Projekt Erdbebenfrühwarnung
im WiSe 2010/11

Entwicklung verteilter eingebetteter Systeme

Prof. Dr. Joachim Fischer
Dipl.-Inf. Ingmar Eveslage
Dipl.-Inf. Frank Kühnlenz

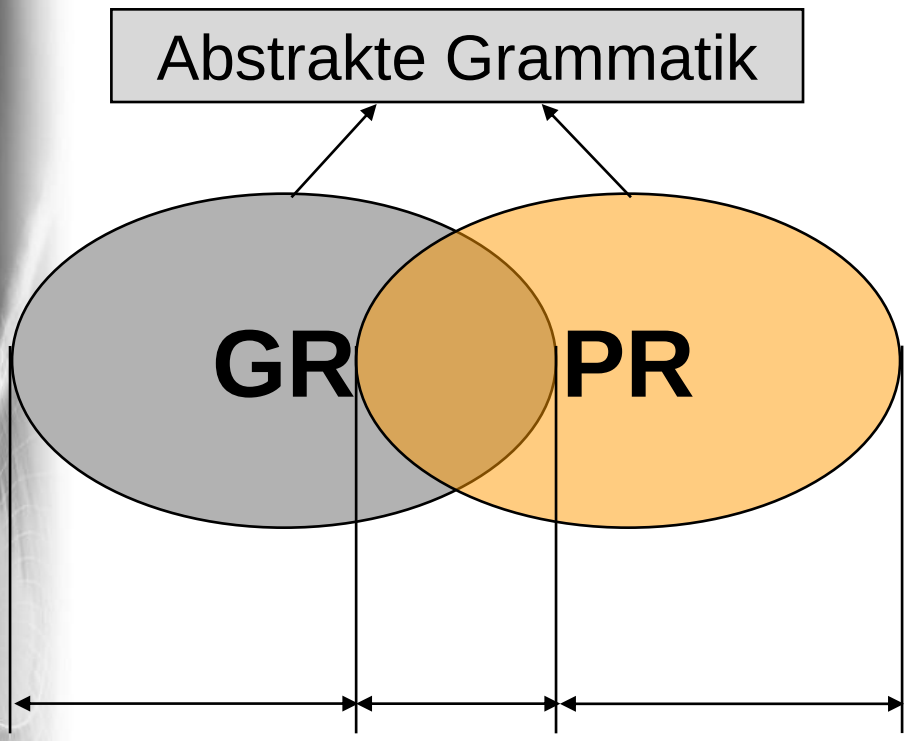
fischer|eveslage|kuehnlenz@informatik.hu-berlin.de

5. SDL als UML-Profil

1. ITU-Standard Z.100
2. UML und SDL-Zustandsmaschinen im Vergleich
3. Werkzeuge
4. SDL-Grundkonzepte
5. Musterbeispiel (in UML-Strukturen)
6. Struktur- und Verhaltensbeschreibung in SDL-RT

Repräsentationsformen

Z.100 (konkrete, abstrakte Syntax, statische u. dynamische Semantik



CIF

Common Interchange Format (Z.107)

UML Profile

(Z.109)

5. SDL

1. ITU-Standard Z.100
2. UML und SDL-Zustandsmaschinen im Vergleich
3. Werkzeuge
4. SDL-Grundkonzepte
5. Musterbeispiel (in UML-Strukturen)
6. Struktur- und Verhaltensbeschreibung in SDL-RT

Geklärte semantische Variationen in SDL

- Datentypen, Action-Syntax und Action-Semantik (C- Style),
  SDL erlaubt Werte- und Referenz-Semantik

- Beziehungen zwischen aktiven Klassen und Zustandsautomaten
  Agenten sind die Vereinigung von Aktiver Klasse und Zustandsautomat
  Mehrfachvererbung für Agenten sind ausgeschlossen

- Agenten besitzen systemweit 1-deutige Referenzen, deren Kenntnis initial nur lokal gegeben ist. *Referenzen werden zur Signaladressierung benutzt.*

- *Keine Aussage zum quantitativen Zeitverhalten der Zustandsübergänge. Übertragungskanäle verbrauchen keine oder nur geringe Zeit*

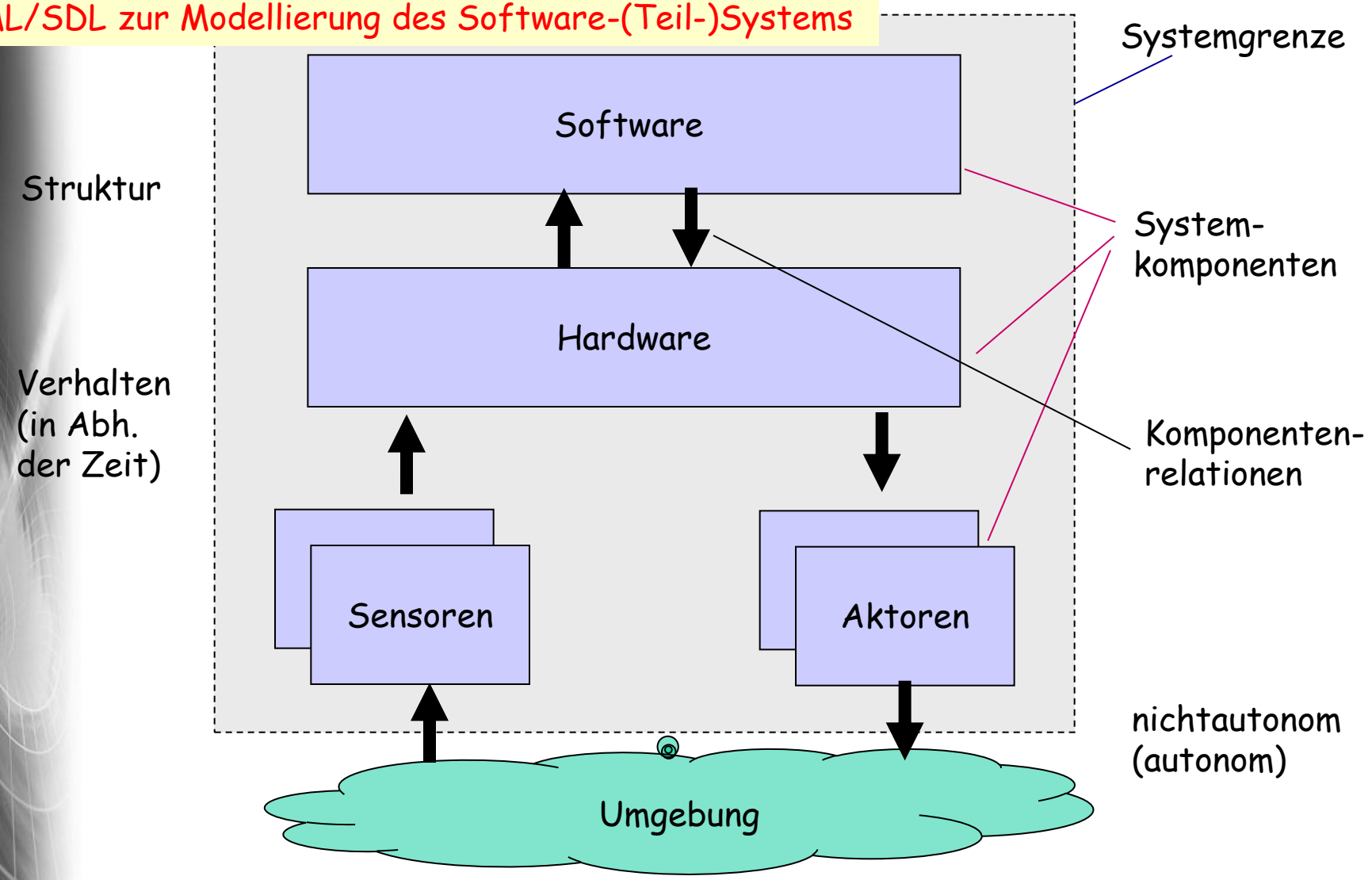
- Ereignisverwaltung (Signalempfangspuffer) ist präzisiert

- *Remote-Procedure-Call ist über Ersetzungsmodell präzisiert*

- *Systeminstanzen lassen sich definieren*

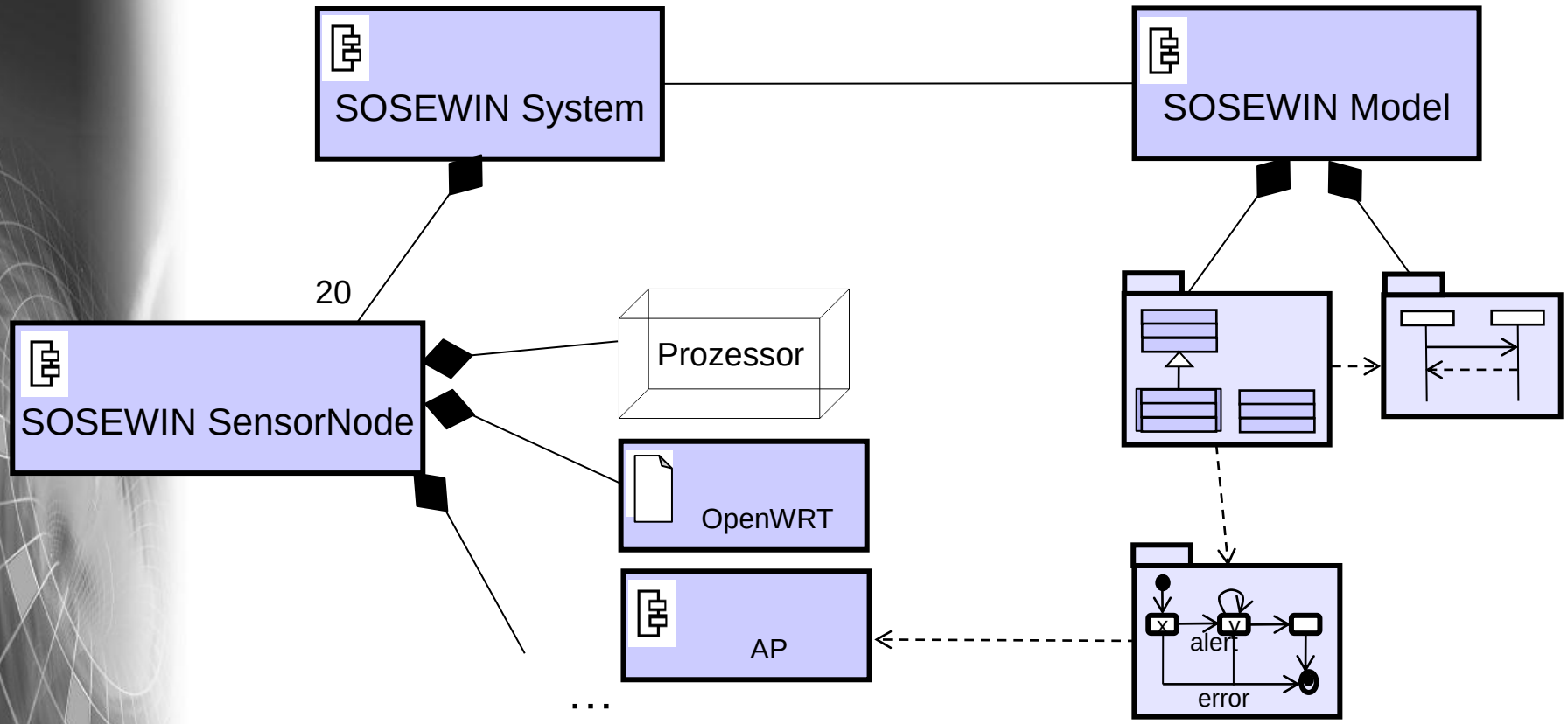
Eingebettete u. reaktive Systeme

UML/SDL zur Modellierung des Software-(Teil-)Systems



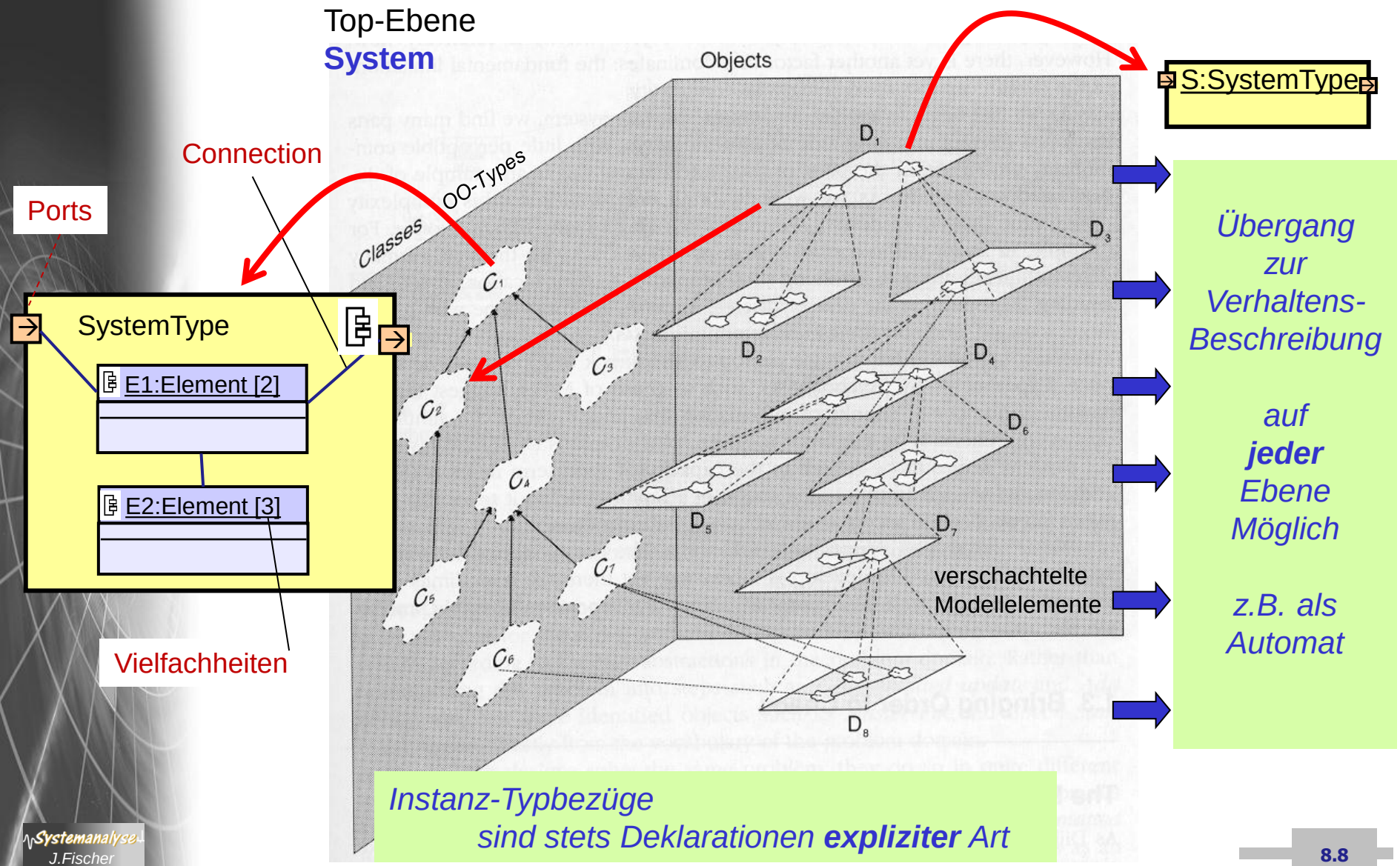
SOSEWIN in UML-Darstellung (präzisiert)

instanziierbar

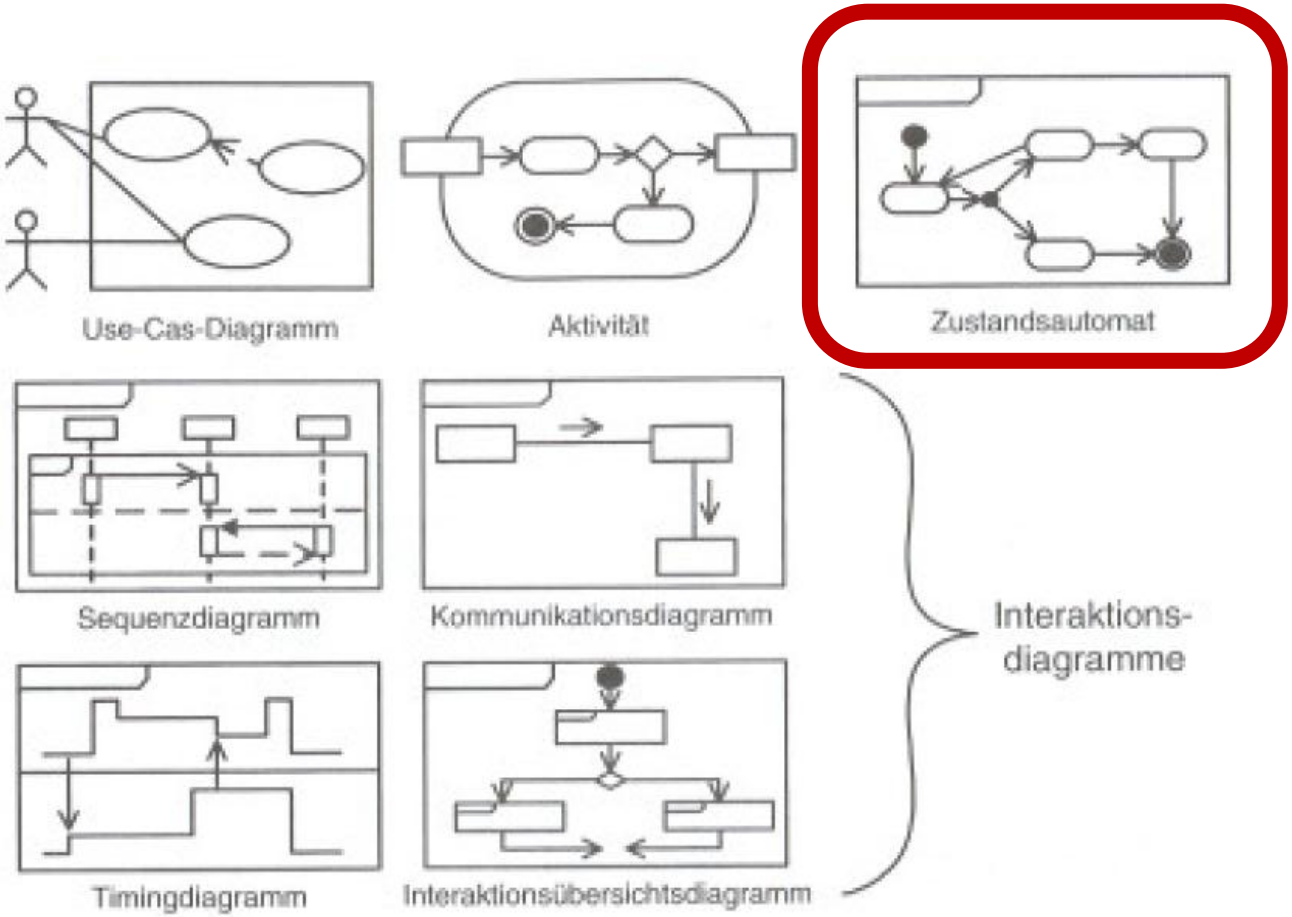


Verwendung von UML-Komponente zur Darstellung von: SystemTyp

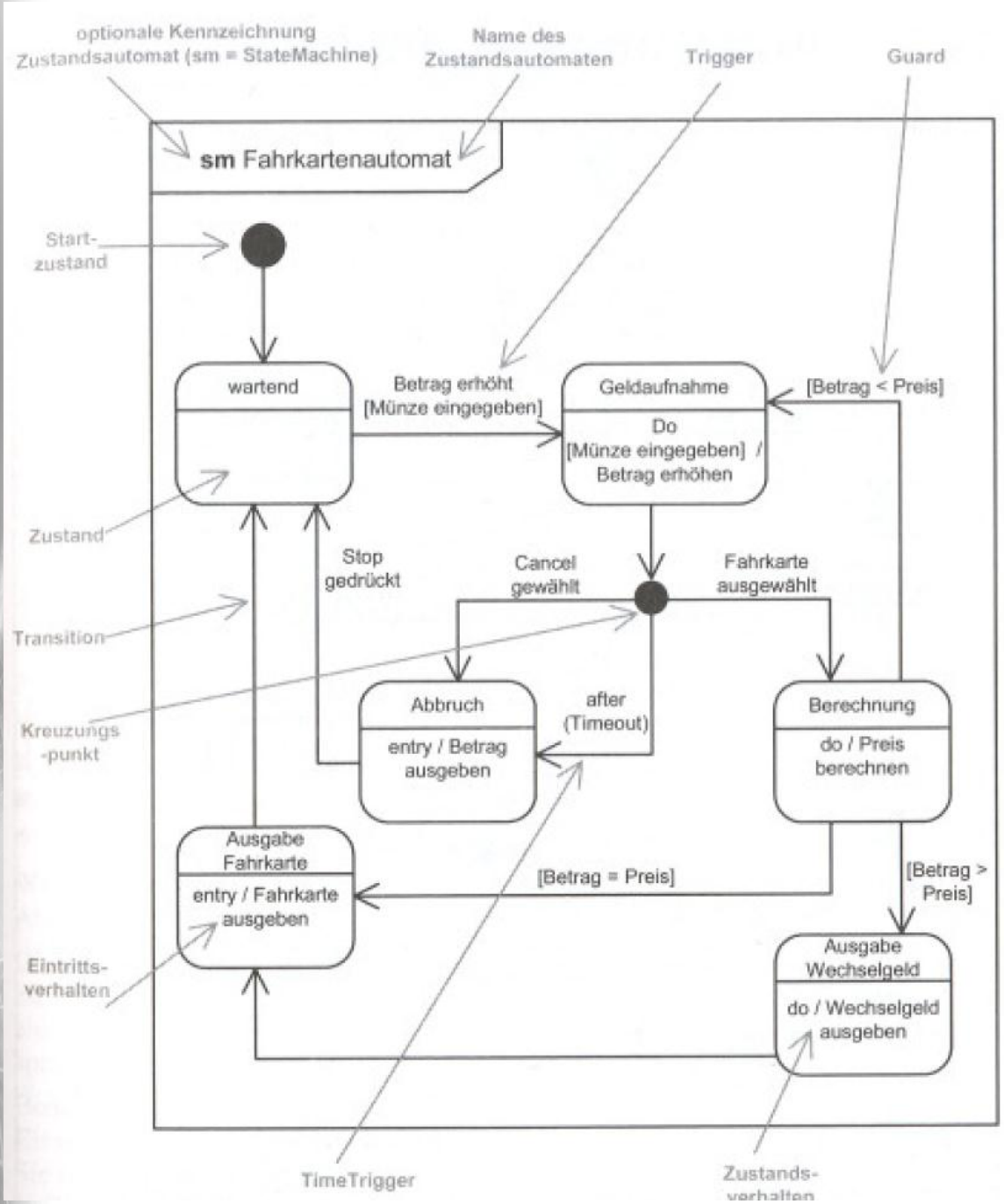
Die strukturelle Systemsicht in UML



Verhaltensdiagramme in UML



Beschreibungs-elemente



- Zustand
- Trigger
- Time-Trigger
- Guard
- Entscheidung
- Eintrittsverhalten
- Zustandsverhalten
- Austrittsverhalten

Korrekte Spezifikation?
Quelle: „UML-Glasklar“)

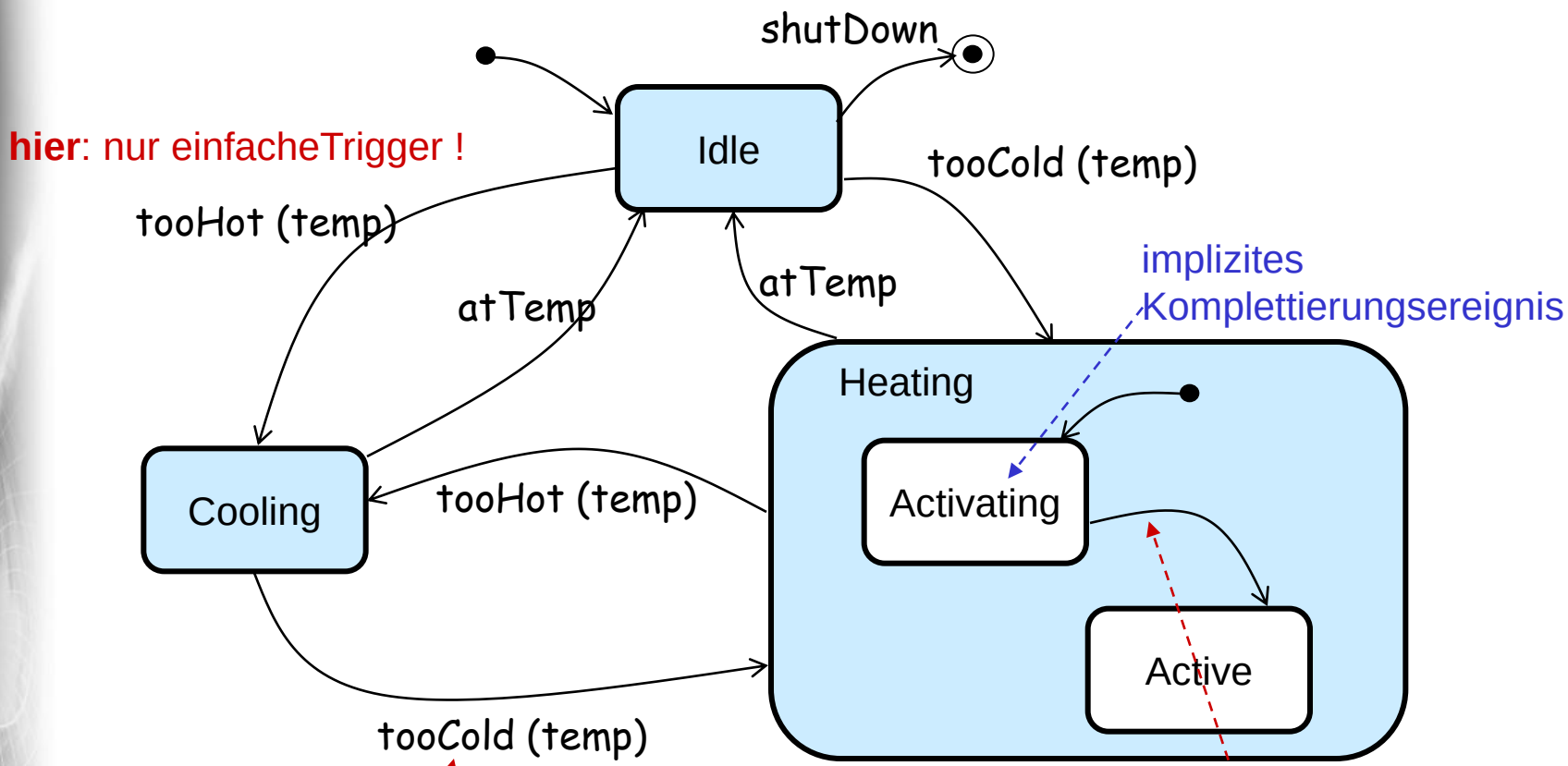
UML-Zustandsdefinition

Bestandteile

- Name
  (optional), kein Name, dann anonymer Zustand
- Entry-/ Exit- Aktionen bei Eintritt bzw. Austritt
- interne Do-Aktivität (durch externes Ereignis unterbrechbar)
- Substates: verschachtelte Struktur
  - sequentiell aktive
  - gleichzeitig aktive
- interne Transitionen (ohne Wechsel des Zustandes)
- Liste von Ereignissen, die nicht in diesem Zustand behandelt werden (verzögerte Ereignisse)

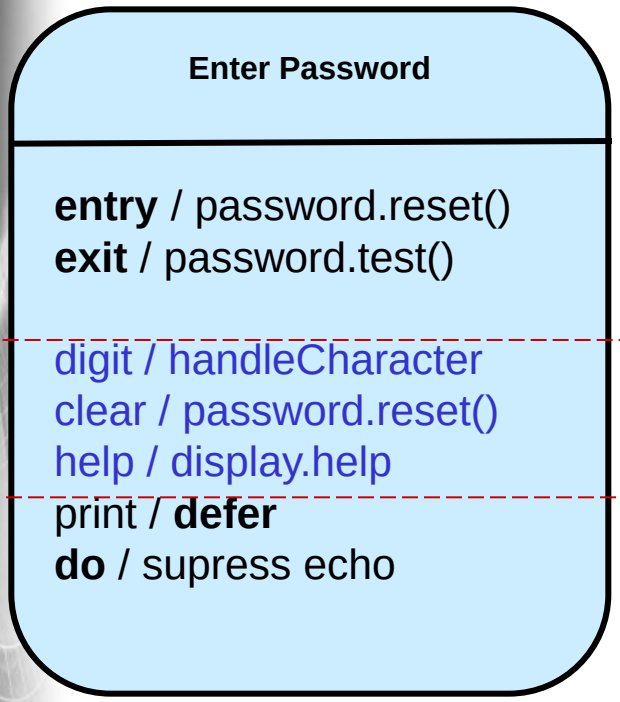
Ereignisse, Trigger, Transitionen

Unterscheidung zwischen externen und internen Transitionen



- Ereignisse lösen Transitionen aus, wenn dafür Trigger vorhanden sind
- Trigger können explizit angegeben sein, oder sie ergeben sich implizit

Zustände, Interne Transitionen, Unterbrechbarkeit von Aktivitäten



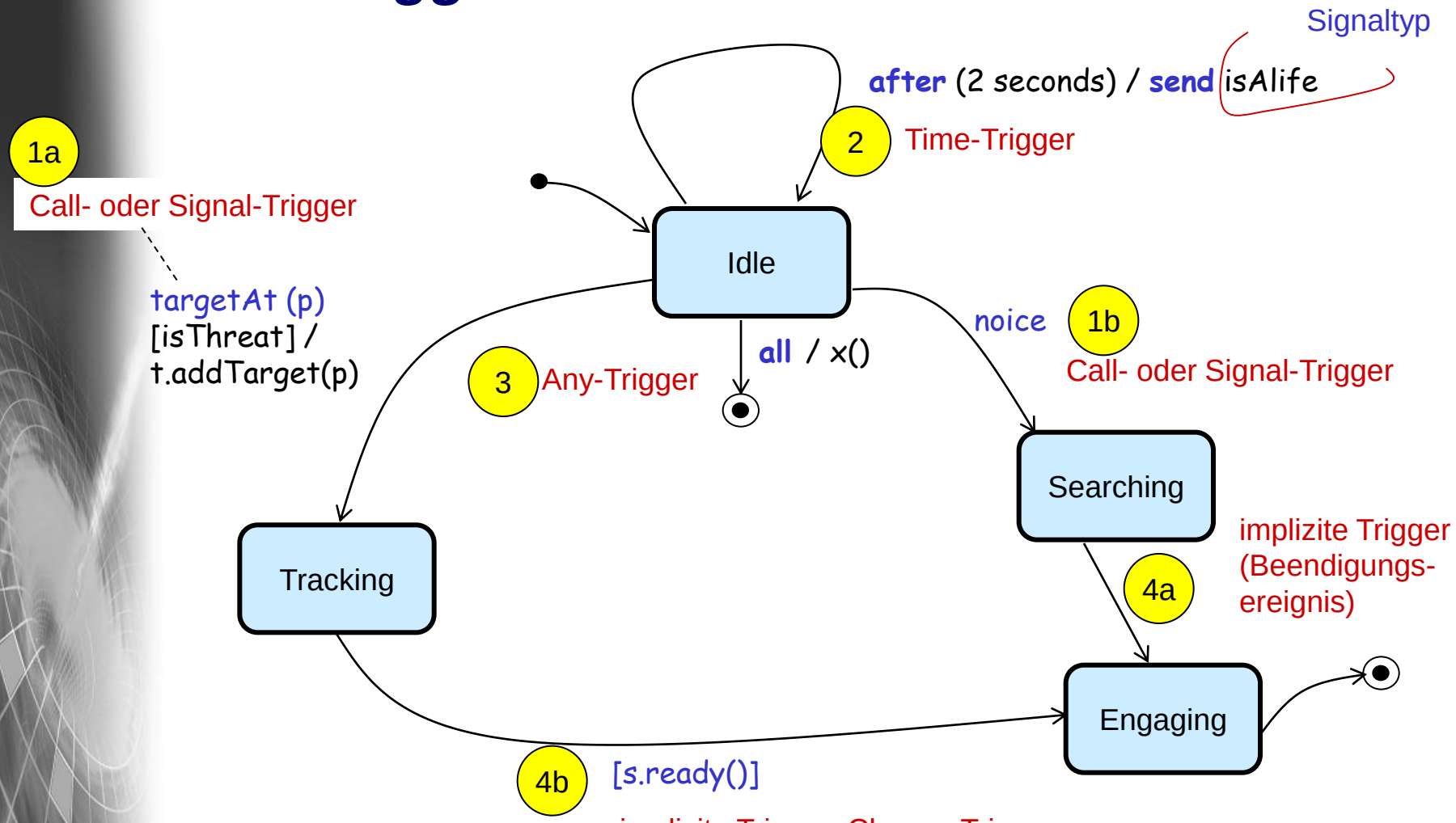
Bestandteile

- Name
  (optional), kein Name, dann anonymer Zustand
- Entry-/ Exit- Aktivität bei Eintritt bzw. Austritt
- interne Transitionen (ohne Wechsel des Zustandes)
- Liste von Ereignissen, die nicht in diesem Zustand behandelt werden (verzögerte Ereignisse)
- interne Do-Aktivität
- Substates: verschachtelte Struktur
  - sequentiell aktive
  - gleichzeitig aktive

explizite und implizite Ereignisse zum Abbruch der internen Do-Aktivität

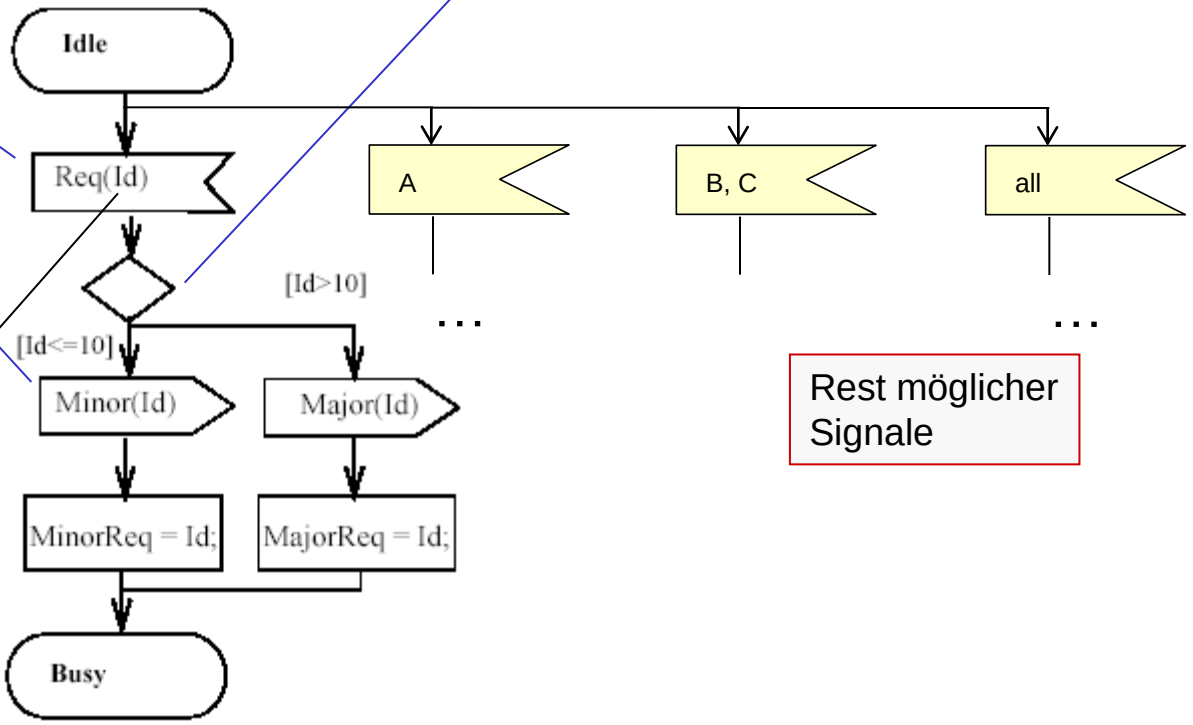
Aktivitäten:= als Folge Aktionen
bei Auszeichnung von elementaren **nicht unterbrechbaren** Aktionen

UML-Trigger-Arten



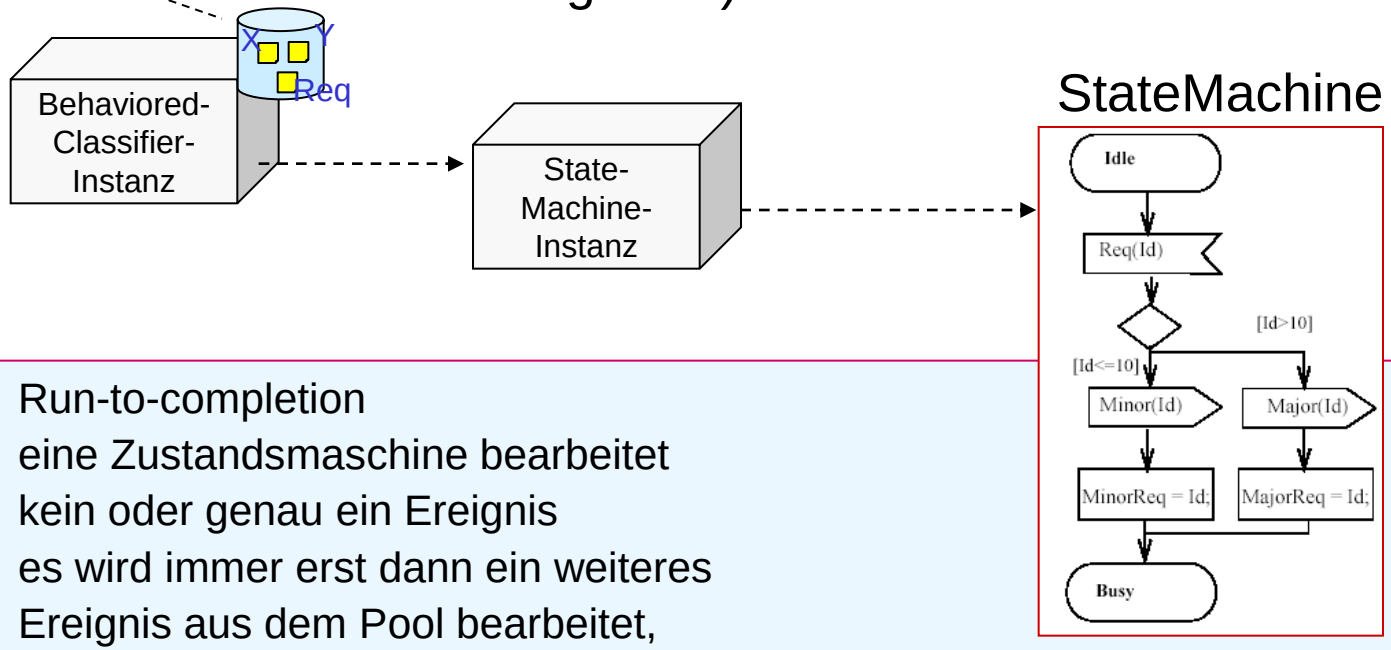
SDL-ähnliche Notation

- Empfangen eines Signals
- Senden eines Signals
- eine Aktionssequenz



Run-to-Completion-Semantik

Ereignispool (ankommende Signale, eingehende Call-Requests, Zustandsereignisse)



- Run-to-completion
- eine Zustandsmaschine bearbeitet kein oder genau ein Ereignis
- es wird immer erst dann ein weiteres Ereignis aus dem Pool bearbeitet, wenn das vorhergehende Ereignis vollständig abgearbeitet wurde
- Reihenfolge der Abarbeitung von Ereignissen ist nicht definiert (aus der Menge feuerbarer Transitionen wird eine nichtdeterminiert ausgewählt)
  → erlaubt den Einsatz verschiedener Priorisierungsverfahren bei UML-Profilen

Run-to-Completion-Semantik (1)

- Run-to-completion
  sichert die Zustandsübergangsfunktion, keine Concurrency-Probleme bei Ereignis-Bearbeitung

- Abarbeitung von Ereignissen resultiert in der Feuererlaubnis (enable) von keiner, einer oder mehreren Transitionen (bei parallelen Ausführungszweigen)
  - falls keine Transition feuerbar ist und es kein zu verzögerndes Ereignis vorliegt wird das Ereignis gelöscht und der Run-to-completion-Schritt ist beendet

- Während einer Transition können Aktionen ausgeführt werden
  - falls eine solche Aktion der Ruf einer synchronen Operation an einem anderen Objekt ist,
    so wird die Transition erst dann beendet, wenn die Zustandsmaschine des gerufenen Objektes ihrerseits den Run-to-completion-Schritt abgeschlossen hat

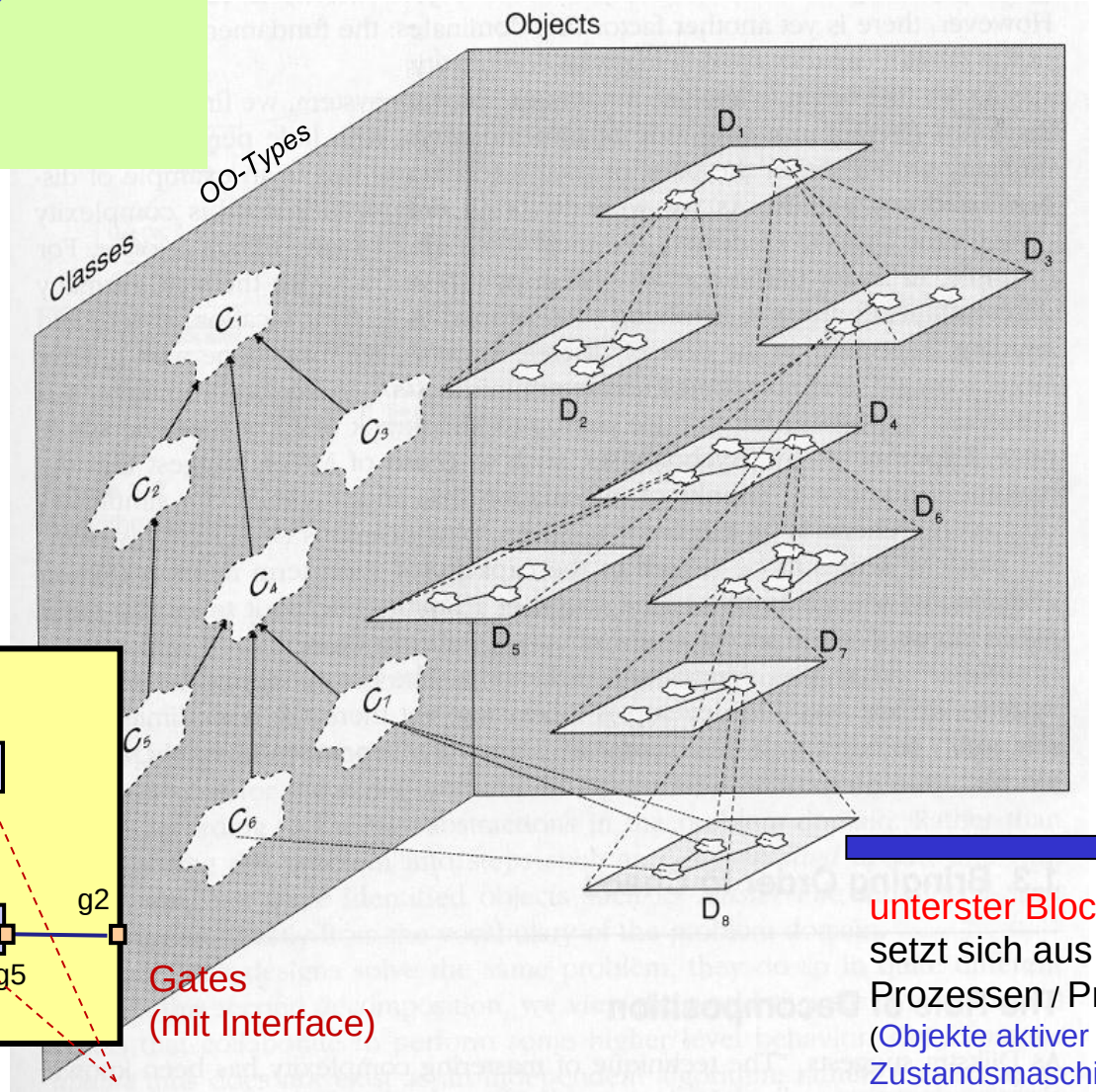
Run-to-Completion-Semantik (2)

Ergänzung

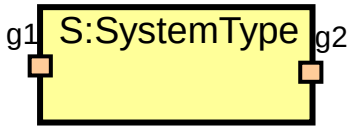
- Run-to-completion-Semantik und Thread-Semantik
  - Run-to-completion wird von aktiven Objekten (Instanzen von Active Classes) in ihren eigenen Threads ausgeführt
  - Innerhalb dieser Threads muss einfache run-to-completion implementiert werden, jedoch können Threads zwischendurch wiederum wechselseitig suspendiert werden
  - Run-to-completion ist unabhängig vom Thread-Scheduling !

Die strukturelle Systemsicht in SDL

Instanz-Typbezüge können
- impliziter und
- expliziter Art
sein



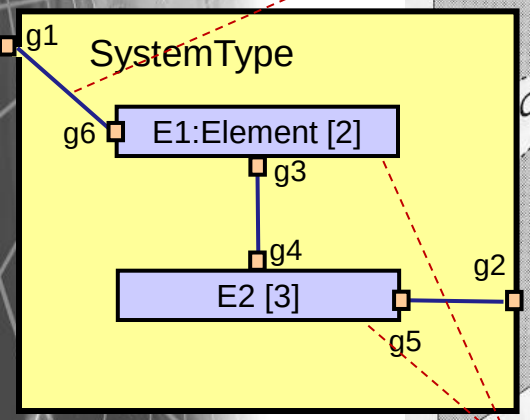
oberster „Block“
System



Übergang zur Verhaltens-Beschreibung

*nur auf **unterster** Ebene als Automat möglich*

Kanäle

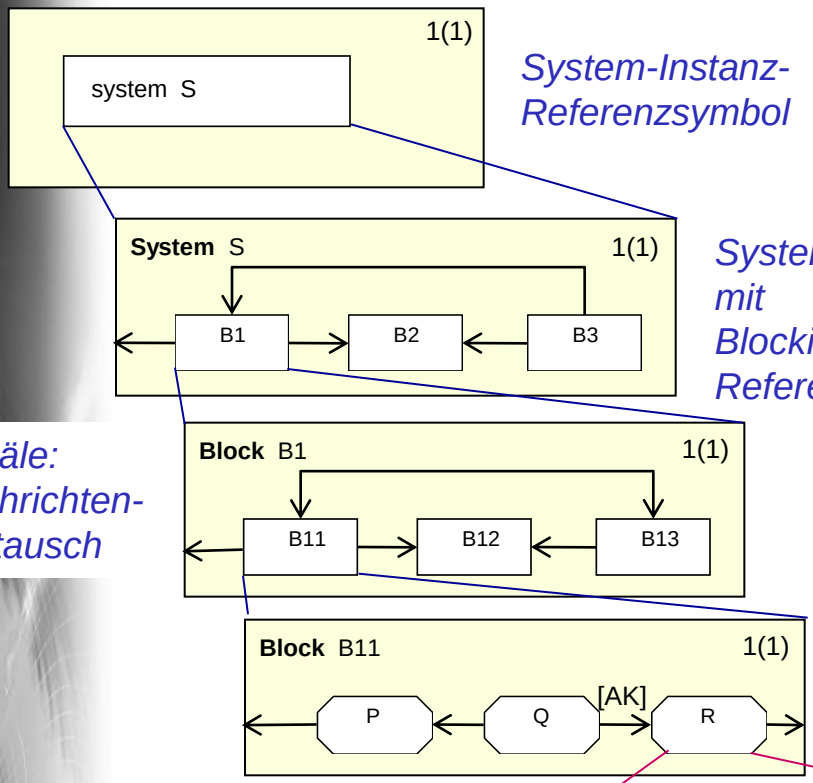


Gates
(mit Interface)

Blockinstanzmengen

unterster Block
setzt sich aus
Prozessen / ProzessInstanzmengen
(Objekte aktiver Klassen + Zustandsmaschinen)
zusammen

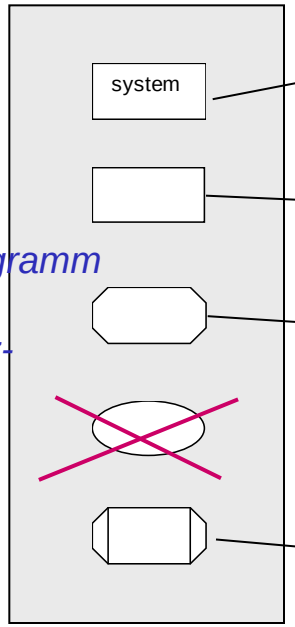
SDL/GR-Diagrammkomposition



System-Instanz-Referenzsymbol

System-Diagramm mit Blockinstanz-Referenzen

Kanäle: Nachrichten-austausch



Referenzsymbole

System-Instanz

Block-Instanz(menge)

Prozess-Instanz(menge)
(Default-Kardinailität=1)

Service-Instanz

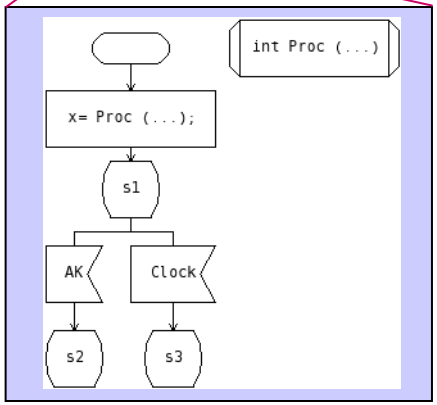
nicht in SDL-RT

Prozedur

SDL-Editor-Anforderung:
Diagramm- und Seitennavigation

Prozess als Zustandsautomat

- *kommunizieren per asynchronen Nachrichtenaustausch (impliziter Empfangspuffer)*
- *können andere Prozesse erzeugen*
- *Verhalten: endlich,unendlich*



- *lokale Variablen (auch Assoziationsenden)*
- *lokale Datentypen*
- *lokale Prozeduren/Funktionen*
- *lokale Zustände*
- *Zustandsübergänge als Ereignistrigger*

Systemsichtweisen von SDL

- **strukturelle Sichtweise**
  - Instanzsicht

    Beschreibung der Konfiguration eines Systems (system) bestehend aus funktionalen Einheiten (block) und ihren Relationen (channel) bei Identifikation der Systemgrenzen
  - Typsicht (falls für Instanzen/Instanzmengen Typen festgelegt sind)
  - Pakete (package) zur bequemen Wiederverwendung von Typen

- **verhaltensorientierte Sichtweise**
  - Verhalten einer funktionalen Einheit wird durch kommunizierende nebenläufig agierende Zustandsautomaten erbracht (agent/process)
  - Prozessverhalten: zeitdiskret
  - Strukturierungsmöglichkeiten von Verhaltensbeschreibungskonstrukten (procedure, service)

5. SDL

1. ITU-Standard Z.100
2. UML und SDL-Zustandsmaschinen im Vergleich
3. Werkzeuge
4. SDL-Grundkonzepte
5. Musterbeispiel (in UML-Strukturen)
6. Struktur- und Verhaltensbeschreibung in SDL

Überblick

- TAU-Werkzeug, IBM Rational
  (ursprünglich schwedische Firma TeleLogic)

- Cinderella (Standard-SDL, SDL 96, dänische Firma)
  mit Codegenerator der HU Berlin

- SDL-2000 ASM-Interpreter
  (MicroSoft, HU Berlin, Uni Kaiserslautern)

- PragmaDev Developer Studio (pragmatische SDL-Variante)
  - Eingeschränktes SDL
  - C/C++ Actionsprache
  - Kombination mit UML: KlassenDG, UseCaseDG, SequenceDG, DeploymentDG

  mit modifiziertem Codegenerator der HU Berlin

- SDL-Werkzeug auf UML-Basis nur rudimentär
  (HU Berlin)

5. SDL

1. ITU-Standard Z.100
2. UML und SDL-Zustandsmaschinen im Vergleich
3. Werkzeuge
4. SDL-Grundkonzepte
5. Musterbeispiel (in UML-Strukturen)
6. Struktur- und Verhaltensbeschreibung in SDL

SDL-Basis (Wdh.)

Agent= aktive Klasse mit Process-Beschreibung
- Classifier-Eigenschaft

- ein SDL-System besteht zur Laufzeit aus einer Menge von kommunizierenden Zustandsmaschinen
  definiert durch je einen Repräsentanten der festgelegten Process (Agenten)- Instanzmengen

  die in ihrer Wechselwirkung untereinander und mit der Umgebung des Systems das Verhalten erbringen

- die Wechselwirkungen werden über einen **asynchronen** Nachrichtenaustausch realisiert
  - Sender und Empfänger sind damit entkoppelt

- jede Prozessinstanz besitzt (genau) einen Empfangspuffer zur Speicherung ankommender Nachrichten
  dieser ist idealerweise a priori unbeschränkt
  - keine Blockierung des Senders aufgrund eines vollen Puffers

SDL-Laufzeitsystem