Interaktionsdiagramme in UML

Interaktionsdiagramm ist ein Oberbegriff für eine Reihe von Diagrammen, die das Verhalten eines objektorientierten Systems durch Objektinteraktionen beschreiben

• Ein **Sequenzdiagramm** betont die zeitliche Abfolge von Objektinteraktionen

• Ein **Kollaborationsdiagramm** betont die an der Objektinteraktion beteiligten Objekte

Zur Beschreibung von Objektinteraktionen (der dynamischen Semantik) werden zusätzlich zum Begriff des Objekts folgende Begriffe benötigt:

• **Nachricht** (asynchrones **Signal** und synchroner **Methodenaufruf**)
• **Aktivierung** eines Objekts
• **aktives Objekt**

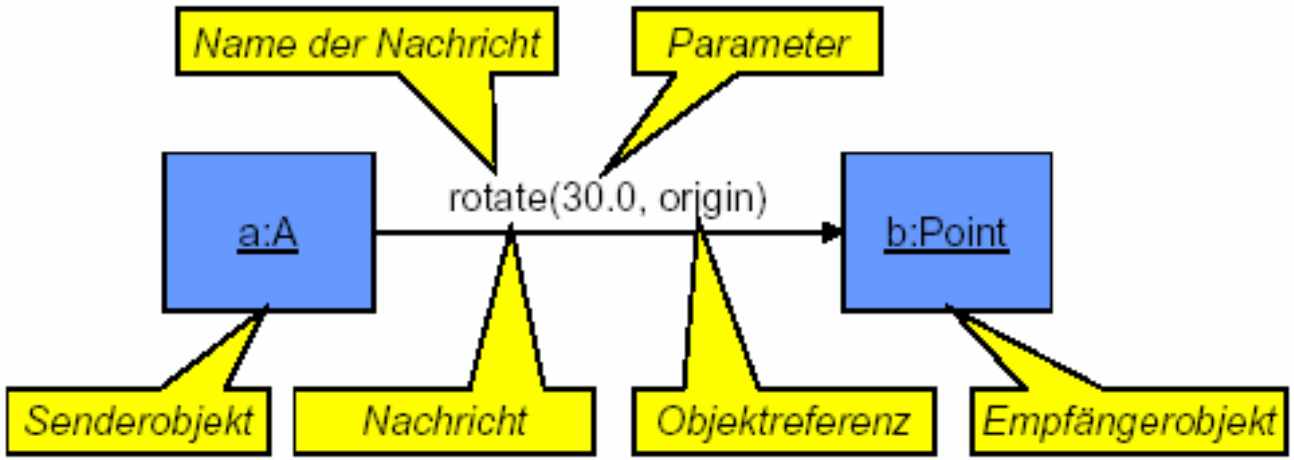
Konzeptuelle Modellierung von Interaktionen

Interaktion erfolgt durch **Nachrichten** zwischen Objekten
Eine Nachricht

- wird von einem (aktiven) **Senderobjekt** verschickt,
- wird an ein **Empfängerobjekt** geschickt,
- besitzt einen **Namen,**
- besitzt eine geordneten Folge von **Parametern** und
- besitzt eine **Nachrichtenlaufzeit** vom Sender zum Empfänger

Ein Parameter ist ein Wert oder eine Objektreferenz
Eine Objektreferenz in einem Parameter wird durch einen Variabelennamen bezeichnet das Verschicken einer Nachricht ist eine **Aktion**

Konzeptuelle Modellierung von Interaktionen




```
class A {
Point b = ;
public test(Point
        b.rotate(300,
}
}

a = new A();
a.test(p1);
```

Konzeptuelle Modellierung von Interaktionen

Übersicht der Nachrichtenarten:

Name	Pfeiltyp	Sender	Empfänger
Signal		ist aktiv	ist aktiv
Aufruf sequentiell		ist aktiv, blockiert	wird aktiviert
Aufruf asynchron		ist aktiv	wird aktiviert
Ergebnis		Aktivierung fortgesetzt	wird inaktiv

Asynchrone Interaktion

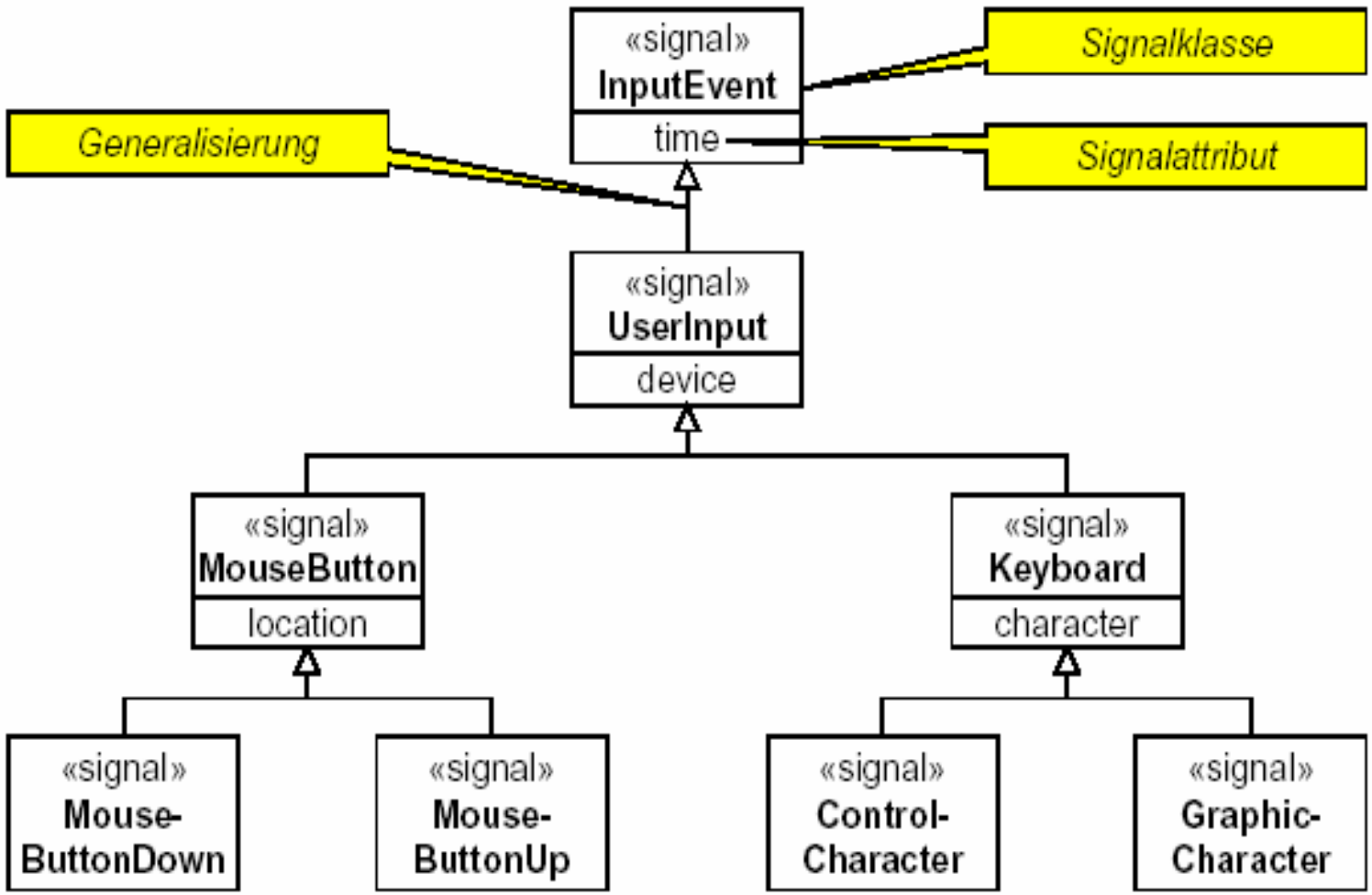
Charakteristika

- Sender- und Empfängerobjekt sind beide **aktive Objekte**: Sie besitzen einen Thread und können Aktionen auslösen (vgl Stereotyp Prozeßobjekt)
- Die Kommunikation erfolgt unidirektional über ein **Signal**
- Der Empfang des Signals löst ein (asynchrones) **Ereignis** beim Empfänger aus
- Der Empfänger kann auf das Eintreten eines Ereignisses **warten**

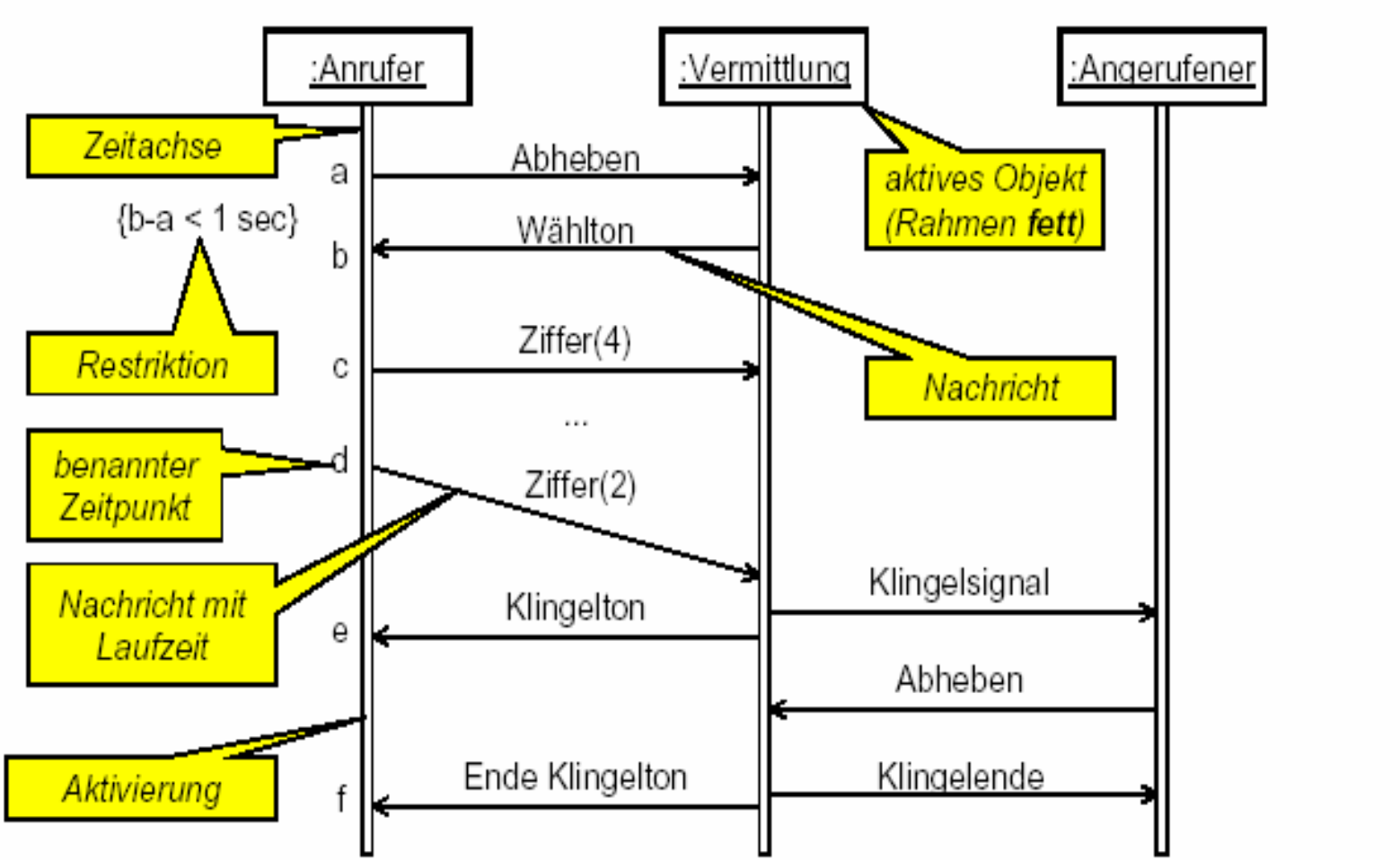
Signale werden durch **Klassen** klassifiziert und beschrieben

- Der Namen der Klasse ist der Name des Signals
- Das Stereotyp «signal» kann verwendet werden
- Die Attribute der Klasse beschreiben die Parameter des Signals

Taxonomie(„Zerlegung komplexer Strukturen“) für Signale



Sequenzdiagramm: Asynchrone Interaktion

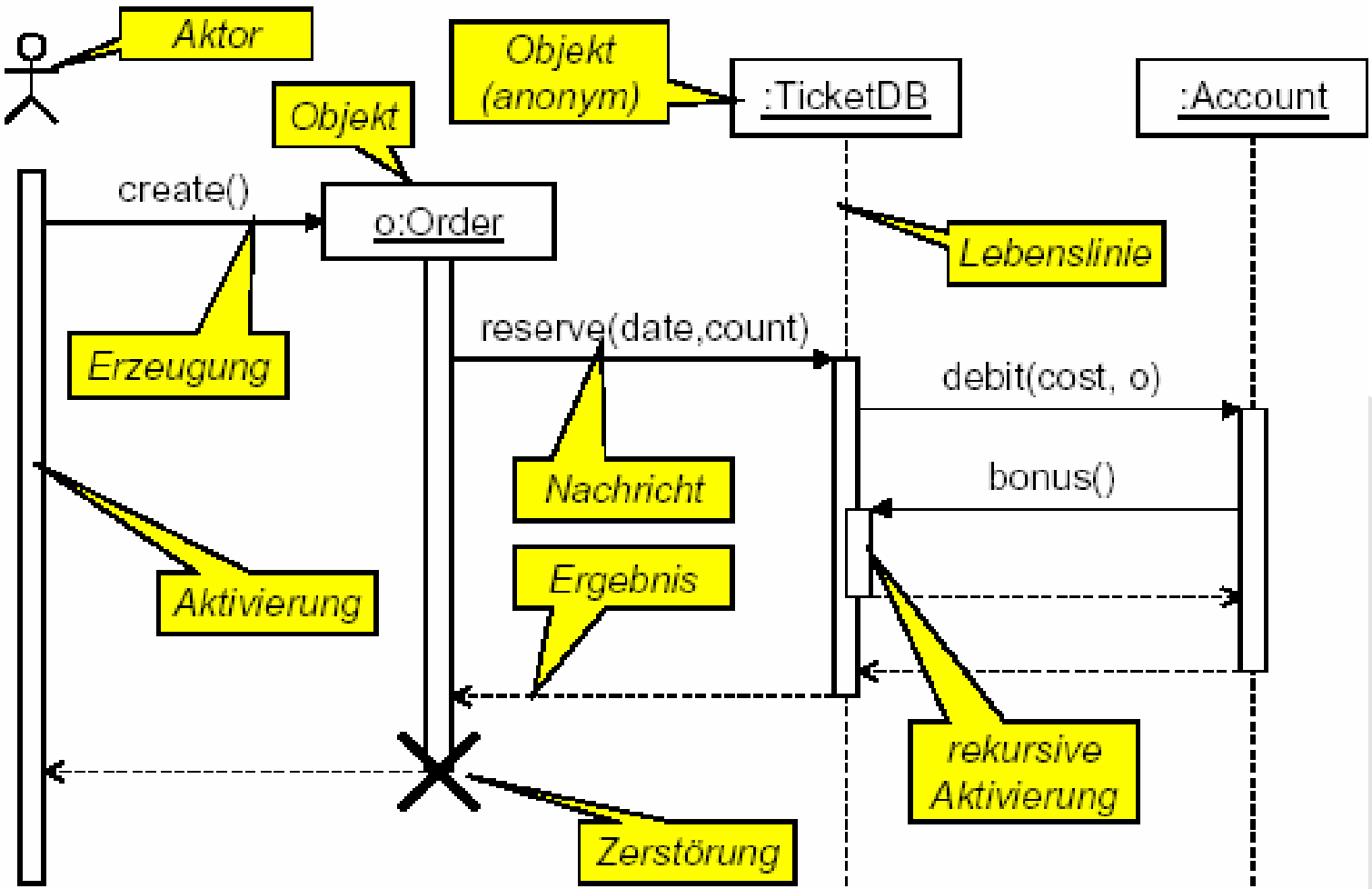


Synchrone (prozedurale) Interaktion

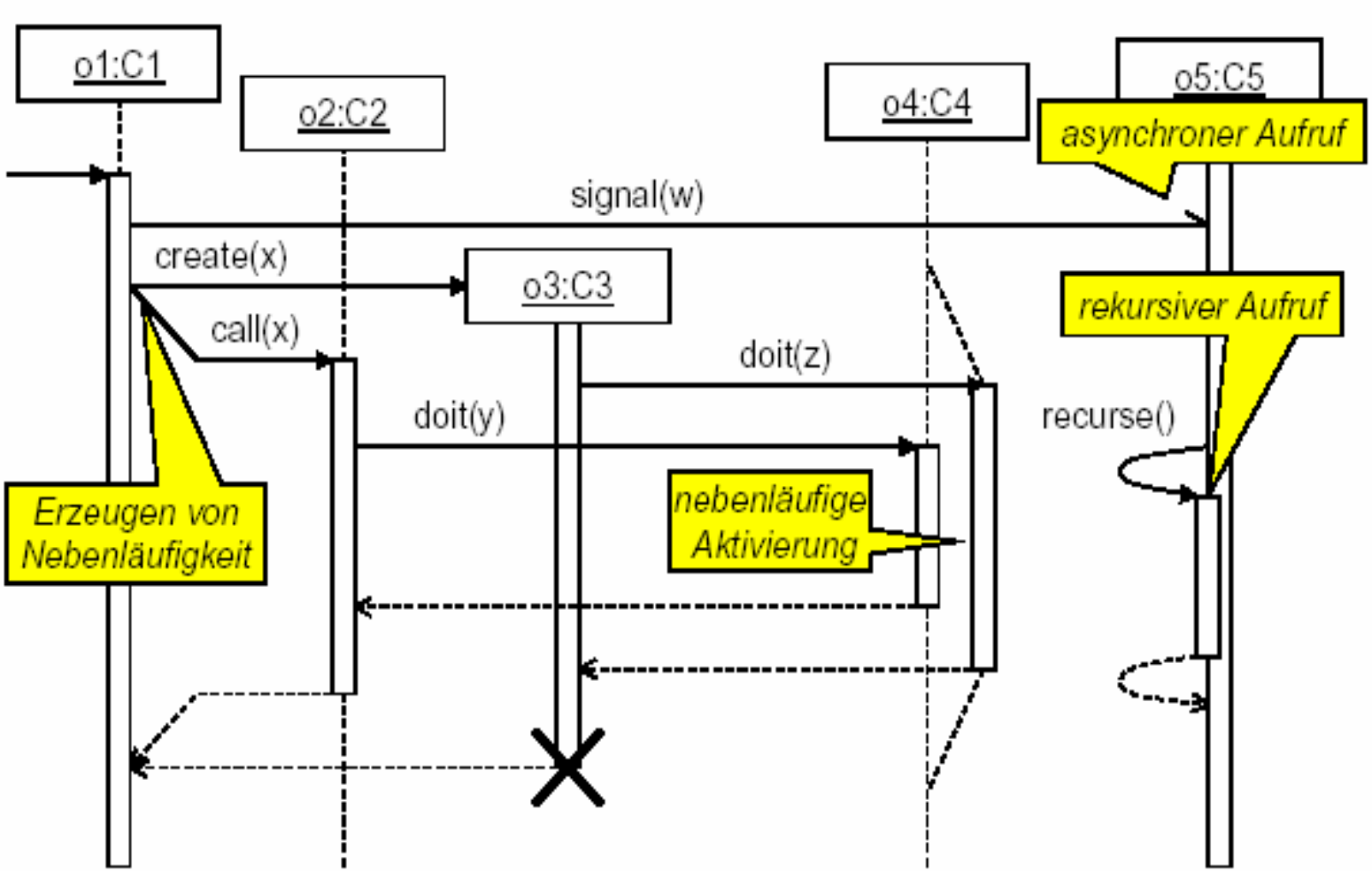
Charakteristika:

- Das aufrufende Objekt muß **aktiviert** sein
- Das aufrufende Objekt aktiviert das aufgerufene Objekt durch einen **Aufruf**
- Der Aufruf ist einer **Nachricht**, die auch in einem Netzwerk entfernt verschickt werden kann (zB RPC, RMI)
- Es findet eine **dynamische Bindung** an die zur Nachricht gehörende Methode beim Empfänger statt
- Es findet eine **Aktivierung** der Methode (Operation) des aufgerufenen Objekts statt
- Während der Aktivierung bleibt das aufrufende Objekt aktiviert (**blockiert**)
- Die Aktivierung des aufgerufenen Objekts endet mit der Ausführung der Methode
- Es wird eine **Rückgabenachricht** verschickt, die als Argument ein Ergebnis besitzen kann
- Die Ausführung der aufrufenden Methode wird fortgesetzt

Sequentielle prozedurale Interaktion



Nebenläufige prozedurale Interaktion



Kollaborationsdiagramm

Statische Elemente

- Aktive oder passive **Objekte** verbunden durch **Referenzen** (Instanzen von Assoziationen oder Aggregationen) mit **Rollennamen**

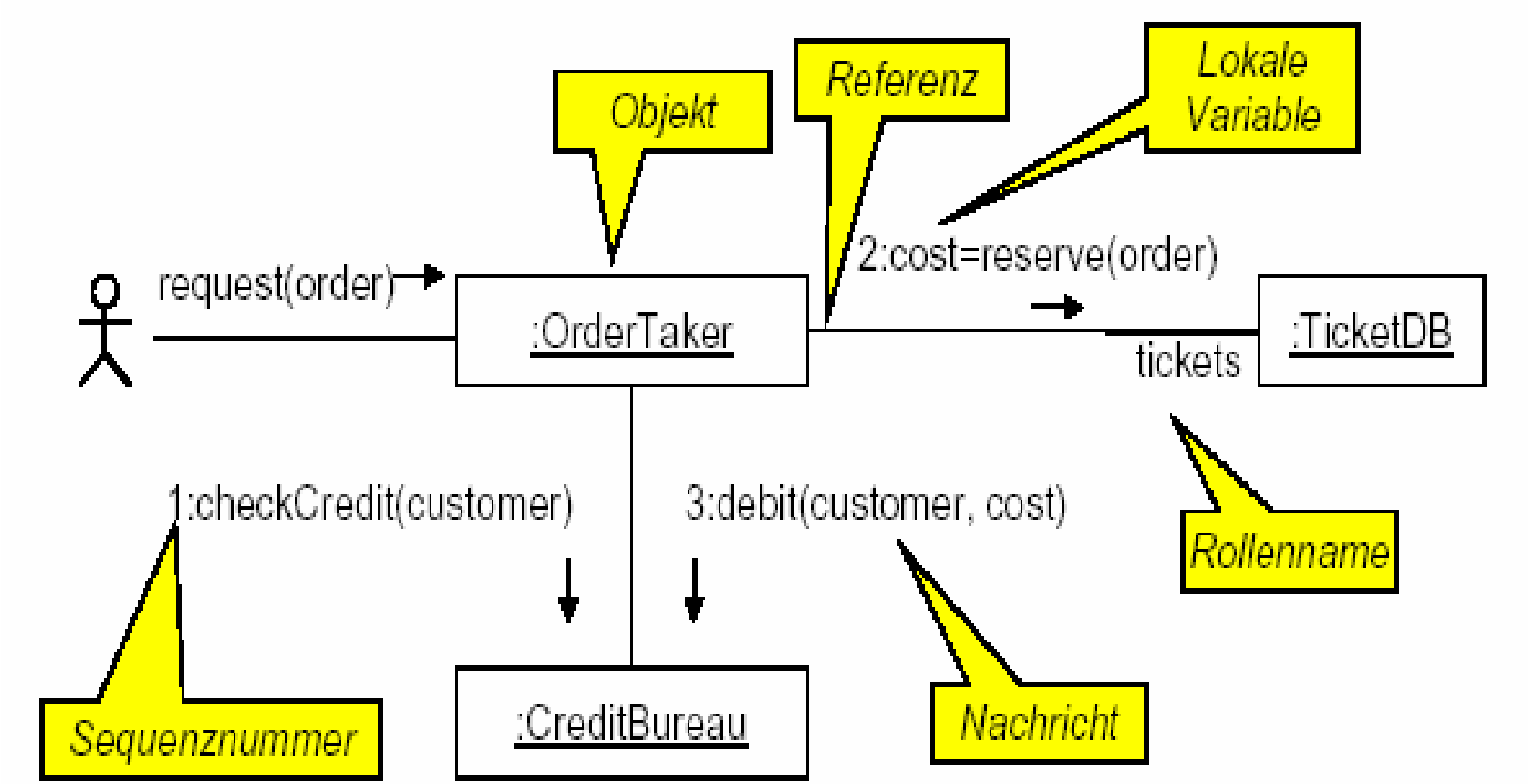
Dynamisch Elemente

- Dynamisch erzeugte **Objekte** {new}
- **Nachrichten** (seltener: Signale) mit **Argumenten**
- **Lokale Variablen**

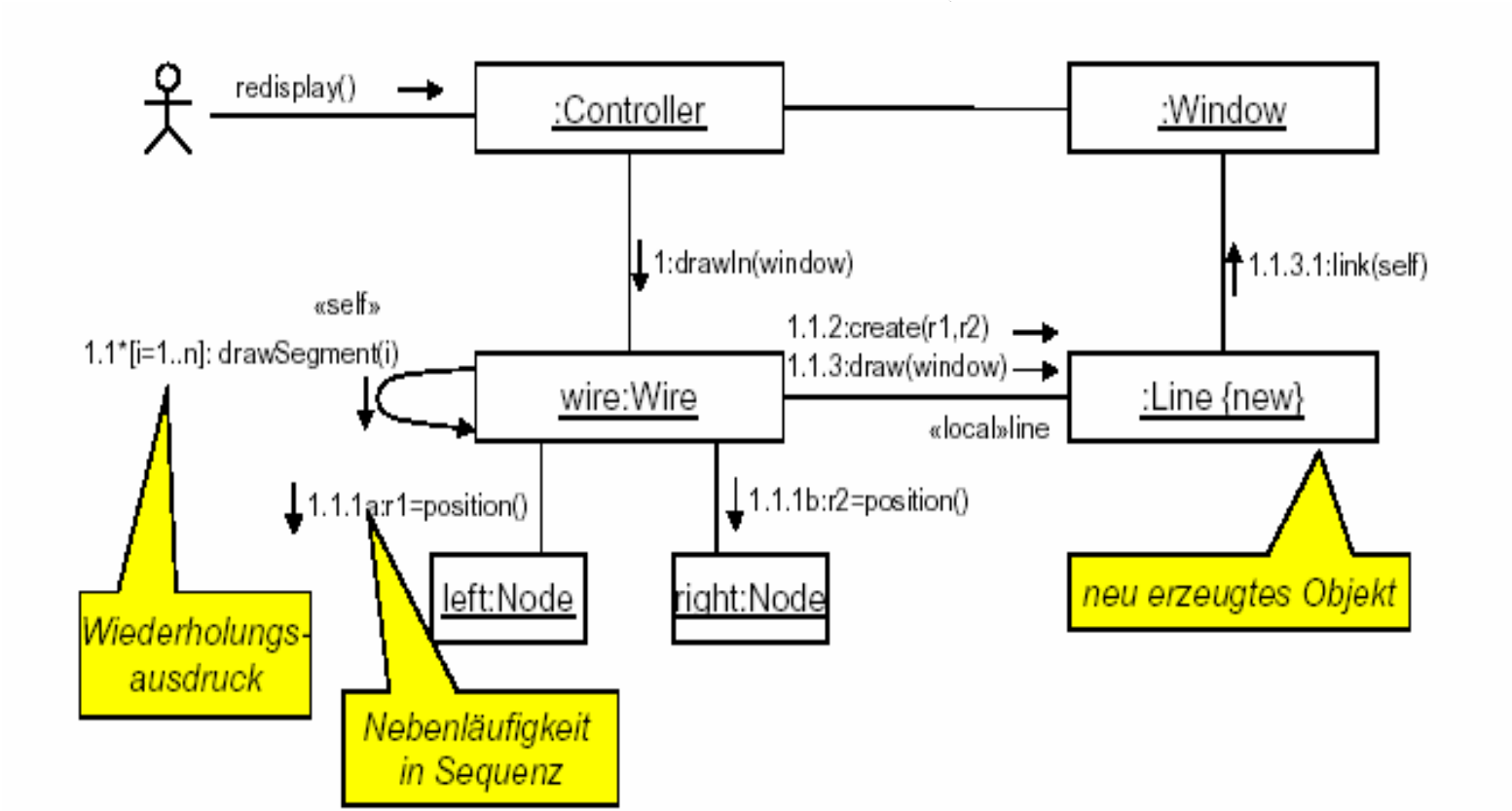
Implizite Elemente

- Zeit (angedeutet durch Sequenznummern) Aktivierungen

Kollaborationsdiagramm: Beispiel



Kollaborationsdiagramm: Beispiel



Kollaborationsdiagramm: Nachrichtenbeschriftung

- 2: display(x, y) Einfache Nachricht mit Sequenznummer
- 1.2.1: p=find(name) geschachtelter Aufruf
- [x<4] 2: invert(rect) bedingte Nachricht
- 3.1*: update Iteration
- 3.2*[i=1..8] 5: draw(v[i]) Iteration
- Nebenläufigkeit und Synchronisation wird besser durch Sequenzdiagramme beschrieben.

Nebenläufigkeit und Synchronisation wird besser durch Sequenzdiagramme beschrieben.

Methodisches Vorgehen

Die **Erstellung von Interaktionsdiagrammen** im Rahmen der *Objektorientierten Analyse* hat das Ziel, alle (externen, d.h. für den Auftraggeber relevanten) Interaktionen zwischen dem System und seiner Umgebung zu identifizieren und ihre *dynamische Semantik* zu beschreiben.

Sie verwendet *Klassendiagramme* und *Interaktionsdiagramme* als Hilfsmittel der Kommunikation und zur Dokumentation der Ergebnisse.

Das *Vorgehensmodell* besteht aus zwei **Phasen**:
1) Identifikation **externer Ereignisse** (Sammlung im **Ereigniskatalog**)
2) Zuordnung von **Verantwortlichkeiten** für diese Ereignisse an **extern sichtbare** Klassen (Methoden)

Ausblick: Im objektorientierten Entwurf erfolgt eine Verfeinerung der Ereignisse und der Verantwortlichkeiten.

Methodisches Vorgehen

Unterscheide:

- **Konzeptuelles Modell** in objektorientierter Analyse
  - Aktoren, Entitätsklassen, Kontrollklassen, Grenzklassen
  - externe Ereignisse und Interaktionen
  - Systemverantwortlichkeiten

- **Architekturmodell** im objektorientierten Entwurf
  - Klassen, Pakete
  - interne Interaktionen
  - Klassenverantwortlichkeiten
  - Prä- und Postkonditionen für Methoden System

- **Ausführbares Modell** bei der Programmierung
  - ...

Methodisches Vorgehen

Eingabe: *Pflichtenheft.Anwendungsfallmodell, Pflichtenheft.Klassendiagramme, Pflichtenheft.Objektdiagramme*

Ausgabe: *Pflichtenheft.Sequenzdiagramme, Pflichtenheft.Kollaborationsdiagramme Methoden* in *Pflichtenheft.Klassendiagramme*

Änderung: *Pflichtenheft*

X = die Menge der Anwendungsfälle in *Pflichtenheft.Anwendungsfallmodell*
Ereigniskatalog = {}
While x **in** X
{
*Identifikation externer Ereignisse (*Benutzeraktionen*)*,
die in x erwähnt werden
Ereigniskatalog = Ereigniskatalog + neu identifizierte Ereignisse
}