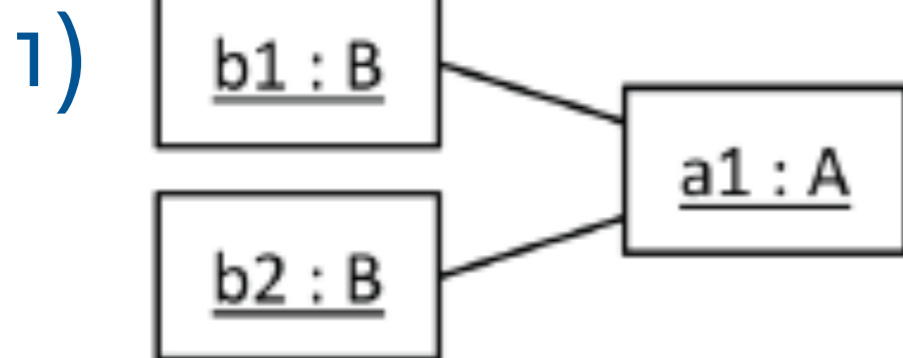
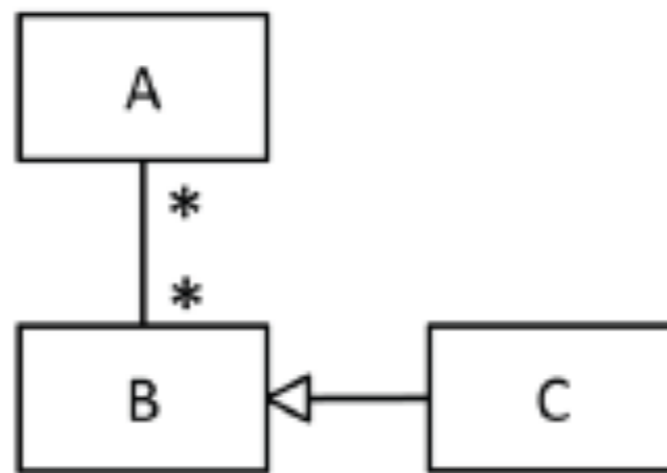
# Quiz: Assoziation und Multiplizitäten

- Welche der folgenden Objektdiagramme passen zu diesem Klassendiagramm?



# Quiz: Assoziationen und Spezialisierung

- Welche der folgenden Objektdiagramme passen zu diesem Klassendiagramm?

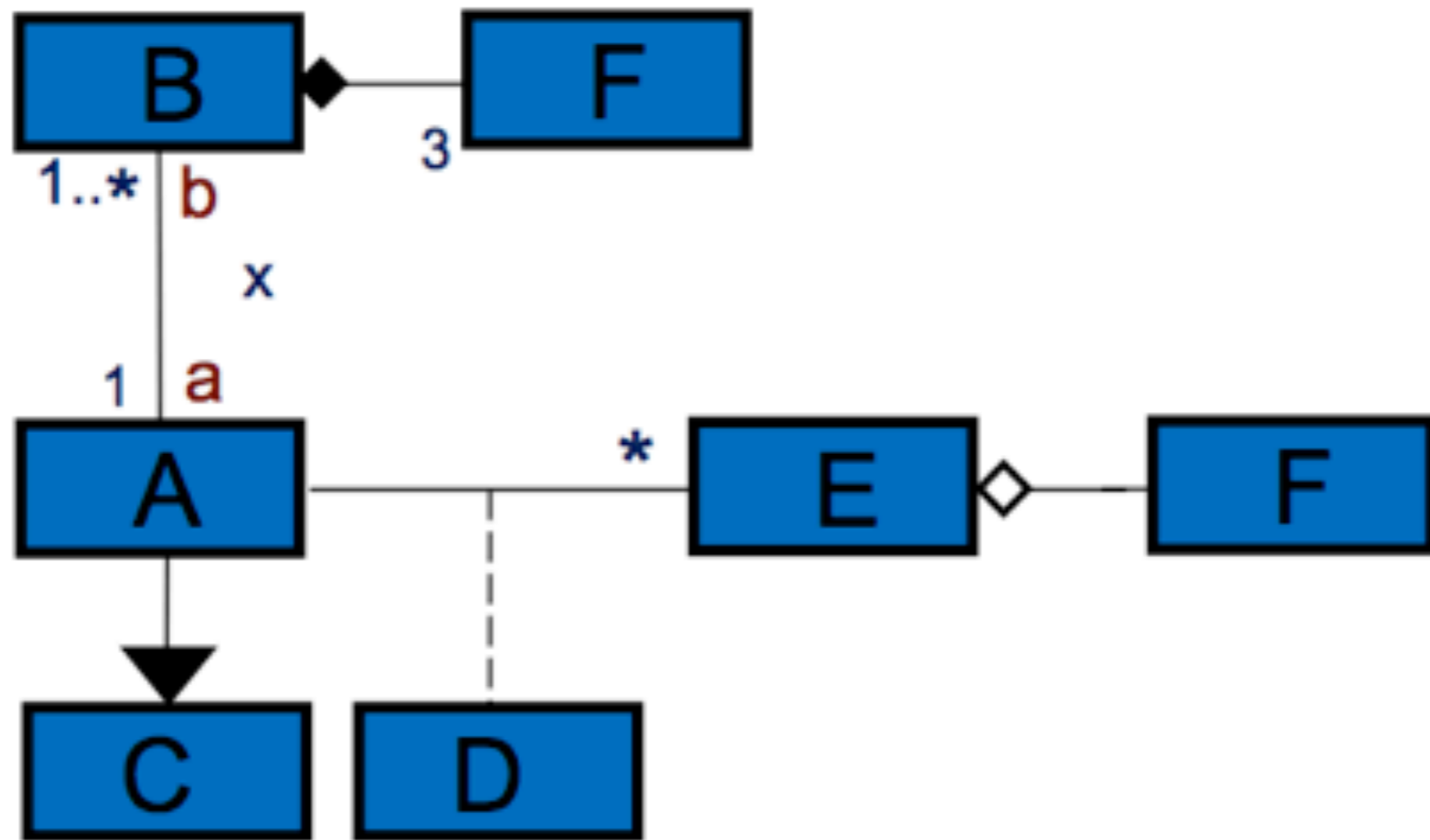




# Quiz: Eigenschaften an Assoziationsenden

- Was ist die Bedeutung von weiteren Eigenschaften an Assoziationsenden?
  - {unordered} <-> {ordered}
    - default: unordered
  - {unique} <-> {nonunique}
    - default: unique

# Quiz: Arten von Assoziationen

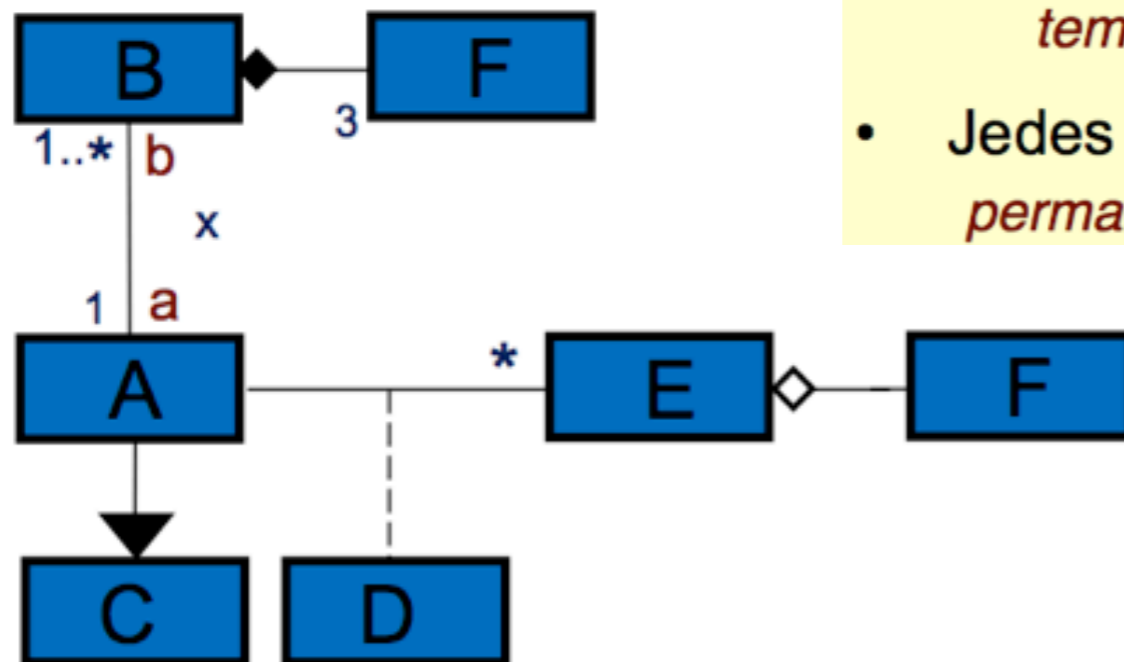


# Quiz: Arten von Assoziationen

A Property has an aggregation property, of type AggregationKind. This is an enumeration with the following literal values:

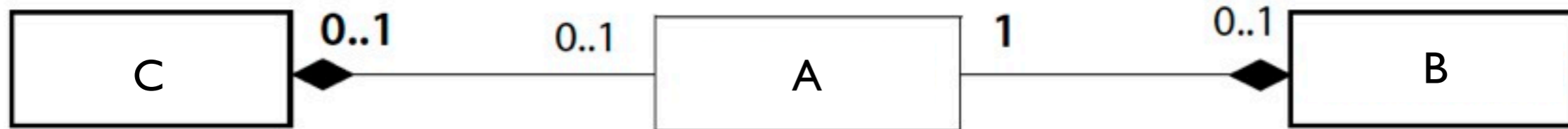
none	Indicates that the Property has no aggregation semantics.
shared	Indicates that the Property has shared aggregation semantics. <u>Precise semantics of shared aggregation varies by application area and modeler.</u>
composite	Indicates that the Property is aggregated compositely, i.e., <u>the composite object has responsibility for the existence and storage of the composed objects (parts).</u>

Composite aggregation is a strong form of aggregation that requires a part (see 11.2.3) instance be included in at most one composite instance at a time. If a composite instance is deleted, all of its parts are normally deleted with it.



- Jedes E-Objekt besitzt ein F-Objekt (Aggregation)  
*temporäre Teil-Ganzes-Beziehung*
- Jedes B-Objekt besitzt 3 F-Objekte als (Komposition)  
*permanente Teil-Ganzes-Beziehung*

# Komposition



**Figure 14-88.** *Shared part class*

In **Figure 14-88**, an **A** may optionally be part of either a **B** or a **C**. It can't be part of both at one time (by the rules of composition). This model does not prevent an **A** from starting as part of a **C** and then becoming part of a **B** (at which time it must cease being part of the **C**). In fact, an **A** need not be part of anything. Also, from this model fragment, we cannot preclude the possibility that **A** is optionally part of some other class that is not shown on the diagram or that might be added later.

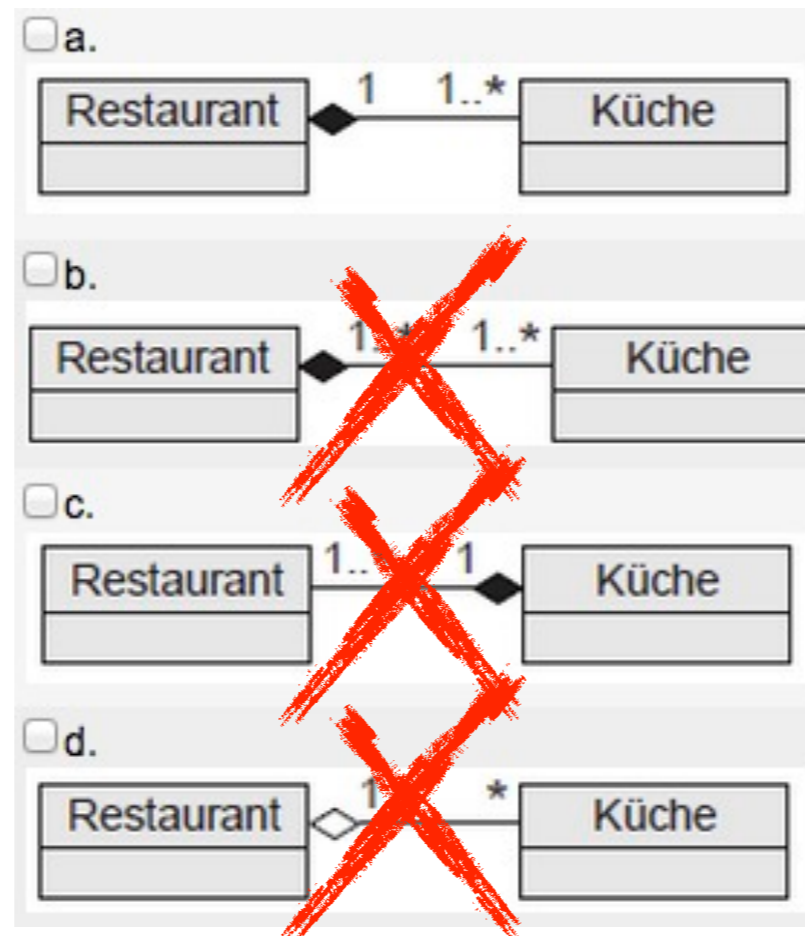
entnommen aus UML Reference Manual

# Komposition

part of only one other object at a time chosen from one of those classes. The entire graph of composition and aggregation links and objects must be acyclic, even if the links come from different associations. Note that these constraints apply to the instance domain—the aggregation associations themselves often form cycles, and recursive structures always require cycles of associations.

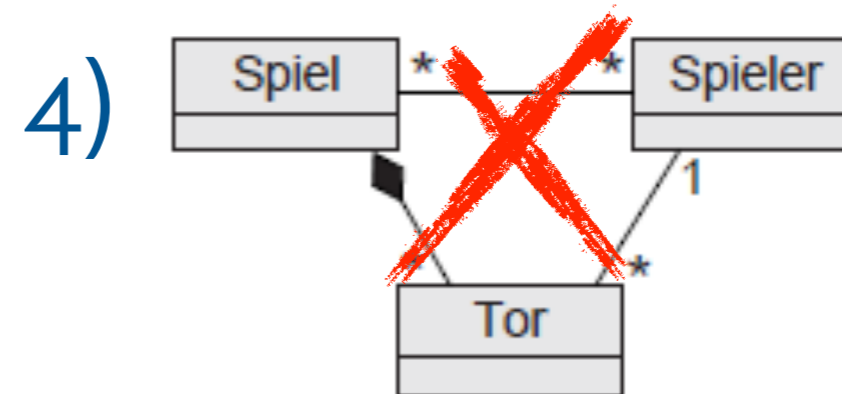
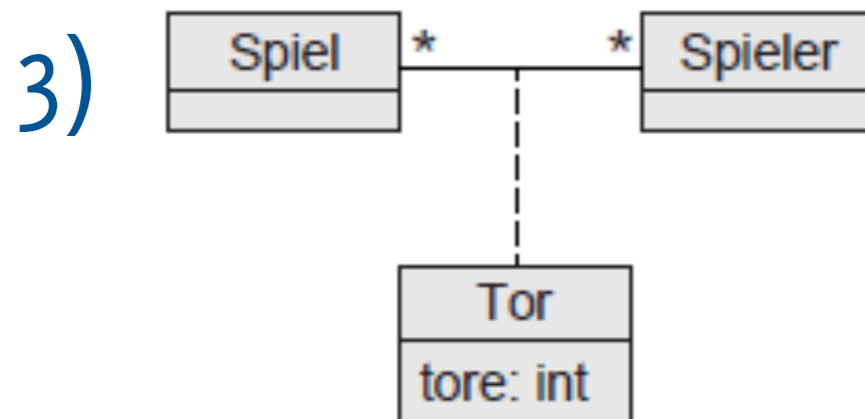
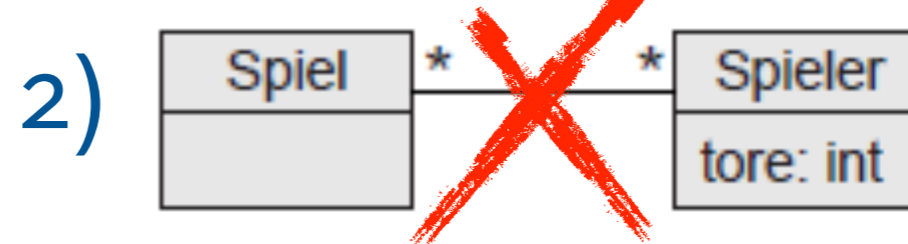
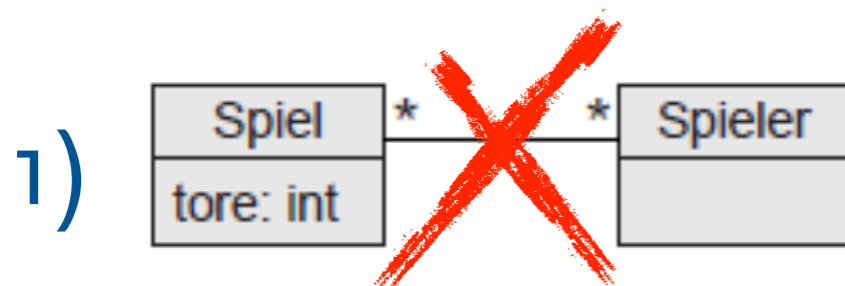
# Quiz

- Wie sollte folgender Sachverhalt modelliert werden?
  - Jedes Restaurant beinhaltet mindestens eine Küche, eine Küche gehört zu genau einem Restaurant.



# Quiz

- Wie sollte folgender Sachverhalt modelliert werden?
  - Im Laufe einer Fußballsaison nehmen mehrere Spieler an mehreren Spielen teil. Jeder Spieler schießt bei jedem Spiel eine bestimmte (unterschiedliche) Anzahl an Toren, die jeweils zu erfassen ist.

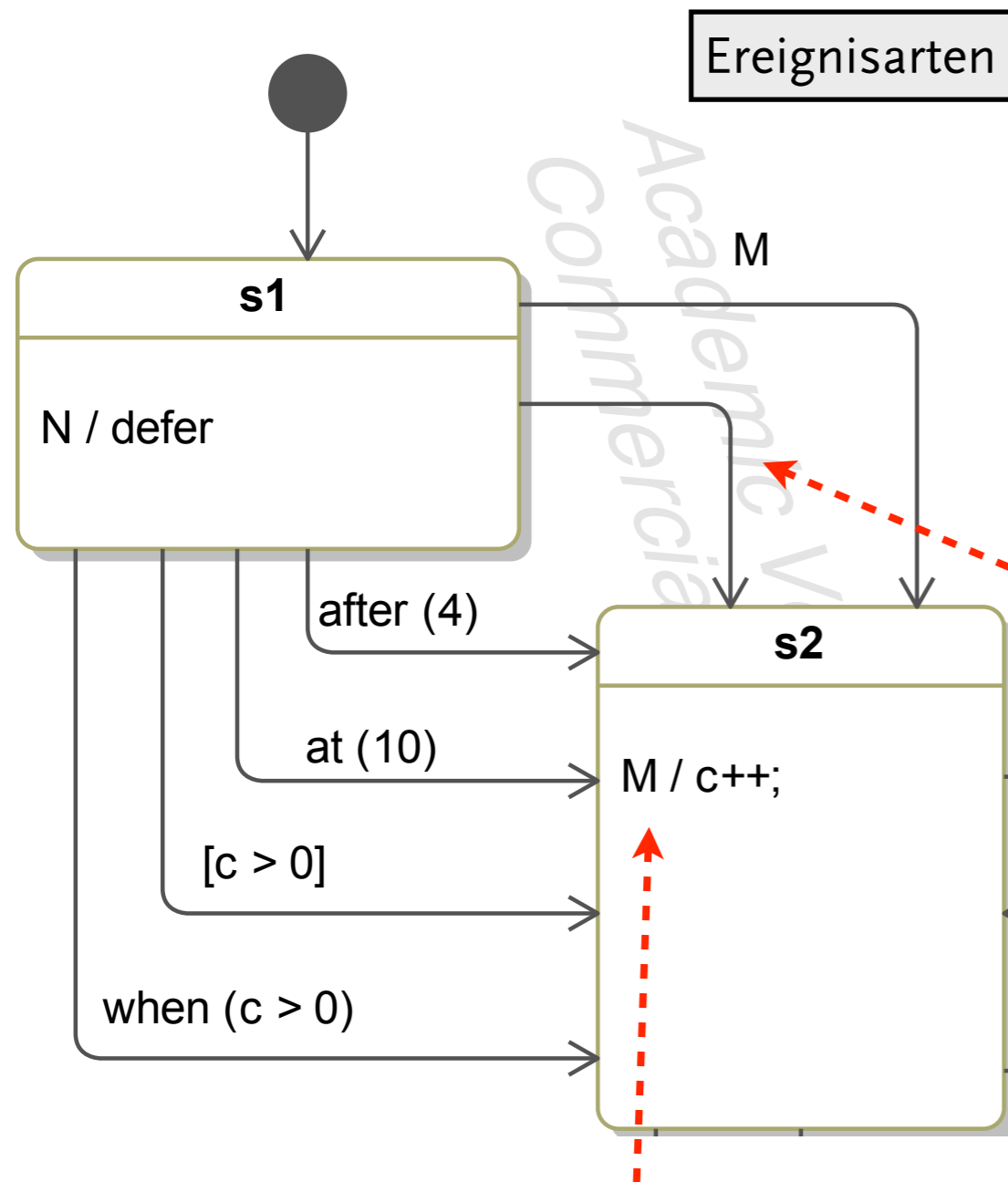


# 2. Teil

- Verhaltensmodellierung mit UML-Zustandsautomaten
- Ziel: Verwendung eines UML-Modells als Grundlage für eine Simulation
- D.h. Ausführung und nicht nur Veranschaulichung
- Dazu Präzisierung der Ausführungssemantik
  - Ereignisse
  - Einfache Zustände, Zusammengesetzte Zustände
- Bei Unklarheiten: UML-Standard



# Zustandsübergänge



Ereignisarten

## Ereignisarten:

- Zeit (TimeEvent - absolut oder relativ)
- Signalempfang (SignalEvent)
- best. Zustand liegt vor (ChangeEvent)
- Fertigstellung (CompletionEvent)

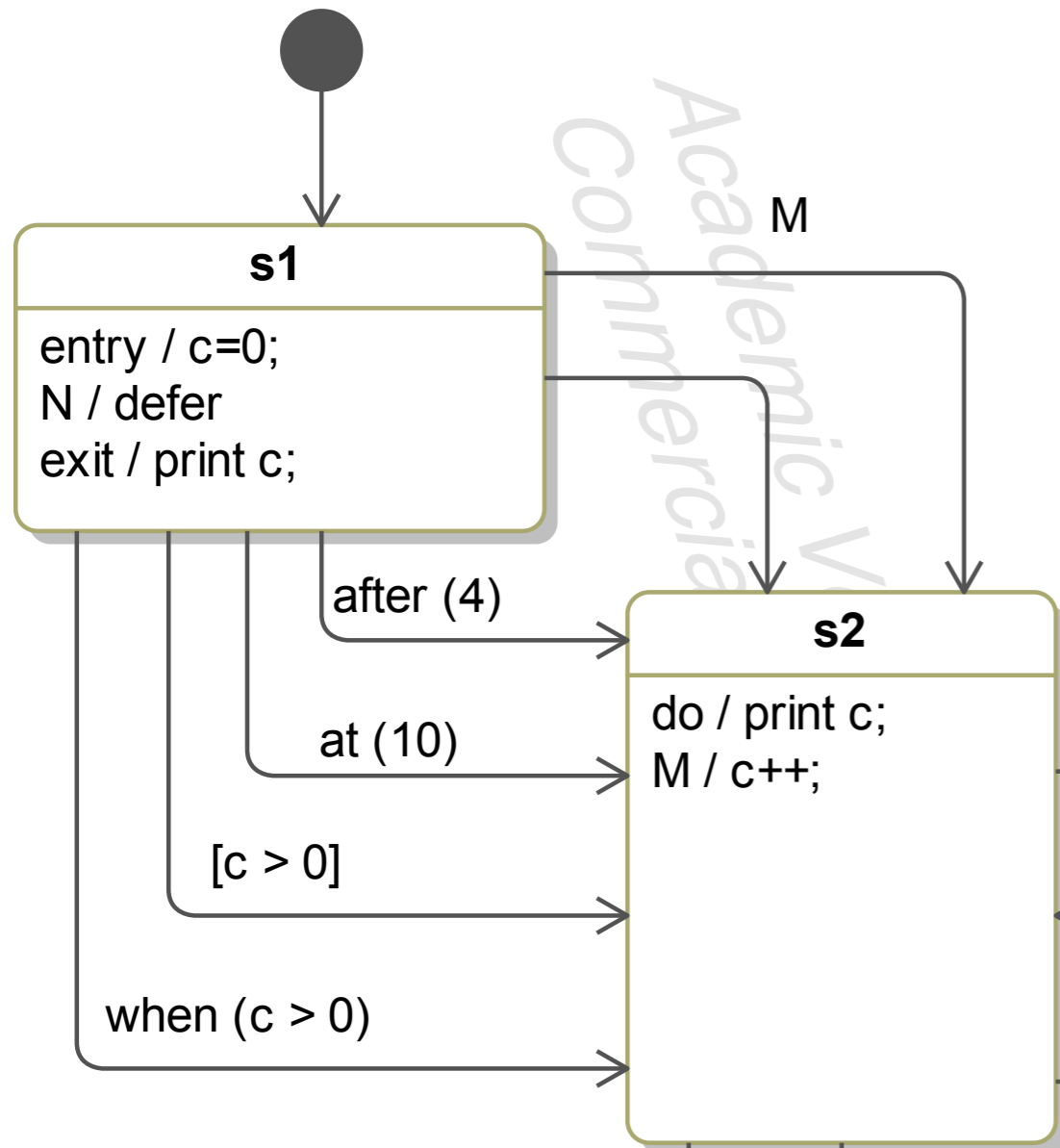
## Reihenfolge der Abarbeitung

CompletionEvent wird automatisch nach Abarbeitung von do oder wenn kein do existiert, dann nach entry erzeugt.

CompletionEvents sind immer höher priorisiert als alle anderen Ereignisse.

Bei internen Transitionen wird der Zustand nicht verlassen. Es finden also keine entry- und exit-Aktivitäten statt.

# Aktionen



Welche Aktionsarten gibt es?

**Aktionsarten:**

- entry
- exit
- do
- effect

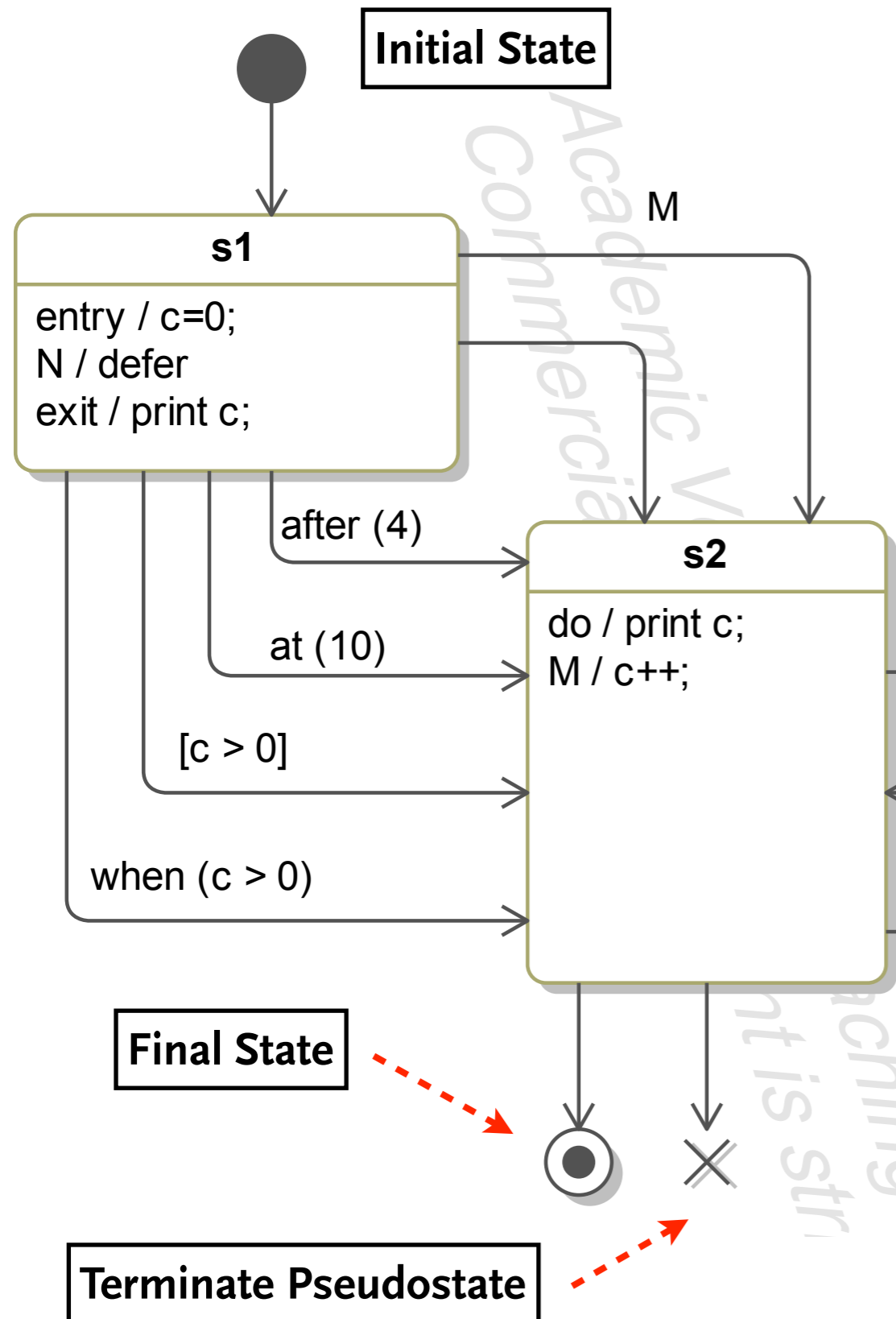
Was passiert wenn eine do-Aktivität läuft und ein Ereignis tritt ein, das zum Verlassen des aktuellen Zustands führt?



## *Exiting a State*

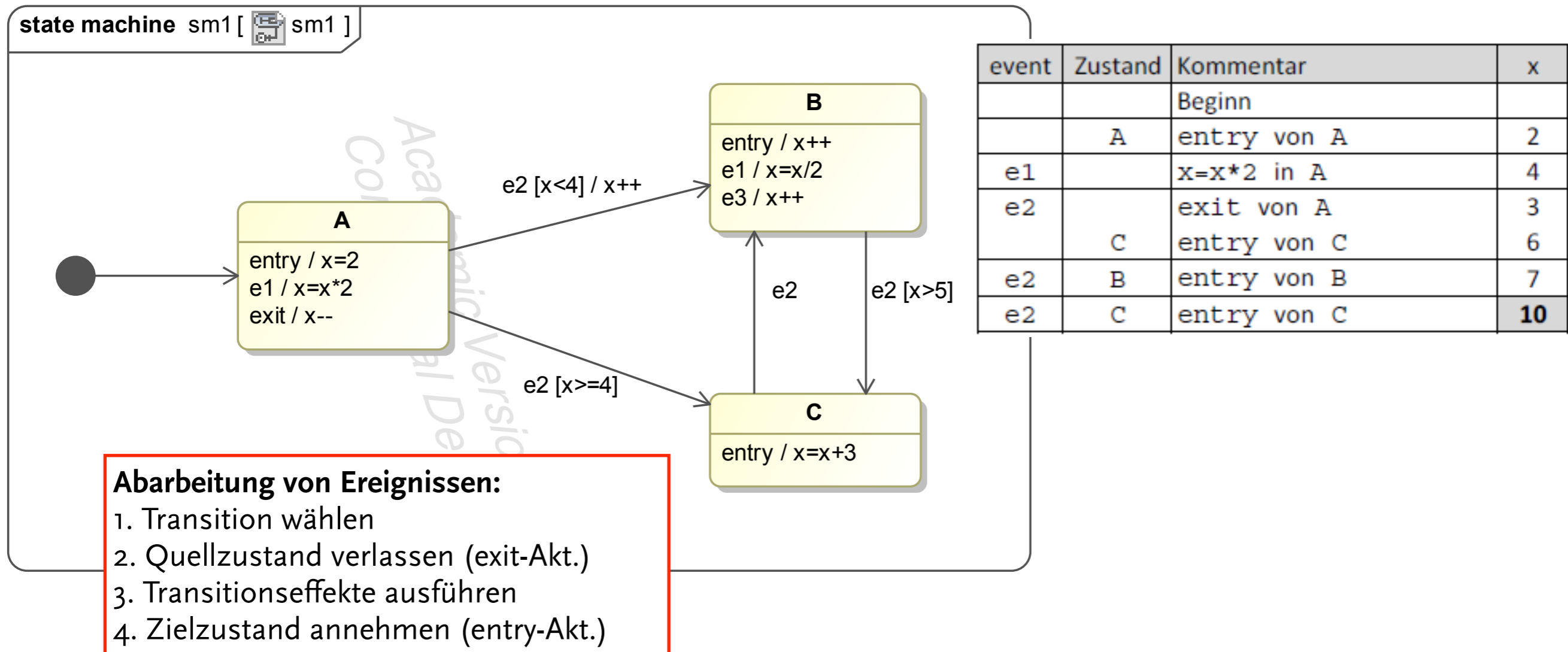
When exiting a State, regardless of whether it is simple or composite, the final step involved in the exit, *after all other Behaviors associated with the exit are completed*, is the execution of the exit Behavior of that State. If the State has a doActivity Behavior that is still executing when the State is exited, that Behavior is aborted *before* the exit Behavior commences execution.

# Spezielle Zustände



# 1. Quiz

Gegeben ist folgender Ausschnitt eines Zustandsdiagramms. Welchen Wert hat  $x$  nach der Ereignisfolge  $e1\ e2\ e2\ e2$  ?



# Zusammengesetzte Zustände

Bei einem zusammengesetzten Zustand ist gleichzeitig der zusammengesetzte Zustand und einer seiner Unterzustände aktiv.

Unterzuständen können wiederum zusammengesetzte Zustände sein.

## Transitionsauswahl:

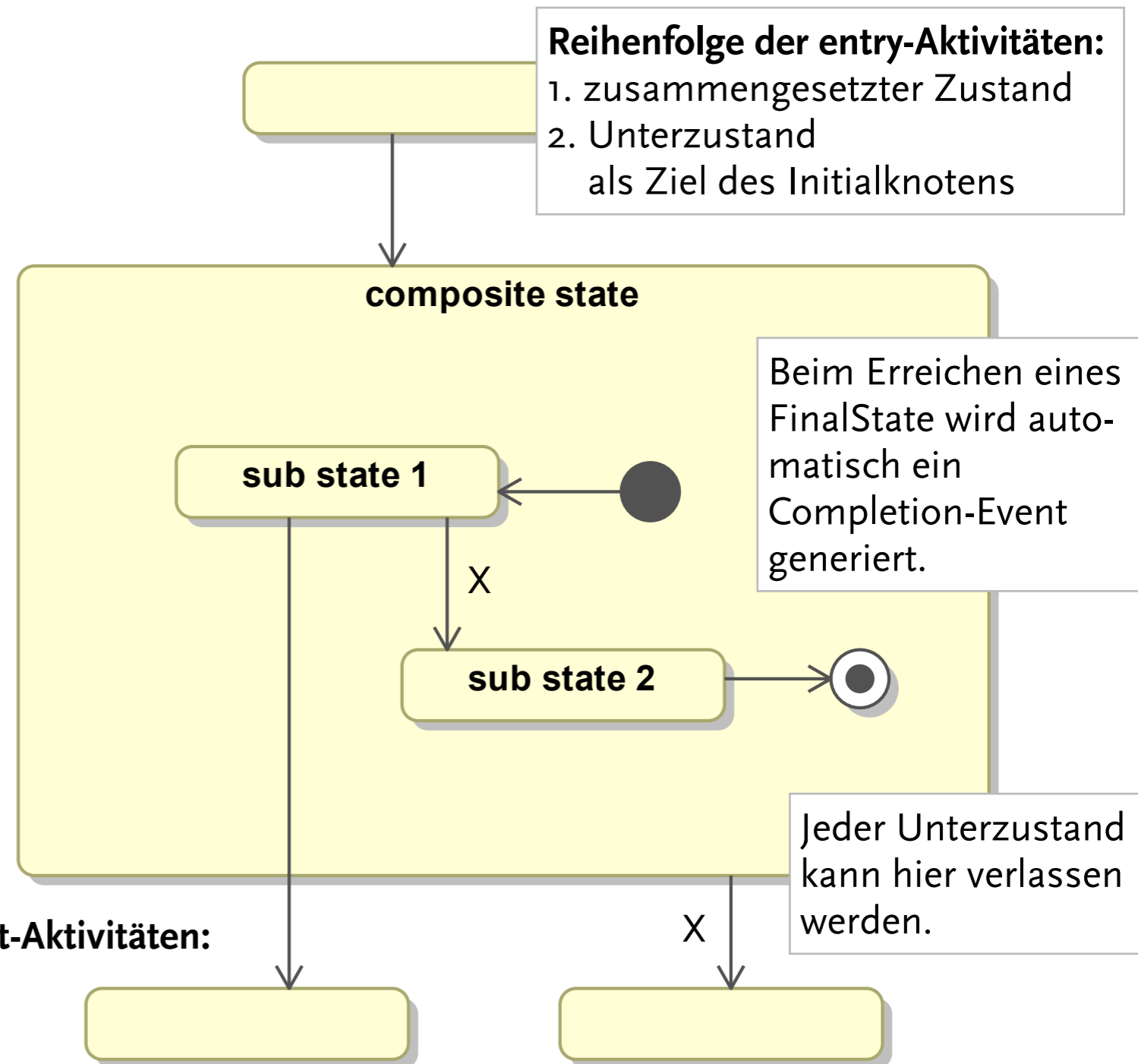
Sollte ein zusammengesetzter Zustand und der aktive Unterzustand auf das gleiche Ereignis reagieren können, so besitzt der innerste Zustand stets die höchste Priorität. Die Priorität nimmt nach außen ab.

## Reihenfolge der exit-Aktivitäten:

1. Unterzustand
2. zsg. Zustand

## Reihenfolge der entry-Aktivitäten:

1. zusammengesetzter Zustand
  2. Unterzustand
- als Ziel des Initialknotens

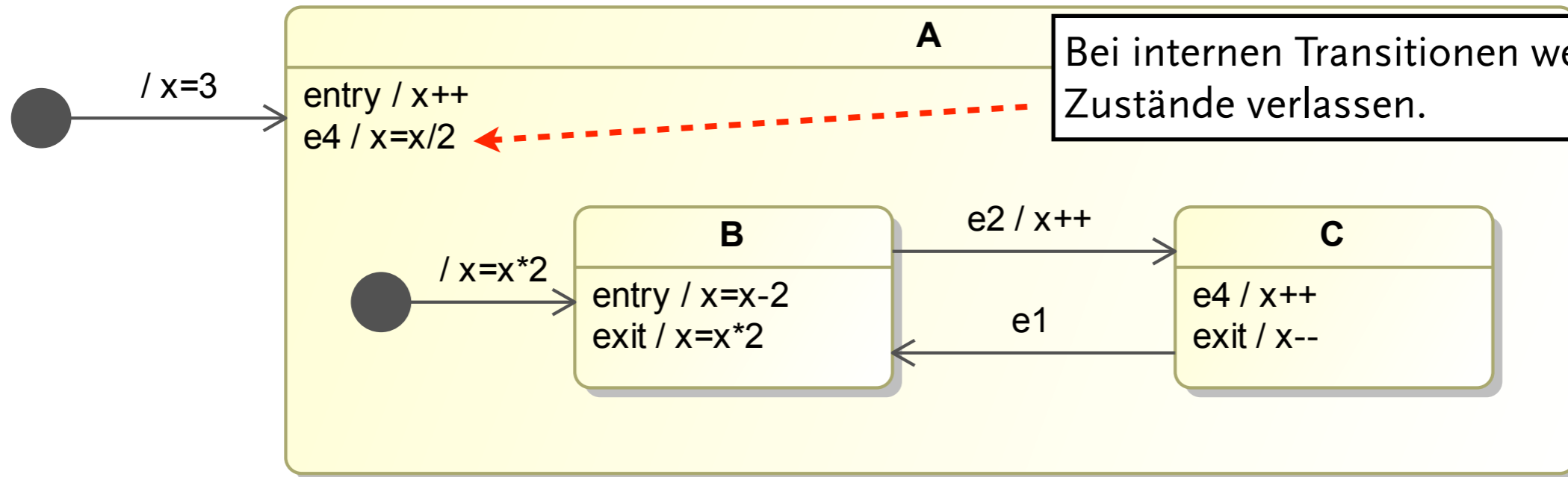


# 2. Quiz

Gegeben ist folgender Ausschnitt eines Zustandsdiagramms.  
 Welchen Wert hat  $x$  nach der Ereignisfolge  $e2\ e4\ e4\ e1\ e4\ e2$  ?

event	Zustand	Kommentar	x
		Beginn	3
	A	entry von A	4
		$x=x*2$ (Startknoten vor B)	8
	A/B	entry von B	6
e2		exit von B $x++$ (Transition)	12
	A/C		13
e4		$x++$ in C	14
e4		$x++$ in C	15
e1		exit von C	14
	A/B	entry von B	12
e4		$x=x:2$ in A	6
e2		exit von B $x++$ (Transition)	12
	A/C		13

state machine sm2 [ sm2 ]



Bei internen Transitionen werden keine Zustände verlassen.

# Werkzeugvorstellung

## MagicDraw UML

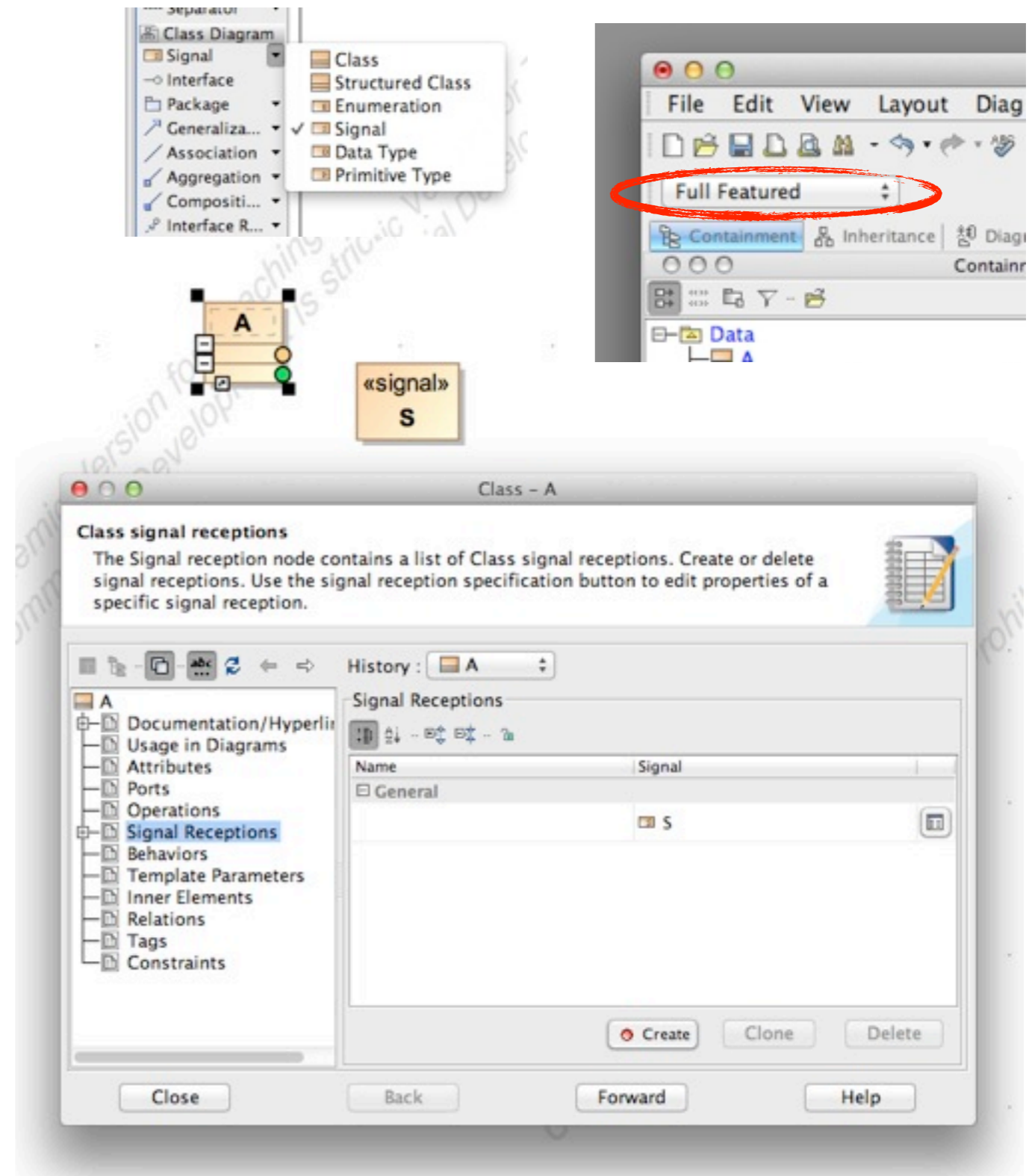
# MagicDraw UML

- MagicDraw UML Personal Edition
  - Akademische Lizenz
  - Versch. Plattformen: Unix, Windows, MacOS
  - Download über die Homepage der Lehrveranstaltung
  - Passwortgeschützter Bereich
    - Benutzername: omg
    - Passwort: \*\*\*\*



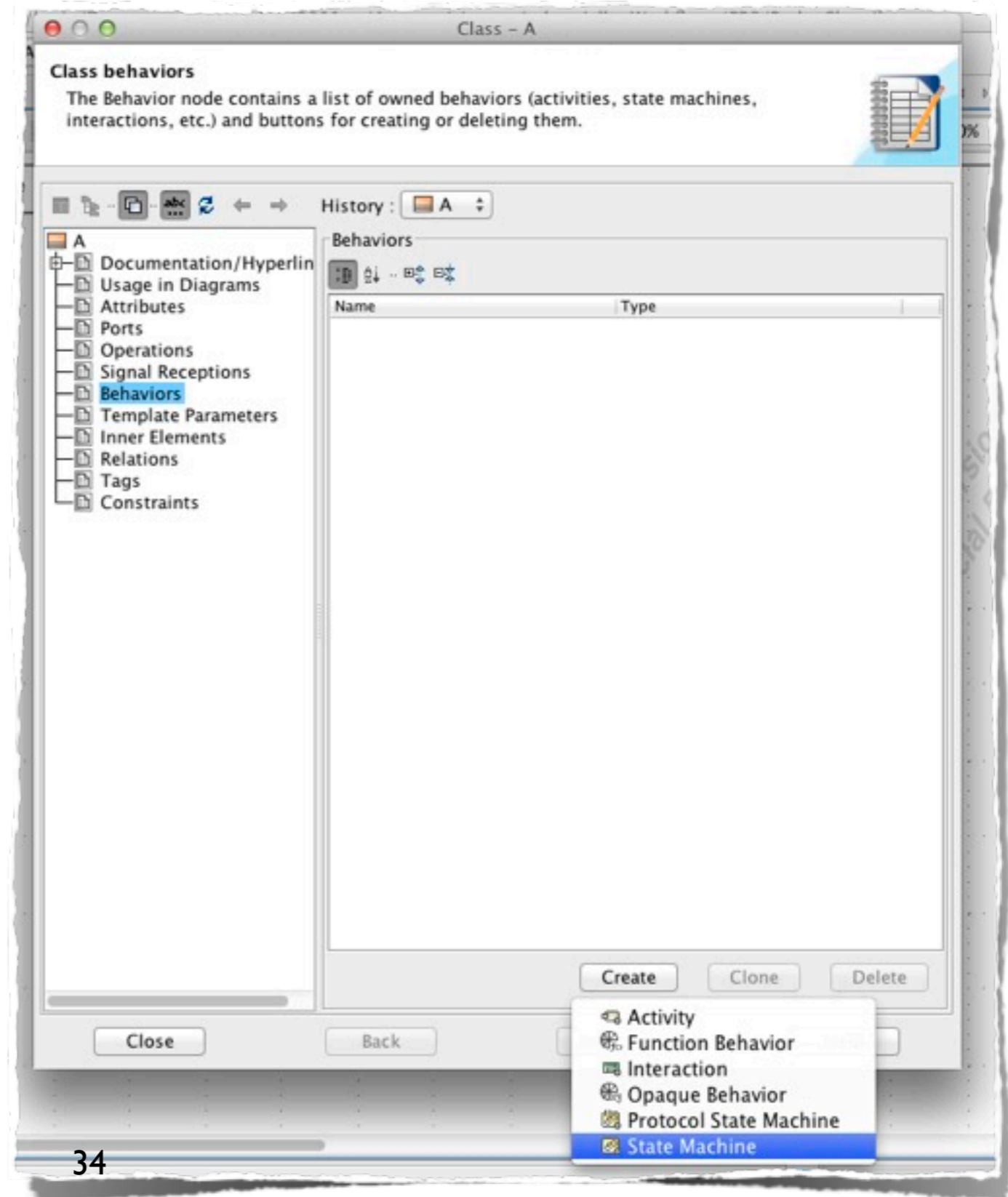
# Hinweise zur Modellierung mit MagicDraw

- Anlegen von Signalen
  - Umstellen auf „Full Featured“
  - Signal als spezielle Klasse erstellen
  - Klassen-Eigenschaft „Signal Receptions“ wählen
  - Empfangbare Signal anlegen



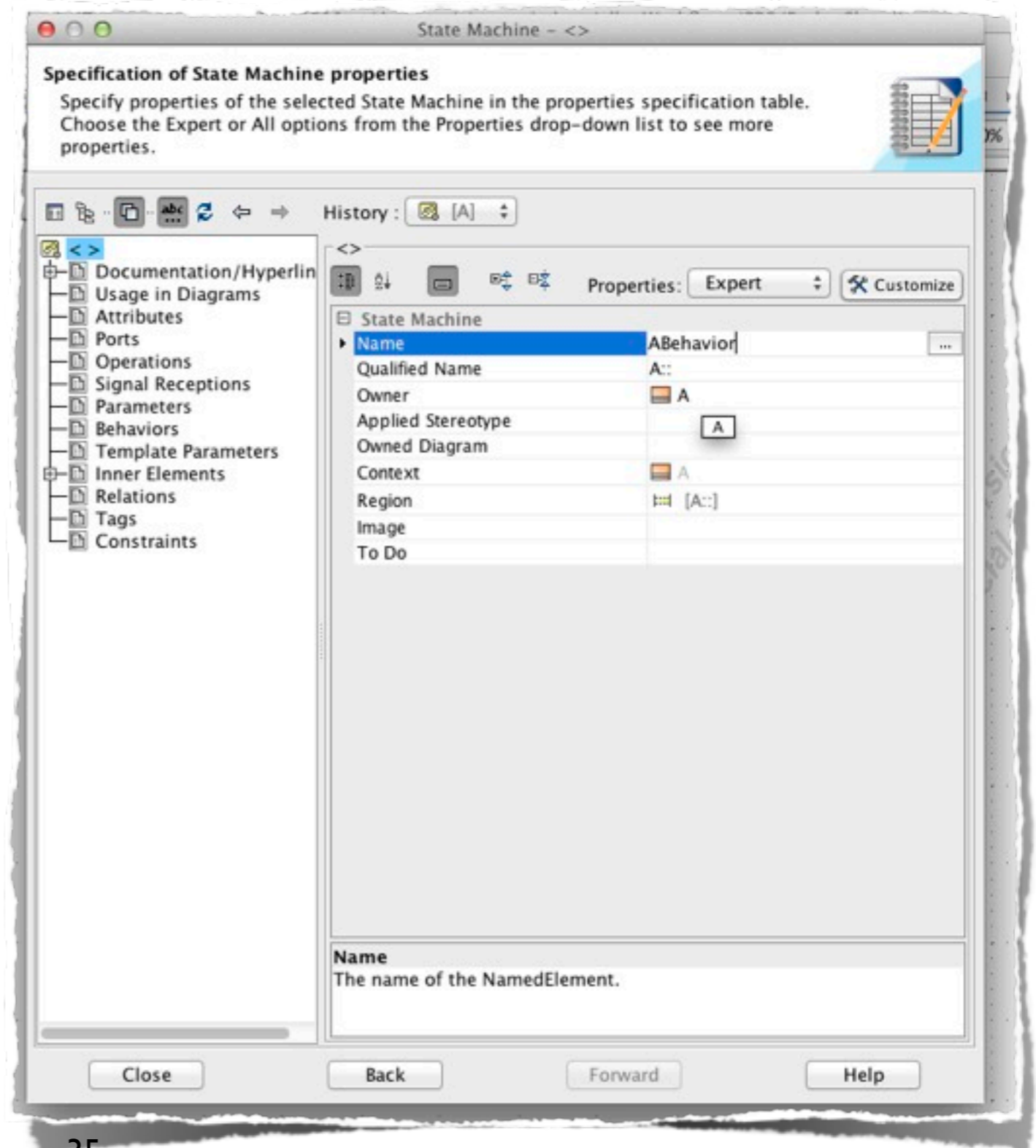
# Hinweise zur Modellierung mit MagicDraw

- Verhalten für aktive Klassen hinterlegen
  - Eigenschaften der Klasse aufrufen
  - Eigenschaft Behaviors wählen
  - Behavior in Form einer StateMachine erstellen



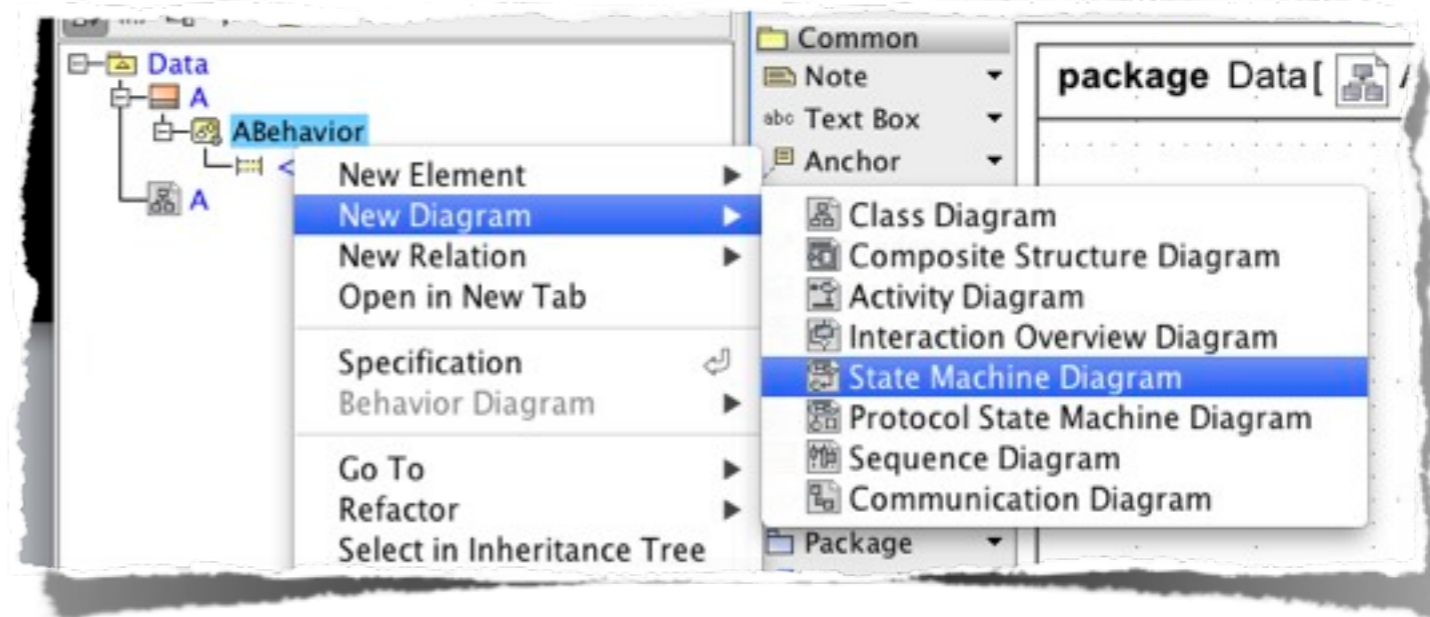
# Hinweise zur Modellierung mit MagicDraw

- Verhalten für aktive Klassen hinterlegen
  - Name für die StateMachine eingeben
  - Dann schließen



# Hinweise zur Modellierung mit MagicDraw

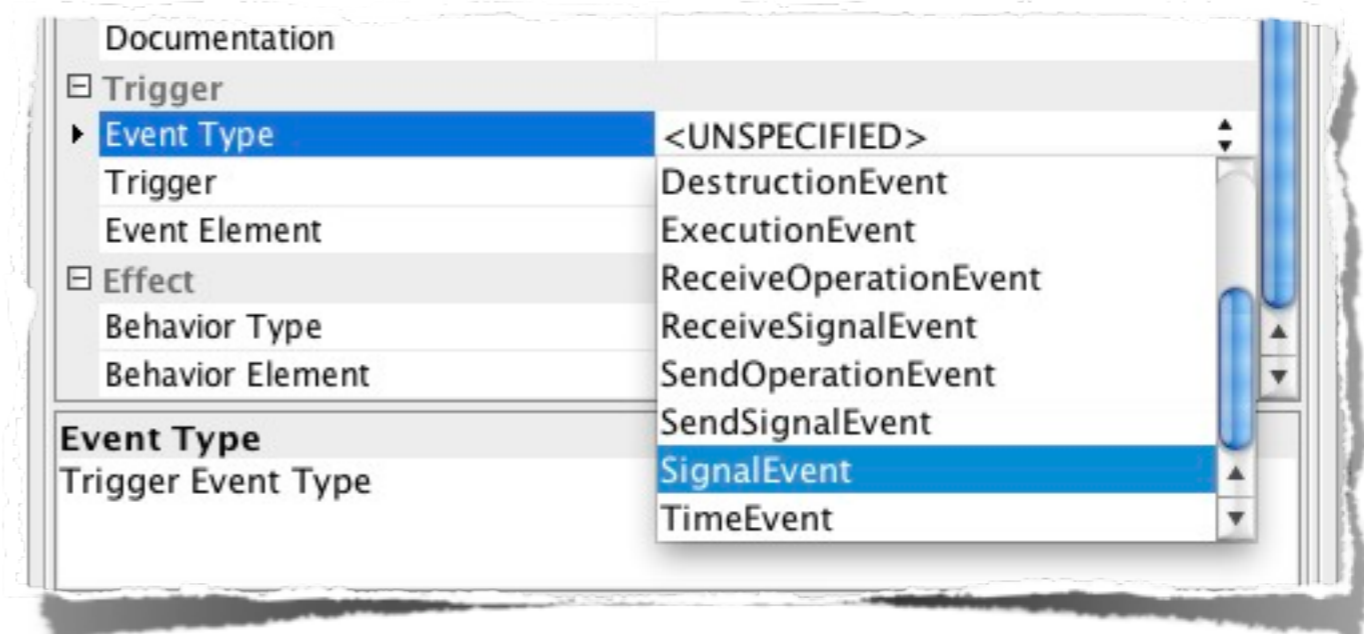
- Verhalten für aktive Klassen hinterlegen
  - StateMachine der Klasse im Navigationsbaum ausklappen
  - Neues Diagramm „State Machine Diagram“ anlegen



# Hinweise zur Modellierung mit MagicDraw

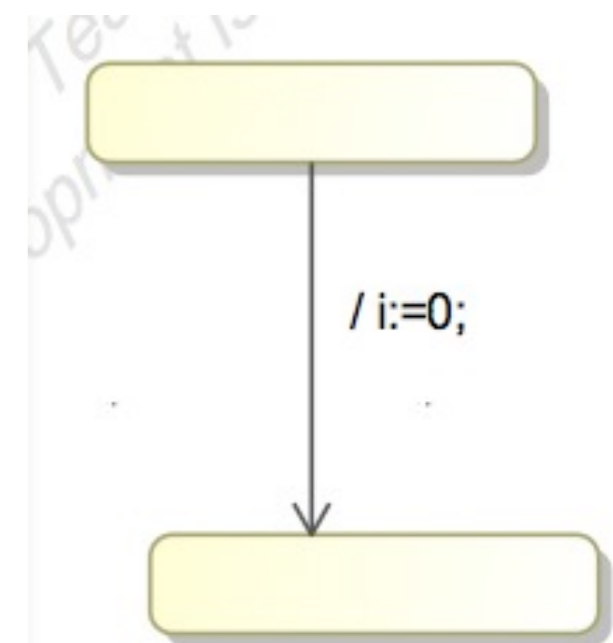
- Ereignisse an Transitionen festlegen
  - Ereignisart werden als Trigger-Eigenschaft festgelegt
  - Häufig verwendete Ereignisarten
    - **SignalEvent** für Signalempfangs-Ereignisse benutzen (nicht ReceiveSignalEvent)
    - **TimeEvent** für Zeitereignisse
    - **ChangeEvent** für Zustandsergebnisse

Ausschnitt aus  
den Eigenschaften  
einer Transition:



# Hinweise zur Modellierung mit MagicDraw

- Aktivitäten als entry/exit/do/effect hinterlegen
  - BehaviorType „OpaqueBehavior“
  - Beschreibung der Aktivität unter der Eigenschaft „Name“ (da „Body“ im Diagramm nicht angezeigt wird)



Bsp.: effect an Transition:

The screenshot shows the MagicDraw interface with three property editors:

- Left Editor:** Shows a tree view with 'Effect' expanded and 'Behavior Type' selected.
- Middle Editor:** Shows the 'Anmelden' activity with 'SpielBehavior [Spiel]' selected. The 'OpaqueBehavior' option is highlighted in the dropdown menu.
- Right Editor:** Shows the properties for the selected 'OpaqueBehavior' effect.

Property	Value
Behavior Type	OpaqueBehavior
Behavior Element	i:=0; [A::AVerhalten:::]
Name	i:=0;
Qualified Name	A::AVerhalten:::i:=0; i:=0; [A::AVerhalten:::]
Owned Diagram	
Specification	
Language	
Body	i:=0;