

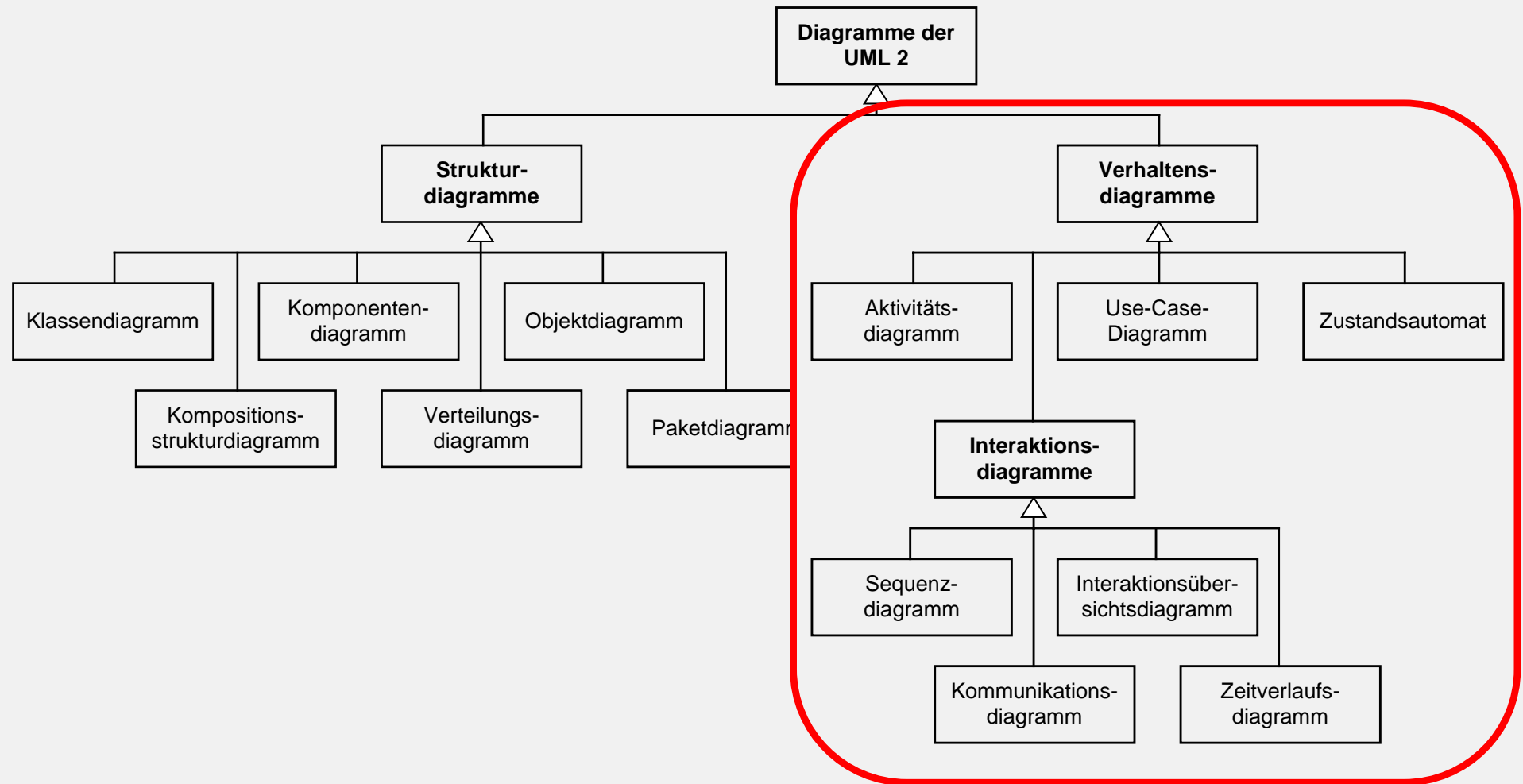
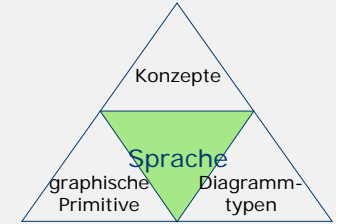
# Vorlesung "Software-Engineering"

---

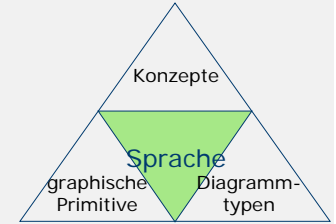
Prof. Ralf Möller, TUHH, Arbeitsbereich STS  
Übung: Miguel Garcia

- Thema: Spezifikation mit UML
  - Vorher: Strukturdiagramme
  - Heute: Verhaltensdiagramme Teil 1

# Die Verhaltensdiagramme

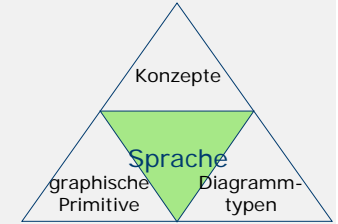


# Die Verhaltensdiagramme

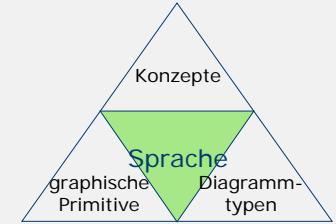


- **Use-Case Diagramm**  
Eine abstrakte funktionale Sicht auf das Gesamtsystem aus Sicht des späteren Anwenders
- **Aktivitätsdiagramm**  
Darstellung eines dynamischen Ablaufs
- **Zustandsautomat**  
Beschreibt die internen Zustände eines Classifiers
- **Sequenzdiagramm**  
Beschreibt den Intra- und Intersystem-Datenaustausch
- **Kommunikationsdiagramm**  
Statische Sicht auf dynamische Interaktion
- **Timing-Diagramm**  
Zeitabhängige Zustandsdarstellung
- **Interaktionsübersichtsdiagramm**  
Darstellung des Zusammenspiels verschiedener Interaktionen

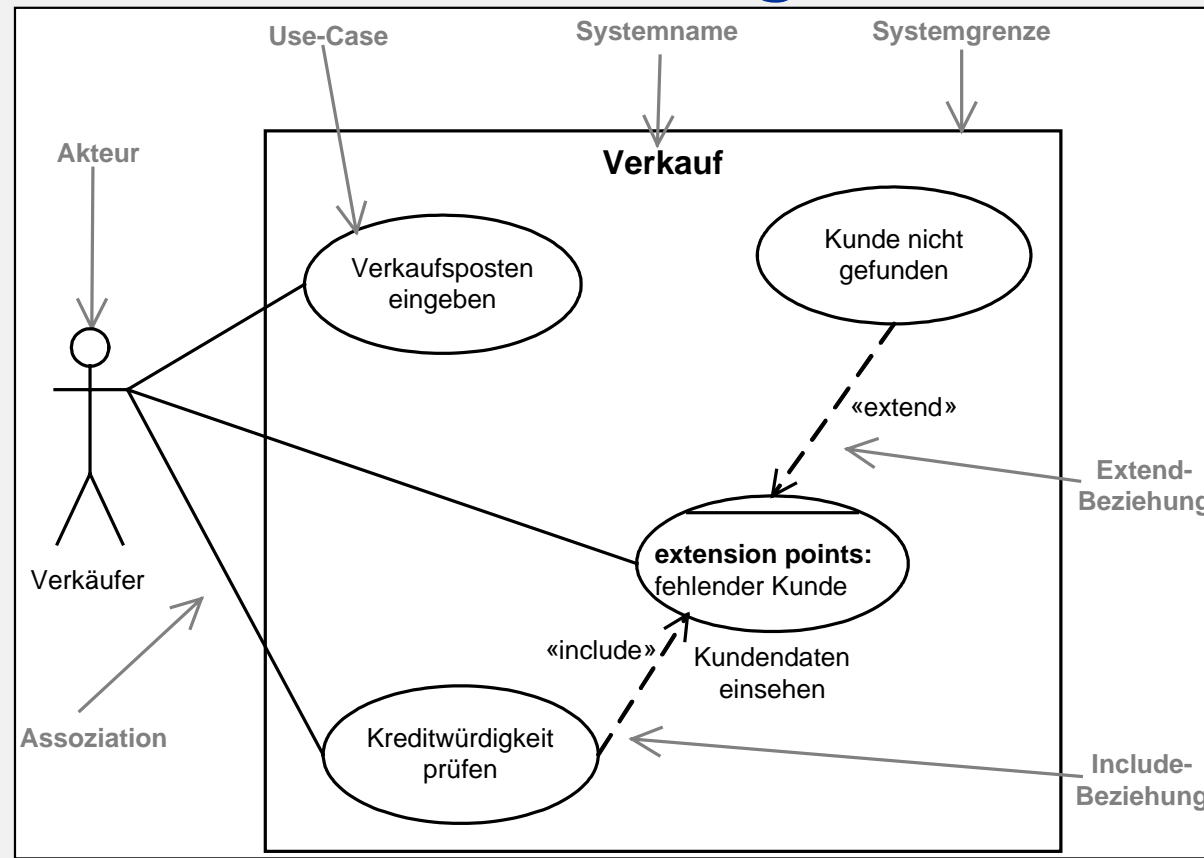
# Use-Case-Diagramm



- **Aufgabe:**
  - Darstellung einer abstrakten funktionalen Sicht auf das Gesamtsystem aus Sicht des späteren Anwenders
- **Aussage:**
  - Technikfern spezifizierter Leistungsumfang des Systems
- **Aufgabe im Projekt:**
  - Dokumentation erster früher Analyseergebnisse
- **Änderungen durch UML 2:**
  - Akteur muß zwingend benannt sein
  - Vorbedingungen und Erweiterungspunkte werden als Notiz notiert
  - Classifier können Use Cases besitzen
  - Classifier können Use Cases realisieren

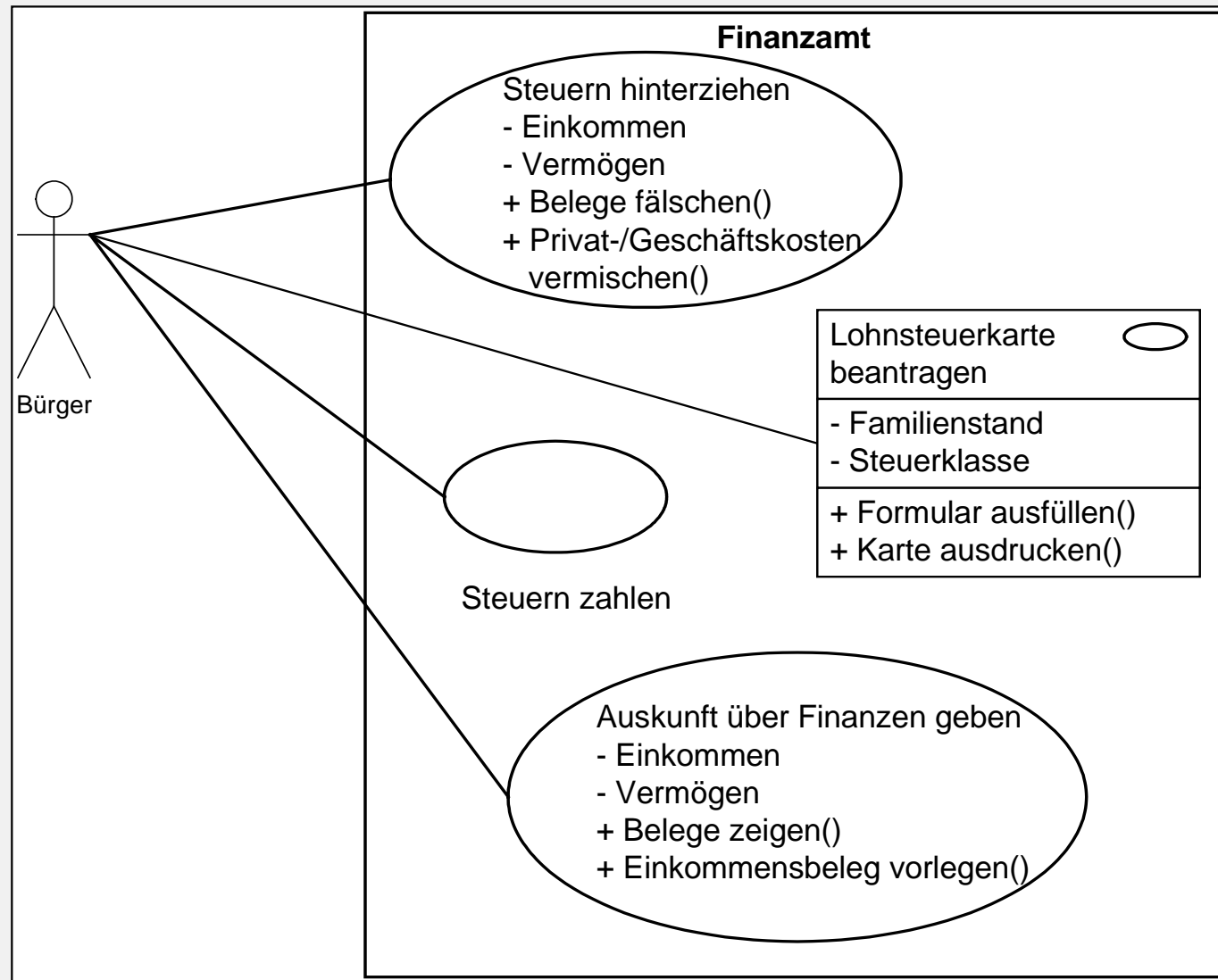
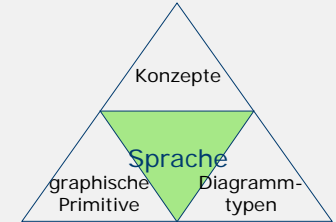


# Use-Case-Diagramm

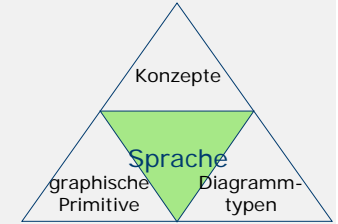


- System (Betrachtungsgegenstand):
  - Umgrenzt die Einheit, welche die Use-Cases realisiert
  - Darstellung ist nicht zwingend notwendig
- Assoziationen:
  - beschreiben die Beziehungen zwischen Akteuren und Use-Cases
  - extend-Beziehung: Verhalten eines Use-Case kann durch einen anderen erweitert werden
  - include-Beziehung: Verhalten eines Use-Case ist vollständig in einem anderen enthalten

# Use-Case-Diagramm

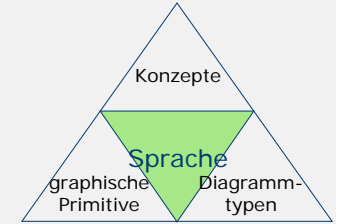
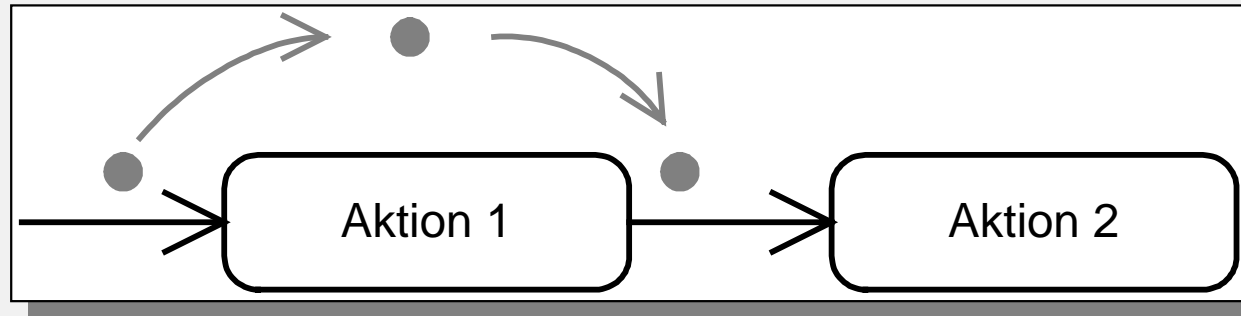


# Aktivitätsdiagramm



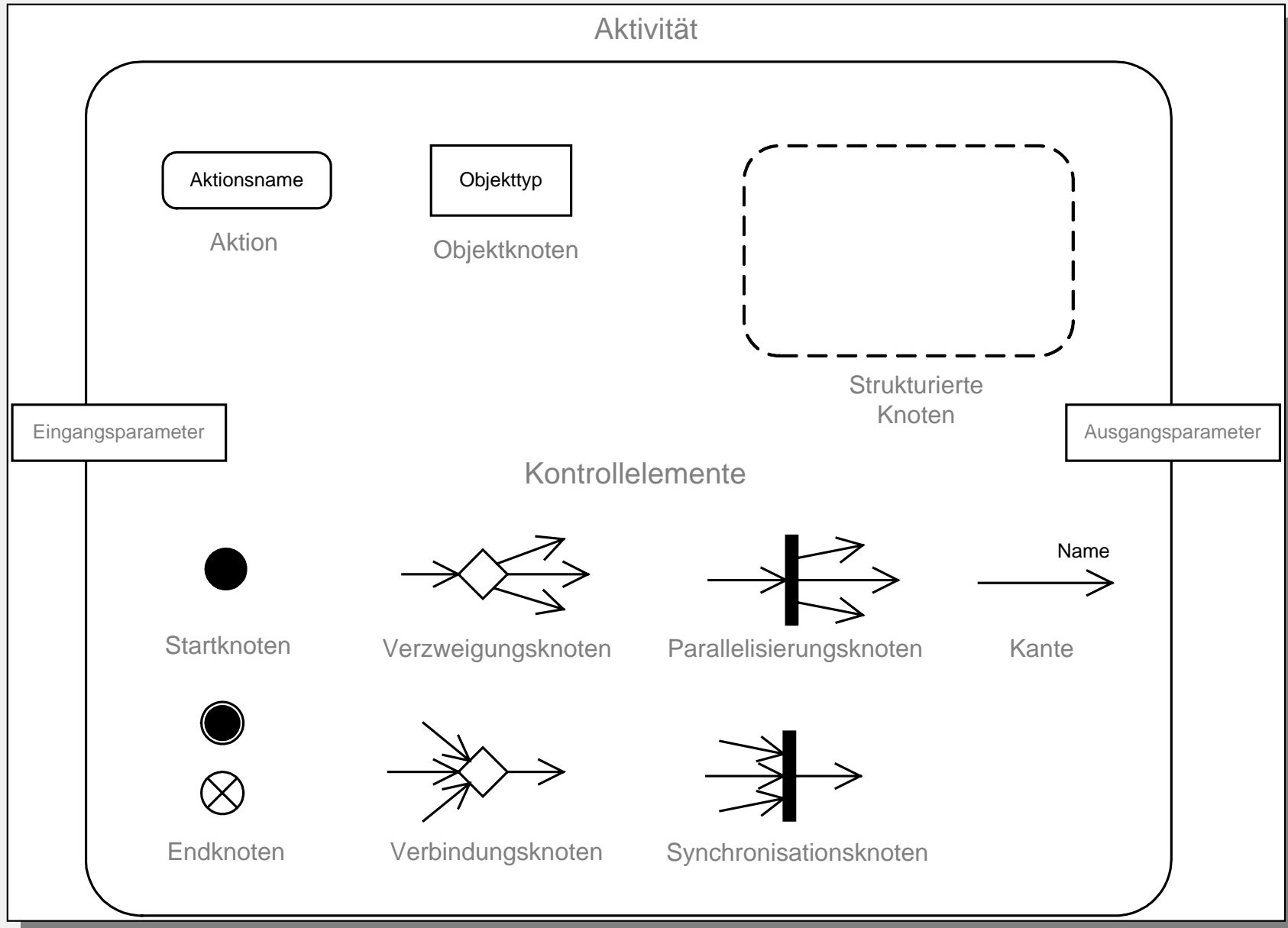
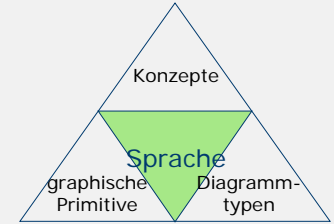
- **Aufgabe:**
  - Darstellung eines dynamischen Ablaufs
- **Aussage:**
  - Realisierung eines bestimmten Verhaltens durch das System
- **Aufgabe im Projekt:**
  - Geschäftsprozeßmodellierung
  - Beschreibung von Use Cases
  - Dokumentation der Implementierung einer Operation
- **Änderungen durch UML 2:**
  - Aktivitäten unabhängig von Zustandsautomaten
  - Petri-Netz-ähnliche Semantik
  - Diagrammtyp wird als *Aktivität* bezeichnet
  - Vor- und Nachbedingungen
  - Notation der Aktion und des Zustandes vereinheitlicht
  - Multiple Startknoten
  - Verschiedene End(-knoten)-Semantiken
  - Neue Notationselemente
  - ...

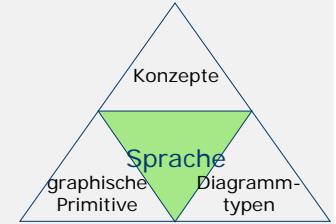
# Aktivitätsdiagramm



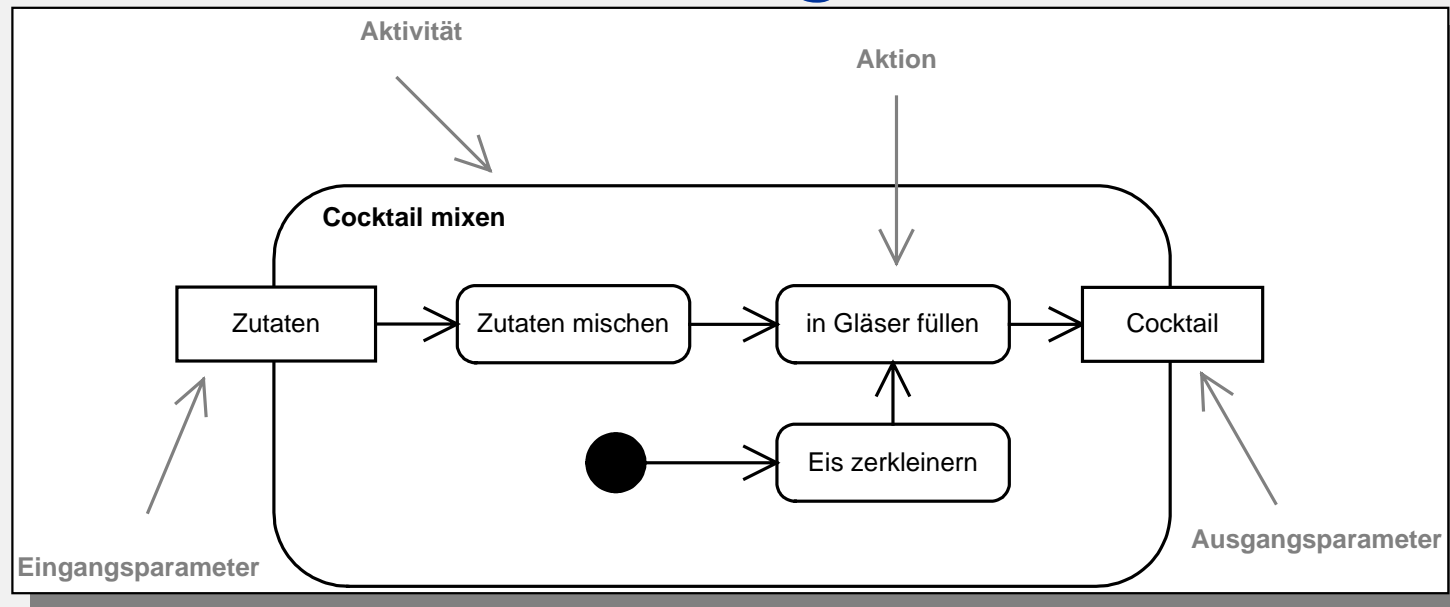
- Tokenkonzept
  - aus den Petri-Netzen übernommen
  - Tokenfluss steuert Ablauf einer Aktivität
  - Ermöglicht die präzise Beschreibung des Verhaltens
  - nur gedankliches Konstrukt (keine explizite Modellierung)

# Aktivitätsdiagramm





# Aktivitätsdiagramm

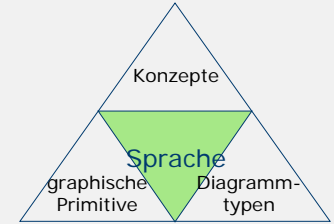


- Diagrammtyp heißt *Aktivität*
- Eine Aktivität kann Ein- und Ausgangsparameter besitzen
- Aktionen sind Verhaltensaufrufe
- Summe der Aktionen realisiert die Aktivität

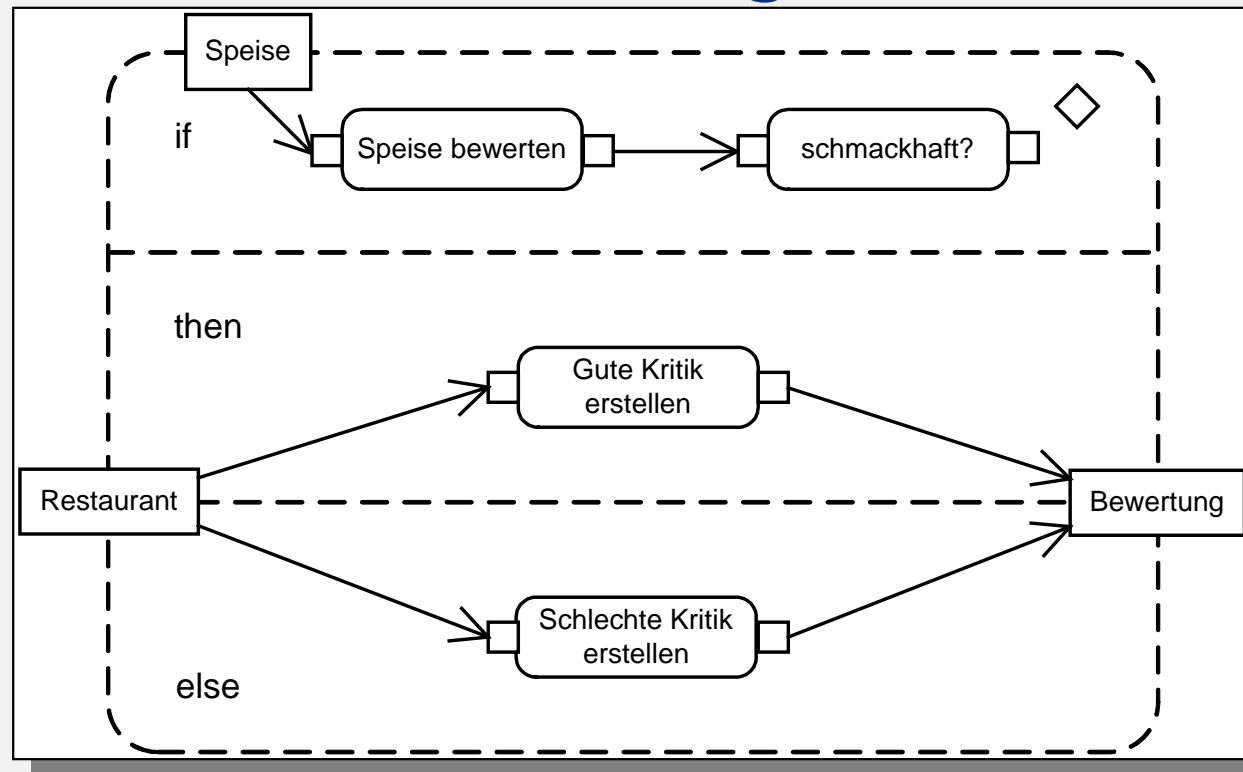




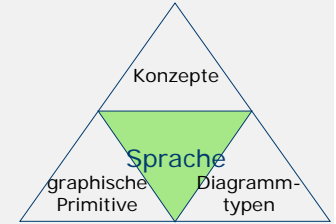




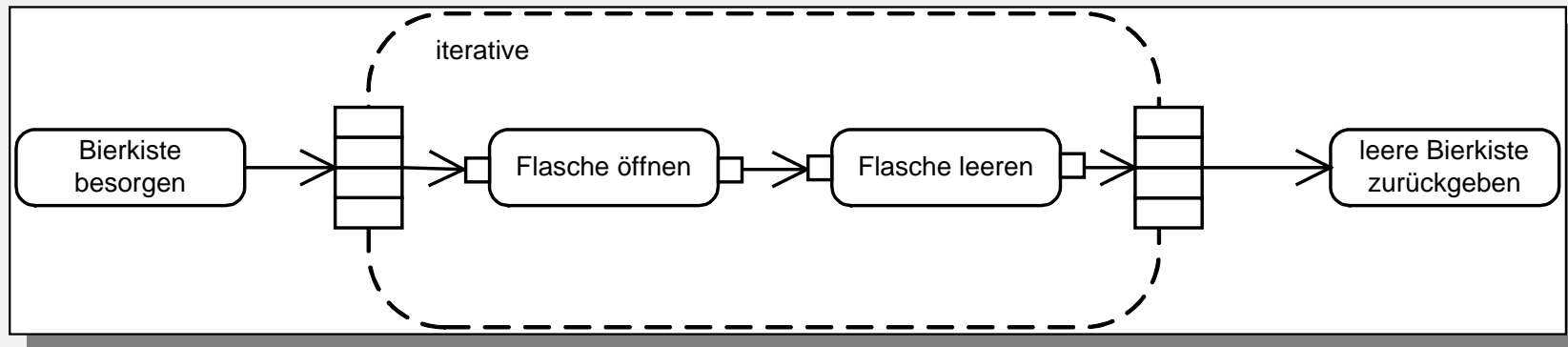
# Aktivitätsdiagramm



- Strukturierte Knoten zur Visualisierung komplexer Entscheidungen
- `if`: prüfen der Bedingung
- `then`: auszuführende Elemente
- `else`: möglicher Ablauf, wenn kein `if`-Bereich zutrifft
- `else if`: wie `if`-Bereich nur mit vorgegebener Prüfreihenfolge

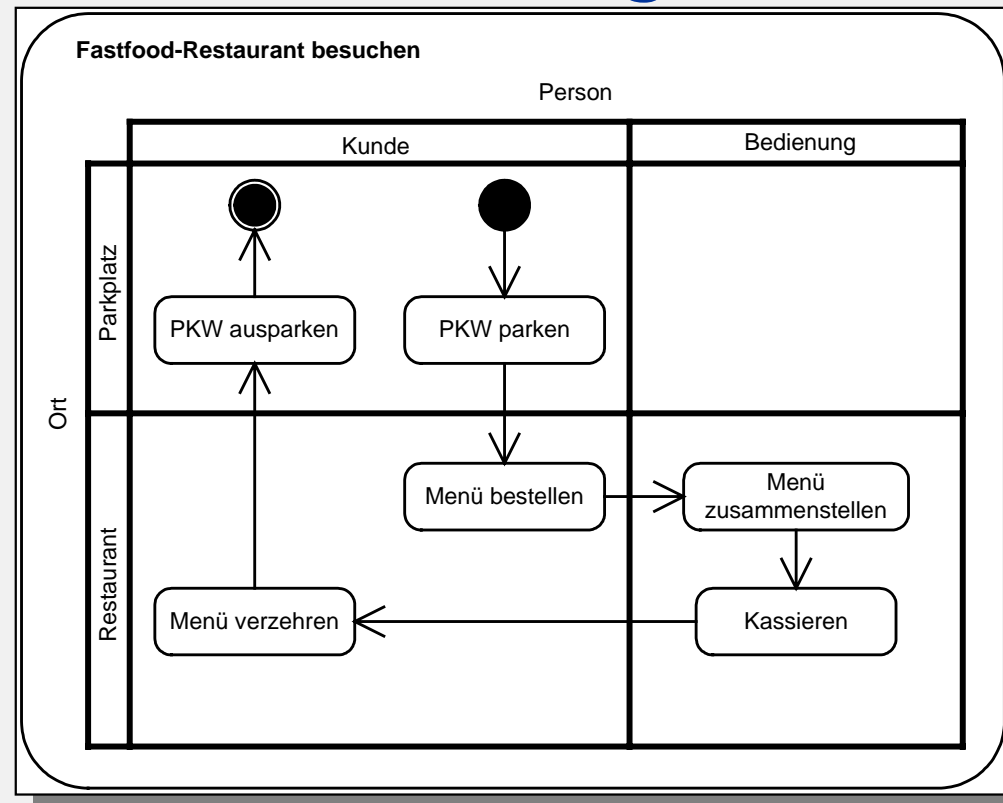
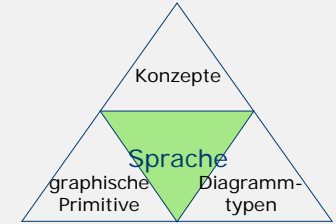


# Aktivitätsdiagramm



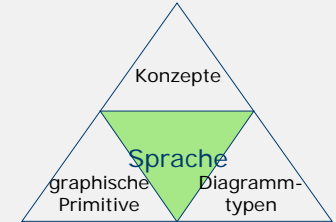
- Mengenverarbeitung
  - Einzelne Betrachtung der Elemente welche in der restlichen Aktivität nur als Sammlung betrachtet werden
  - z.B. Listen, Vektoren, hashtable...
  - Elemente werden als Objektknoten (Pin) übergeben

# Aktivitätsdiagramm



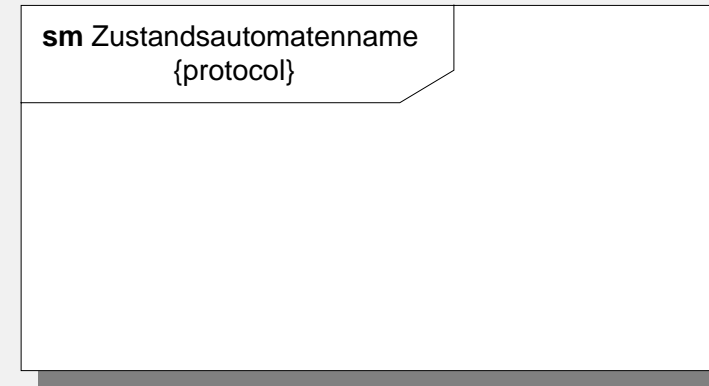
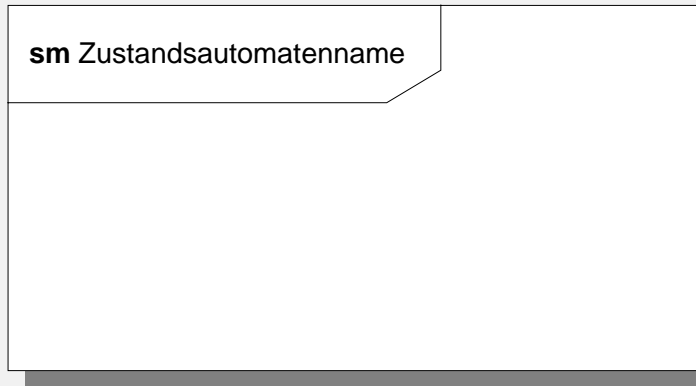
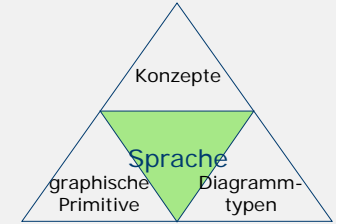
- Aktivitätsbereiche
  - Teilung des Diagramms logisch gruppierte Partitionen
  - Hierarchische und mehrdimensionale Partitionierung möglich

# Zustandsdiagramm



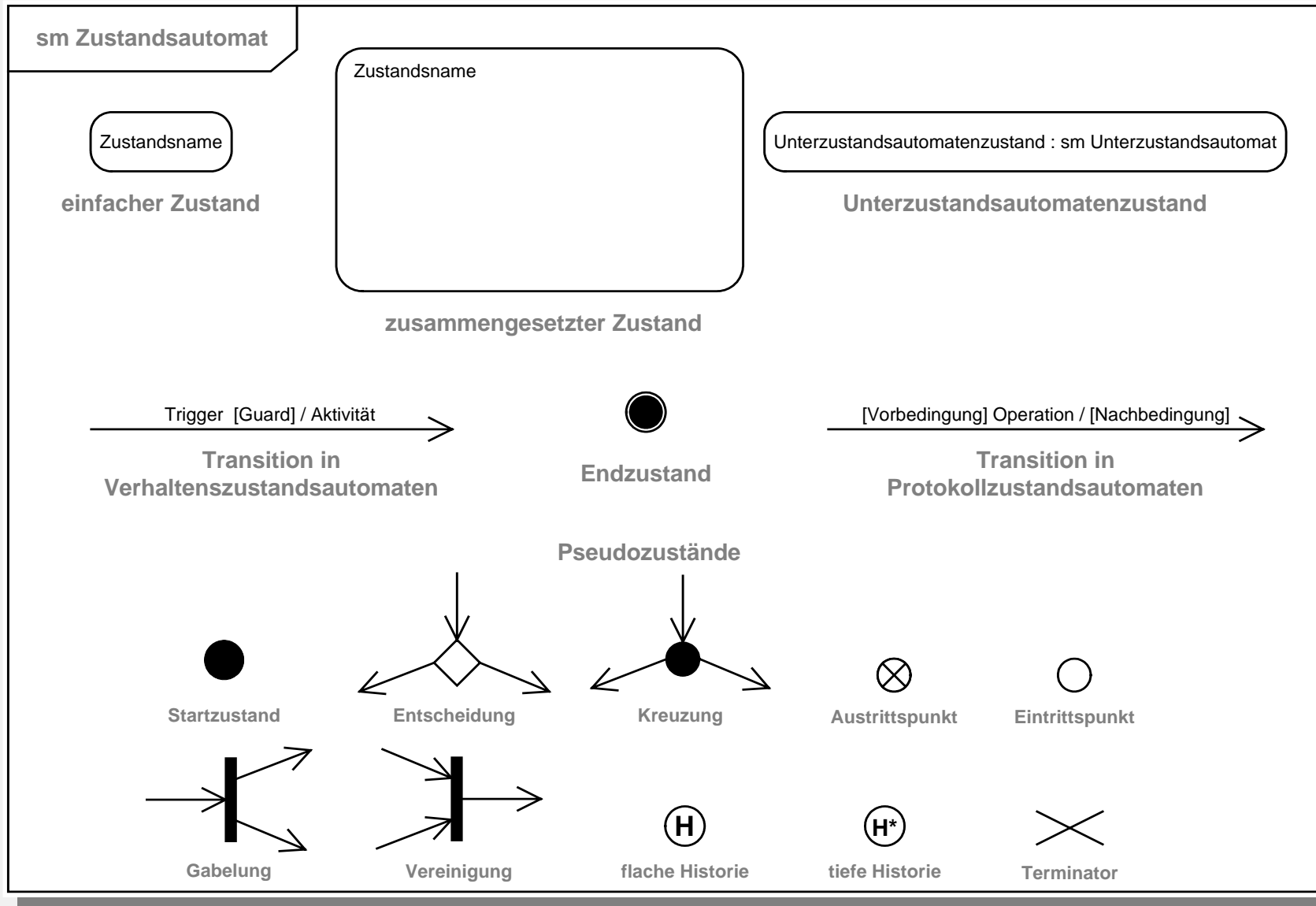
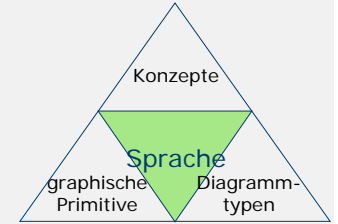
- **Aufgabe:**
  - Beschreibt die internen Zustände eines Classifiers
- **Aussage:**
  - Zugelassene Status eines Classifiers durch Betrachtung als Zustandsautomat
- **Aufgabe im Projekt:**
  - Zustandsbeschreibung eines Classifiers
  - Detaillierung eines Use Cases
  - Verhaltensbeschreibung einer extern angebotenen Schnittstelle
- **Änderungen durch UML 2:**
  - *Protokollzustandsautomat* neu eingeführt (Spezialisierung des allgemeinen Zustandsdiagramms)
  - Explizierung von Ein- und Austrittspunkten sowie Terminatoren
  - Vererbungssemantik (Overriding und Extension) geregelt

# Zustandsdiagramm

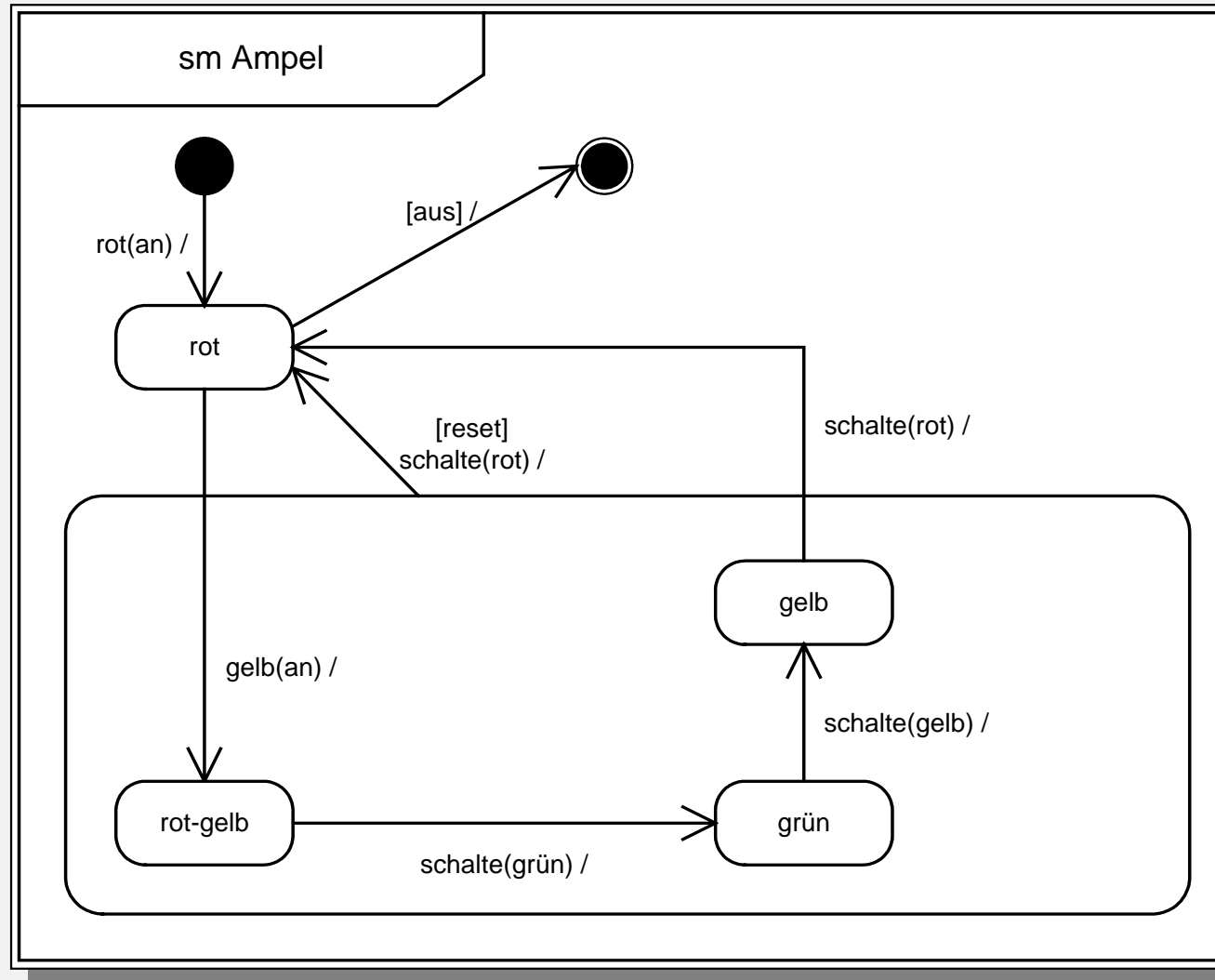
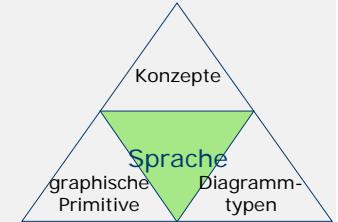


- Unterscheidung:
  - Verhaltenszustandsautomaten (Zustandsdiagramm)
  - Protokollzustandsautomaten
- Ein Verhaltenszustandsautomat bildet das diskrete Verhalten einer Instanz eines Classifiers ab.
- Ein Protokollzustandsautomat beschreibt die erlaubte Aufrufsabfolge der Instanz eines Classifiers.

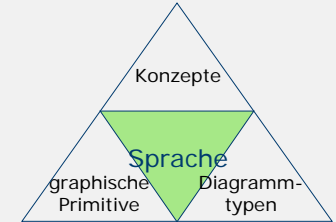
# Zustandsdiagramm



# Zustandsdiagramm

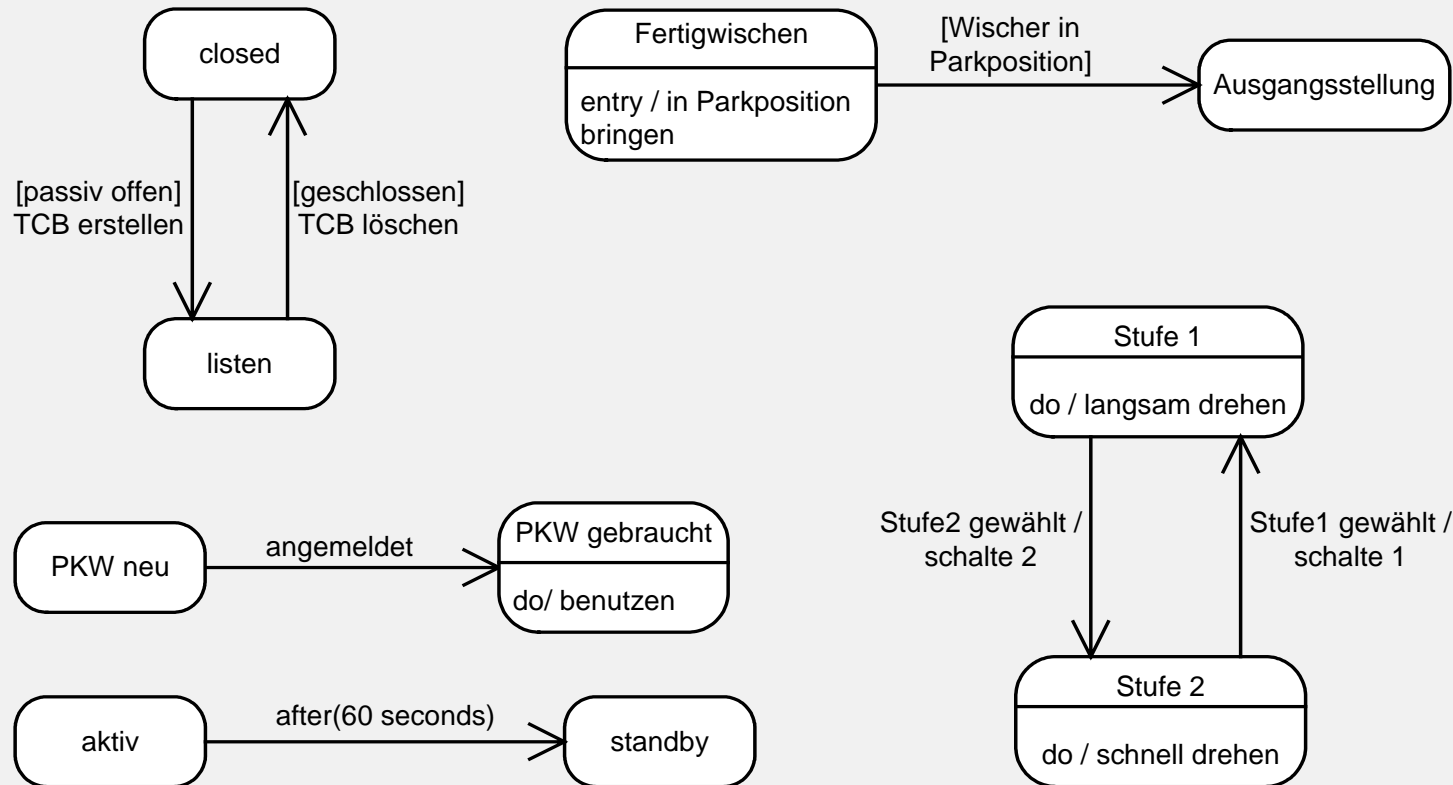
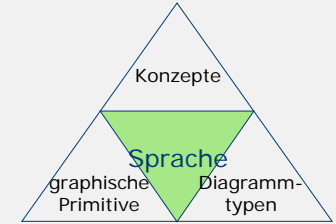


# Zustandsdiagramm



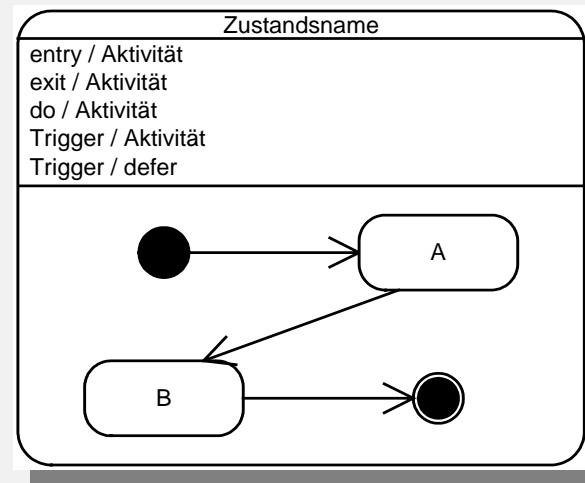
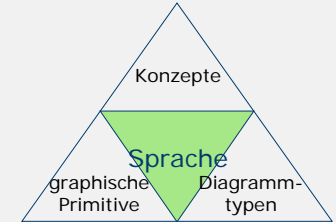
- Ein Zustand beschreibt eine bestimmte Ausprägung:
  - eine statische Situation
  - auf ein externes Ereignis wartend
- Zuständige können Aktivitäten enthalten:
  - entry, do und exit activity
  - verzögerte Ereignisse
  - Eine *Transition* ist der Übergang von einem Quell- in einen Zielzustand

# Zustandsdiagramm



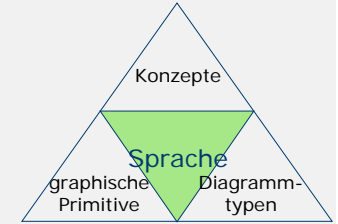
- Verschiedene Zustandsübergänge

# Zustandsdiagramm



- Zusammengesetzter Zustand:
  - Setzt sich aus Zuständen, Pseudozuständen und Transitionen zusammen
  - Steht stellvertretend für einen vollständigen Zustandsautomaten
  - Kann Ein- und Austrittspunkte besitzen

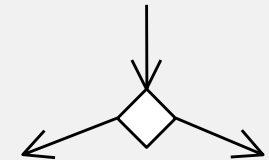
# Zustandsdiagramm



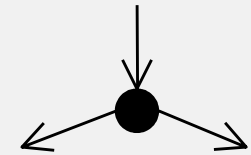
- Startzustand:
  - Verweist auf den ersten Zustand
  - Einer pro Region
- Entscheidung:
  - Ausgehende Transition wird während der Ausführung der Transition bestimmt
- Kreuzung:
  - Ausgehende Transition ist vor der Ausführung der Transition bekannt
- Ein- und Austrittspunkt:
  - Zum Betreten und Verlassen von Unterzustandsautomaten



Startzustand



Entscheidung



Kreuzung



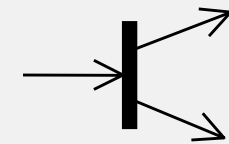
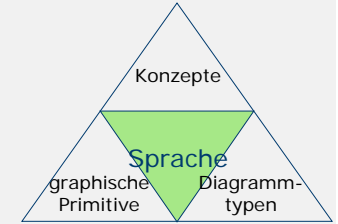
Austrittspunkt



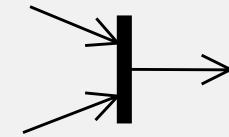
Eintrittspunkt

# Zustandsdiagramm

- Gabelung und Vereinigung:
  - Teilen eine Transition auf mehrere parallele Zustände auf bzw. fügen mehrere Transitionen zu einer zusammen
- Flache Historie:
  - Speichert den zuletzt aktiven Unterzustand eines komplexen Zustands
- Tiefe Historie:
  - Speichert den zuletzt aktiven Unterzustand eines in einem komplexen Zustand enthaltenen Zustand
- Terminator:
  - Bei Erreichen endet die Lebensdauer der Instanz des beschriebenen Classifiers



Gabelung



Vereinigung



flache Historie

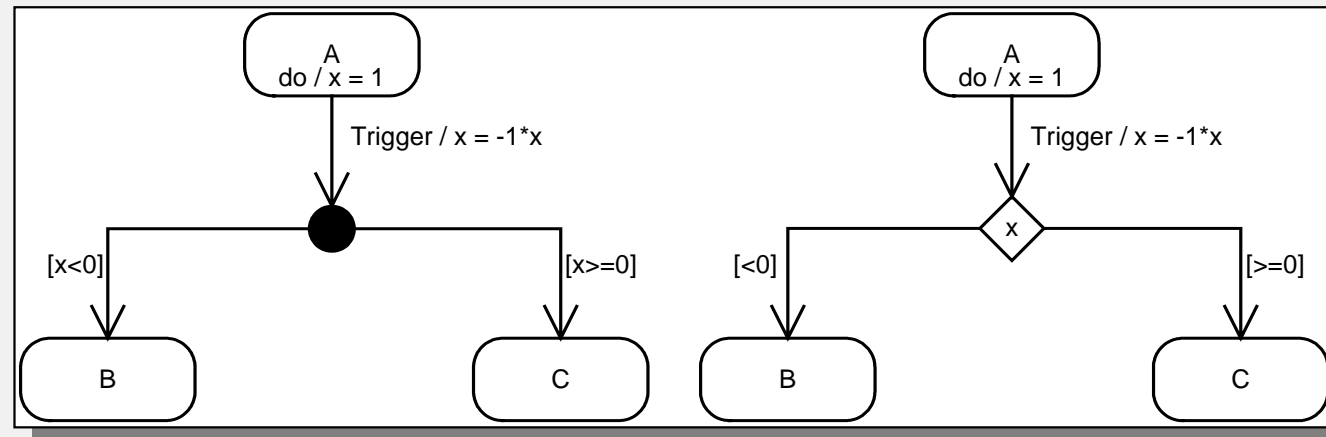
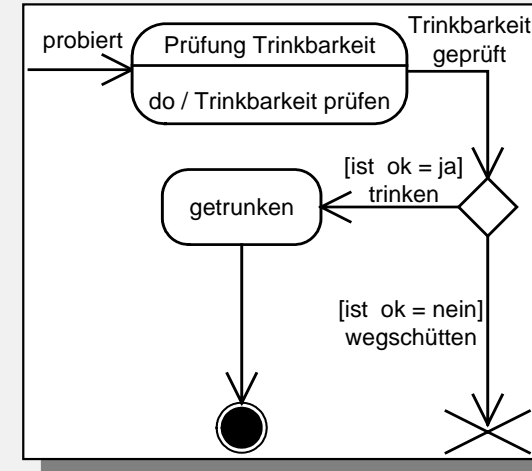
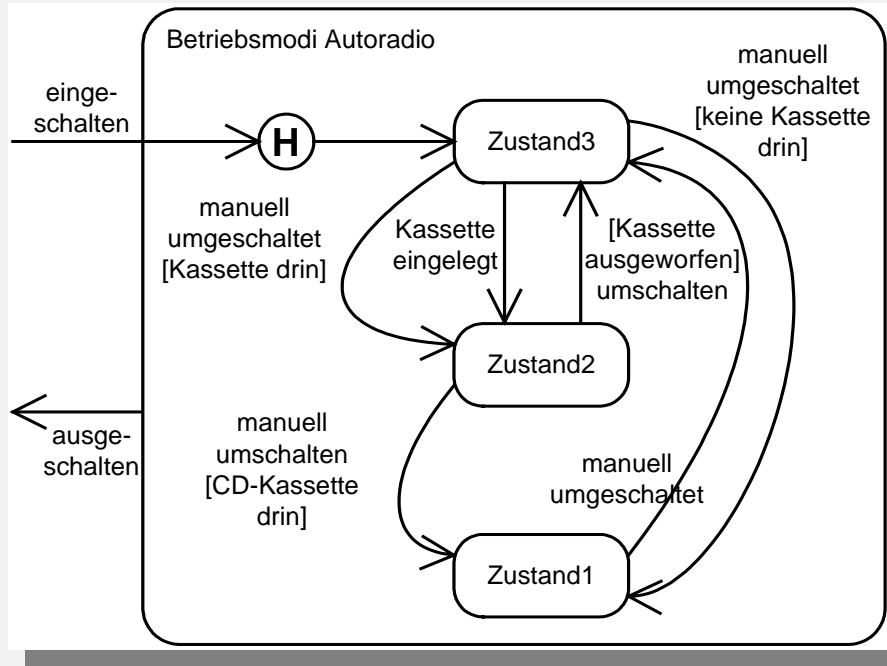
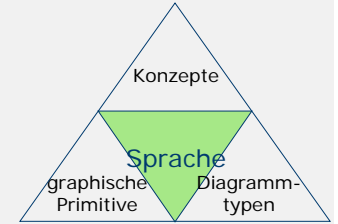


tiefe Historie

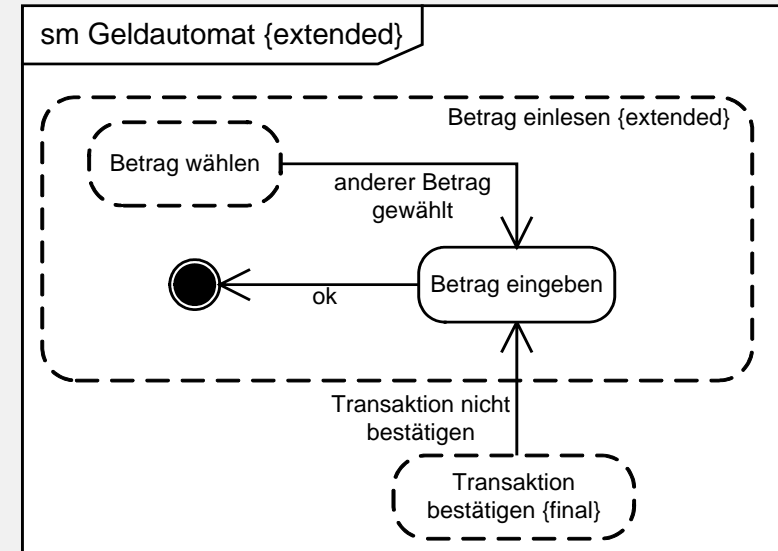
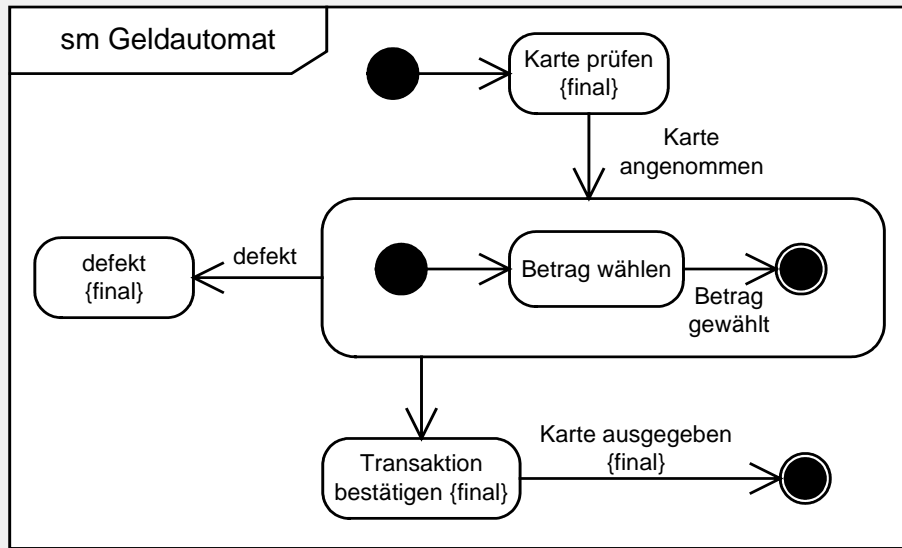


Terminator

# Zustandsdiagramm

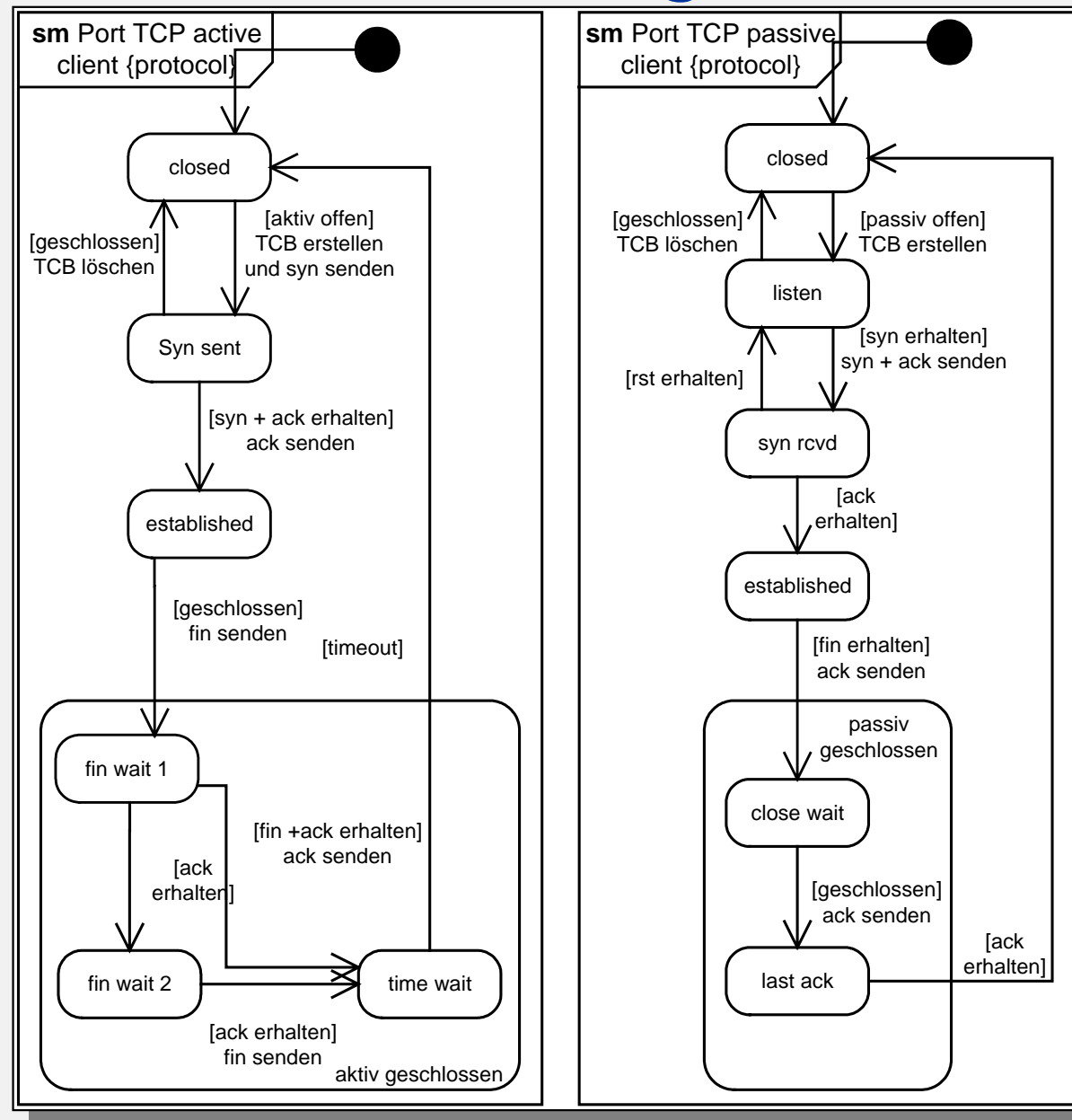
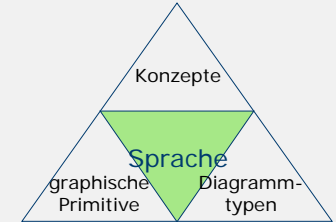


# Zustandsdiagramm



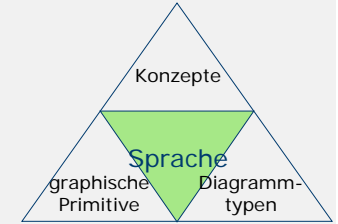
- Spezialisierung von Zustandsautomaten durch
  - Erweiterung um Regionen, Zustände und Transitionen
  - Erweiterung von Regionen und Zuständen
  - Erweiterung von Transitionen

# Zustandsdiagramm



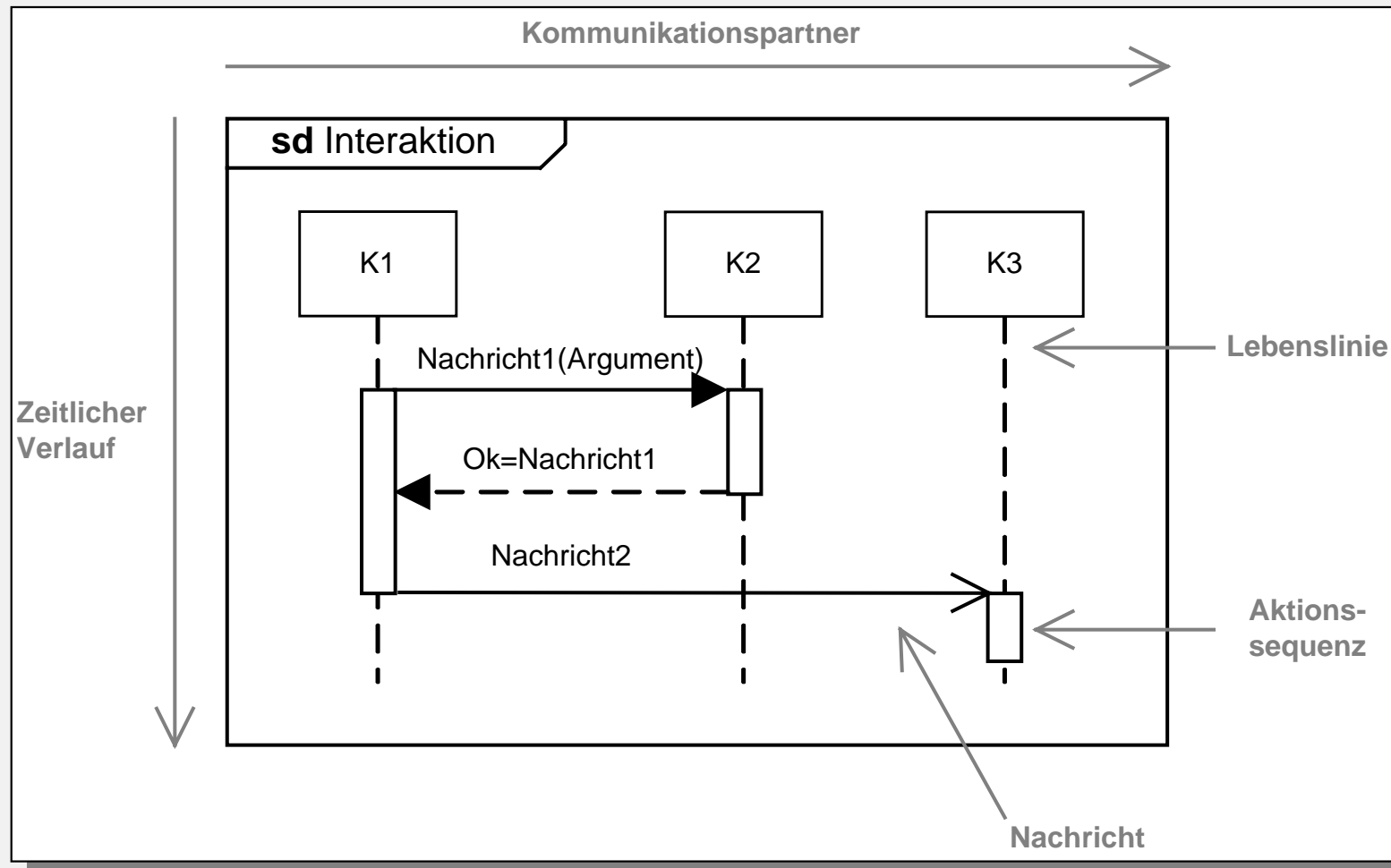
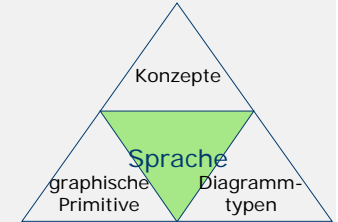
- Protokollzustandsautomat

# Sequenzdiagramm

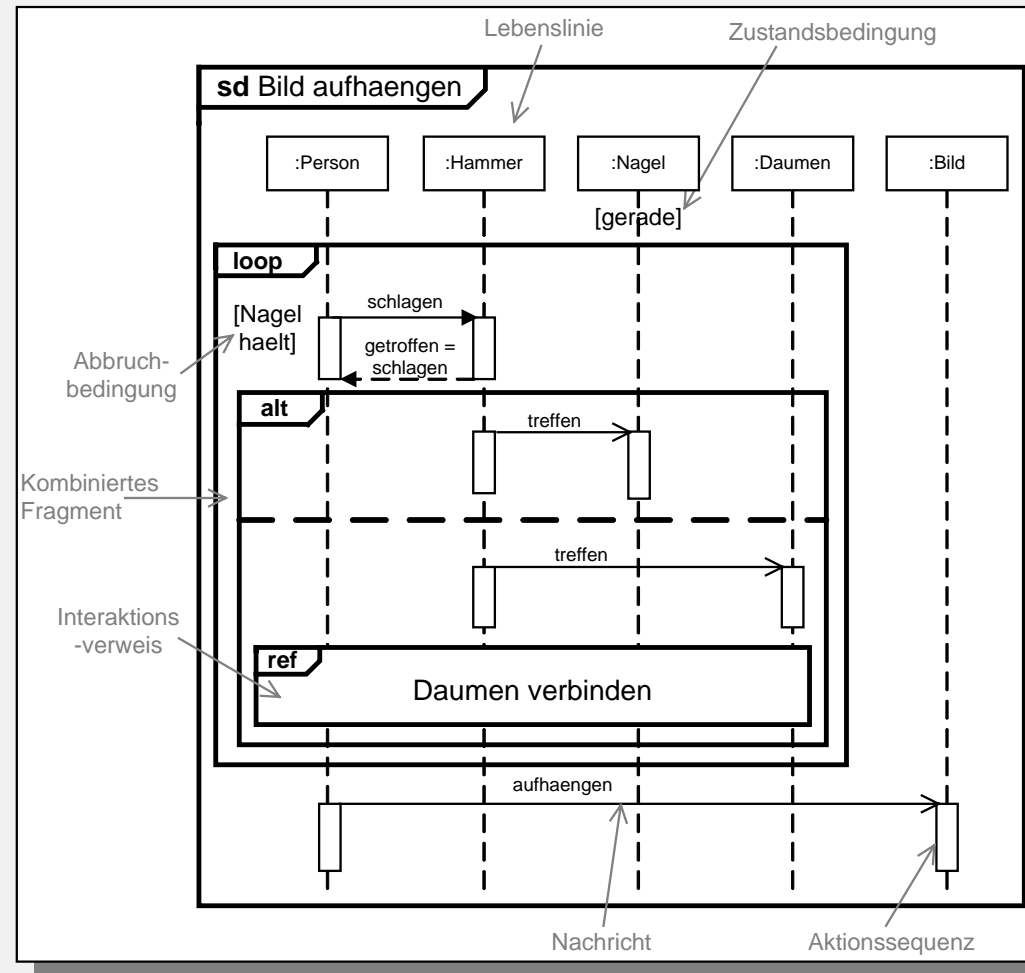
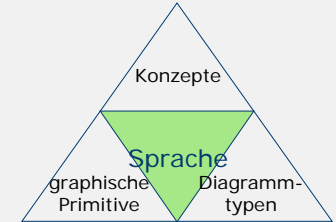


- **Aufgabe:**
  - Beschreibt den Intra- und Intersystem-Datenaustausch
- **Aussage:**
  - Wie spielen die einzelnen Systeme oder –komponenten zusammen
- **Aufgabe im Projekt:**
  - Darstellung der dynamischen Ausrufbeziehungen
- **Änderungen durch UML 2:**
  - Erweiterung der möglichen Kommunikationspartner
  - Referenzierung und Hierarchisierung möglich
  - Kontrollflüsse ausdrückbar
  - Neue Elemente
    - Interaktionsrahmen
    - Kombinierte Fragmente
    - Sprungmarken und Coregionen
    - Interaktionsreferenzen
    - *Gates* für Nachrichten

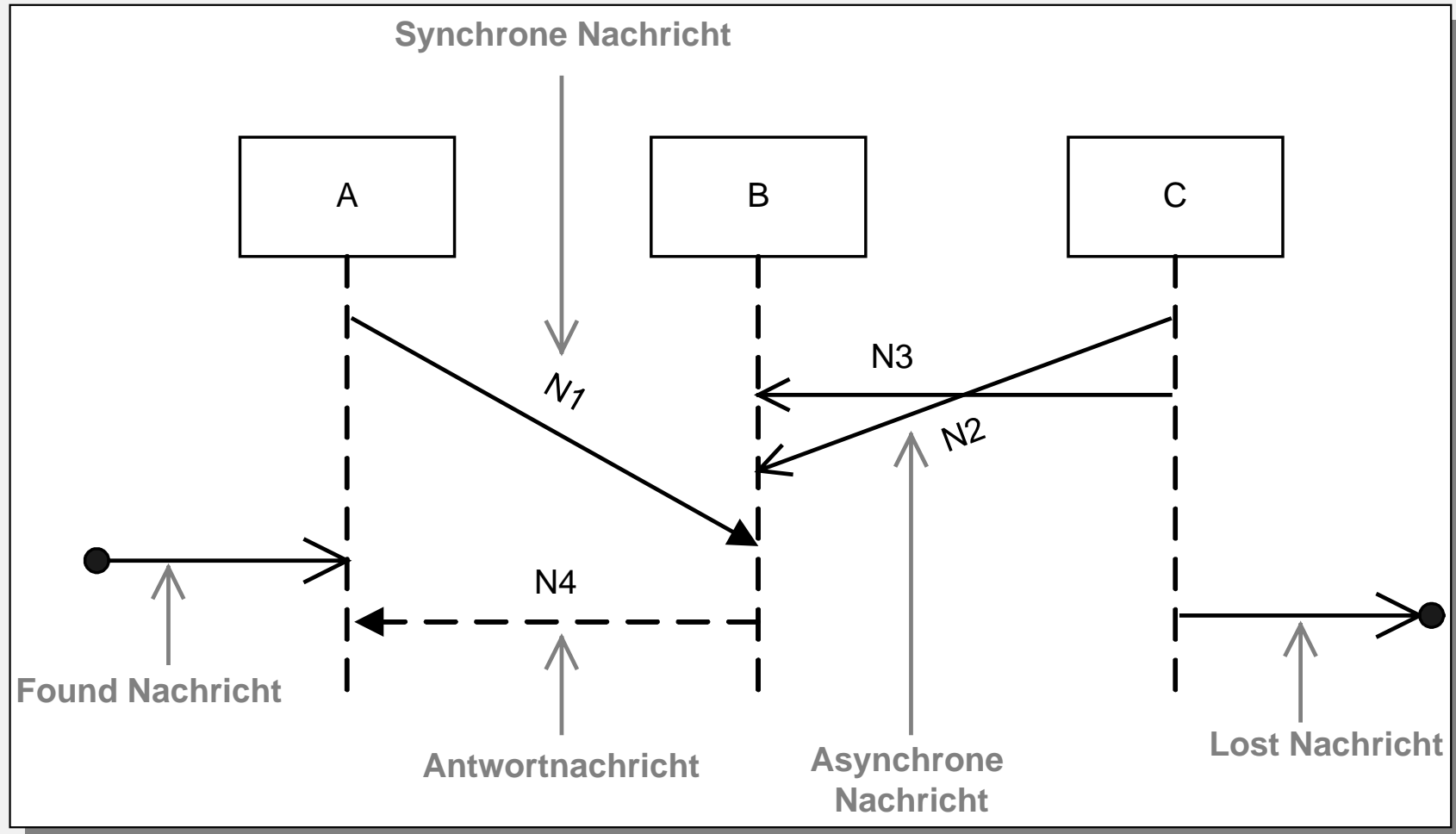
# Sequenzdiagramm



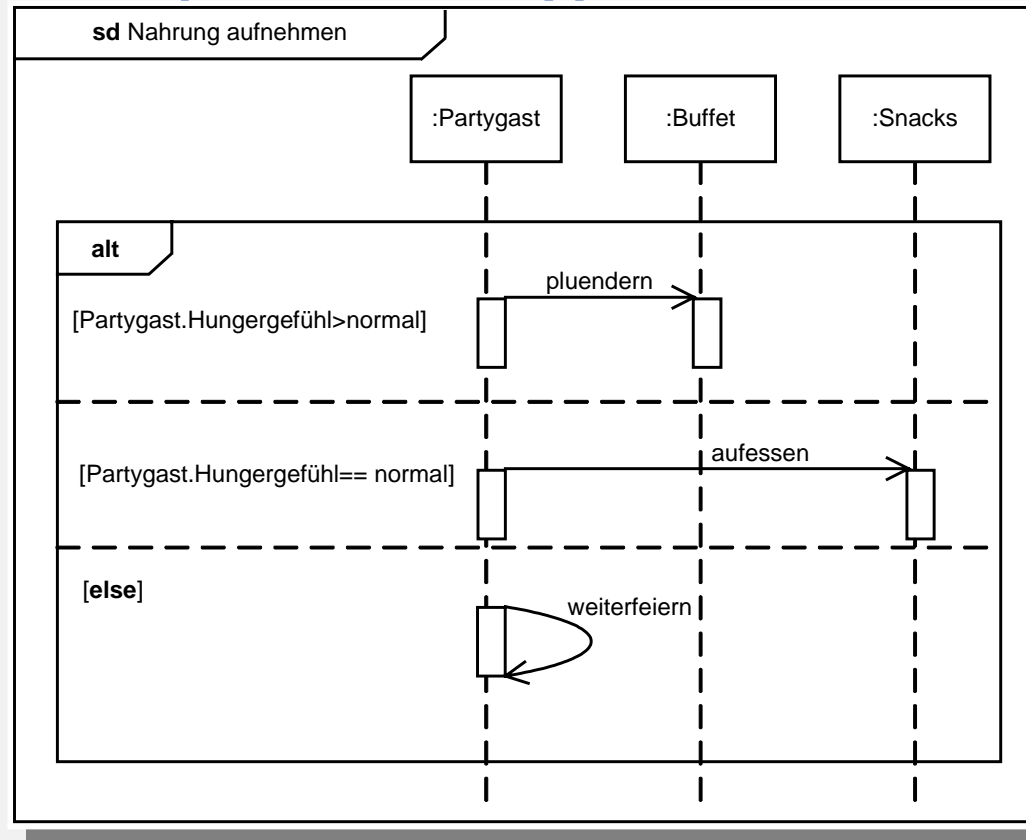
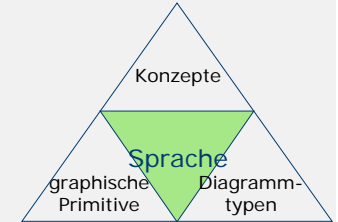
# Sequenzdiagramm



# Sequenzdiagramm

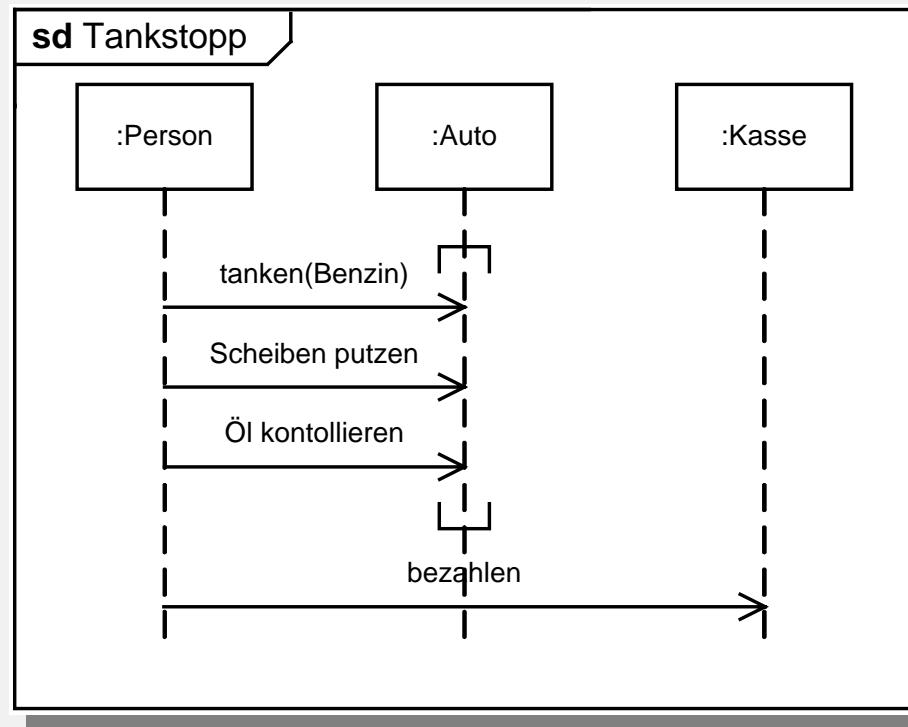
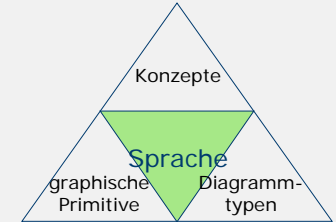


# Sequenzdiagramm



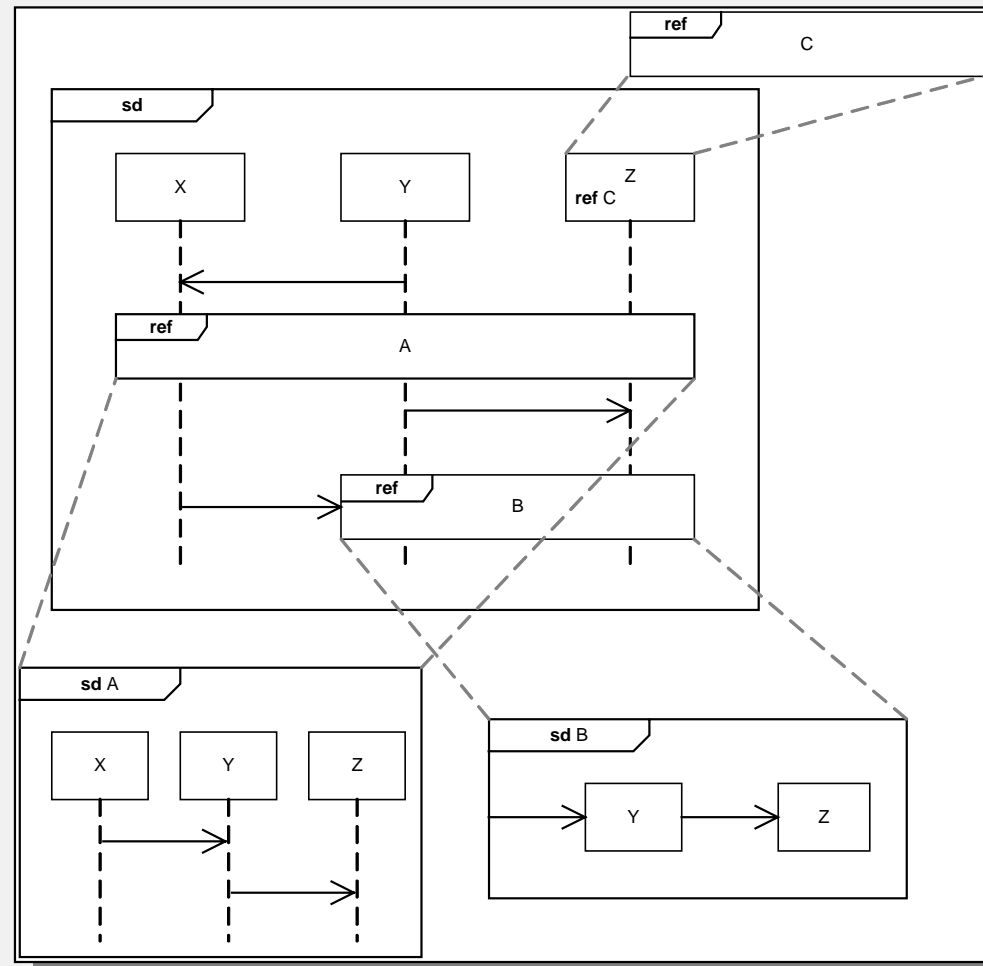
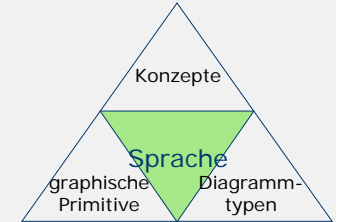
- alt: Bedingungsgesteuerte Alternativen (min. zwei)
- ignore: Gezielte Unterspezifikation (d.h. Realitätsausschnitt fehlt)
- consider: Betonung der Bedeutung
- opt: Optionale Ausführung
- loop: Zählschleife
- neg: Nicht zugelassener Ablauf
- assert: Zusicherung, die gelten muß
- par: Nebenläufigkeit oder Parallelität (wird nicht unterschieden)
- critical: Ununterbrechbarer kritischer Abschnitt

# Sequenzdiagramm



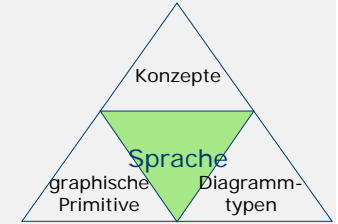
- Coregion:
  - Alternativdarstellung zum parallel kombinierten Fragment
  - Nur zugelassen wenn genau eine Lebenslinie betroffen ist
  - Ablaufreihenfolge innerhalb der Coregion nicht festgelegt

# Sequenzdiagramm



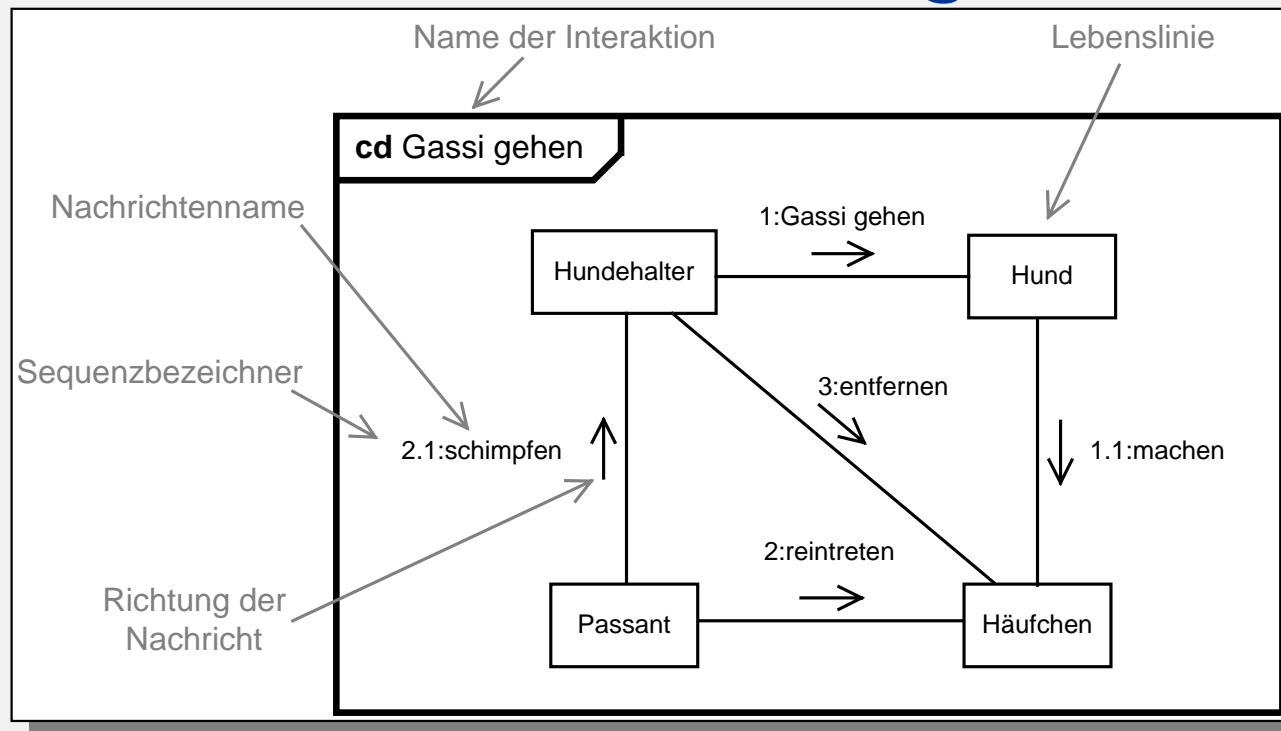
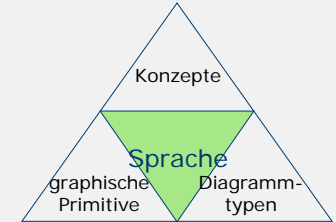
- Referenziert (`ref`) auf eine beliebige Interaktion
- Wiederverwendung in mehreren Diagrammen möglich
- „Zooming“-Gedanke
- Auch für Lebenslinien möglich

# Kommunikationsdiagramm



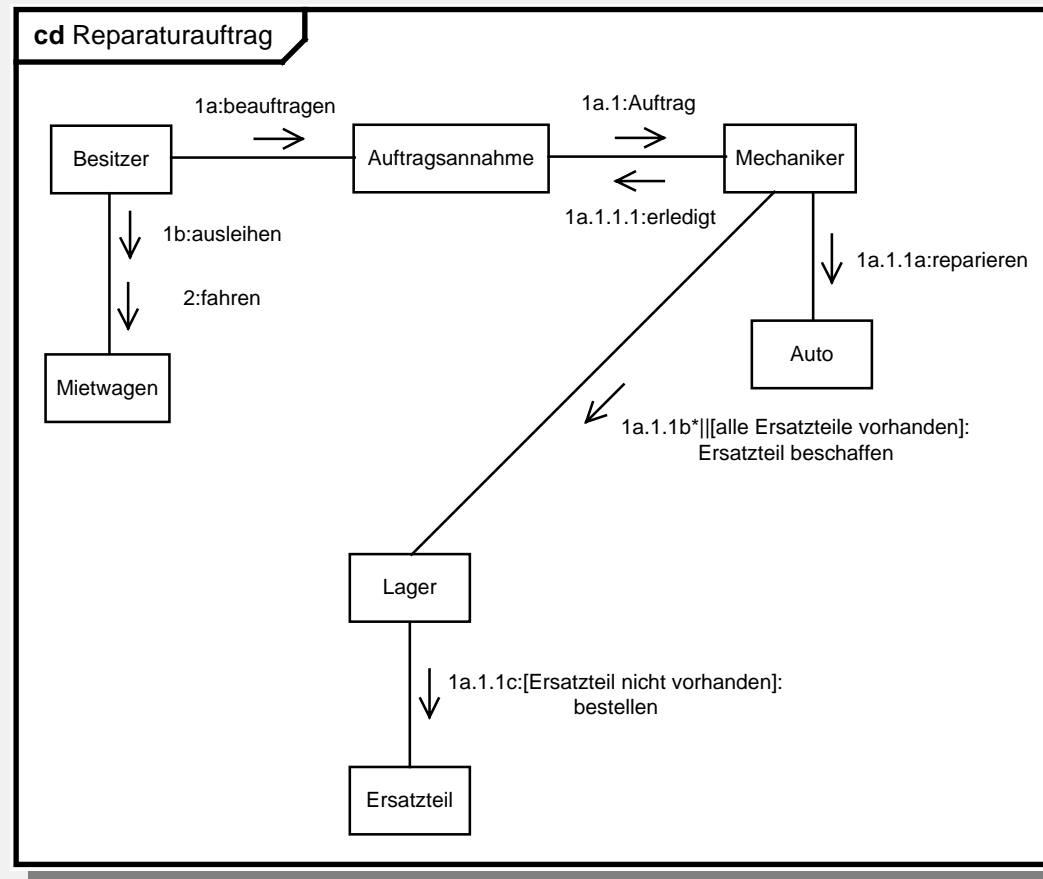
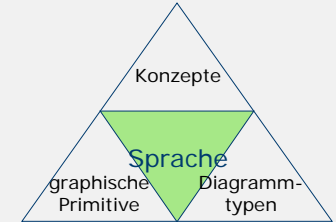
- **Aufgabe:**
  - Statische Sicht auf dynamische Interaktion
- **Aussage:**
  - Stellt Teile einer komplexen Struktur und ihre Beziehungen in der Zusammenschau dar
- **Aufgabe im Projekt:**
  - Dokumentation aller ausgetauschten Nachrichten
- **Änderungen durch UML 2:**
  - Diagrammtyp neu eingeführt (entspricht inhaltlich und konzeptionell dem *Kollaborationsdiagramm*)
  - Untermenge des Sequenzdiagramms
    - Keine Verweise
    - Keine kombinierten Fragmente
    - Keine Berücksichtigung der Ereignisreihenfolge

# Kommunikationsdiagramm



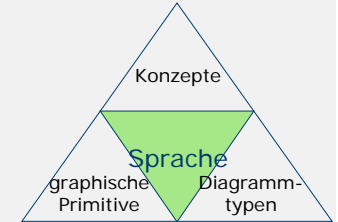
- Notationselemente:
  - Interaktion
  - Lebenslinien
  - Nachrichten
  - Sequenzbezeichner

# Kommunikationsdiagramm



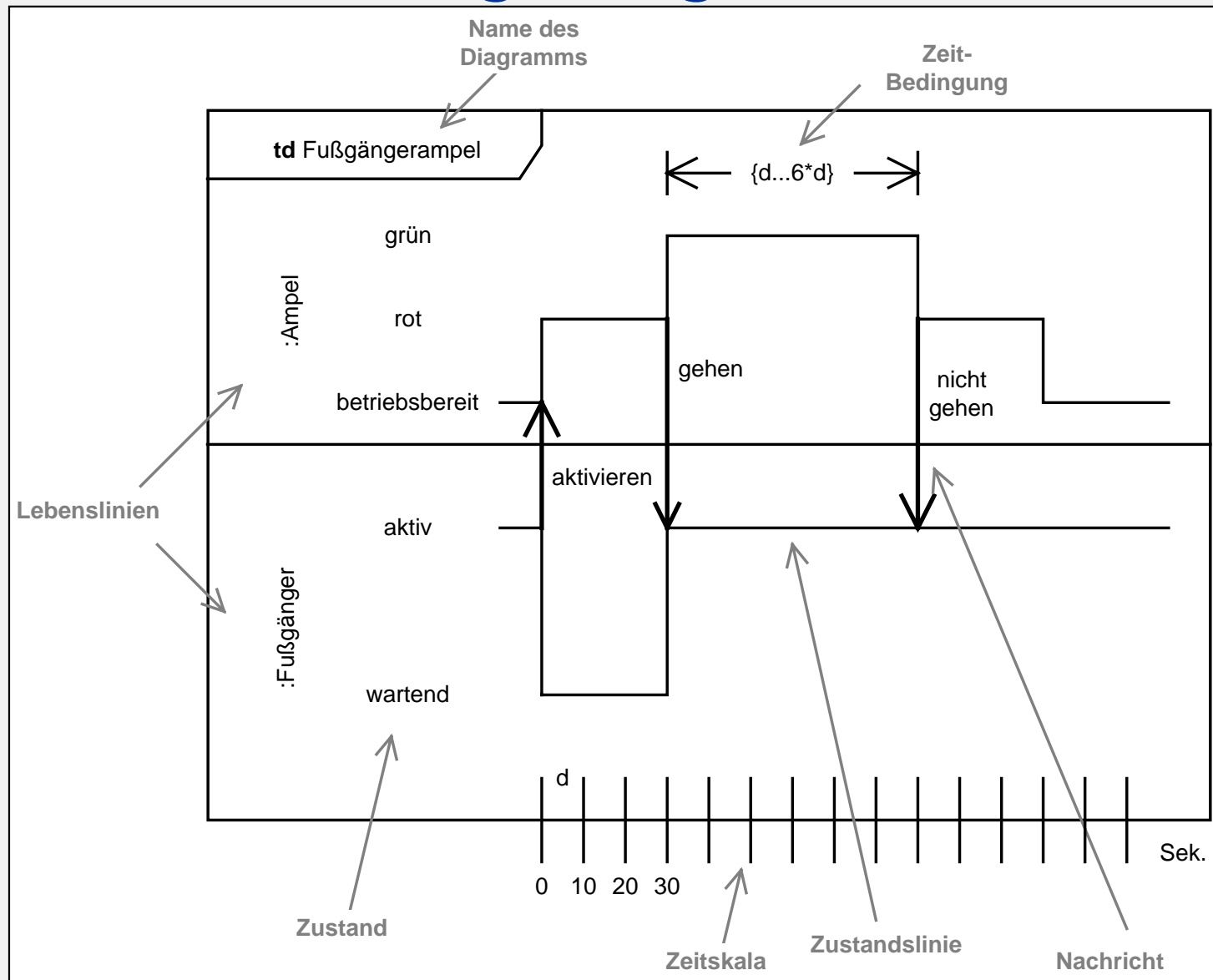
- Notation etwas unübersichtlich:
  - Nebenläufigkeit dokumentiert durch Buchstaben im Sequenzbezeichner
  - Definition von Schleifen mit einem Stern „\*“
  - Kennzeichnung von nebenläufigen Schleifen-durchläufen mit Doppelstrich „||“

# Timing-Diagramm

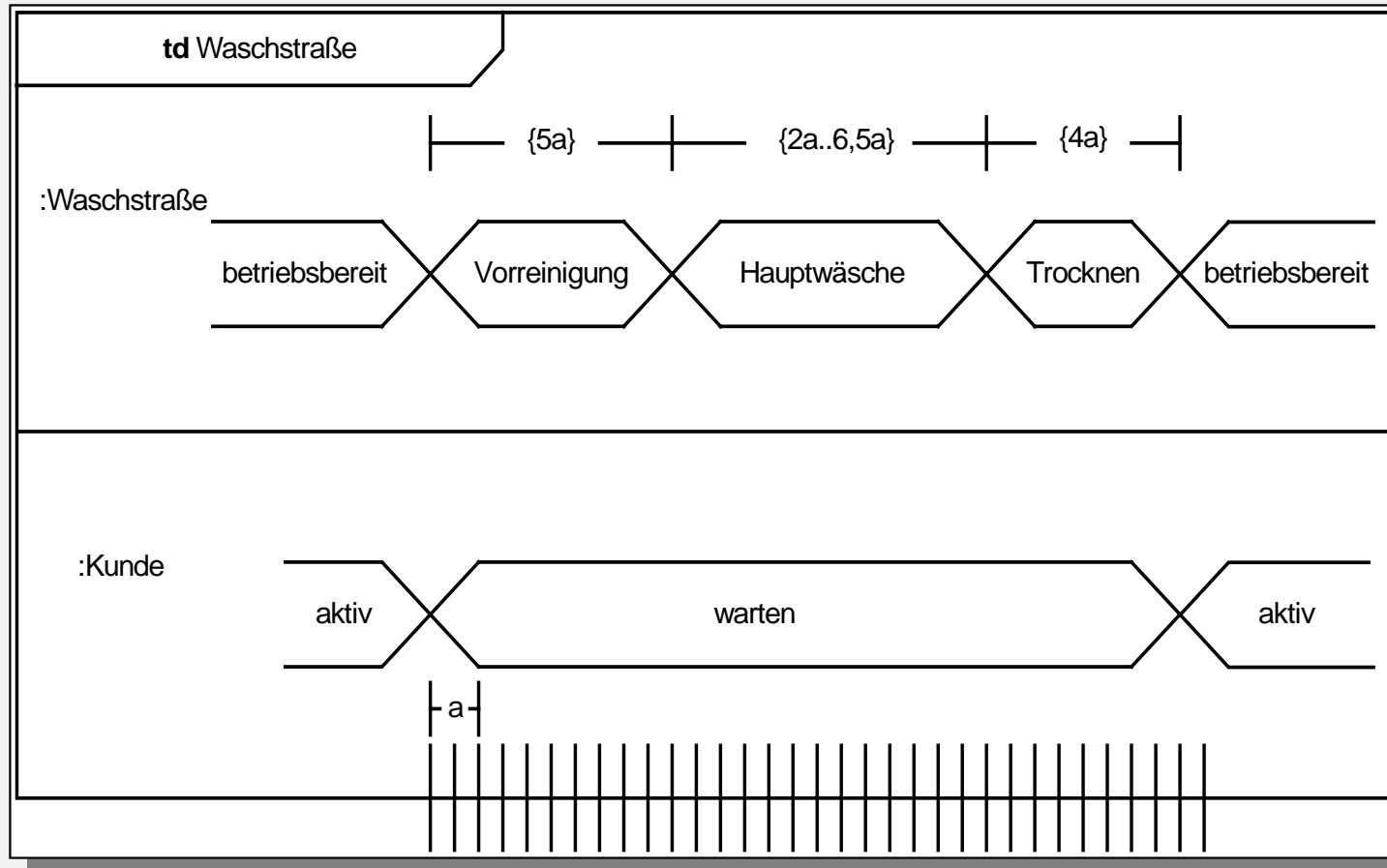
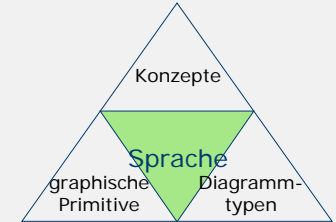


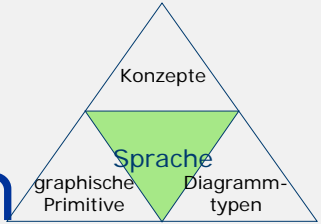
- **Aufgabe:**
  - Zeitabhängige Zustandsdarstellung
- **Aussage:**
  - Dokumentation des Zeitpunktes eines Zustandswechsels eines Kommunikationspartner
- **Aufgabe im Projekt:**
  - Dokumentation des zeitlichen (System-)Verhaltens analog einer Schaltung
- **Änderungen durch UML 2:**
  - Diagrammtyp neu eingeführt

# Timing-Diagramm



# Timing-Diagramm

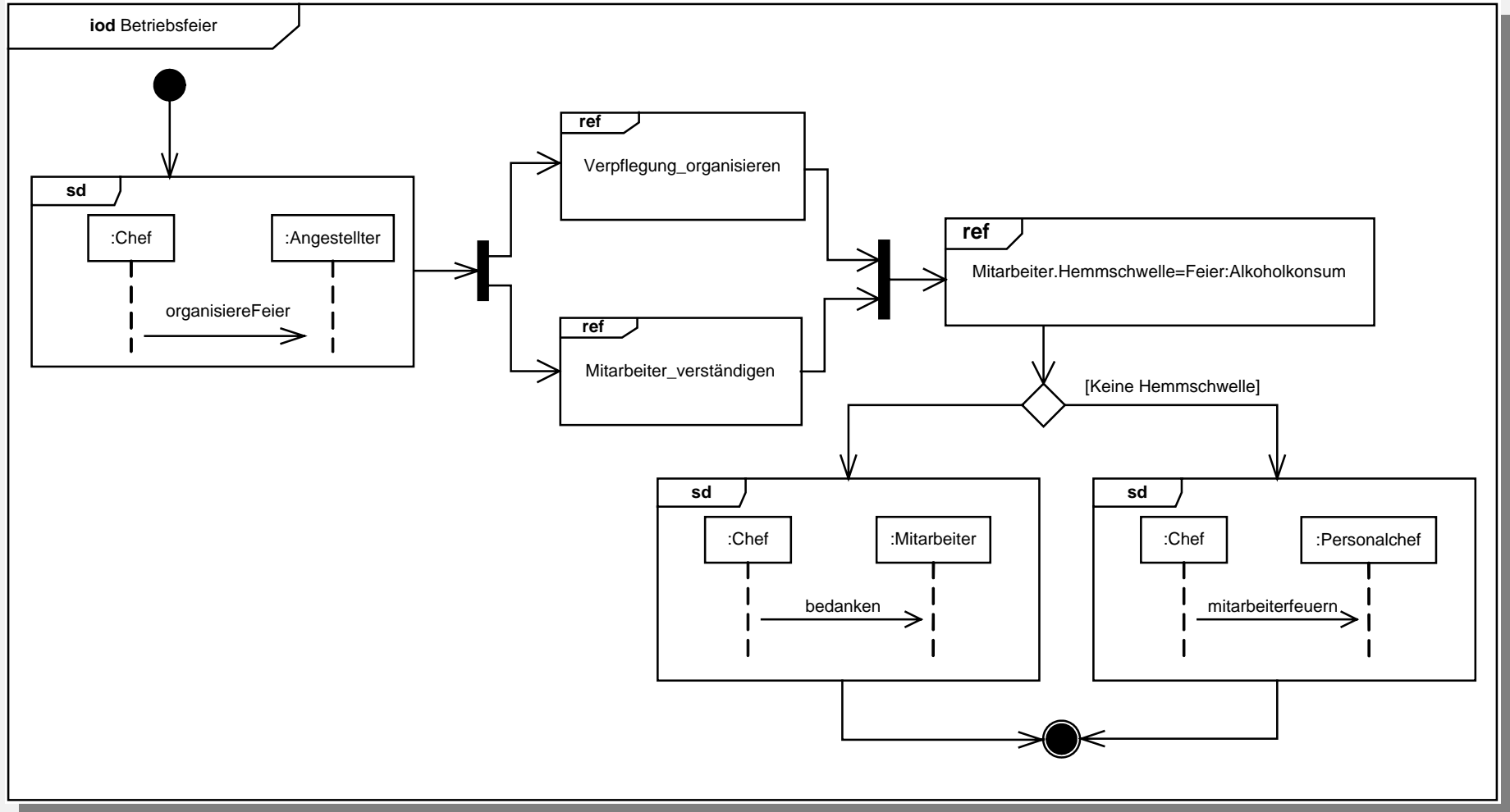




# Interaktionsübersichts-Diagramm

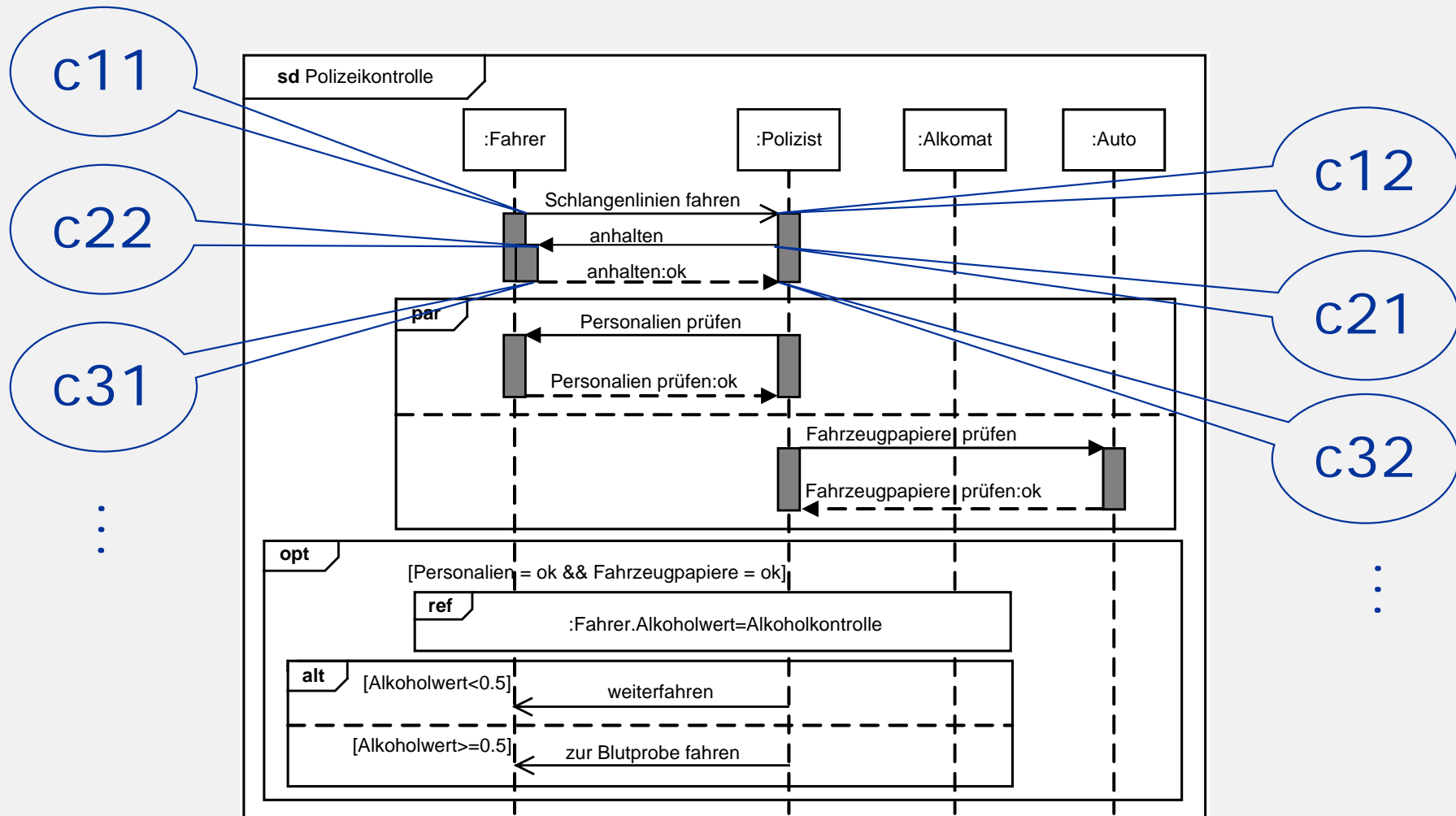
- **Aufgabe:**
  - Darstellung des Zusammenspiels verschiedener Interaktionen
- **Aussage:**
  - Dokumentation der Bedingungen und Reihenfolgen der Interaktionsausführungen
- **Aufgabe im Projekt:**
  - Zusammenhang zwischen Aktivitätsdiagrammen
  - Übersichtlichkeitserhalt oder -gewinn
- **Änderungen durch UML 2:**
  - Diagrammtyp neu eingeführt

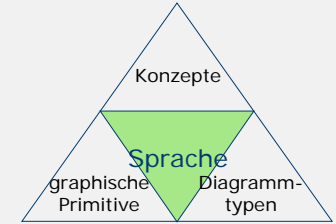
# Interaktionsübersichts-Diagramm



# Nicht-graphische Repräsentationen

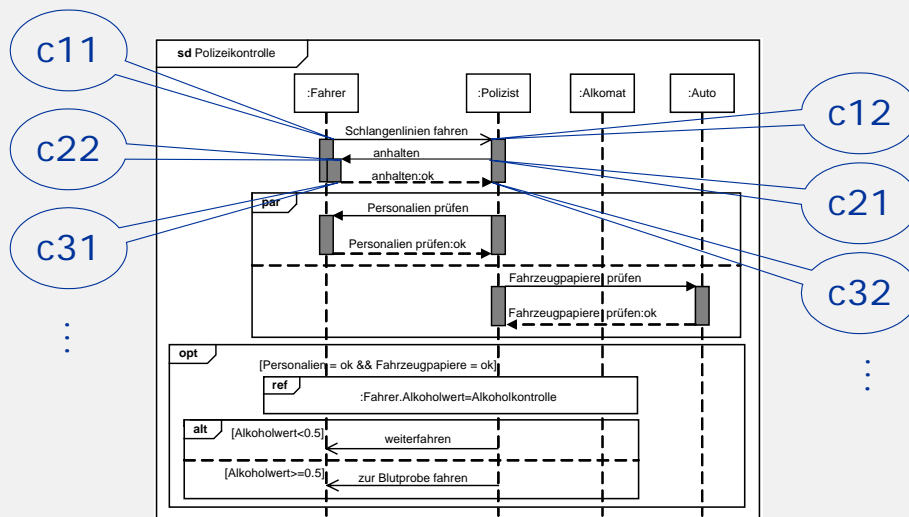
- Als Alternative zu der graphischen Darstellung von Sequenzdiagrammen stehen tabellenartige Texte zur Verfügung





# Nicht-graphische Repräsentationen

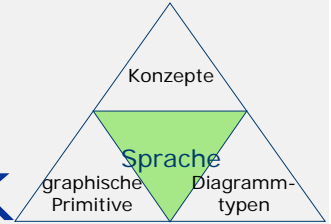
- Als Alternative zu der graphischen Darstellung von Sequenzdiagrammen stehen tabellenartige Texte zur Verfügung



## Optionale tabellarische Darstellung

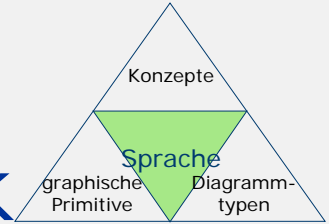
- Maschinell verarbeitbar
- Möglichkeit der Konsistenzprüfung
- Übersichtlichkeit bei großen Diagrammmengen

Lifeline Instance	Sending msg. Instance	Order	Msg. Name	Msg. Receive Instance
Fahrer		c11	Schlangenlinien fahren	Polizist
Polizist	Fahrer	c12	Schlangenlinien fahren	
Polizist		c21	anhalten	Fahrer
Fahrer	Polizist	c22	anhalten	
Fahrer		c31	anhalten:ok	Polizist
	Fahrer	c32	anhalten:ok	

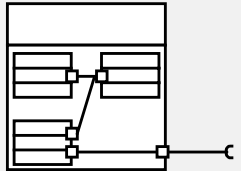
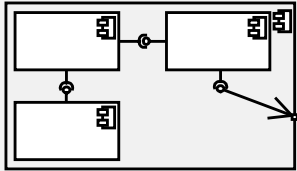
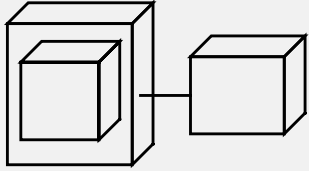


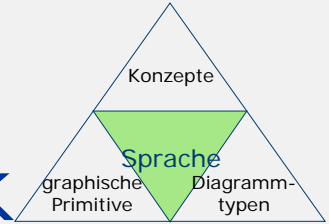
# Die Diagrammtypen im Überblick

Diagrammtyp	Diese zentrale Frage beantwortet das Diagramm	Stärken
Klassendiagramm 	Aus welchen Klassen besteht mein System und wie stehen diese untereinander in Beziehung?	Beschreibt die statische Struktur des Systems. Enthält alle relevanten Strukturzusammenhänge/Datentypen. Brücke zu dynamischen Diagrammen. Normalerweise unverzichtbar.
Paketdiagramm 	Wie kann ich mein Modell so schneiden, dass ich den Überblick bewahre?	Logische Zusammenfassung von Modellelementen. Modellierung von Abhängigkeiten/Inklusion möglich.
Objektdiagramm 	Welche innere Struktur besitzt mein System zu einem bestimmten Zeitpunkt zur Laufzeit (Klassendiagrammschnappschuss)?	Zeigt Objekte u. Attributbelegungen zu einem bestimmten Zeitpunkt. Verwendung beispielhaft zur Veranschaulichung Detailniveau wie im Klassendiagramm. Sehr gute Darstellung von Mengenverhältnissen.



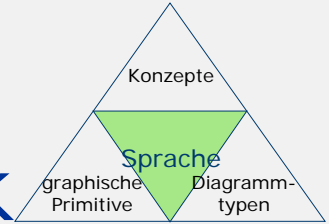
# Die Diagrammtypen im Überblick

Diagrammtyp	Diese zentrale Frage beantwortet das Diagramm	Stärken
Kompositionsstrukturdiagramm 	Wie sieht das Innenleben einer Klasse, einer Komponente, eines Systemteils aus?	Ideal für die Top-Down-Modellierung des Systems (Ganz-Teil-Hierarchien). Zeigt Teile eines „Gesamtelements“ und deren Mengenverhältnisse. Präzise Modellierung der Teile-Beziehungen über spezielle Schnittstellen (Ports) möglich.
Komponentendiagramm 	Wie werden meine Klassen zu wieder verwendbaren, verwaltbaren Komponenten zusammengefasst und wie stehen diese in Beziehung?	Zeigt Organisation und Abhängigkeiten einzelner technischer Systemkomponenten. Modellierung angebotener und benötigter Schnittstellen möglich.
Verteilungsdiagramm 	Wie sieht das Einsatzumfeld (Hardware, Server, Datenbanken, ...) des Systems aus? Wie werden die Komponenten zur Laufzeit wohin verteilt?	Zeigt das Laufzeitumfeld des Systems mit den „greifbaren“ Systemteilen. Darstellung von „Softwareservern“ möglich. Hohes Abstraktionsniveau, kaum Notationselemente.



# Die Diagrammtypen im Überblick

Diagrammtyp	Diese zentrale Frage beantwortet das Diagramm	Stärken
Use-Case-Diagramm 	Was leistet mein System für seine Umwelt (Nachbarsysteme, Stakeholder)?	Außensicht auf das System. Geeignet zur Kontextabgrenzung. Hohes Abstraktionsniveau, einfache Notationsmittel.
Aktivitätsdiagramm 	Wie läuft ein bestimmter flussorientierter Prozess oder ein Algorithmus ab?	Sehr detaillierte Visualisierung von Abläufen mit Bedingungen, Schleifen, Verzweigungen. Parallelisierung und Synchronisation. Darstellung von Datenflüssen.
Zustandsautomat 	Welche Zustände kann ein Objekt, eine Schnittstelle, ein Use Case, ... bei welchen Ereignissen annehmen?	Präzise Abbildung eines Zustandsmodells mit Zuständen, Ereignissen, Nebenläufigkeiten, Bedingungen, Ein- und Austrittsaktionen. Schachtelung möglich.
Sequenzdiagramm 	Wer tauscht mit wem welche Informationen in welcher Reihenfolge aus?	Darstellung des Informationsaustauschs zwischen Kommunikationspartnern Sehr präzise Darstellung der zeitlichen Abfolge auch mit Nebenläufigkeiten.



# Die Diagrammtypen im Überblick

Diagrammtyp	Diese zentrale Frage beantwortet das Diagramm	Stärken
Kommunikationsdiagramm 	Wer kommuniziert mit wem? Wer „arbeitet“ im System zusammen?	Stellt den Informationsaustausch zwischen Kommunikationspartnern dar. Überblick steht im Vordergrund (Details und zeitliche Abfolge weniger wichtig).
Timingdiagramm 	Wann befinden sich verschiedene Interaktionspartner in welchem Zustand?	Visualisiert das exakte zeitliche Verhalten von Klassen, Schnittstellen, ... Geeignet für die Detailbetrachtungen, bei denen es wichtig ist, dass ein Ereignis zum richtigen Zeitpunkt eintritt.
Interaktionsübersichtsdiagramm 	Wann läuft welche Interaktion ab?	Verbindet Interaktionsdiagramme (Sequenz-, Kommunikations- und Timingdiagramme) auf Top-Level-Ebene. Hohes Abstraktionsniveau.