



Universität Bremen

Unified Modeling Language

Thomas Röfer

Motivation

Entwicklung

Spracheinheiten

Diagramme (Struktur-/Verhaltensdiagramme)



Rückblick „Textsuche“

Naive Suche

abrakadabra...
aber
aber
aber
aber
aber
aber
aber

Boyer-Moore

abrakadabra...
 aber
 aber
 aber
 a...

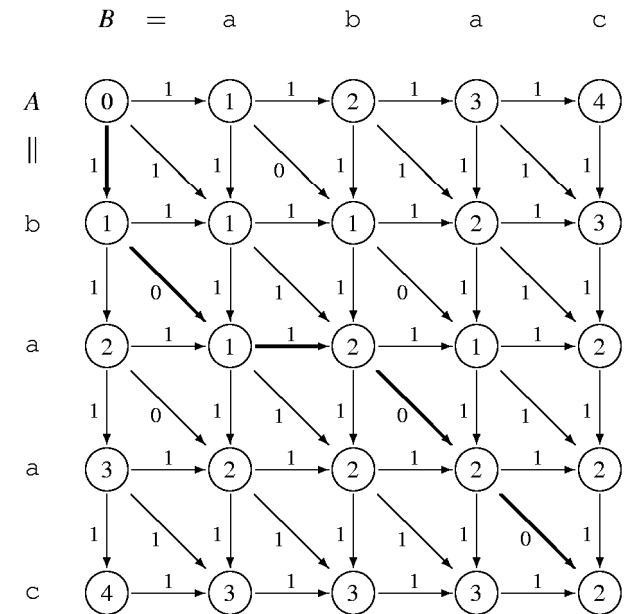
Knuth-Morris-Pratt

abrakadabra...
aber
aber
aber
aber
aber
aber

Ähnlichkeitssuche

abrakadabra...
 aber
 aber
 aber
 aber
 aber

Editierdistanz



Motivation

- ▶ **Anforderungen an Software**

- ▶ nützlich und nutzbar
- ▶ zuverlässig
- ▶ flexibel
- ▶ kostengünstig
- ▶ verfügbar

- ▶ **Gegenbeispiel: Ariane 5**

- ▶ Fehler

```
pragma suppress(numeric_error, horizontal_veloc_bias);  
...  
horizontal_veloc_bias := integer(horizontal_veloc_sensor);
```

- ▶ Überlauf, konnte bei Ariane 4 beweisbar noch nicht auftreten
- ▶ Verlust
 - ▶ *DM 250 Millionen Startkosten*
 - ▶ *DM 850 Millionen Cluster-Satelliten*
 - ▶ *DM 600 Millionen für nachfolgende Verbesserungen*
 - ▶ *Verdienstausfall für 2 bis 3 Jahre*



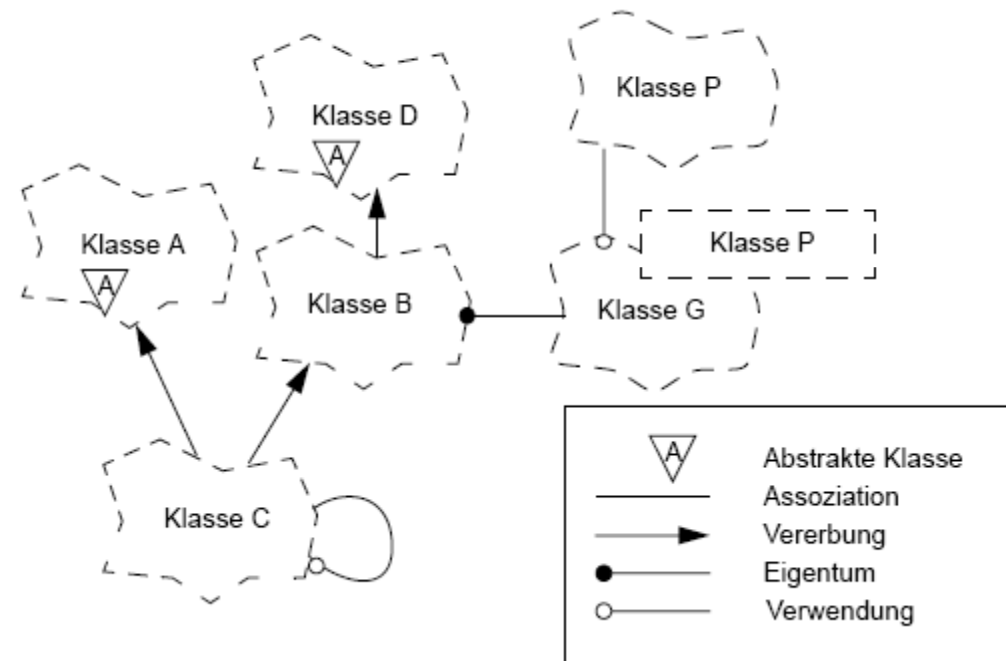
Motivation

▶ Ein Modell

- ▶ ist eine abstrakte Repräsentation eines Systems
- ▶ ist eine Vereinfachung der Realität
- ▶ stellt die Kernpunkte eines Systems dar

▶ Abstraktion

- ▶ bedeutet nicht kompliziert, sondern auf das Wesentliche reduziert
- ▶ Dadurch verhindern wir, uns beim Entwurf eines Systems in Details zu verlieren



Booch-Notation, 1991



Motivation

▶ Warum modellieren?

- ▶ Die meisten Systeme sind für Heldenprogrammierung zu groß.
- ▶ Teamarbeit → Kommunikation ist notwendig
- ▶ Nicht nur Programmierer, auch Manager, Vertrieb und Kunden sind in den Entwicklungsprozess involviert
→ gemeinsame, verständliche Sprache ist notwendig
- ▶ Das Rad muss nicht jedesmal neu erfunden werden → Dokumentation

▶ Beschreibung des Systems

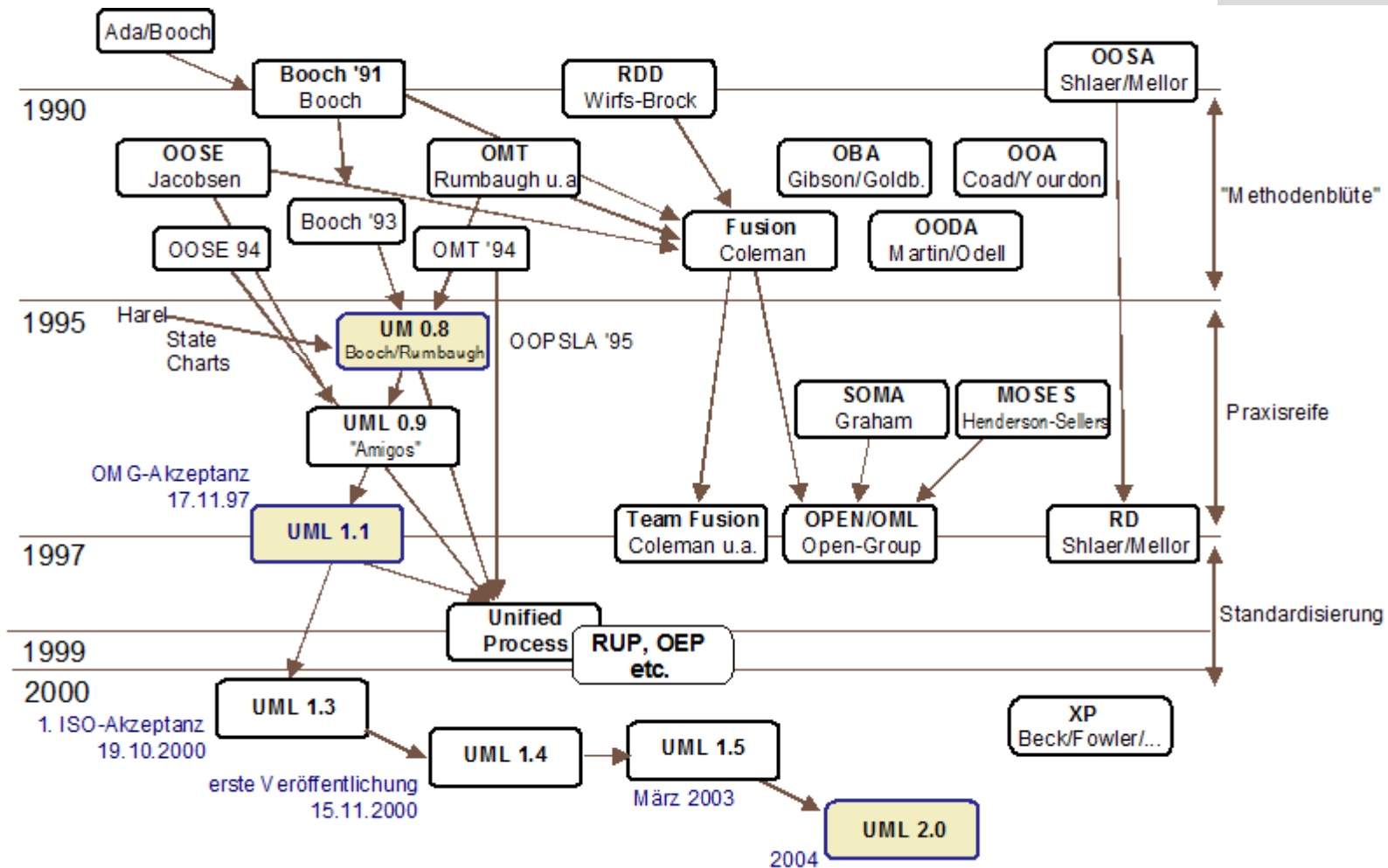
- ▶ Natürliche Sprache ist mehrdeutig und unpräzise
- ▶ Programmiersprachen sind zu präzise und nicht für alle (z.B. Manager) verständlich
- ▶ Graphische Modellierung mit UML füllt diese Lücke aus

▶ Literatur

- ▶ Christoph Kecher (2006). UML 2.0 - Das umfassende Handbuch. Galileo Computing.
- ▶ Wikipedia, Begriff „UML“



Historie





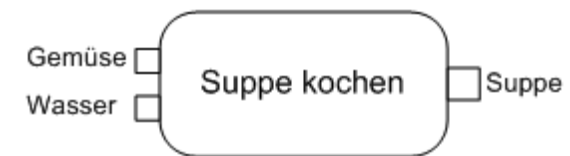
Unified Modeling Language

- ▶ **Unified Modeling Language (UML)**
 - ▶ Derzeit in Version 2.0
 - ▶ Seit 1995, definiert von Grady Booch, Jim Rumbaugh und Ivar Jacobson
 - ▶ Mittlerweile de facto Standard
- ▶ **Eigenschaften**
 - ▶ Graphisch orientierte Modellierungssprache
 - ▶ Objektorientiert
 - ▶ Unabhängig von Entwicklungsprozessen und -methoden
- ▶ **Verschiedene Aspekte von Softwaresystemen widerzuspiegeln durch**
 - ▶ Strukturdiagramme
 - ▶ Verhaltensdiagramme
 - ▶ Spracheinheiten (Grundbausteine der Diagramme)

Spracheinheiten

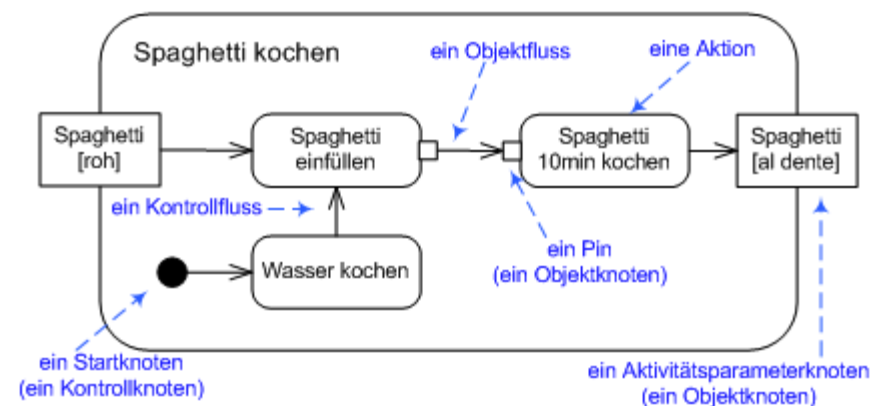
▶ Aktionen

- ▶ Elementare Bausteine für die Modellierung eines Verhaltens
- ▶ Eingabewerte über *Eingabepins*
- ▶ Ausgabewerte über *Ausgabepins*



▶ Aktivitäten

- ▶ Modelle für Verhalten
 - ▶ *Beschrieben als Menge von elementaren Aktionen zwischen denen Kontroll- und Datenflüsse existieren*
- ▶ Graphstruktur
 - ▶ *Knoten: Objekt- und Kontrollknoten*
 - ▶ *Kanten: Objekt- und Kontrollflüsse*
- ▶ Können verschachtelt und mit Kontrollstrukturen modularisiert werden



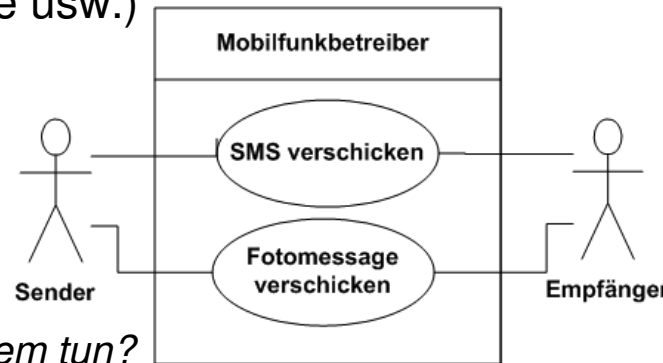
Spracheinheiten

▶ Allgemeines Verhalten

- ▶ Fasst Modellelemente zusammen, die für Aktivitäten, Interaktionen oder Zustandsautomaten benötigt werden
- ▶ Verhalten startet aufgrund von diskreten Ereignissen
- ▶ Behandlung von Zeit (Dauer, Intervalle usw.)

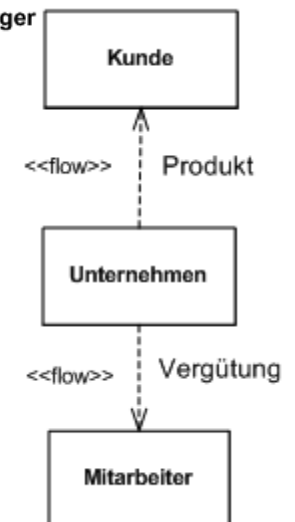
▶ Anwendungsfälle

- ▶ Modellierung von Anforderungen an ein System
- ▶ Was soll ein System tun?
- ▶ Akteure
 - ▶ *Wer (Person) soll etwas mit dem System tun?*
 - ▶ *Was (anderes System) soll etwas mit dem System tun?*



▶ Informationsflüsse

- ▶ Informationseinheit, Informationsfluss
- ▶ Verbinden Klassen, Anwendungsfälle, Ausprägungsspezifikationen, Akteure, Schnittstellen, Ports usw.



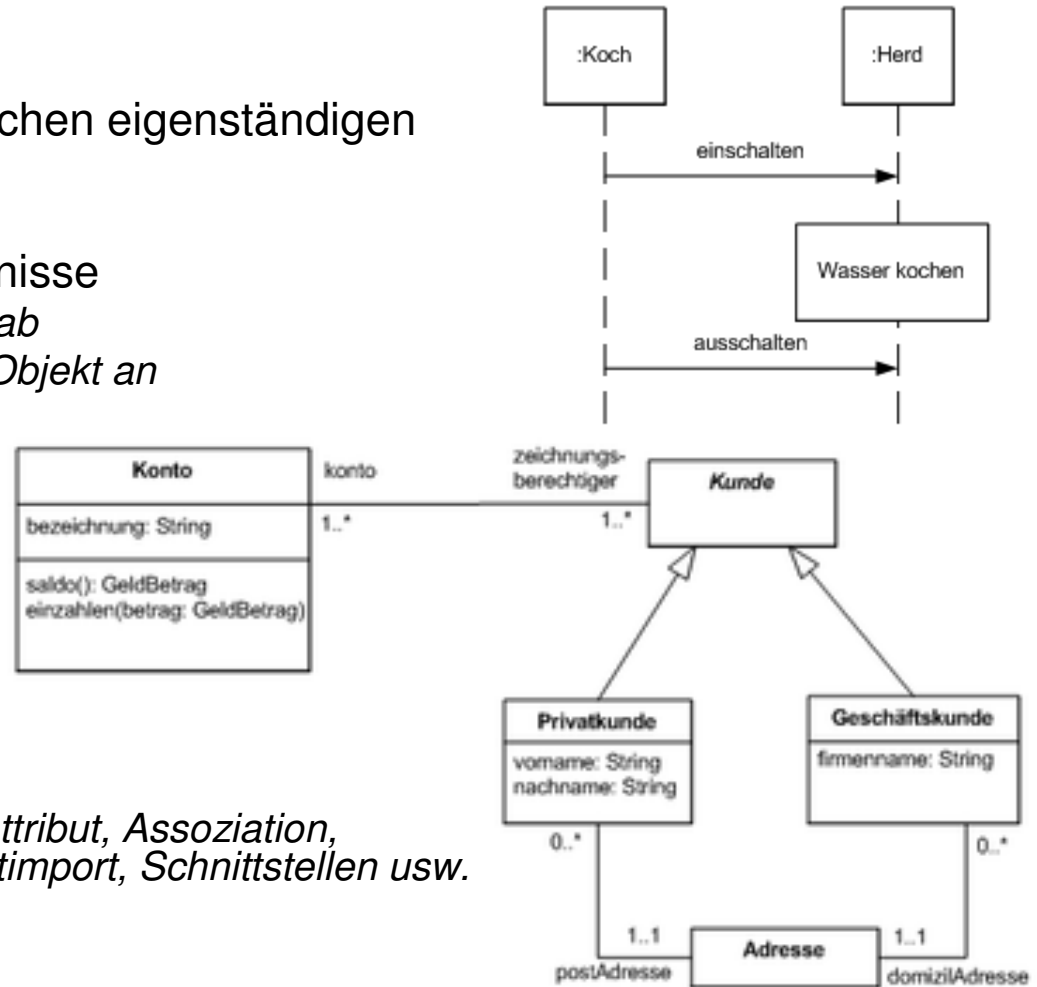
Spracheinheiten

▶ Interaktionen

- ▶ Austausch von Meldungen zwischen eigenständigen Objekten
- ▶ Lebenslinien
- ▶ Meldung versenden: zwei Ereignisse
 - ▶ *Erstes Objekt schickt Meldung ab*
 - ▶ *Meldung kommt beim zweiten Objekt an*
- ▶ Spur
 - ▶ *Folge von Ereignissen*
- ▶ Interaktionen
 - ▶ *Zwei Mengen von Spuren (gültige, ungültige)*

▶ Klassen

- ▶ Kern von UML
- ▶ Elemente
 - ▶ *Klasse, Namensraum, Paket, Attribut, Assoziation, Abhängigkeitsbeziehung, Paketimport, Schnittstellen usw.*



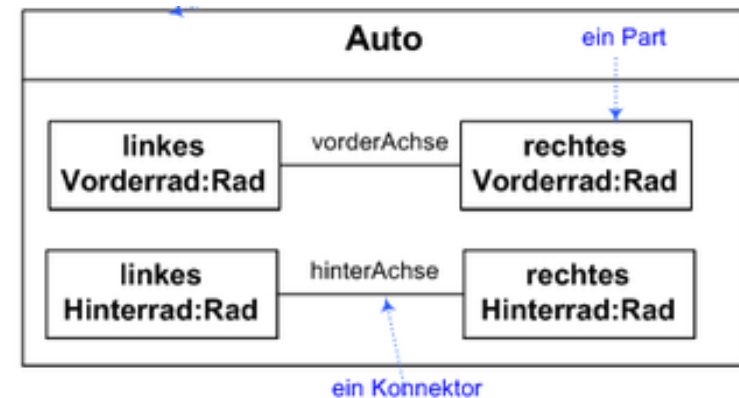
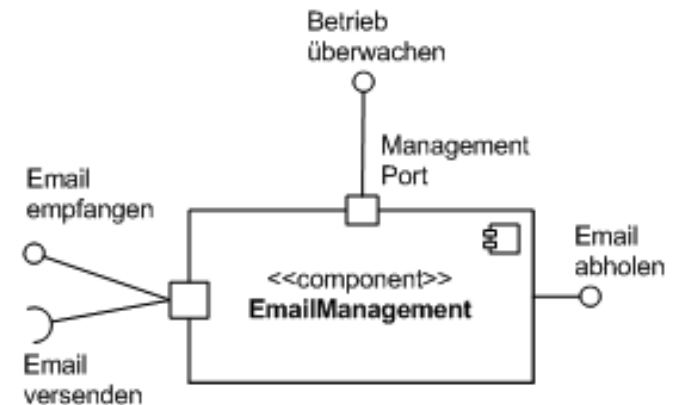
Spracheinheiten

▶ Komponenten

- ▶ Modulare Teile eines Systems
 - ▶ Könnten durch eine andere, äquivalente Komponente ersetzt werden
- ▶ Black Box-Sicht
 - ▶ Angebotene/erforderliche Schnittstellen
 - ▶ Ports

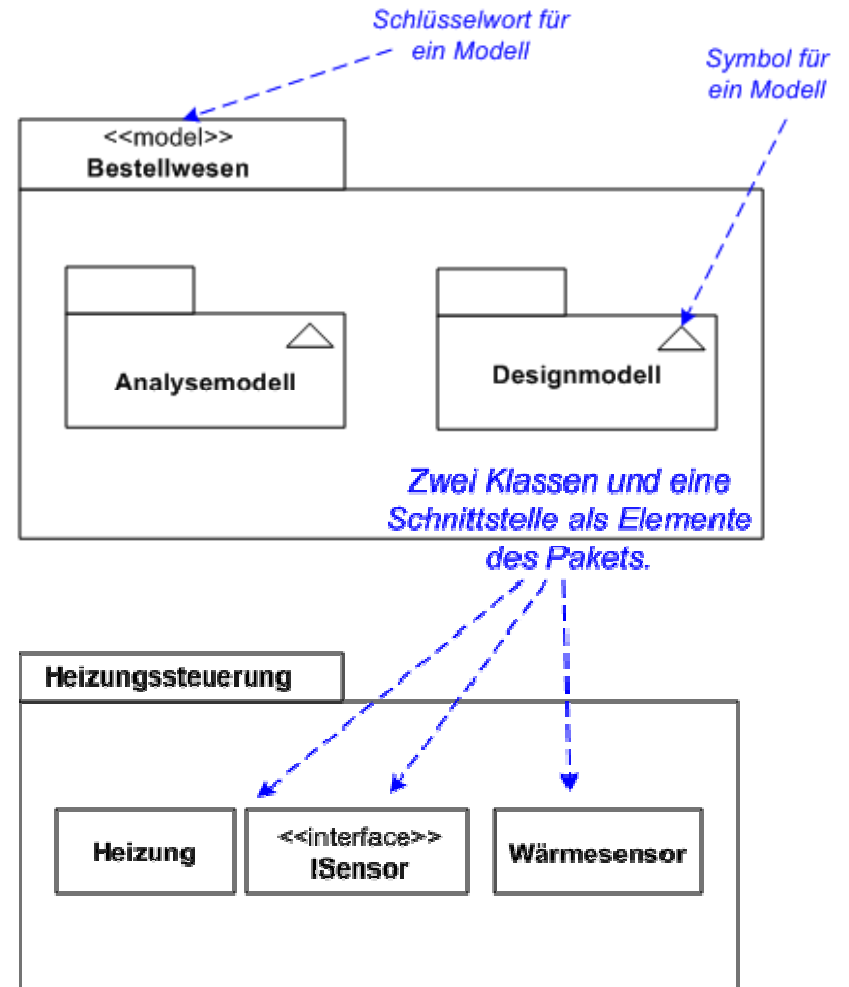
▶ Kompositionsstrukturen

- ▶ Modellierung der inneren Struktur eines zusammengesetzten Ganzen
- ▶ Das Ganze enthält Teile (Parts)
- ▶ Grenze zwischen Innen und Außen kann durchlässig sein
 - ▶ Ein- und Ausgangspunkte (Ports) auf der Hülle



Spracheinheiten

- ▶ **Modelle**
 - ▶ halten eine bestimmte Sicht auf das modellierte System fest
- ▶ **Pakete**
 - ▶ enthalten mehrere Klassen
- ▶ **Schablonen**
 - ▶ Parameterisierbare Klassen/Pakete
 - ▶ Wie generische Datentypen in Java



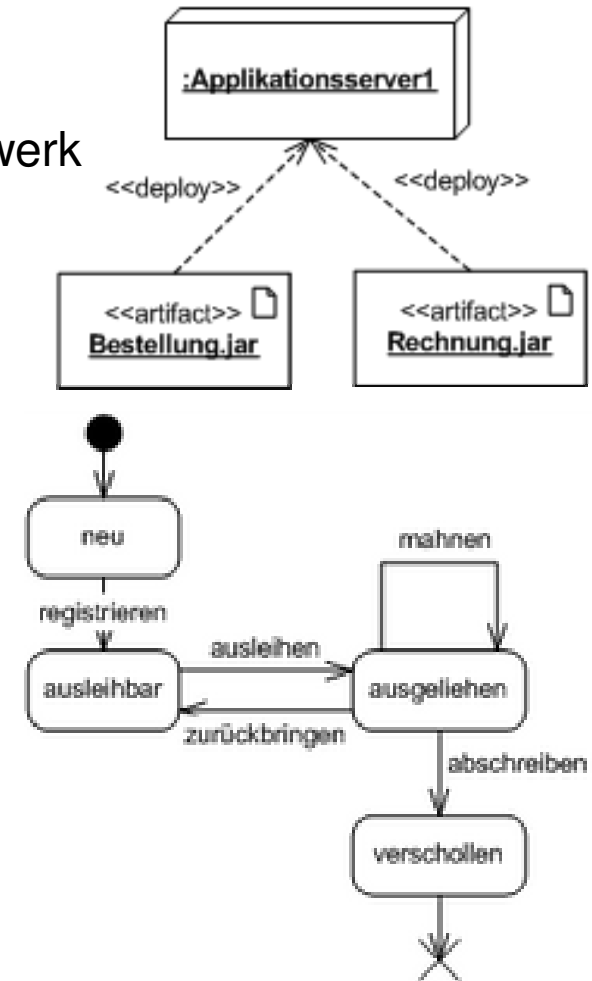
Spracheinheiten

▶ Verteilungen

- ▶ Verteilung von lauffähiger Software in einem Netzwerk
- ▶ Installation von Artefakten auf Knoten
- ▶ Knoten
 - ▶ Geräte
 - ▶ Ausführungsumgebungen

▶ Zustandsautomaten

- ▶ Verhaltenszustandsautomaten
 - ▶ Spezifikation des Verhaltens eines Systems
- ▶ Protokollzustandsautomaten
 - ▶ In welcher Reihenfolge dürfen Operationen einer Schnittstelle aufgerufen werden?





Diagramme

▶ **Strukturdiagramme**

- ▶ Klassendiagramm
- ▶ Kompositionsstruktur/
diagramm
- ▶ Komponentendiagramm
- ▶ Verteilungsdiagramm
- ▶ Objektdiagramm
- ▶ Paketdiagramm

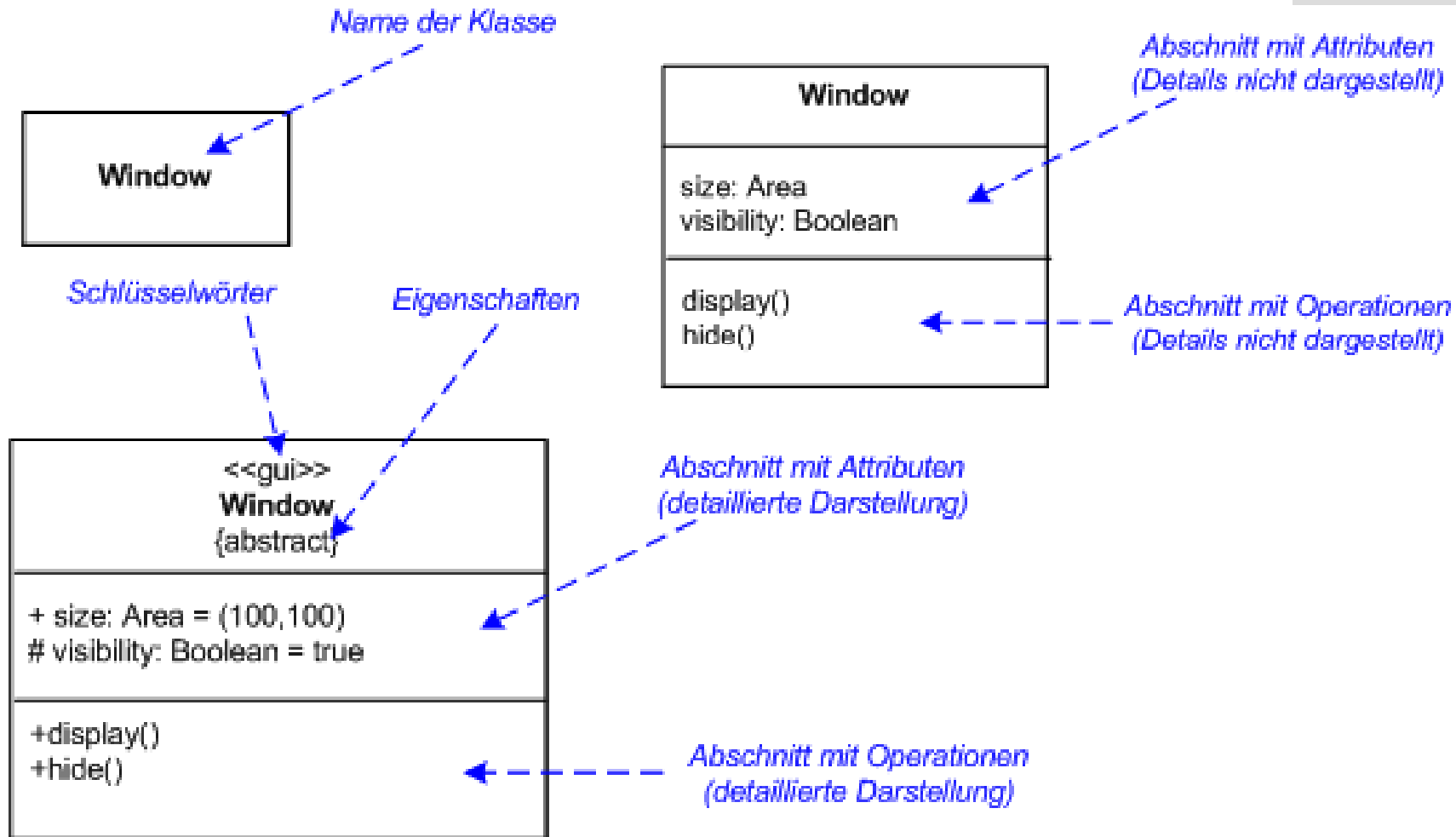
▶ **Hinweis**

- ▶ UML 2.0 erlaubt die Mischung verschiedener Spracheinheiten innerhalb der Diagramme

▶ **Verhaltensdiagramme**

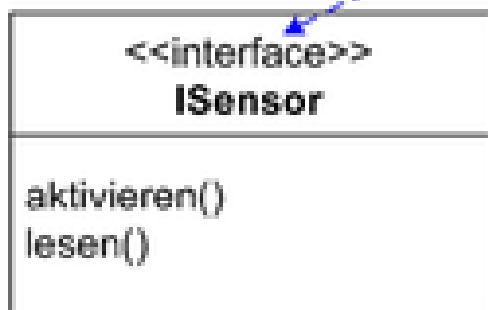
- ▶ Anwendungsfalldiagramm
- ▶ Aktivitätsdiagramm
- ▶ Sequenzdiagramm
- ▶ Kommunikationsdiagramm
- ▶ Interaktionsübersichts-
diagramm
- ▶ Zeitverlaufdiagramm
- ▶ Zustandsdiagramm

Klassendiagramme

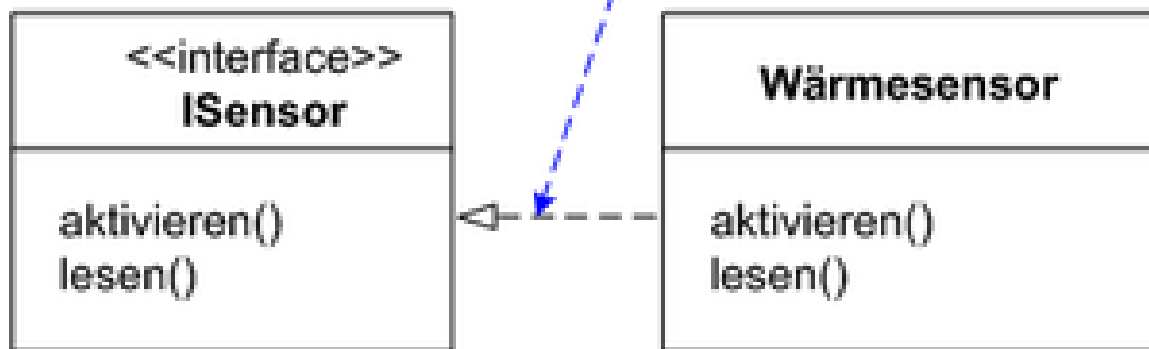


Klassendiagramme

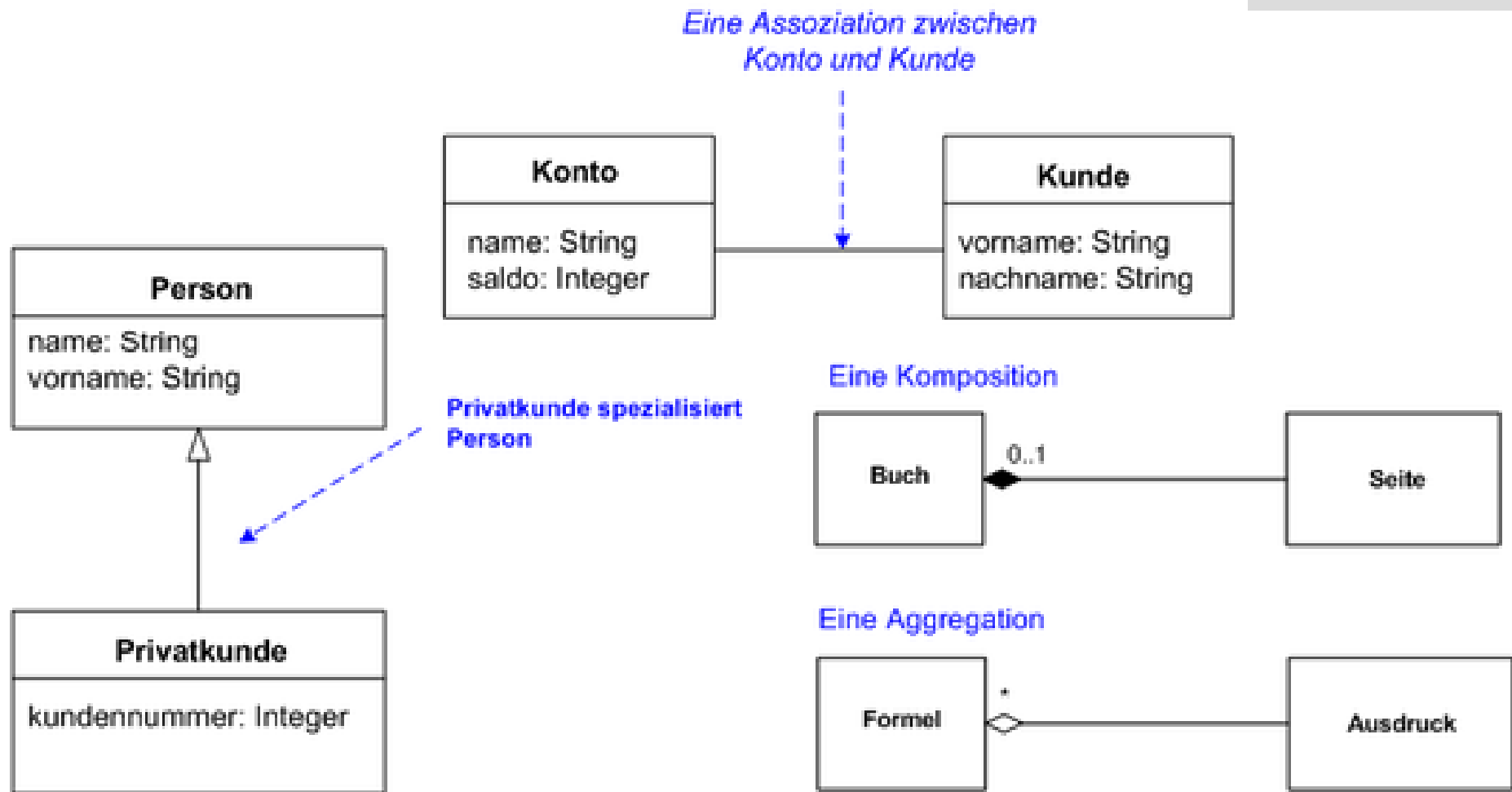
Schlüsselwort markiert
eine Schnittstelle



Notation für die Abhängigkeit
Schnittstellenrealisierung.
Wärmesensor realisiert die
Schnittstelle ISensor.

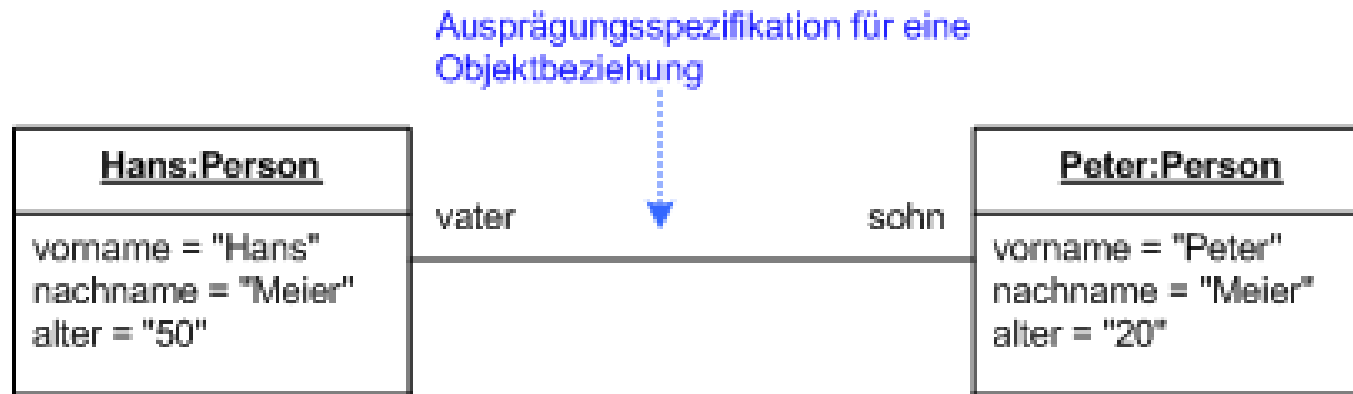


Klassendiagramme





Objektdiagramme

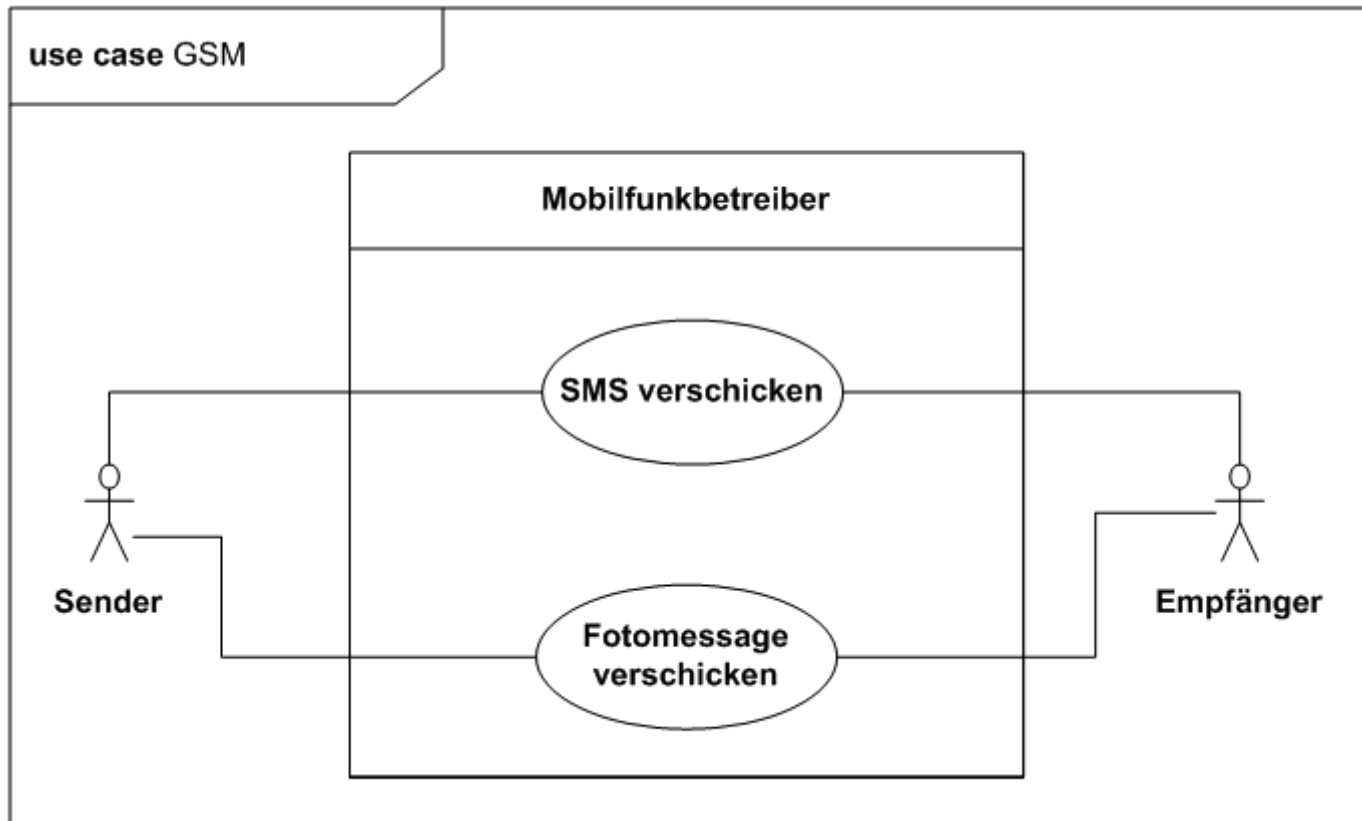




Paketdiagramme

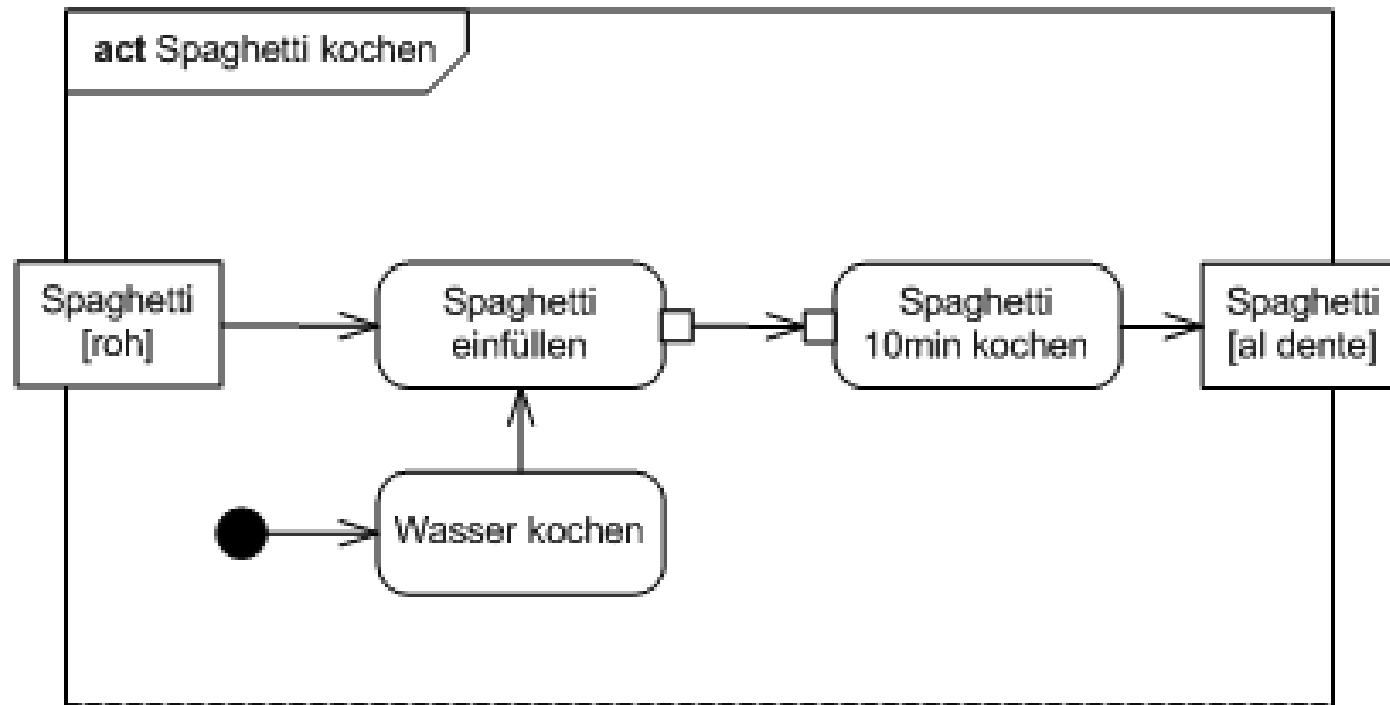


Anwendungsfalldiagramme



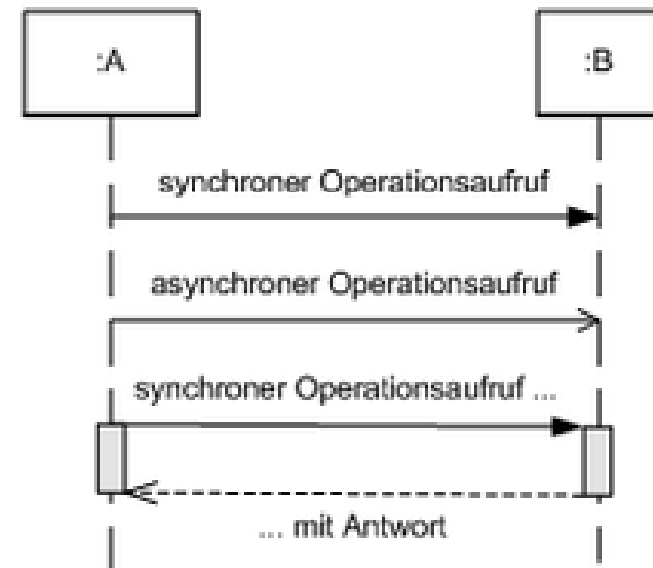
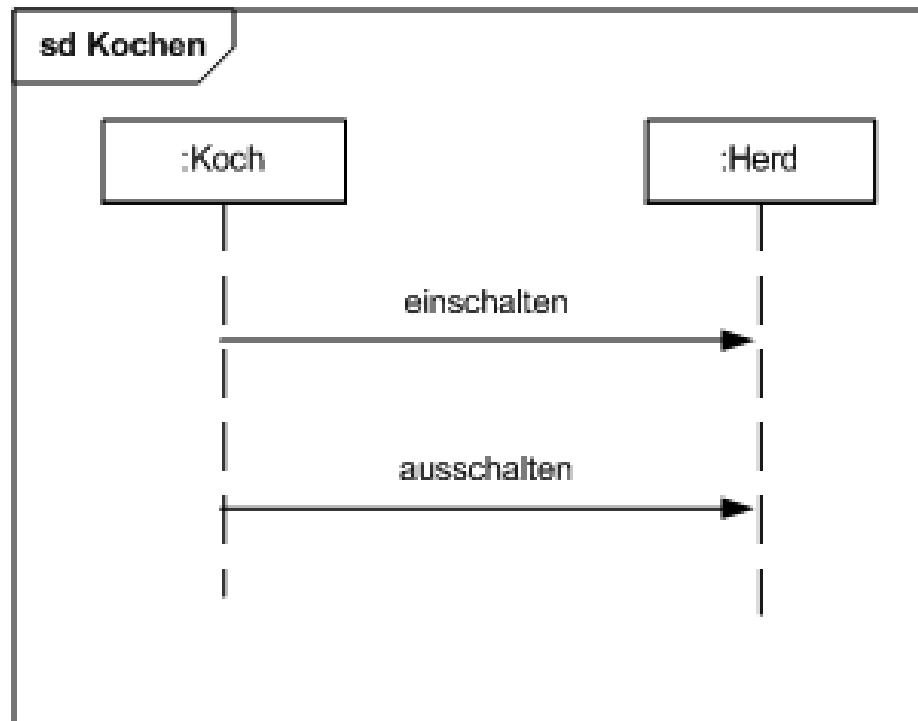


Aktivitätsdiagramme

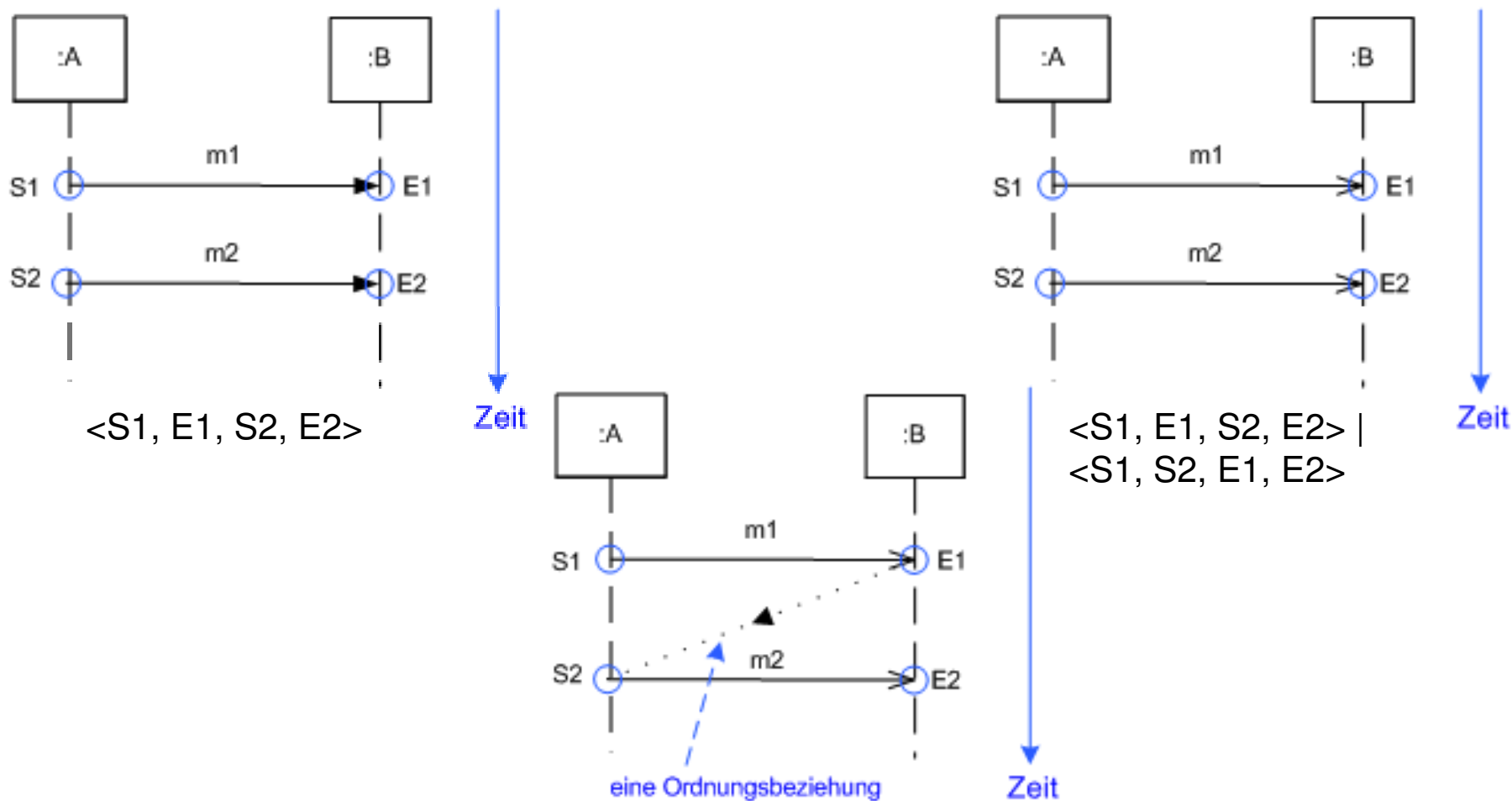




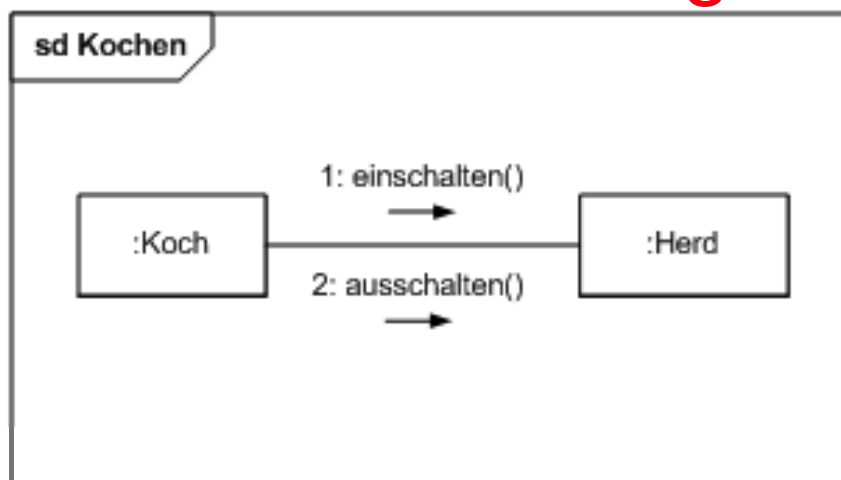
Sequenzdiagramme



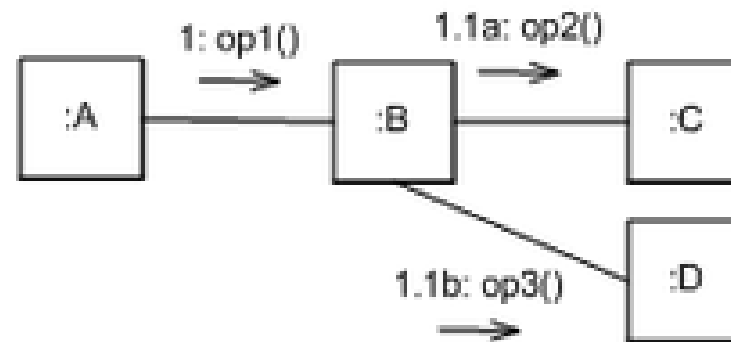
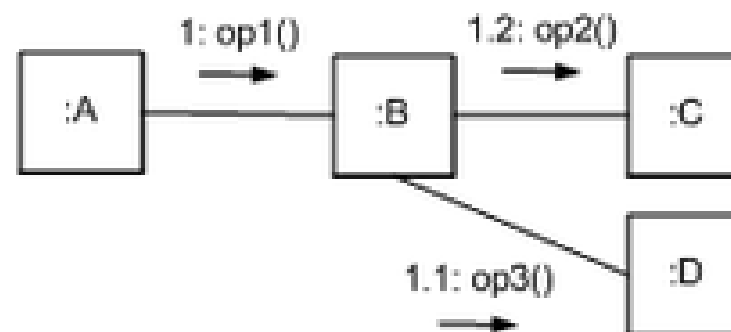
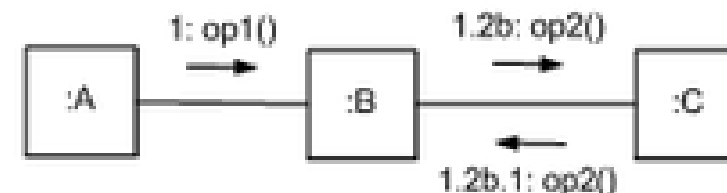
Sequenzdiagramme



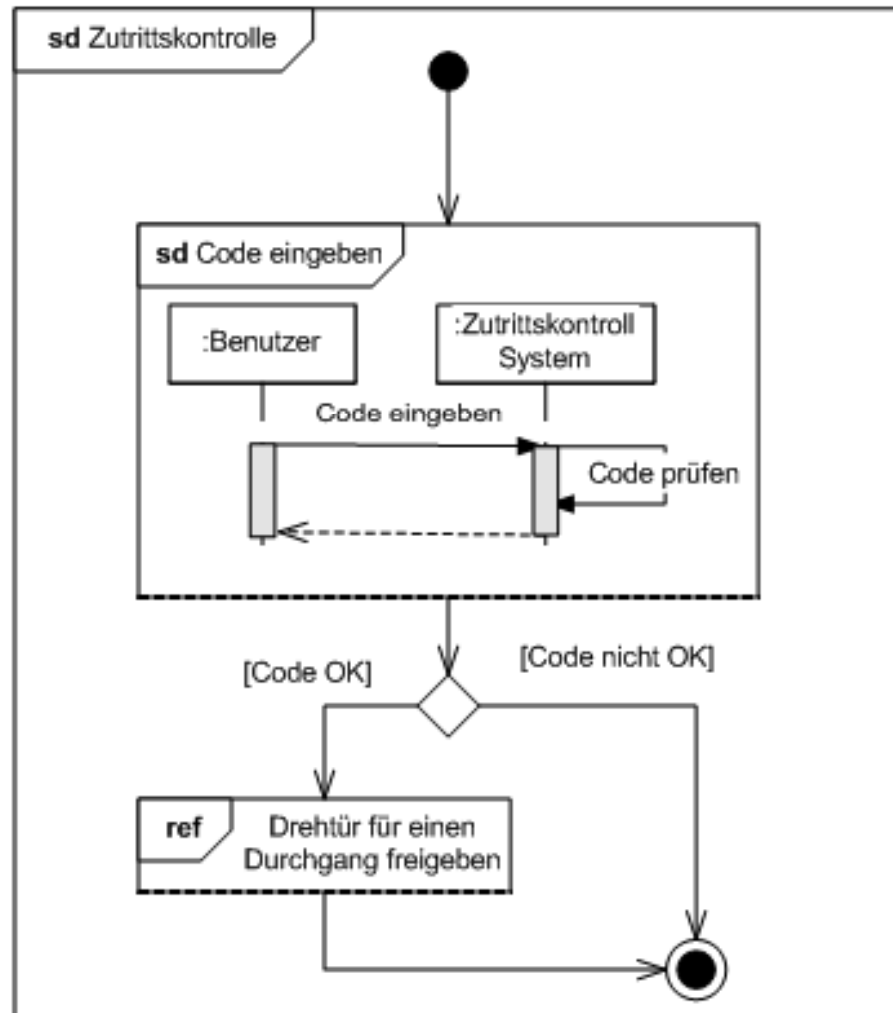
Kommunikationsdiagramme



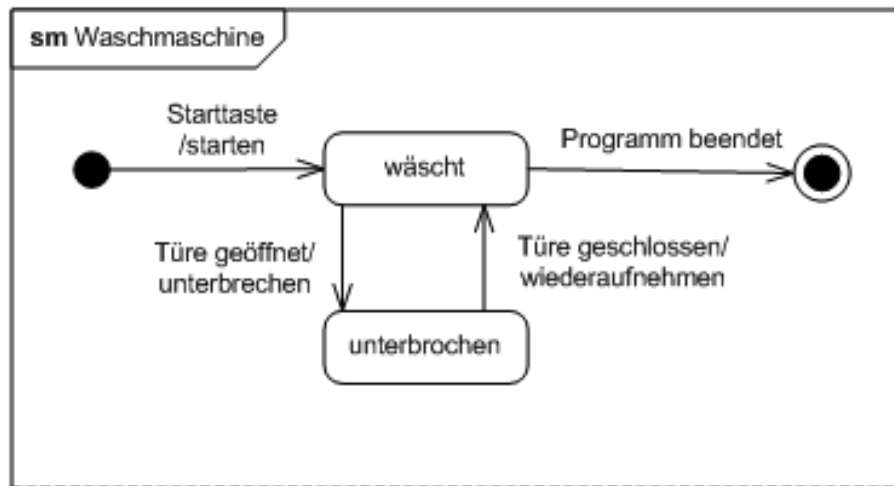
- ▶ **Zeitliche Ordnung der Nachrichten**
 - ▶ Angabe durch Verschachtelungstiefe
 - ▶ Zahlen
 - ▶ *Verschickt Nachrichten entsprechend der numerischen Ordnung*
 - ▶ Buchstaben am Ende
 - ▶ *Verschickt Nachrichten gleichzeitig*



Interaktionsübersichtsdiagramme



Zustandsdiagramme



Verhaltenszustandsdiagramm

Protokollzustandsdiagramm

