

# Anforderungsanalyse

- Motivation
- Begriffe, Überblick
- Dokumentieren von Anforderungen
  - Use Case-Diagramme
  - Aktivitätsdiagramme

# Warum Anforderungsanalyse? (1)

- 50 % der *im Quellcode gefundenen* Fehler sind auf *mangelhafte Anforderungen* zurückzuführen
  - D.h. es wurde etwas realisiert, was der Kunde (so) nie wollte
  - Kunden sind nicht geübt darin, sich gegenüber einem (fachfremden) Informatiker klar auszudrücken

# Mögliche Probleme aufgrund der Anforderungsanalyse



How the customer explained it



How the project leader understood it



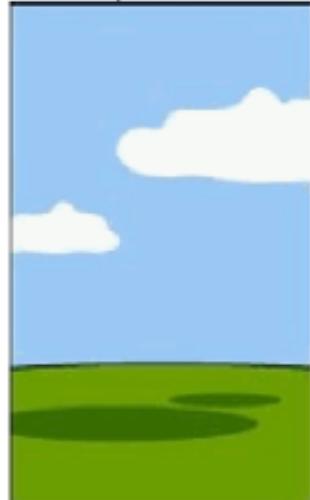
How the engineer designed it



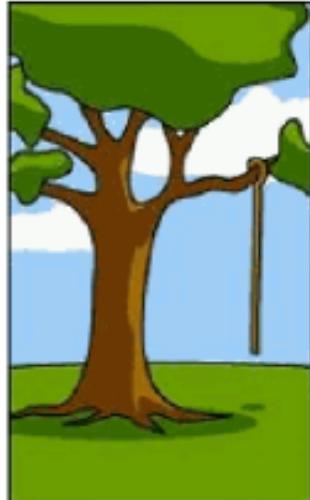
How the programmer wrote it



How the sales executive described it



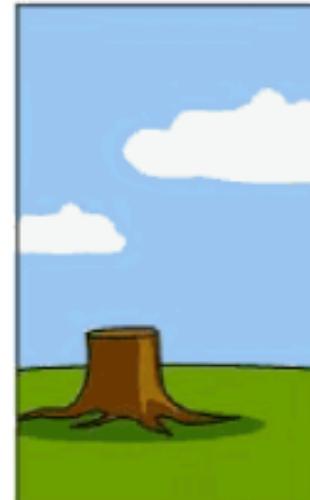
How the project was documented



What operations installed



How the customer was billed



How the helpdesk supported it



What the customer really needed

Alan Chapman

## Warum Anforderungsanalyse? (2)

- Die Beseitigung eines Fehlers, der in der Programmierphase gefunden wird, ist um den *Faktor 20* teurer als die Beseitigung während der Anforderungsphase.

Im Abnahmetest: *Faktor 100*

- D.h., es lohnt, bei den Anforderungen lieber einmal mehr „nachzufragen“

## Warum Anforderungsanalyse? (3)

- Gutes Requirements Engineering bewirkt...
  - ... niedrigere Kosten
    - ▶ Geringere Herstellungskosten  
(Senken der Fehlerkosten!)
    - ▶ Weniger Reklamationen und Nachbesserungen
    - ▶ Geringere Pflegekosten
  - ... kleinere / weniger Risiken
    - ▶ Kundenerwartungen besser erfüllen
    - ▶ Zuverlässigere Prognosen für Termine und Kosten

Aber: Die wirtschaftliche *Wirkung* von Requirements Engineering ist immer *indirekt*; das RE selbst kostet nur!

# Arten von Anforderungen

- **Funktionale Anforderung**
  - definiert eine vom System oder einer Systemkomponente bereitzustellende Funktion bzw. Service
  - engl.: functional requirements, FR
- **Qualitätsanforderung**
  - definiert eine qualitative Eigenschaft des Systems, einer Systemkomponente oder einer Funktion
  - Synonym: nichtfunktionale Anforderung
  - engl.: non-functional requirements, NFR
- **Randbedingung**
  - ist eine organisatorische oder technologische Anforderung, die die Art und Weise einschränkt, wie ein Produkt entwickelt wird
  - engl.: constraint

Pohl, Rupp: Basiswissen Requirements-Engineering, 2009

# Definition

Als **Anforderungsspezifikation** bezeichnet man die Zusammenstellung aller Anforderungen an ein System.

Synonym:

- Anforderungsdokument
- Requirements Specification*
- Functional Requirements Specification*

Anmerkung:

1. Der Begriff „Spezifikation“ ist im Alltag nicht immer eindeutig:  
Prozess oder Dokument (= Ergebnis des Prozesses)?
  - ▶ **andere Beispiele: Design, Entwurf**
2. Inhalt der Anforderungsspezifikation: z.B. IEEE 830-1998

# Definition und Beispiele

Ein Stakeholder ist ...

- eine Person oder Organisation, welche (direkt oder indirekt) Einfluss auf die Anforderungen des betrachteten Systems hat.

Pohl, Rupp: Basiswissen Requirements-Engineering, 2009

Stakeholder sind alle, die ein Interesse an dem Produkt haben:

- Sie bezahlen es
- Sie nutzen es (heute, morgen)
- Sie verwalten es
- Sie warten/betreuen es
- Sie bauen es
- Sie kennen die Domänen, für die das Produkt entwickelt wird
- Sie kennen den Markt, in dem das Produkt angeboten wird
- Sie sind durch das Produkt in irgendeiner Weise betroffen



# Aufgaben der Anforderungsanalyse

## Anforderungen...

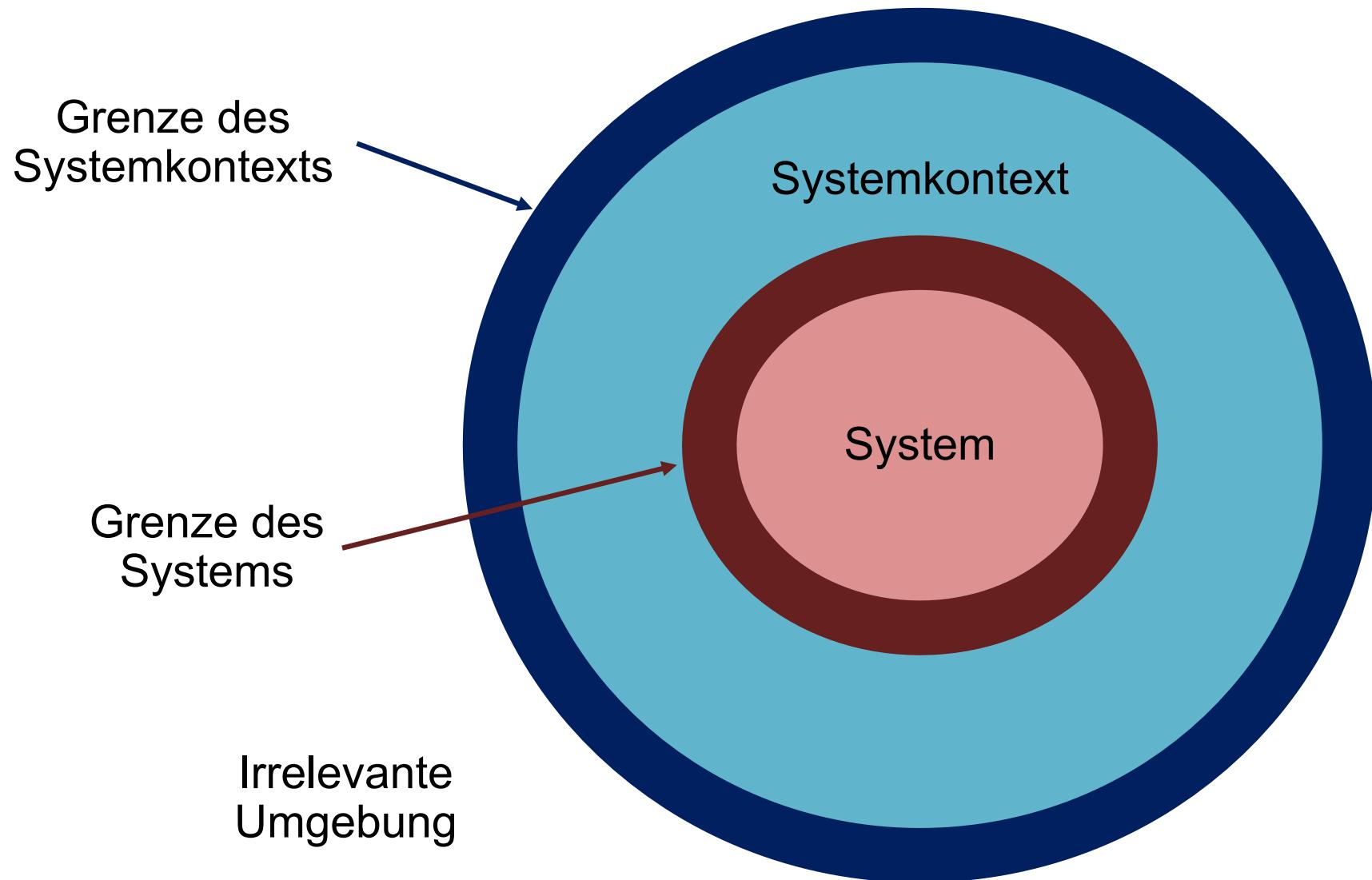


- ... erheben *Elicitation*
- ... konsistent zusammenstellen, priorisieren *Analysis*
- ... dokumentieren *Specification*
- ... überprüfen *Validation*

# *Erheben* von Anforderungen

- Wissen sammeln über
  - Beteiligte Stakeholder und ihre Ziele
  - Nutzeraufgaben
  - Gegenwärtige Situation (Ist-Situation)
  - Erwartungen
  - Wissen über die Domäne
- Der schwierigste Schritt der Anforderungsphase
  - Geht über das Abfragen von Wissen hinaus.
  - Die Quellen sind zu Beginn der Anforderungsphase oft nicht bekannt.
  - Beinhaltet häufig auch die Generierung einer neuen Idee (insbesondere bei innovativen/neuen Produkten).

## Erster Schritt: Systemkontext festlegen



# Bedeutung des Systemkontexts

- Anforderungen werden für einen bestimmten Kontext definiert
- Erst die Kenntnis des Kontexts macht die Anforderungen interpretierbar und bestimmt deren Umfang
- Die Anforderungen müssen aus dem Systemkontext erhoben werden
  - Leute / Stakeholder / Nachbarsysteme haben bestimmte Anforderungen / Erwartungen
  - Geschäftsprozesse implizieren gewisse Anforderungen
  - Richtlinien der Organisation werden in Anforderungen übersetzt
  - etc.
- Je besser das Systemkontext verstanden wird, desto geringer ist das Risiko falscher oder unvollständiger Anforderungen

Zur Darstellung benutzen wir UML-Diagramme:

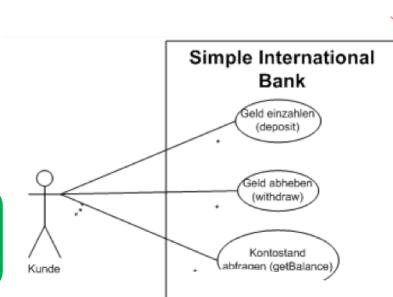
Man erhofft sich davon,

- dass der Übergang von der Analyse (was soll das System tun?) zum Design (wie ist das System aufgebaut, wie arbeitet es?) leichter wird und
- dass weniger Abstimmungsfehler passieren

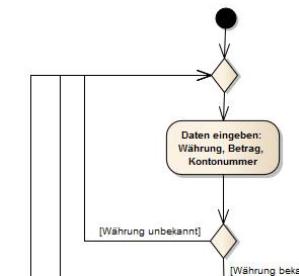
# Use Case

- *Deutsch: Anwendungsfall*
- Zur strukturierten Dokumentation der *Funktionalität* existierender oder geplanter Systeme durch *Interaktionsfolgen*

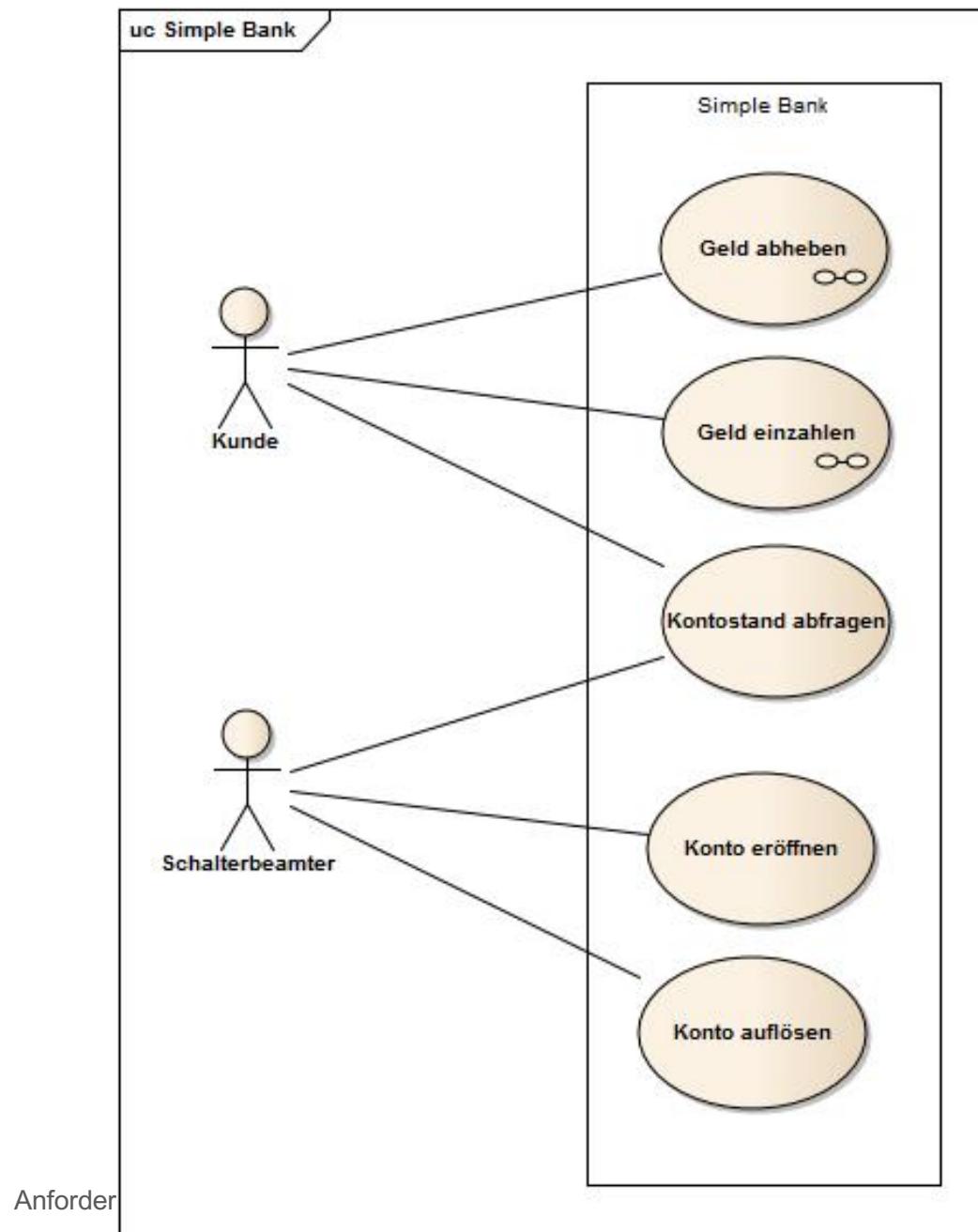
- Besteht aus
  - Use Case-Diagramm
  - und*
  - Use Case-Tabellen oder
  - Aktivitätsdiagrammen



Anwendungs-fallname	HebeGeldAb
Akteure	Kunde
Ziel	Geld abheben
Hauptablauf	1. Der Kunde meldet sich mit seiner Kontonummer/PIN am Geldautomaten an. 2. Das System überprüft die Identität. 3. Der Kunde gibt einen Betrag ein und wählt eine Währung aus. 4. Das System überprüft, ob die Währung vorrätig und der Betrag

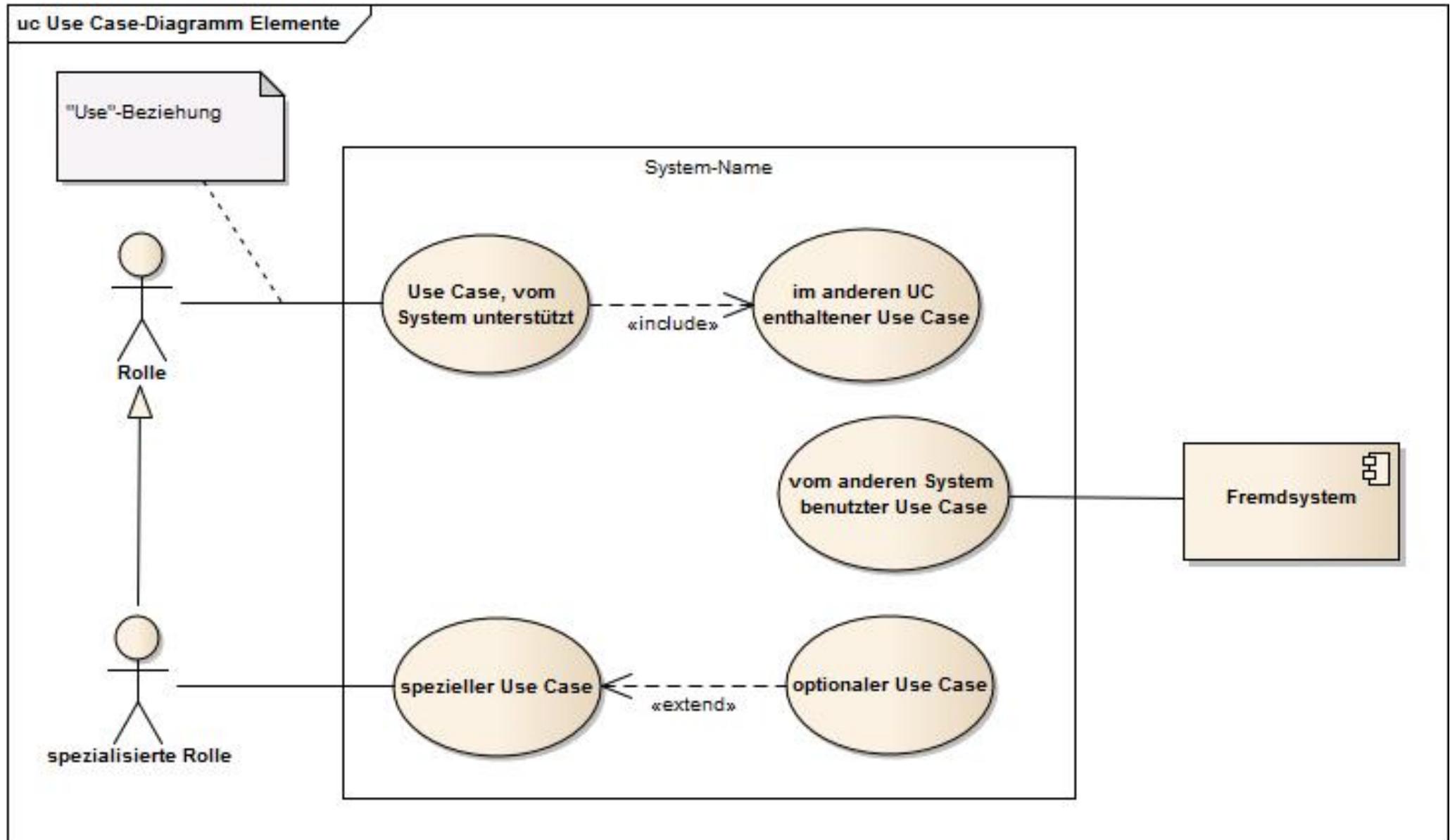


# Beispiel: Use Case „Hebe Geld Ab“



- UML Use Case-Diagramme zeigen,
  - *welche* Anwendungsfälle (= welche Funktionalität) ein System bietet (bzw. welche nicht) und
  - *wer* diese in Anspruch nehmen kann

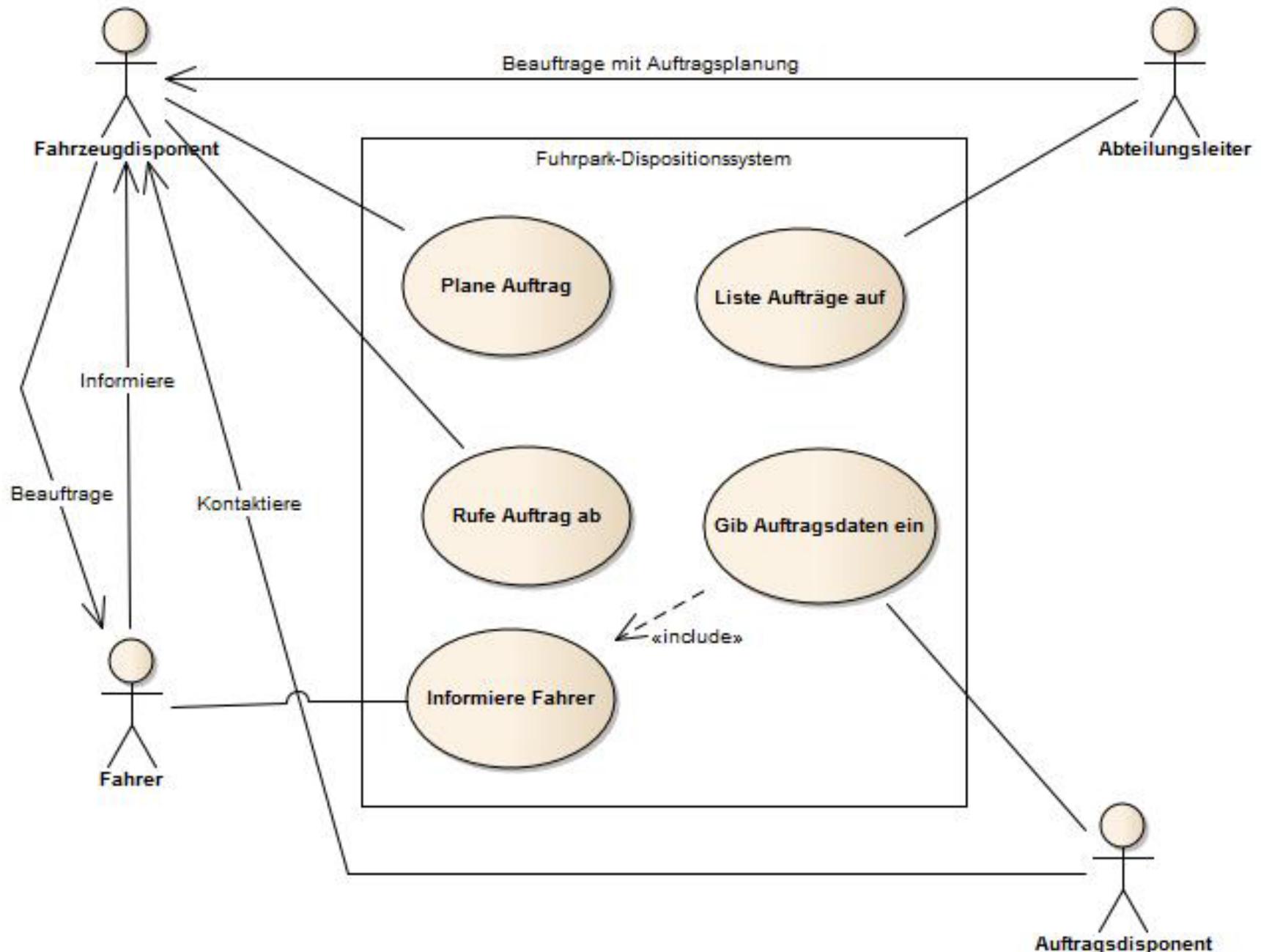
# Elemente von Use Case-Diagrammen



# Regeln für Use Cases

1. *Use Cases* sollen mit *Verbalphrasen* bezeichnet werden. Der Name sollte anzeigen, was der Anwender bewirken möchte.
  - z.B.      Melde Vorfall  
                Eröffne Vorfall
2. *Rollen* sollen mit *Nominalphrasen* benannt werden.
  - z.B.      Außenbeamter  
                Dienstleiter  
                Zentraler DB-Server
3. Ein Use Case soll eine *vollständige Transaktion* eines Anwenders schildern.
  - z.B.      „Login“ ist *kein Use Case*

# Der Systemkontext als Use Case-Diagramm



## Nächster Schritt

Für (fast) jeden Anwendungsfall aus dem *Use Case-Diagramm* beschreiben, wie er abläuft

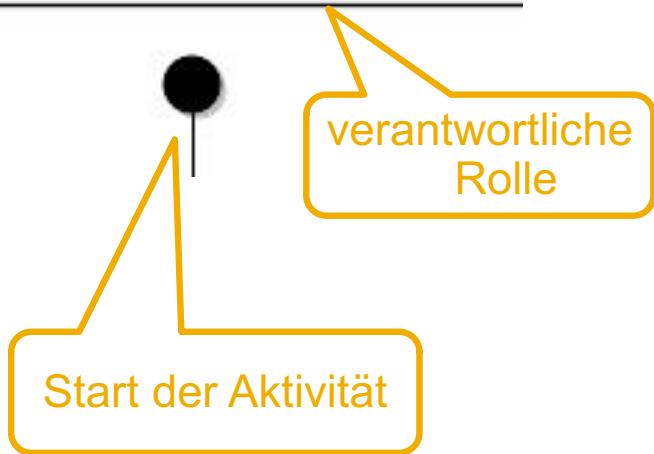
Es gibt zwei Alternativen:

- Beschreibung in Tabellenform
- Beschreibung als Aktivitätsdiagramm

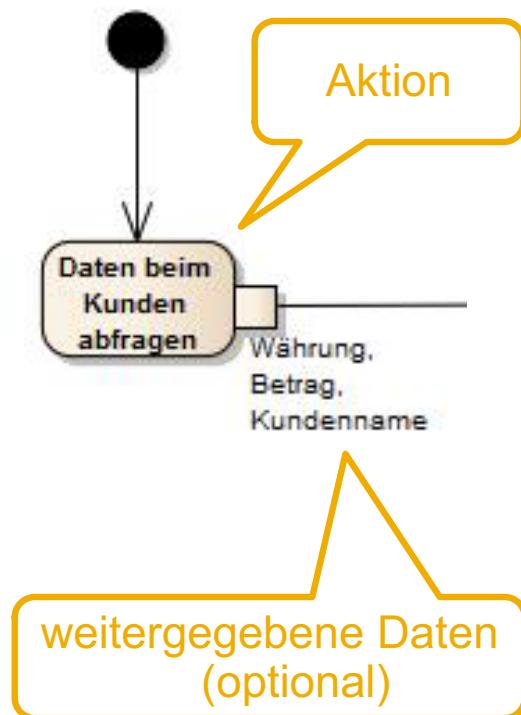
Anwendungsbeispiel

- Ein Kunde geht zur Bank und weist sich mit seinem Namen aus, um ein Konto zu eröffnen. Er bekommt dann eine (neue) Kontonummer zugewiesen. Für die Kontoeröffnung ist eine Mindesteinzahlung von € 50 verpflichtend. Die Einzahlung kann in allen bekannten Währungen erfolgen.

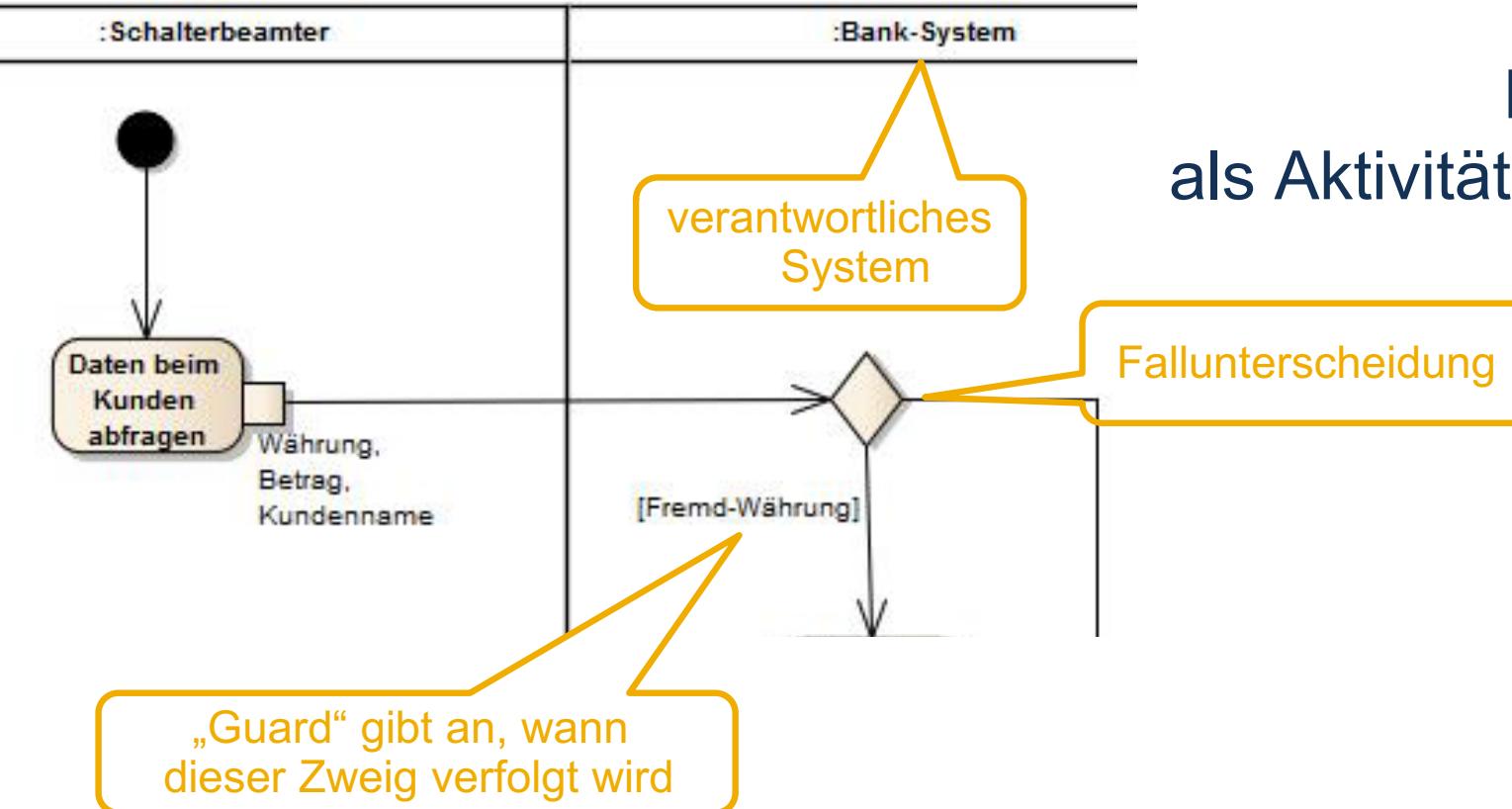
:Schalterbeamter



## Darstellung als Aktivitätsdiagramm



## Darstellung als Aktivitätsdiagramm

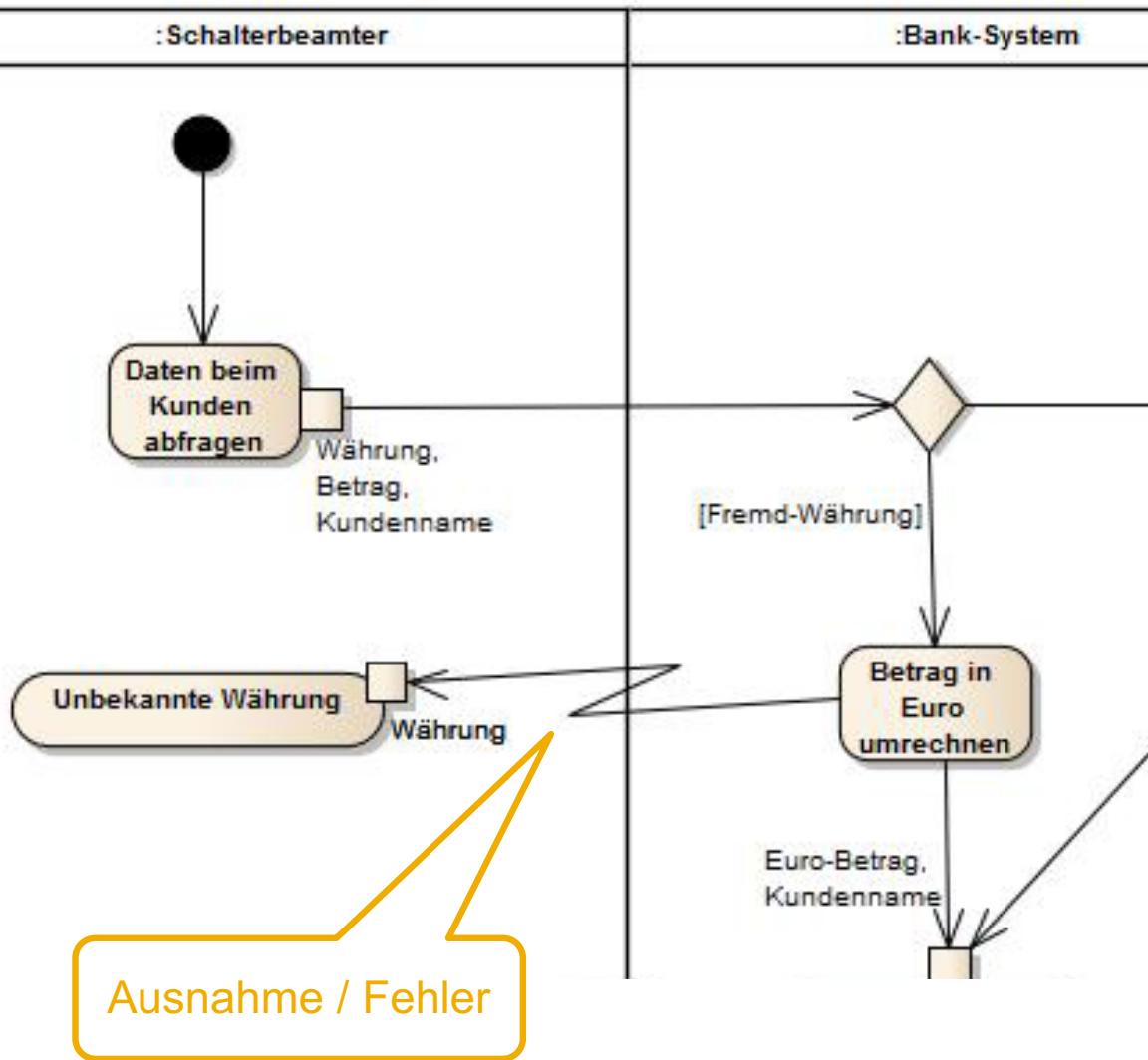


## Darstellung als Aktivitätsdiagramm

:Schalterbeamter

:Bank-System

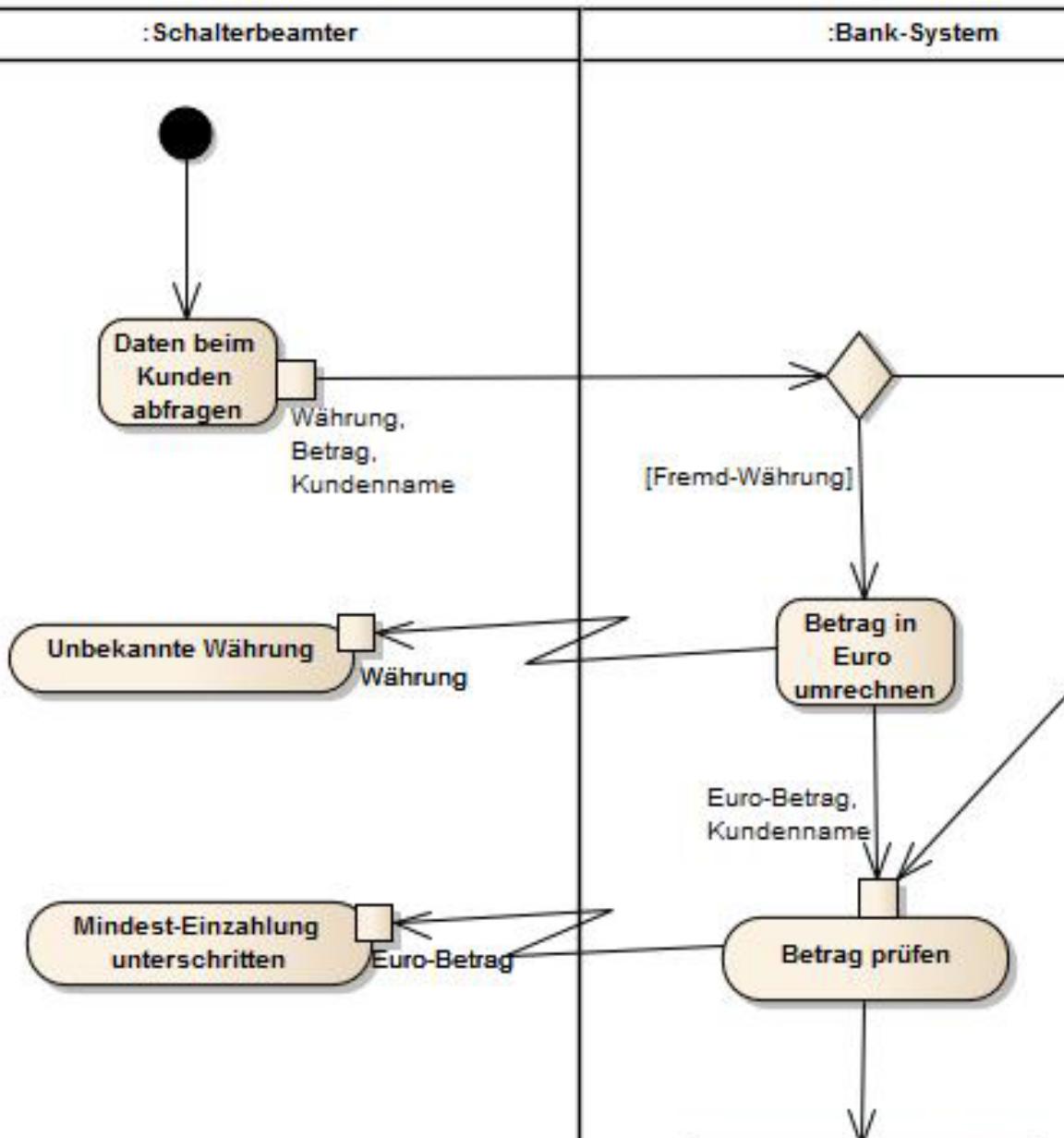
## Darstellung als Aktivitätsdiagramm



:Schalterbeamter

:Bank-System

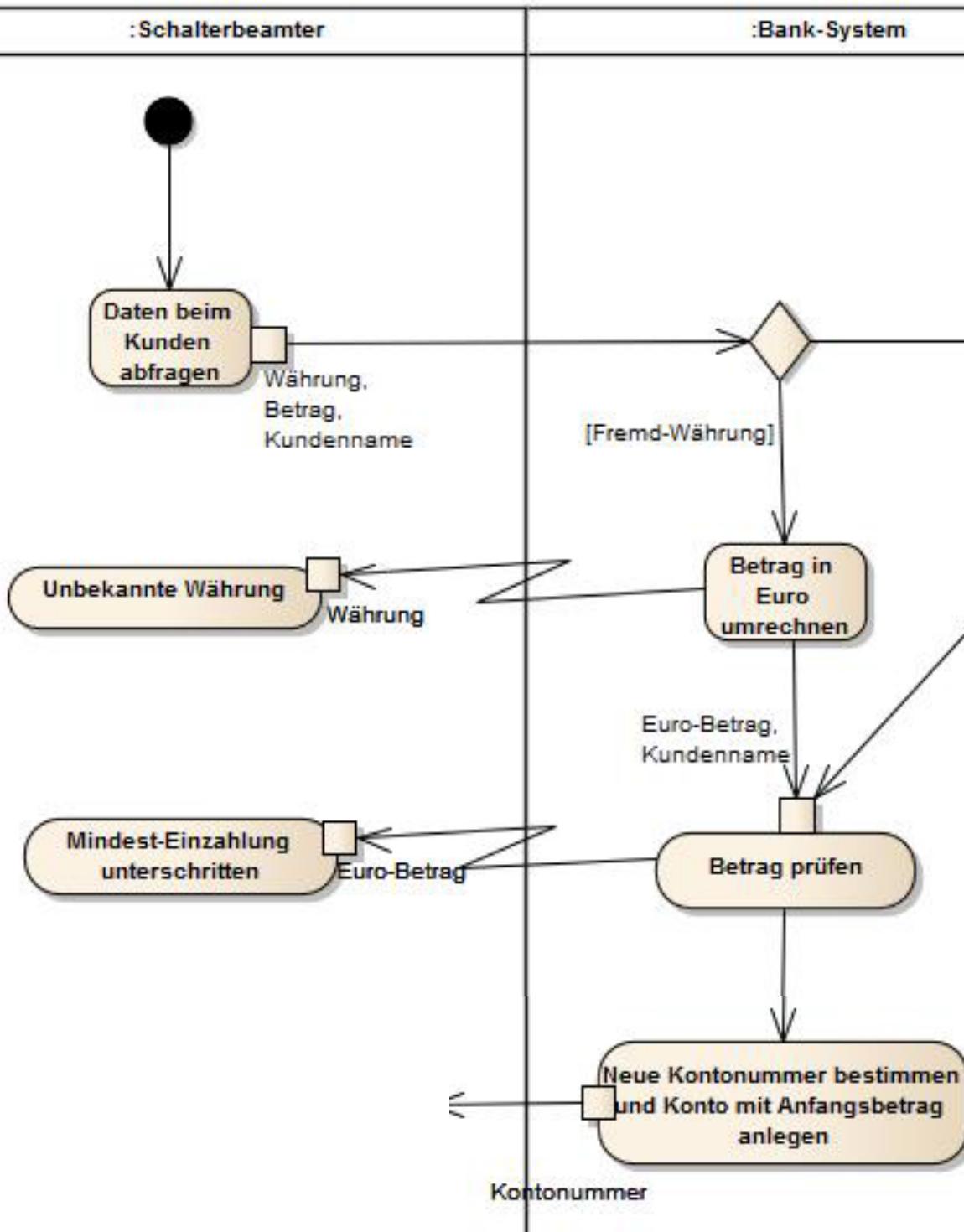
## Darstellung als Aktivitätsdiagramm



:Schalterbeamter

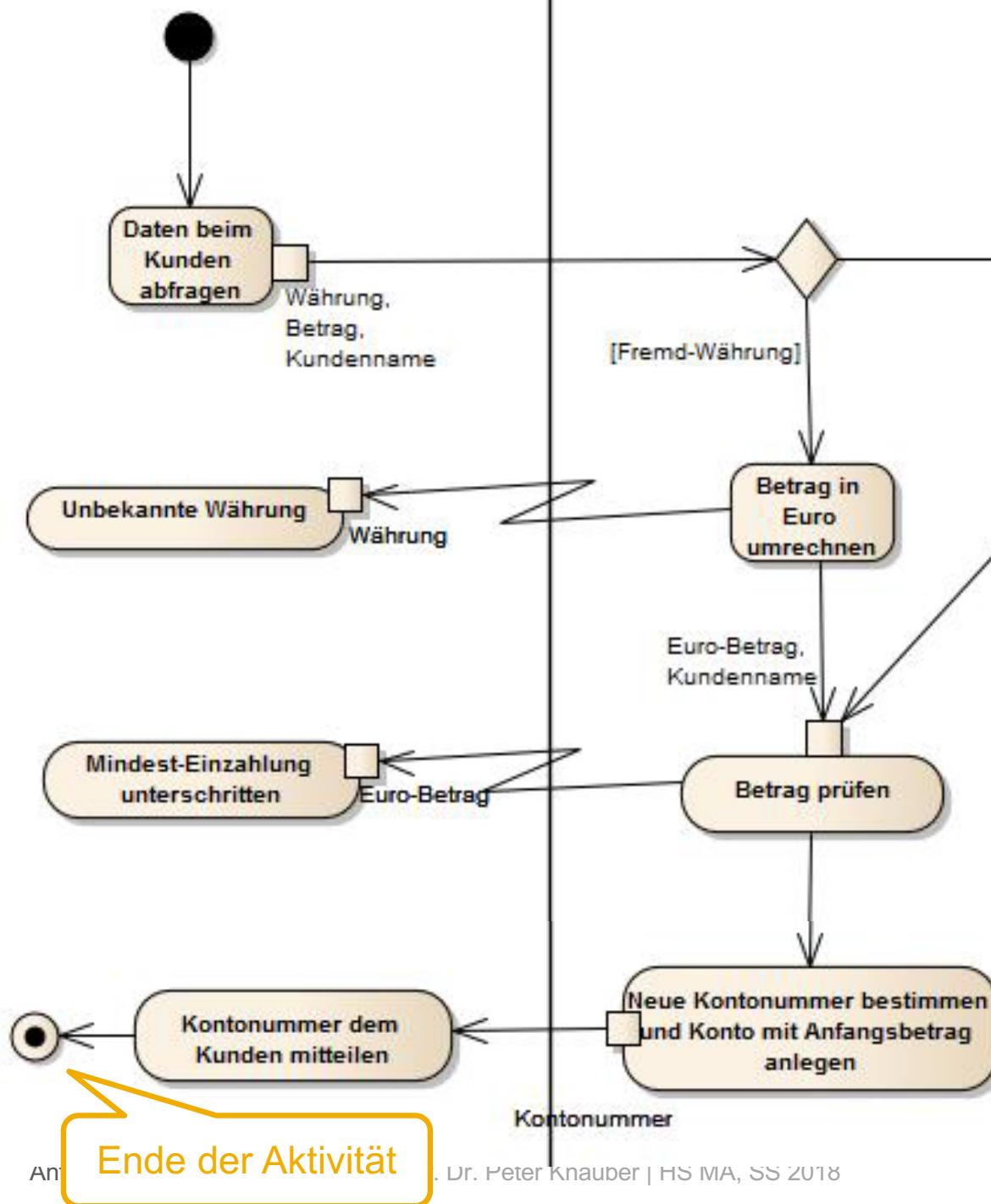
:Bank-System

## Darstellung als Aktivitätsdiagramm



:Schalterbeamter

:Bank-System



## Darstellung als Aktivitätsdiagramm

# Ziel der Erstellung von Aktivitätsdiagrammen

**Alle** Unklarheiten finden und beseitigen

*bevor* es ans Design,  
die Implementierung  
und das Testen geht  
(spätere Korrekturen sind viel teurer!)

Diagramme sind weniger missverständlich  
als Prosa-Text

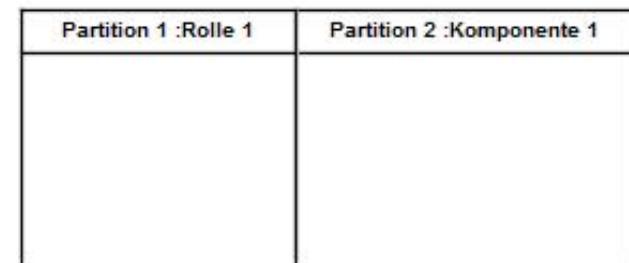
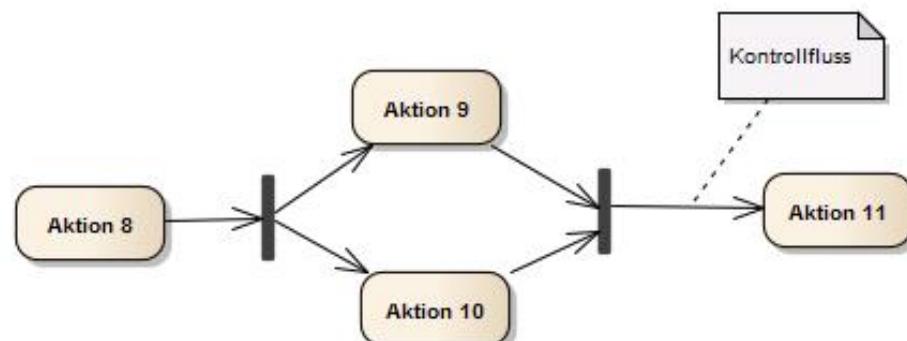
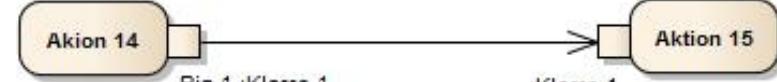
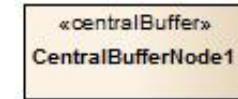
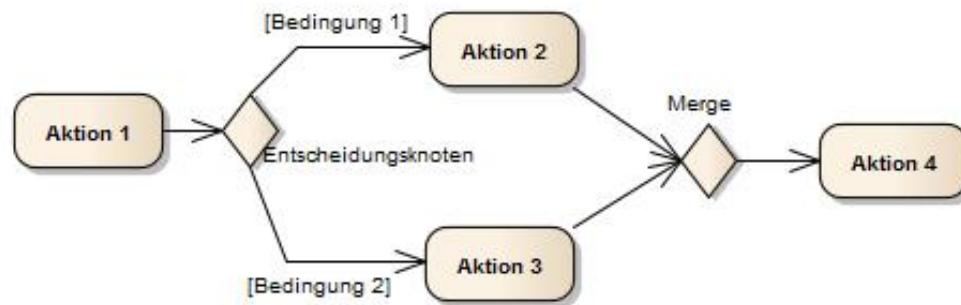
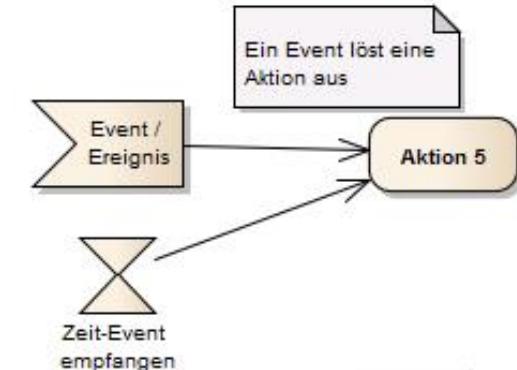
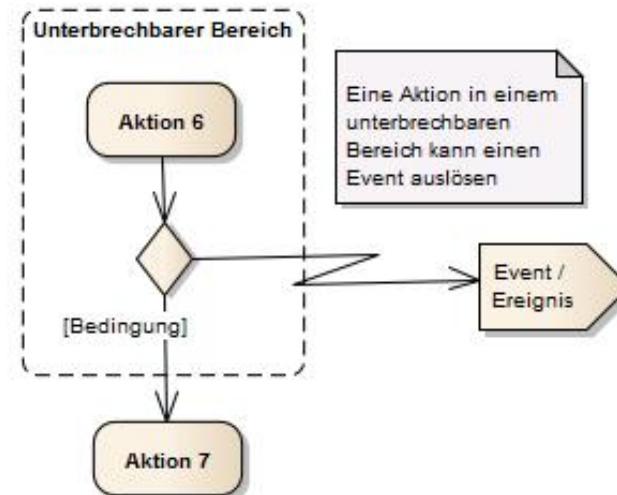
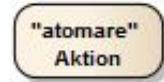
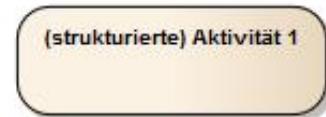
# Allgemeines

**Aktivitätsdiagramme** sind eine der vielen Diagrammformen der UML

- Sie beschreiben Folgen von Aktivitäten inklusive bedingtem und parallelem Verhalten
- Aktivitätsdiagramme in der Systemanalyse sind *immer* assoziiert mit einem *Use Case*: Sie beschreiben diesen Anwendungsfall, eventuell unter Benutzung anderer (Aktivitäts-)Diagramme  
Umgekehrt wird jeder *Use Case* durch ein Aktivitäts- (oder ein Sequenz-) Diagramm beschrieben
- Aktivitätsdiagramme ersetzen früher gebräuchliche Kontroll- und Datenfluss-Diagramme, Struktogramme etc.

# Elemente von Aktivitätsdiagrammen

## act Aktivitätsdiagramm Elemente



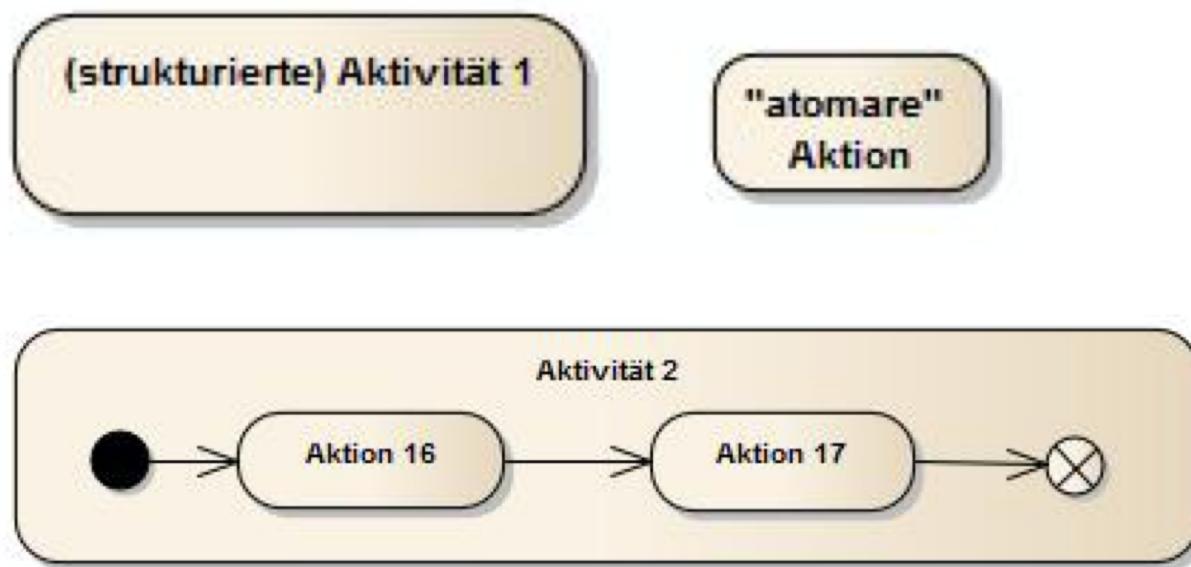
## Anfangs- und Endknoten



- Anfangsknoten bestehen aus einem dicken schwarzen Punkt
- Endknoten
  - Wird der Endknoten einer (gesamten) Aktivität erreicht, enden alle (möglicherweise parallelen) Teilflüsse
  - Der Endknoten eines Teilflusses beendet nur diesen Teilfluss

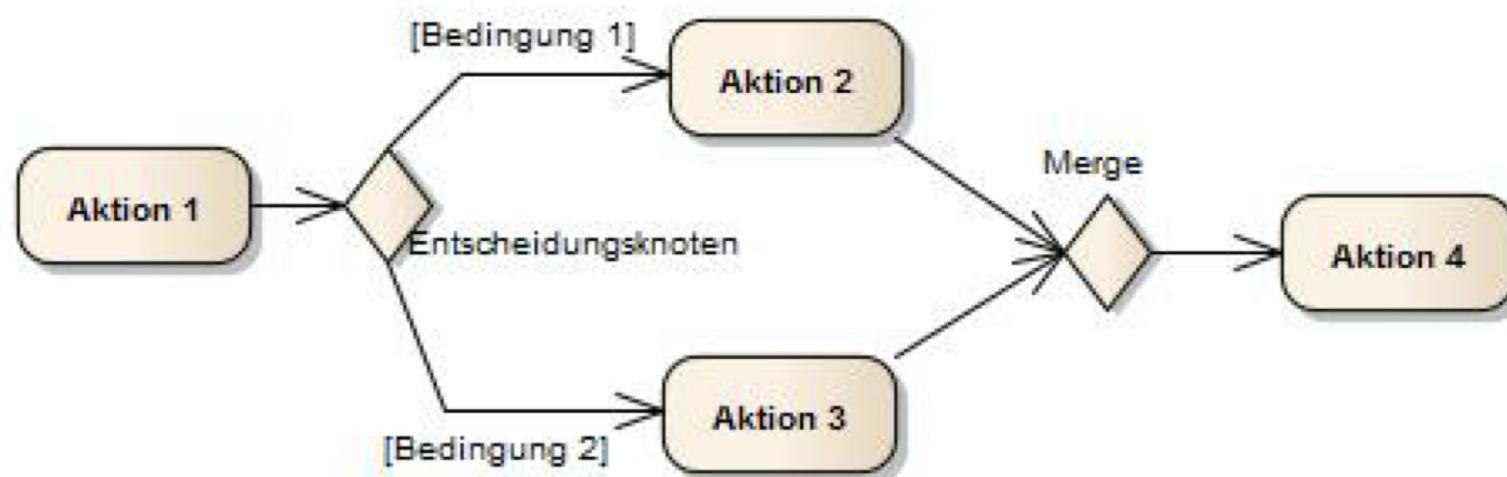
# Aktivitäten und Aktionen

- Eine Aktivität kann (meist) weiter unterteilt werden in „feinere“ Aktivitäten oder Aktionen
- Eine Aktion beschreibt einen einzelnen (atomaren) Schritt innerhalb einer Aktivität
- Der Unterschied ist graphisch nicht unbedingt sichtbar



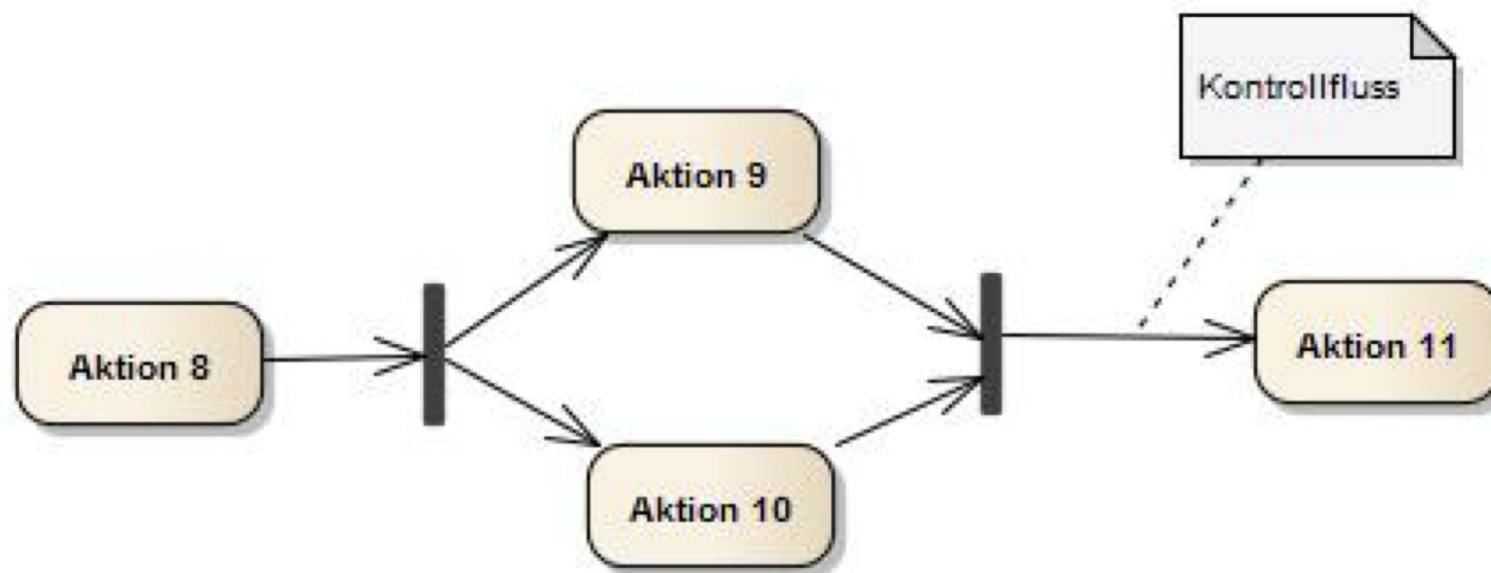
# Verzweigungs- und Vereinigungsknoten

- Verzweigungsknoten (decision) und Vereinigungsknoten (merge) werden jeweils durch eine Raute dargestellt
  - Beide können (zur Klarstellung) mit einem Namen versehen werden
  - Verzweigungsknoten haben einen eingehenden und mindestens zwei ausgehende Pfeile
  - Die davon ausgehenden Pfeile müssen (mit Ausnahme von einem) mit Bedingungen (guards) versehen werden



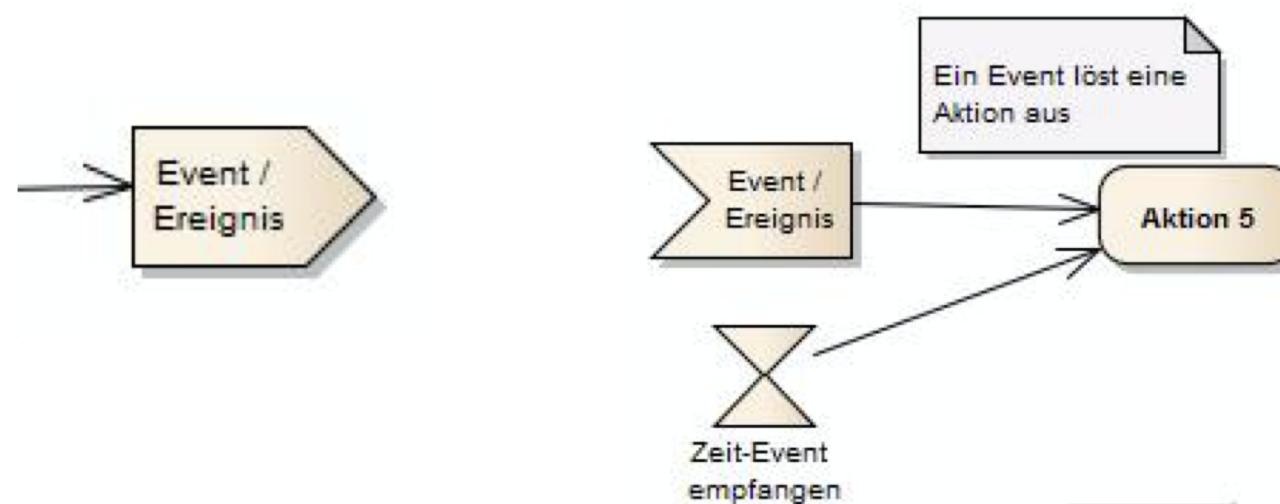
## Nebenläufigkeit (fork und join)

- Nebenläufigkeitsverzweigungen werden jeweils durch einen horizontalen oder vertikalen Balken dargestellt
  - Es gibt keinen Bedeutungsunterschied zwischen horizontalen und vertikalen Balken
- Diese Verzweigungen drücken aus, dass die Aktivitäten/Aktionen zwischen ihnen parallel ablaufen können



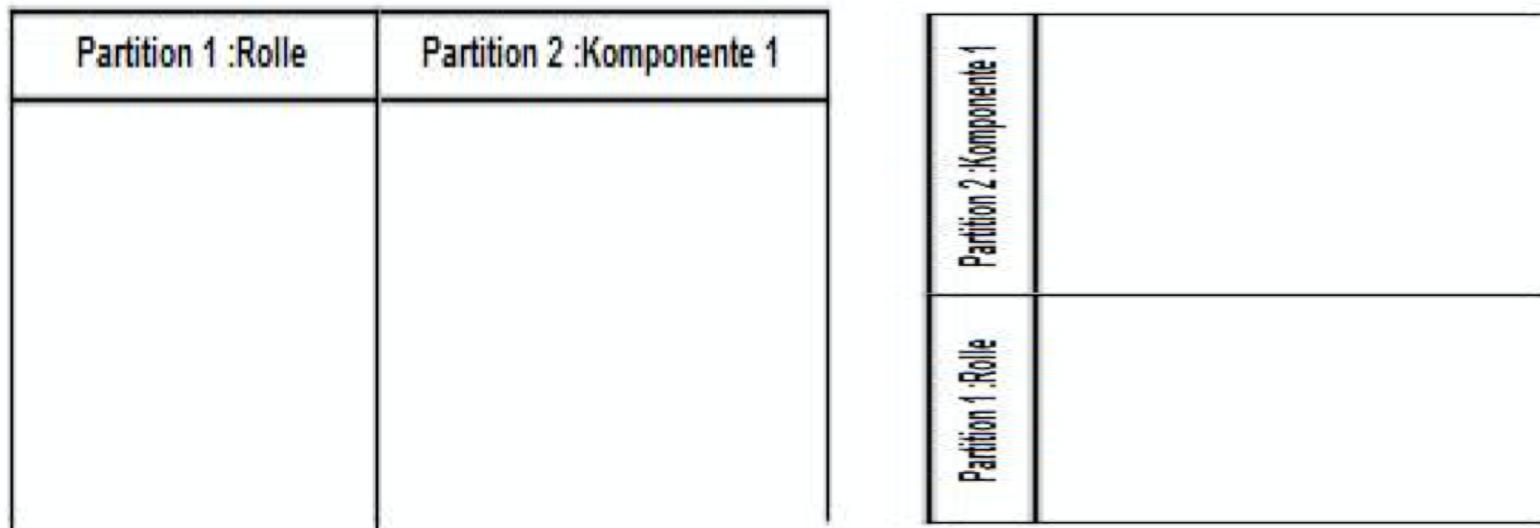
# Events / (Fehler-)Ereignisse

- Es gibt spezielle Darstellungsformen für Aktionen, die Events auslösen oder auf sie reagieren
  - Das Auslösen von Events wird durch ein konkaves Fünfeck mit dem Namen des Events dargestellt
  - Das Empfangen von Events wird durch ein konvaves Fünfeck mit dem Namen des Events dargestellt
  - Das zeitlich geplante Empfangen eines Events wird durch eine Sanduhr mit dem Namen des Events dargestellt



# Partitionen (Swim Lanes)

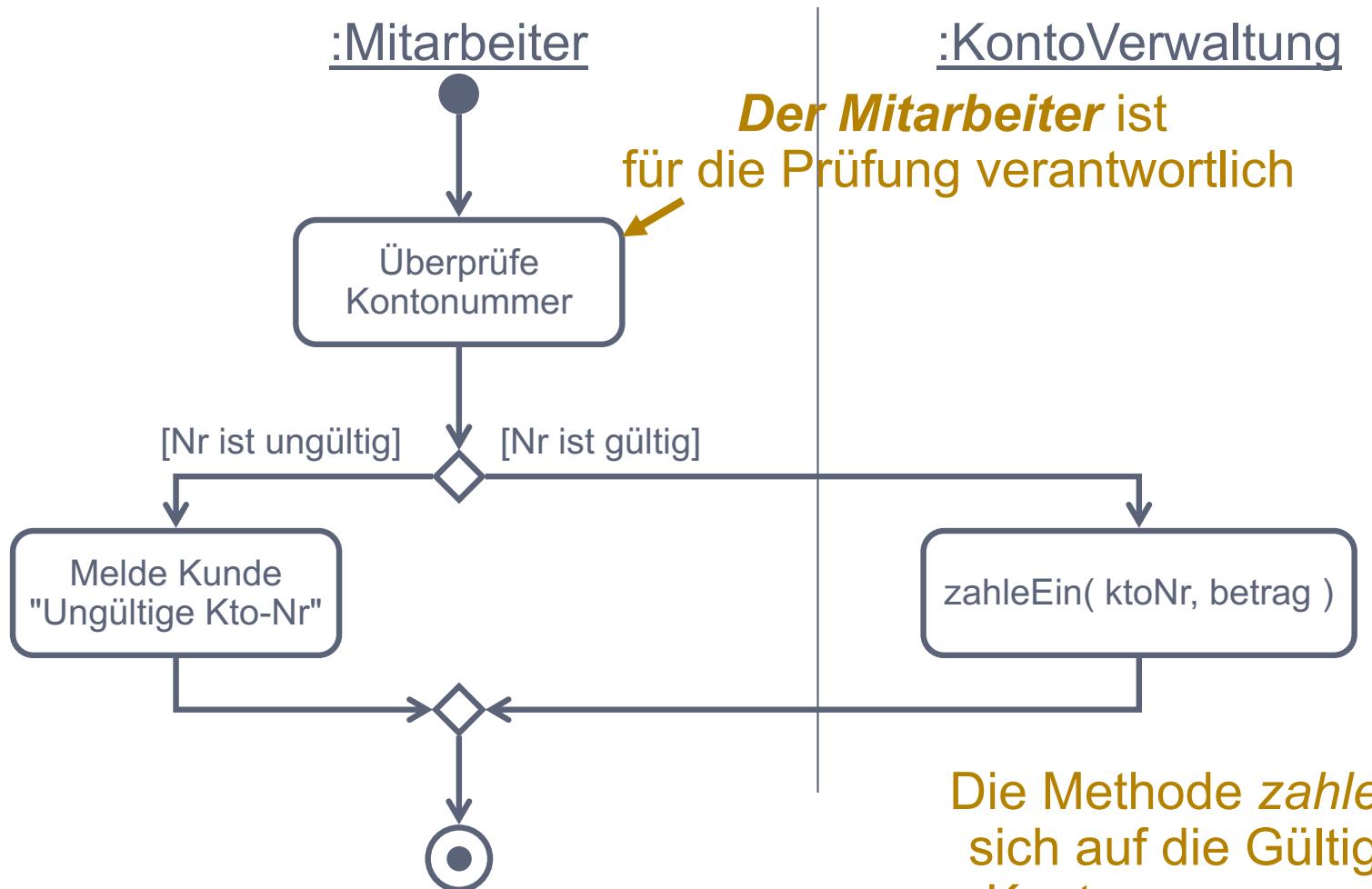
- Partitionen (hießen früher swim lanes) kennzeichnen Bereiche, für die bestimmte Rollen oder Systeme / Systemteile verantwortlich sind  
Die darin enthaltenen Aktivitäten / Aktionen werden von diesen Rollen / Systemen ausgeführt
- Partitionen können waagrecht oder senkrecht dargestellt werden



# Beispiel für die Anwendung von Partitionen: Variante 1

Was passiert, wenn beim Einzahlen auf ein Konto  
eine falsche Kontonummer angegeben wird?

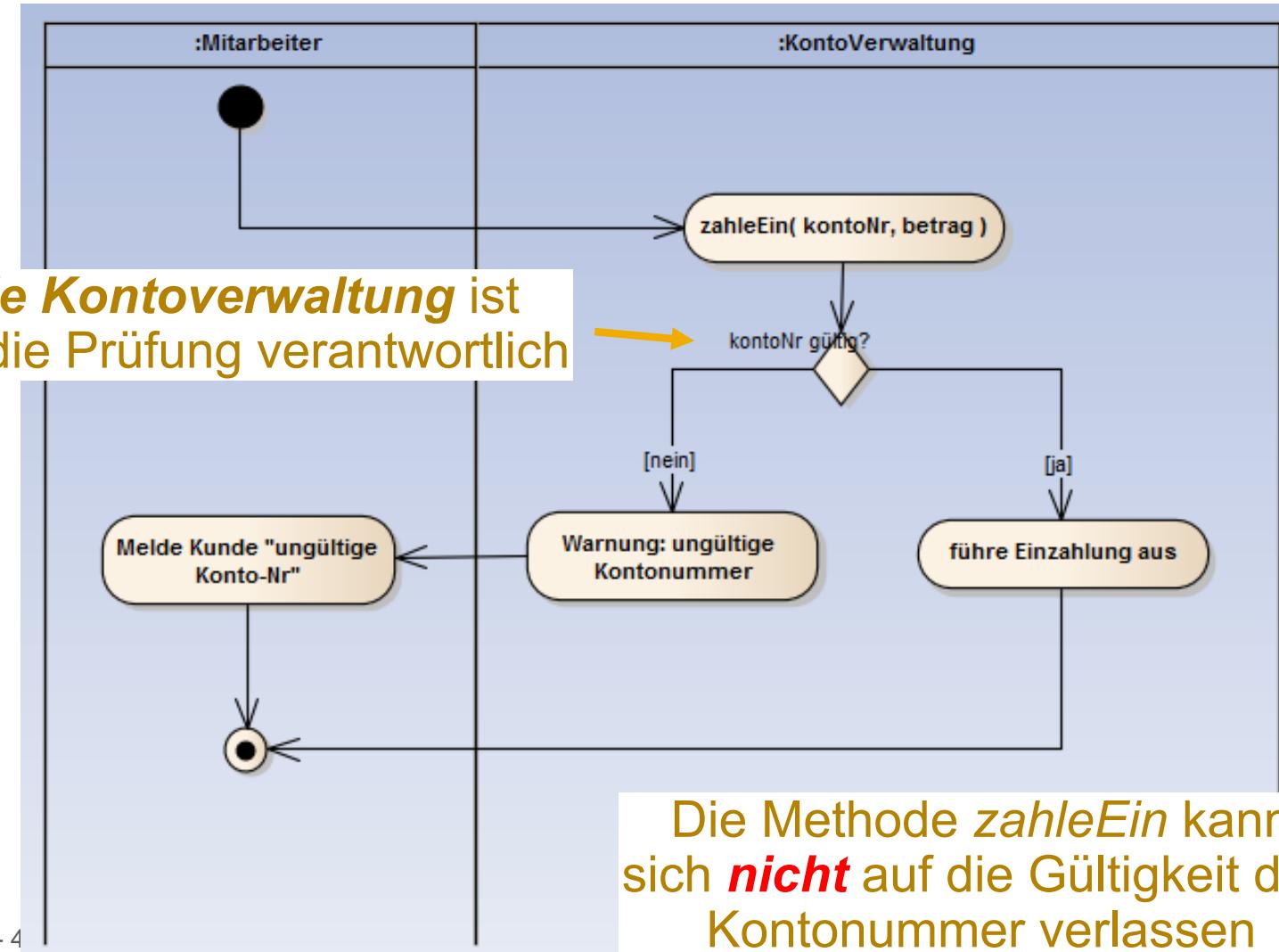
Wer überprüft das?



# Beispiel für die Anwendung von Partitionen: Variante 2

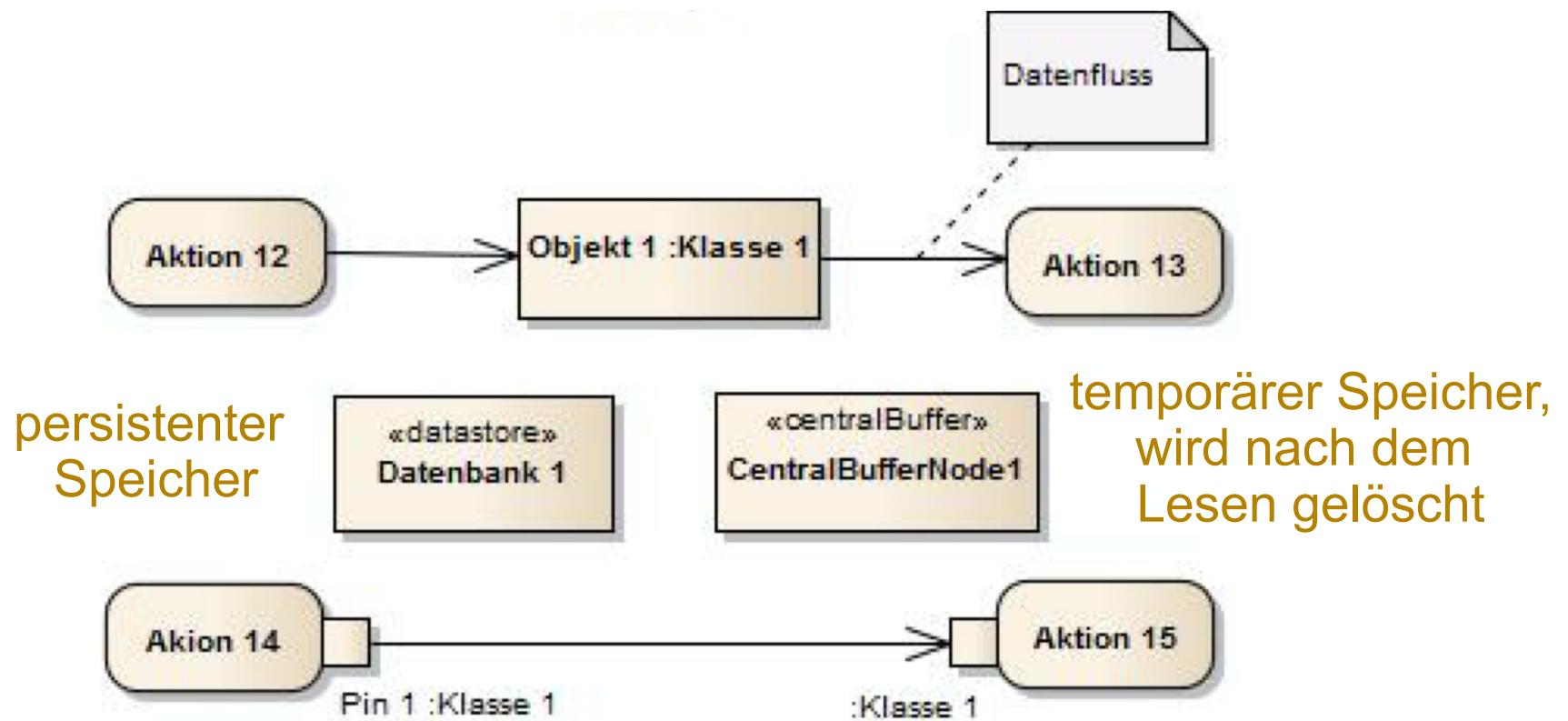
Was passiert, wenn beim Einzahlen auf ein Konto  
eine falsche Kontonummer angegeben wird?

Wer überprüft das?



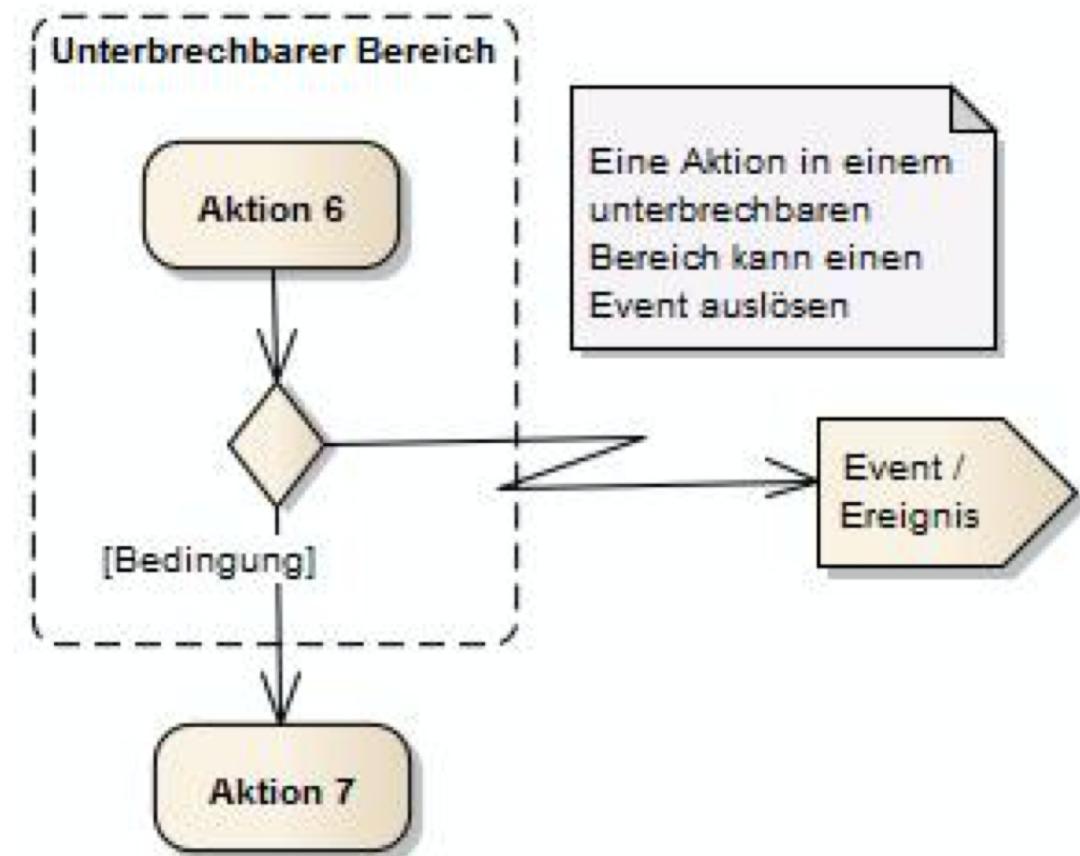
# Objektfluss

- Mittels **Objektfluss** wird dargestellt, welche Ein- oder Ausgaben bestimmte Aktivitäten / Aktionen haben
- Es gibt unterschiedliche, aber gleichbedeutende Darstellungsformen

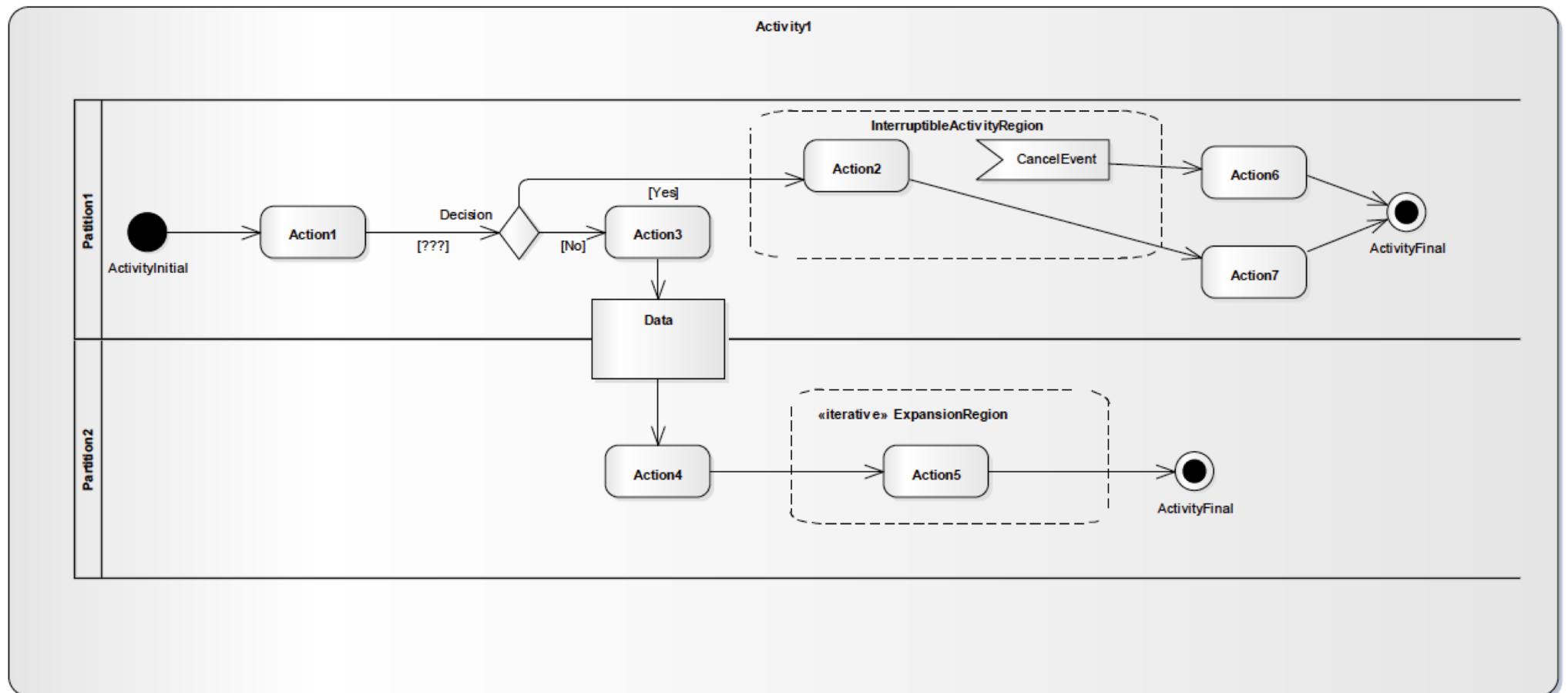


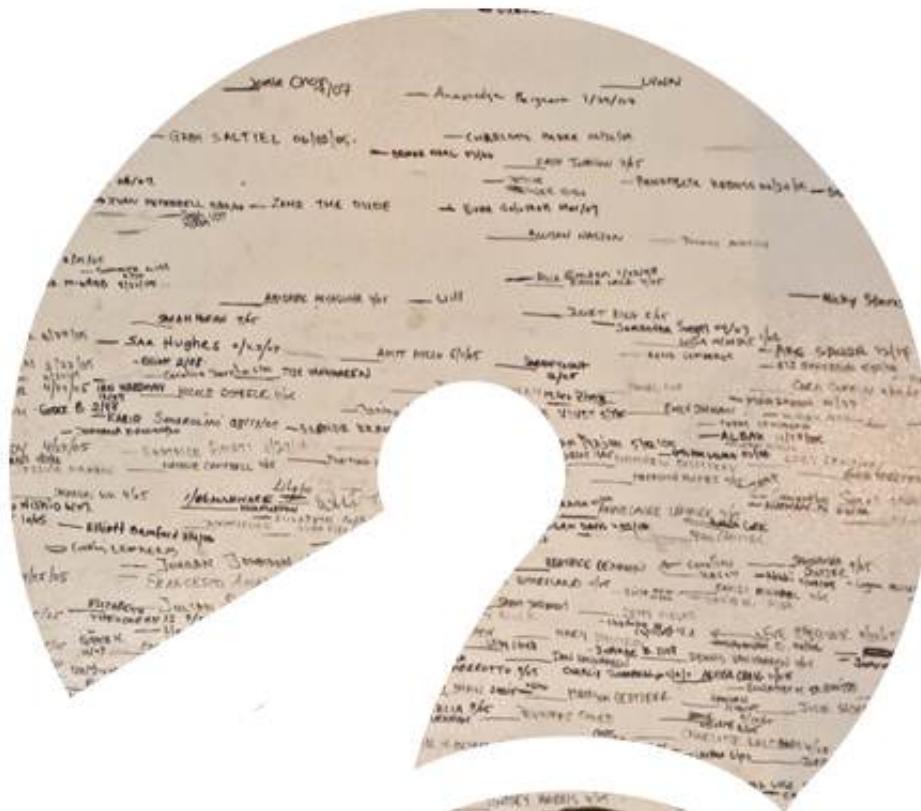
## Bereiche (regions)

- Bereiche dienen dazu, zusammengehörige Aktivitäten zu kennzeichnen
- Es gibt
  - iterative,
  - parallele,
  - stream- und
  - unterbrechbare Bereiche



# Komplett-Illustration





photography: woodleywonderworks  
<http://www.flickr.com/photos/wwworks/2350106729>  
art work: Peter Kaiser

# F R A G E N