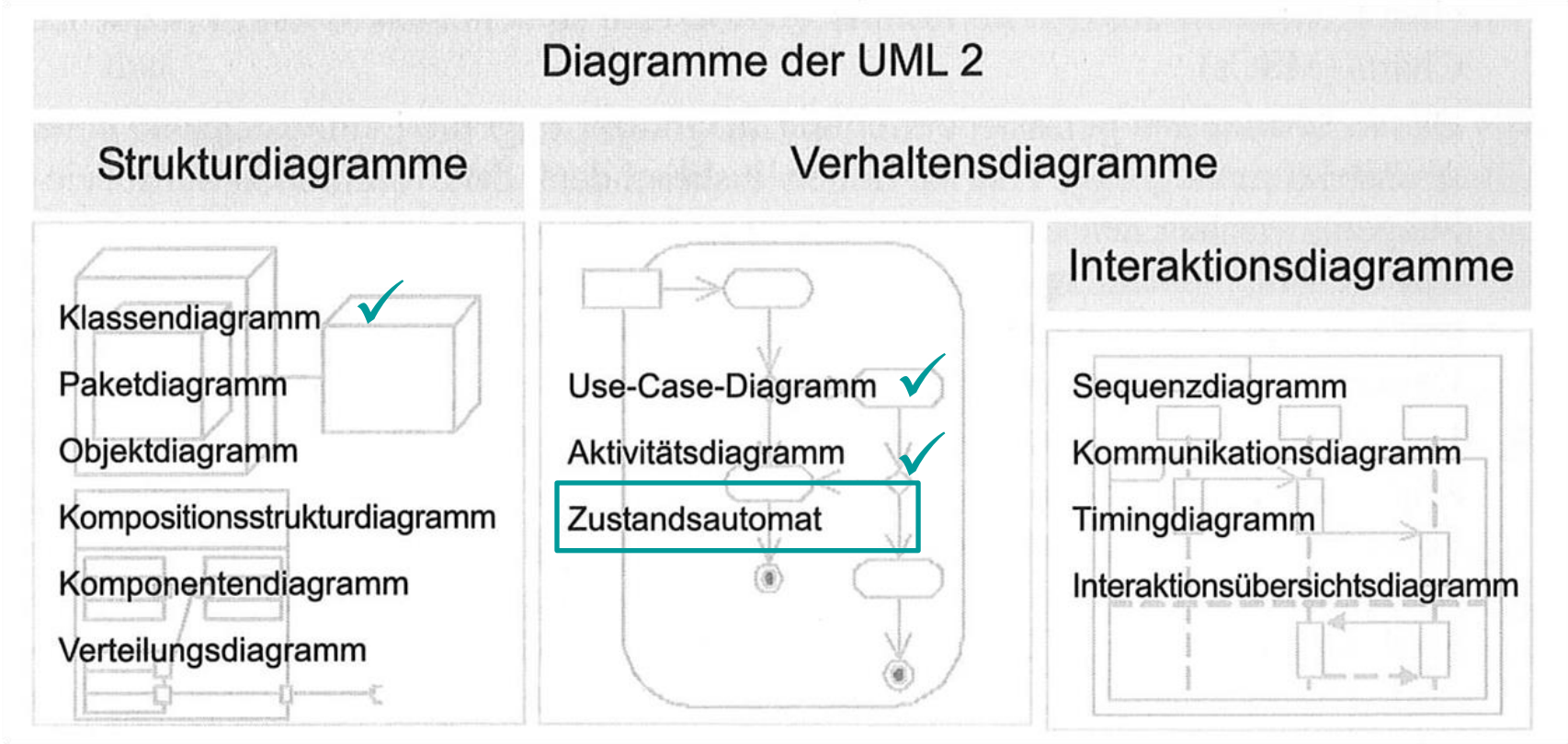
Softwaretechnik

Modellierung von Zustandsautomaten

Prof. Dr. Bodo Kraft

FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Übersicht UML-Diagramme
Zustandsautomat



Quelle: UML 2 glasklar, Chris Rupp

Motivation
Zustandsautomat

- Zeigen die (internen) Zustände eines Systems oder Objektes
- Zeigen die Ereignisse, auf die ein System in einem bestimmten Zustand reagiert
- *Zeigen die Zustandsübergänge, als Reaktion des Systems auf ein Ereignis*
- *Visualisieren Zustände/Ausprägungen von Klassen und*
- *Visualisieren Verhalten von Use-Cases*
- *Alternative für Aktivitätsdiagramme*

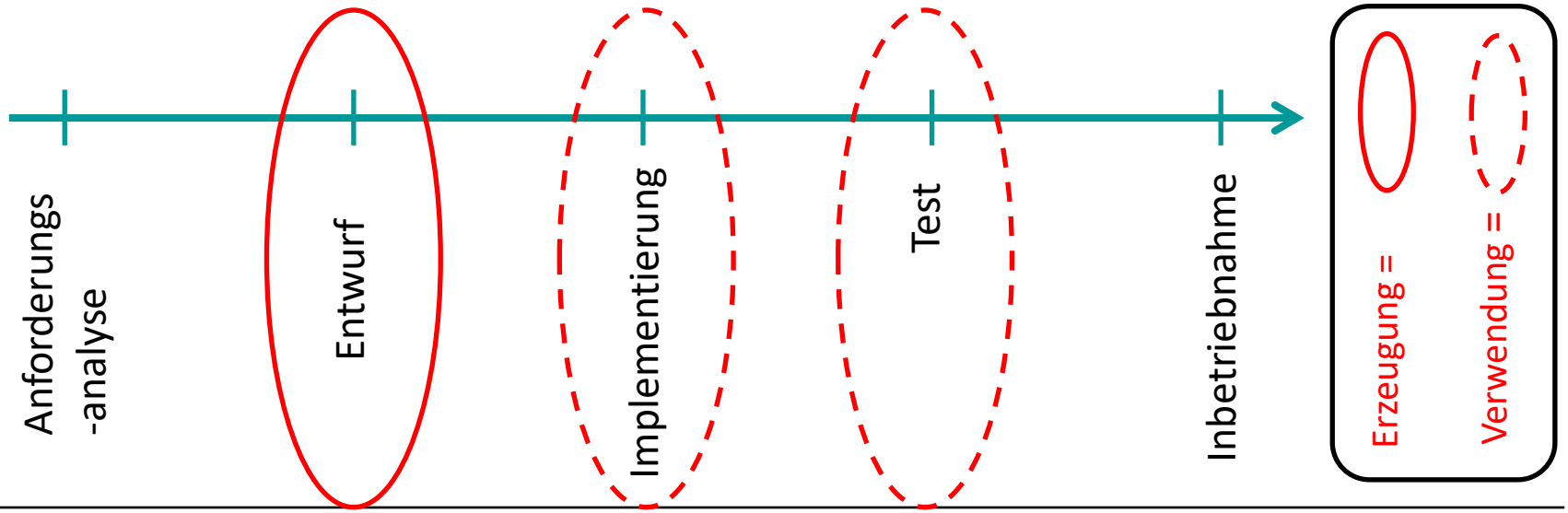
Liefern Antwort auf die Frage:

„Wie verhält sich das System in einem bestimmten Zustand bei gewissen Ereignissen?"

Zeitliche Einordnung in SW-Lifecycle
Zustandsautomat

Bei welchen Schritten des Software-Lifecycle sind Zustandsdiagramme(SM=state machine) relevant?

- SM verdeutlichen konkrete interne Abläufe eines Systems mit Fokus auf existierende Zustände
- Visualisieren Programmabläufe in Form von Zustandsübergängen
- Visualisieren Verhalten von Klassen

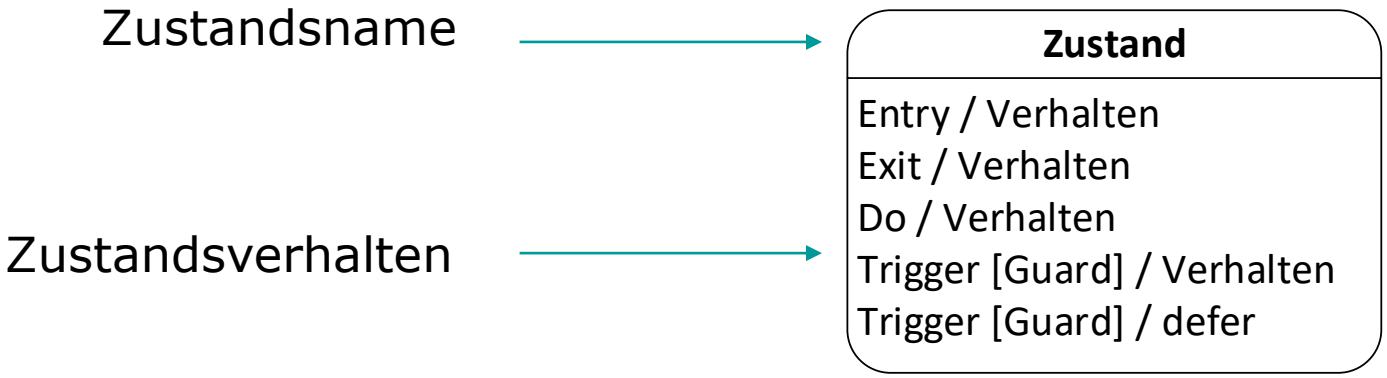


Zustände
Notationselemente

- Zustand: Situationen, in denen sich das Objekt nicht ändert

- Ein Objekt, dass in einem konkreten Zustand vorliegt, zeigt ein konstantes Verhalten.

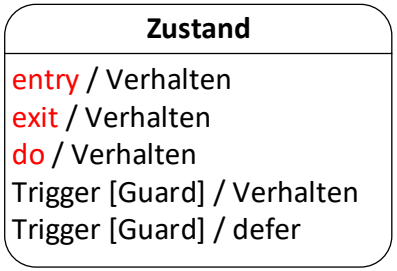
Standardnotation eines Zustands in der UML [ohne explizites Verhalten]

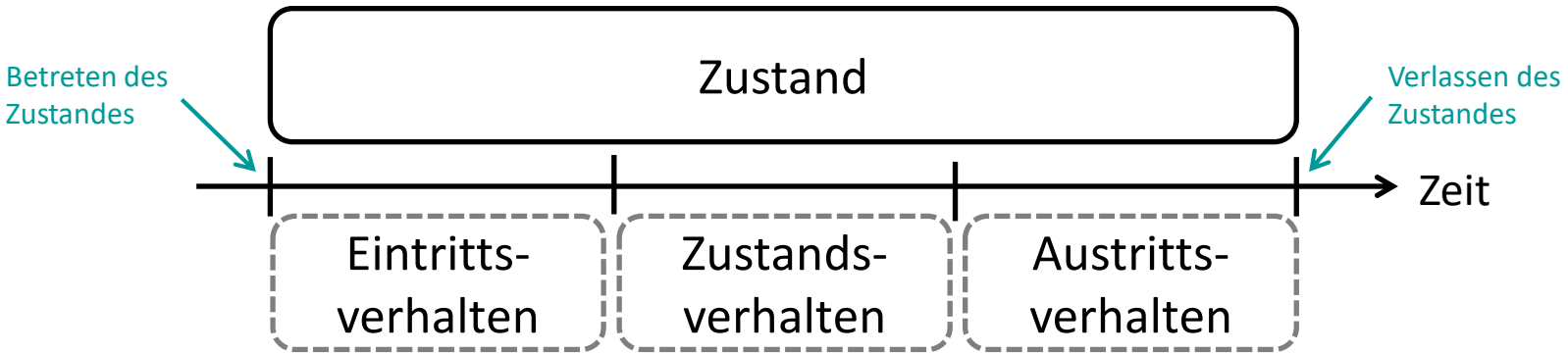


Standardnotation eines Zustands in der UML [mit explizitem Verhalten]

Auslöser und Verhalten
Zustände im Detail

- Zustandsverhalten werden mit einem Schlüsselwort eingeleitet
- Es sind drei Verhalten definiert:
  a) Eintrittsverhalten: entry
     wird genau 1 mal ausgeführt bei „aktiv werden" des Zustandes
  b) Zustandsverhalten: do
     wird permanent ausgeführt (nach a), aber vor c))
  c) Austrittsverhalten: exit
     wird genau 1 mal ausgeführt bei „inaktiv werden" des Zustandes

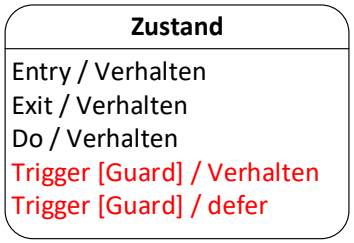




Transitionen/Zustandsübergänge
Notationselemente

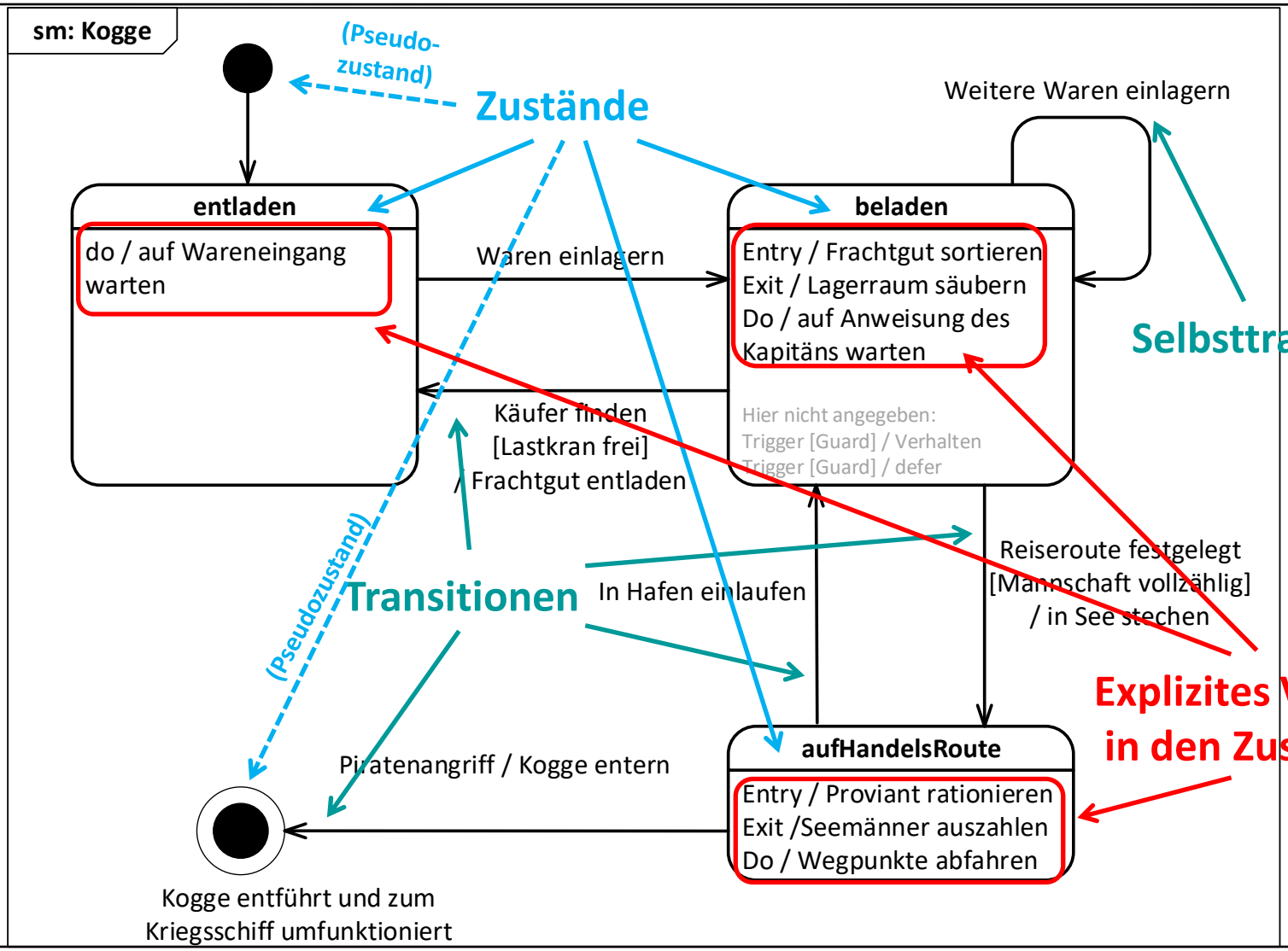
- Transitionen beschreiben die Beziehungen zwischen den Einzelzuständen eines Automaten
- Darstellung als gerichtete und beschriftete Kante
- Syntax:

Hinweis: Das interne Verhalten von Zuständen kann auch über Trigger beschrieben werden

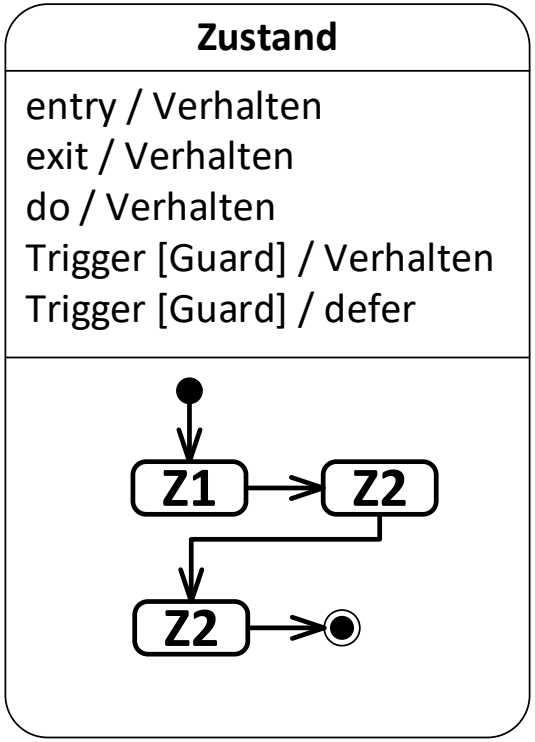


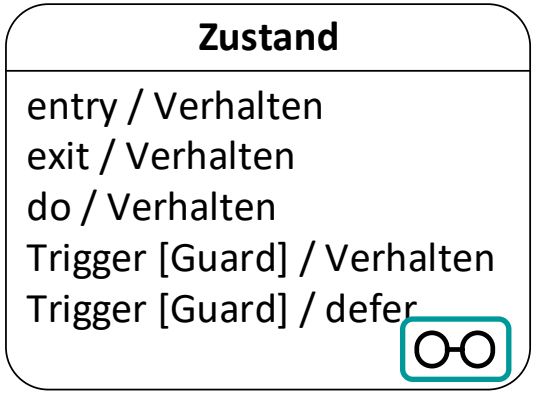
- Transitionen teilen sich in drei Bereiche auf:
  a) Trigger (oder Auslöser)
     - stoßen eine Transition an
     - Mehrere Trigger werden durch Komma getrennt
  b) Guard
     - Boolsche Bedingung, die entscheidet, ob Transition tatsächlich durchlaufen wird.
  c) Verhalten
     - Das Verhalten wird beim Durchlaufen der Transition durchgeführt

Konkrete Darstellung am Beispiel einer Kogge
Notationselemente

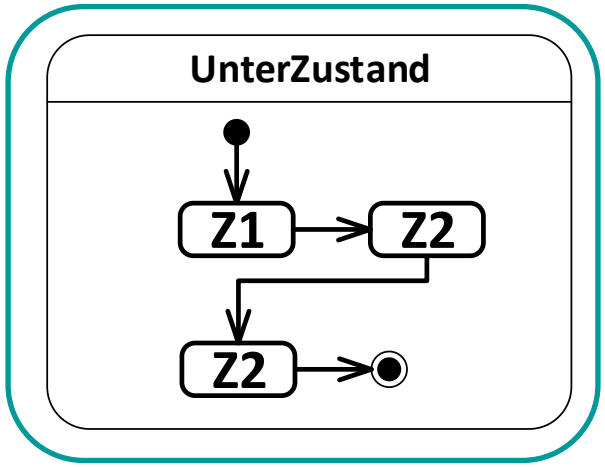


Substates
Notationselemente





Zustände können erweitert werden durch Unterzustände.

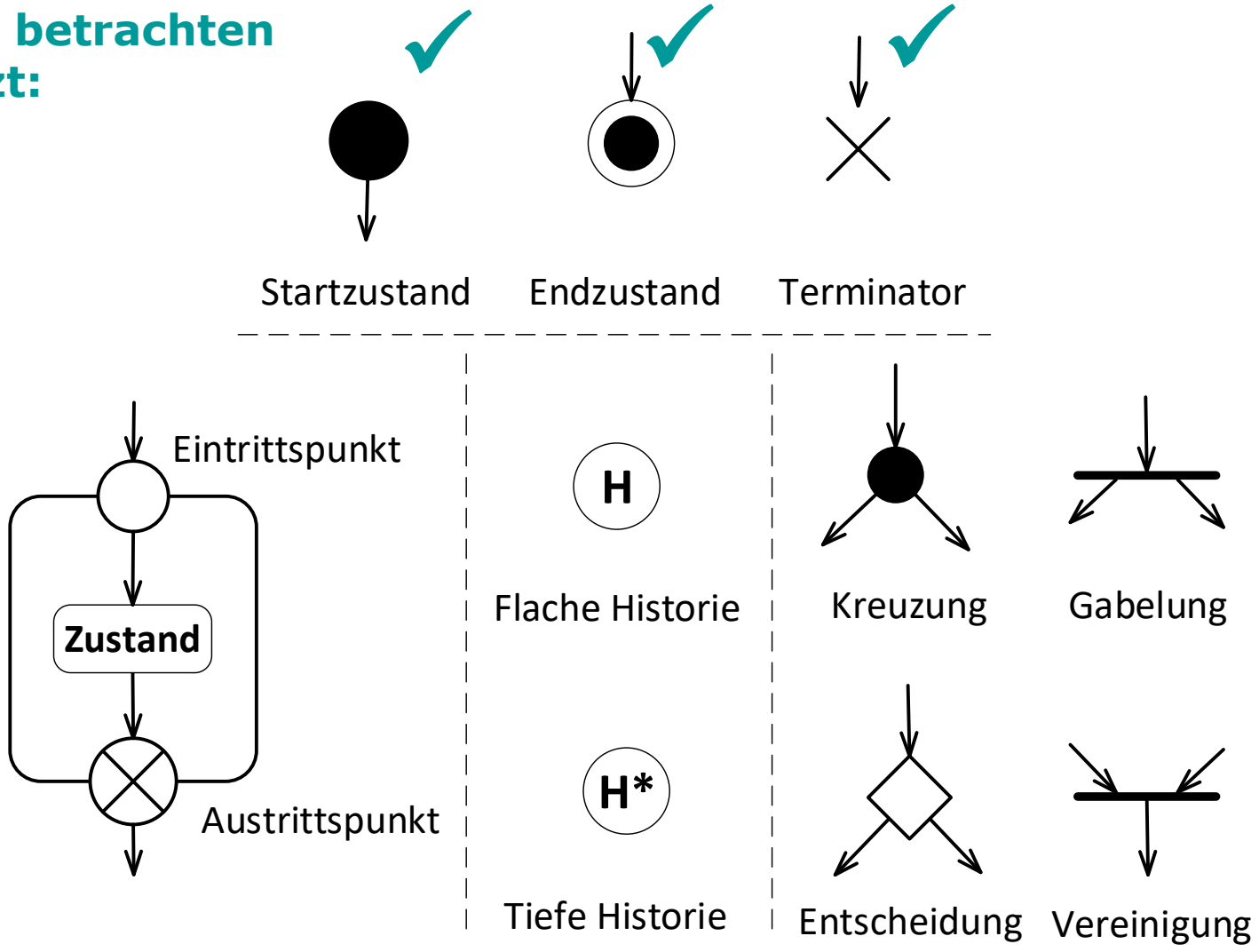


Unterzustände werden dabei

- direkt in einen Zustand hineingezeichnet
- ausgelagert (dann Bezug bspw. durch Namen deutlich machen)
  - ➢ Name := <Unterzustand>: sm <Hauptzustand>

Übersicht Pseudozustände
Notationselemente

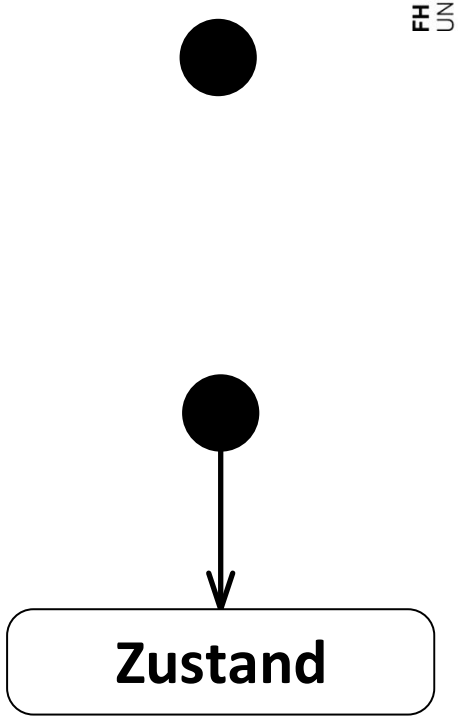
Wir betrachten jetzt:



Startzustand
Notationselemente

Ein Startzustand (initial pseudostate) …

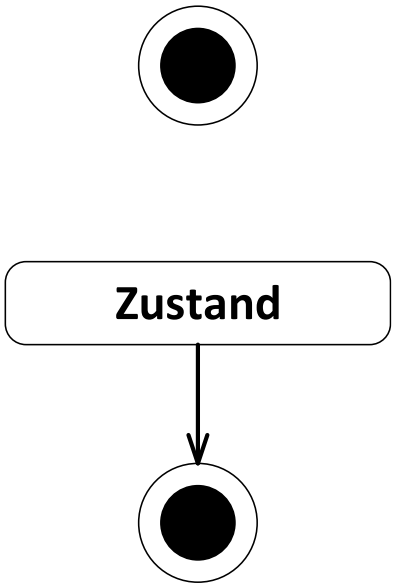
- Ist (semantisch) vergleichbar mit Startknoten aus Aktivitätsdiagramm
- Hat gleiche Notation wie Startknoten im Aktivitätsdiagramm
- Verweist auf genau einen Zielzustand (Eindeutigkeit)
- Ausgehender Kante darf keine Bedingung beinhalten (untriggered)
- Keine eingehenden Transitionen erlaubt
- Genau 1 Startzustand pro Zustandsautomat (Global eindeutig)



Endzustand
Notationselemente

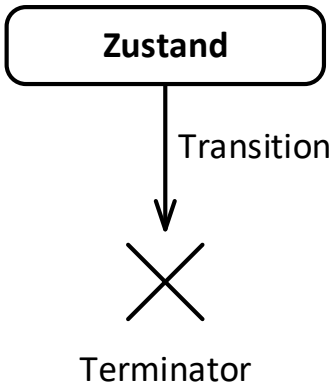
Der Endzustand (final state)

- Hat gleiche Standardnotation wie Endknoten aus Aktivitätsdiagramm
- Beendet gesamten Zustandsautomaten
- D.h. kein weiteres Verhalten wird ausgeführt
- Keine ausgehenden Transitionen erlaubt
- Genau 1 Endzustand pro Zustandsautomat (Global eindeutig)



Terminatoren (I)
Notationselemente

Funktionieren ähnlich wie Endzustände



Aber:
Bei Erreichen wird **gesamtes Objekt zerstört**.
(vgl. Endzustand beendet nur den Zustands-automaten)

Betonung auf Laufzeitverhalten.

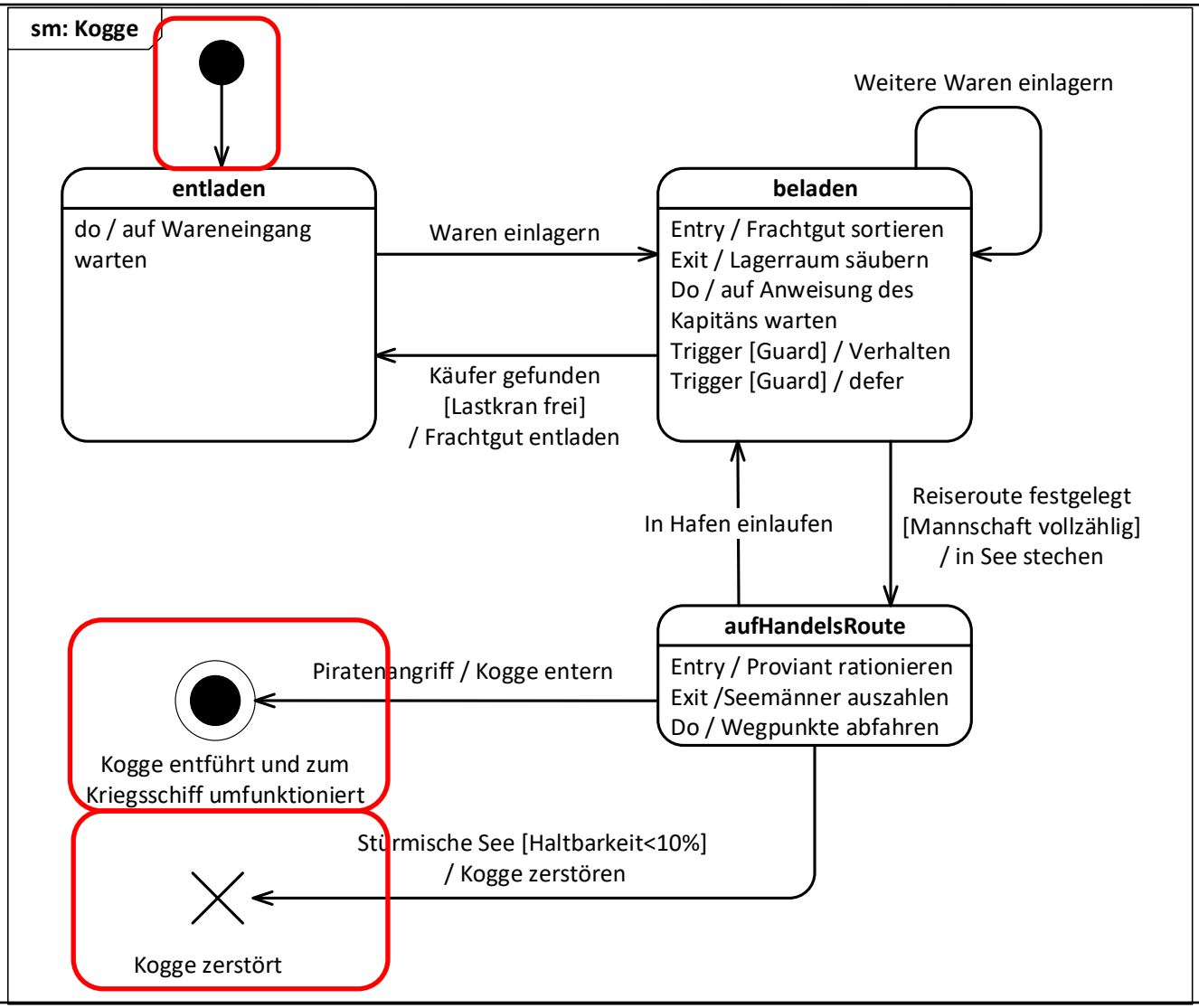
In der Regel Verwendung, wenn Objekte dynamisch erzeugt und/oder zerstört werden.

Notation wie Objektzerstörung im Sequenzdiagramm

Beispiel:

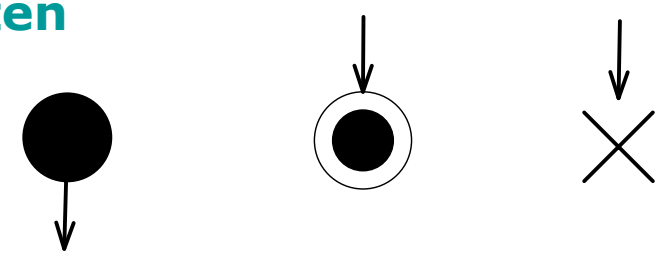
Zustandsbeschreibung terminierter Automat → Kogge zerstört

Terminatoren (II)
Notationselemente



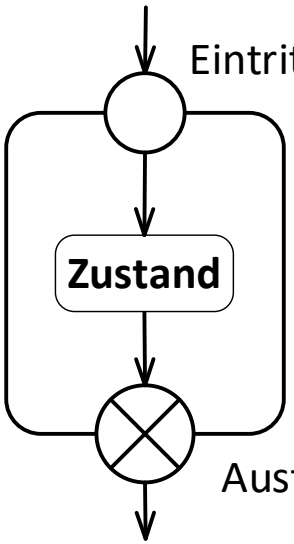
Übersicht Pseudozustände
Notationselemente

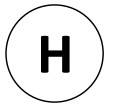
Wir betrachten jetzt:



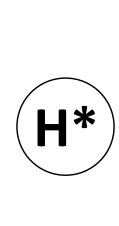
Startzustand Endzustand Terminator

Zur Erinnerung: Tokensemantik aus dem Aktivitätsdiagramm

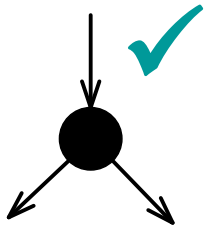




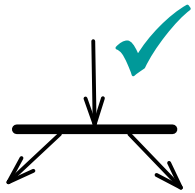
Flache Historie



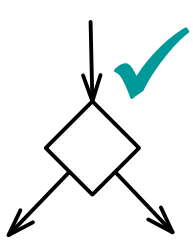
Tiefe Historie



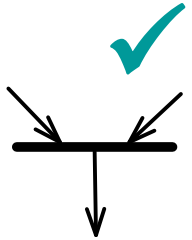
Kreuzung



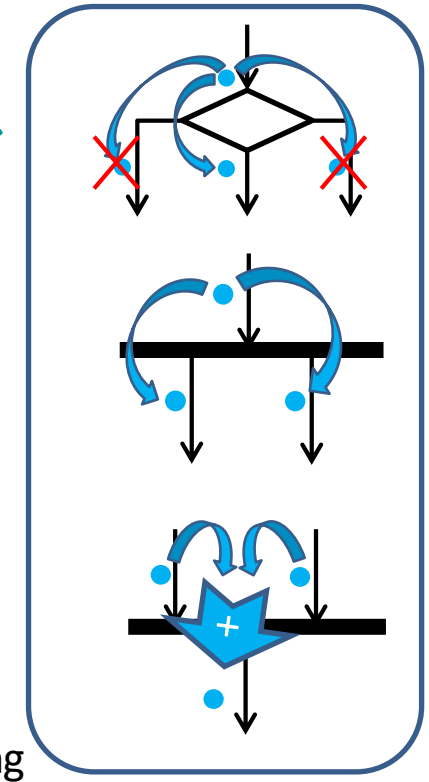
Gabelung



Entscheidung

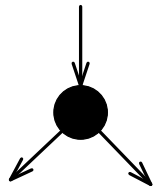


Vereinigung

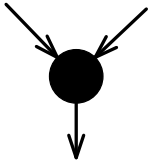


Kreuzungspunkte (I)
Notationselemente

Kreuzungspunkte (Junction Points) vereinfachen Transitionspfade, indem sie gemeinsame Teile bündeln.



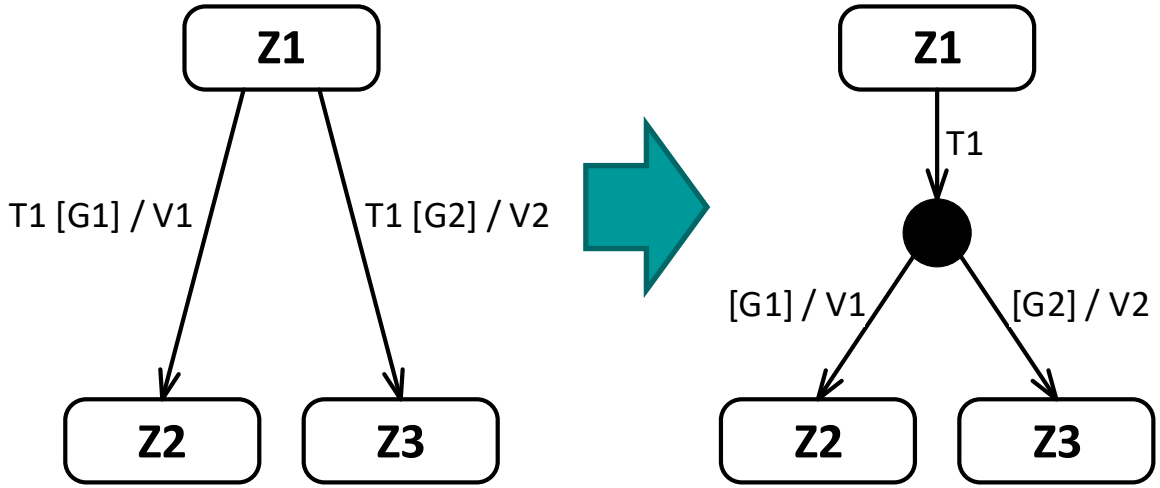
Kreuzungspunkte



Man unterscheidet drei Verarbeitungsarten:

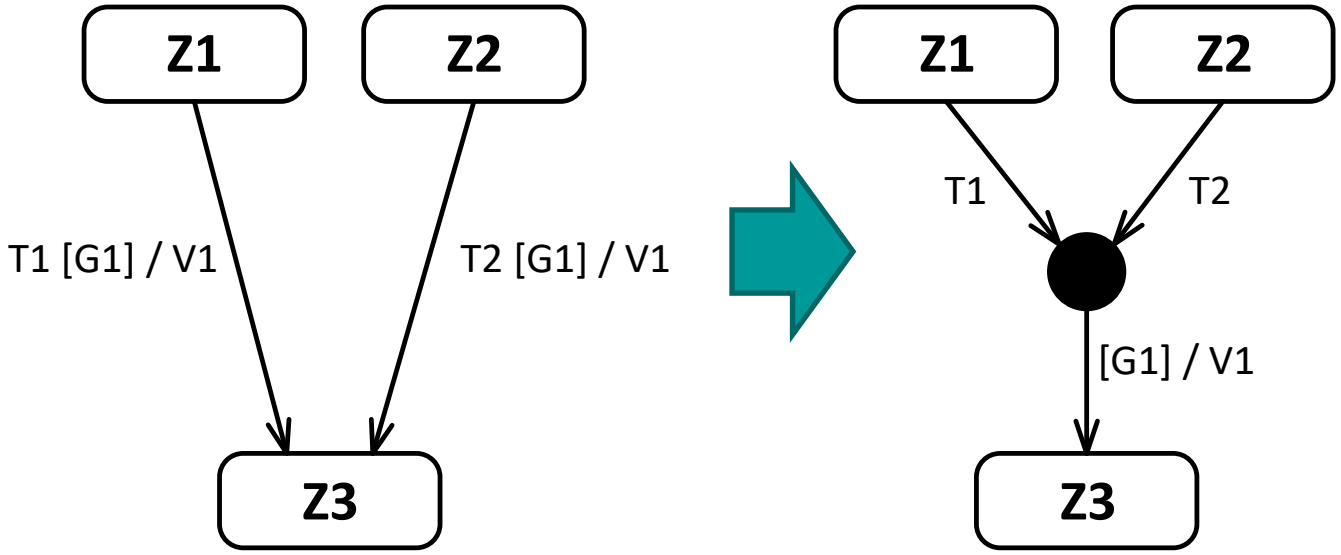
a) Vereinigung von Triggern
b) Vereinigung von Bedingung & Verhalten
c) Kombination aus beidem

Beispiel 1: Kreuzungspunkte für Trigger



Kreuzungspunkte (II)
Notationselemente

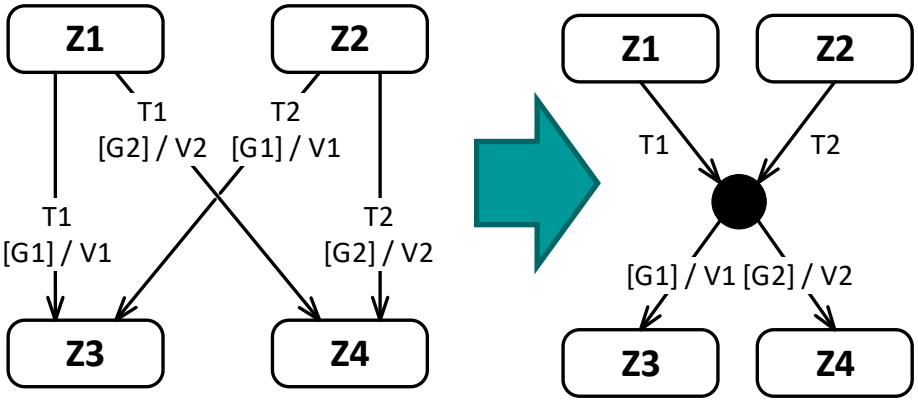
Beispiel 2:
Kreuzungspunkte bei gemeinsamen Guards bzw. Verhalten



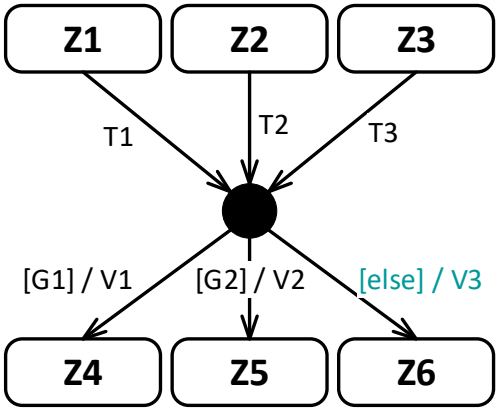
Kreuzungspunkte (III)
Notationselemente

FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Beispiel 3: Kombinationen von Transitionen



Beispiel 4: Behandlung von Restfällen

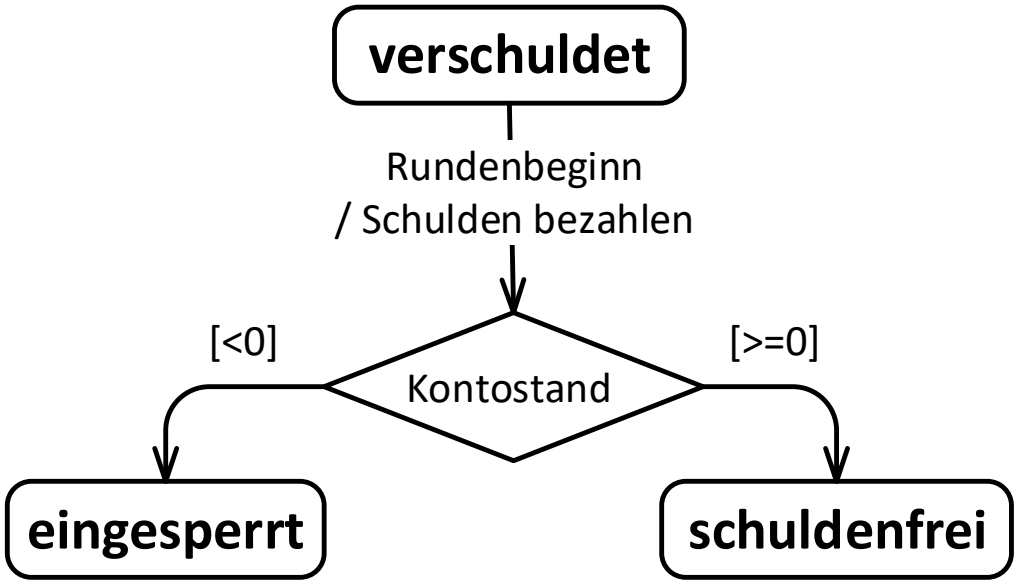


Bündelung von „Restfällen“

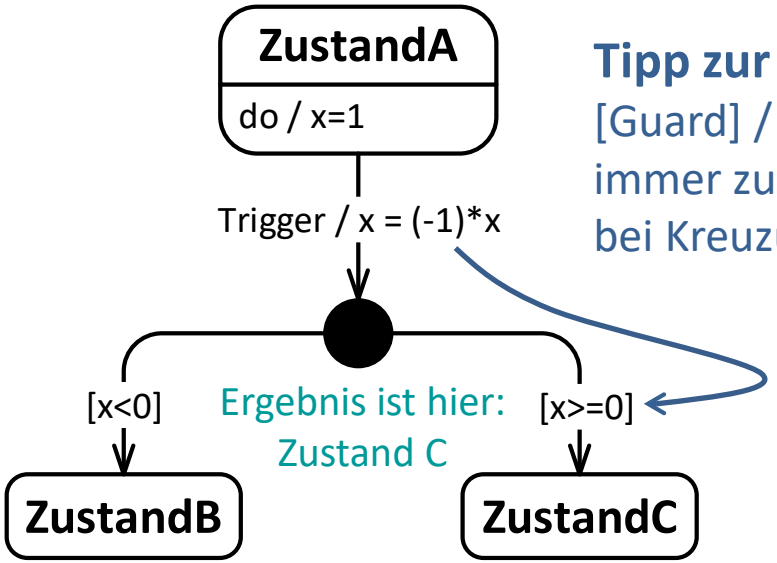
Entscheidungspunkte (Choice)
Notationselemente

- Entscheidungsknoten werden mithilfe des Raute-Zeichens notiert (analog zu den Aktivitätsdiagrammen)
- Das Entscheidungskriterium kann in die Raute gezeichnet werden
- Die Bedingung wird als Guard notiert.

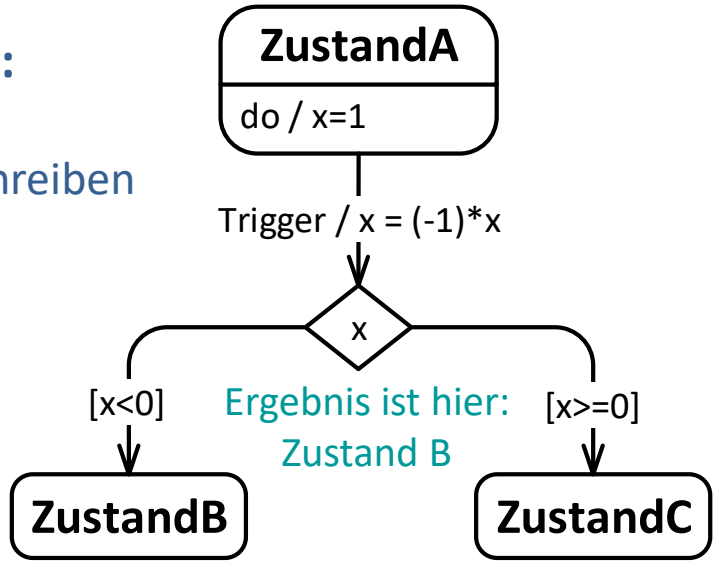
Beispiel:



Entscheidungs- vs. Kreuzungspunkte
Notationselemente

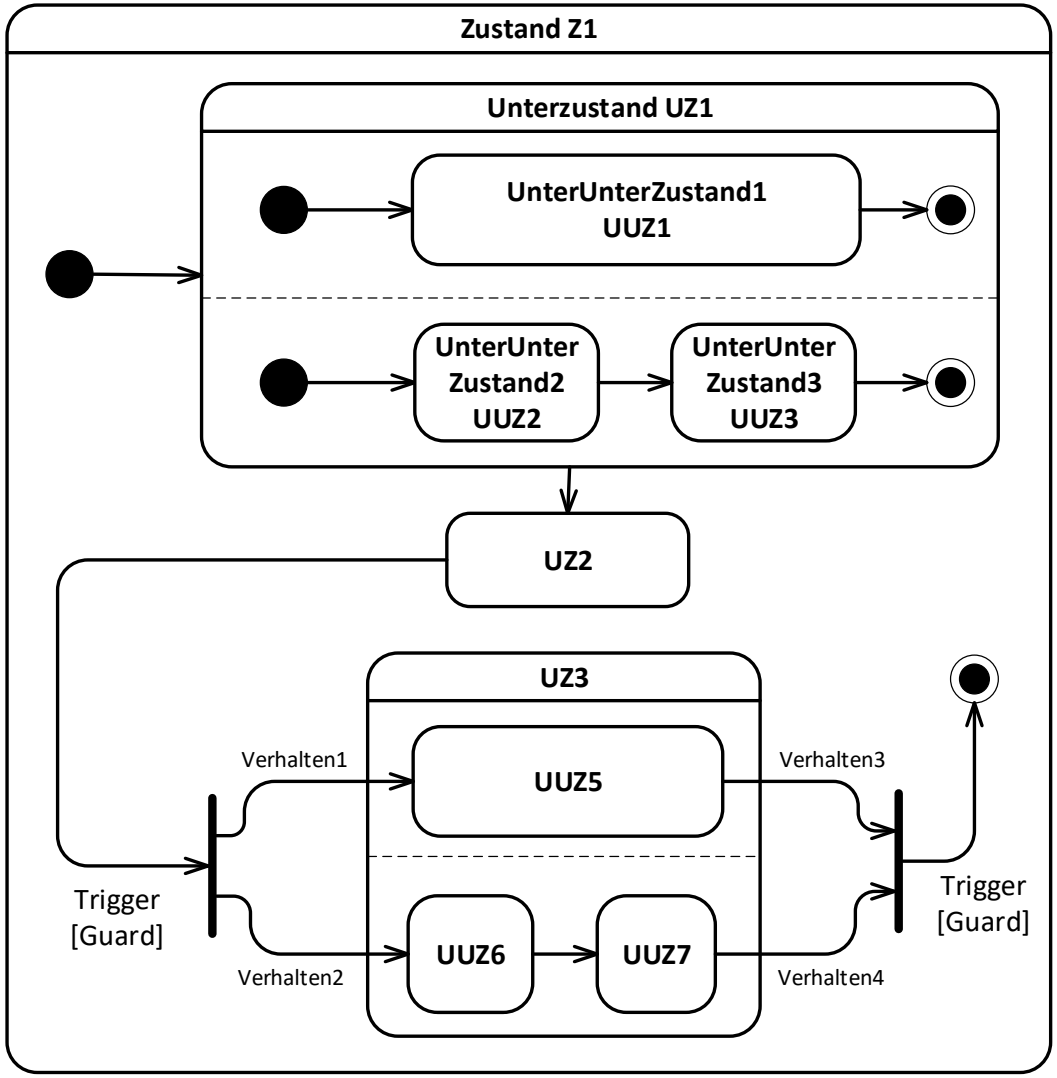


Tipp zur Lesbarkeit:
[Guard] / Verhalten
immer zusammen schreiben
bei Kreuzungen



Kreuzung	Entscheidung
Auslöser + Bedingung im Vorfeld ausgewertet. Danach erst Verhalten ausgeführt	Verhalten auf Trigger wird erst ausgeführt. Anschließend Bedingung geprüft und ausgewertet
Teilweise Abdeckung reicht (Zustand A wird ggfs. Nicht verlassen)	Komplette Abdeckung des Wertebereichs der Bedingung
Alle Bedingungen disjunkt	Alle Bedingungen disjunkt

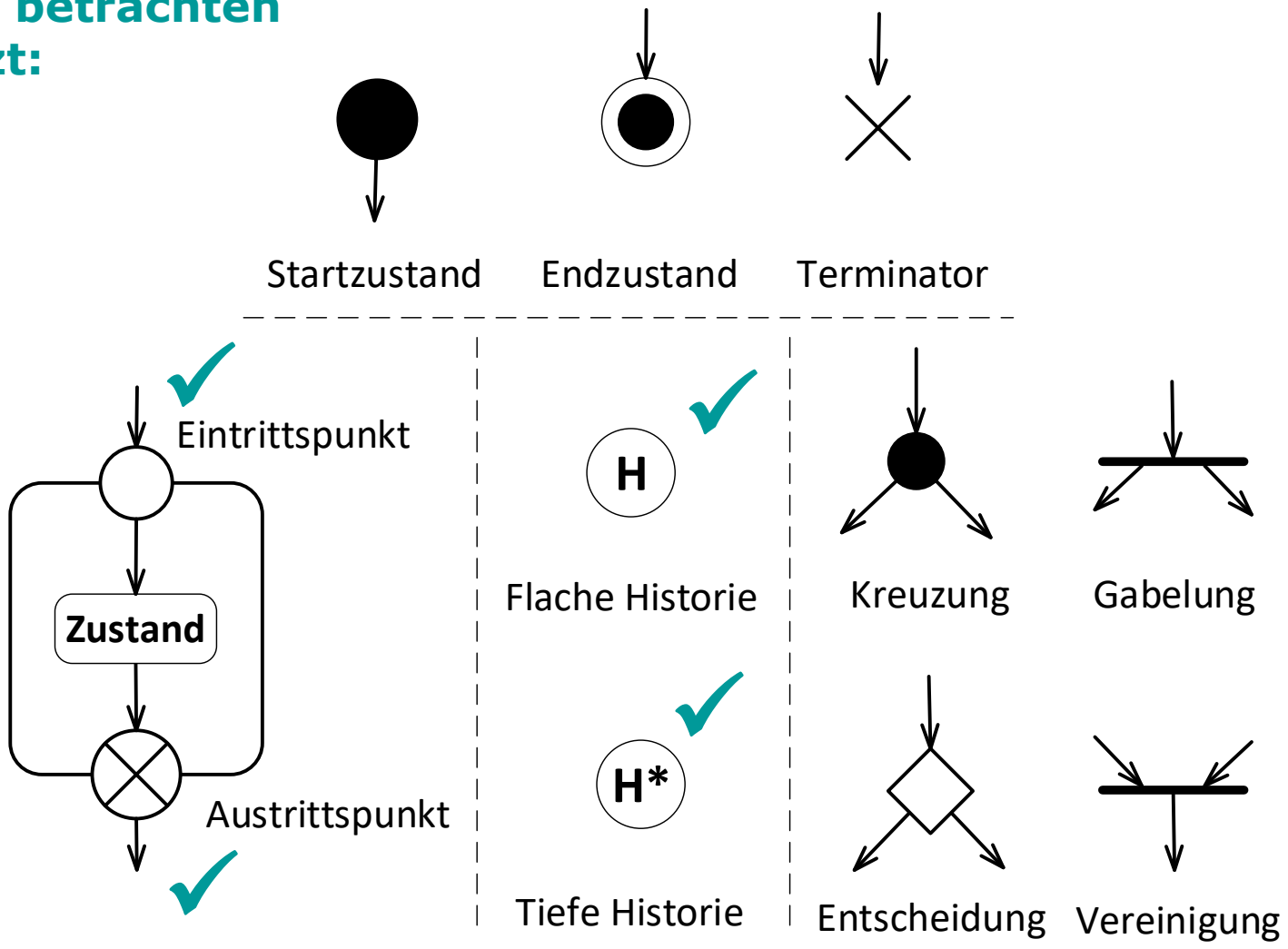
Vereinigung (join) & Gabelung(fork)
Notationselemente



- Man kann parallele Zustände modellieren
- Jeder „Parallelzweig" muss sich immer in einem eindeutigen Zustand sein.
- Darstellung
  - über Trennlinie
  - Über Join/Fork

Übersicht Pseudozustände
Notationselemente

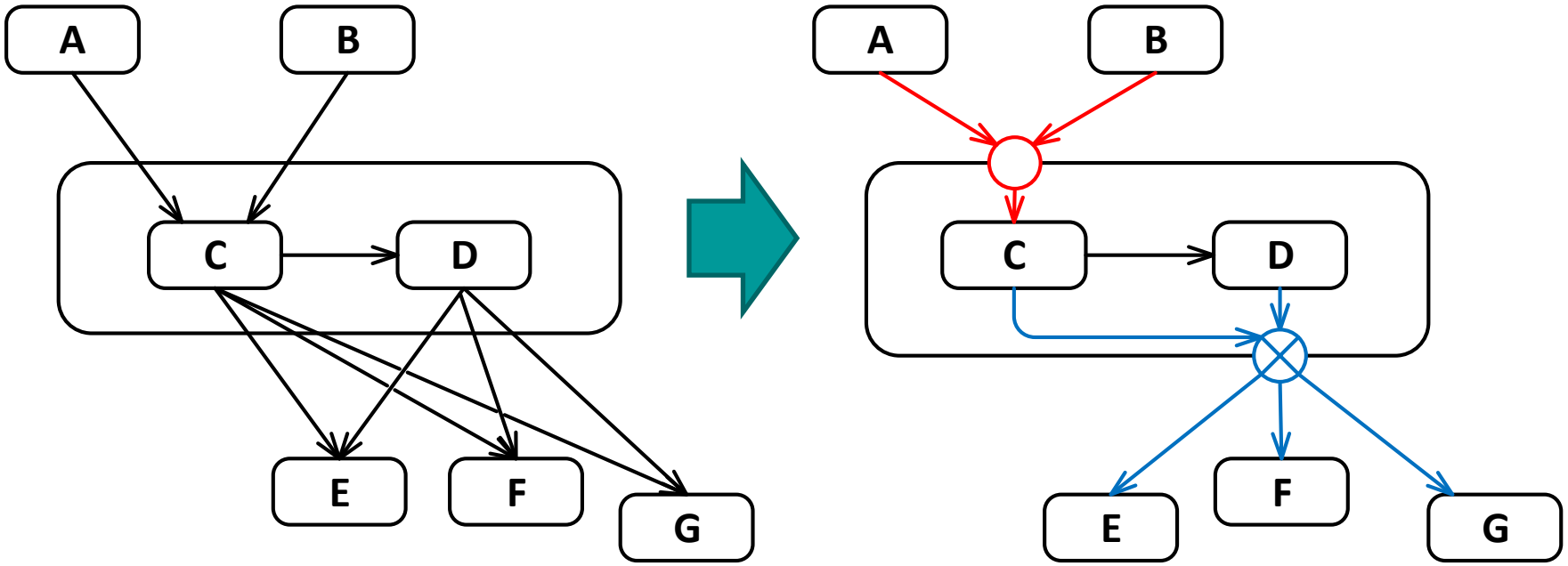
Wir betrachten jetzt:



Eintritts- und Austrittspunkt
Notationselemente

- Ein/Austrittspunkte dienen der Übersichtlichkeit
- Hilfreich bei ineinander verschachtelten Zuständen

- Jede Region hat maximal 1 Eintritts- und 1 Austrittspunkt
- Zählen als Start- bzw. Endzustand in dem betroffenen Diagrammbereich



Historie
Notationselemente

Man unterscheidet flache und tiefe Historienzustände (H bzw. H*)

Historienzustände(H bzw. H*) agieren als „Merker“:

- Bei Verlassen eines Zustands wird sich letzter aktiver Unterzustand gemerkt.
- Beim Betreten von H(*) wird gemerkter Zustand aktiv gesetzt
- Erreichung eines Endzustands löscht Historie
- Bei leerer Historie wird Default Entry aufgerufen (verweist auf Startzustand)

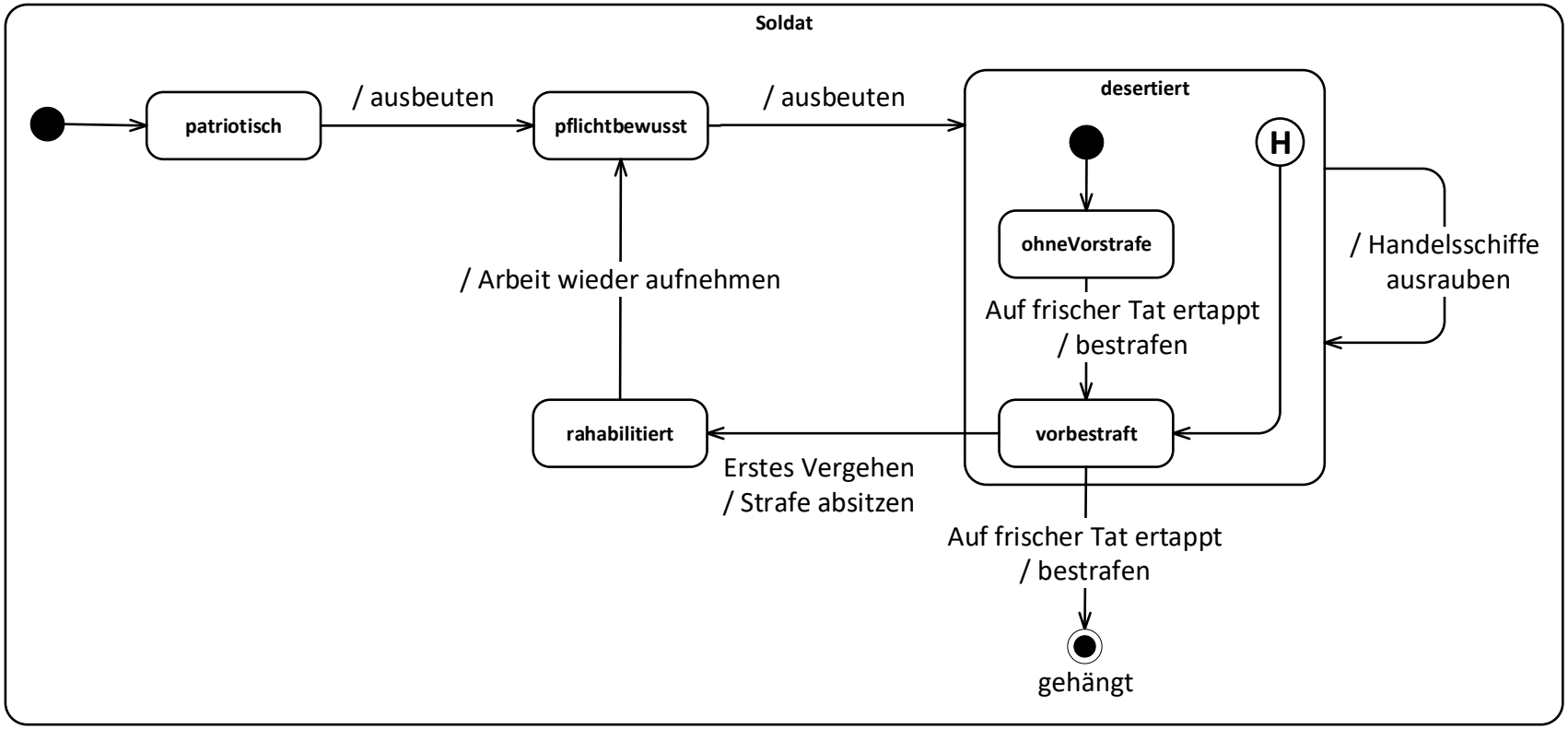
Historienzustände müssen eindeutig sein in ihrem Kontext

➔ Max. 1 Historienelement pro Zustand auf gleicher Ebene

➔ Multiple Historienelemente über Hierarchie möglich

Historie (II)
Notationselemente

Beispiel: Zustandsübergänge mit flacher Historie (H)



Historie (II)
Notationselemente

Unterschied zwischen flacher und tiefer Historie (H <--> H*):

- H merkt sich nur Zustand der gleichen Ebene
- H* merkt sich auch alle Unterzustände beliebiger Tiefe des aktuellen Zustandes

Hinweis:
Vergleichbar mit Konzept der flachen/tiefen Kopie eines Javaobjekts.

Präsenzaufgabe

Interne Zustände von Objekten

Präsenzaufgabe1: Lebenssituation Spieler

Modellieren Sie folgendes Szenario als Zustandsdiagramm:

Spieler beginnen als ledige Person. Sie sind dabei Partner suchend. Nach der Vermählung zählen sie als glücklich verheiratet. Dieser Zustand hält solange an, bis der Ehepartner fremd geht. Der Spieler wird daraufhin unglücklich verheiratet. Er kann durch Scheidung wieder ledig werden. Falls der Partner stirbt, gilt der Spieler als verwitwet. In diesem Zustand kann er erneut heiraten.

Welche Zustände werden benötigt?

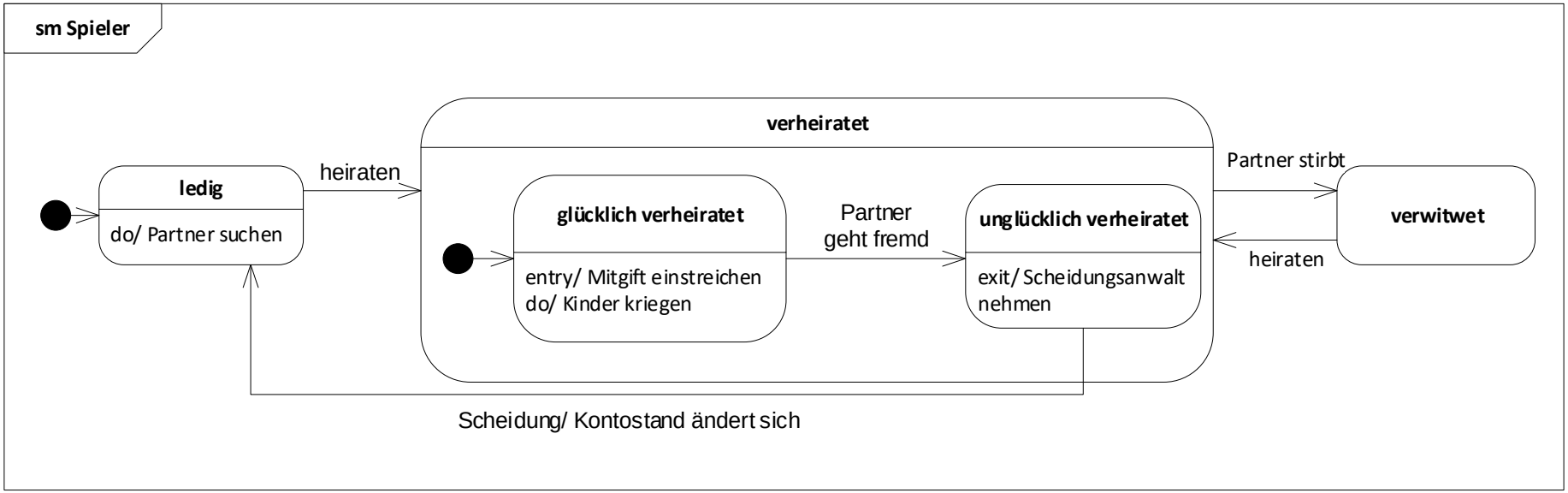
Wo macht ein übergeordneter Zustand Sinn?

Kann man Zustände zusammen fassen?

Interne Zustände von Objekten

Präsenzaufgabe1: Lebenssituation Spieler

Lösung



Präsenzaufgabe

Interne Zustände von Objekten

Präsenzaufgabe2: Reputation des Spielers

Jeder Spieler hat eine Reputation und startet als unbedeutender Bürger. Sollte er heiraten (reiche Frau) oder Geld für den Dom spenden, kann er zu einem angesehenem Bürger aufsteigen. Ein angesehener Bürger kann durch weitere Spenden beliebt werden.

Scheidung führt zum Rufverlust. Beliebte Bürger (Prominente) werden anschließend sogar verachtet, angesehene Bürger fallen „nur" in die Bedeutungslosigkeit zurück.

Angesehene und unbedeutende Bürger sind auch bei dubiosen Geschäften anzutreffen. Wer dabei erwischt wird, gilt als verachtet. Einziger Ausweg aus der Verachtung ist eine Geldspende für den Dom(Aufstieg zu unbedeutend).

Beliebte Bürger verlieren ihr Ansehen, wenn ihr Kontostand unter wohlhabend sinkt. Sie sind dann nur noch angesehen.

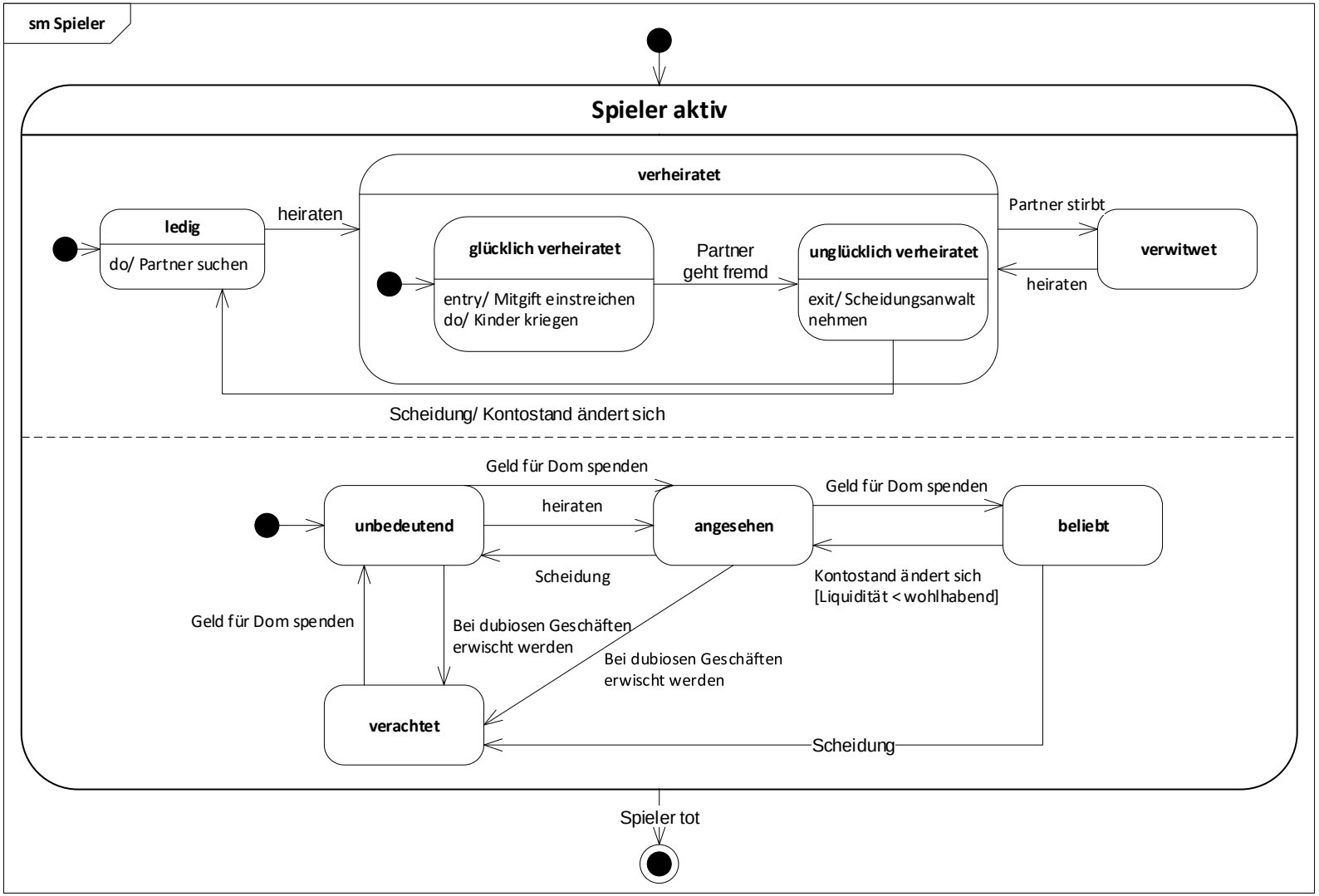
Erweitern Sie das letzte Diagramm.

Wo macht hier eine Bedingung (guard condition) Sinn?

Interne Zustände von Objekten

Lösung

Präsenzaufgabe2: Reputation des Spielers



Präsenzaufgabe

Interne Zustände von Objekten

Präsenzaufgabe3: Liquidität des Spielers

Jeder Spieler hat einen finanziellen Status.

Nach jeder Änderung des Kontostandes(KS) wird der Liquiditätsstatus festgelegt:

- KS < 0: bankrott
- KS >= 0: liquide
- KS > 250.000: wohlhabend
- KS > 1.000.000: reich.

Spieler starten mit 10.000 Goldmünzen.

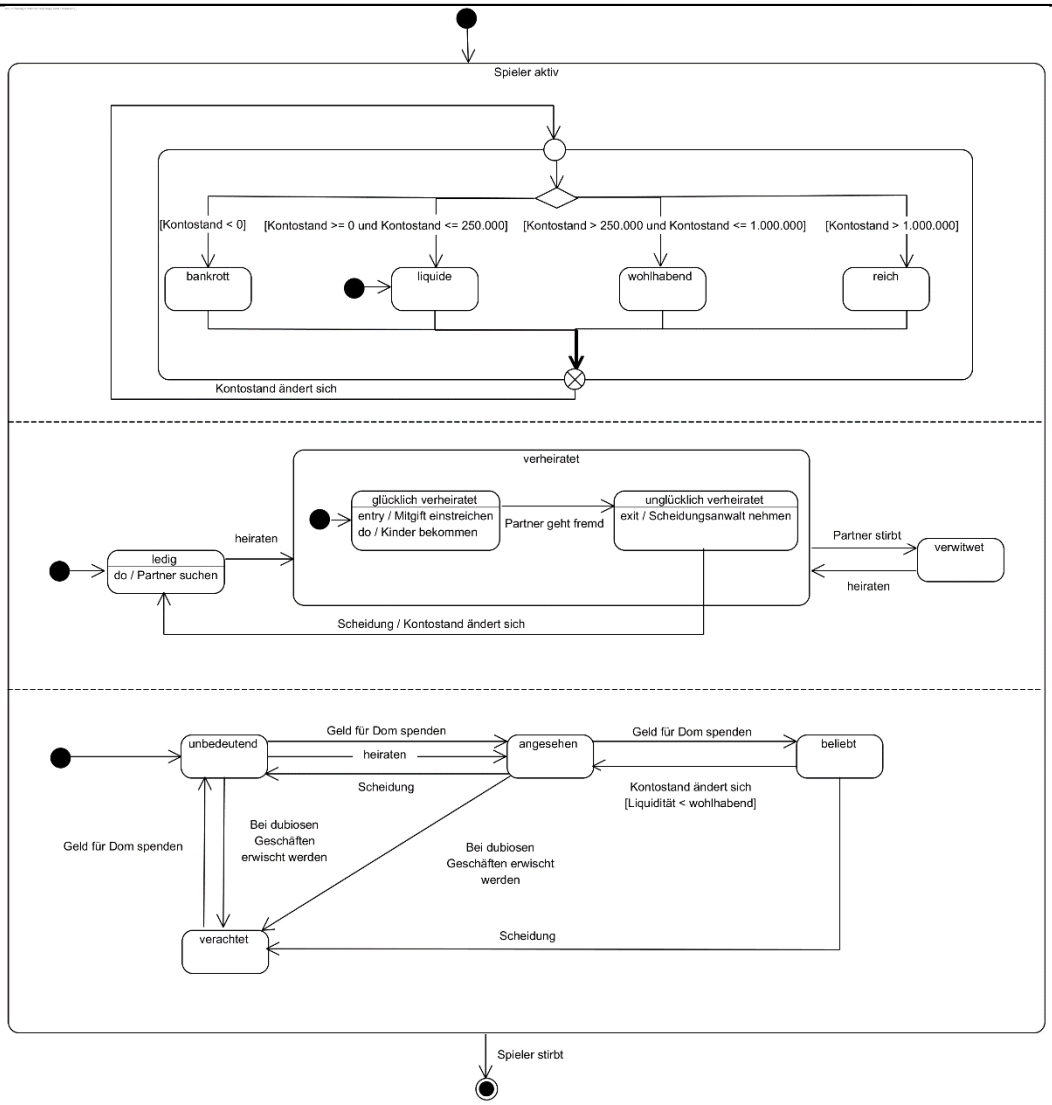
Erweitern Sie Ihr Diagramm erneut.

Verfeinern Sie die existierenden Zustände des Gesamtdiagramms mit Ein- und Austrittspunkten.

Interne Zustände von Objekten

Präsenzaufgabe3: Liquidität des Spielers

Lösung



Vielen Dank!