Modellbasierte Entwicklung von Technologien für selbstorganisierende dezentrale Informations-systeme im Katastrophen-management

METRIK

Projekt Erdbebenfrühwarnung
im SoSe 2011

HUMBOLDT-UNIVERSITÄT ZU BERLIN

Entwicklung verteilter echtzeitfähiger Sensorsysteme

Joachim Fischer
Klaus Ahrens
Ingmar Eveslage

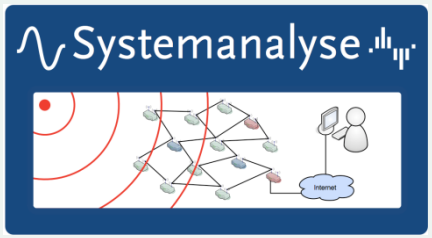
fischer|ahrens|eveslage@informatik.hu-berlin.de

SAFER
Seismic eArly warning For EuRope

EDIM

SOSEWIN-extented

GFZ
Helmholtz-Zentrum
POTSDAM

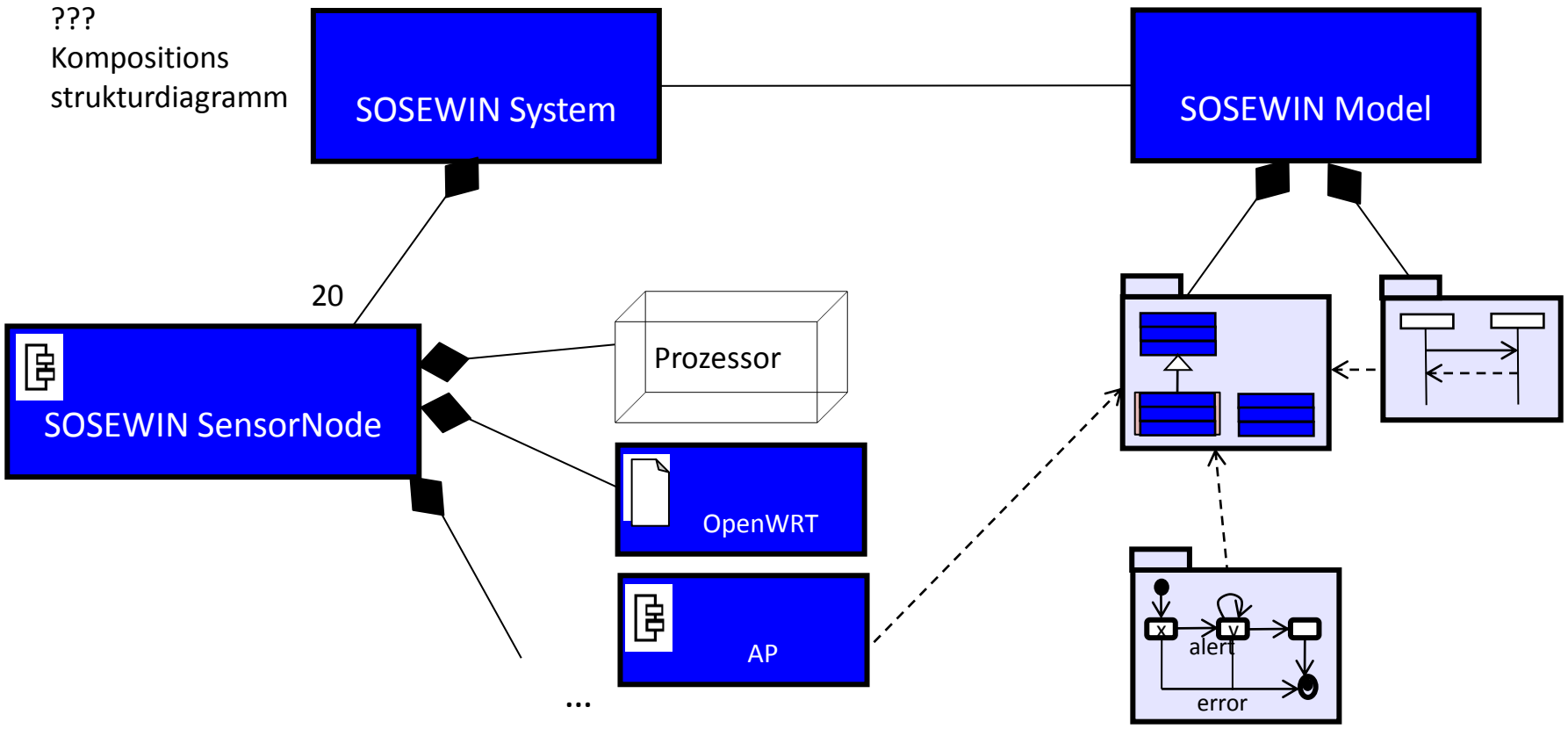


4. SDL als UML-Ersatz

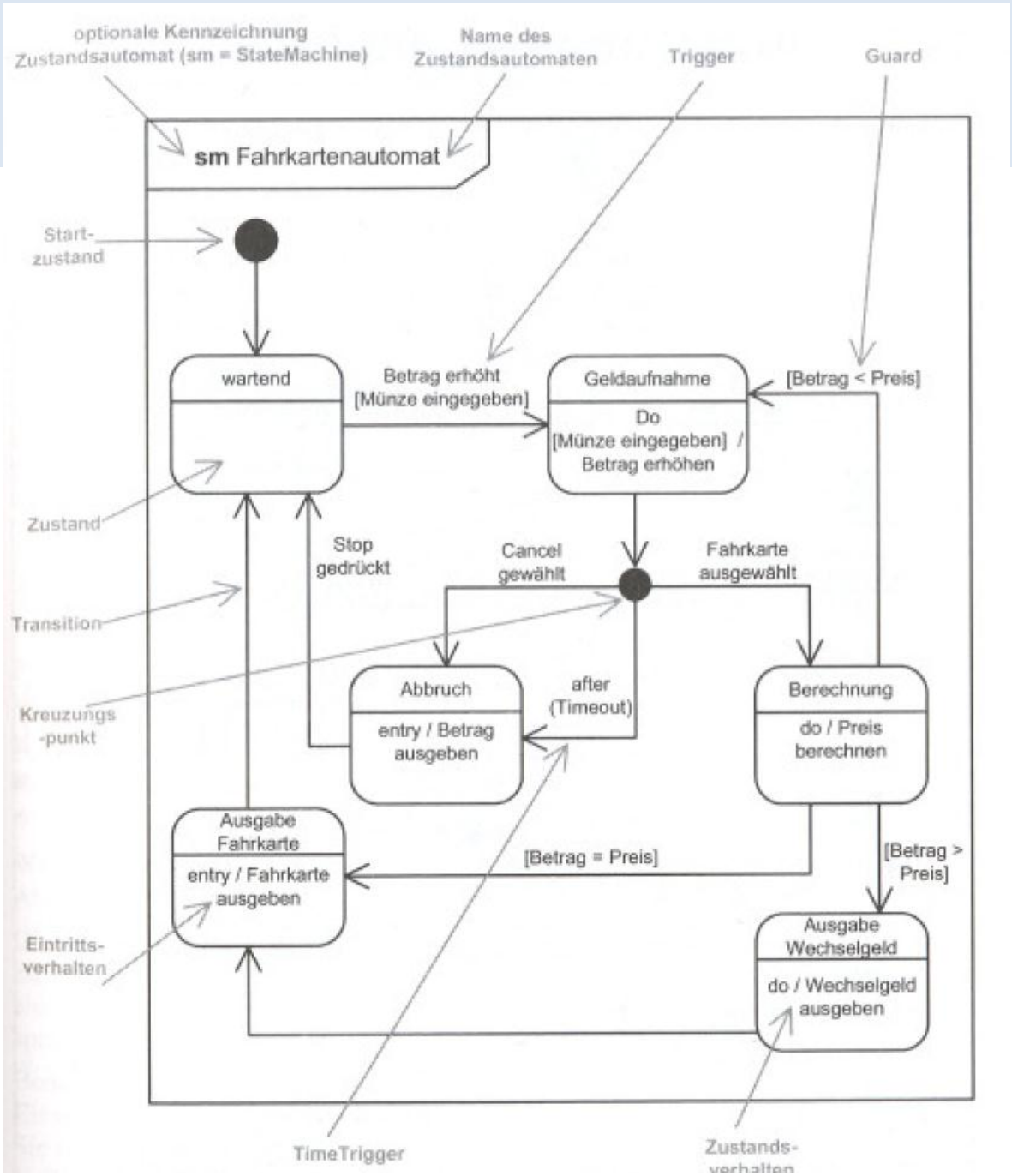
1. UML und SDL-Zustandsmaschinen im Vergleich
2. Werkzeuge
3. ITU-Standard Z.100
4. SDL-Grundkonzepte
5. Musterbeispiel (in UML-Strukturen)
6. Struktur- und Verhaltensbeschreibung in SDL-RT

SOSEWIN in UML-Darstellung

(vage Vorstellung)



Beschreibungs-elemente



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

UML-Ereignisse

1. Signal (-ankunft)
2. Op-Call-Request
3. Zeitereignis (absolute, relative Zeitangabe)
4. Zustandsereignis (mit Bedingung)
5. Implizite Ereignisse (Beendigung einer Aktivität)ein Beendigungsereignis wird vor allen anderen Ereignissen ausgeführt

- Signal-Parameter werden als Attribute eines Signal-Classifiers repräsentiert

- Senden von Signalen/Aufruf von Operationen sind asynchrone Vorgänge, an denen mindestens zwei Objekte beteiligt sind
  Sonderfälle:
  - Broadcast (Empfangsobjekt ist ein Aggregationsobjekt (System/Teilsystem)
  - Multicast (Empfänger ist eine Kollektion von Objekten)

UML-Zustandsdefinition

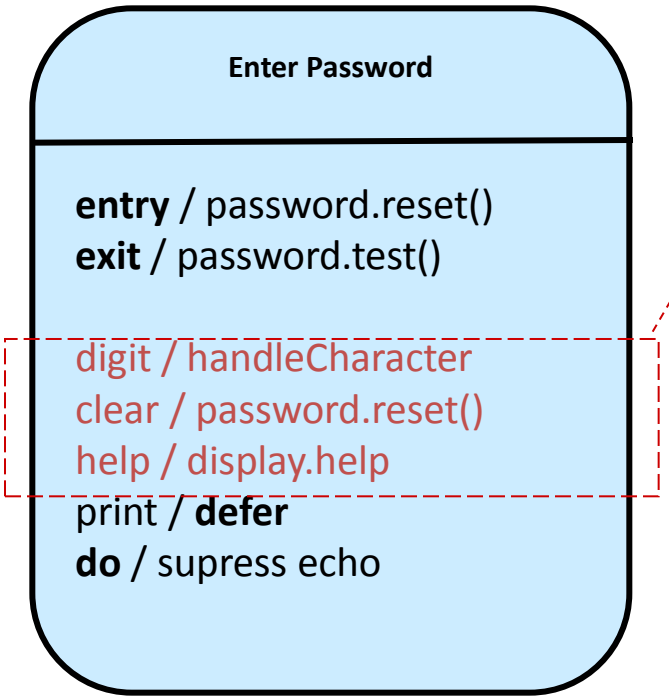
Bestandteile/Charakteristik

- Name
  (optional), kein Name, dann anonymer Zustand
- Entry-/ Exit- Aktionen bei Eintritt bzw. Austritt
- interne Do-Aktivität
  durch externes Ereignis unterbrechbar,
  d.h. per Signal-, Call-, Time-, State-Event

ohne Parameter

- Substates/Unterzustände: verschachtelte Struktur
  - sequentiell aktive
  - gleichzeitig aktive
- interne Transitionen (ohne Wechsel des Zustandes)
  durch externes Ereignis unterbrechbar, d.h. per Signal-, Call-, Time-, State-Event
- Liste von Ereignissen, die nicht in diesem Zustand behandelt werden
  (verzögerte Ereignisse)

Zustände, Interne Transitionen, Unterbrechbarkeit von Aktivitäten



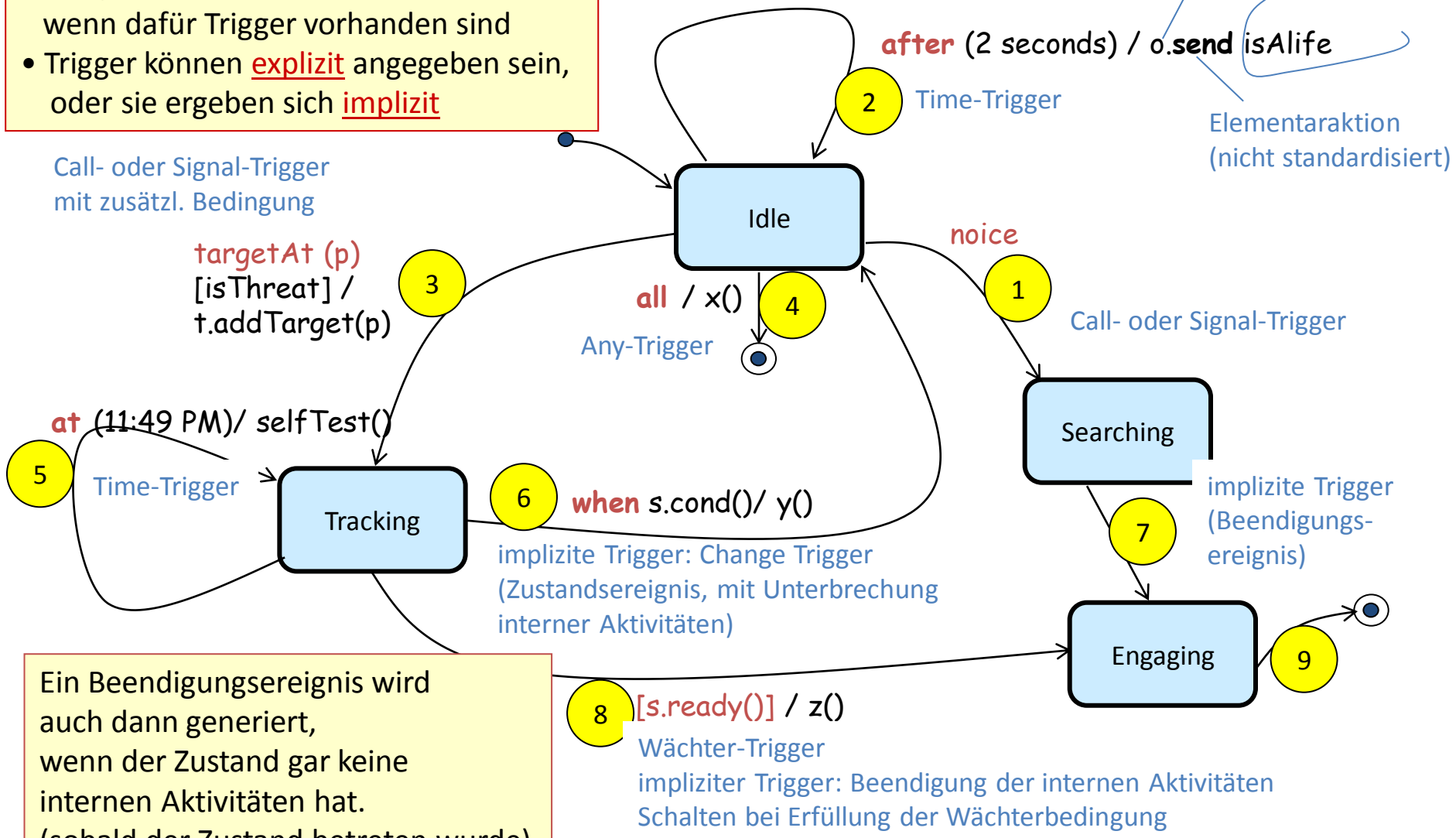
Bestandteile

- 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

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

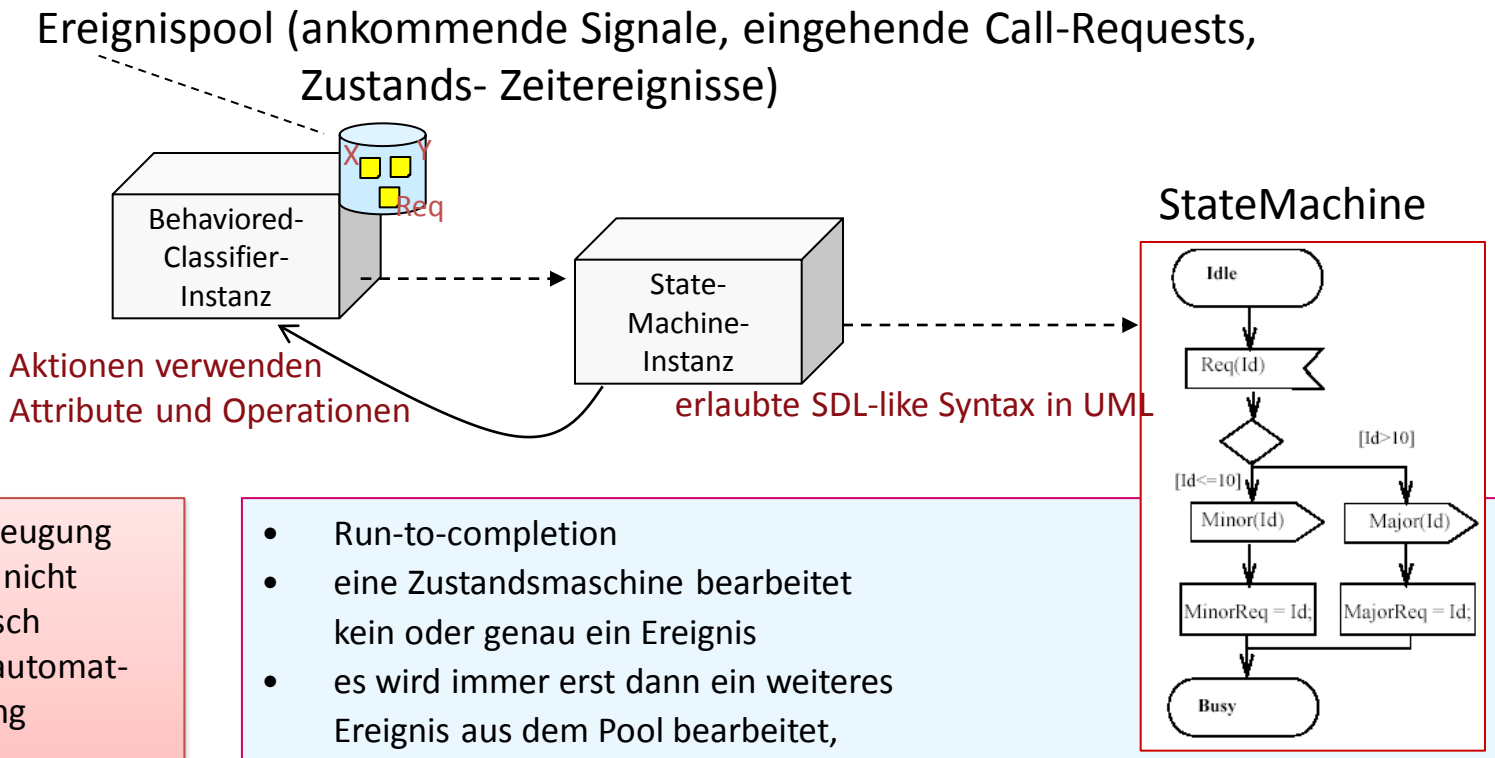
UML-Trigger-Arten

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



Ein Beendigungsereignis wird auch dann generiert, wenn der Zustand gar keine internen Aktivitäten hat. (sobald der Zustand betreten wurde)

Run-to-Completion-Semantik (1)



Objekterzeugung bedeutet nicht automatisch Zustandsautomat-Aktivierung

OFFEN:
Wie können beide Aktionen verbunden werden?

- 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

Probleme in UML

- 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
- Adressierung von Signalen **offen**
- Semantik von Konnektoren **offen**
- Mehrfachvererbung ist für Zustandsautomaten **nicht** definiert

Run-to-Completion-Semantik (2)

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

- Abarbeitung von Ereignissen resultiert in der Feuererlaubnis 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

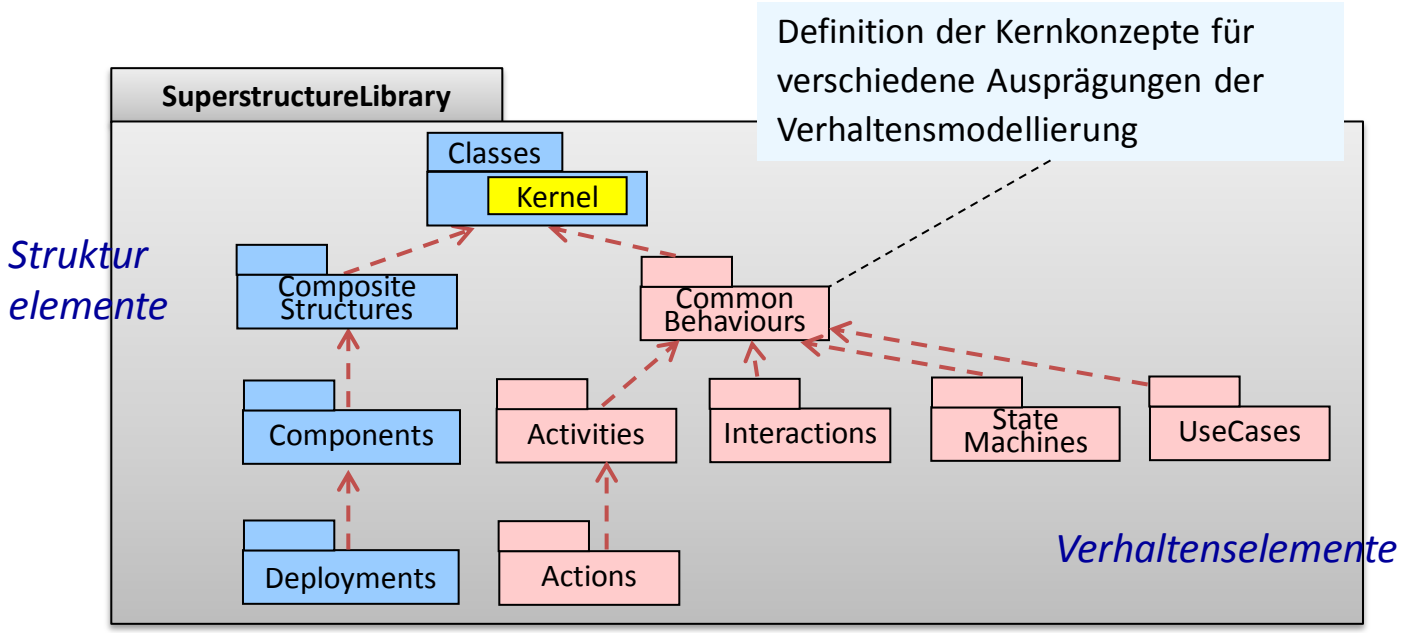
- Während einer Transition können Aktionen ausgeführt werden
  - falls eine solche Aktion der Ruf einer synchronen Operation an einem anderen Objekt ist,
    so wird die Transition erst dann beendet, wenn die Zustandsmaschine des gerufenen Objektes ihrerseits den Run-to-completion-Schritt abgeschlossen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

Run-to-Completion-Semantik (3)

- 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 !

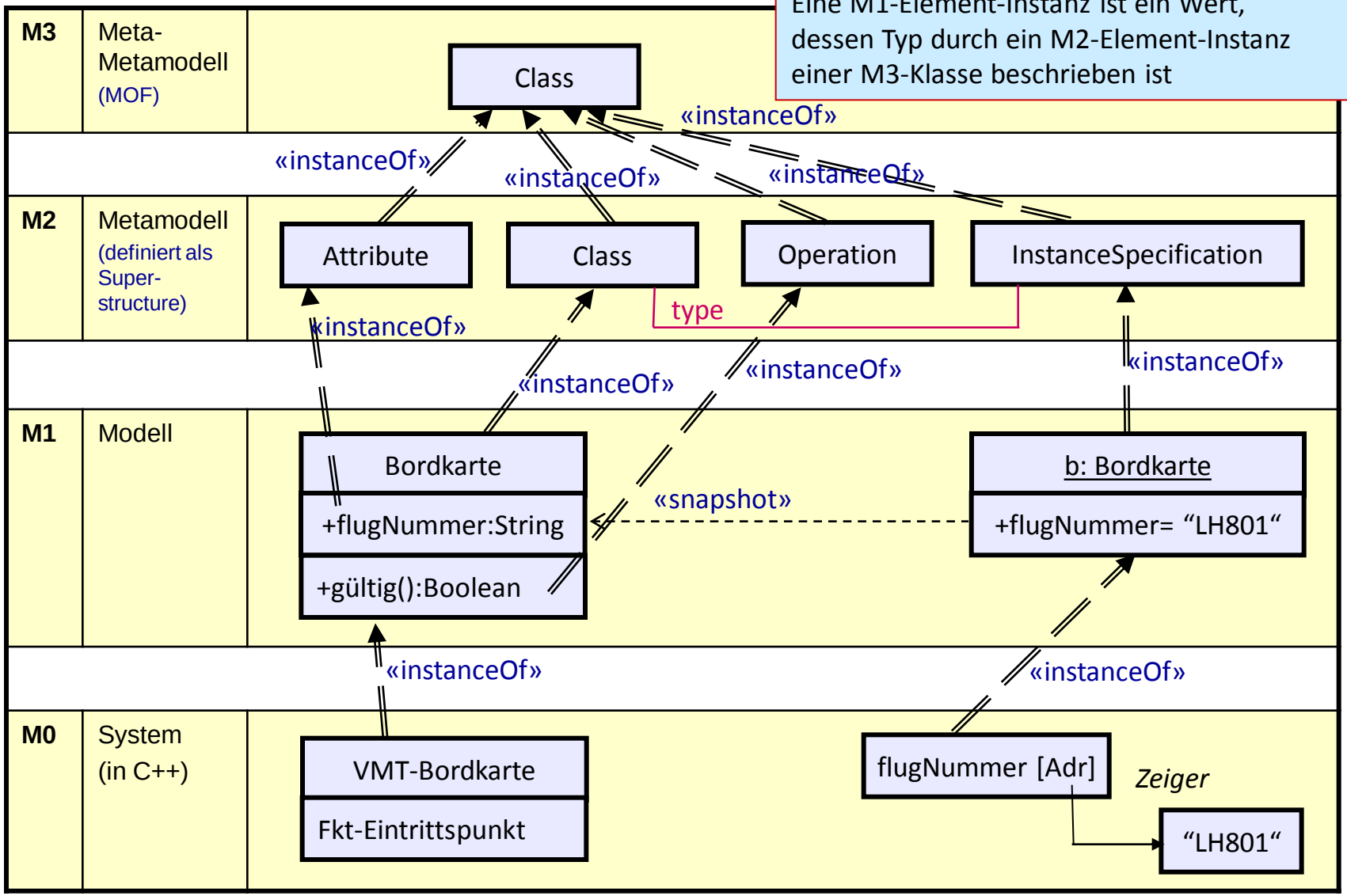
UML-Metamodell

- UML-Metamodell ~ definiert als Superstructure
  besteht aus weiteren Unterpaketen



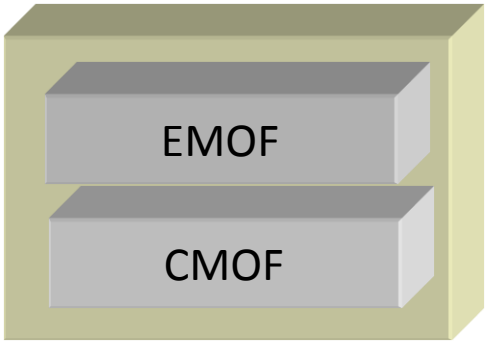
Die UML-Abstraktionsebenen, definiert durch MOF

MOF-Instanziierungsemantik
Eine M1-Element-Instanz ist ein Wert, dessen Typ durch ein M2-Element-Instanz einer M3-Klasse beschrieben ist

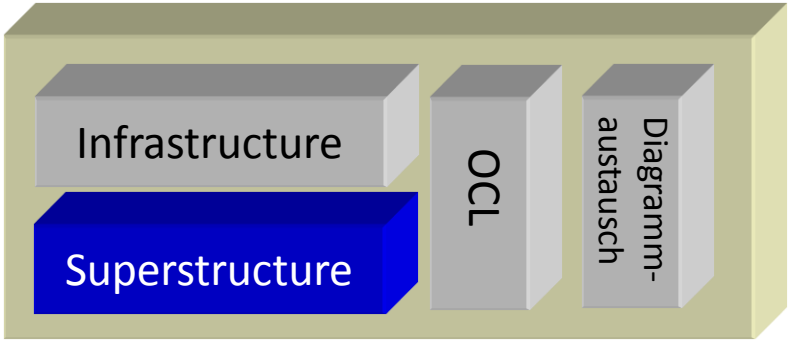


Standards und ihre Zusammenhänge

Definition von MOF



Definition von UML als MOF-basiertes Metamodells



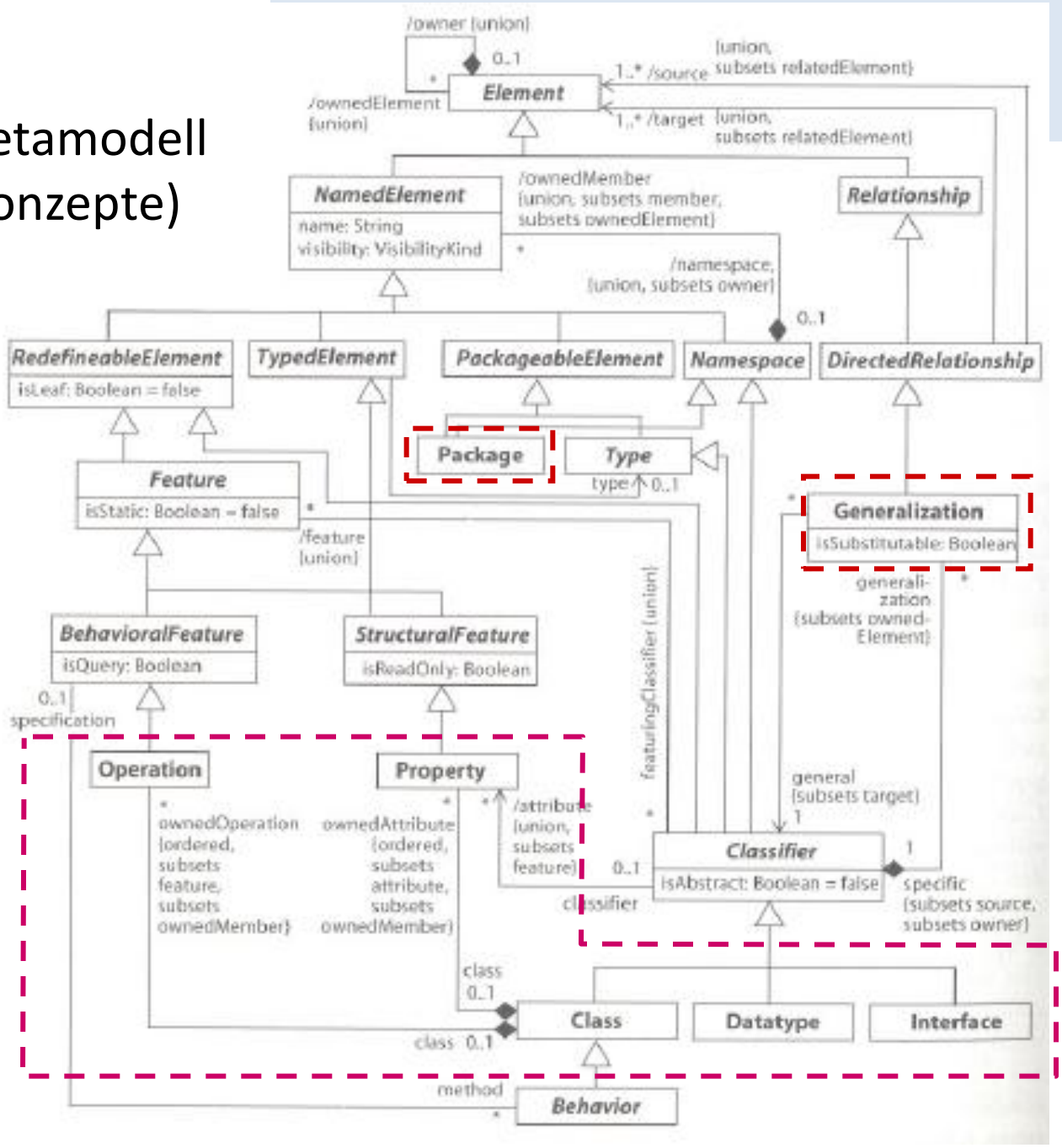
Infrastructure: kleiner UML-Sprachkern wird an MOF ausgerichtet

- liefert die Konzepte für das Meta-Metamodell (MOF)
  - eine Teilmenge der Infrastructure (Core) ist mit MOF identisch
- alle Konzepte der Infrastructure sind entweder Core-Konzepte oder Erweiterungen davon

Spezialisierung
Festlegung semantischer Variationspunkte

UML-Metamodell (Kernkonzepte)

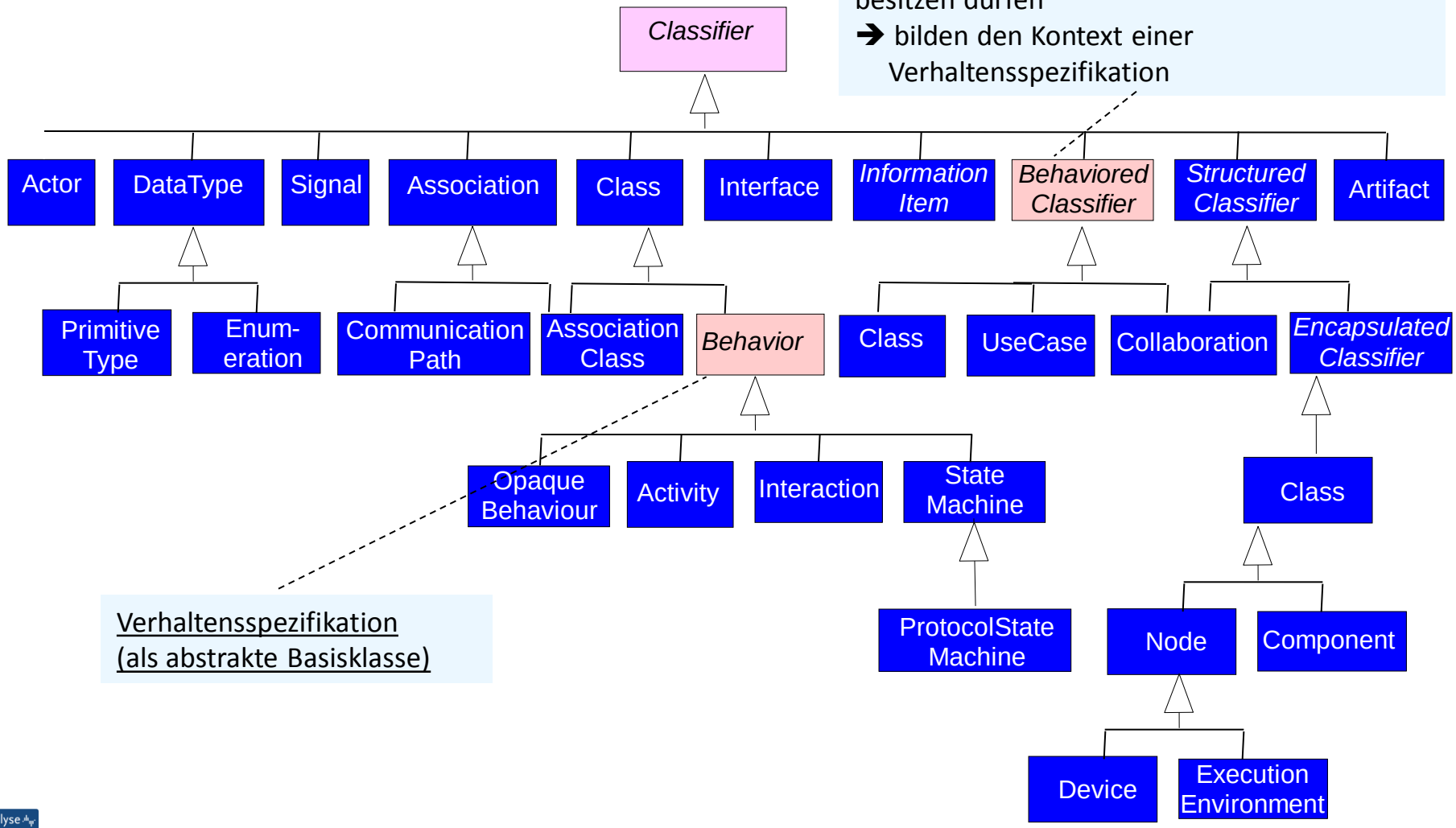
Konkrete Metaklassen



Superstructure: Übersicht der Classifier

… bei Paketauflösung

verhaltensspezifischer Classifier:
beschreibt alle Classifier, die eine Verhaltensspezifikation besitzen dürfen
➔ bilden den Kontext einer Verhaltensspezifikation



Verhaltensspezifikation
(als abstrakte Basisklasse)

Zwei Varianten in UML

- SM = StateMachine

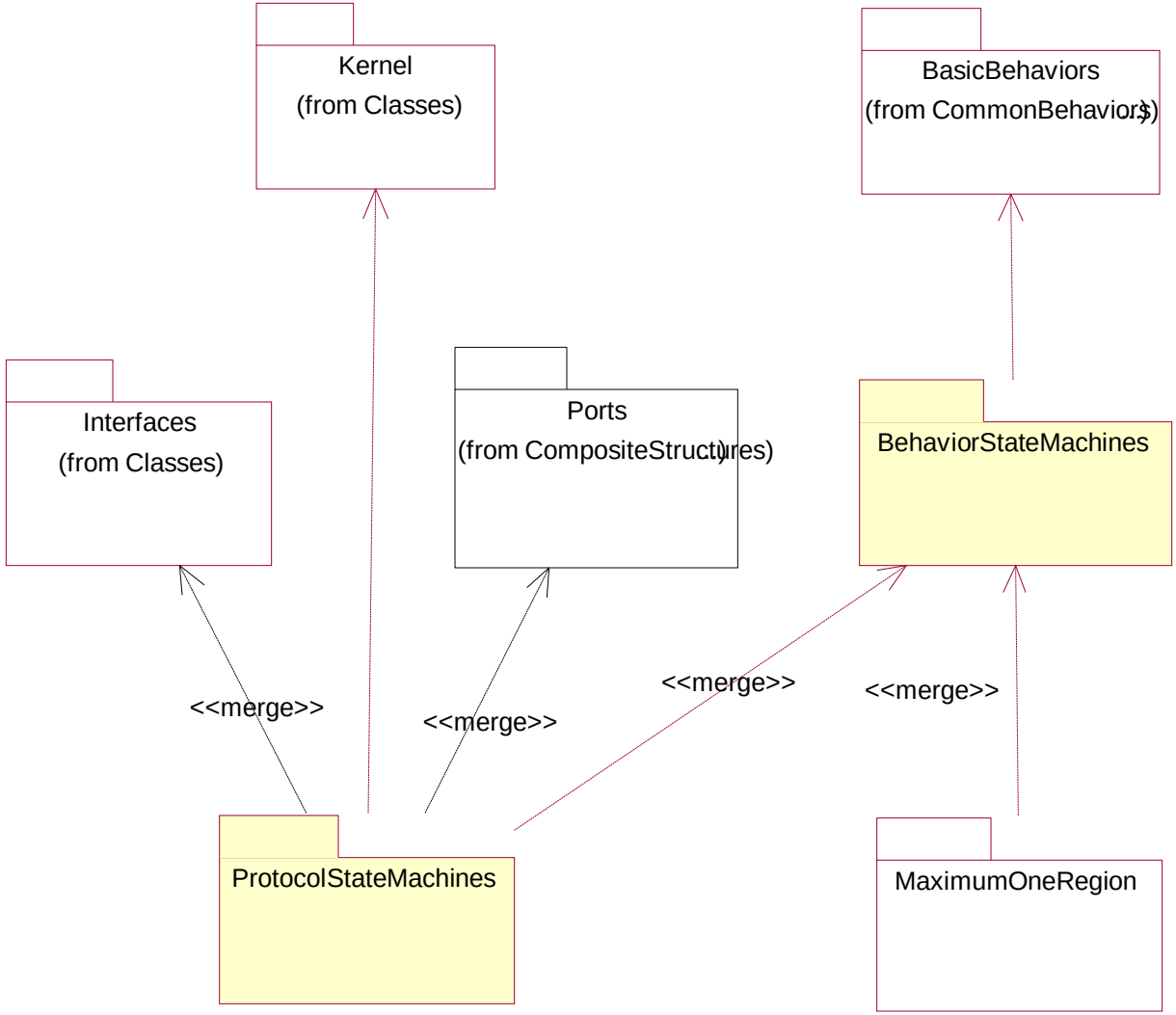
- SM beschreibt (unter Bezug auf andere Struktur- und Verhaltensdiagramme)
  - das vollständige Verhalten eines Classifiers oder
  - eines seiner Verhaltensmerkmale

- Ausprägungen
  1. Behavioral State Machines (internes Verhalten)
     - Spezifikation des Verhaltens individueller Classifier
     - objektorientierte Variante von State Charts von Harel

  2. Protocol State Machines (externes Verhalten)
     - Spezifikation von Protokollen (Dienstspezifikation)
     - Lifecycle als legale Aufrufszenerien bereitgestellter Operationen oder Ereignis-Trigger (geeignet für Interfaces und Ports)

Für konkrete Zuordnung von Strukturdiagramm (Kontext-Classifier) und Zustandsmaschine gibt es keine UML-Notation (tool-abhängig)

Zustandsautomat: Paket-Struktur



Zustandsdiagramm

= Graph

- dessen **Knoten** Zustände entsprechen, die Objekte (Instanzen des zugeordneten Classifiers) einnehmen, und dessen gerichtete **Kanten** den Zustandsübergängen entsprechen
- beim Traversieren führt die Zustandsmaschine Aktivitäten aus, die über Merkmale des Objektes operieren

Zustandsübergänge

- werden i.allg durch externe Ereignisse ausgelöst (getriggert)
- sind mit nicht **unterbrechbaren** Aktivitäten verbunden (quasi atomar)

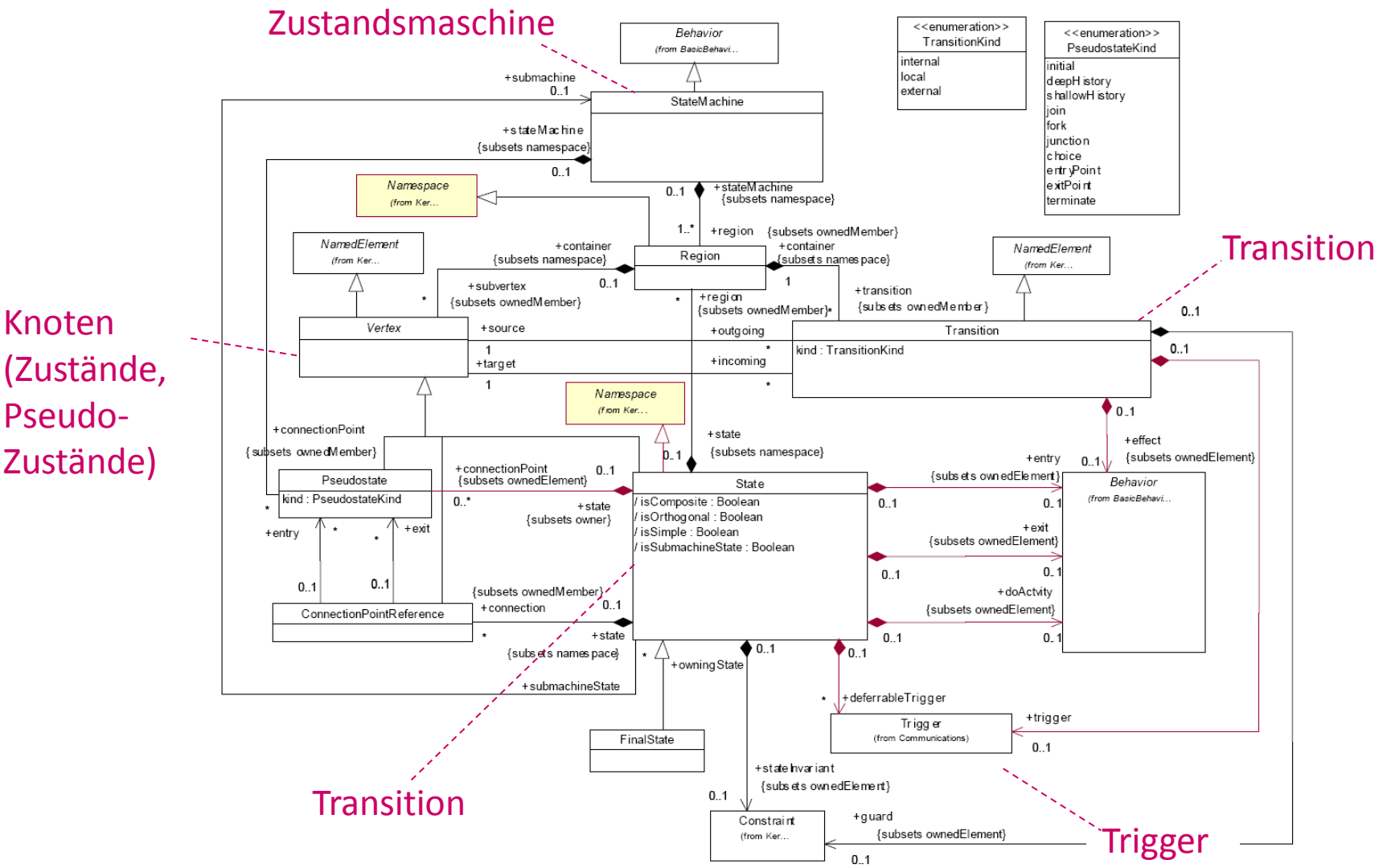
Zustand eines Objektes

- Belegung sämtlicher Attribute/Assoziationsenden zu einem Zeitpunkt

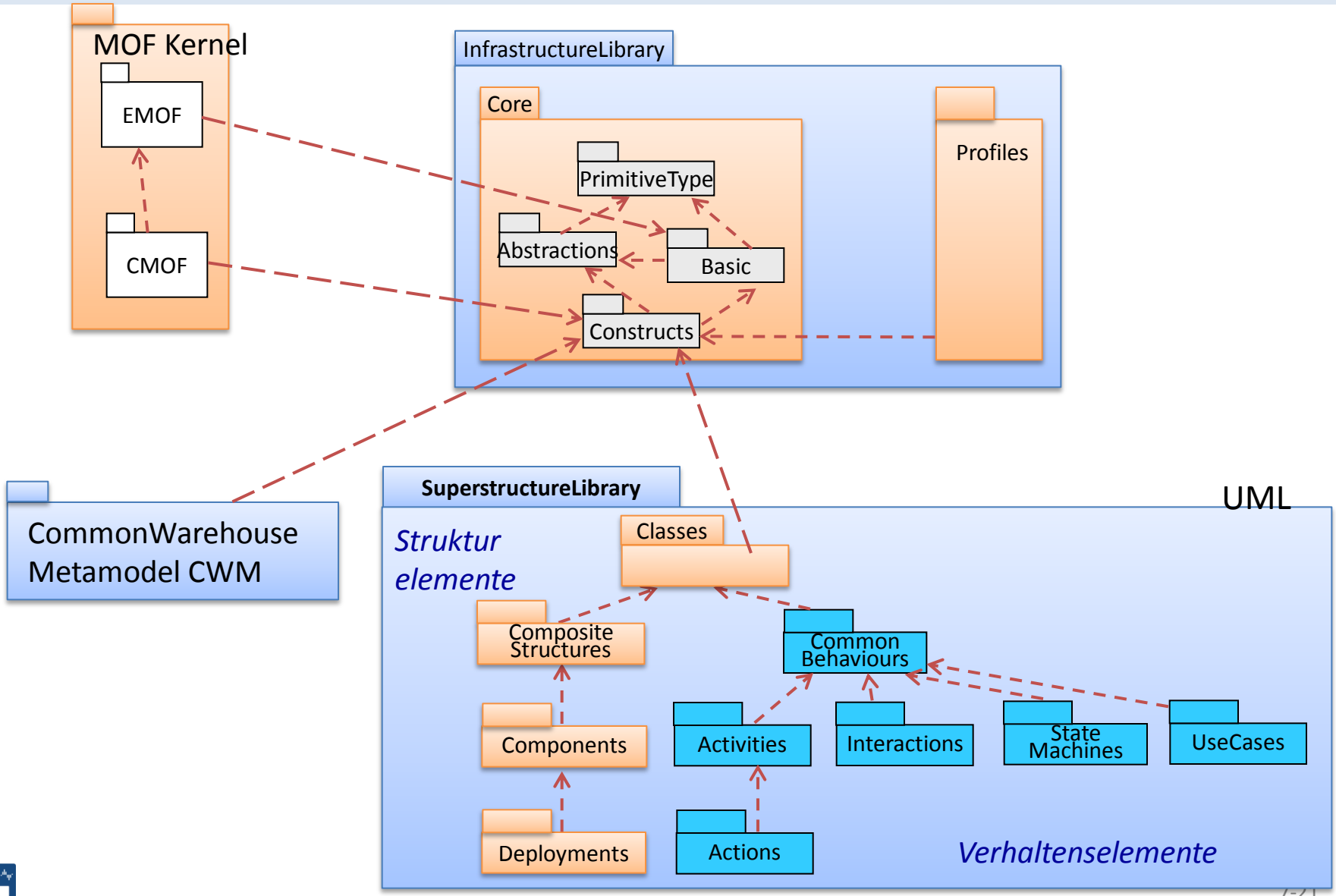
Zustand eines Automaten (ist grobkörniger)

- ist ein (Grund-)zustand, bei dem der Zustand des Objektes unterschiedlich ausgelegt sein kann
- Objekte verweilen in ihrem Grundzustand (u.U. ändern sich sogar die Attribute) bis ein Ereignis ein Zustandsübergang auslöst
  d.h. Automatenzustände können mit **unterbrechbaren** Aktivitäten verbunden sein

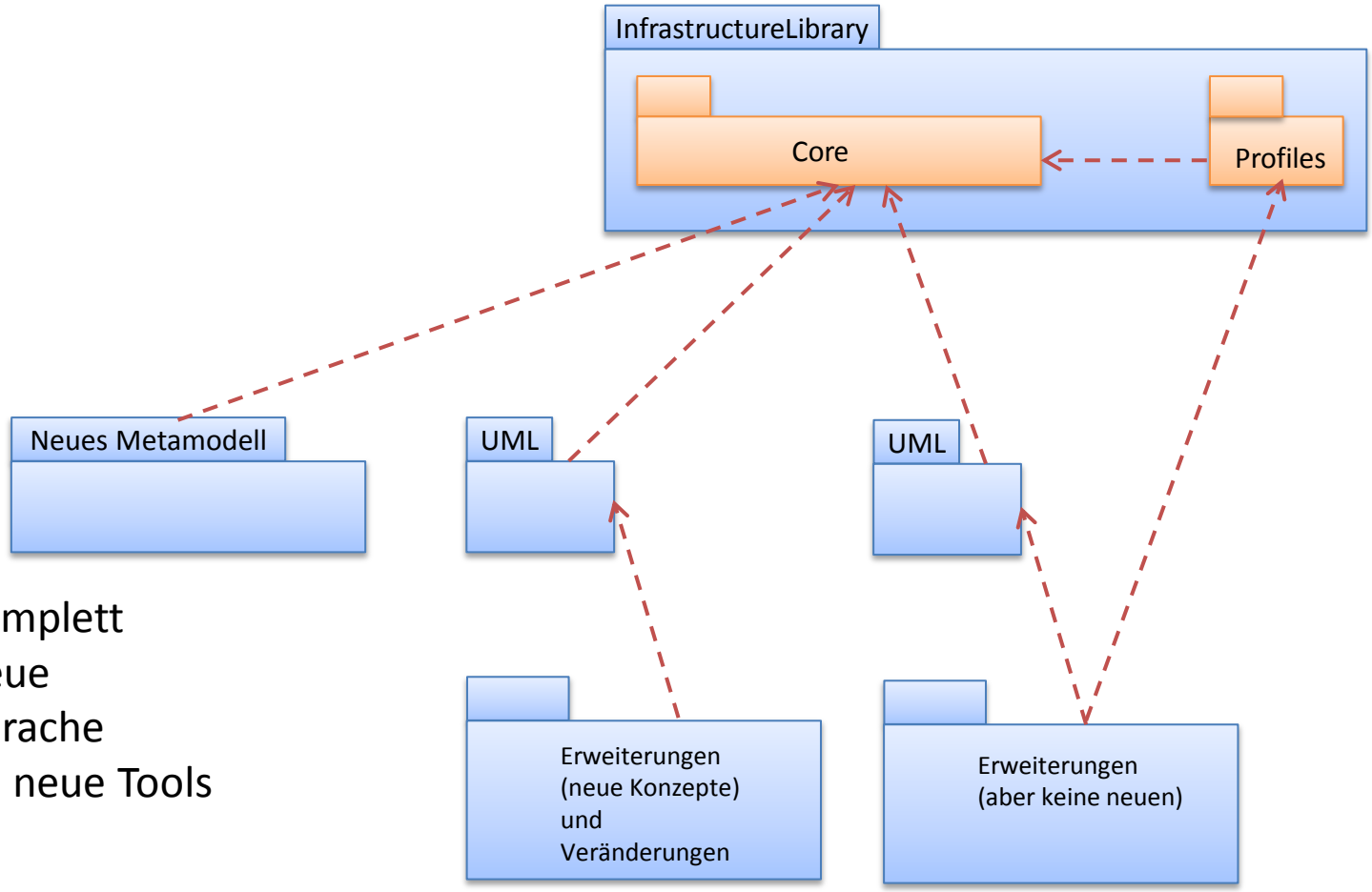
Metamodell für Zustandsmaschine



Kernkonzepte: Basis für Meta-Metamodell und Metamodell von UML



UML-Spracherweiterungsmöglichkeiten



komplett
neue
Sprache
→ neue Tools

Objektorientierte
Erweiterung der Konzepte,
Redefinition
→ neue Tools

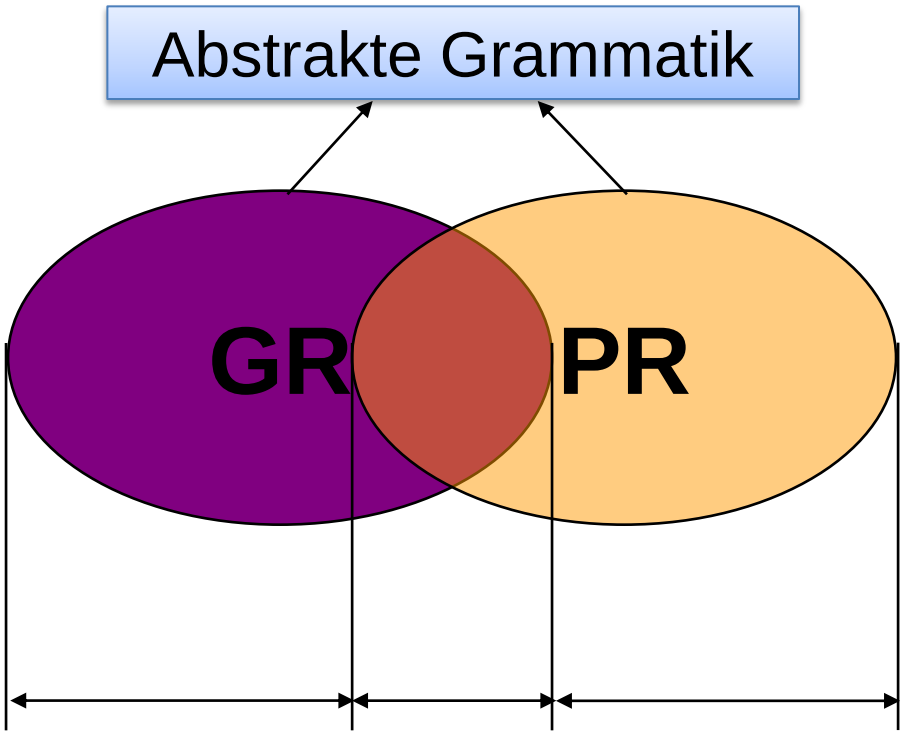
Objektorientierte
Einschränkung der Konzepte,
→ keine neuen Tools

4. SDL als UML-Ersatz

1. UML und SDL-Zustandsmaschinen im Vergleich
2. ITU-Standard Z.100
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)

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 eine nicht spezifizierte Zeit

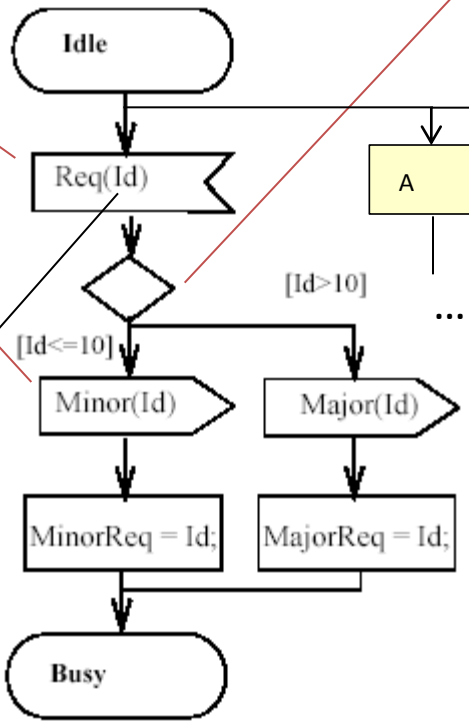
- Ereignisverwaltung (Signalempfangspuffer) ist präzisiert

- *Remote-Procedure-Call (guardeed operation) ist über Ersetzungsmodell präzisiert*

- *Systeminstanzen lassen sich definieren*

SDL-ähnliche Notation

- Empfangen eines Signals
- Senden eines Signals
- eine Aktionssequenz

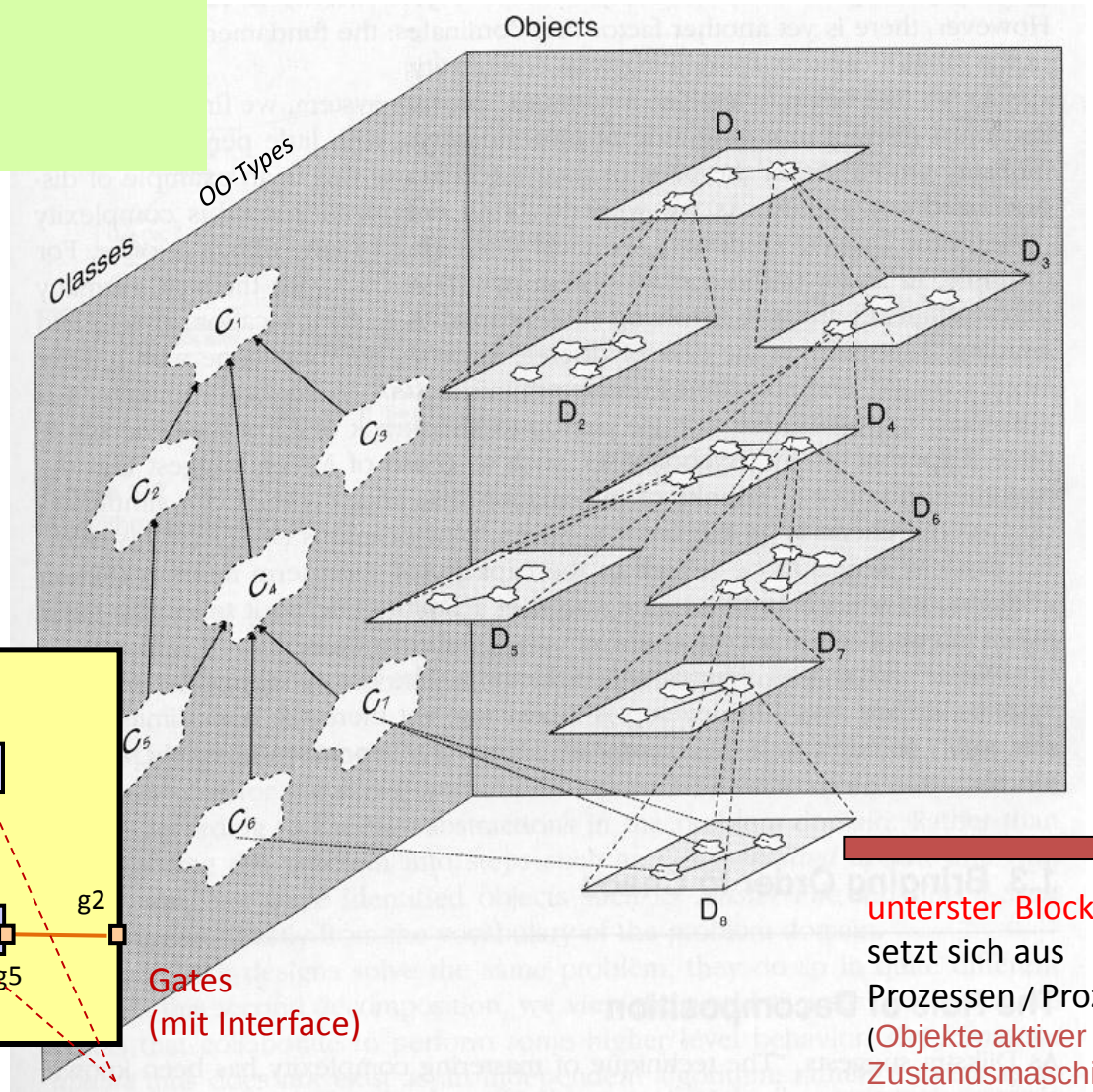


Übernahme des Signalparameters in ein Attribut Id des zugehörigen Classifiers

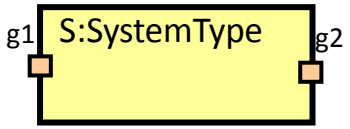
Rest möglicher Signale

Die strukturelle Systemsicht in SDL

Instanz-Typbezüge können
- impliziter und
- expliziter Art
sein



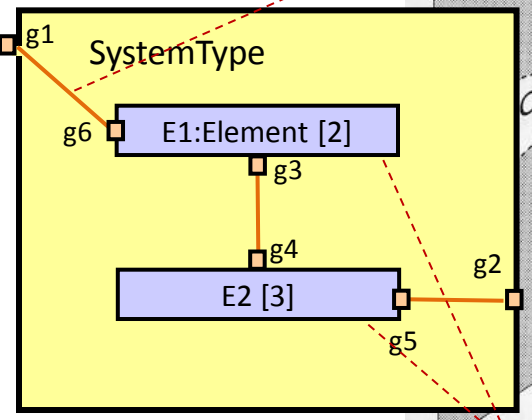
oberster „Block“
System



Übergang zur Verhaltens-Beschreibung

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

Kanäle

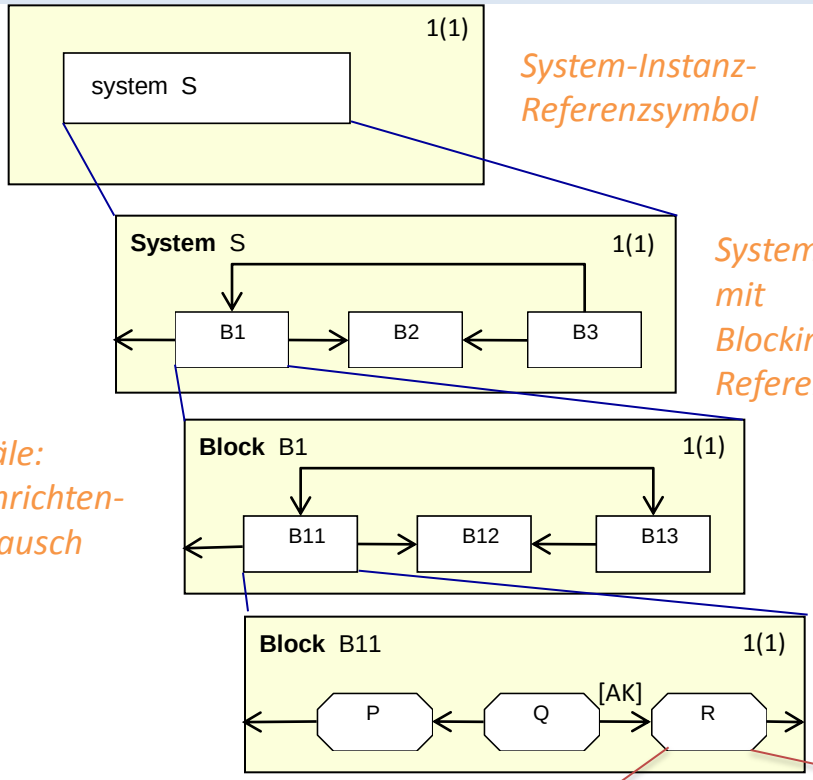


Gates
(mit Interface)

Blockinstanzmengen

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

SDL/GR - Syntax

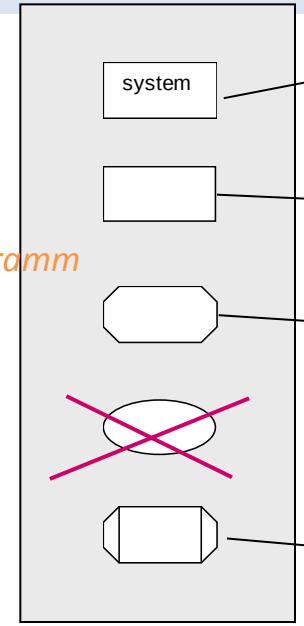


System-Instanz-Referenzsymbol

System-Diagramm mit Blockinstanz-Referenzen

Kanäle: Nachrichten-austausch

Referenzsymbole



System-Instanz

Block-Instanz(menge)

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

Service-Instanz

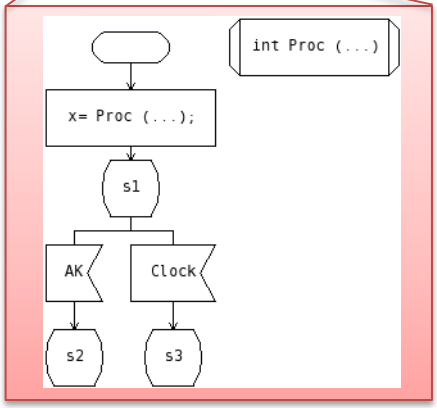
nicht in SDL-RT

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)

4. SDL

1. UML und SDL-Zustandsmaschinen im Vergleich
2. ITU-Standard Z.100
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)

4. SDL als UML-Ersatz

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

SDL-Basis

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