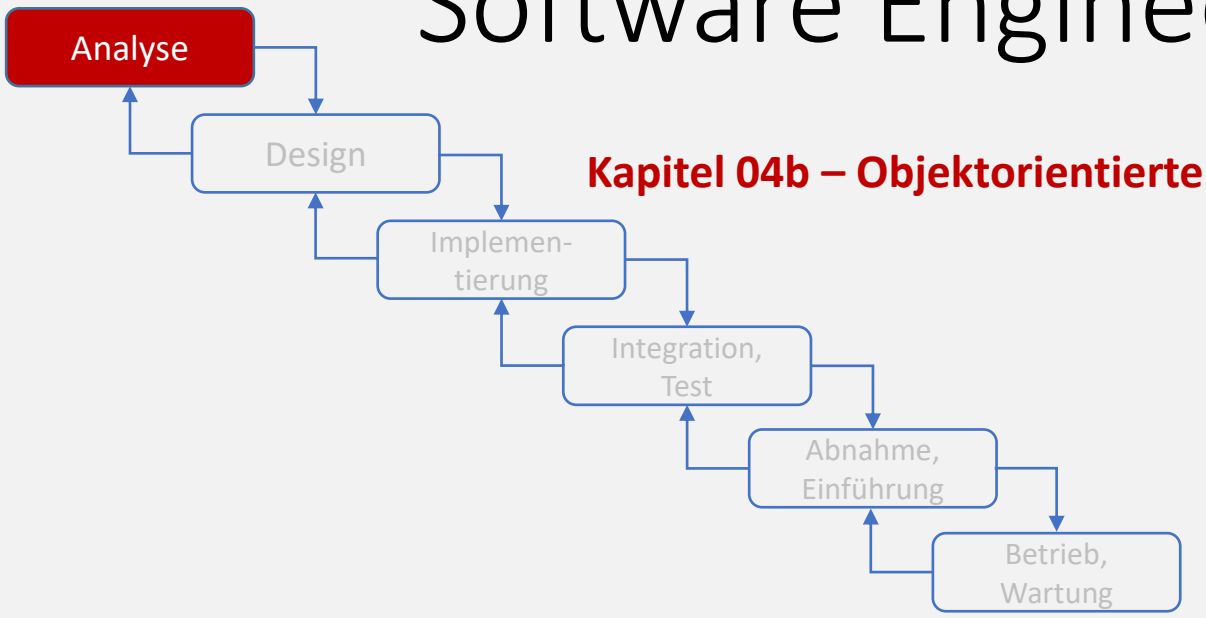
Software Engineering



Kapitel 04b – Objektorientierte Analyse (OOA)

Zustandsdiagramme

OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG
INFORMATIK UND MATHEMATIK

Analyse im Großen

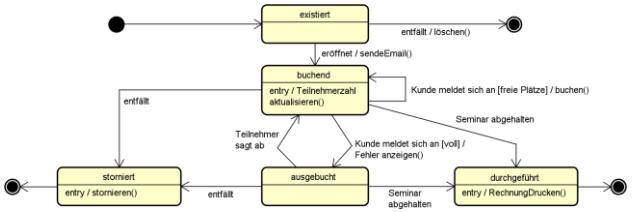
Use Case Modell aufstellen, liefert als Ergebnisse:

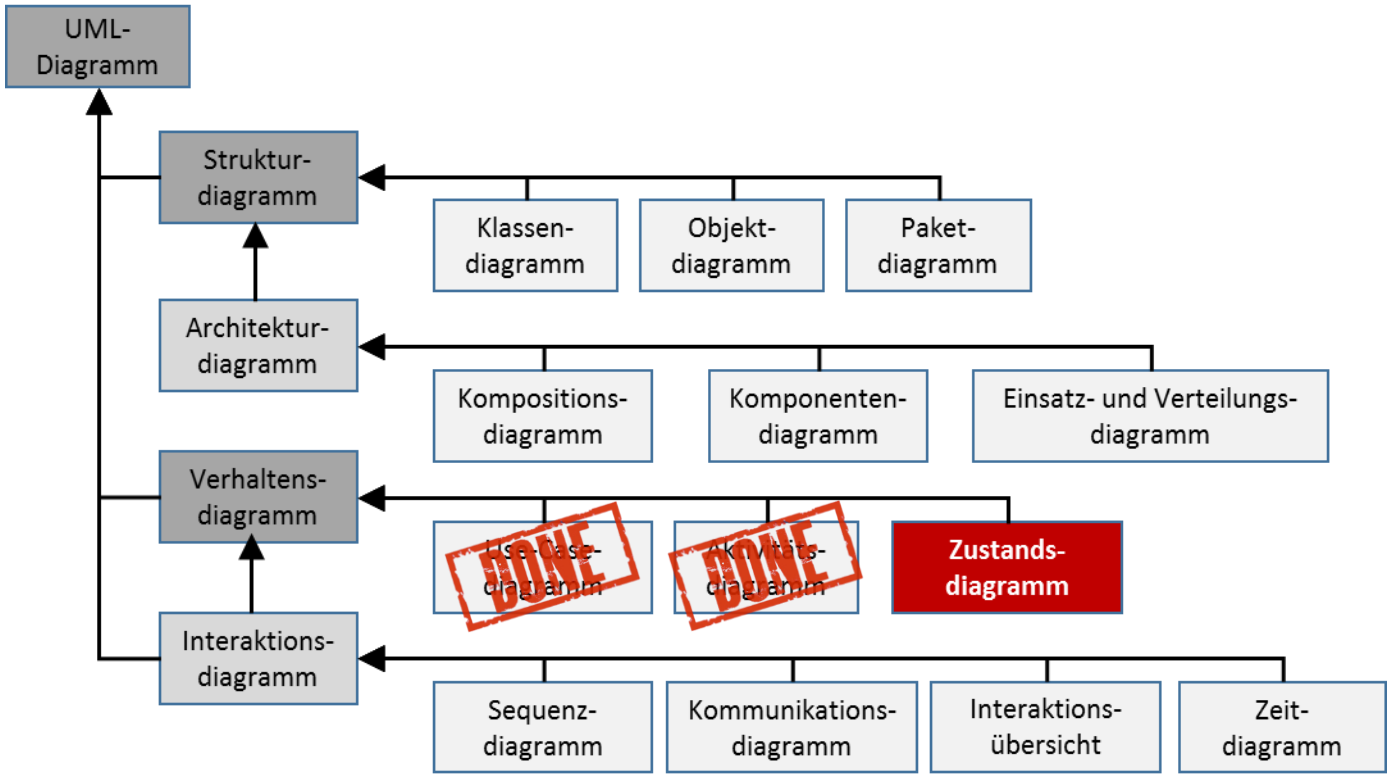
- Use-Case-Diagramm
- Use-Case-Beschreibung
- Aktivitätsdiagramme
- Zustandsdiagramm

Pakete bilden, beinhaltet:

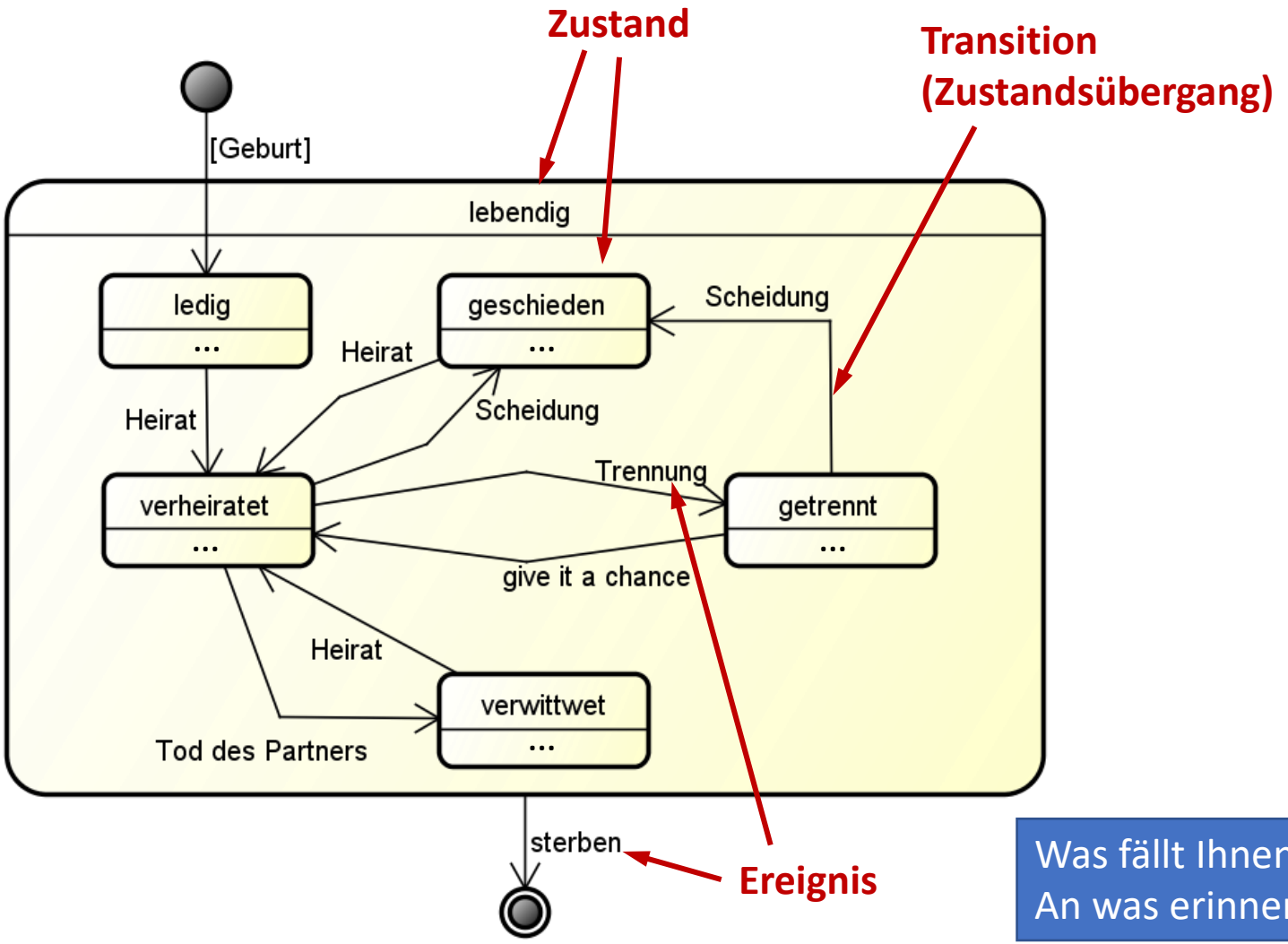
- Teilsysteme festlegen (Modellelemente zu Paketen zusammenfassen)
- Bei großen Systemen erfolgt Paketbildung meist zu Anfang
- Ergebnis: Paketdiagramm

UML: Zustandsdiagramm (State Chart von D. Harel)



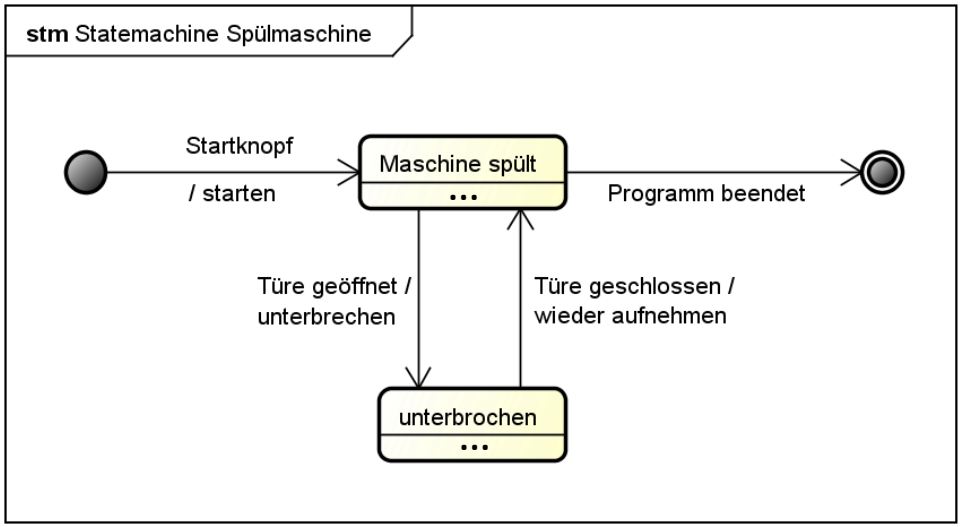


UML: Zustandsdiagramm
Unvollständiges erstes Beispiel



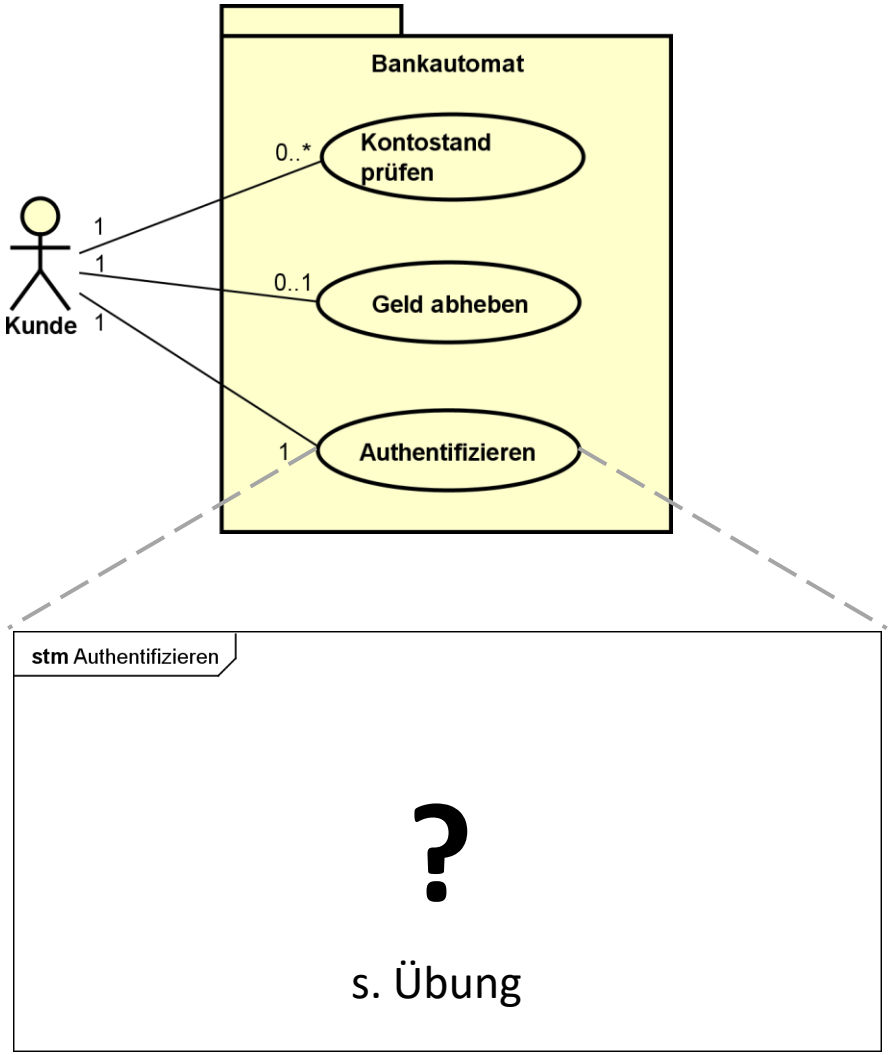
Was fällt Ihnen auf?
An was erinnert es?

UML: Zustandsdiagramm Beispiel



- Fokus: Modellierung von Verhalten von Klassen, das an internem Zustand hängt
- Beantwortet die Frage: "*Wie verhält sich System in bestimmtem Zustand bei gewissen Ereignissen*?"
- Nutzung, wenn Reaktionen einer Klasse direkt von einem bestimmten (internen) Zustand abhängig sind:
  - Spülmaschine hat Zustände "spült" und "unterbrochen"
  - Nur im Zustand "spült" kann das Spülprogramm beendet werden

UML: Zustandsdiagramm
Beispiel: Zusammenhang mit Use Cases

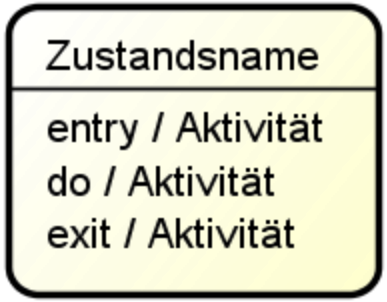


UML: Zustandsdiagramm-Elemente
Zustand (1)

Zustand:

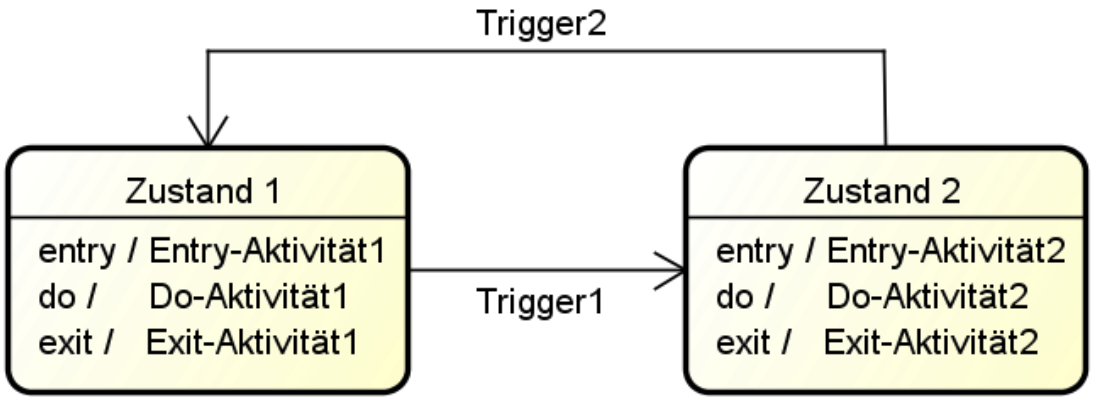
Beispiele?

- Zustand = Objekt in bestimmter Situation
- Kann auf äußere Ereignisse reagieren
- **Notation:** Zustandsname + Aktivität:
  - `entry/:` Aktivitäten bei Eintritt in Zustand
  - `do/:` Aktivitäten während Aufenthalt im Zustand
  - `exit/:` Aktivitäten bei Verlassen des Zustandes



- Verarbeitungsmodell:

UML: Zustandsdiagramm-Elemente Zustand (2)



Ausführungssemantik (informell):

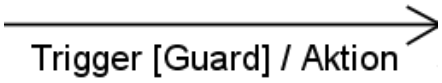
- Ist Zustand 1 Endpunkt einer durchlaufenen Transition, so wird er betreten
- Ist Zustand 1 Anfang einer durchlaufenen Transition, so wird er verlassen

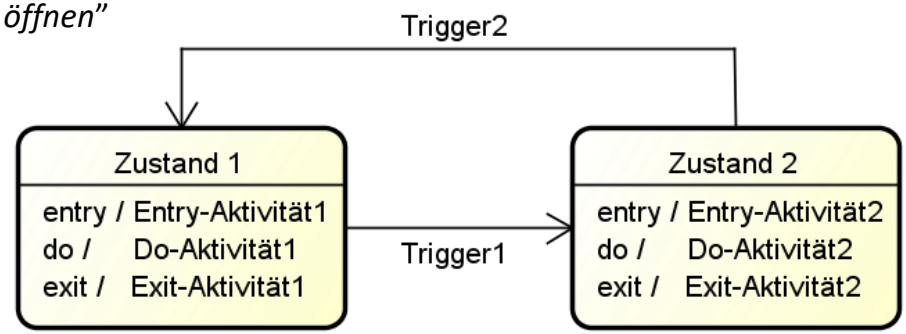
UML: Zustandsdiagramm-Elemente Transition (Zustandsübergang) (1)

Transition:

- Modelliert Übergang von einem Quell- in einen Zielzustand
- Quellzustand kann gleich Zielzustand sein (Selbsttransition)
- Transition schaltet (wird durchlaufen), wenn zugehöriges Triggerereignis eintritt
- Konvention: Durchlaufen der Transition nimmt keine Zeit in Anspruch
- **Notation:**
  - **Trigger:** Externes Ereignis, das Transition auslöst
    - Bsp.: "*Nutzer klickt den Transfer-Button*"
  - **Guard:** Durchlaufbedingung für Transition (nur bei `true` kann Transition schalten)
    - Bsp.: "*Wenn die gemessene Temperatur 32 Grad Celsius übersteigt*"
  - **Aktion:** Beim Durchlauf durchzuführende Aktion
    - Bsp.: "*Ventil zwischen Behälter1 und 2 öffnen*"

Beispiele?

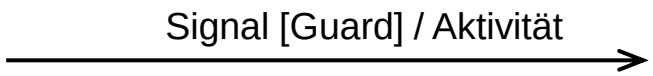




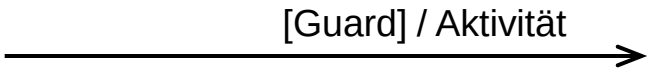
UML: Zustandsdiagramm-Elemente Transition (Zustandsübergang) (2)

Weitere Transitionen:

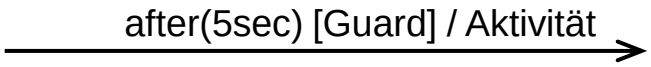
- mit Verwendung eines SignalTriggers:
  - Beispiele: Sensor, Tastatur, Exception, Timer, Interrupt
  - Signale werden extern generiert
  - Reaktion: Zustandsübergang

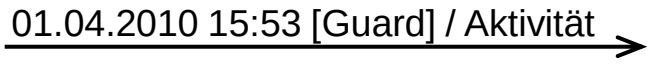


- mit Verwendung eines CompletionTrigger:
  - Ereignis wird erzeugt, sobald do/-Aktion des Quellzustandes beendet ist



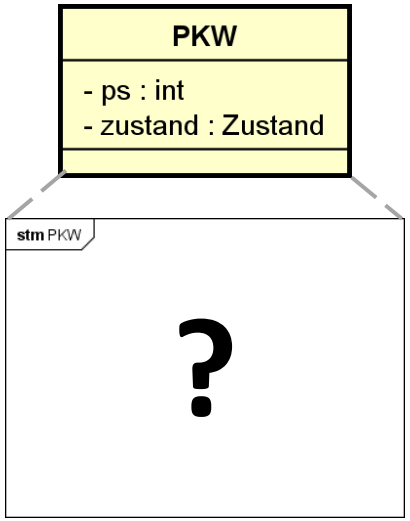
- mit Verwendung eines TimeTriggers:
  - Relative Angabe: `after(Zeitangabe)`
  - Absolute Angabe: Datum Uhrzeit





UML: Zustandsdiagramm-Elemente Transition (Zustandsübergang) (3)

Modellieren Sie das Leben eines PKW als Zustandsdiagramm (4 Zustände)



Vorgehensweise:

- Versuchen Sie zunächst nur die Zustände zu finden
- Anschließend die Kanten zu ziehen
- Und dann die Kanten und Zustände auszumodellieren
  - `entry`, do, `exit`
  - Trigger, Guards, Actions

Ein Neuwagen gilt als neu, bis er angemeldet und benutzt wird.

Durch Benutzung befindet sich der Wagen so lange im Zustand gebraucht, bis er durch Verschleiß nach 20 Jahren schrottreif wird oder durch einen Unfall defekt ist.

Ist der Defekt reparierbar, gilt der Wagen wieder als gebraucht.

Ist er nicht reparierbar, hilft nur mehr abmelden und verschrotten.

(Hinweis: Start- und Endzustände wie in Aktivitätsdiagrammen)

UML: Zustandsdiagramm-Elemente Transition (Zustandsübergang) (4)

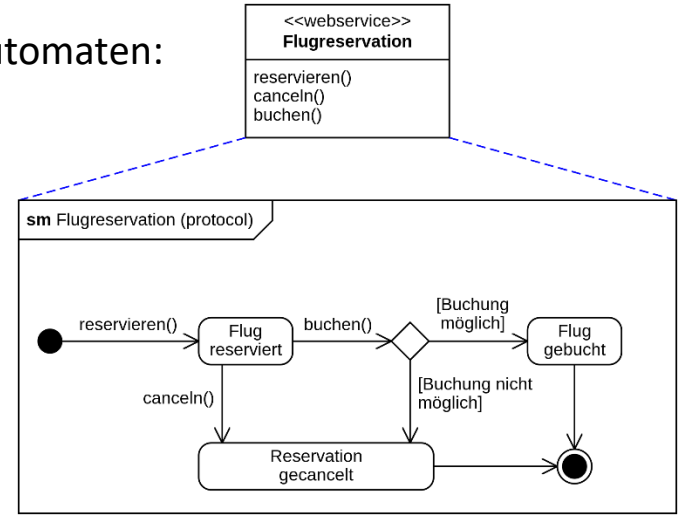
Weitere Transitionen:

- mit Verwendung eines ChangeTriggers:
  - Wird erzeugt, sobald Bedingung von `false` nach `true` wechselt
  - Beispiel: `when [x>y]`
  - Implementierung evtl. schwierig

when[Guard] / Aktivität

- mit Verwendung eines CallTriggers:
  - Event ist UML-Operation (d.h. Methodenaufruf)
  - Sollte in der Klasse deklariert sein
  - Angewandt in Protokollzustandsautomaten:

Operation [Guard] / Aktivität



UML: Zustandsdiagramm-Elemente

Pseudozustände:

Notation:

- Startzustand:
  - Startpunkt für das Betreten des Zustandsautomaten
  - Einschränkungen:
    - Transitionstrigger und Bedingung sind i.d.R. leer
    - Genau eine abgehende Kante

- Endzustand:
  - Endpunkt der Abarbeitung des Zustandsautomaten
  - Einschränkungen:
    - Keine Ausführung von Aktivitäten mehr
    - Keine ausgehenden Kanten

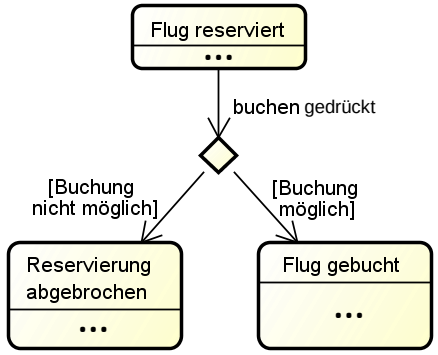
- Terminator:
  - Abarbeitung des Zustandsautomaten wird abgebrochen
  - Objekt, dessen Verhalten modelliert wird, hört auf zu existieren
  - Einschränkungen:
    - Keine ausgehenden Kanten

UML: Zustandsdiagramm-Elemente

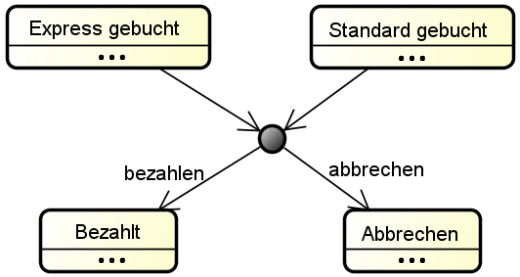
Pseudozustände:

- Entscheidung:
  - Beschreiben Entscheidungen, die vom Wert des Transitionsübergangs abhängen
  - Guards der ausgehenden Transitionen werden ausgewertet, nachdem eingehende Transition durchlaufen

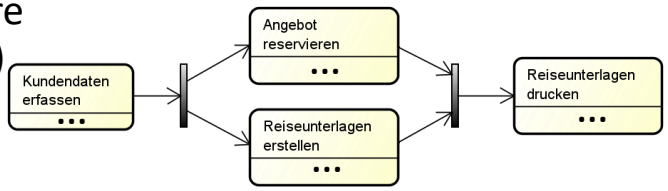
Notation:



- Kreuzung/Verbindung (Junction-Point):
  - Genutzt um viele Transitionen zu wenigen zusammenzuführen (Übersichtlichkeit)

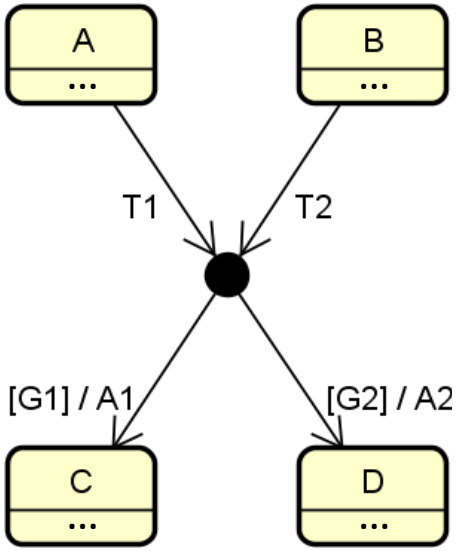


- Teilung und Synchronisation (Fork/Join):
  - Fork: Aufteilung einer eingehenden Transition auf mehrere ausgehende Zustände (keine Guards auf ausgehenden Tr.)
  - Join: Zusammenfassung paralleler Zweige (keine Guards auf eingehenden Transitionen)

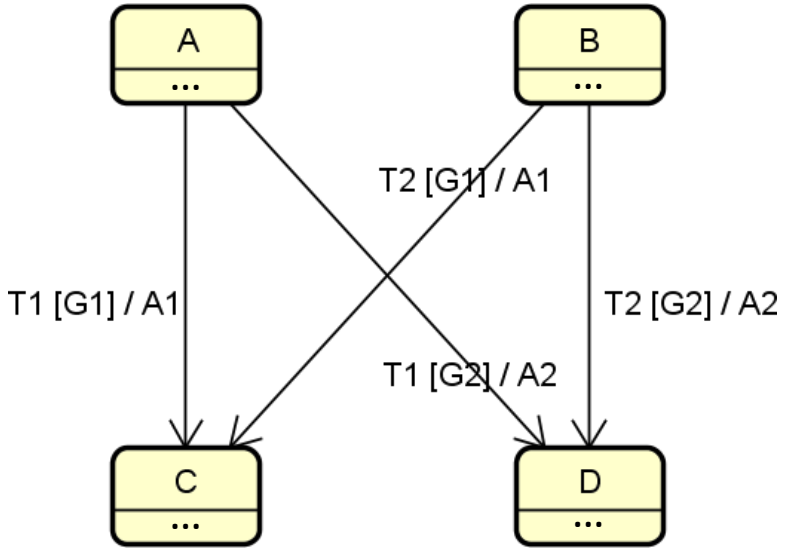


UML: Zustandsdiagramm-Elemente

Hinweis zu Kreuzungen:



entspricht:



UML: Zustandsdiagramm-Elemente

Pseudozustände:

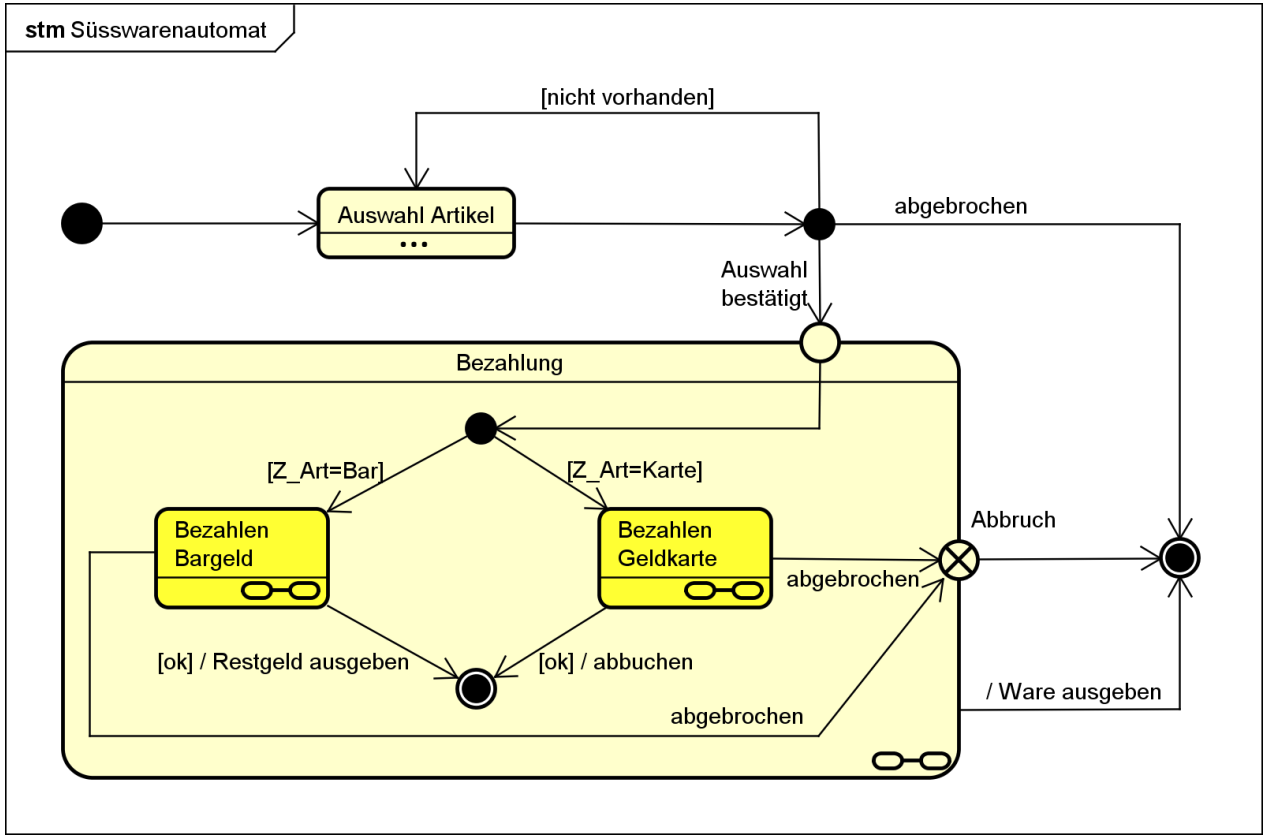
- Eintrittspunkt:
  - Eintrittspunkt eines Zustandsdiagramms oder eines zusammengesetzten Zustandes
- Austrittspunkt:
  - Austrittspunkt aus Zustandsdiagramm oder zusammengesetzten Zustand
- Historie:
  - Merken des letzten aktiven Zustandes vor Eintritt in tiefere Ebene
  - Flache Historie:
    - Speichert eine Ebene
  - Tiefe Historie
    - Speichert alle Zustände über alle Ebenen hinweg

Notation:

UML: Zustandsdiagramm-Elemente

Beispiel zu Entry- und Exitpoints:

- Sinn: Können der Übersichtlichkeit dienen

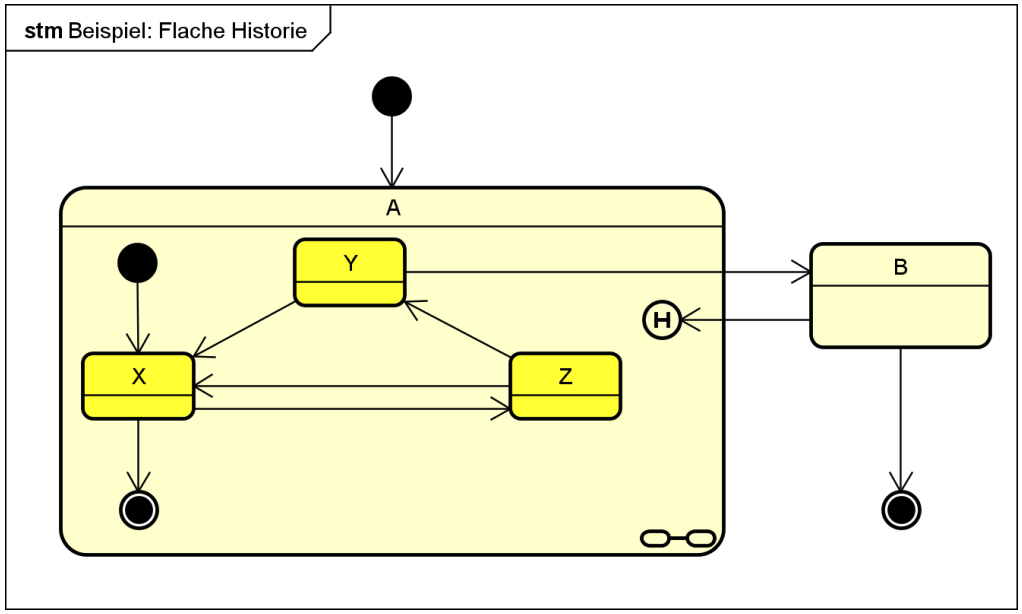


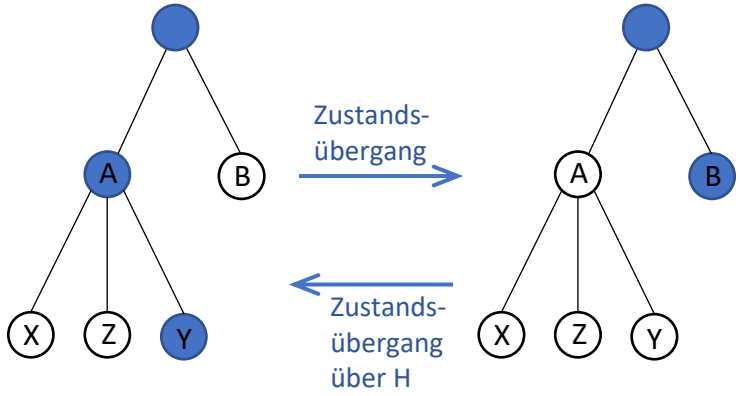
[Quelle: UML 2 glasklar]

UML: Zustandsdiagramm-Elemente

Beispiel zu Historie-Pseudozuständen:

- flach:





: aktiver Zustand

: nicht aktiver Zustand

[Quelle: UML 2 glasklar]

UML: Zustandsdiagramm

Modellieren Sie ein Autoradio, das anfangs im Zustand “aus” ist und dann zwischen dem Zustand “ein” und “aus” wechseln kann. Im Zustand “ein” kann das Radio in einem der Betriebsmodi “Kassettenbetrieb”, “CD-Wechslerbetrieb” oder “Radiobetrieb” sein. Nach dem “Aus-” und wieder “Ein-” schalten soll sich das Radio so verhalten, wie kurz vor dem Ausschalten. Ab Werk wird das Auto so ausgeliefert, dass beim initialen Einschalten das Autoradio im Betriebsmodus “Radiobetrieb” läuft.

Hinweise:

- Nutzen Sie Historie-Pseudozustände
- Nutzen Sie hierarchische Zustände (s. Aktivitätsdiagramme)
- Modellieren Sie sinnvolle Zustandsübergänge, nutzen Sie Trigger und Guards

UML: Zustandsdiagramm Fahrkartenautomat

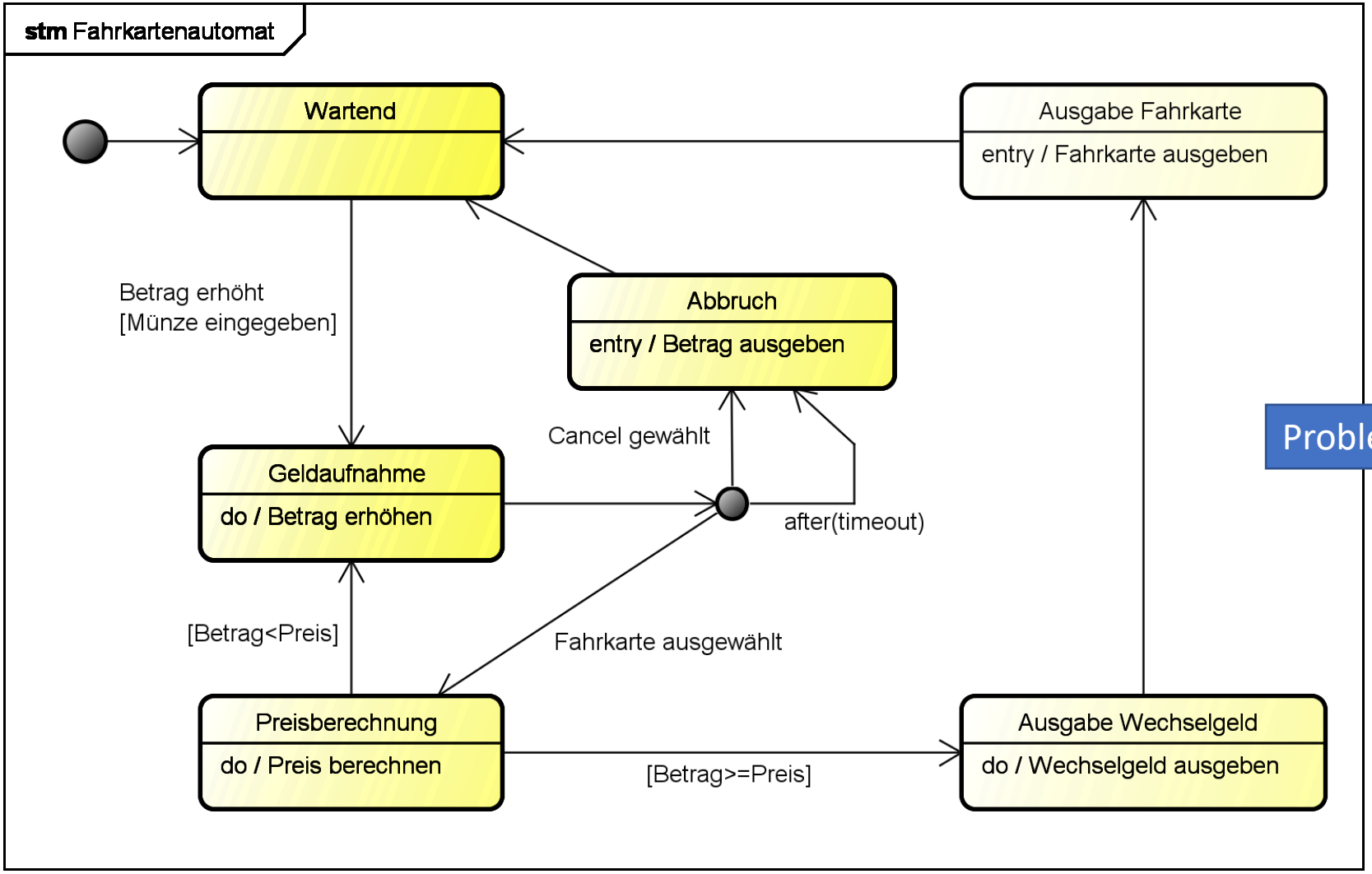
Modellieren Sie einen Fahrkartenautomaten:

Vorgehensweise:

- Versuchen Sie zunächst nur die Zustände zu finden
- Anschließend die Kanten zu ziehen
- Und dann die Kanten und Zustände auszumodellieren
  - `entry`, do, `exit`
  - Trigger, Guards, Actions

(Hinweis: ein Start und kein Endzustand)

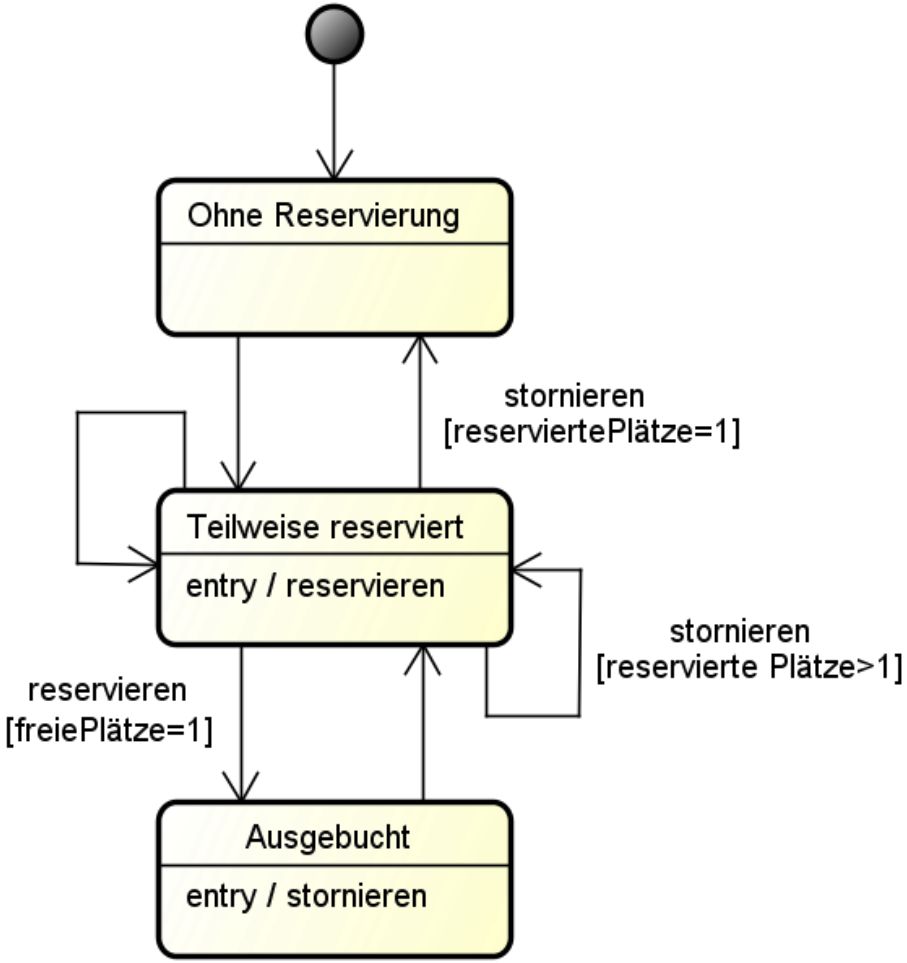
UML: Zustandsdiagramm Fahrkartenautomat



Probleme?

UML: Zustandsdiagramm

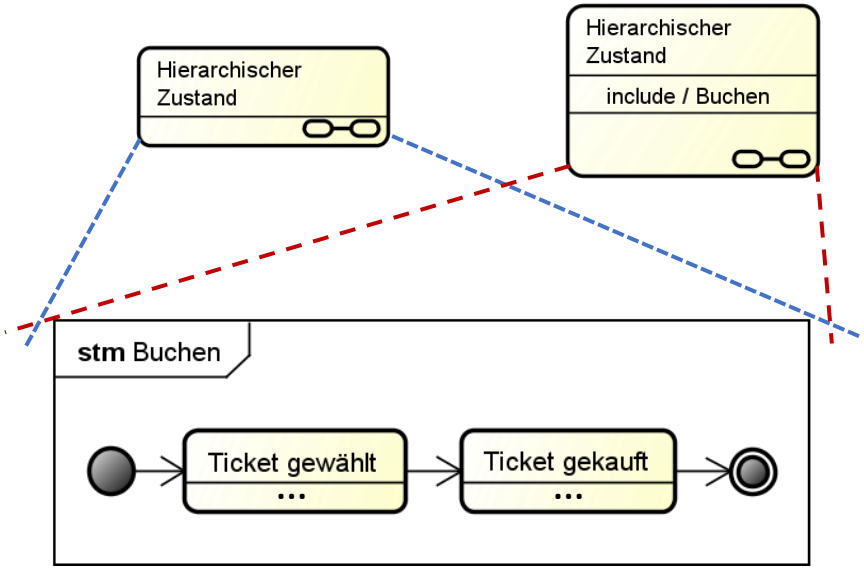
Finden Sie die Fehler und korrigieren Sie das Zustandsdiagramm:



UML: Zustandsdiagramm-Elemente
Hierarchische Zustände (1)

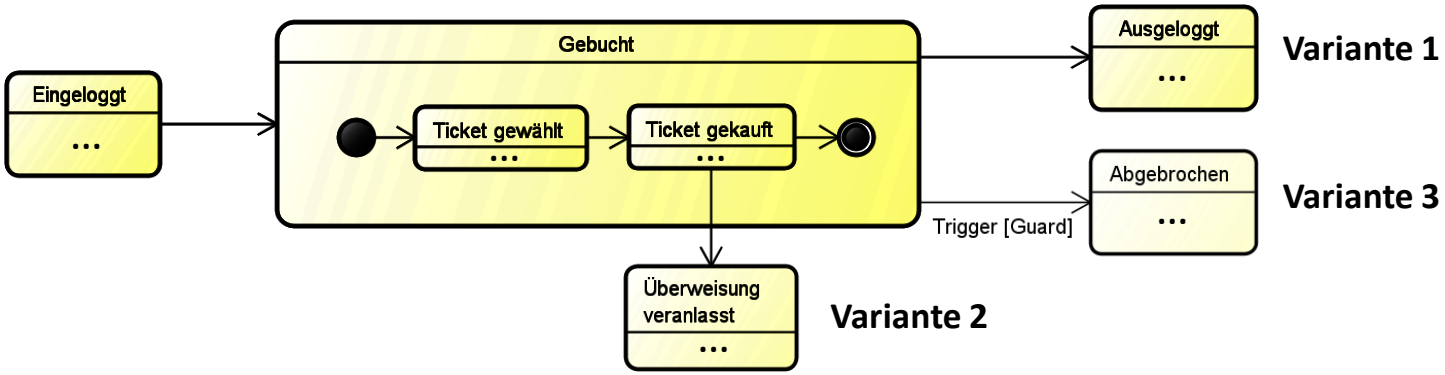
Hierarchischer Zustand:

- Setzt sich aus einem oder mehreren Zuständen zusammen
- Kann alle Arten von (Pseudo-)Zuständen enthalten (auch weitere hierarchische)
- Nach außen Abstraktion zu einzelnem Zustand
- Besitzt aber innere Struktur



UML: Zustandsdiagramm-Elemente Hierarchische Zustände (2)

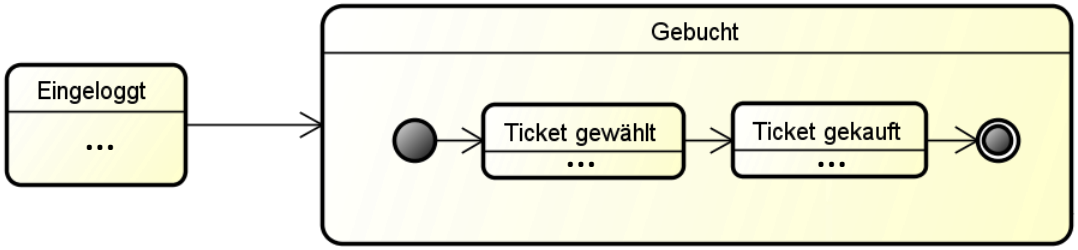
Beispiel:



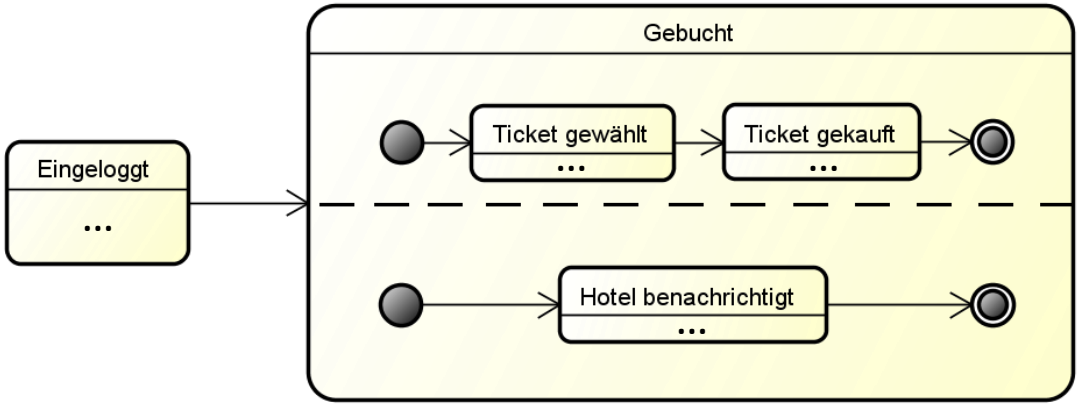
- Eintritt in hierarchischen Zustand:
  - Eingehende Transition endet am Rand des zusammengesetzten Zustands
  - Verarbeitung von „Gebucht“ beginnt am Startzustand des hierarchischen Zustands
- Verlassen des hierarchischen Zustands:
  - Variante 1: hierarchischer Zustand wird explizit über inneren Endzustand verlassen
  - Variante 2: hierarchischer Zustand wird explizit über ausgehende Transition verlassen
  - Variante 3: hierarchischer Zustand wird verlassen, sobald in einem der Unterzustände das Ereignis “Trigger” eintritt und die Bedingung “Guard” erfüllt ist

UML: Zustandsdiagramm-Elemente
Hierarchische Zustände (3)

- **Sequentielle Unterzustände:**



- **Parallele Unterzustände:**



UML: Zustandsdiagramm Konstruktion (1)

- **Idee:**
  - Für jeden nicht-trivialen Lebenszyklus ist ein Zustandsdiagramm zu erstellen

- **Notwendige Schritte (Standardelemente):**

  1. Prüfen, ob ein nicht-trivialer Lebenszyklus für eine Klasse existiert
     - Das gleiche Ereignis kann (in Abhängigkeit des aktuellen Zustandes) unterschiedliches Verhalten bewirken
     - Operationen sind nur in bestimmten Situationen (Zuständen) auf ein Objekt anwendbar und werden sonst ignoriert

  Entwicklung eines Zustandsautomaten:

  2. Zustände des Automaten bestimmen:
     - Ausgangsbasis ist der Startzustand
     - Ereignisse bestimmen, die zu Zustandswechseln führen
     - Folgezustände ermitteln

UML: Zustandsdiagramm Konstruktion (2)

- **Idee:**
  - Für jeden nicht-trivialen Lebenszyklus ist ein Zustandsdiagramm zu erstellen

- **Notwendige Schritte (Standardelemente):**

  3. Endzustände bestimmen (falls vorhanden):
     - Das Objekt hört auf zu existieren oder
     - Das Objekt existiert weiterhin, aber sein dynamisches Verhalten ist nicht länger von Interesse

  4. Aufzurufende Operationen bestimmen:
     - Zustandsabhängige Operationen aus dem Klassendiagramm eintragen
     - Evtl. weitere Operationen definieren, falls notwendig

  5. Zu berücksichtigende Ereignisse ermitteln:
     - Externe Ereignisse: vom Benutzer oder anderen Objekten
     - Zeitliche Ereignisse: Zeitdauer, Zeitpunkt
     - Intern generierte Ereignisse des betrachteten Objekts

UML: Zustandsdiagramm Konventionen

- Geeignete Zustandsnamen:
  - Name beschreibt eine bestimmte Zeitspanne
  - Name ist kein Verb
  - (Kann entfallen, falls Name keine zusätzliche Information enthält)
- Ist Objekt-Lebenszyklus konsistent mit Liste der Operationen?
  - Existiert für jede Operation mindestens ein Zustand, in dem das Objekt auf die entsprechende Botschaft reagieren kann?
  - Sind alle Aktivitäten und Aktionen auch Operationen des Klassendiagramms?
- Sind alle Transitionen korrekt eingetragen?
  - Ist jeder Zustand erreichbar?
  - Kann jeder Zustand (Endzustände ausgenommen) verlassen werden?

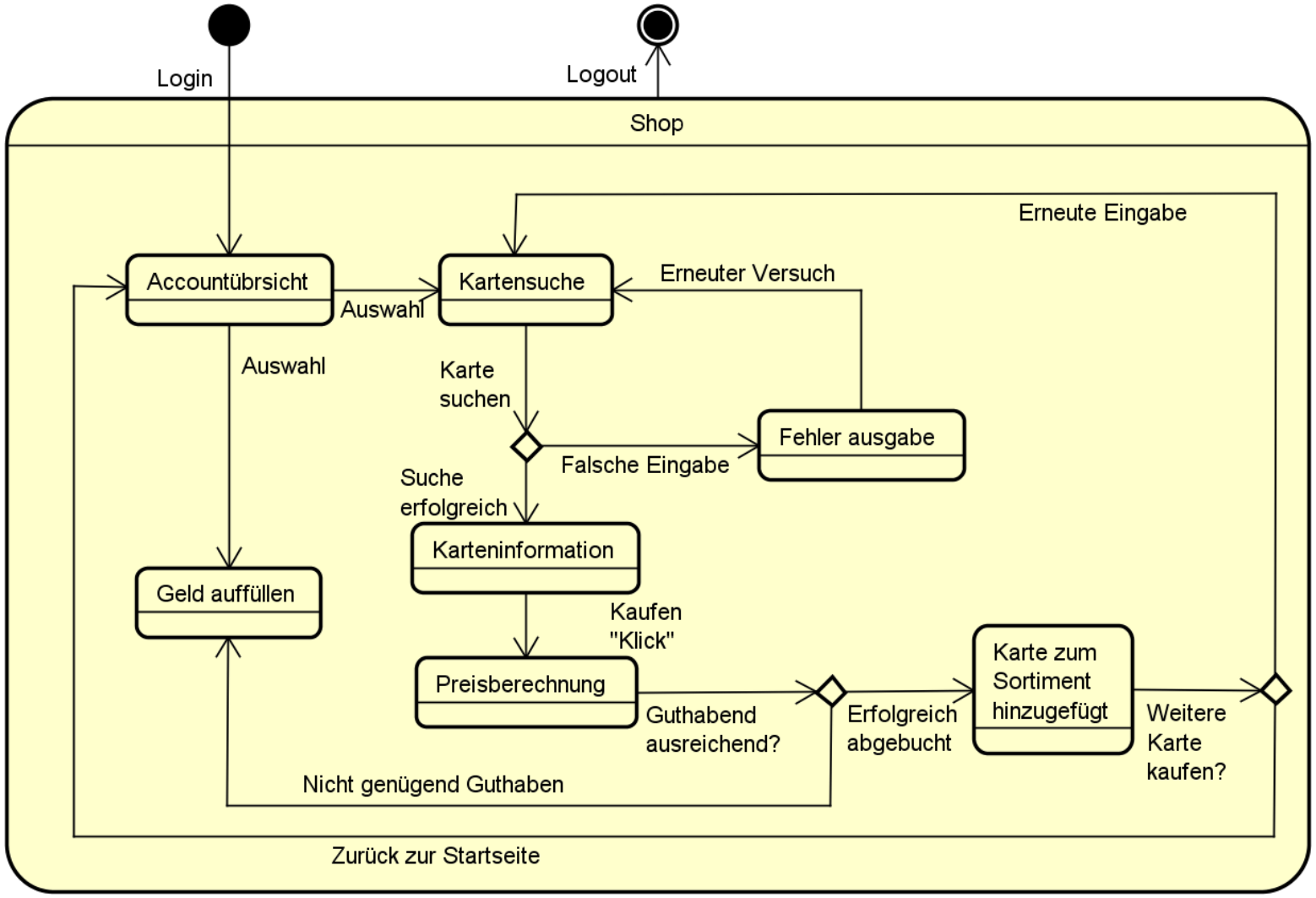
UML: Zustandsdiagramm

Modellieren Sie einen Zustandsautomaten für eine digitale Armbanduhr

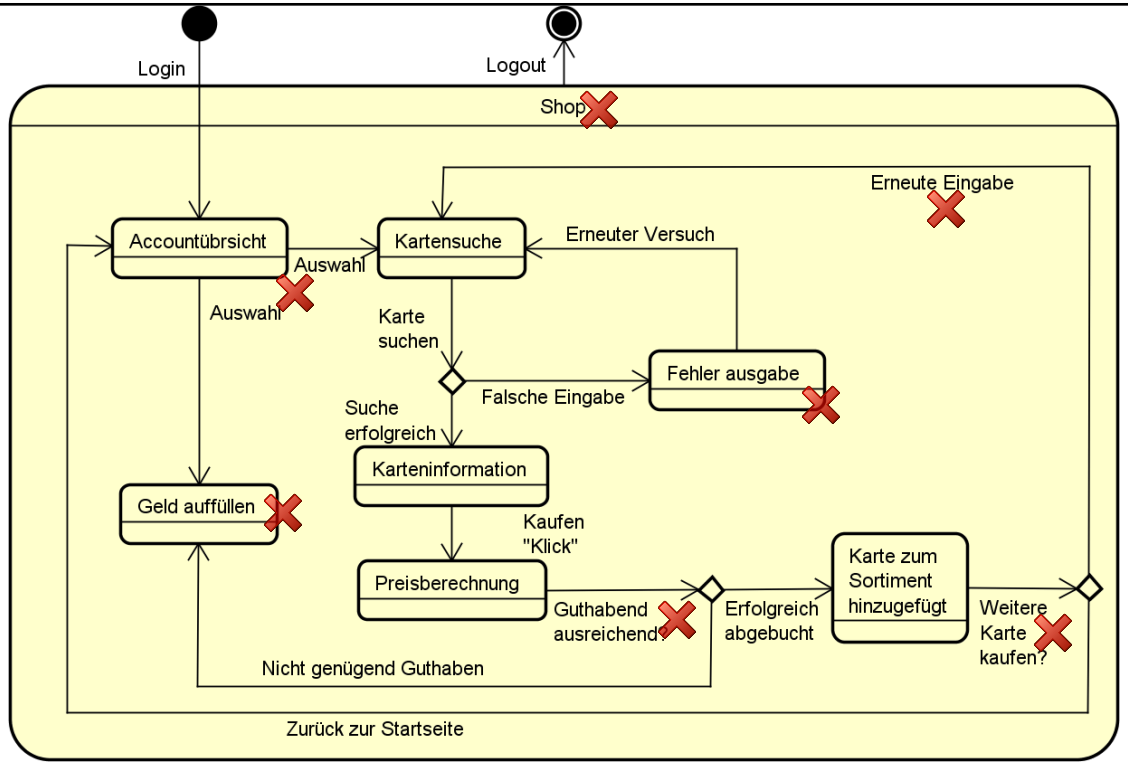
- Eine einfache digitale Armbanduhr hat eine Anzeige und zwei Knöpfe (A und B), um die Uhr zu stellen.
- Die Uhr hat zwei Betriebsarten: “Zeit anzeigen” und “Zeit einstellen”.
- Im Modus “Zeit anzeigen” werden Stunden und Minuten angezeigt.
- Stunden und Minuten sind durch einen blinkenden Doppelpunkt getrennt.
- Im Modus “Zeit einstellen” gibt es Untermodi “Stunden einstellen” und “Minuten einstellen”.
- Mit Knopf A werden die Modi gewählt, mit jedem Drücken wird zum nächsten Modus gewechselt.
- Es gilt die Reihenfolge: anzeigen, Stunden einstellen, Minuten einstellen, anzeigen, ….
- In Untermodi werden durch Drücken von B die Stunden/Minuten je um eine Einheit vorgestellt.
- Die Knöpfe müssen losgelassen werden, bevor sie ein anderes Ereignis veranlassen können.

Hinweis: versuchen Sie die Zustände mit Inhalt zu füllen (mindestens mit do-Akt.)

Zustandsdiagramme: Typische Modellierungsfehler



Zustandsdiagramme: Typische Modellierungsfehler



- Zustände sinnvoll benennen (z.B. ist “Shop” nicht der Zustand eines Systems)
- Keine nicht-deterministischen Kantenbeschriftungen (zwei Ausgänge mit derselben Beschr.)
- Vorsicht mit black holes (“Geld auffüllen” hat keine ausgehenden Kanten)
- Ein Zustandsdiagramm ist kein Aktivitätsdiagramm! (Es gibt z.B. keine Fragen an Decisions)
- Modelliere Kanten mit Triggern (evtl. Guards und Aktionen etc.; fehlt hier meistens)
- Modelliere Zustände aus, sofern möglich (`entry`, `do`, `exit`; hier ist kein Zustand ausmodelliert)

Literatur

- *UML 2 glasklar*, Chris Rupp et al., Hanser, 2012
- *Lehrbuch der Objektmodellierung*, Heide Balzert, Spektrum Akademischer Verlag, 2011