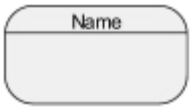
Objektorientierte Modellierung

ZUSTANDSDIAGRAMM

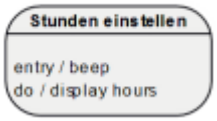
- beschreibt die **mögliche Folge von Zuständen eines Modellelements**
- normalerweise werden Zustände eines Objekts einer Klasse gezeigt (**Instanzebene**) → ZD wird der Klasse zugeordnet, von deren Objekt der Lebenslauf dargestellt wird
- Zustände werden beschrieben während:
  - des Lebenslaufs des Objekts (Erzeugung bis Destruktion)
  - der Ausführung einer Operation/Interaktion
- **modelliert werden:**
  - die möglichen **Zustände**
  - die möglichen **Zustandsübergänge** (Transitionen)
  - die **Ereignisse**, die Transitionen auslösen
  - die **Akivitäten**, die in Zuständen bzw. während Transitionen ausgeführt werden

Zustände



- **echter Zustand** = System kann sich dauerhaft im Zustand befinden
  - normaler Zustand
  - Endzustand ⦿
- **Pseudozustand** = System kann sich NICHT dauerhaft in einem Pseudozustand befinden
  - Startzustand ●
  - flacher/tiefer History-Zustand
  - Parallelisierungsknoten & Synchronisierungsknoten
  - Terminierungsknoten
  - Entscheidungsknoten

Aktivitäten

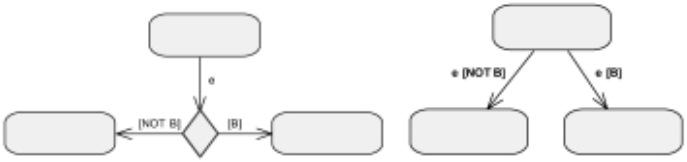


- **entry**
  - wird beim Eingang in den Zustand ausgeführt
- **exit**
  - wird beim Verlassen des Zustands ausgeführt
- **do**
  - wird ausgeführt; Parameter sind erlaubt
- **event** – behandelt Ereignis innerhalb des Zustands (Name des Events wird angegeben)
  - wird ausgeführt, wenn sich das System in dem Zustand befindet und das Ereignis eintritt

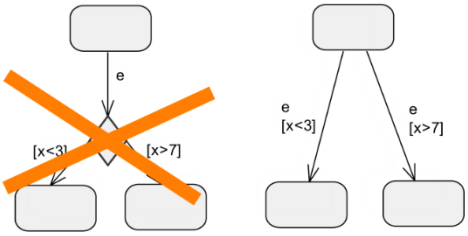
Transitionen

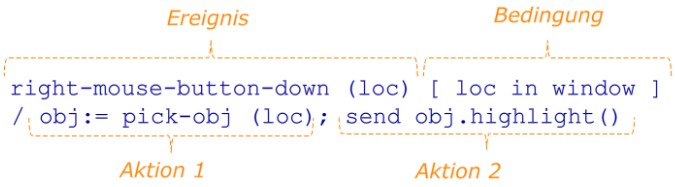
- Zustandsübergänge erfolgen, wenn
  - das Ereignis eintritt, das die Transition triggert
    - eine evt. noch andauernde Aktivität im Vorzustand wird unterbrochen
  - UND die Bedingung (guard) erfüllt ist
    - bei Nicht-Erfüllung geht das nicht „konsumierte“ Ereignis verloren → wenn die Bedingung erst zu einem späteren Zeitpunkt erfüllt wird, kann die Transition ohne neuerliches Ereignis nicht durchgeführt werden

- durch Bedingungen könne **Entscheidungsbäume** mit **Entscheidungsknoten** modelliert werden



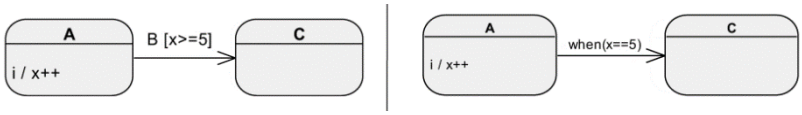
  - **VORSICHT bei Entschiedungsknoten:** Zustandsautomat kann im Entscheidungsknoten hängen bleiben, wenn z.B. die guard-Bedingungen nicht den gesamten Lösungsraum abdecken → lieber ohne Entscheidungsknoten modellieren
- default-Werte für Transitionen
  - kein Ereignis = „Aktivität abgeschlossen“
  - keine Bedingung = true
- **Syntax für Transitionen**
  - **Ereignis(Parameter) [Bedingung] / Aktivität**
  - Beispiel:





Ereignistypen

- **CallEvent**
  - Empfang einer Nachricht (Operationsaufruf)
  - z.B. stornieren(), kollidiertMit(Termin)
- **SignalEvent**
  - Empfang eines Signals
  - z.B. right-mouse-button-down, ok-Taste-gedrueckt
- **ChangeEvent**
  - Bedingung wird wahr
  - Bedingung wird **permanent** überprüft → sobald sie wahr wird, tritt das ChangeEvent ein (außer zugehörige Überwachungsbedingung blockiert das)
    - **Unterschied zu Überwachungsbedinung:** wird nur überprüft, wenn zugeordnetes Ereignis eintritt; kann selbst keinen Zustandsübergang auslösen
  - z.B. when(x<y), when(a=1), when(terminBestaetigt)



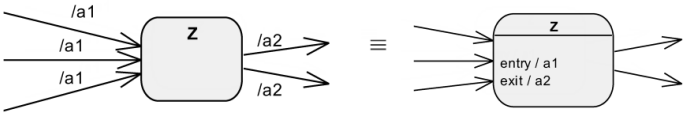
- **TimeEvent**
  - Zeitablauf oder Zeitpunkt
  - z.B. after(5sec), when(date=03.04.2017)
- Startzustand
  - Pseudozustand
  - „Beginn“ des ZD
  - keine eingehenden Transitionen
  - **genau eine** ausgehende Transition
    - wird sofort ausgelöst, wenn sich das System im Startzustand befindet
    - keine Bedingungen und Ereignisse (außer zur Erzeugung des Objektes)
    - Aktivitäten dürfen angegeben werden

- Endzustand
  - **keine** ausgehenden Transitionen
  - echter Zustand
- Terminierungsknoten
  - betrachtetes Objekt des ZD wird terminiert
  - modelliert durch ein X

Innere Transitionen

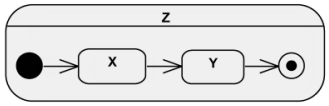
- werden auch von Ereignissen ausgelöst, verlassen aber den aktuellen Zustand nicht
- **äquivalent zur Selbsttransition**, sofern keine entry/exit-Aktivitäten vorhanden sind

- gleiche Aktivitäten können in den Zustand hineingezogen werden, um zusammenzufassen



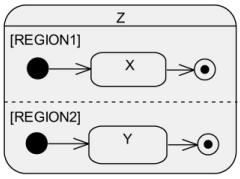
Komplexe Zustände

- **Zustände, die selbst aus mehreren Subzuständen zusammengesetzt sind → geschachteltes ZD**
- Subzustände sind disjunkt = genau ein Subzustand ist aktiv, wenn der komplexe Zustand aktiv ist



Zu einem Zeitpunkt kann nur X ODER Y aktiv sein!

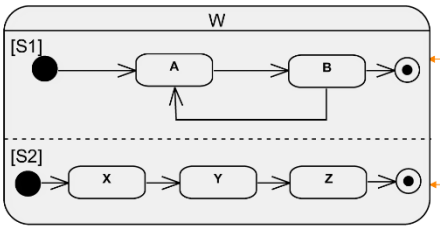
- Teilung des Superzustandes in mehrere Regionen
  - Subzustände sind nebenläufig, also gleichzeitig aktiv
  - **Superzustand ist orthogonal**



Zu einem Zeitpunkt sind X UND Y aktiv!

- **man kann durch komplexe Zustände UND- bzw. ODER-Verfeinerungen modellieren**

  - Beispiel



Zu einem Zeitpunkt kann nur A ODER B aktiv sein!

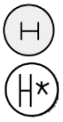
Zu einem Zeitpunkt kann nur X ODER Y ODER Z aktiv sein!

Zu einem Zeitpunkt ist jeweils ein Subzustand jeder der beiden orthogonalen (=parallelen) Regionen von W aktiv!

  - Mögliche Kombinationen von gleichzeitig aktiven Zuständen:
    A & X *oder* A & Y *oder* A & Z *oder* A & Endzustand von [S2]
    B & X *oder* B & Y *oder* B & Z *oder B* & Endzustand von [S2]
    Endzustand von [S1] & X *oder* Endzustand von [S1] & Y *oder* Endzustand von [S1] & Z *oder* Endzustand von [S1] & Endzustand von [S2]

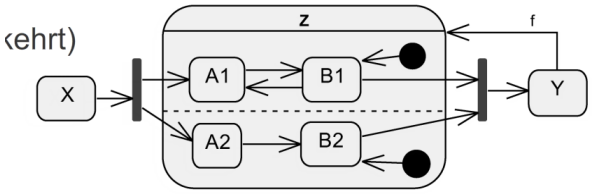
History – Zustände

- **History-Zustand merkt sich, in welchem Subzustand sich das System beim Verlassen eines übergeordneten Zustandes befunden hat, wenn es wieder in diesen Zustand zurückkehrt**
- merken sich also jenen internen Zustand in einem komplexen Zustand, von dem die letzte Transition ausgegegangen ist
- zu einem späteren Zeitpunkt kann zu diesem Zustand über Transitionen aus übergeordneten Zuständen wieder zurückgekehrt werden
- **flacher History-Zustand**: merkt sich **eine Ebene**
- **tiefer History-Zustand**: merkt sich **alle Ebenen über die gesamte Schachtelungstiefe**
    - man kann auch eine bestimmte Schachtelungstiefe angeben, z.B. H^4 statt H*

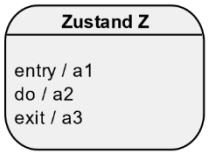


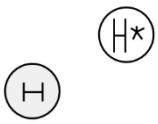
Parallele Abläufe

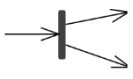
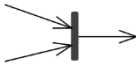
- wird ein orthogonaler Zustand aktiviert (Zustand mit mehreren Regionen), so werden alle seine nebenläufigen Regionen aktiviert
- **man kann den Kontrollfluss von orthogonalen Zuständen aufspalten:**
    - → Verwendung einer komplexen Transition in Form von **Parallelisierungs- bzw. Synchronisierungsknoten**
    - komplexe Tranisition ändert den Grad der Nebenläufigkeit
    - es werden nicht automatisch in den Regionen die Startzustände aktiviert



ZUSAMMENFASSUNG

Name	Syntax	Beschreibung
Zustand		Bei Erreichen des Zustands Z wird die Aktivität a1 ausgeführt, während Z der aktuelle Zustand ist, wird a2 ausgeführt und beim Verlassen von Z wird a3 ausgeführt.
Transition		Zustandsübergang
Startzustand		Beginn des Zustandsdiagramms

Name	Syntax	Beschreibung
Endzustand		Ende
Terminierungs-knoten		Das modellierte Objekt hört auf zu existieren.
Flacher/tiefer History-Zustand		"Rücksprungadresse"

Name	Syntax	Beschreibung
Entscheidungs-knoten		Knoten, von dem mehrere alternative Transitionen ausgehen können.
Parallelisierungs-knoten		Aufspaltung des Kontrollflusses in mehrere parallele Zustände
Synchronisierungs-knoten		Zusammenführung des Kontrollflusses von mehreren parallelen Zuständen

Checkliste

- Sie haben diese Lektion verstanden, wenn Sie wissen..

- Was mit dem Zustandsdiagramm modelliert wird
- Was Ereignisse und Aktivitäten sind und wie sie eingesetzt werden
- Wozu Bedingungen benötigt werden und was der Unterschied zu Ereignissen ist
- Welche Aktivitäten es innerhalb eines Zustands gibt
- Wozu und wie ein Historischer Zustand eingesetzt wird
- Was komplexe und orthogonale Zustände sind