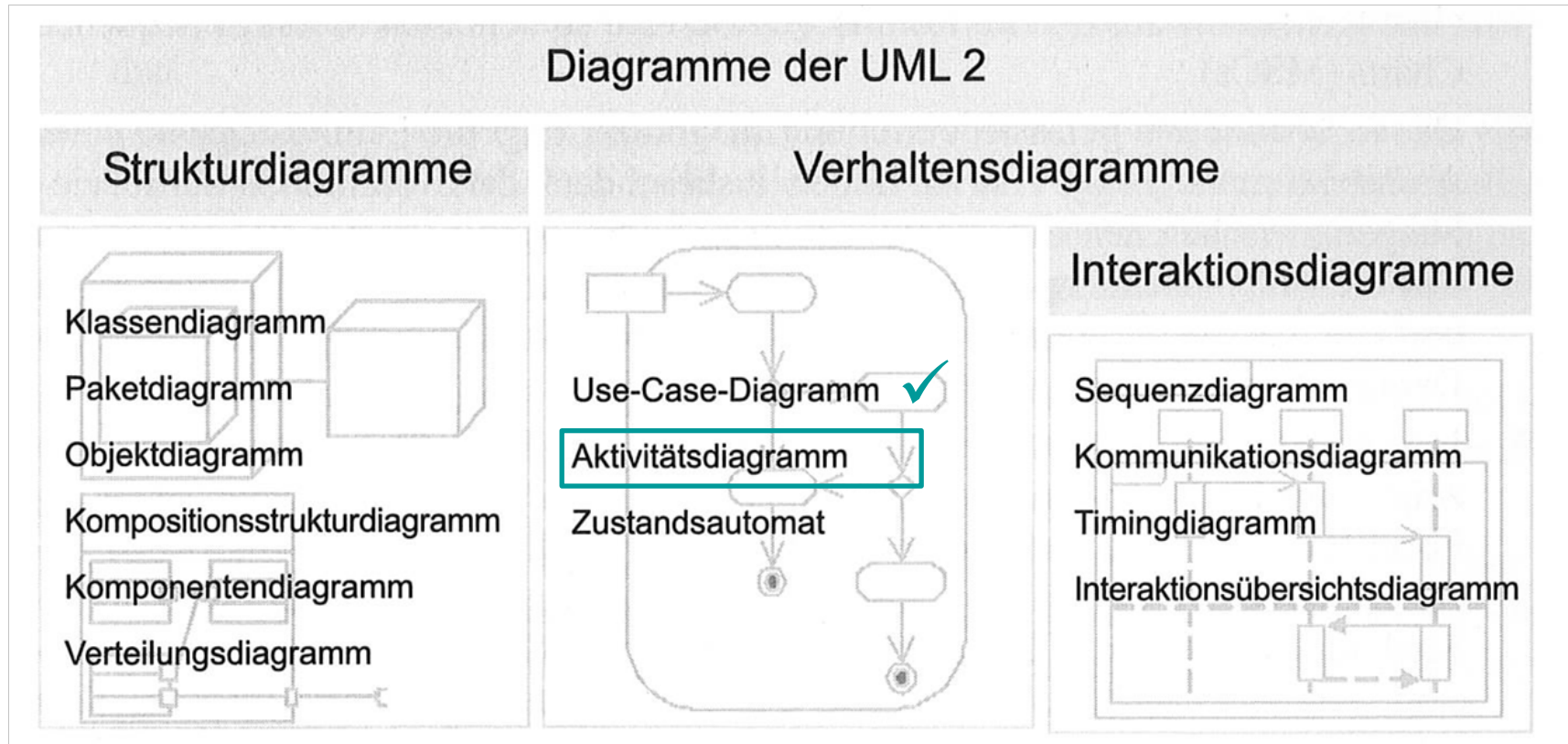


# Softwaretechnik

## Modellierung mit Aktivitäts- Diagrammen

Prof. Dr. Bodo Kraft

# Übersicht UML-Diagramme



Quelle: UML 2 glasklar, Chris Rupp

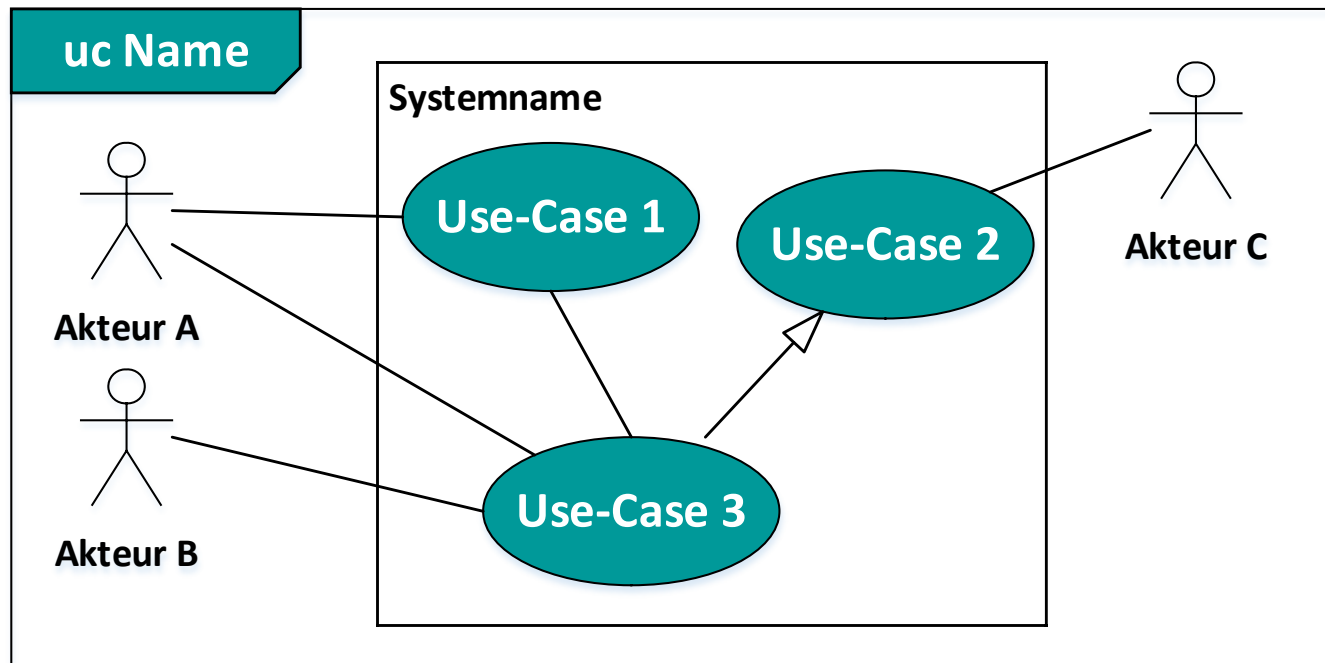
# Motivation

## Aktivitätsdiagramme

### Use-Case Diagramme

- Liefern eine Antwort auf die zentrale Frage:

Was soll mein System für seine Umwelt leisten?



# Motivation

## Aktivitätsdiagramme

### Aktivitätsdiagramme (AD)

Ein AD stellt konkrete Abläufe dar von:

- einem Anwendungsfall/Use-Case
- einer Operation
- oder einem Geschäftsvorfall

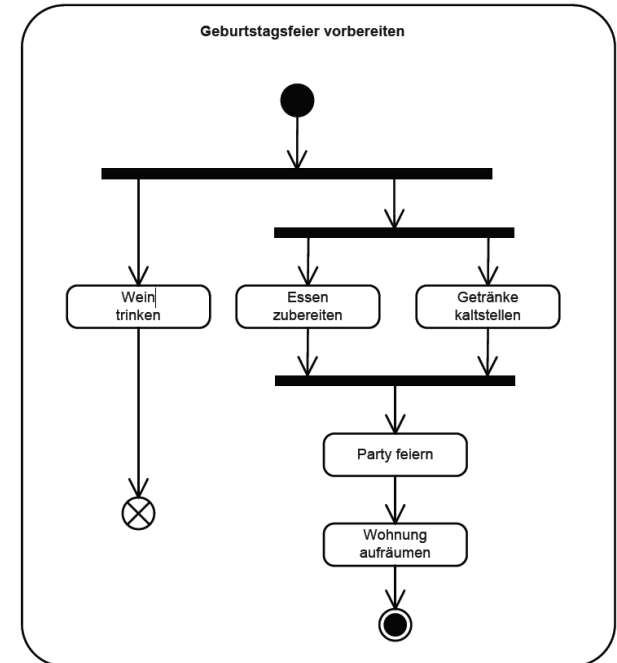
AD zeigen komplexe Verläufe u.a. mit:

- Nebenläufigkeiten
- Alternative Entscheidungswegen
- Zerlegung von Aufgaben in Einzelschritten

AD modellieren die Regeln für mögliche Abläufe.

AD Liefern Antwort auf die Frage:

*„Wie realisiert mein System ein bestimmtes Verhalten?“*



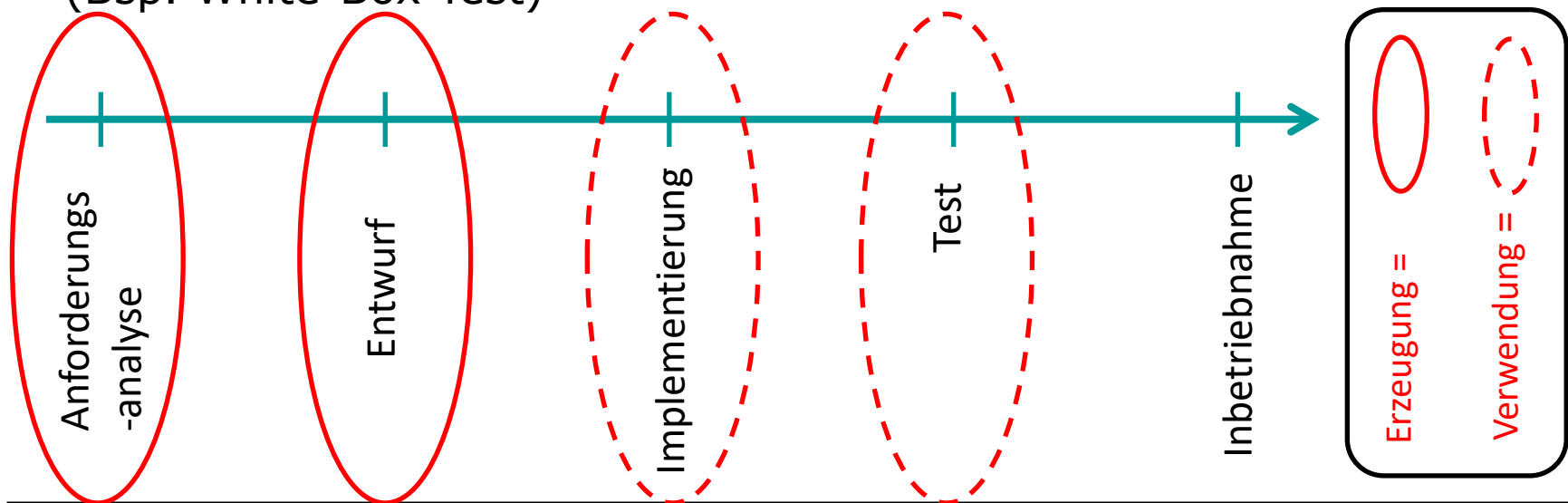
[nach Rupp, UML2 glasklar]

# Zeitliche Einordnung in SW-Lifecycle

## Aktivitätsdiagramme

### Bei welchen Schritten des Software-Lifecycle kann ich Aktivitätsdiagramme brauchen?

- Aktivitätsdiagramme werden hauptsächlich bei der Anforderungsanalyse und der Entwurfsphase verwendet.
- Die modellierten Abläufe werden in der Implementierungsphase realisiert.
- Die modellierten Abläufe können über Tests verifiziert werden (Bsp. White-Box-Test)

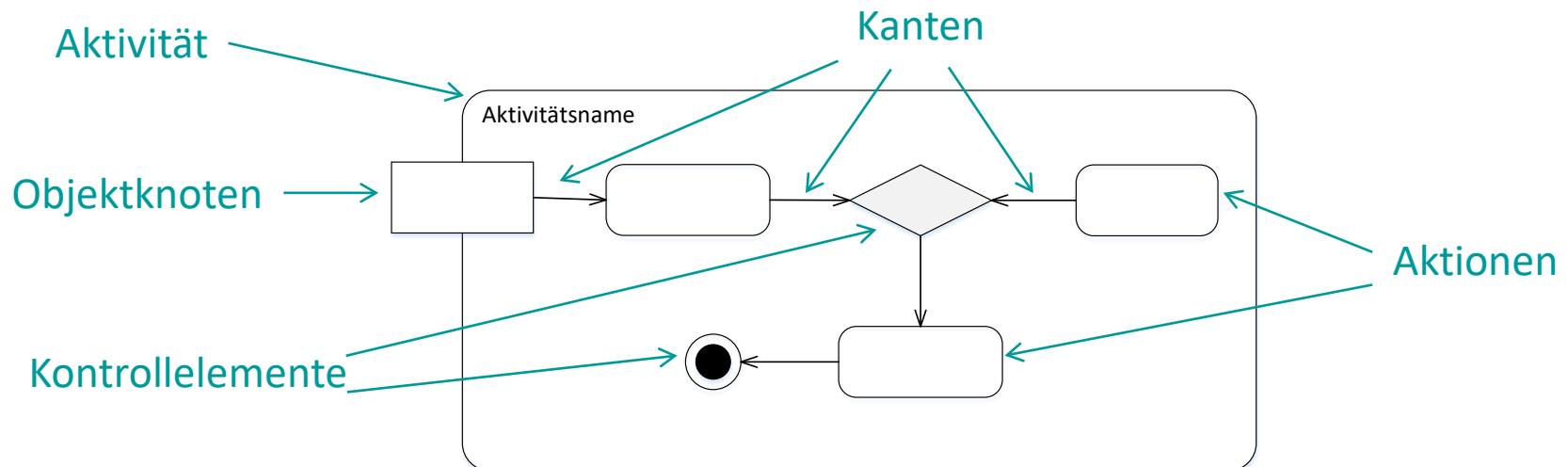


# Grundlagen (I)

## Übersicht Elemente von Aktivitätsdiagrammen

Wesentliche Elemente des Aktivitätsdiagramms sind:

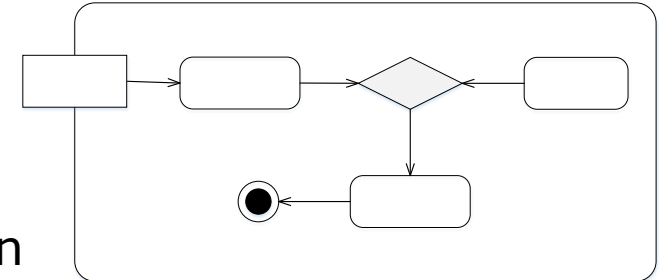
- 1) Aktivitäten
- 2) Aktionen
- 3) Objektknoten
- 4) Kontrollelemente zur Ablaufsteuerung
- 5) Verbindende Kanten



# Grundlagen (II)

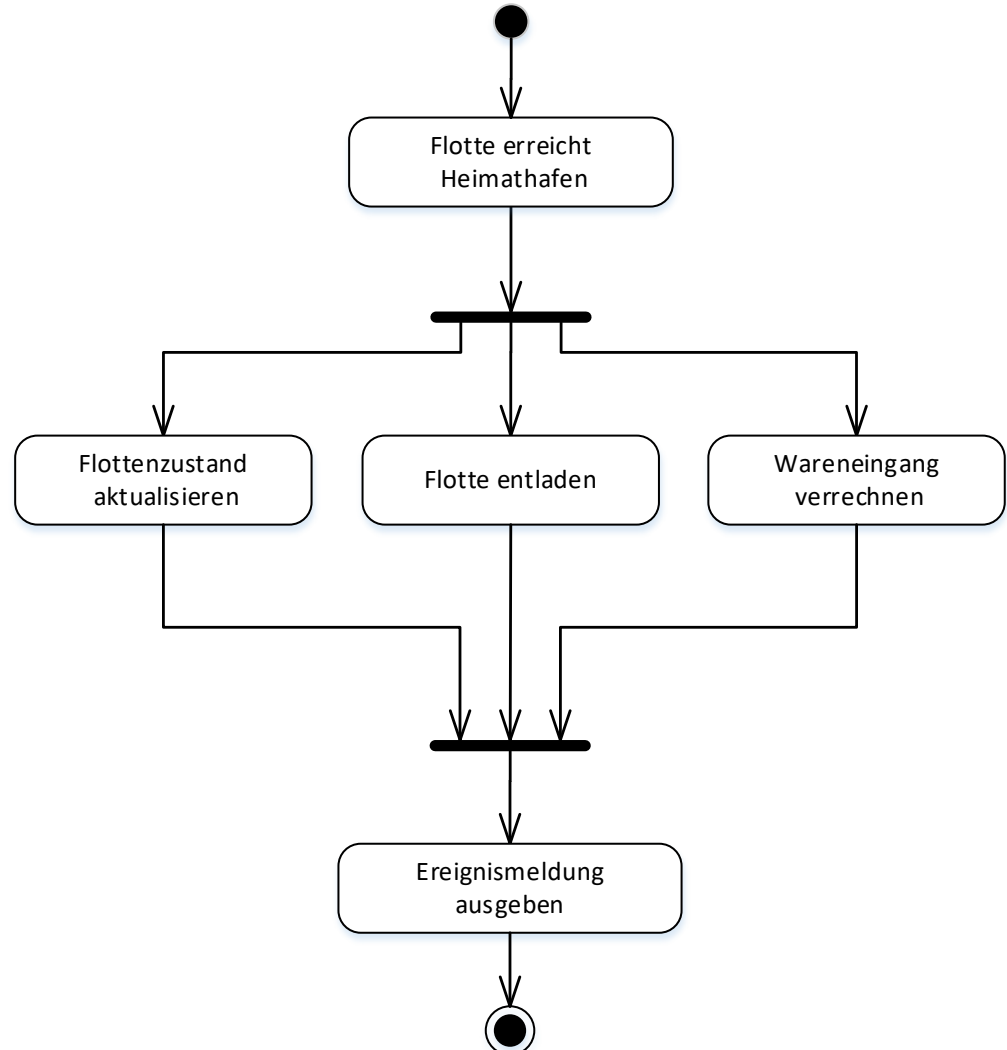
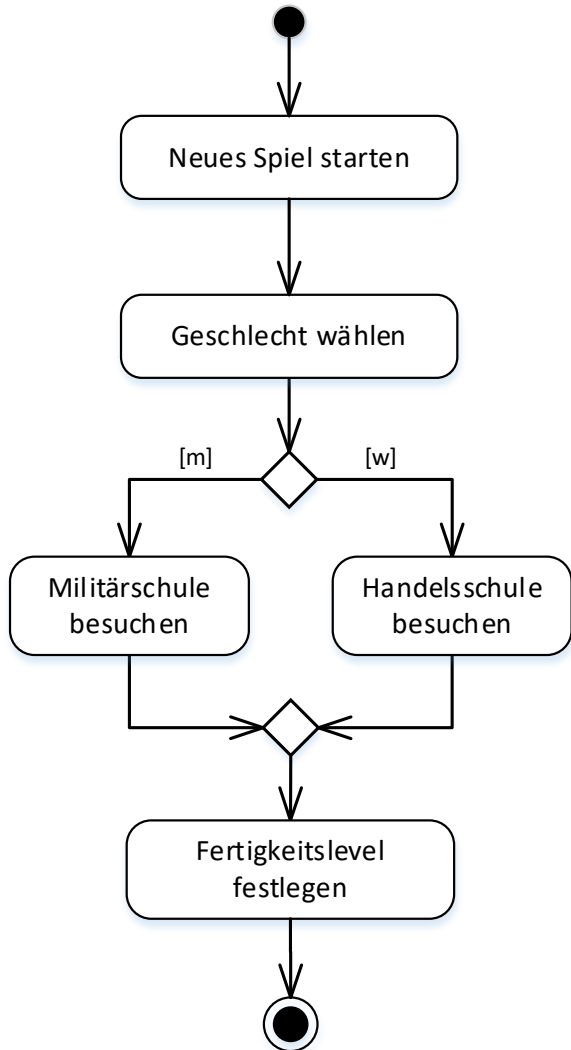
## Übersicht Elemente von Aktivitätsdiagrammen

- 1) Aktivitäten
  - Gesamtheit aller Abläufe
  - **Symbol:** Rahmen mit abgerundeten Ecken
- 2) Aktionen
  - Einzelschritt, den ein Ablauf unter Zeitaufwand durchschreitet + bei dem etwas „getan wird“
  - **Symbol:** Rechteck mit abgerundeten Ecken
- 3) Objektknoten
  - Beteiligte Daten/Schnittstellen einer Aktion
  - **Symbol:** Rechtecke
- 4) Kontrollelemente zur Ablaufsteuerung
  - Entscheidungsregeln + Bedingungen für Ablauf
  - **Symbol:** verschieden
- 5) Verbindende Kanten
  - Verlaufsweg mögl. Programmflüsse
  - **Symbol:** Pfeile zwischen den anderen Elementen



# Beispielabläufe

## Aktivitätsdiagramme





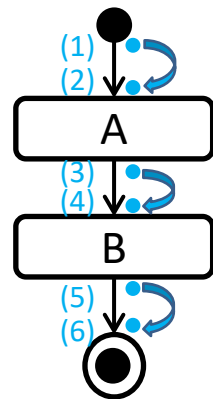
# Programmfluss und Tokensemantik (I)

## Konzeptmodell Aktivitätsdiagramm

- Abläufe in Aktivitätsdiagrammen werden als Kontrollfluss bzw. Datenfluss modelliert.
- Bildliche Vorstellung mithilfe von **Token-Semantik**.

- Ein Token
  - als Marke vorstellbar
  - wird an verschiedenen Punkten im Ablauf durchgereicht
  - Unterscheidung zwischen **Kontrolltoken(1)** und **Datentoken(2)**

- Beispiel dazu:



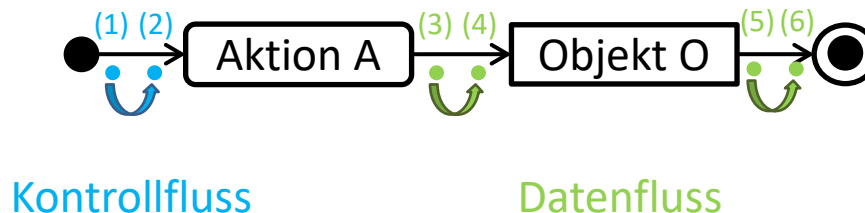
- (1) Startknoten erzeugt (Kontroll-)Token
- (2) Sobald Token an Eingangskante von A anliegt, wird Aktion A ausgelöst
- (2.5) Aktion A konsumiert Token, arbeitet etwas ab
- (3) Aktion A erzeugt Token bei Abschluss
- (4), (5) analog
- (6) Endknoten für Aktivitäten konsumiert Token

# Programmfluss und Tokensemantik (II)

## Konzeptmodell Aktivitätsdiagramm

### Datentoken

- Dient als Transportmittel für Daten oder Werte
- Datenfluss über Objektknoten-Kanten
- Eingehende Token repräsentieren Daten/Werte, die im Objektknoten gesetzt bzw. gesammelt werden
- Ausgehende Token repräsentieren das Objekt selbst  
(oder Eingangsdaten in die Folgeaktion)



# Kontrollelemente zur Ablaufsteuerung (I)

## Syntax Aktivitätsdiagramme

### Kontrollelemente im Detail:

- **Startknoten**

- Markiert den Startpunkt eines Ablaufs bei Aktivieren einer Aktivität
- Eine Aktivität kann beliebig viele Startknoten haben (Nebenläufige Ausführung an allen Startknoten)
- Eine Aktivität muss keinen Startknoten besitzen (Eingangsparameter dienen als Startpunkt)



Startknoten

- **Endknoten für Aktivitäten**

- Beendet sofort die gesamte Aktivität
- Parallel ausgeführte Aktionen werden ebenfalls beendet.



Endknoten  
für Aktivitäten

- **Endknoten für Kontrollflüsse**

- Beendet nur einen einzelnen Ablauf
- Nebenläufig ausgeführte Aktionen werden nicht beendet.
- Beendigung der gesamten Aktivität daher nur bei nicht parallelisierten Abläufen.



Endknoten  
für Kontrollflüsse

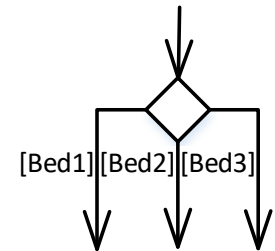
# Kontrollelemente zur Ablaufsteuerung (II)

## Syntax Aktivitätsdiagramme

### Kontrollelemente im Detail:

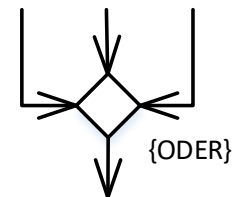
- **Verzweigungsknoten**

- Spaltet eine Kante in mehrere Alternativen auf
- Programmfluss wird von Bedingungen abhängig gemacht
- Bei mehreren ausgehenden Kanten müssen alle Bedingungen disjunkt sein + Programmfluss eindeutig entscheidbar sein



- **Verbindungsknoten**

- Führt mehrere Kanten zusammen, in einen gemeinsamen Programmfluss über
- „Logisches ODER“



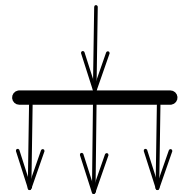
# Kontrollelemente zur Ablaufsteuerung (III)

## Syntax Aktivitätsdiagramme

### Kontrollelemente im Detail:

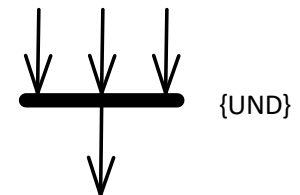
- **Parallelisierungsknoten**

- Teilt den Ablauf einer eingehenden Kante in mehrere parallele Abläufe (ausgehende Kanten)
- Ermöglicht Nebenläufigkeit von Aktionen
- Kontroll- bzw. Objektfluss müssen am Ende wieder zusammengeführt oder getrennt beendet werden.



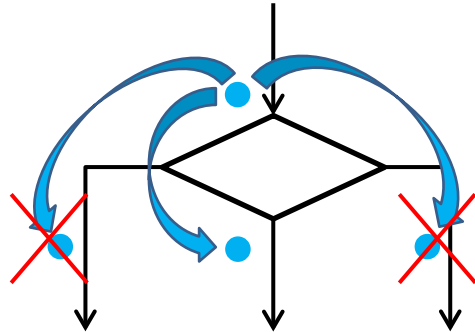
- **Synchronisationsknoten**

- Vereint parallele Abläufe
- Ablauf wird erst fortgesetzt, wenn alle vorherigen Aktionen(eingehende Kanten) durchgeführt wurden
- Entspricht logischer „UND“-Verknüpfung

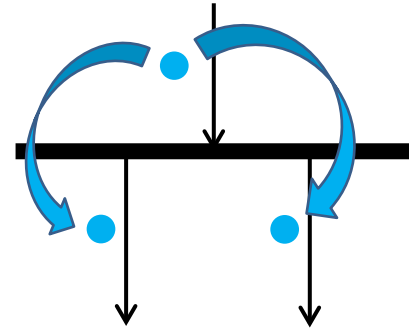


# Kontrollelemente zur Ablaufsteuerung (III)

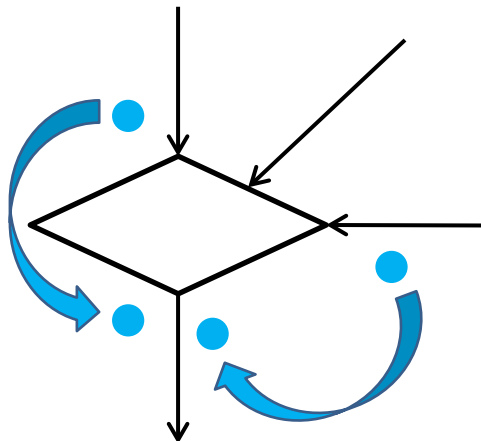
## Tokensemantik über Kontrollflussknoten



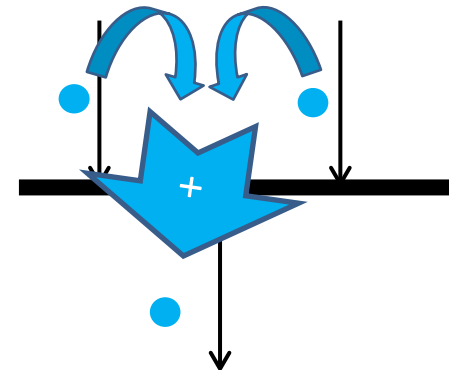
Verzweigungsknoten



Parallelisierungsknoten



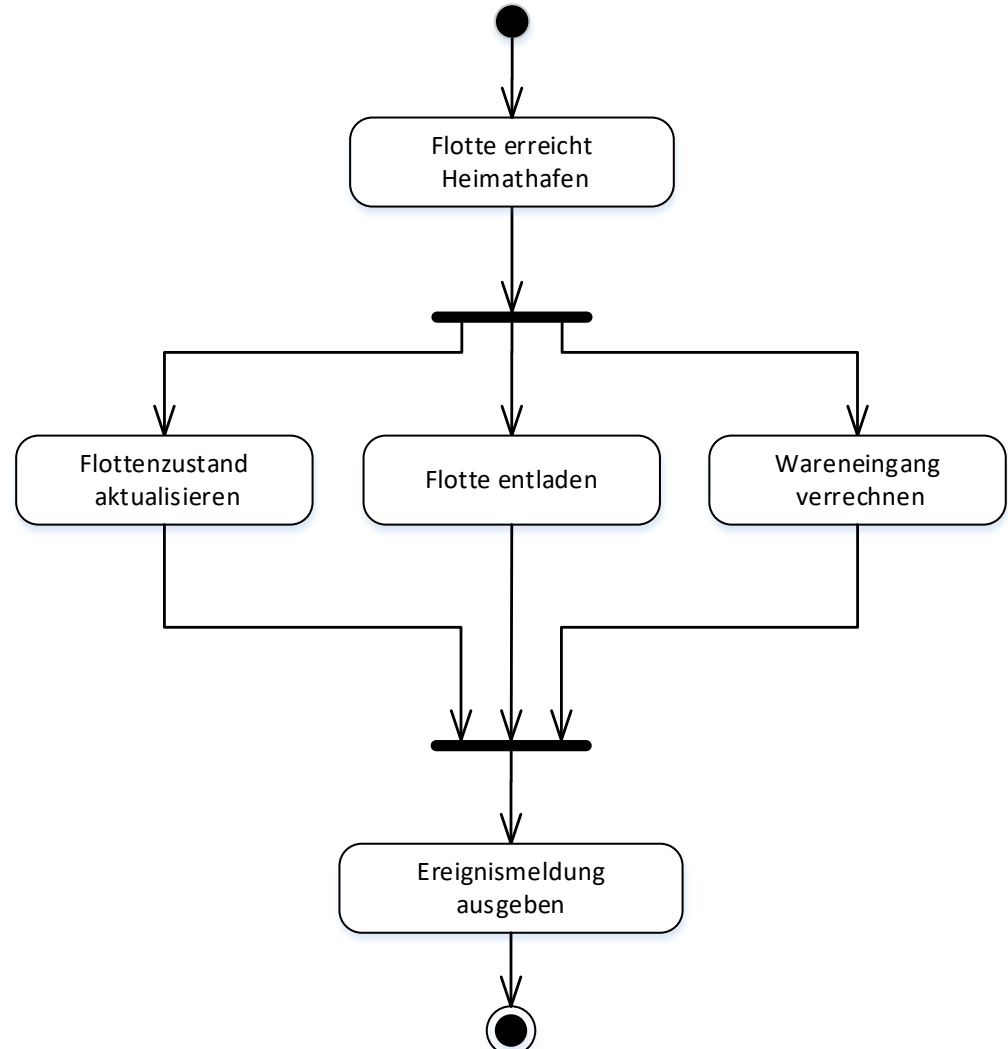
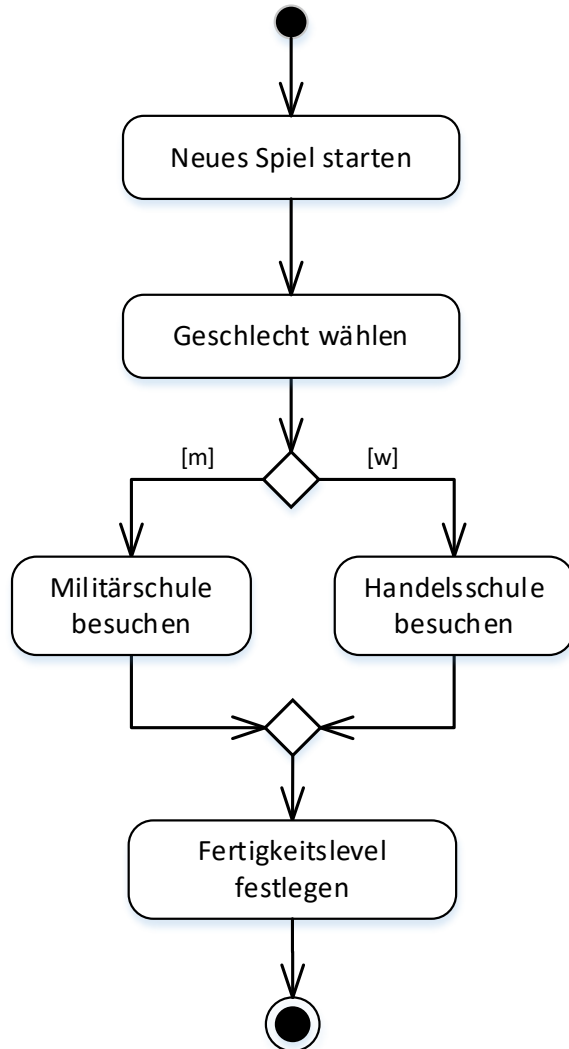
Verbindungsknoten



Synchronisationsknoten

# Beispielabläufe: Knoten & Verzweigung


## Syntax Aktivitätsdiagramme

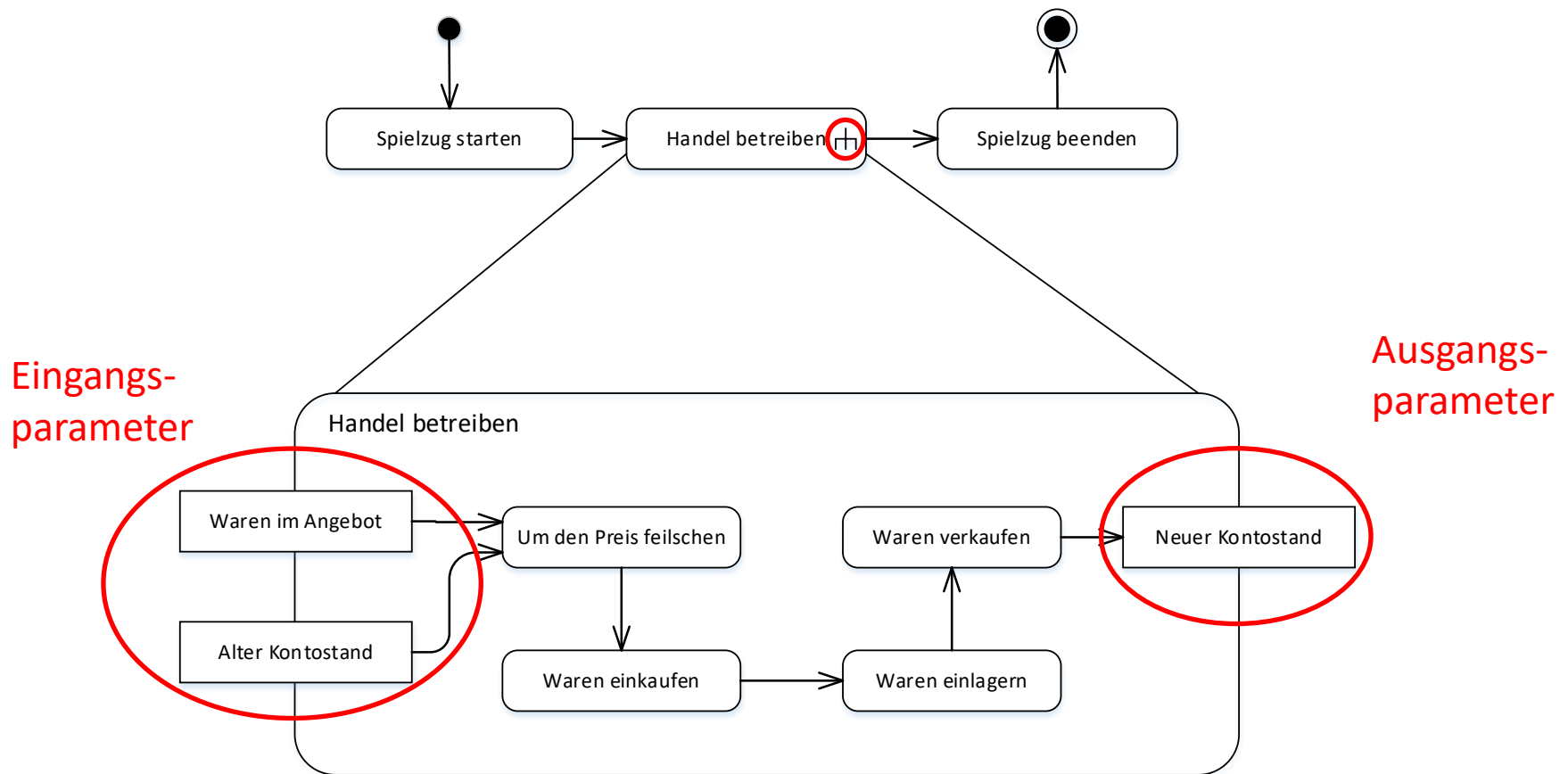


# Strukturierung durch Aufrufhierarchie

## Syntax Aktivitätsdiagramm

Aktionen im Aktivitätsdiagramm können selbst wiederum als eigene Aktivitäten beschrieben werden.

- **Symbol:**  „stilisierte Harke“

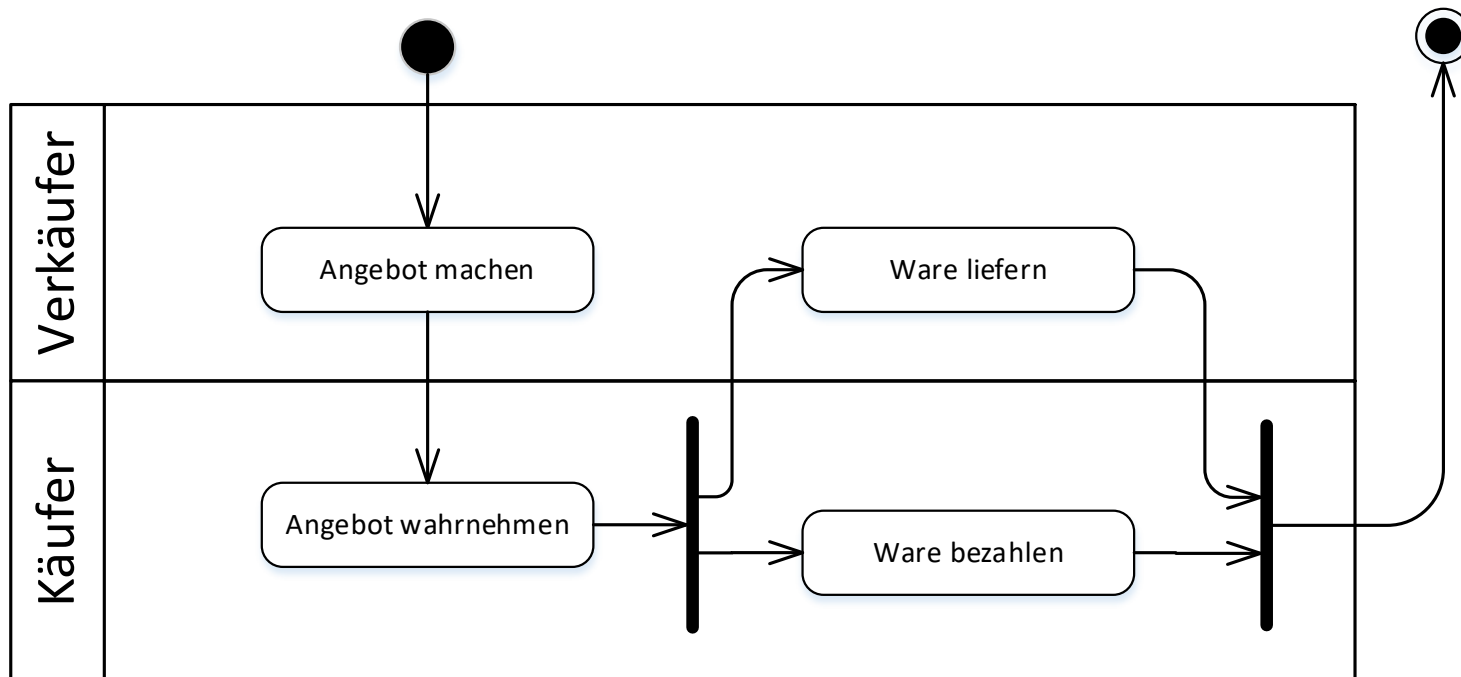




# Strukturierung durch Aktivitätsbereiche

## Syntax Aktivitätsdiagramm

- Aktivitäten lassen sich mithilfe von Aktivitätsbereichen aufteilen (**activity partitions**)
- Ein Aktivitätsbereich umfasst dabei mehrere Aktionen mit **gemeinsamen Eigenschaften** (oder Rollen)
- Ziel ist die schnell erkennbare **Aufteilung in Verantwortungsbereiche**



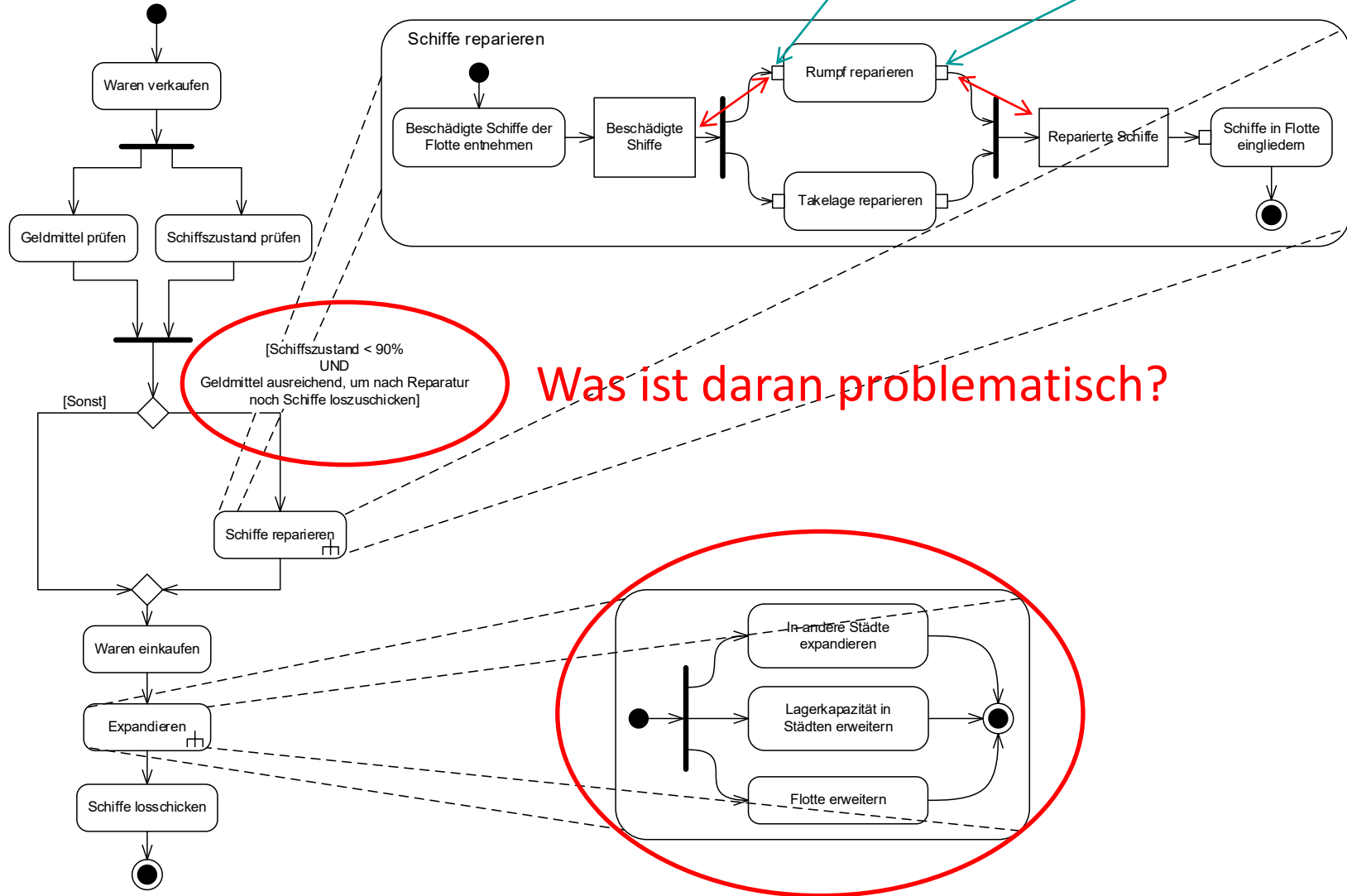
# Beispiel Spielzug

## Syntax Aktivitätsdiagramme

Pin-Notation für Objektknoten

Eingabe-Pin

Ausgabe-Pin



Was ist daran problematisch?

# Signale und Ereignisse (I)

## Sonderformen von Aktionen

Neben den klassischen (bisher betrachteten) Abläufen gibt es weitere Elemente **zur asynchronen Flusssteuerung**.  
Übergänge werden modelliert mithilfe von **Signalen & Ereignissen**

Aktionen treten dabei in zwei Rollen auf:

- Als Signalsender:
  - löst Sonderaktion (*SendSignalAction*) aus
- Als Ereignisempfänger:
  - löst Sonderaktion (*AcceptEventAction*) aus:



# Signale und Ereignisse (II)

## Sonderformen von Aktionen

---

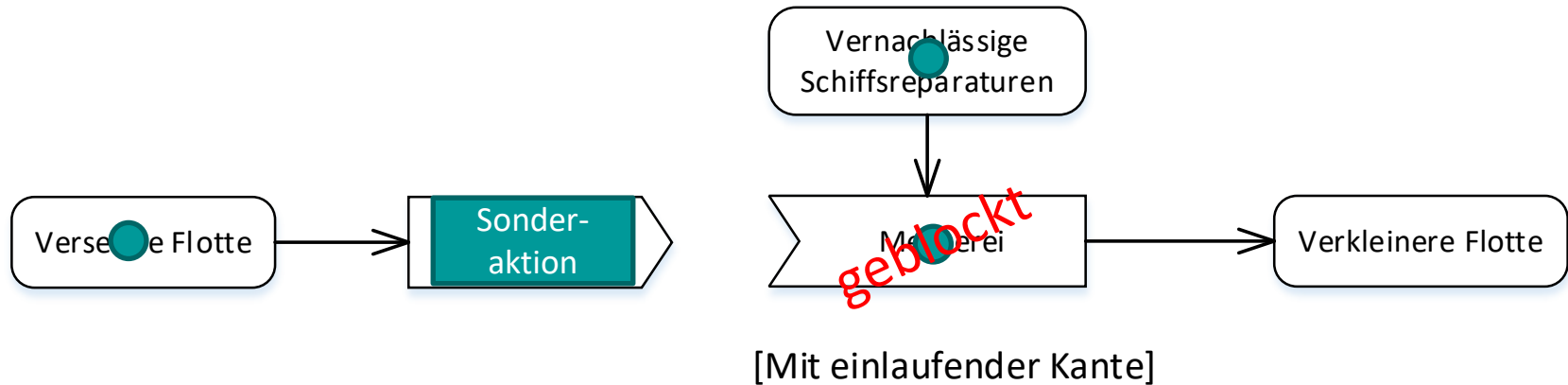
Empfangsereignisse treten in drei Konstellationen auf.

1. Ereignisempfänger **mit einlaufender Kante**
  - Ereignisempfänger reagiert erst, wenn sowohl Empfänger aktiv (also den Fokus hat) als auch Sendesignal empfangen wurde.
2. Ereignisempfänger **ohne einlaufende Kante**
  - Ereignisempfänger reagiert unmittelbar nach Erhalt des Sendesignals wie eine normale Aktion
3. Ereignisempfänger **(ohne einlaufende Kante, aber) mit Unterbrechungskante**
  - Ereignisempfänger reagiert unmittelbar nach Erhalt des Sendesignals
  - Zusätzlich wird Gruppierungsbereich unterbrochen

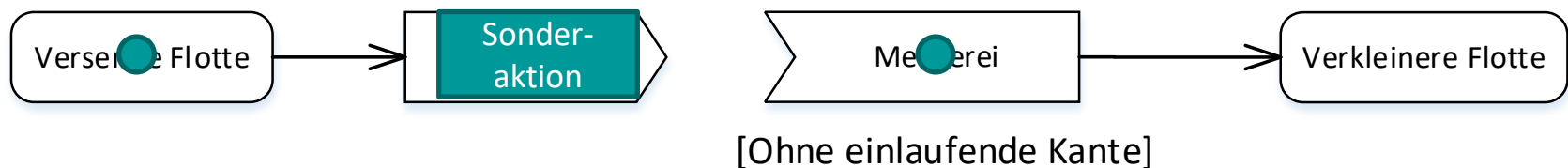
# Signale und Ereignisse (III)

## Beispiele für Sonderformen von Aktionen

### Fall1: Ereignisempfänger mit einlaufender Kante



### Fall2: Ereignisempfänger ohne einlaufende Kante

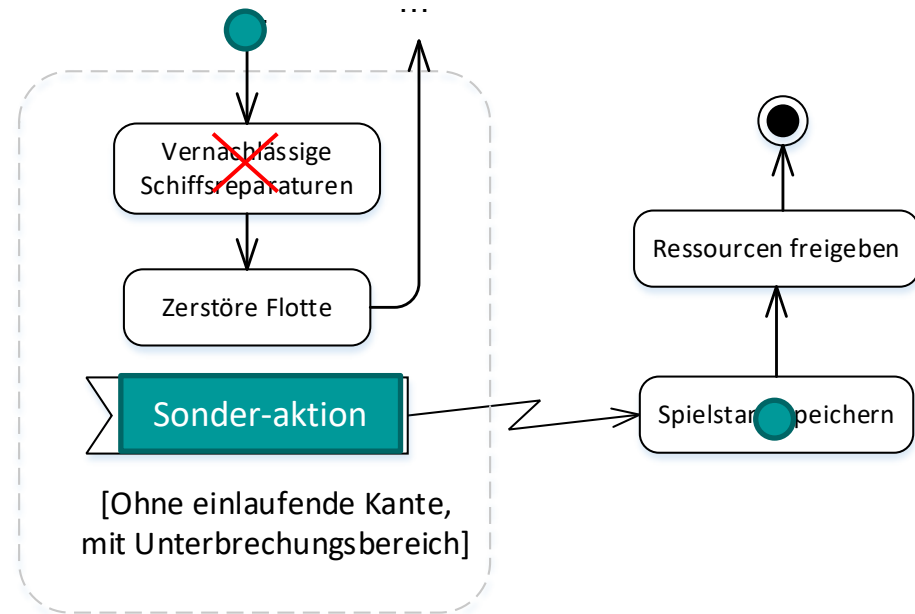


# Signale und Ereignisse (IV)

## Beispiele für Sonderformen von Aktionen

### Fall3: Ereignisempfänger mit Unterbrechungskante

- Aktionen und Abläufe können gruppiert werden
- Gruppierungen heißen **Activity Regions**
  - Symbol: gestrichelter Rahmen



- **Unterbrechungskanten** sorgen für das sofortige Verlassen des gesamten aktiven Bereichs bzw. einer Activity Region.
  - Symbol: **gezackter Pfeil**
- Hier:  
Empfangsereignis beendet den aktiven Bereich und läuft bei Zielaktion → **Spielstand speichern** weiter.
- Die Flotte wird ggfs. nicht mehr zerstört!

- [RS] C. Rupp, SOPHIST GROUP, Requirements- Engineering und – Management, Hanser Fachbuchverlag, 2004
- [OW] B. Oestereich, C. Weiss, C. Schröder, T. Weilkiens, A. Lenhard, Objektorientierte Geschäftsprozessmodellierung mit der UML, dpunkt.Verlag, 2003

---

# Vielen Dank!