



Objektmodellierung

UML – Unified Modeling Language

Magdeburg Research and Competence Cluster
Arbeitsgruppe Wirtschaftsinformatik

Institut für Technische und Betriebliche Informationssysteme
Fakultät für Informatik

Otto-von-Guericke-Universität Magdeburg

Prof. Dr. Klaus Turowski

klaus.turowski@ovgu.de
<https://mrcc.ovgu.de>

Literaturempfehlungen

- Systems Analysis and Design with UML, 4th Edition. By Barbara Haley Wixom; David Tegarden; Alan Dennis. Published by Wiley, 2012
 - Digital und kostenlos verfügbar über den O'Reilly-Zugang der Uni-Bibliothek, vgl. <https://www.ub.ovgu.de/Literatursuche/Nutzungshinweise+f%C3%BCr+eRessourcen/Spezifische+Hinweise+zu+Verlagsangeboten.html>

Allgemeines

- **Zweck**
 - Grafische Modellierungssprache zum Visualisieren, Spezifizieren und Dokumentieren von Softwaresystemen
- **Entstehung**
 - Anfang der 90er existierte eine Vielzahl unabhängiger objektorientierter Modellierungssprachen
 - *Object-Oriented Design* (OOD) von Grady Booch
 - *Object Modeling Technique* (OMT) von James Rumbaugh
 - *Object-Oriented Software Engineering* (OOSE) von Ivar Jacobsen
 - Zusammenarbeit der „drei Amigos“ bei Rational Software führte Mitte der 90er zur Idee der strukturierten Zusammenführung

Allgemeines

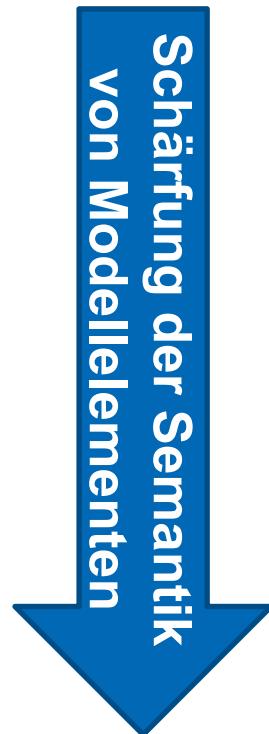
- **Ziele**
 - Objektorientierte Modellierung einer Software vom Konzept bis hin zum ausführbaren Artefakt (Werkzeug)
 - Abbildung von komplexen, unternehmenskritischen Anwendungen möglich
 - Verständlichkeit der Sprache für Menschen und Maschinen
- **Notationsübersicht**
<http://www.oose.de>

Allgemeines

- Historie
 - 1996: UML-Version 0.9
 - Zusammenschluss führender Software Unternehmen (z. B. IBM, Microsoft, Oracle, Hewlett-Packard, Rational Software Corporation) zum UML-Konsortium
➤ Ergebnis: UML-Version 1.0
 - 19.11.1997: UML-Version 1.1 als Standard mit Zertifizierung der *Object Management Group* (OMG)

Allgemeines

- **Historie**
 - 2005: UML 2.0
 - 2007: UML 2.1
 - 2009: UML 2.2
 - 2010: UML 2.3
 - 2011: UML 2.4
 - 2012: UML 2.5 (aktuell)



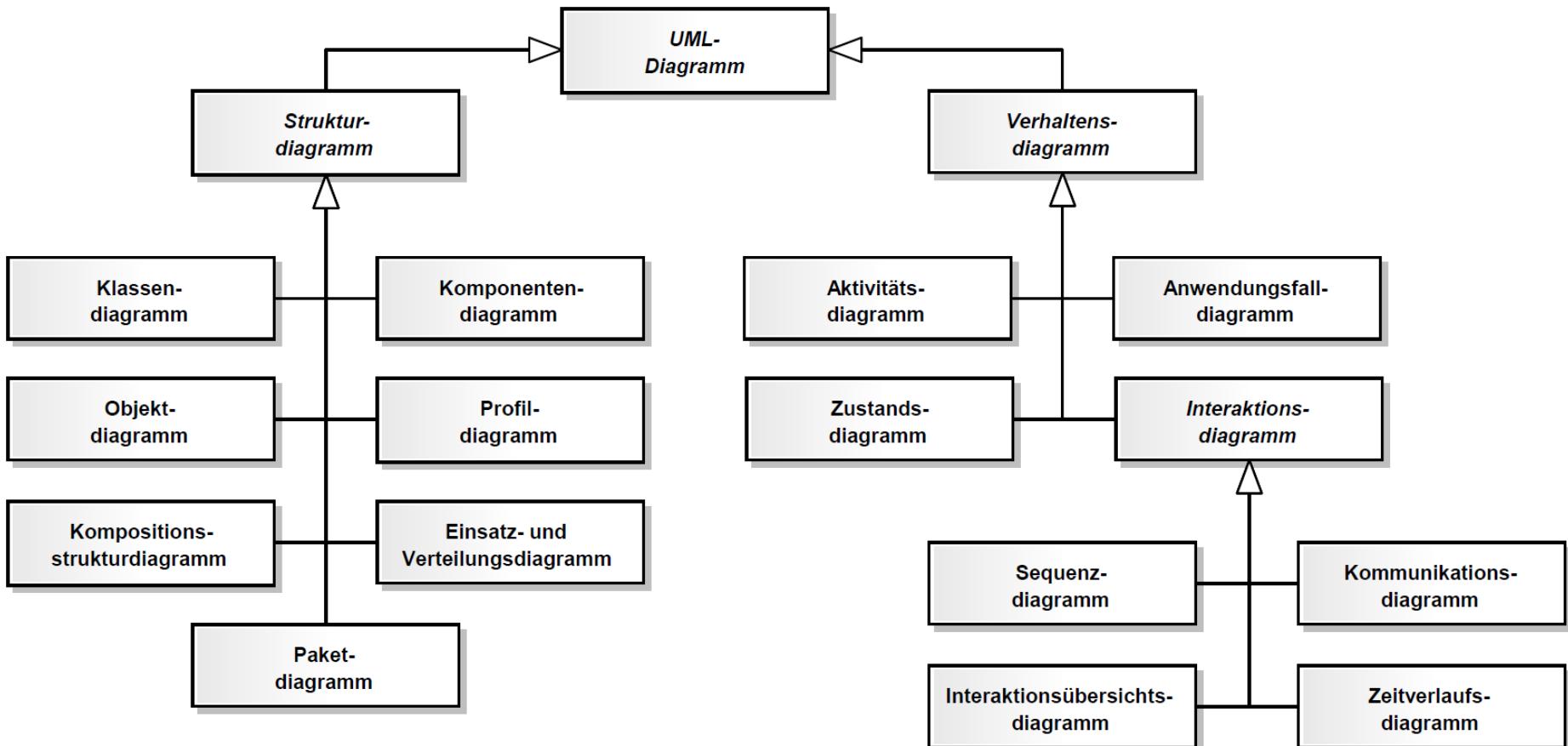
Strukturierung

- Verschiedene Perspektiven auf ein objektorientiertes Modell
- **Strukturdiagramme**
 - Beschreibung aller statischen Elemente eines Systems und deren Beziehung zueinander
 - Beschreibung des Aufbaus von Systemen
 - Beispiele:
 - Klassen und Schnittstellenklassen
 - Komponenten und Schnittstellen
- **Verhaltensdiagramme**
 - Beschreibung der dynamischen Aspekte eines Systems einschließlich der Reaktion auf eintretende Ereignissen
 - Beispiel:
 - Prozessbeschreibung

Strukturierung

- **Interaktionsdiagramme**
 - Spezielle, kommunikationsbezogene Verhaltensdiagramme
 - Beschreibung des Austauschs von Meldungen zwischen eigenständigen Objekten
 - Beispiel:
 - Abbildung von synchroner bzw. asynchroner Kommunikation zwischen Komponenten

Strukturierung



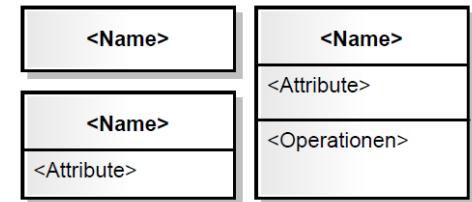
Klassendiagramm

Strukturdiagramme: Klassendiagramm

- Darstellung von Klassen und Schnittstellenklassen sowie deren Eigenschaften und Beziehungen untereinander
- Konzeptuelle und spezifizierende Sicht auf eine Software
 - Spezifizierende Sicht: Attribute und Operationen einer Klasse sowie deren Sichtbarkeiten
 - Konzeptuelle Sicht: Beziehungen und Abhängigkeiten zwischen einzelnen Klassen
 - Implementierungsnah, Grundlage für automatische Code-Generierung

Strukturdiagramme: Klassendiagramm

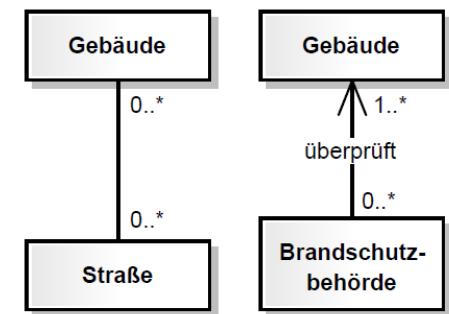
- Notation der spezifischen Sicht
 - Beschreibung von Klassen



Klasse mit Attributen und Methoden



Abstrakte Klasse und Schnittstellenklasse



Ungerichtete
Assoziation

Gerichtete
Assoziation

Assoziationen in Java (1/2)

- Ungerichtete Assoziation

```
import java.util.List;

public class Gebäude {
    List<Straße> ungerichteteAssoziation;
    public void assoziiereStraße(Straße straßenRef) {
        ungerichteteAssoziation.add(straßenRef);
    }
}

public class Straße {
    List<Gebäude> ungerichteteAssoziation;
    public void assoziiereGebäude(Gebäude gebäudeRef) {
        ungerichteteAssoziation.add(gebäudeRef);
    }
}
```

Assoziationen in Java (2/2)

- Gerichtete Assoziation

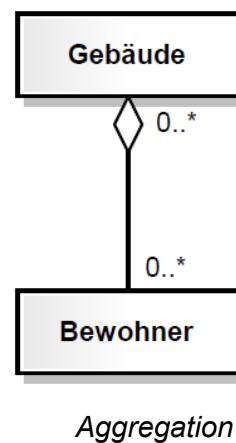
```
import java.util.List;
```

```
public class Gebäude {  
}
```

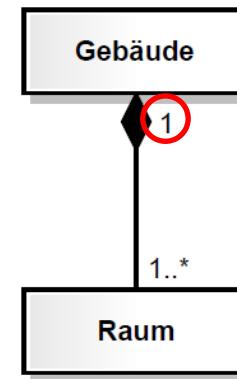
```
public class Brandschutzbehörde {  
    List<Gebäude> überprüft;  
    public void assoziiereGebäude(Gebäude gebäudeRef) {  
        überprüft.add(gebäudeRef);  
    }  
}
```

Strukturdiagramme: Klassendiagramm

- Notation der konzeptuellen Sicht
 - **Aggregation**
 - Keine gleichwertige Beziehung zwischen Klassen sondern logische Hierarchie zwischen Ganzen und seinen Teilen
 - **Komposition**
 - Spezialform der Aggregation bei der die Teile eines Ganzen von diesem existenzabhängig sind



Aggregation



Komposition

Aggregation und Komposition in Java

- Aggregation
 - Die Aggregation entspricht in Java der gerichteten Assoziation
- Komposition (mit Multiplizität Raum: 0..n)

```
public class Gebäude {  
    List<Raum> räume;  
  
    public void erstelleRaum() {  
        // Räume existieren nur in  
        // Abhängigkeit von Gebäuden  
        räume.add(new Raum());  
    }  
}
```

Komposition in Java

- Komposition (Multiplizität der Klasse Raum: 1..n wobei n=4)

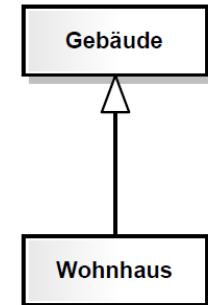
```
public class Gebäude {  
    Raum schulungsraum;  
    Raum sekretariat;  
    Raum büro1;  
    Raum büro2;  
  
    public void erstelleSchulungsraum() {  
        schulungsraum = new Raum();  
    }  
  
    public void erstelleSekretariat() {  
        sekretariat = new Raum();  
    }  
    ...
```

Strukturdiagramme: Klassendiagramm

- Notation der konzeptuellen Sicht

- **Generalisierung / Spezialisierung**

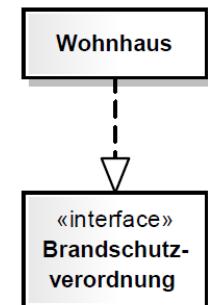
- Vererbung von Attributen und Operationen einer allgemeiner Klasse an eine spezielle
 - Spezielle Klasse kann zusätzliche Eigenschaften hinzufügen und verhält sich kompatibel zur allgemeinen



Generalisierung /
Spezialisierung

- **Realisierung**

- Implementierung einer oder mehrerer Schnittstellenklassen und derer Attribute sowie Operationen durch eine Klasse



Realisierung

Generalisierung und Realisierung in Java

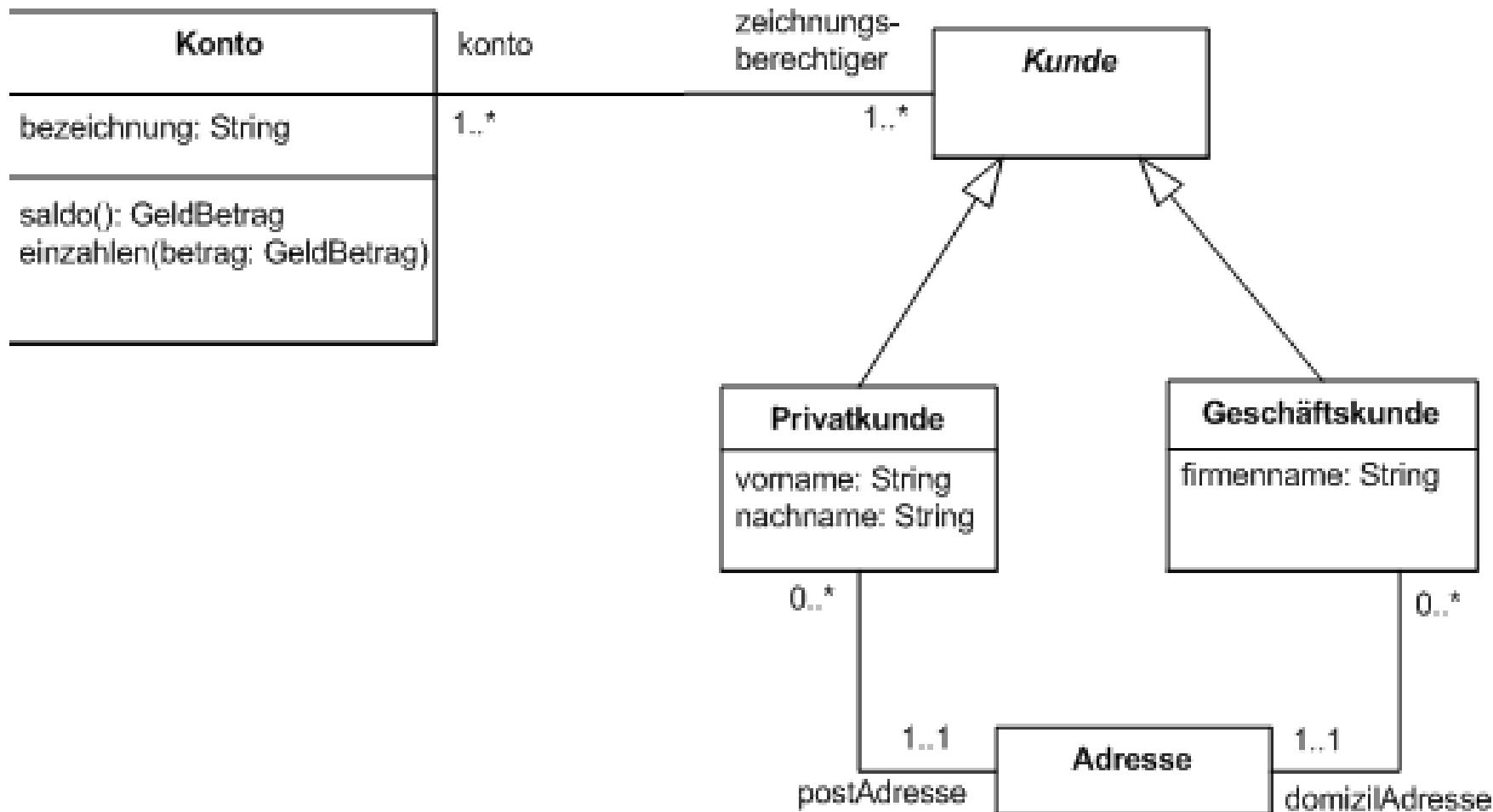
- Generalisierung / Spezialisierung

```
public class Wohnhaus extends Gebäude {  
}
```

- Realisierung

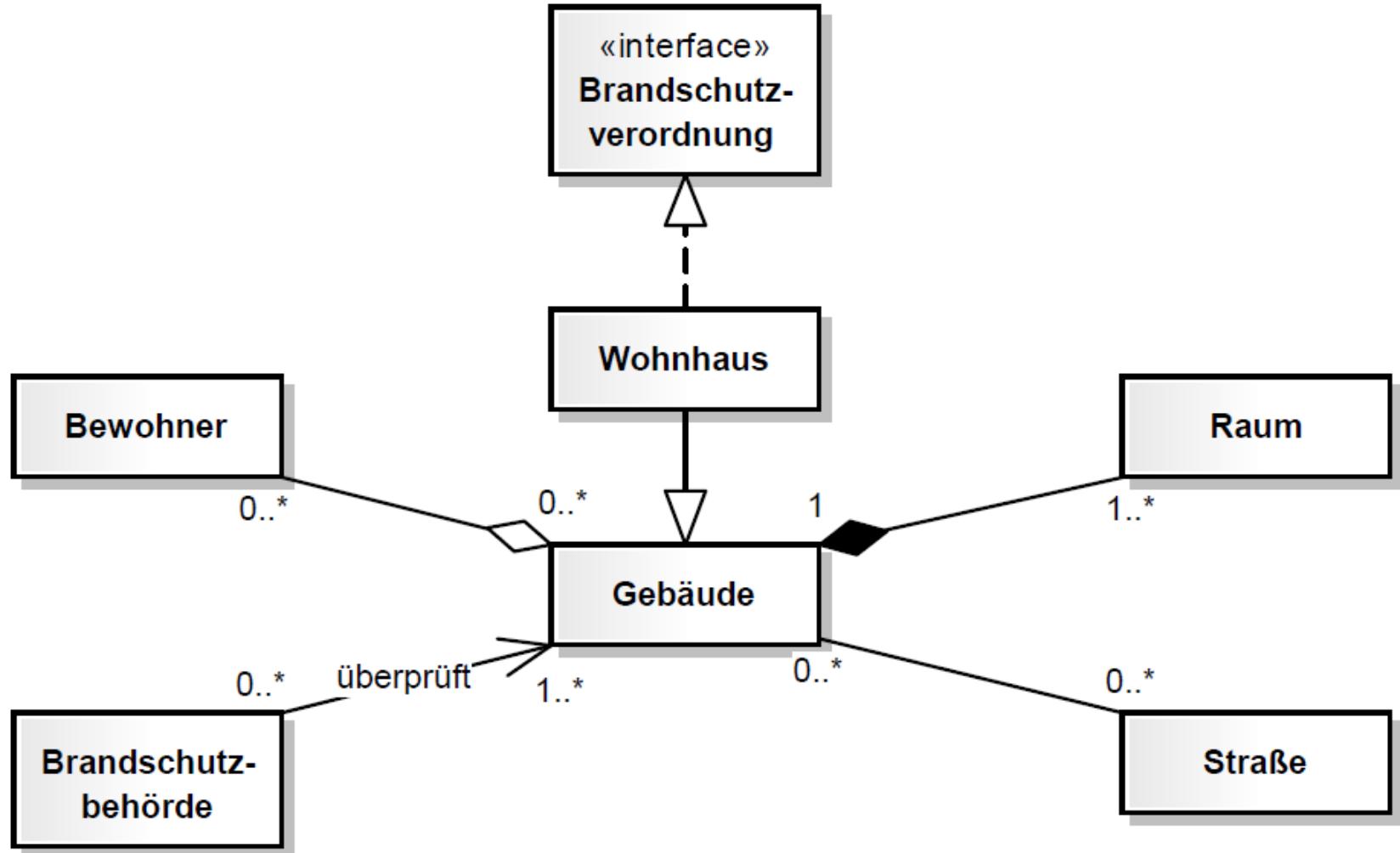
```
public interface Brandschutzverordnung {  
    public int getAnzahlSprinkleranlagen();  
    public int getAnzahlGaslöschanlagen();  
}  
  
public class Wohnhaus implements Brandschutzverordnung {  
    public int getAnzahlSprinkleranlagen() {  
        // Methodeninhalt  
    }  
    public int getAnzahlGaslöschanlagen() {  
        // Methodeninhalt  
    }  
}
```

Strukturdiagramme: Klassendiagramm



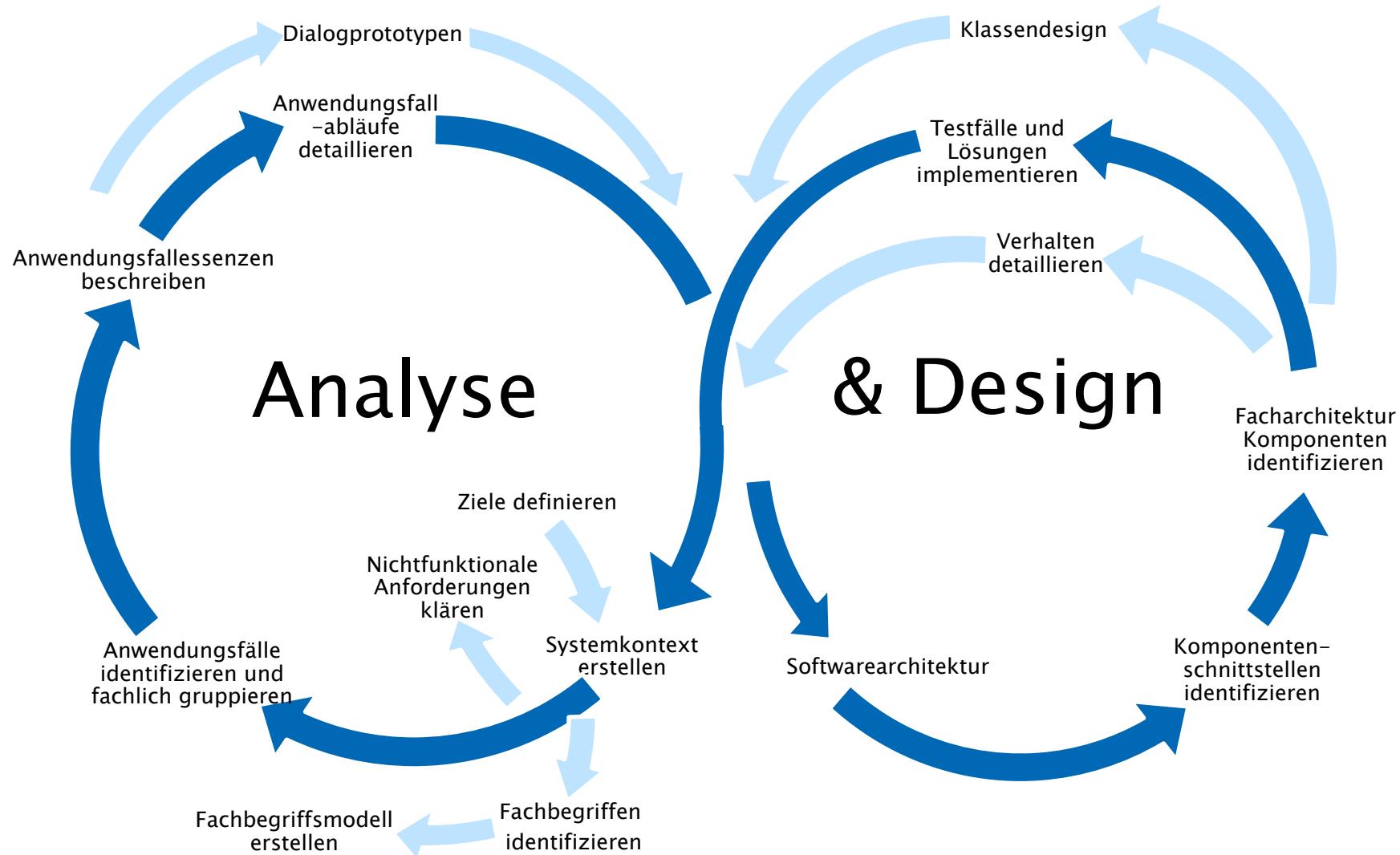
Strukturdiagramme: Klassendiagramm

- Übersicht



Klassendiagramm: Beschreibung des Umfelds von Gebäuden

UML-Methodikleitfaden für Analyse und Design



Analyse: Ziele und Systemkontext

- **Ziele definieren**
 - **Wozu?** Klarheit, Motivation, Sinn
 - **Was?** Grundsätzliches, Features, Visionen, Wünsche, Absichten
 - **Wie?** Produktkarton, aktiv & überzeugend, kurz & knackig,
5–20 Sätze
 - **Wer?** Analytiker, Stakeholder, Projektleiter
- **Systemkontext erstellen**
 - **Wozu?** Abgrenzung des Systems nach außen: Was ist drinnen?
Was ist draußen?
 - **Wie?** Systemkontextdiagramm, <<system>> Stereotyp für
Klasse, Akteure
 - **Wer?** Analytiker in mehrfacher Abstimmung mit Stakeholdern
und Architekten im Workshop



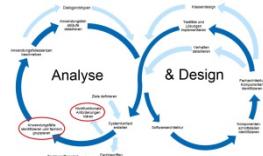
Analyse: Fachbegriffe und Fachbegriffsmodell

- **Fachbegriffe identifizieren**
 - **Wozu?** Einheitliches Begriffsverständnis aller Beteiligten
 - **Was?** Fachliche systemrelevante Substantive identifizieren (z.B. Gegenstand, Konzept, Ort, Person)
 - **Wie?** Einfache Klassen mit Attributen
 - **Wer?** Analytiker, anschließend abstimmen mit Fachexperten
- **Fachbegriffsmodell erstellen**
 - **Wozu?** Zusammenhänge zwischen Fachbegriffen erkennen
 - **Wie?** Einfaches Klassendiagramm mit Assoziationen – soweit notwendig – zum Erfüllen der Anforderungen
 - **Wer?** Analytiker, anschließend abstimmen mit Fachexperten



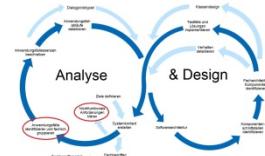
Analyse: Nichtfunktionale Anforderungen und Anwendungsfälle (1 / 2)

- **Nichtfunktionale Anforderungen klären**
 - **Wozu?** Rahmenbedingungen und Qualitätsmerkmale des Systems festlegen, um darauf Architekturentscheidungen zu treffen
 - **Wie?**z.B. nach dem FURPS-Modell (Functional, Usability, Reliability, Performance, Supportability)
 - **Wer?** Analytiker zusammen mit Fachexperten



Analyse: Nichtfunktionale Anforderungen und Anwendungsfälle (2/2)

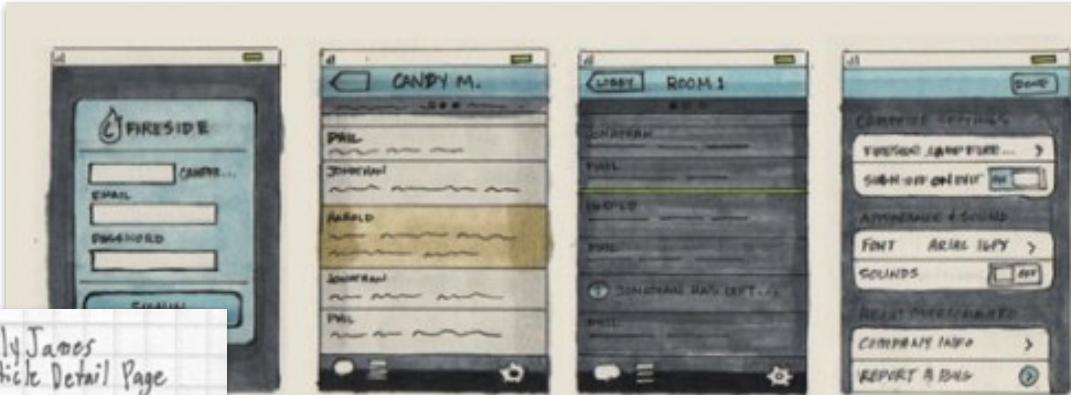
- **Anwendungsfälle identifizieren und fachliche gruppieren**
 - **Wozu?** funktionale Anforderungen, Systemdienstleistungen identifizieren
 - **Was?** Systemanwendungsfall als zielgerichtete und kohärente Interaktion eines Außenstehenden mit dem System
Motiviert durch einen fachlichen Auslöser, liefert Ergebnis von fachlichem Wert
 - **Wie?** Anwendungsfalldiagramme fachlich strukturiert;
Workshop mit Fachexperten des Systems
 - **Wer?** Fachexperten und Analytiker im Workshop



Analyse: Anwendungsfallessenzen und Dialogprototypen

- **Anwendungsfallessenzen beschreiben**
 - **Wozu?** Schneller Überblick über den Gutfall, Redundanzen frühzeitig erkennen, einheitliches Verständnis
 - **Was?** Fachliche Intention des Gutfall–Ablaufs, generalisiert, technologieunabhängig, abstrakt
 - **Wie?** textuell
 - **Wer?** Analytiker, anschließend abstimmen mit Fachexperten
- **Dialogprototypen**
 - **Wozu?** visuelles Verständnis des System erhalten, Benutzbarkeitsanforderungen erarbeiten
 - **Wie?** Stift & Papier, um ermüdende Diskussionen zu vermeiden oder elektronisch, um auch Navigation zu simulieren
 - **Wer?** Analytiker in Zusammenarbeit mit Fachexperten





- xx Place nav links above logo?
- xx Twitter icon placed on right side.
- xx Much larger arrows to fill space
- xx Should intro/sugary be heavily emphasized?

Only James

Home

Articles

Blog

About

This is my latest tweet from Twitter.
(Follow Me)

Summary of the following content

Why in the world Twitter Rules!

Issue #2

← Newer Article Older Article →

*** This is the content of the article, and it will depend on what is being talked about.

...
Like This? Share It! •

• click transforms into social media links

Analyse: Anwendungsfallabläufe (1 / 2)

■ Anwendungsfallabläufe detaillieren

- Wozu?
- Wie?

Detailliertes fachliches Verständnis der Systemabläufe erhalten

- Eine Aktivität pro Anwendungsfall formulieren
- Für jeden essenziellen Schritt eine Aktion formulieren und mit Stereotyp <<essential>> markieren
- Reihenfolge der Aktionen bestimmen
- Durch Modellieren des Standardablaufs Variationen im Standardablauf bestimmen
- Aktivitätsdiagramm z.B. um Wiederholungen erweitern
- Mögliche Fehler und Ausnahmen eines jeden Schrittes dem Aktivitätsdiagramm hinzufügen

...



Analyse: Anwendungsfallabläufe (2/2)

...

- Abbrüche z.B. durch Signale und unterbrechbare Bereiche modellieren
- Schritte mit vielen Ausnahmen, Entscheidungen, hohen Komplexitäten oder Schritte, die in mehreren Aktivitäten auftauchen, verfeinern
- **Wer?** Analytiker, anschließend diskutieren mit Fachexperten und Feedback einarbeiten



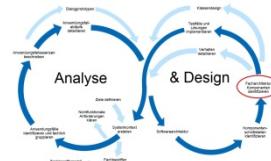
Design: Softwarearchitektur und Komponentenschnittstellen

- **Softwarearchitektur**
 - Vorhandene Rahmenbedingungen und grundlegende Architektur
- **Komponentenschnittstellen identifizieren**
 - **Wozu?** Komponentenunabhängigkeit und technische Kommunikation festlegen
 - **Wie?** Aktivitätsdiagramm mit Objektfluss, Sequenzdiagramme oder Zustandsdiagramme, Kompositionsstrukturdiagramme mit Schnittstellenkonnektoren
 - **Wer?** Softwarearchitekten und –entwickler



Design: Facharchitektur

- **Facharchitektur: Komponenten identifizieren**
 - **Wozu?** Fachliche Infrastrukturkomponenten auswählen und bestimmen
 - Zur Realisierung eines Anwendungsfalls
 - Fachliche Komponenten aus Anwendungsfallpaketen ermitteln
 - Infrastrukturkomponenten identifizieren
 - Kompositionsstrukturdigramm für die Architektenübersicht
 - **Wie?** Softwarearchitekten und –entwickler



Design: Klassendesign

- **Klassendesign**
 - **Wozu?** Detailentscheidungen treffen
 - **Was?** Detailliertes Klassendesign als Realisierungsgrundlage
 - **Wie?** Klassendiagramme mit Navigationsrichtungen, Kompositionen, Aggregation, Interfacerealisierung, Kollaborationen, Designprinzipien
 - **Wer?** Softwarearchitekten und –entwickler



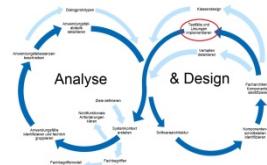
Design: Verhalten

- **Verhalten detaillieren**
 - **Wozu?** Detailentscheidungen treffen
 - **Was?** Detaillierte Aktivitätsdiagramme mit Objektfluss, Sequenzdiagramme, Zustandsdiagramme
 - Für ausgezeichnetes Verhalten, komplexe Abläufe, kritische Situationen
 - Nach Bedarf Zustandsautomaten
 - Aktivitätsdiagramm mit Objektfluss zum Identifizieren von Operationen und Parametern
 - Sequenzdiagramme für den Nachrichtenaustausch beschreiben
 - **Wie?**
 - **Wer?**



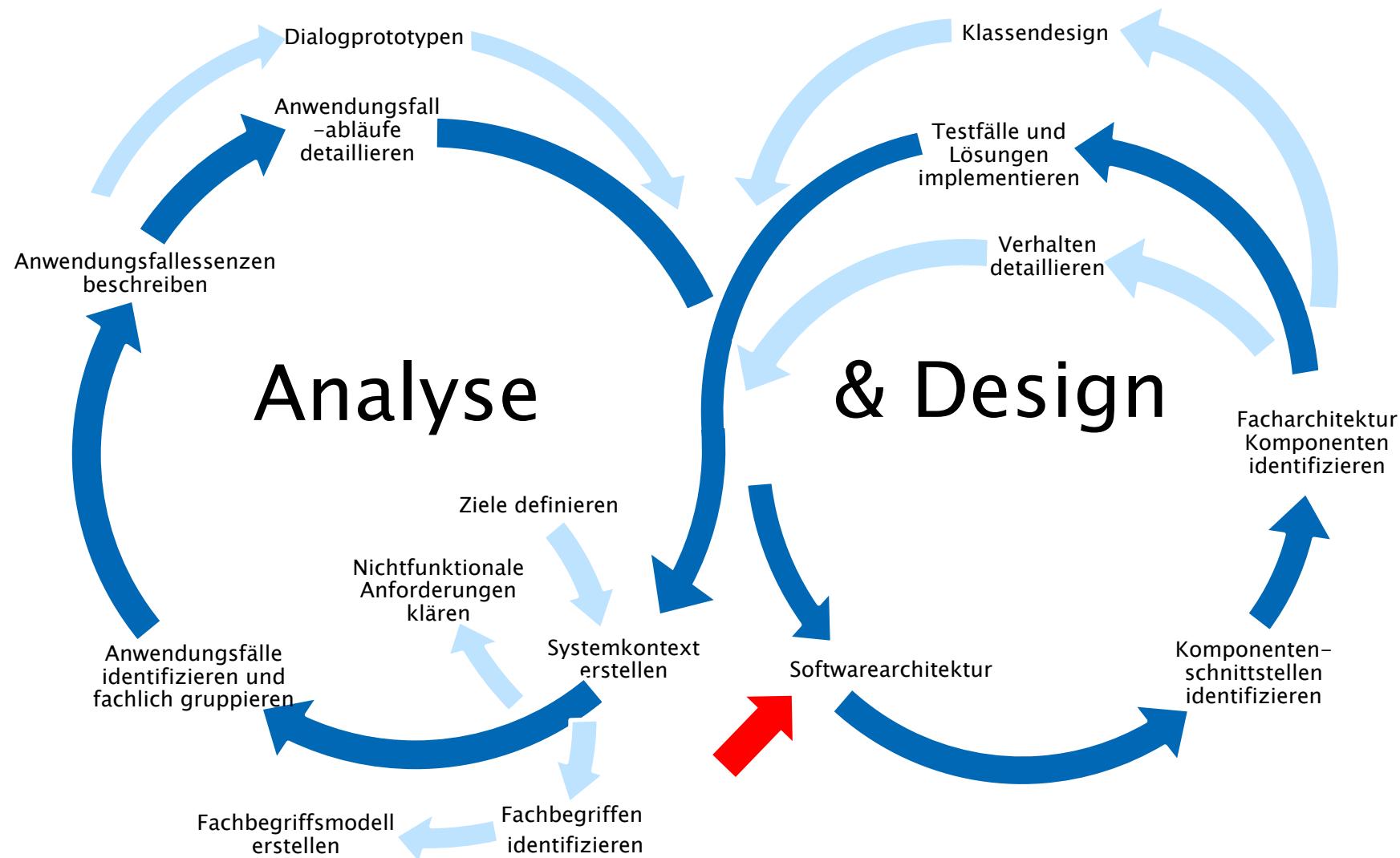
Design: Testfälle und Lösung

- **Testfälle und Lösung implementieren**
 - **Was?** Erst den Test, dann die Funktionalität schreiben
 - Sicherstellen, dass nicht mehr oder weniger programmiert wird als erforderlich, d.h. herausfinden, was wirklich zu programmieren ist
 - Anforderungen formal beschreiben
 - Testbarkeit und Testautomatisierung erleichtern
 - **Wozu?** Gerade so viel Test und Funktionalität programmieren, dass Test und Code kompilieren. Dann Funktionalität erweitern, damit Test erfolgreich durchläuft.
 - **Wie?** Softwareentwickler
 - **Wer?**



Komponentendiagramm

UML-Methodikleitfaden für Analyse und Design



Quelle: Oestereich, B., Scheithauer, A.: Analyse und Design mit der UML 2.5: Objektorientierte Softwareentwicklung. Oldenbourg Wissenschaftsverlag, München (2012) (Poster zum Buch)

Strukturdiagramme: Komponentendiagramm

- Beschreibt die Architektur eines Systems sowie die Schnittstellen und Abhängigkeiten zwischen einzelnen Systembausteinen
- Notation
 - **Komponente**
 - Einzelter unabhängiger Systembaustein
 - **Komponentencontainer**
 - Logische Zusammenfassung mehrerer Komponenten



Komponente



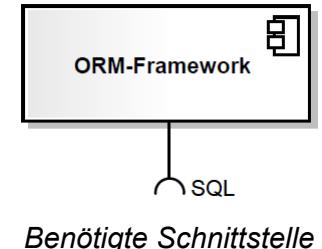
Komponentencontainer

Strukturdiagramme: Komponentendiagramm

▪ Notation

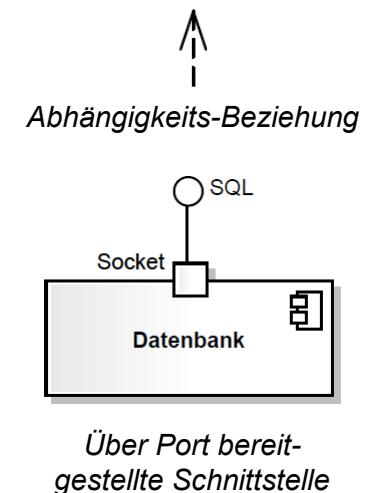
○ Port

- Beinhaltet Schnittstellen
- Stellt Endpunkt für eine Kommunikation dar
- Kommunikation verläuft über Ports, nicht über die Komponenten (Entkopplung von den Komponenten)



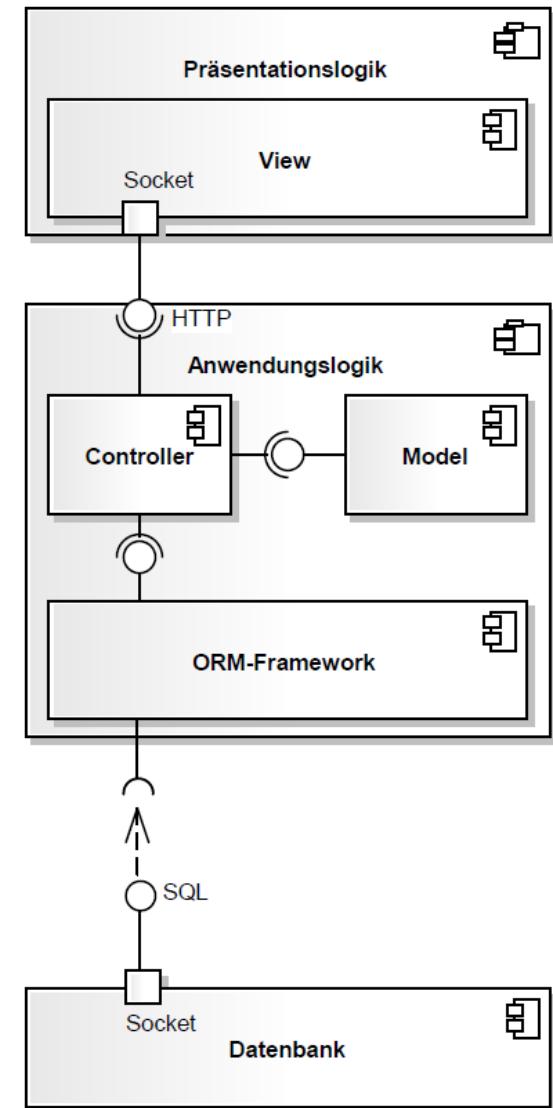
○ Schnittstelle

- Benötigte Schnittstelle (Buchse): Von einer Komponente genutzt
- Bereitgestellte Schnittstelle (Stecker): Von einer Komponente angeboten
- Notation der Beschreibung der Schnittstelle (z.B. Protokoll)
- Verbindung über **Abhängigkeits-Beziehung**



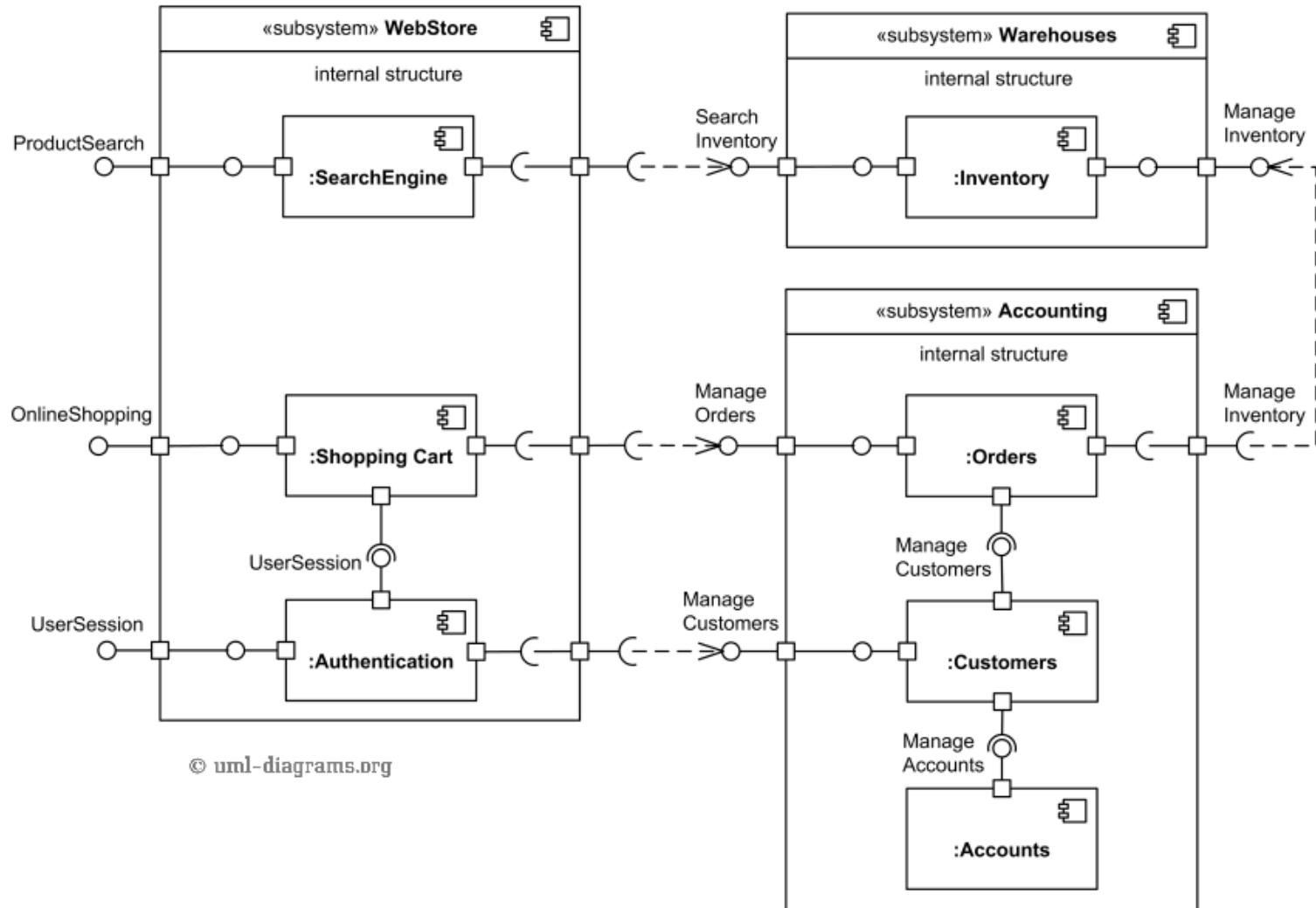
Strukturdiagramme: Komponentendiagramm

- Übersicht



Komponentendiagramm: 3-Tier-Architektur einer Web-Anwendung

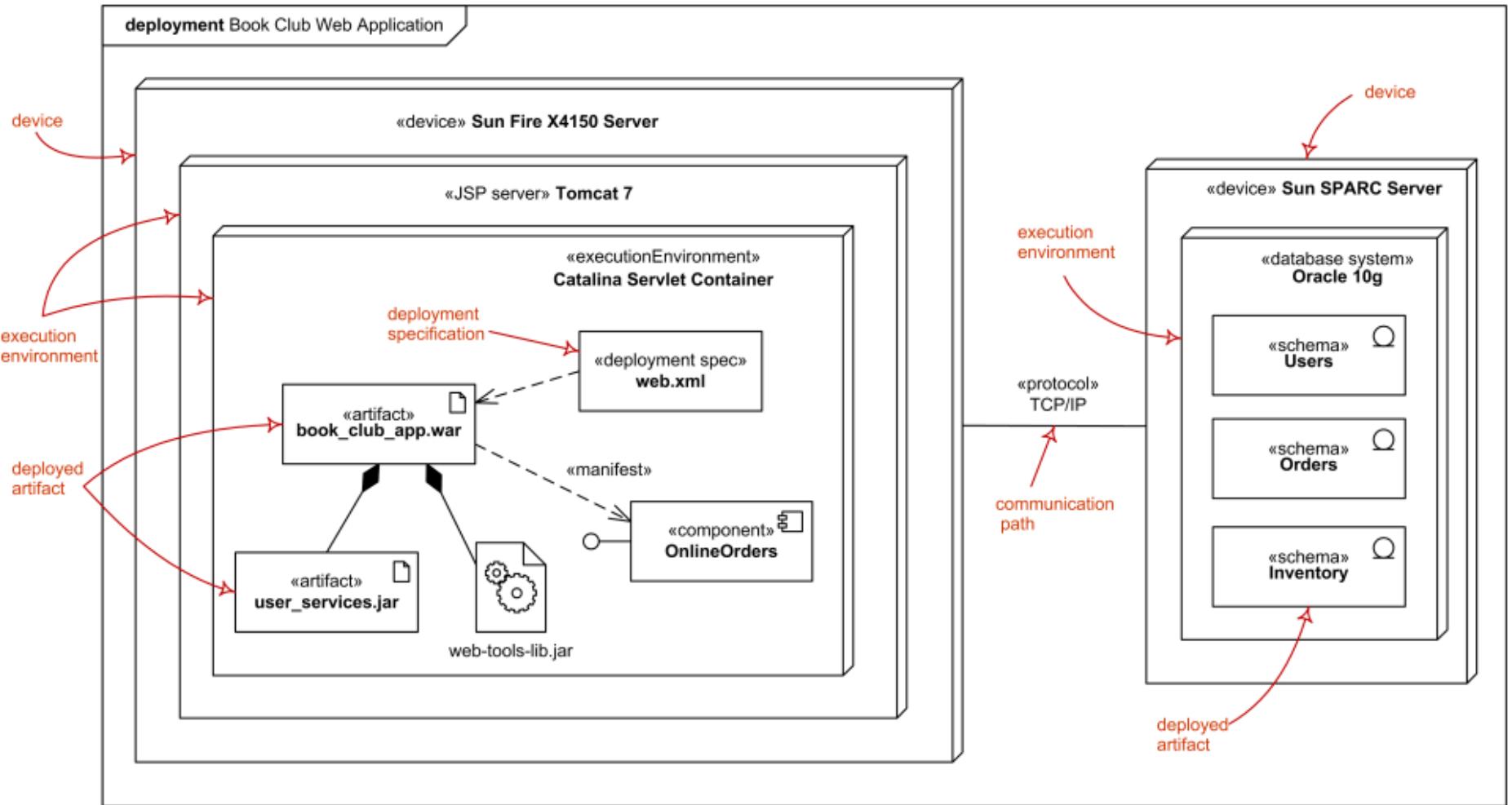
Strukturdiagramme: Komponentendiagramm – Beispiel



Quelle: [http://www.uml-diagrams.org/examples/online-shopping-uml-component-diagram-example.html?context=cmp-examples, Abruf 27.03.2015]

Einsatz- und Verteilungsdiagramm

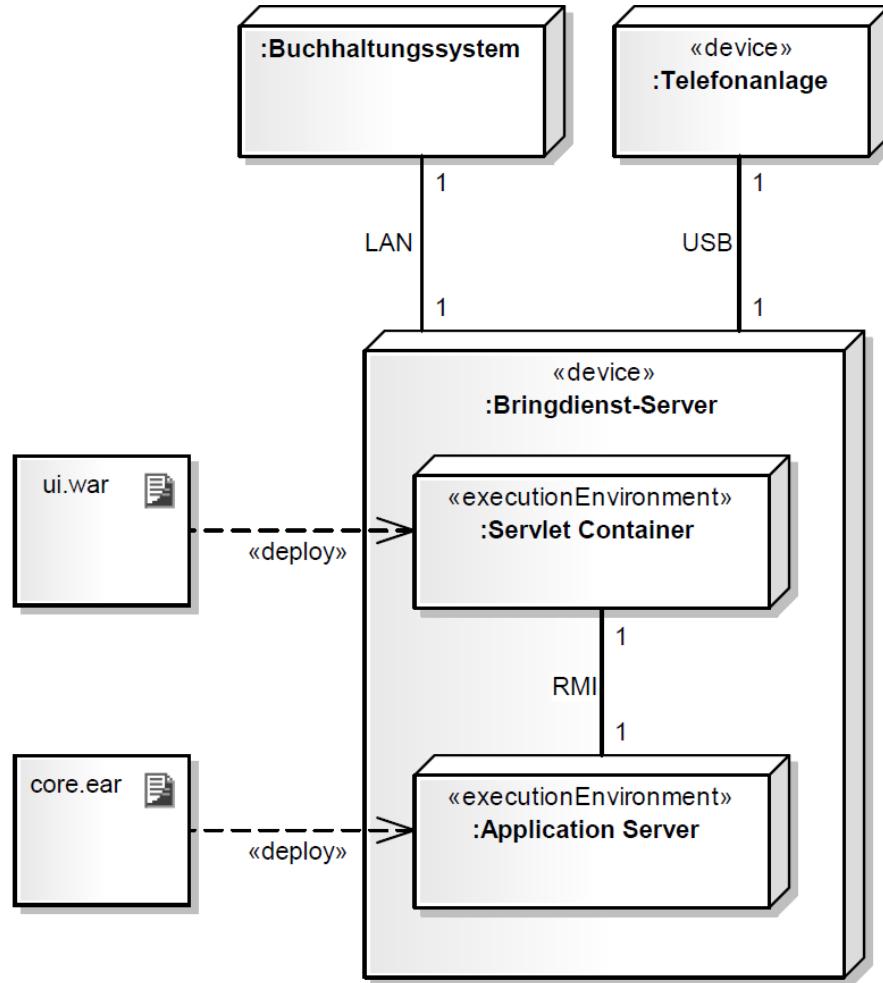
Strukturdiagramme: Einsatz- und Verteilungsdiagramm



Quelle: [http://www.uml-diagrams.org/deployment-diagrams-overview.html, Abruf 27.03.2015]

Strukturdiagramme: Einsatz- und Verteilungsdiagramm

- Beispiel



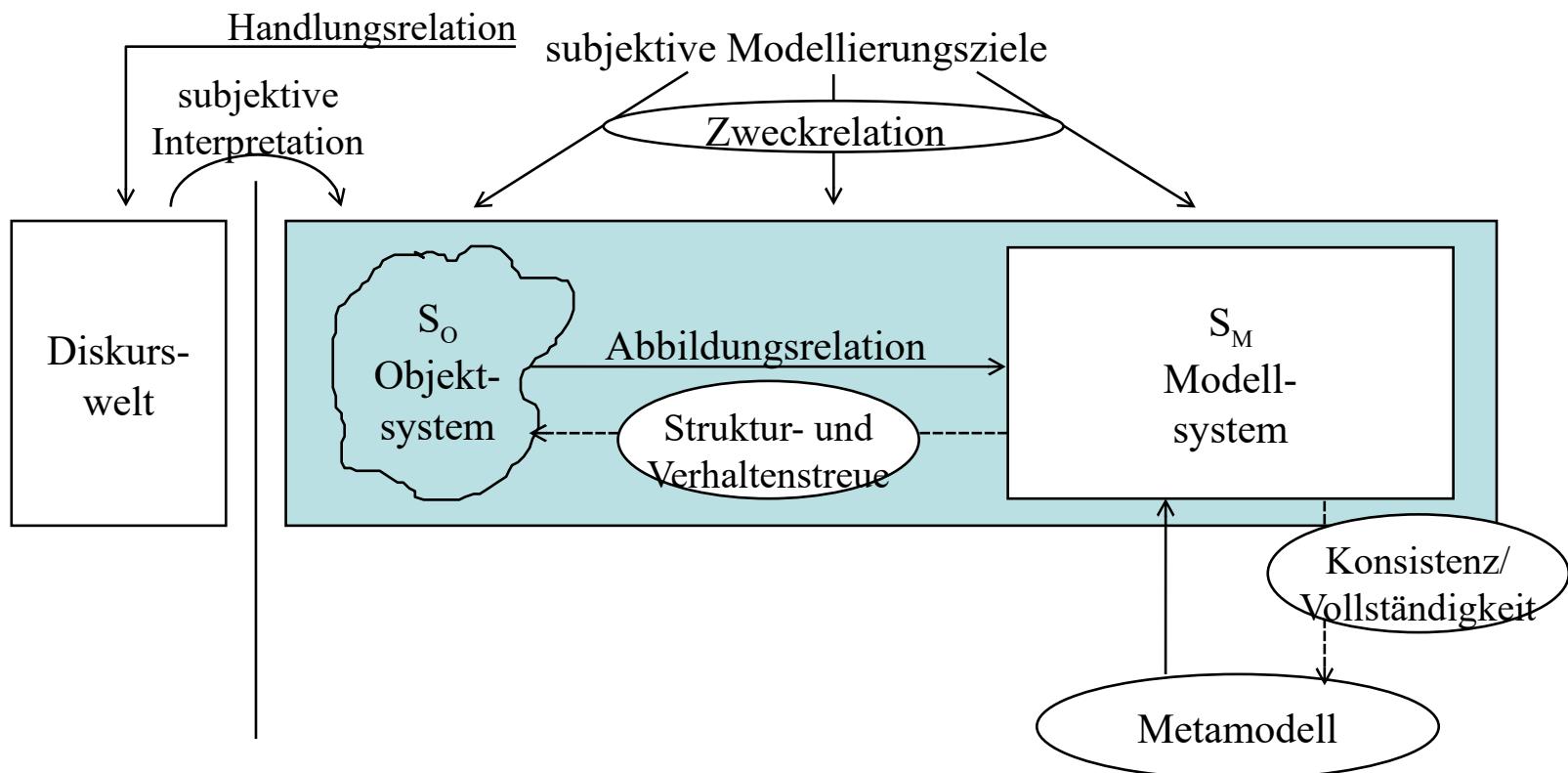
Einsatz- und Verteilungsdiagramm: Beschreibung eines Bringdiensts

Modelltheorie

Modellsysteme

- Die Abbildungsrelation setzt zwei Systeme miteinander in Beziehung
- Komponenten eines Modells
 - Objektsystem (S_O)
 - subjektive Interpretation eines zweckorientiert gewählten Realitätsausschnitts
 - relevantes Umweltystem
 - Modellsystem (S_M)
 - subjektives Abbild des Objektsystems
 - Formulierung mittels geeignetem Metamodell
- Beziehung, die S_O auf S_M abbildet
 - Elimination irrelevanter Sachverhalte.
 - Typisierung relevanter Sachverhalte.
 - Formal: Homomorphismus $r: S_O \rightarrow S_M$

Modellwelt



Modellierungsziele

- Erklärung komplexer oder komplizierter Phänomene
 - Modell als abstrahierende Rekonstruktion der Realität.
- Gestaltung
 - Aufzeigen von Optionen zur (Um-)Gestaltung der Realwelt.
 - Modell hat konstruktiven Charakter.

Informationsmodellierung

Merkmal	Ausprägung			
	Daten	Funktionen	Organisation	Prozesse
Beschreibungs-sicht	Objekte			
Beschreibungs-ebene	Fachkonzept		DV-Konzept	Implementierungs-konzept
Geltungs-anspruch	Istmodell		Sollmodell	Idealmodell
Inhaltliche Individualität	Unternehmens-modell		Referenzmodell	Mastermodell
Abstraktions-grad	Ausprägungs-ebene	Typebene	Metaebene	Meta-Meta-Ebene

Morphologischer Kasten: Informationsmodellierung

Beschreibungssicht

- Daten
 - passive Systemelemente, die durch Funktionen manipuliert werden
 - Träger von Begrifflichkeiten und Beziehungen zwischen diesen
- Funktionen
 - definieren Zustandsänderungen von Daten
 - $f: I \rightarrow O$

Merkmal	Ausprägung			
	Daten	Funktionen	Objekte	Organisation
Beschreibungssicht				Prozesse
Geltungs- anspruch	Istmodell		Sollmodell	Implementierungs- konzept
Inhaltliche Individualität	Unternehmens- modell		Referenzmodell	Mastmodell
Abstraktions- grad	Ausprägungs- ebene	Typebene	Metaebene	Meta-Meta- Ebene

Beschreibungssicht

- Objekte
 - Semantisch zusammenhängende Funktionen und Daten sind in Objekten gekapselt
- Organisation (hier: Aufbauorganisation)
 - Organisationseinheiten und ihre Beziehungen
- Prozesssicht
 - Modell des Systemverhaltens

Merksam	Ausprägung			
	Daten	Funktionen	Organisation	Prozesse
beschreibungs- sicht	Objekte			
Gefungs- anspruch	Fachkonzept	DV-Konzept	Implementierungs- konzept	
Inhaltliche Individualität	Istmodell	Softmodell	Idealmodell	
Abstraktions- grad	Unternehmens- modell	Referenzmodell	Mastermodell	
	Ausprägungs- ebene	Typebene	Metabene	Meta-Meta- Ebene

Prozess aus systemtheoretischer Sicht

- Prozess ist Folge von Systemzuständen
 - Beschreibung erfordert Heranziehung relevanter Systembestandteile und -ausprägungen.
 - Menge aller möglichen Prozesse beschreibt das statische Systemverhalten.

Merksam	Ausprägung			
	Daten	Funktionen	Organisation	Prozesse
Objekte				
beschreibungs- sicht				
beschreibungs- ebene	Istmodell	Fachkonzept	DV-Konzept	Implementierungs- konzept
Geltungs- anspruch	Istmodell	Sollmodell		Idealmodell
Inhaltliche Individualität	Unternehmens- modell	Referenzmodell		Mastermodell
Abstraktions- grad	Ausprägungs- ebene	Typebene	Metabene	Meta-Meta- Ebene

Zusammenfassung

- Daten
 - Was wird manipuliert?
- Funktionen
 - Wie wird manipuliert?
- Organisation
 - Wer manipuliert?
- Prozess
 - In welcher Reihenfolge wird manipuliert?

Merksatz	Ausprägung			
	Daten	Funktionen	Organisation	Prozesse
Beschreibungs-sicht	Objekte			
Beschreibungs-ebene	Fachkonzept	DV-Konzept	Implementierungs-konzept	
Geltungs-anspruch	Istmodell	Sollmodell		Idealmodell
Inhaltliche Individualität	Unternehmens-modell	Referenzmodell	Mastermodell	
Abstraktions-grad	Ausprägungs-ebene	Typebene	Metabene	Meta-Meta-Ebene

Beschreibungsebene

- Fachkonzept
 - (semi-)formale, implementierungsunabhängige Beschreibung einer betriebswirtschaftlichen Konzeption



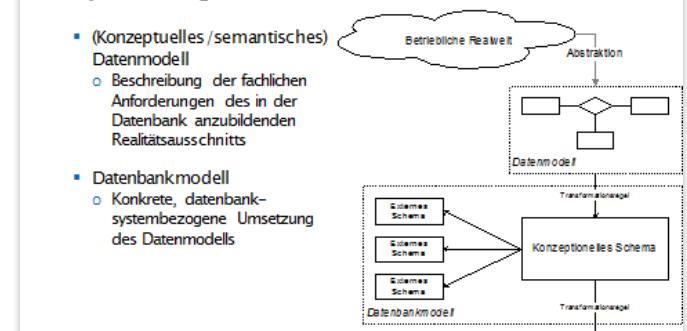
Implementierungsalternativen: Daten- und Datenbankmodelle

(Konzeptuelles /semantisches) Datenmodell

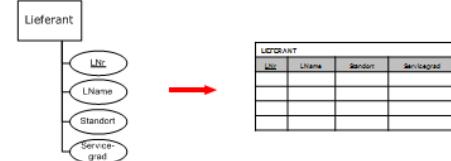
- Beschreibung der fachlichen Anforderungen des in der Datenbank anzubildenden Realitätsausschnitts

Datenbankmodell

- Konkrete, datenbanksystembezogene Umsetzung des Datenmodells



Implementierungsalternativen: Transformation des Entity-Typs



→ Weiterführung: Datenbank-Vorlesung

Sommersemester 2016

Modellierung | 21

- DV-Konzept
 - Anreicherung des Fachkonzepts um Anforderungen und Restriktionen der DV-technischen Umsetzung

- Implementierungskonzept
 - Formulierung der DV-technischen Umsetzung

Merkmal	Ausprägung			
	Daten	Funktionen	Objekte	Organisation
beschreibungs- sicht				Prozesse
beschreibungs- ebene	Fachkonzept	DV-Konzept	Implementierungs- konzept	
Geltungs- anspruch	Istmodell	Sollmodell		Idealmodell
Inhaltliche Individualität	Unternehmens- modell	Referenzmodell		Mastermodell
Abstraktions- grad	Ausprägungs- ebene	Typebene	Metabene	Meta-Meta- Ebene

Referenzmodelle

- Referenz = Empfehlung
- Charakteristisch für Referenzmodelle ist die Forderung nach einer gewissen Allgemeingültigkeit
- Dem Anspruch nach ist ein Referenzmodell ein Soll- oder Idealmodell für einen bestimmten Objektbereich,
 - z. B. ein Referenz-Prozessmodell für die Auftragsabwicklung in der Papier-Industrie

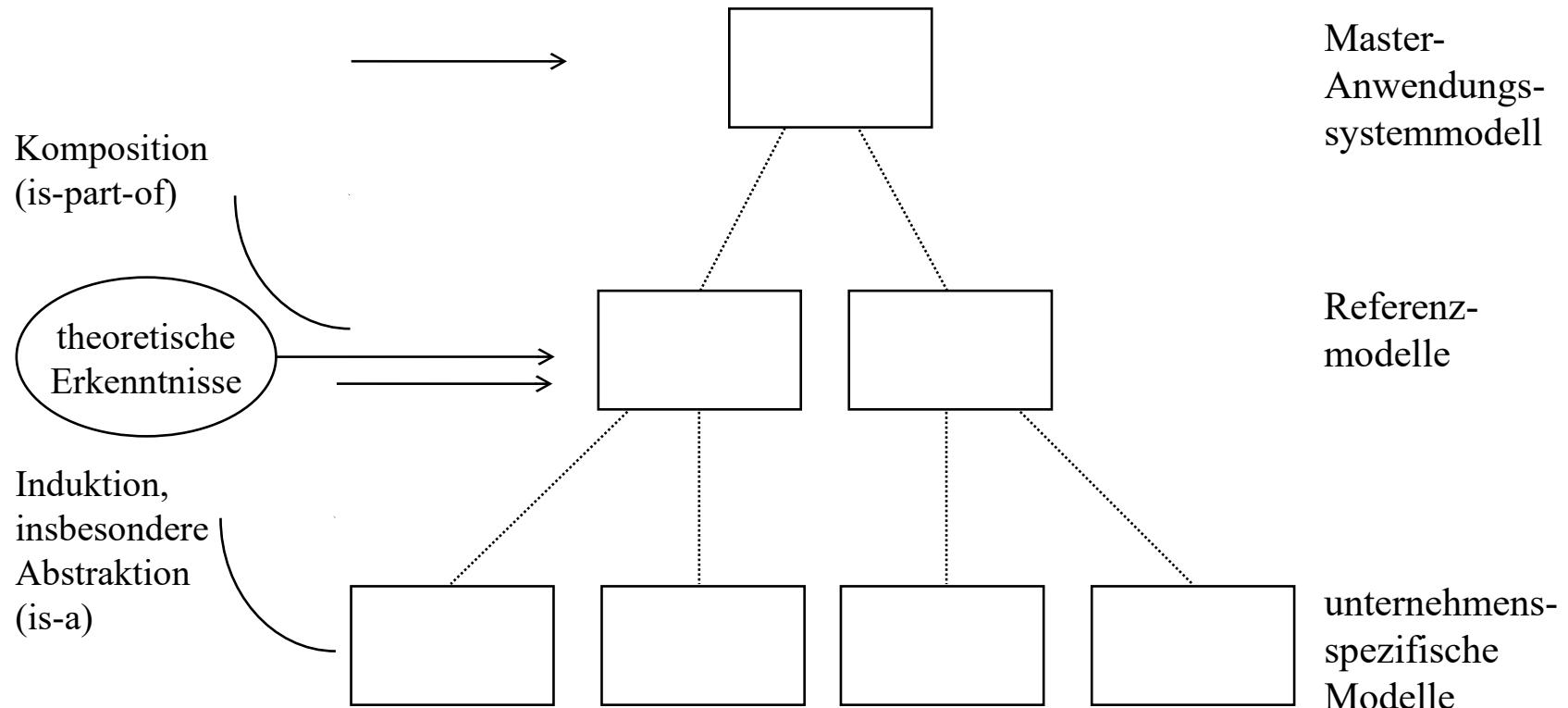
Merksam	Ausprägung			
	Daten	Funktionen	Organisation	Prozesse
beschreibungs- sicht	Objekte			
beschreibungs- ebene	Fachkonzept	DV-Konzept	Implementierungs- konzept	
Geltungs- anspruch	Istmodell	Sollmodell		Idealmodell
Inhaltliche Individualität	Unternehmens- modell	Referenzmodell		Mastermodell
Abstraktions- grad	Ausprägungs- ebene	Typebene	Metabene	Meta-Meta- Ebene

Mastermodelle

- Mastermodelle
 - Entstehen durch Vereinigung (Komposition) mehrerer Referenzmodelle.
 - Sie entsprechen strenggenommen der Vereinigungsmenge verschiedener Referenzmodelle und sind demzufolge „allgemeingültiger“ als Referenzmodelle.

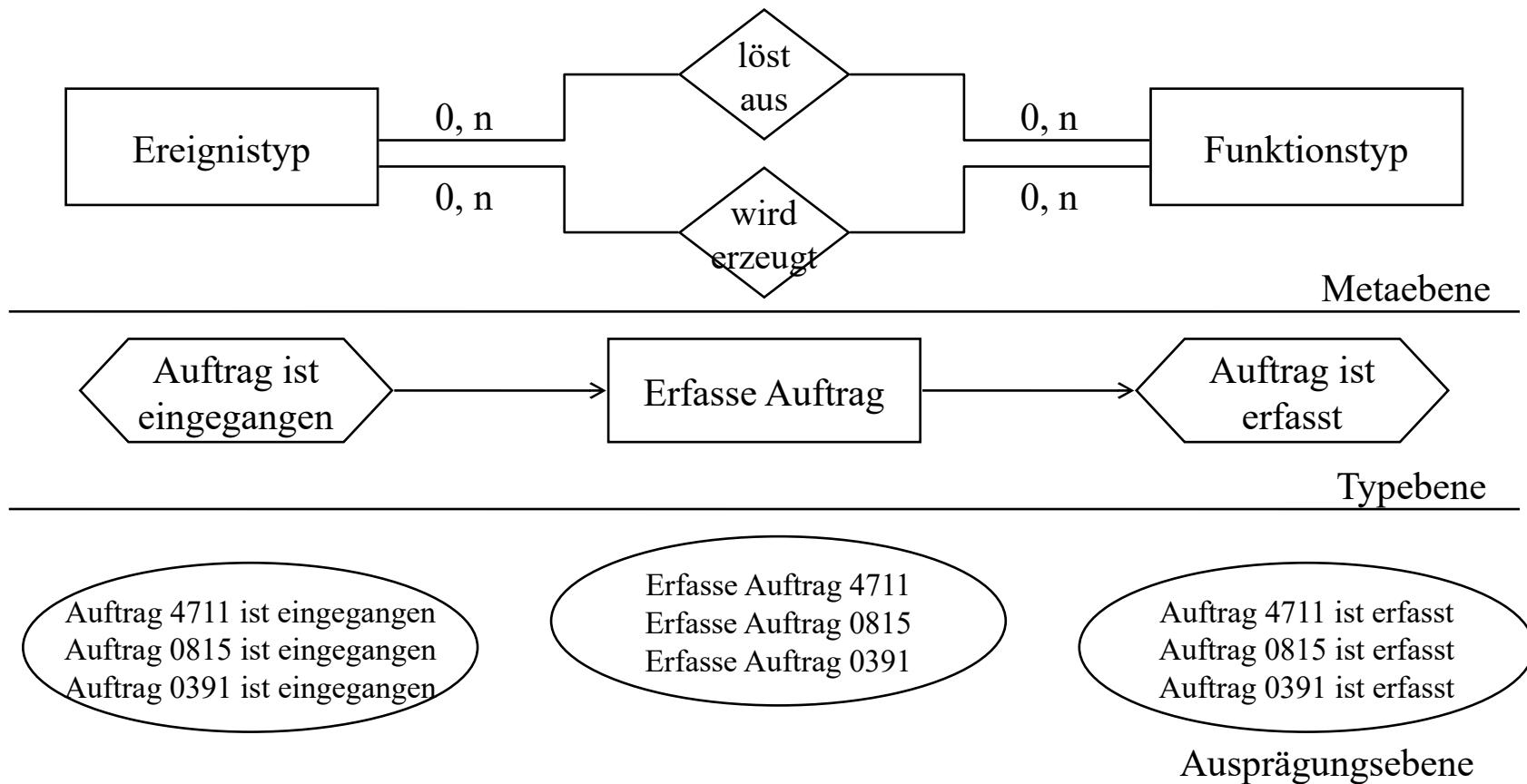
Merksam	Ausprägung			
	Daten	Funktionen	Organisation	Prozesse
beschreibungs- sicht	Objekte			
beschreibungs- ebene	Fachkonzept	DV-Konzept	Implementierungs- konzept	
Geltungs- anspruch	Istmodell	Sollmodell		Idealmodell
Inhaltliche Individualität	Unternehmens- modell	Referenzmodell	Mastermodell	
Abstraktions- grad	Ausprägungs- ebene	Typebene	Metabene	Meta-Meta- Ebene

Mastermodelle



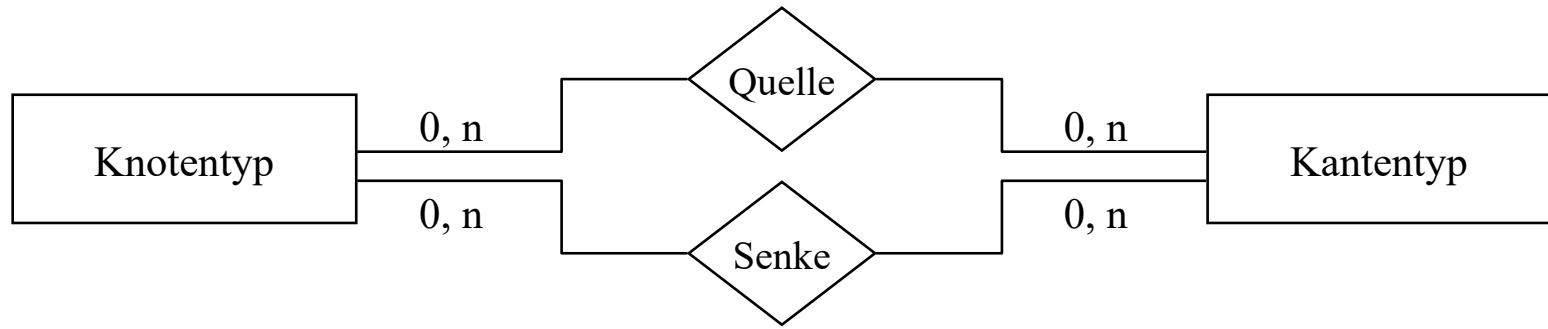
Merkmal	Ausprägung			
	Daten	Funktionen	Organisation	Prozesse
Beschreibungs- sicht		Objekte		
Beschreibungs- ebene		Fachkonzept	DV-Konzept	Implementierungs- konzept
Geltungs- anspruch	Istmodell	Sollmodell		Idealmodell
Inhaltliche Individualität	Unternehmens- modell	Referenzmodell		Mastermodell
Abstraktions- grad	Ausprägungs- ebene	Typebene	Metabene	Meta-Meta- Ebene

Abstraktionsgrad

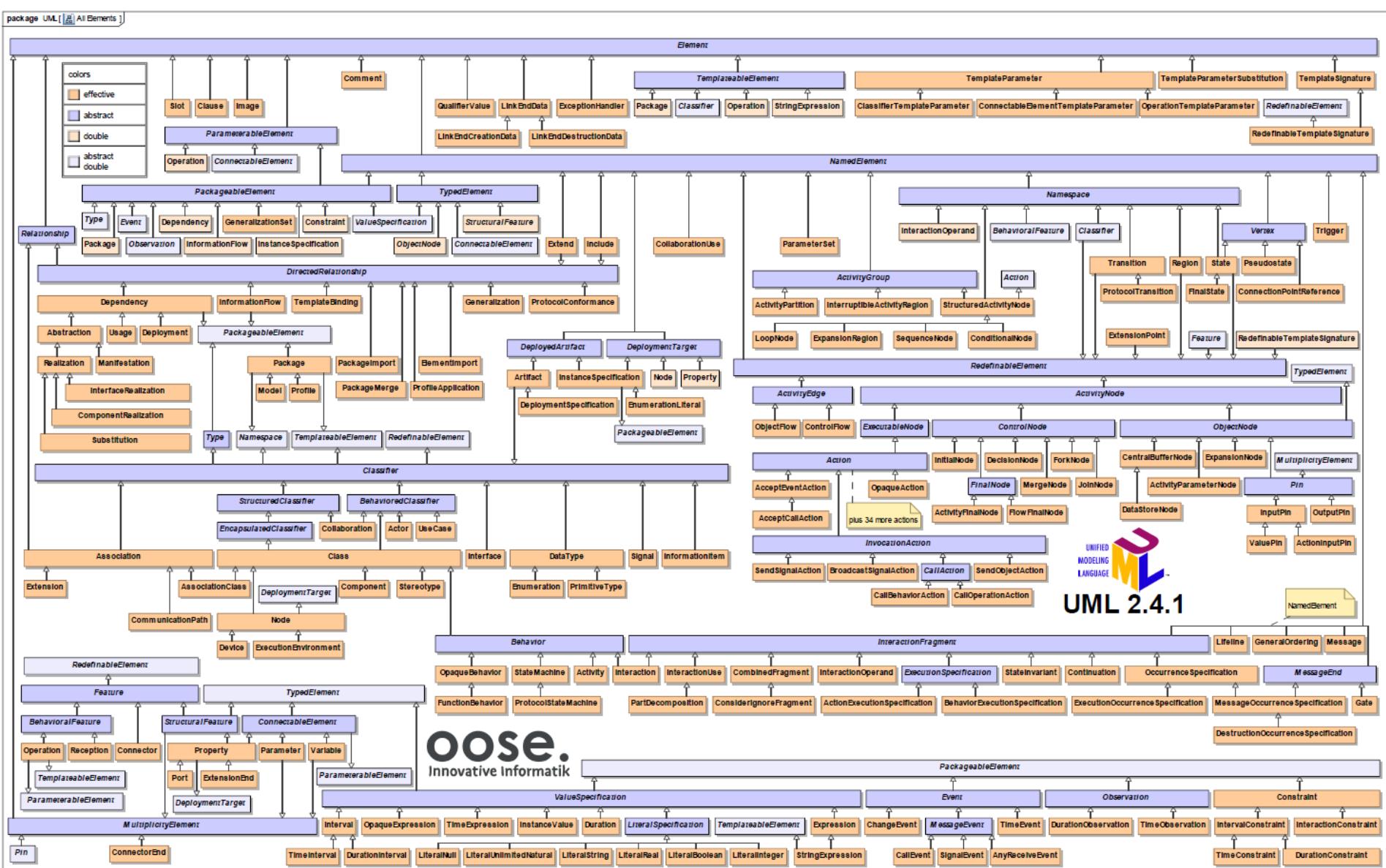


Merksam	Ausprägung			
	Daten	Funktionen	Organisation	Prozesse
beschreibungs- sicht	Objekte			
Geltungs- anspruch	Fachkonzept	DV-Konzept	Implementierungs- konzept	
Inhaltliche Individualität	Istmodell	Sollmodell	Idealmodell	
Abstraktions- grad	Unternehmens- modell	Referenzmodell	Mastermodell	
Ausprägungs- ebene	Typebene	Metaebene	Meta-Meta- Ebene	

Meta-Meta-Ebene



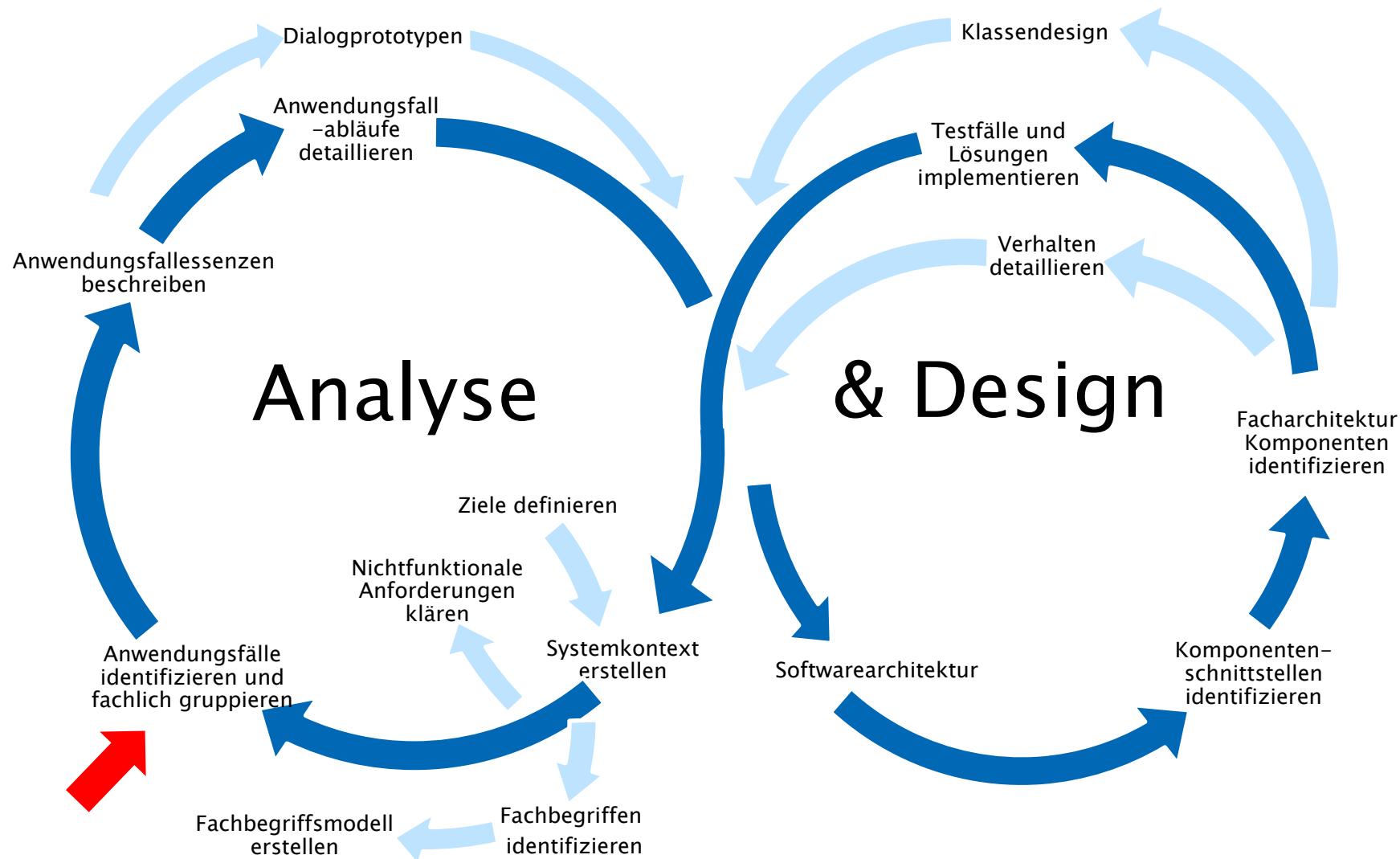
Merksam	Ausprägung			
	Daten	Funktionen	Organisation	Prozesse
beschreibungs- sicht	Objekte			
beschreibungs- ebene	Fachkonzept	DV-Konzept	Implementierungs- konzept	
Geltungs- anspruch	Istmodell	Sollmodell	Idealmodell	
Inhaltliche Individualität	Unternehmens- modell	Referenzmodell	Mastermodell	
Abstraktions- grad	Ausprägungs- ebene	Typebene	Metabene	Meta-Meta- Ebene



 Note: This diagram contains all UML metadlasses (with the exception of 34 actions) and all specialization relationships. Element in a lighter color are doubles of elements shown elsewhere (in order to avoid line crossings). Each element is shown only once in darker color with all its generalizations and specializations. This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Germany License](#). © 2013 oose Innovative Informatik GmbH | www.oose.de/metamodell/uml/

Anwendungsfalldiagramm

UML-Methodikleitfaden für Analyse und Design



Quelle: Oestreich, B., Scheithauer, A.: Analyse und Design mit der UML 2.5: Objektorientierte Softwareentwicklung. Oldenbourg Wissenschaftsverlag, München (2012) (Poster zum Buch)

Verhaltensdiagramme: Anwendungsfalldiagramm

- Abbildung von Anwendungsfällen (Use Cases), Akteuren und deren Beziehungen in einem System
- Definition (funktionaler) Anforderungen an ein System aus unterschiedlichen Benutzerperspektiven
- Verschiedene Abstraktionsebenen (der Funktionen) möglich

Verhaltensdiagramme: Anwendungsfalldiagramm

- Notation

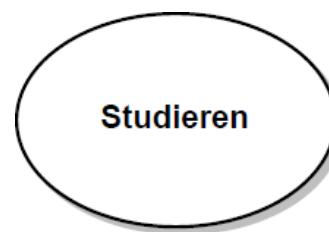
- **Anwendungsfall**

- Geschäftsanwendungsfall: Abstrakter, geschäftlicher Ablauf ohne Berücksichtigung systemtechnischer Umsetzung



Geschäftsanwendungsfall

- Systemanwendungsfall: Konkrete Beschreibung einer Interaktion mit einem System



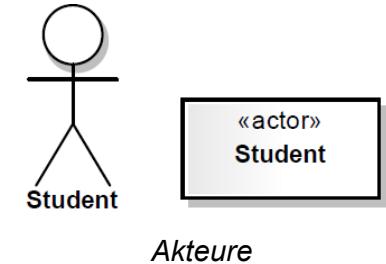
Systemanwendungsfall

Verhaltensdiagramme: Anwendungsfalldiagramm

■ Notation

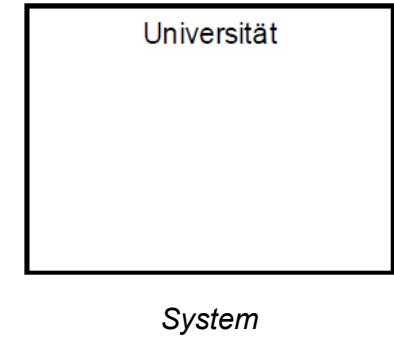
○ Akteur

- Interagiert als eine Rolle von außerhalb mit Anwendungsfällen des Systems
- Personen, (Externe) Systeme, Ereignisse, ...



○ System

- Abbildung von Systemgrenzen mehrerer involvierter Systeme

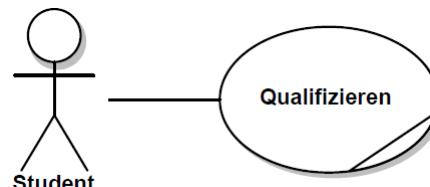


Verhaltensdiagramme: Anwendungsfalldiagramm

- Notation

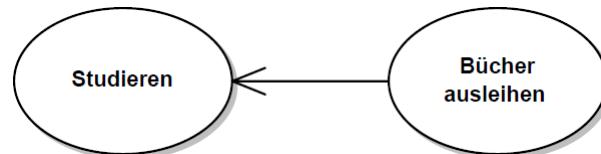
- **Assoziation**

- Ungerichtete Assoziation: Verbindung von Anwendungsfällen mit Akteuren bzw. anderen Anwendungsfällen



Ungerichtete Assoziation

- Gerichtete Assoziation: Ein- und Ausgabefluss zwischen Anwendungsfällen



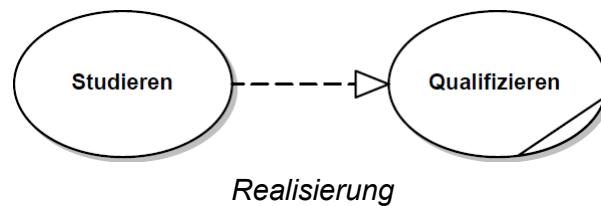
Gerichtete Assoziation

Verhaltensdiagramme: Anwendungsfalldiagramm

- Notation

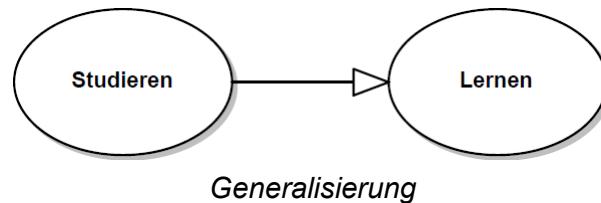
- **Realisierung**

- Umsetzung eines (abstrakteren) Anwendungsfalls durch einen anderen



- **Generalisierung**

- Akteure und Anwendungsfälle
 - Kommunikationsbeziehungen zwischen Akteuren und Anwendungsfällen werden weiter vererbt

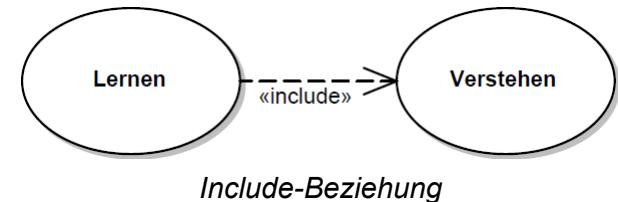


Verhaltensdiagramme: Anwendungsfalldiagramm

■ Notation

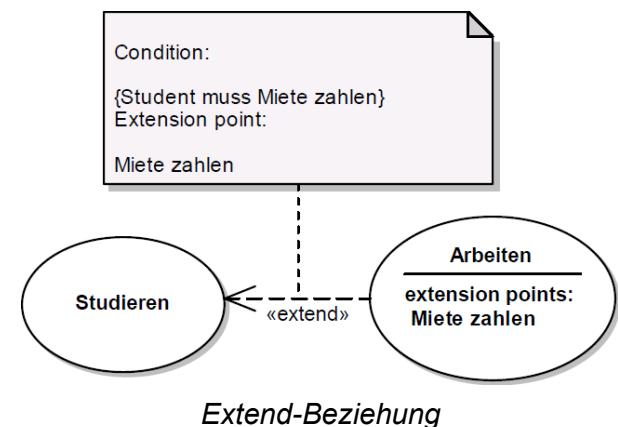
○ Include–Beziehung

- Import des Verhaltens eines Anwendungsfalls durch einen anderen
- Verbot zyklischer Abhängigkeiten

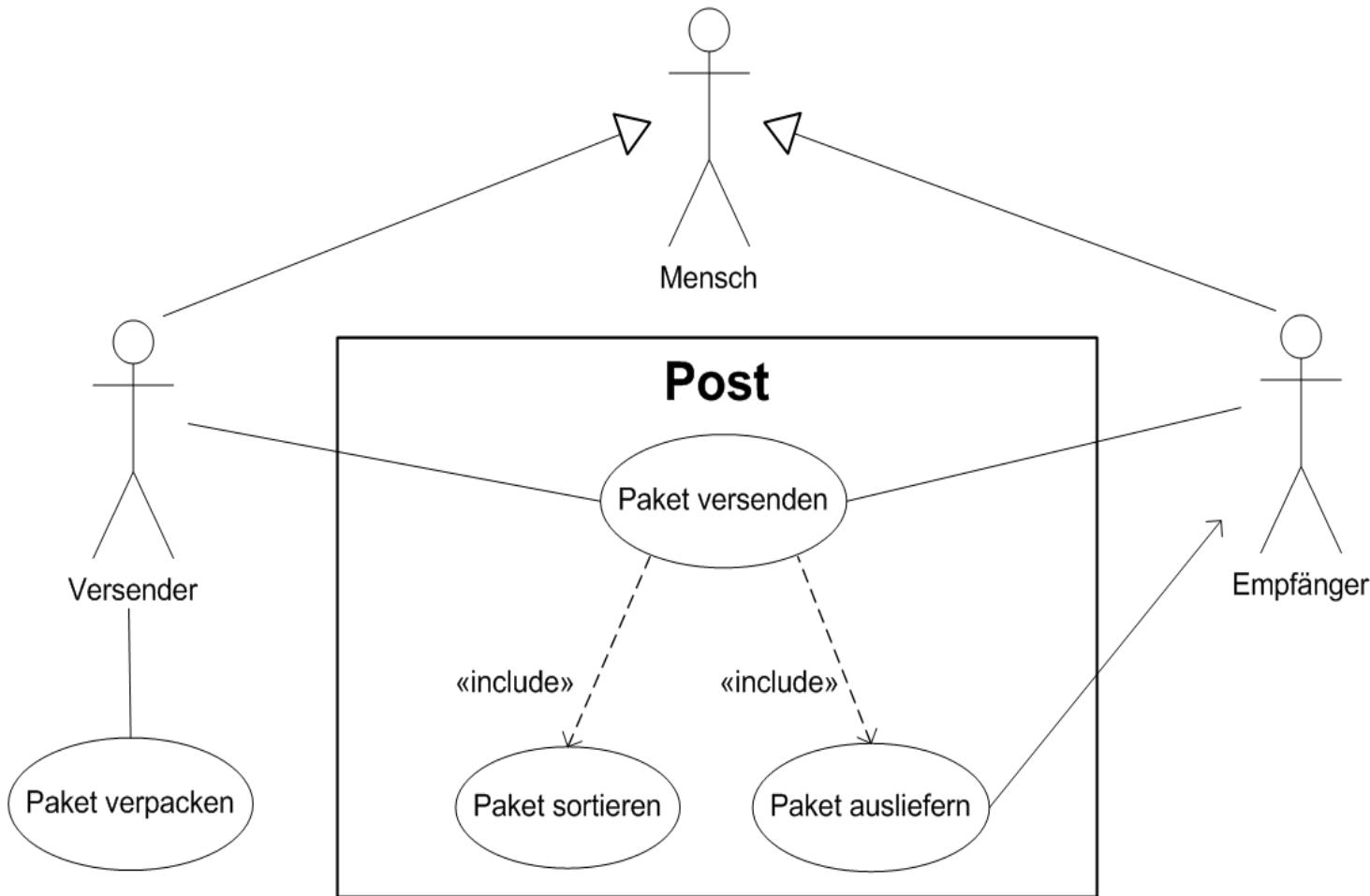


○ Extend–Beziehung

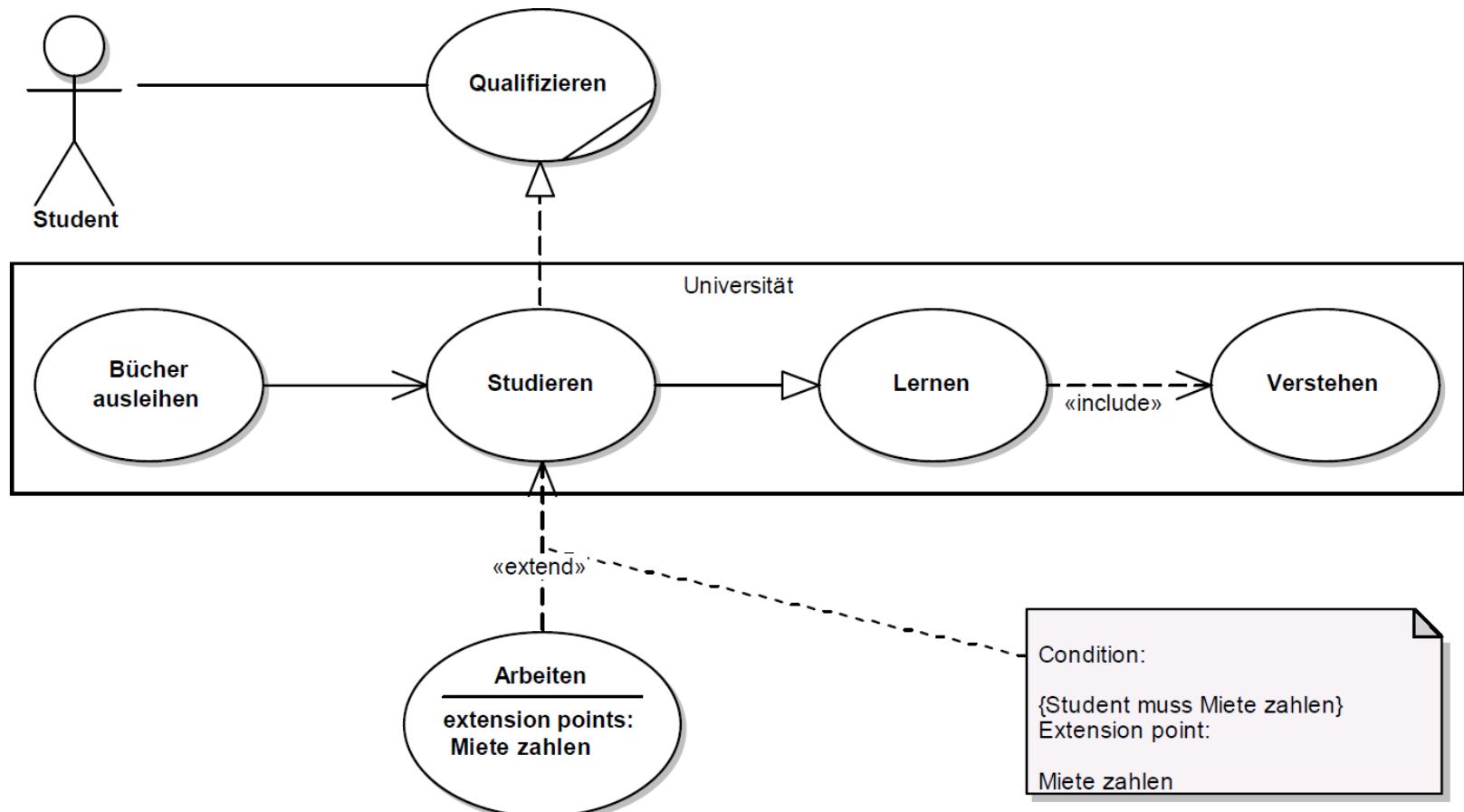
- Das Verhalten eines Anwendungsfalls kann durch einen anderen erweitert werden
- Erweiterung an Bedingung geknüpft



Anwendungsfalldiagramm – Beispiel 1



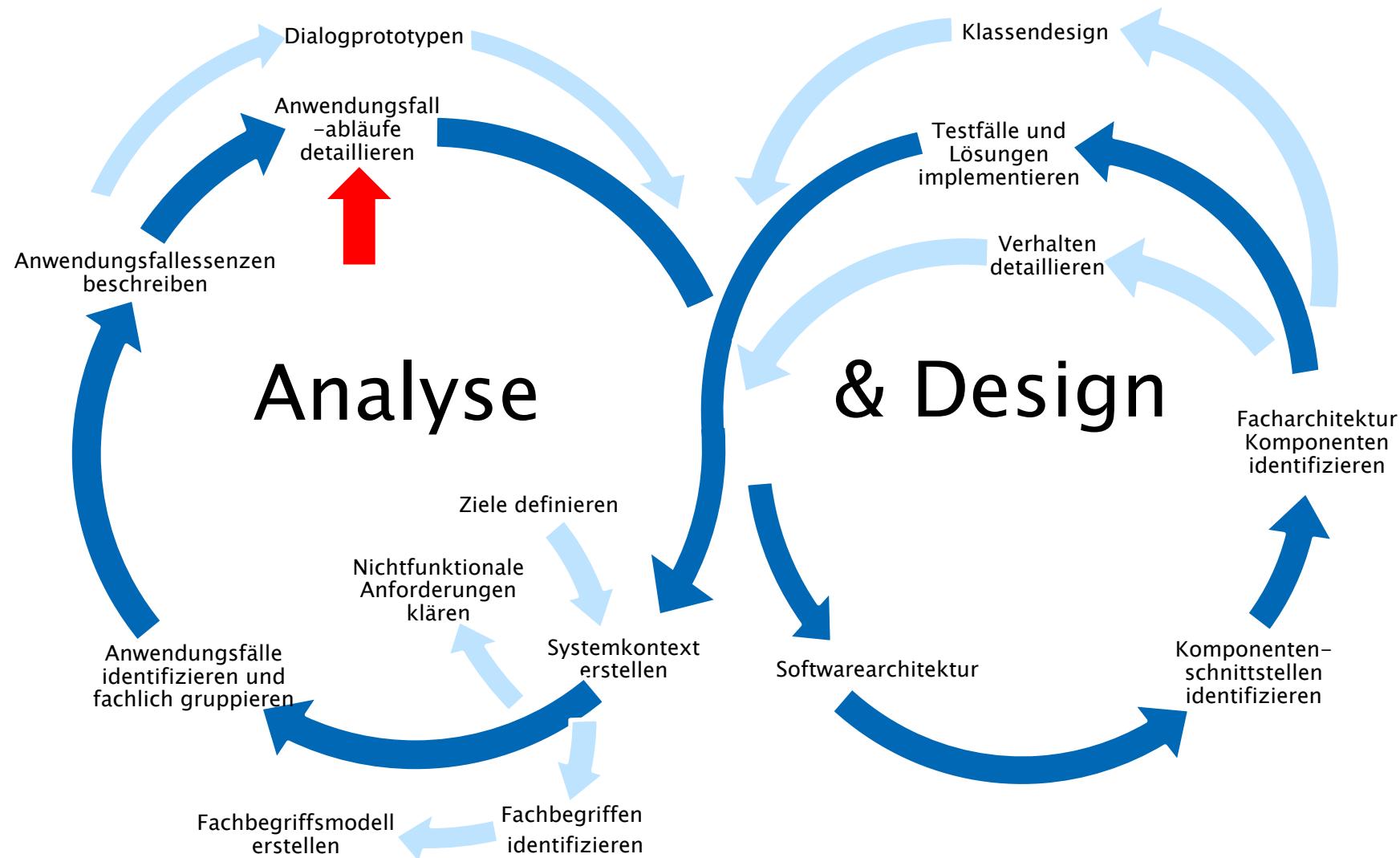
Anwendungsfalldiagramm – Beispiel 2



Anwendungsfalldiagramm: Beschreibung des Anwendungsfalls „Qualifizieren“

Aktivitätsdiagramm

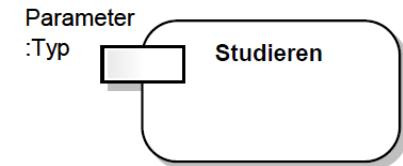
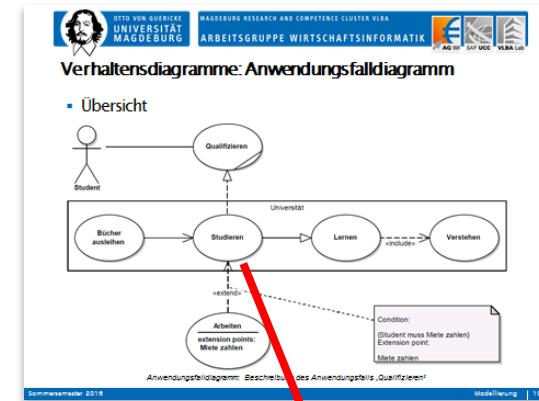
UML-Methodikleitfaden für Analyse und Design



Quelle: Oestereich, B., Scheithauer, A.: Analyse und Design mit der UML 2.5: Objektorientierte Softwareentwicklung. Oldenbourg Wissenschaftsverlag, München (2012) (Poster zum Buch)

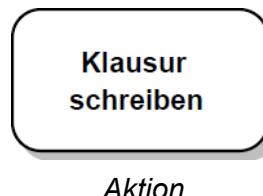
Verhaltensdiagramme: Aktivitätsdiagramm

- Beschreibung von Handlungen (komplexe Prozesse) eines Systems als Graph mit gerichteten Kanten
- Sequenzielle, parallele, alternative und iterative Aktionen möglich
- Notation
 - **Aktivität**
 - In sich abgeschlossener, komplexer Prozess
 - Abfolge von einzelnen atomaren Aktionen
 - Können verschachtelt werden
 - Parameter möglich
 - Verlinkungssymbol: Aktivitätsdiagramm hinterlegt

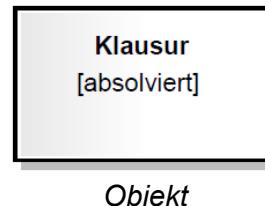


Verhaltensdiagramme: Aktivitätsdiagramm

- Notation
 - **Aktion**
 - Atomarer Vorgang innerhalb einer Aktivität

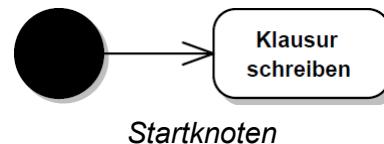


- **Objekt**
 - Während des Ablaufs eines Prozesses erzeugte Instanz einer Klasse
 - Optional: Zustand des Objekts in eckigen Klammern hinter den Objektnamen

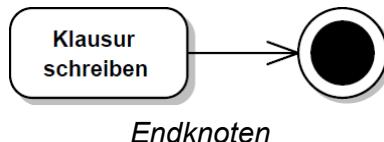


Verhaltensdiagramme: Aktivitätsdiagramm

- Notation von Kontrollknoten
 - **Startknoten**
 - Startpunkt einer Aktivität
 - Beliebig viele Startknoten und weiterführende Kanten möglich

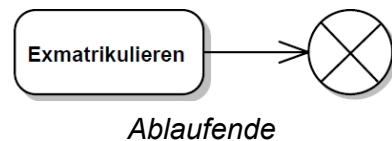


- **Endknoten**
 - Endpunkt einer Aktivität
 - Beliebig viele Endknoten möglich

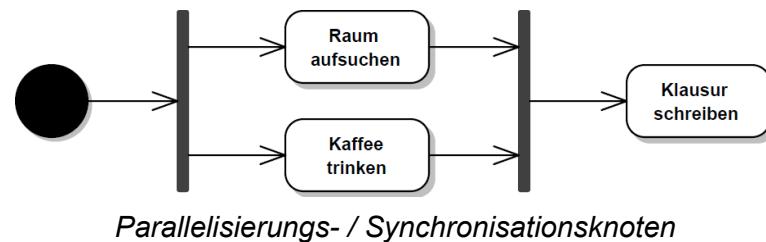


Verhaltensdiagramme: Aktivitätsdiagramm

- Notation von Kontrollknoten
 - **Ablaufende**
 - Endpunkt eines einzelnen Objekt- / Kontrollflusses

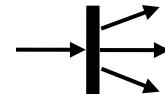


- **Parallelisierungs- / Synchronisationsknoten**
 - Beschreibung paralleler Abläufe
 - Aufteilung bzw. Zusammenführung mehrerer Abläufe

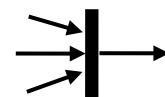


Aktivitätsdiagramm (Notation)

- Parallelisierungsknoten (Kontrollelement):
 - Ein Ablauf wird in mehrere parallele Abläufe geteilt (und)



- Synchronisationsknoten (Kontrollelement):
 - Gegenstück zum Parallelisierungsknoten (und)
 - Mehrere Abläufe werden zu einem zusammengeführt

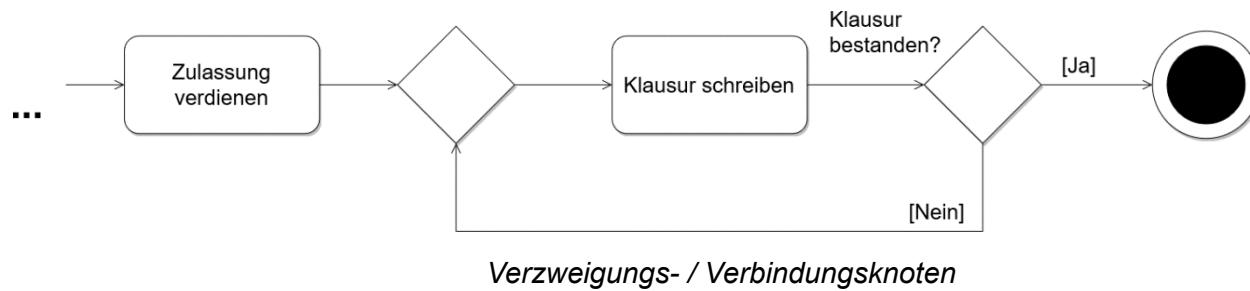


- Kontrollfluss (Kante):
 - Kante zwischen 2 Aktionen oder zwischen einer Aktion und einem Kontrollelement
 - Können mit Bedingungen versehen werden



Verhaltensdiagramme: Aktivitätsdiagramm

- Notation von Kontrollknoten
 - Verzweigungs- / Verbindungsknoten
 - Abläufe können in Varianten aufgespalten bzw. zusammengeführt werden
 - Bedingung für verschiedene Abläufe wird am Knoten notiert
 - Möglicher Zustand für einen Ablauf wird in eckigen Klammern jeweils an der Kante notiert
 - Aufspaltung bzw. Zusammenführung mehrerer Varianten



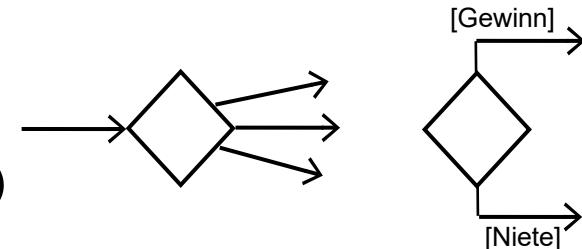
Verzweigungs- / Verbindungsknoten

Aktivitätsdiagramm (Notation)

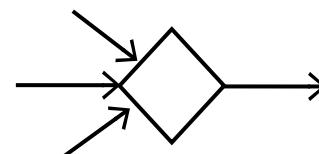
- Endknoten (Kontrollelement):
 - Für Aktivitäten beendet die gesamte Aktivität
 - Für Kontrollflüsse beendet nur einzelne Abläufe
 - Beliebig viele Endknoten pro Aktivität



- Verzweigungsknoten (Kontrollelement):
 - Spaltet Kante in mehrere Varianten auf
 - Ablauf ist von Bedingungen abhängig (oder)



- Verbindungsknoten (Kontrollelement):
 - Gegenstück zum Verzweigungsknoten
 - Führt Kanten wieder zusammen (oder)

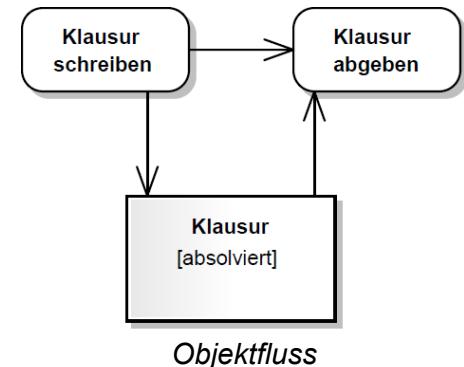


Verhaltensdiagramme: Aktivitätsdiagramm

- Notation

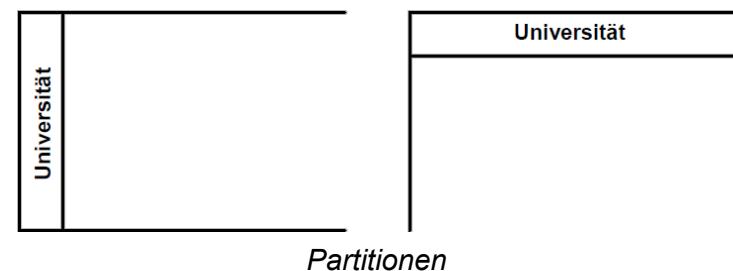
- **Objektfluss**

- Instanziierung eines Objekts ausgehend von einem speziellen Ablauf
 - Zuführung eines Objekts zu einem speziellen Ablauf



- **Partition**

- Bereich mit gemeinsamen Verantwortlichkeiten und Eigenschaften

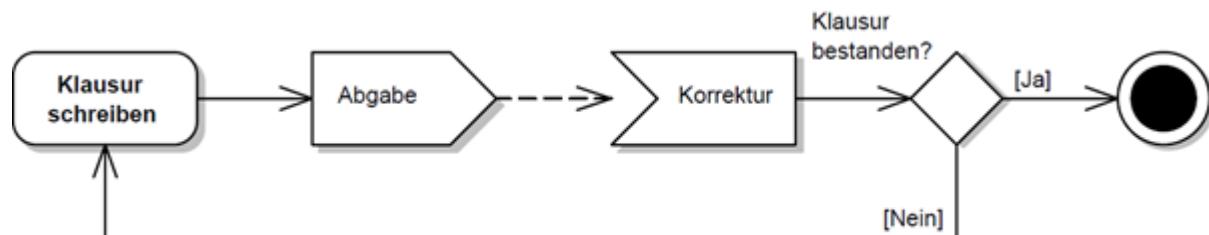


Verhaltensdiagramme: Aktivitätsdiagramm

- Notation

- **Signal senden / empfangen**

- Abbildung asynchroner Abläufe
 - Unterbrechung bzw. Synchronisation nebenläufiger Abläufe möglich
 - Gesendetes Signal stellt Ereignis dar, über das während eines Kontrollflusses benachrichtigt wird
 - Empfangenes Signal stellt Ereignis dar, welches einen Kontroll- bzw. Objektfluss auslöst



Signal übertragen

Verhaltensdiagramme: Aktivitätsdiagramm – Beispiel

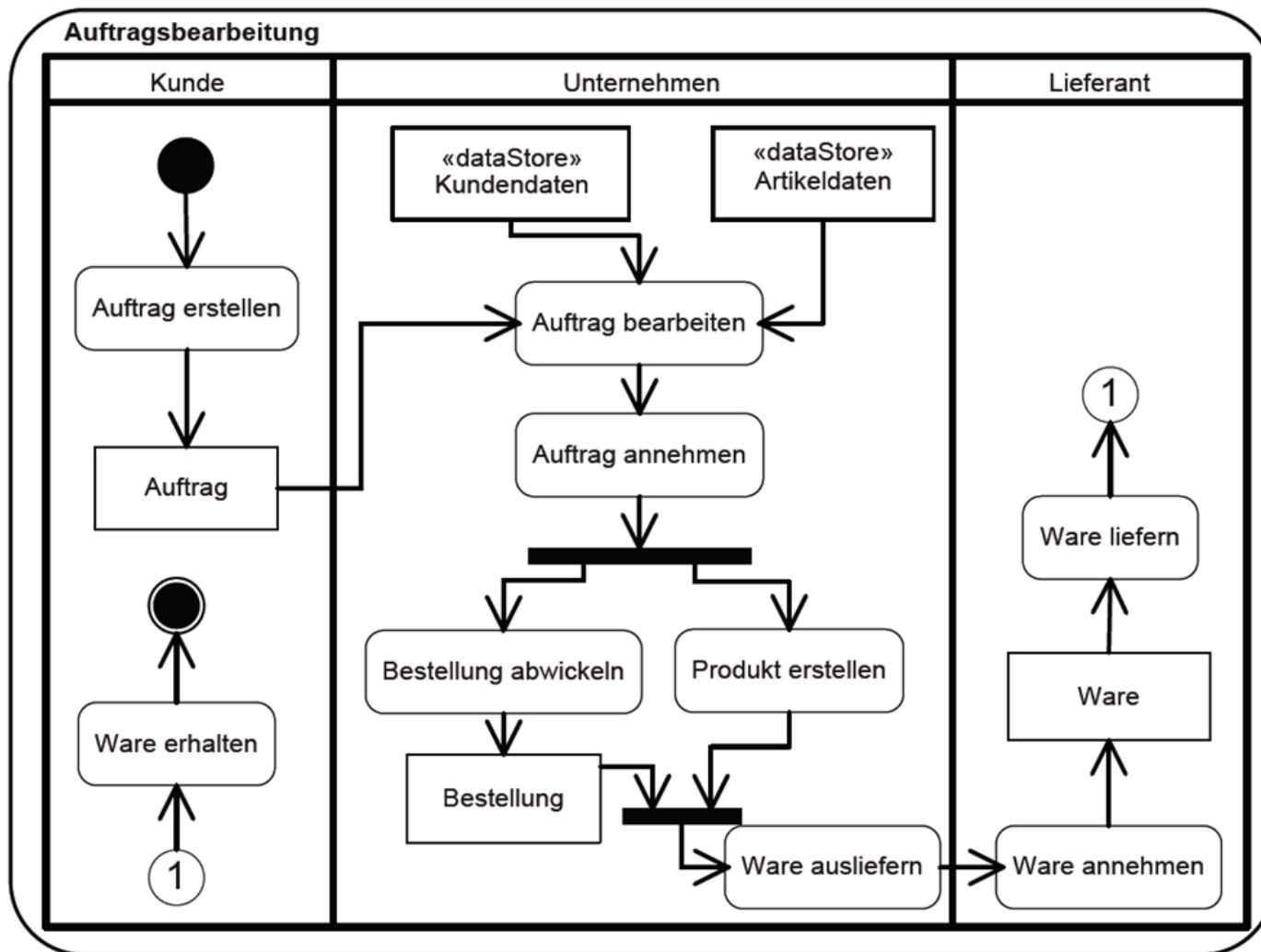


ABBILDUNG 13.10 Vereinfachte Darstellung einer Auftragsbearbeitung

Quelle: Rupp, C. et al. (2005): UML 2 glasklar: Praxiswissen für die UML-Modellierung und –Zertifizierung. München – Wien.

Verhaltensdiagramme: Aktivitätsdiagramm – Beispiel 2

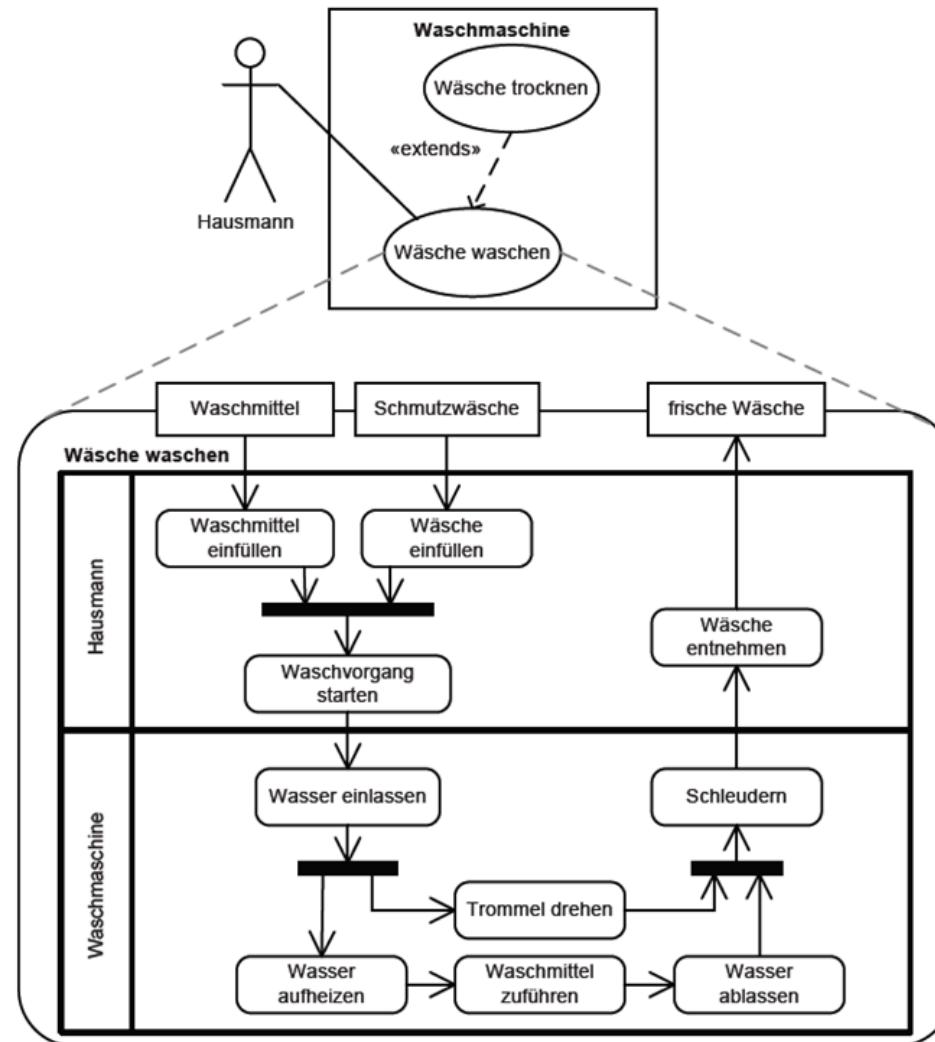
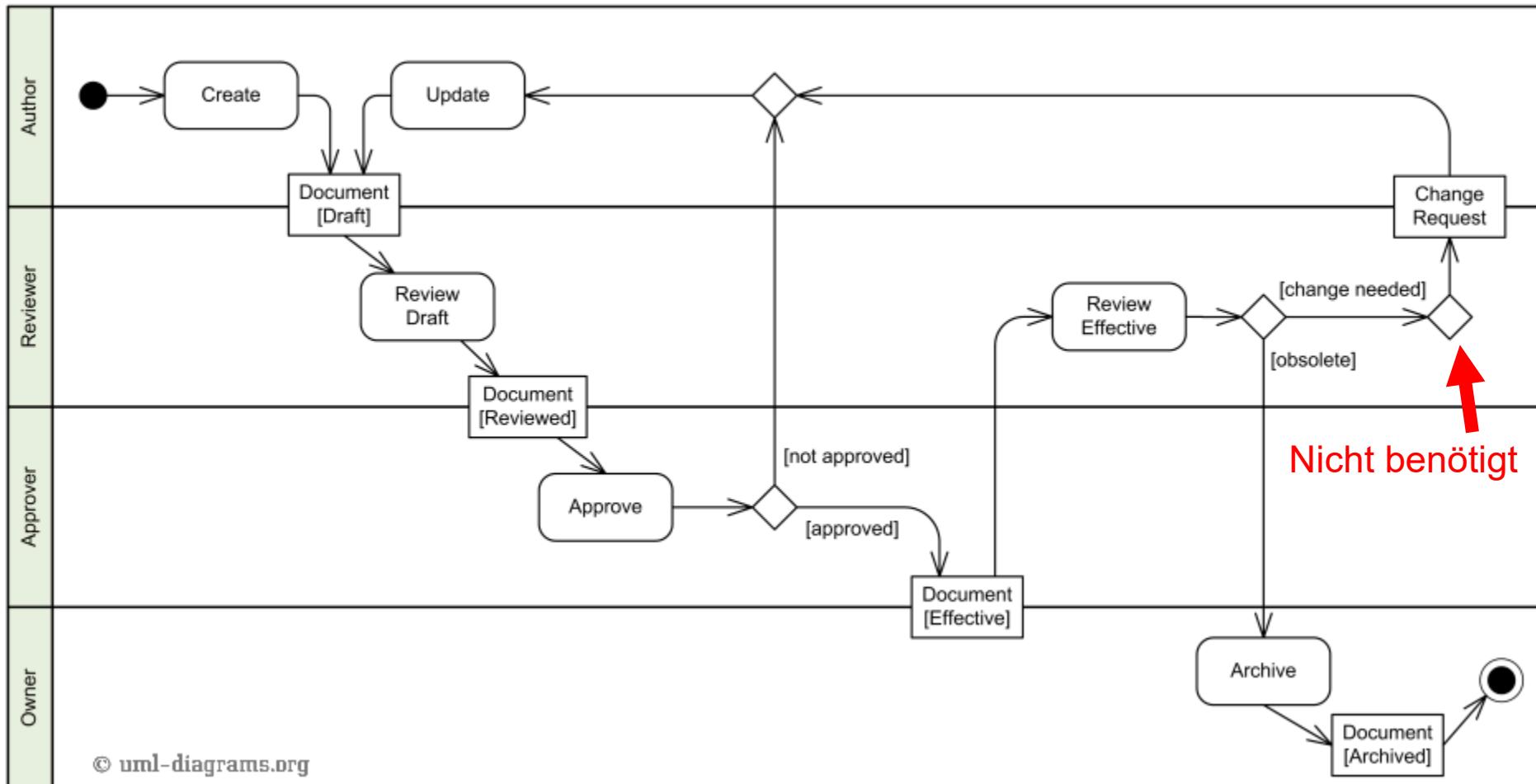


ABBILDUNG 13.11 Der Use-Case Wäsche waschen, dargestellt in einem Aktivitätsdiagramm.
Einblicke in das Leben eines SOPHISTischen Hausmanns.

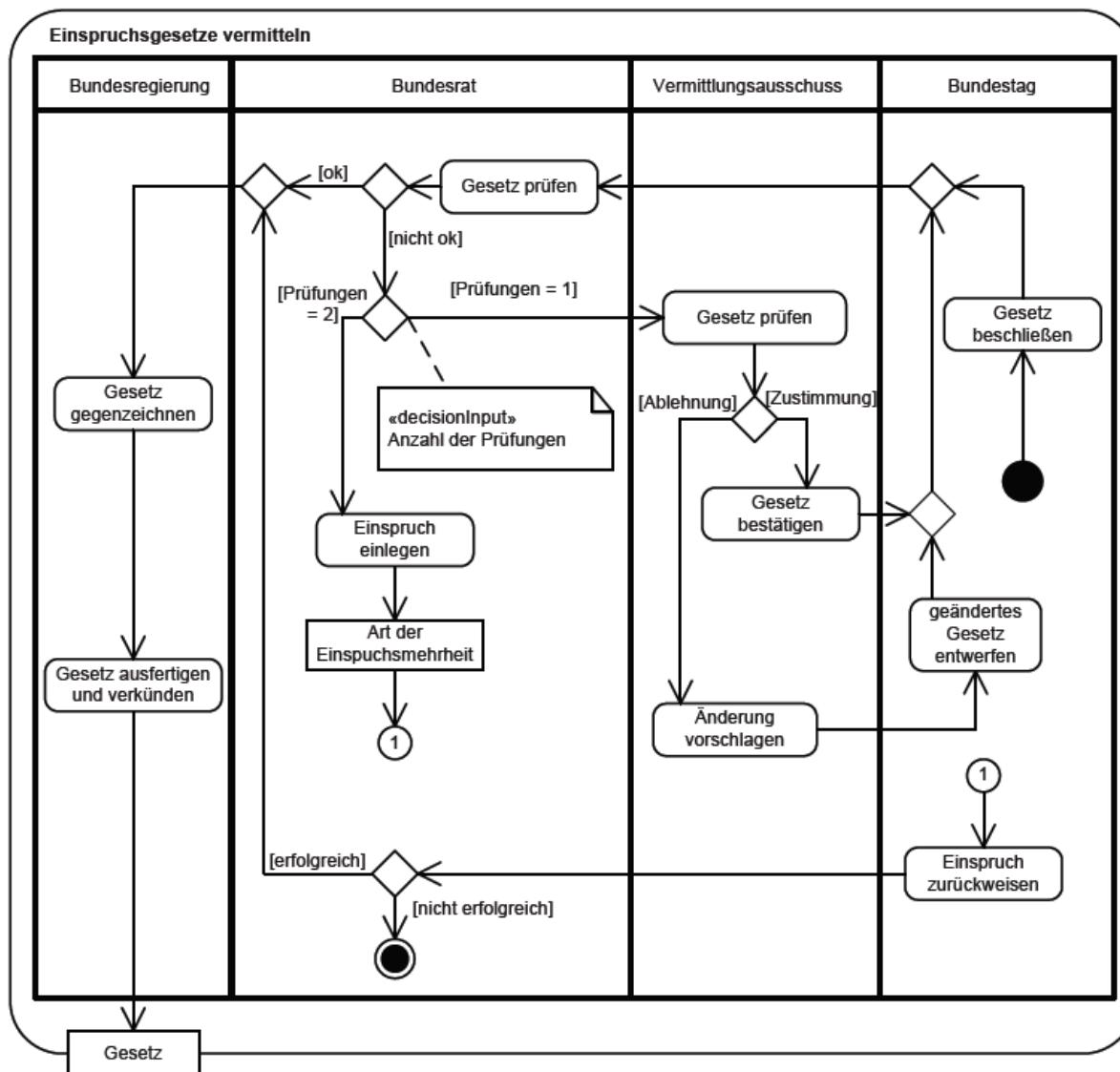
Quelle: Rupp, C. et al. (2005): UML 2 glasklar: Praxiswissen für die UML-Modellierung und –Zertifizierung. München – Wien.

Verhaltensdiagramme: Aktivitätsdiagramm – Beispiel 3



Quelle: [<http://www.uml-diagrams.org/document-management-uml-activity-diagram-example.html>, Abruf 05.05.2015]

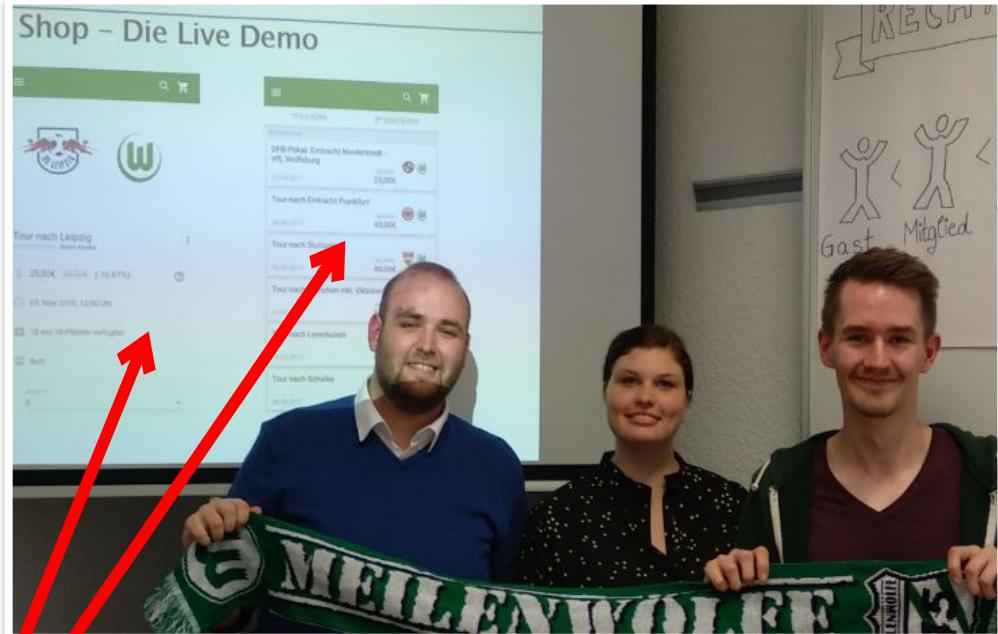
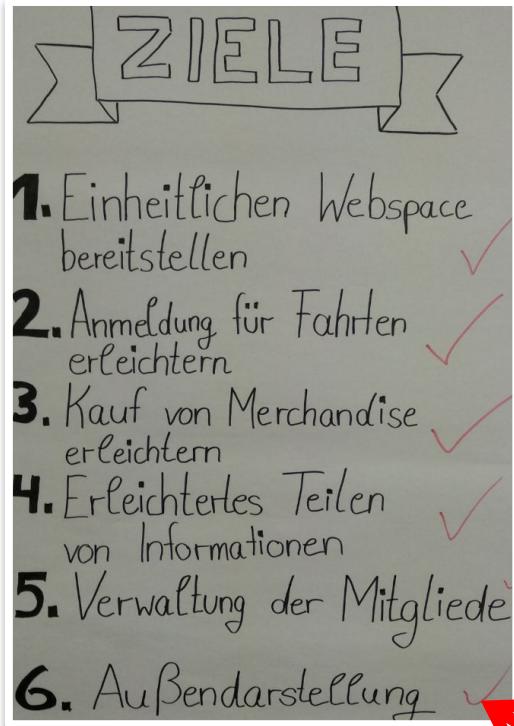
Verhaltensdiagramme: Aktivitätsdiagramm – Beispiel 4



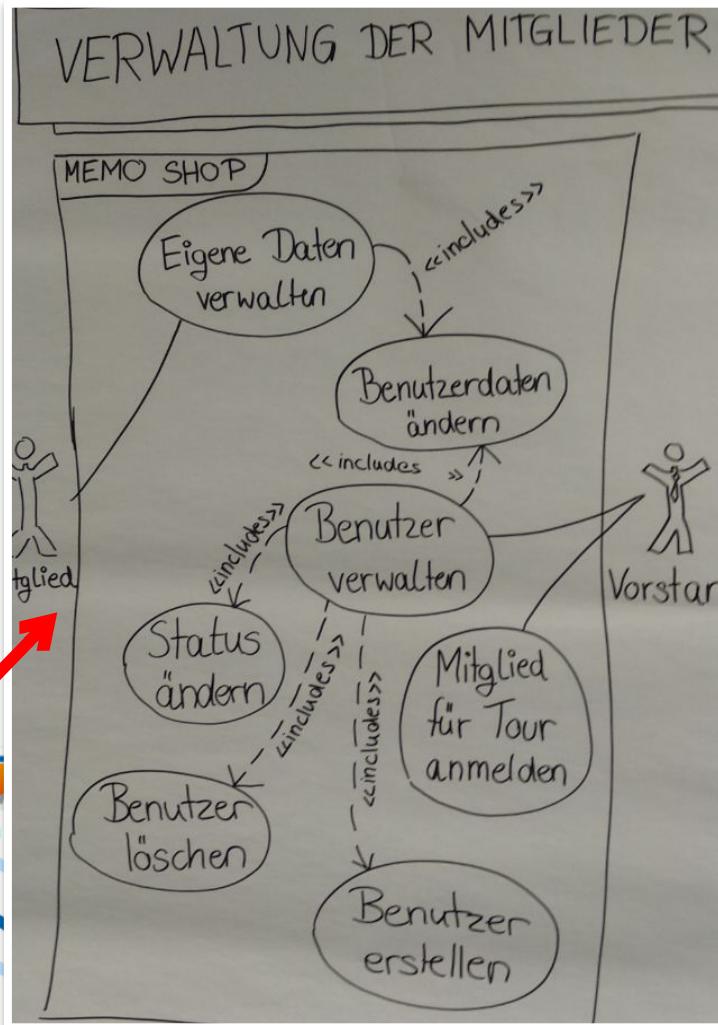
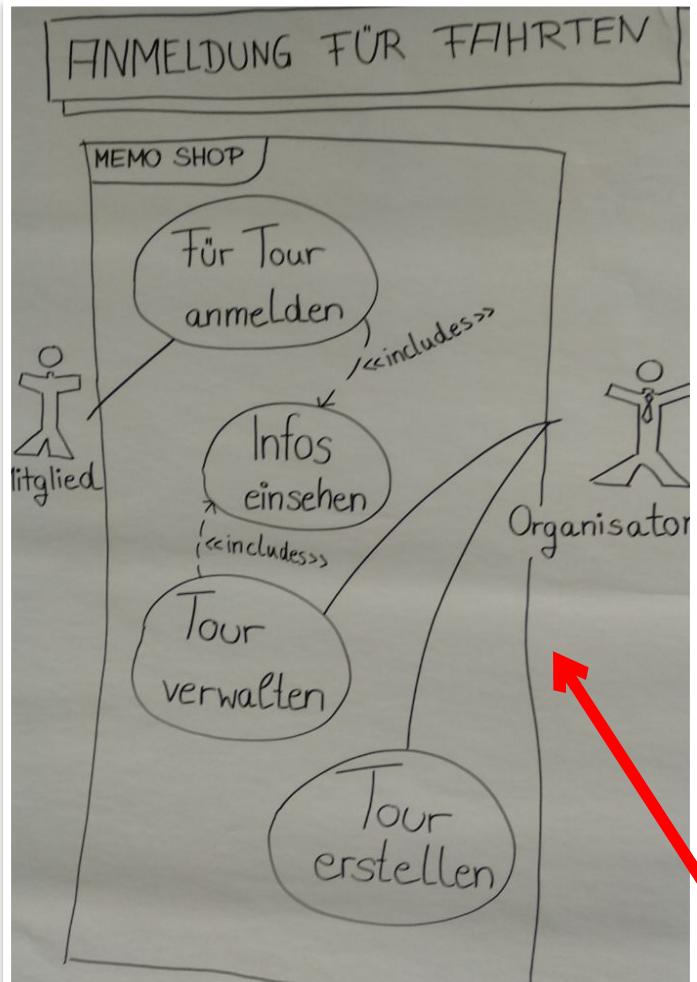
Quelle: Rupp, C. et al.
(2005)

ABBILDUNG 13.102 Anwendung eines mehrdimensionalen Aktivitätsbereichs

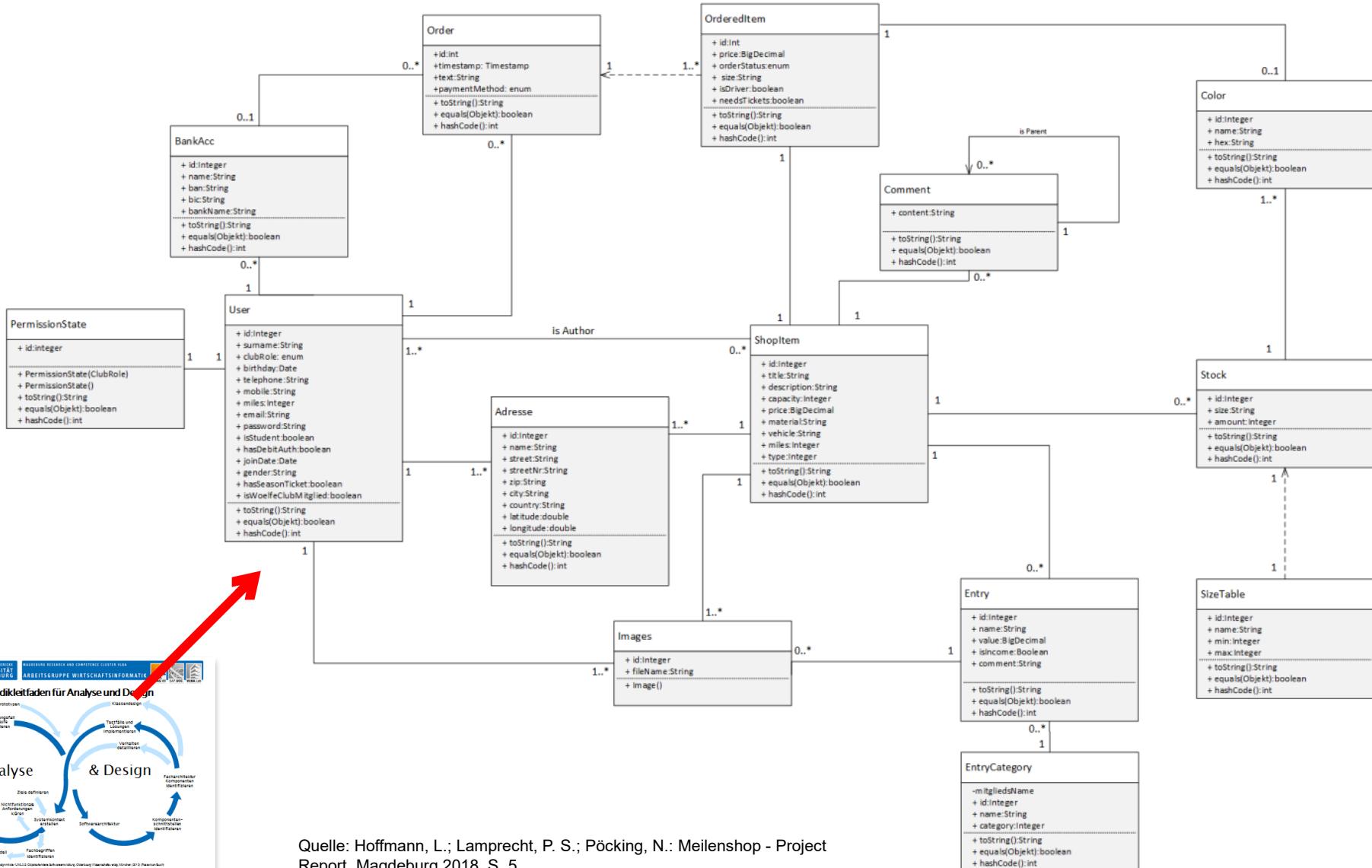
Fallbeispiel „Meilenwölfe“



Fallbeispiel „Meilenwölfe“



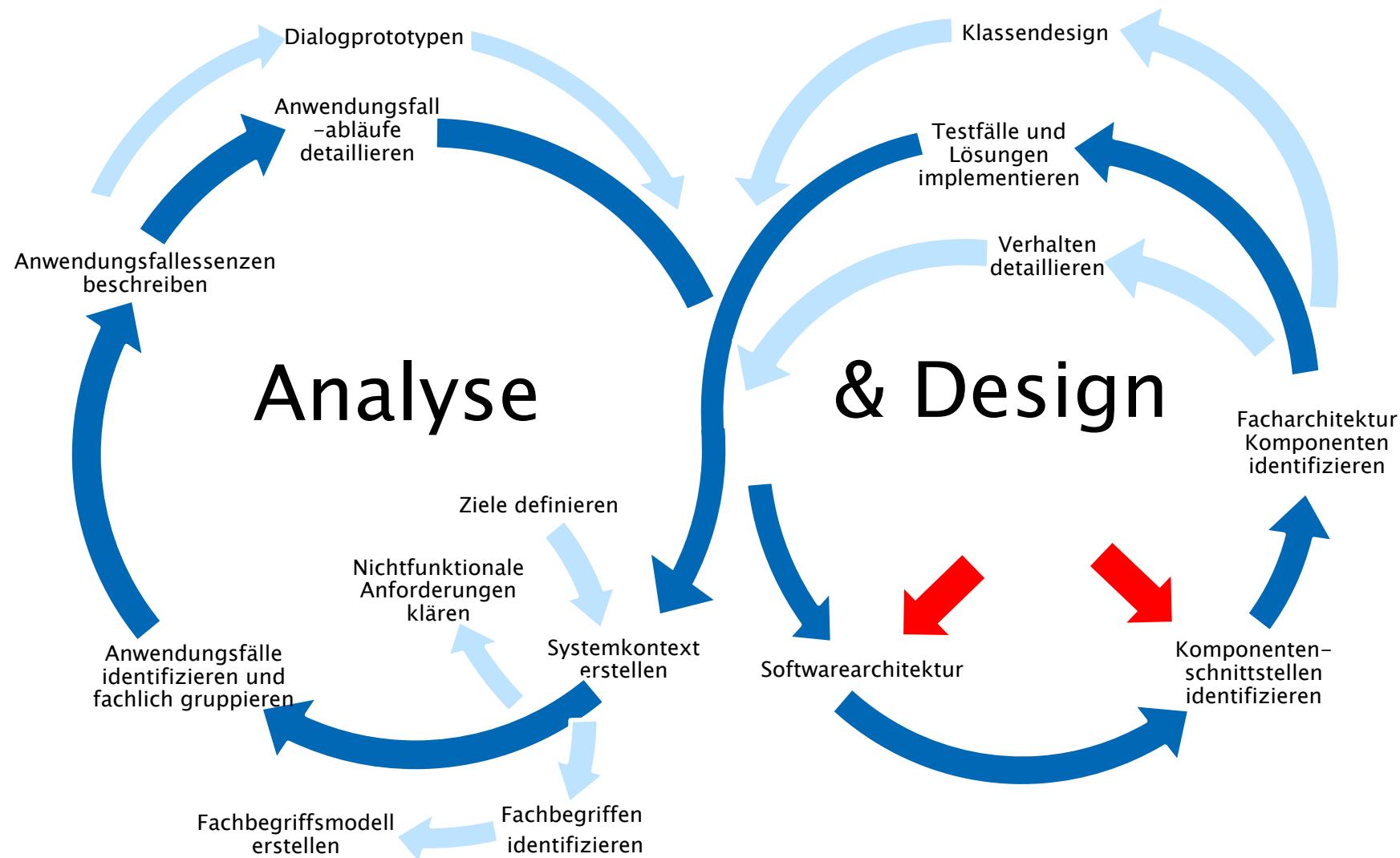
Fallbeispiel „Meilenwölfe“



Quelle: Hoffmann, L.; Lamprecht, P. S.; Pöcking, N.: Meilenshop - Project Report. Magdeburg 2018, S. 5.

Sequenzdiagramm

UML-Methodikleitfaden für Analyse und Design



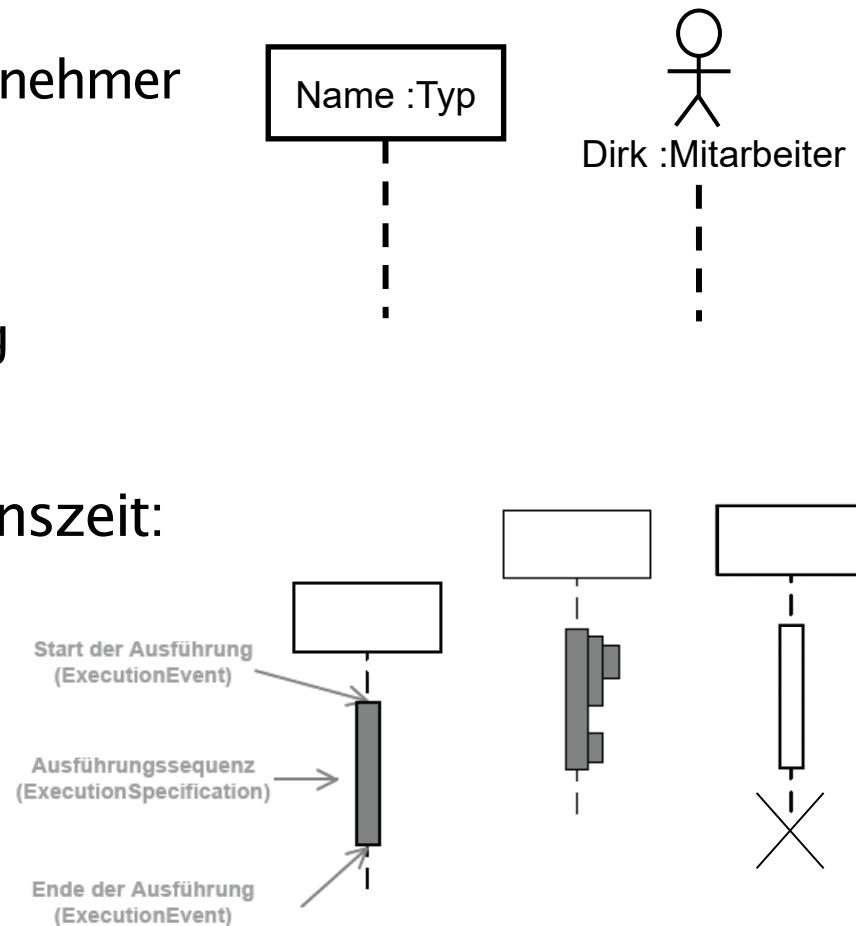
Sequenzdiagramm

- Genaue Darstellung der Interaktionen zwischen Objekten:
 - Nachrichtenaustausch
 - Interaktion von Benutzern mit Oberflächen
- Reihenfolge von Interaktionen
- Graphenstruktur mit gerichteten Kanten
- Parallele, optionale, kritische Kommunikation möglich
- Befehlseerteilung
- Reaktion des Systems

Sequenzdiagramm (Notation)

- Lebenslinie:
 - Repräsentiert genau einen Teilnehmer
 - Person oder Objekt/System
 - Teilnehmer horizontal
 - Zeitlicher Verlauf vertikal
 - Name optional, Typ notwendig

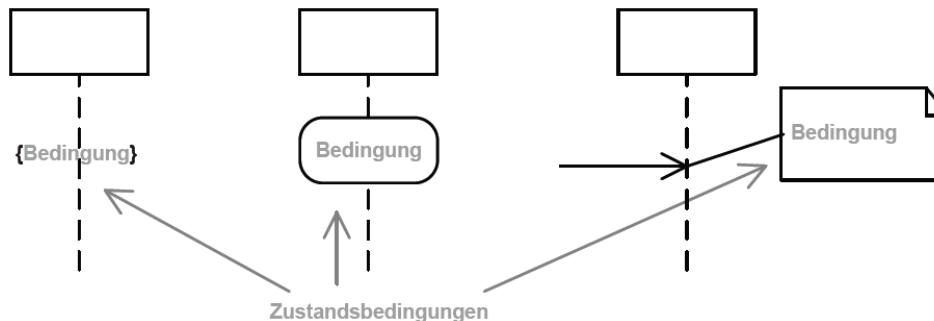
- Ausführungsbalken und Lebenszeit:
 - Aktive Zeit
 - Passive Lebenszeit
 - Parallel Tätigkeiten möglich
 - Destruktion des Teilnehmers



Sequenzdiagramm (Notation)

- Zustandsinvarianten:

- Spezifizierung des Zustands des Teilnehmers
- Bedingung für Interaktion

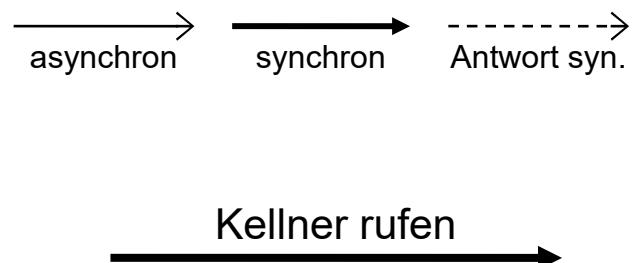


Arbeitet == true

{arbeitet == true}

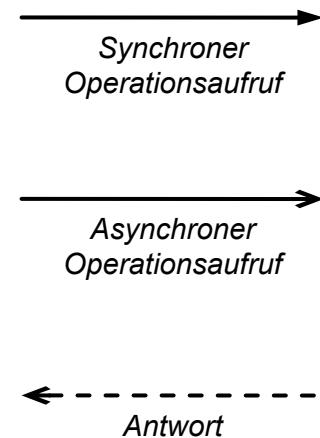
- Nachricht

- Kommunikation zw. zwei Teilnehmern
- Aufruf einer Operation
- Signal senden
- Nachricht an sich selbst



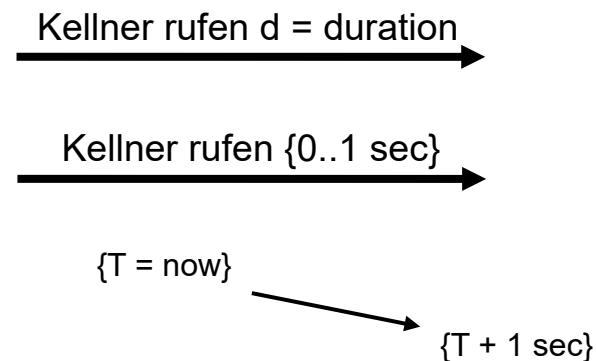
Verhaltensdiagramme: Sequenzdiagramm

- Zeitliche Abfolge von Nachrichten zwischen Objekten
- Fokus auf Austausch von Nachrichten und Ablauf der Kommunikation
- Notation
 - **Synchrone Nachricht**
 - Mit Blockierung des Senders
 - **Asynchrone Nachricht**
 - Ohne Blockierung des Senders
 - **Antwort**

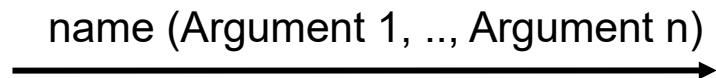


Sequenzdiagramm (Notation)

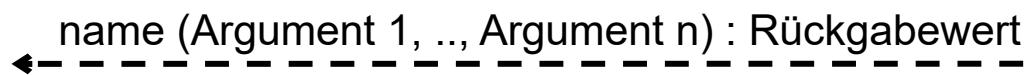
- Zeitbedingungen
 - Zeitmessung
 - Festlegung der Dauer
 - Versetzte Nachrichtenlinie



- Nachrichtarten
 - Sendenachricht



- Antwortnachricht



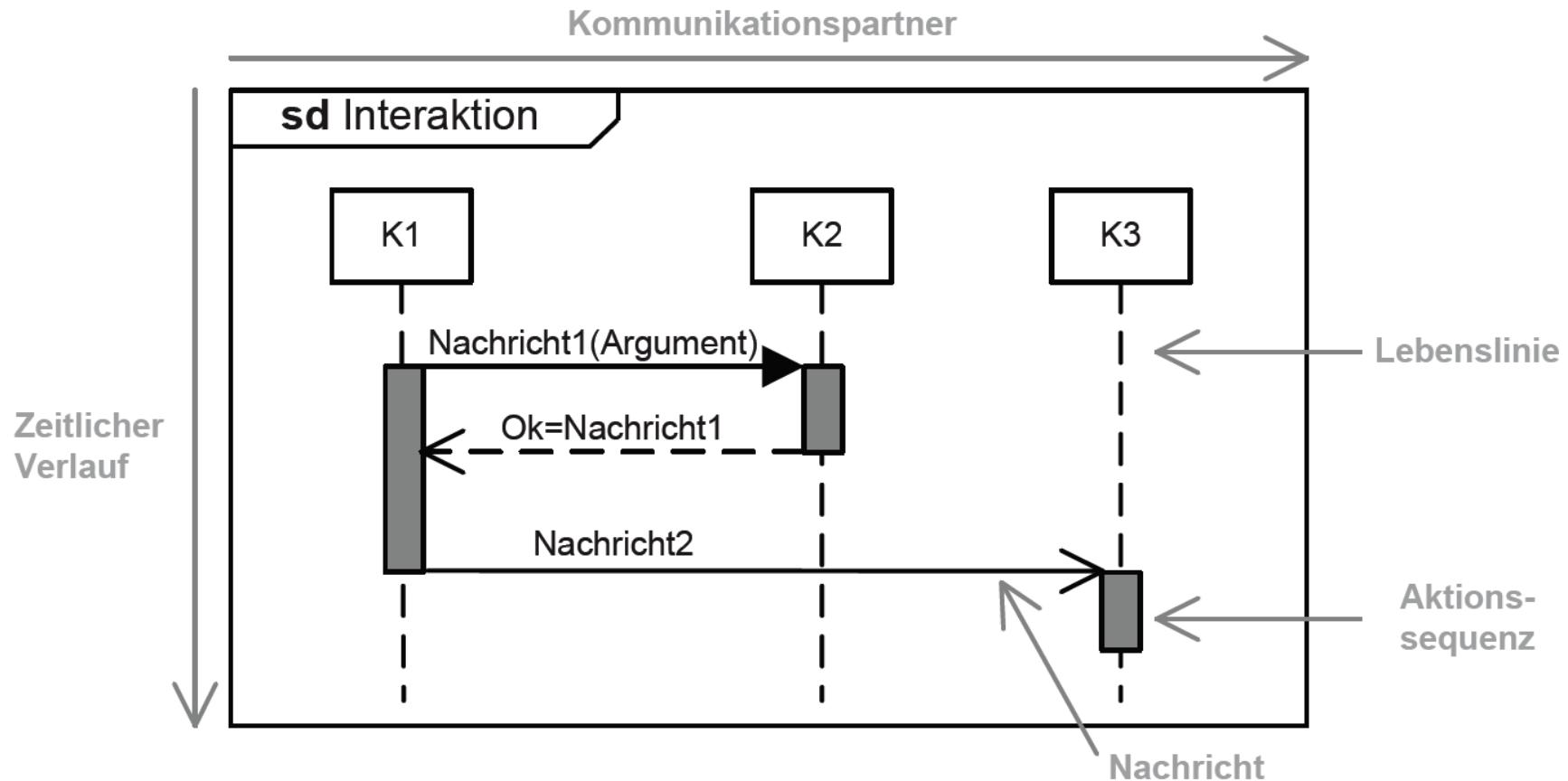
Verhaltensdiagramme: Sequenzdiagramm

- Notation
 - **Kombinierte Fragmente**
 - Modularisierung von Interaktionen
 - Besteht aus
 - Interaktionsoperator (Art des kombinierten Fragments)
 - Interaktionsoperanden (Interaktionsfragmente in kombiniertem Fragment)
 - Bedingung (Bedingung für den Aufruf eines Interaktionsfragments)
 - Interaktionsoperatoren:

Schlüsselwort	Deutsch	Englisch	Einsatzzweck
alt	Alternatives Fragment	Alternative	Alternative Abläufe
break	Abbruchfragment	Break	Ausnahmefälle
loop	Schleife	Loop	Schleifen
opt	Optionales Fragment	Option	Optionaler Teil
par	Paralleles Fragment	Parallel	Nebenläufiger Teil

Auswahl wichtiger, kombinierter Fragmente

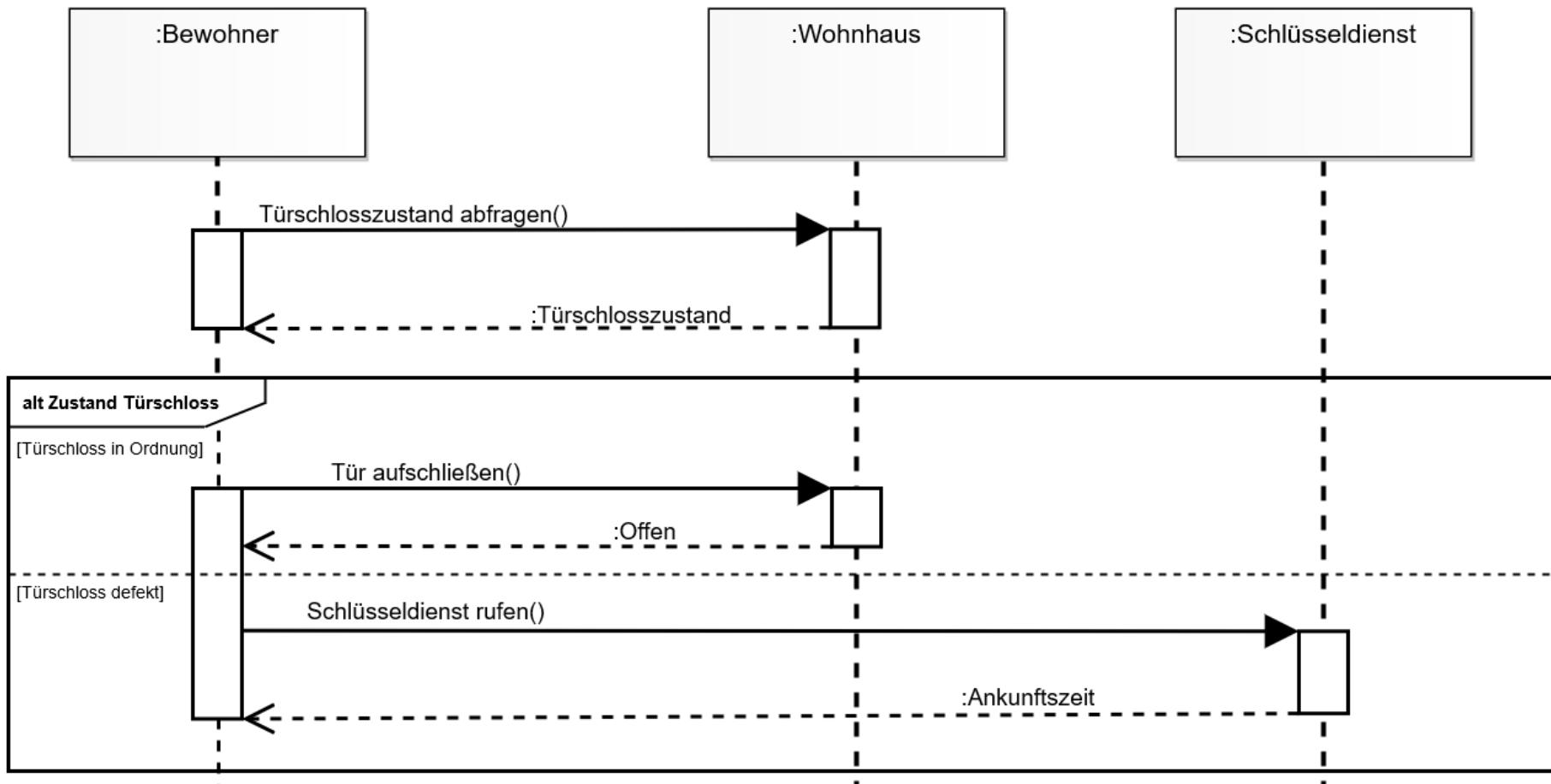
Beispiel



Quelle: Rupp, C. et al. (2005): UML 2 glasklar: Praxiswissen für die UML-Modellierung und –Zertifizierung. München – Wien.

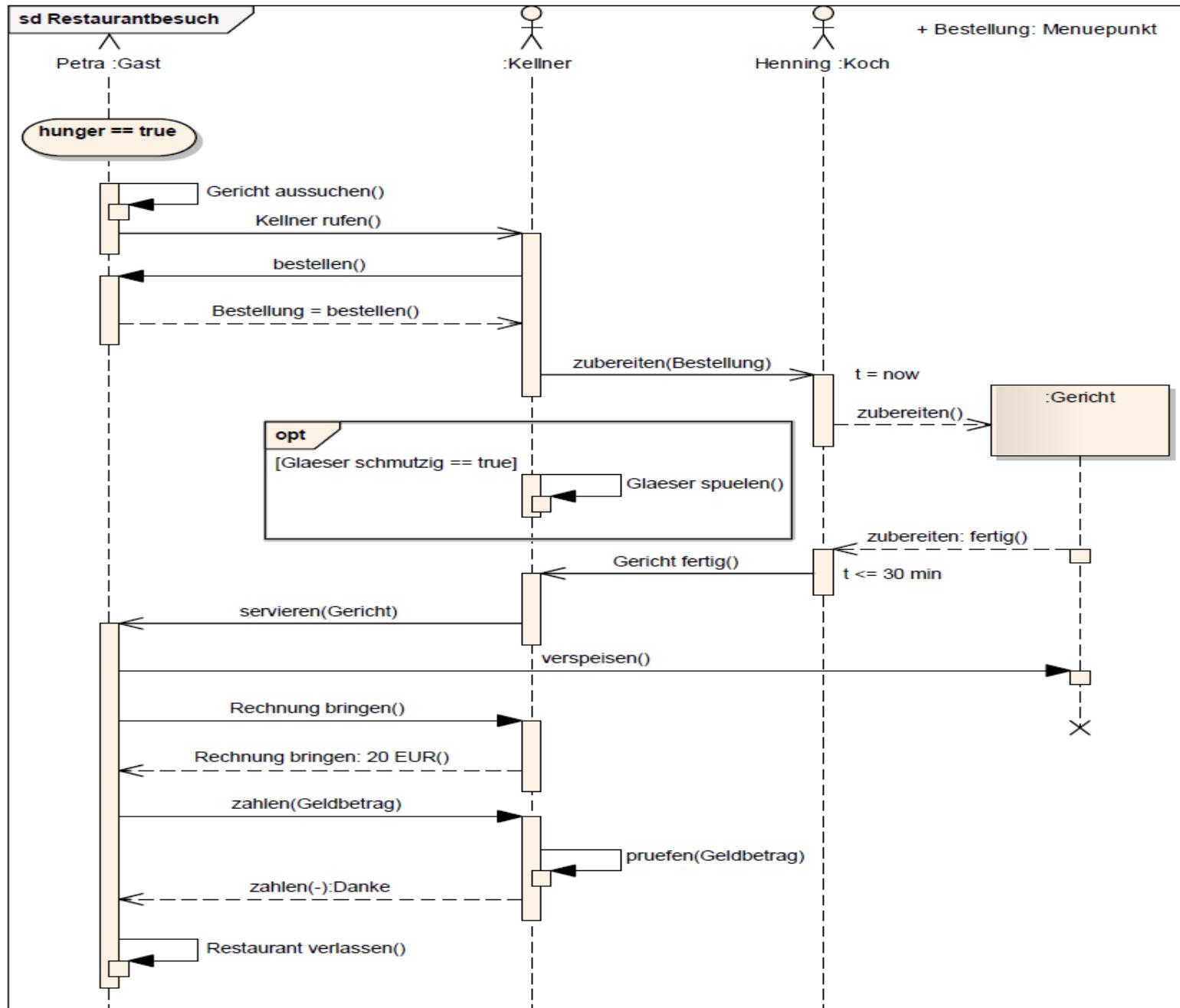
Verhaltensdiagramme: Sequenzdiagramm

- Übersicht



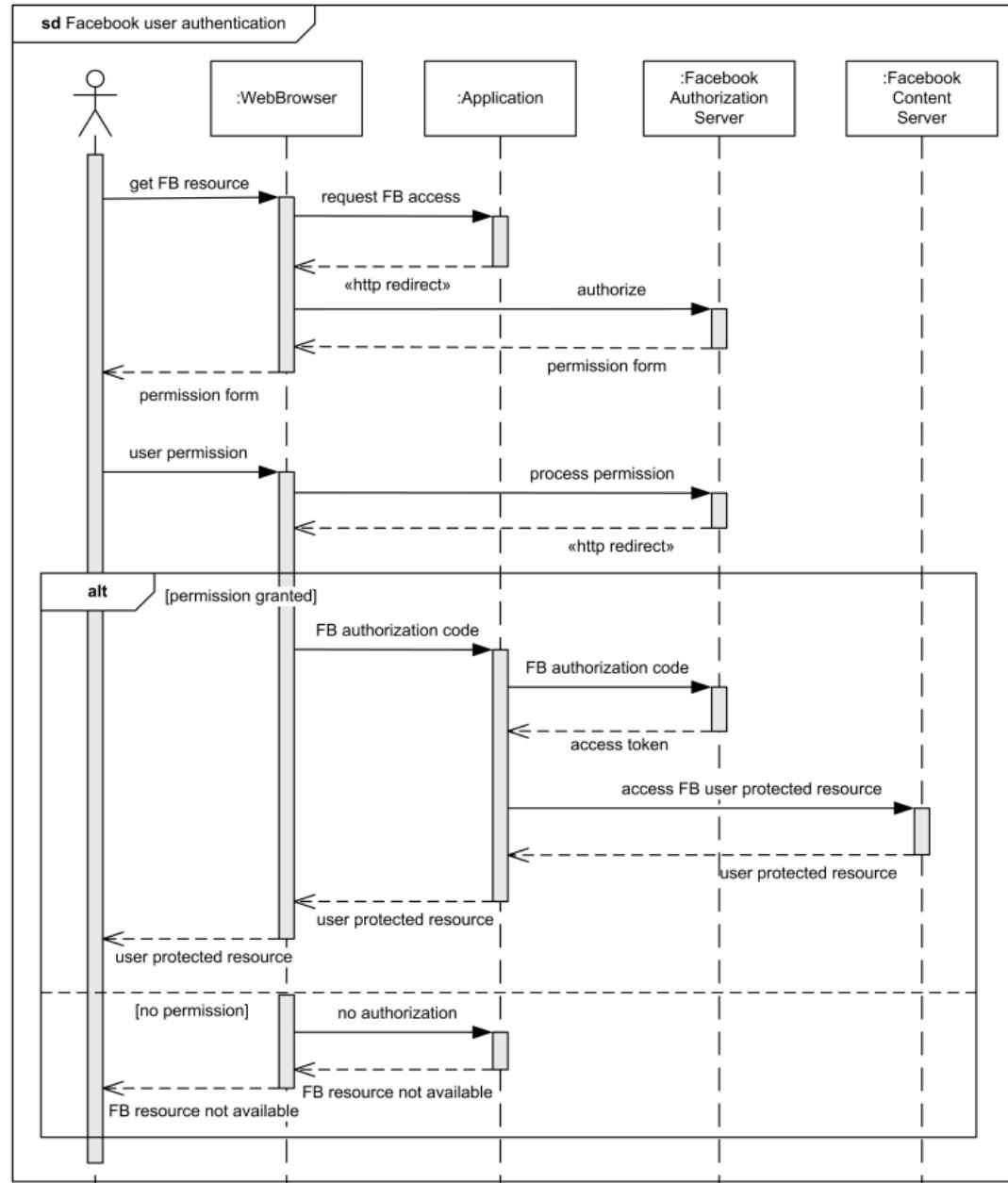
Sequenzdiagramm: Haus betreten

Beispiel



Quelle: Kecher, C. (2011): „UML 2 Das umfassende Handbuch“. fourth edition, Bonn.

Verhaltensdiagramme: Sequenzdiagramm – Beispiel



Quelle: [<http://www.uml-diagrams.org/sequence-diagrams-examples.html#facebook-authentication>, Abruf 05.05.2015]

Zustandsdiagramm

Zustandsdiagramm

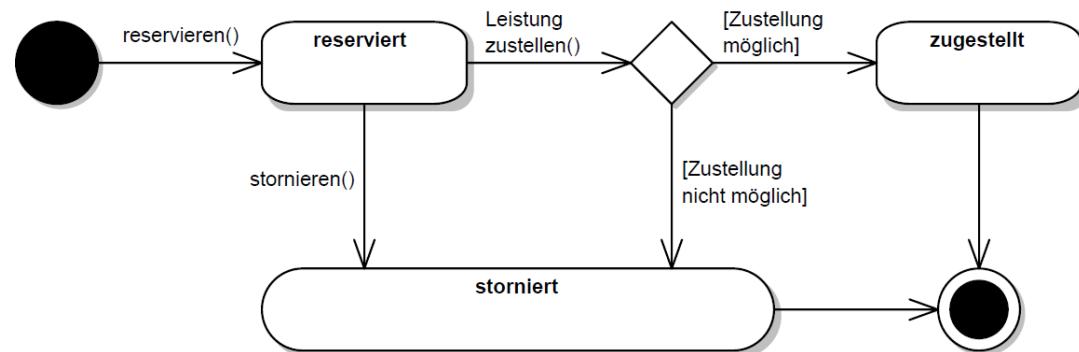
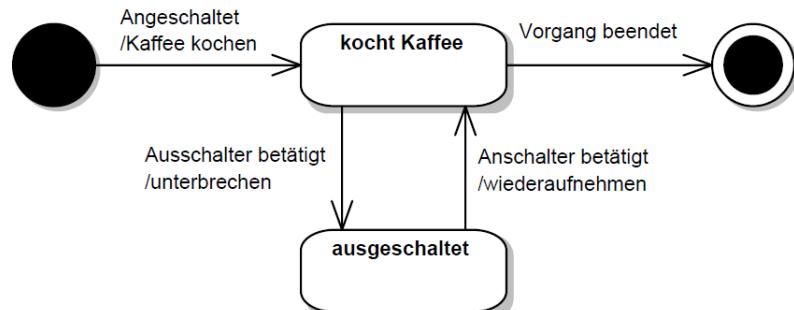
- Genaue Darstellung des dynamischen Verhalten eines Systems
 - Mögliche Zustände
 - Zustandsübergänge
 - Ereignisse
 - Aktionen
- Reihenfolge von Reaktionen
- Graph mit gerichteten Kanten
- Parallele und optionale Situationen möglich
- Basiert auf dem Konzept eines deterministischen, endlichen Automaten

Verhaltensdiagramme: Zustandsdiagramm

- Zustandsänderungen einzelner Objekte zur Lebenszeit
 - Statusinformationen, Transitionen, Ereignisse und Aktivitäten
- Fokus auf Reaktionen des Systems (z.B. GUI)
- Zwei Arten von „Zustandsautomaten“
 - Verhaltenszustandsautomat
 - Modelliert Verhalten eines Modellelements
 - Protokollzustandsautomat
 - Spezifiziert zulässige Nutzung der Verhaltensmerkmale eines Modellelements

Verhaltensdiagramme: Zustandsdiagramm

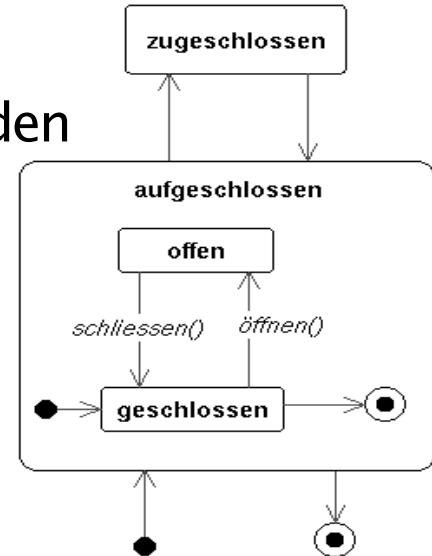
▪ Übersicht



Protokollzustandsautomat: Festlegung der Reihenfolge der Operationen für eine Reservierung

Zustandsdiagramm (Notation)

- Zustand:
 - Elemente durchlaufen endliche Anzahl an Zuständen
 - (stat./dynam.) Situation mit genau definierten Rahmbedingungen
 - Zusammengesetzte Zustände möglich



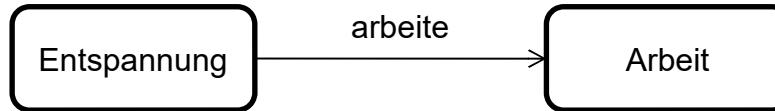
- Transition:
 - Gerichtete Beziehung zwischen zwei Zuständen
 - Zustandsübergang vom Quellzustand zum Zielzustand
 - Default-Wert: fehlendes Ereignis = „Aktivität ist abgeschlossen“



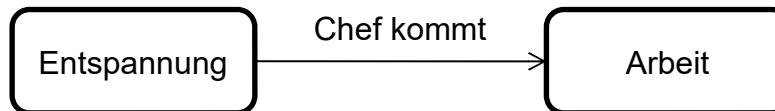
Zustandsdiagramm (Notation)

- Ereignis (Event):

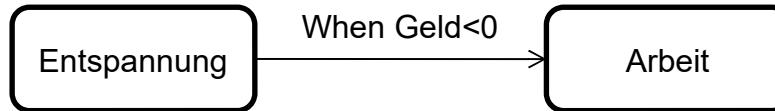
- CallEvent:



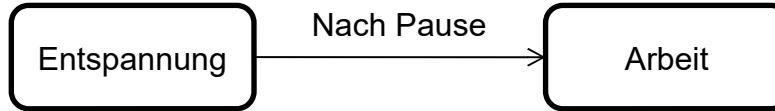
- SignalEvent:



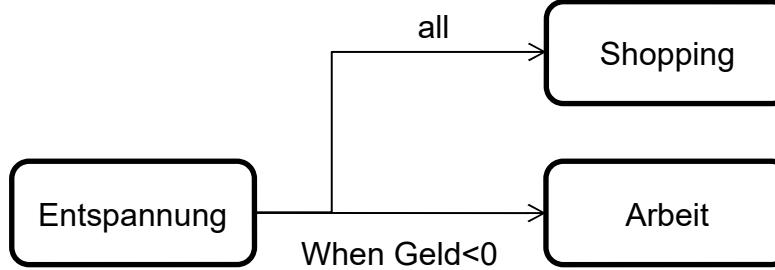
- ChangeEvent:



- TimeEvent:

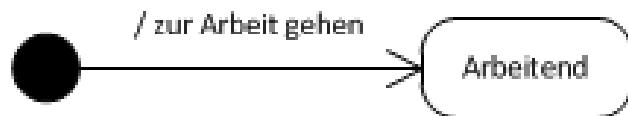


- AnyReceiveEvent:



Zustandsdiagramm (Notation)

- Startzustand:
 - Startpunkt eines Zustandsautomaten
 - Beginn/Generierung eines modellierten Elements
 - Eine wegführende Kante pro Startzustand
 - Ein Startzustand pro Zustandsautomat



- Endzustand:
 - Ende einer Region von Zuständen bzw. „Lebensende“ eines modellierten Elements
 - Beliebig viele Endzustände
 - Beliebig viele eingehende Kanten pro Endzustand

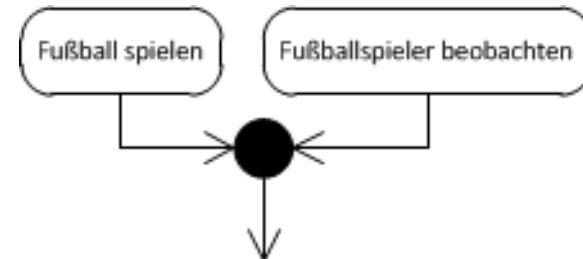
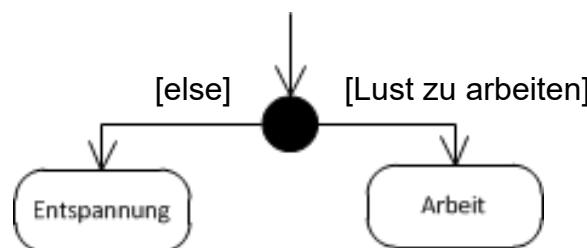


Zustandsdiagramm (Notation)

- Terminator:
 - Beendet kompletten Zustandsautomaten
 - Beliebig viele Endzustände
 - Beliebig viele eingehende Kanten pro Endzustand

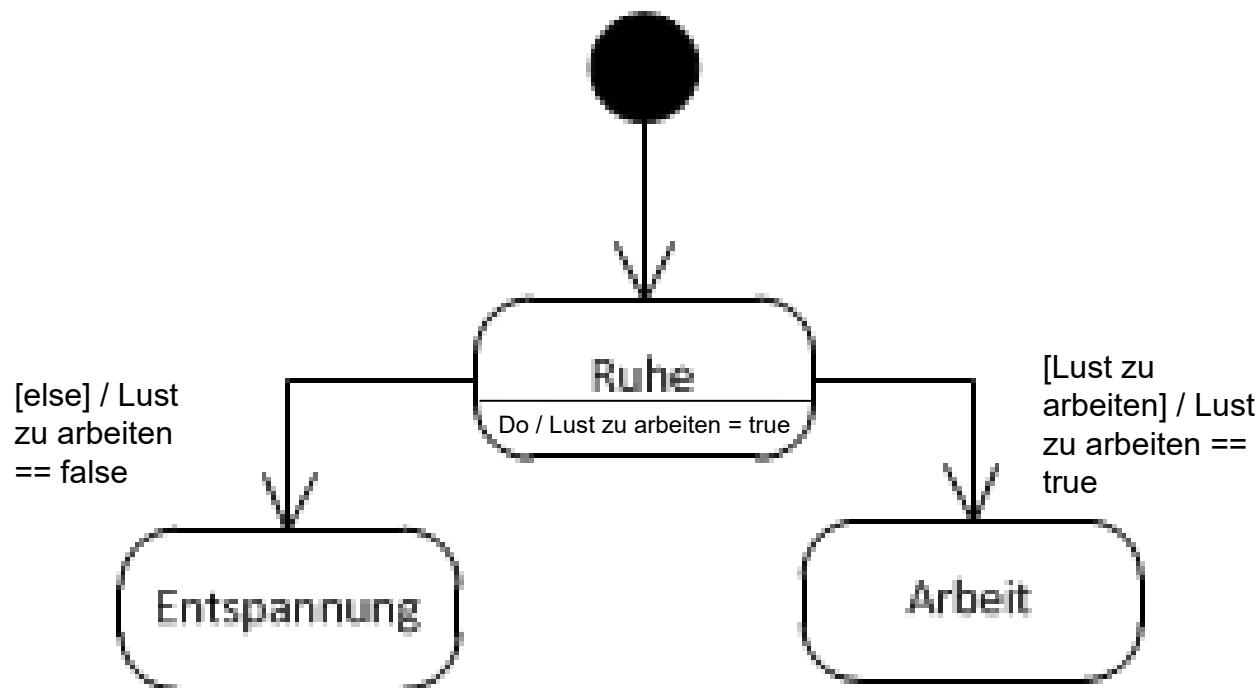


- Kreuzungen:
 - Verzweigt Transitionen statisch (Entscheidung bereits vorher)
 - Fasst mehrere eingehende Transitionen zu einer ausgehenden zusammen



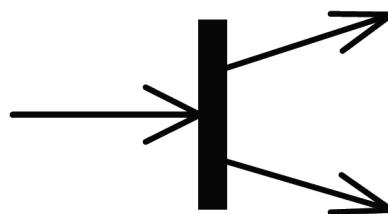
Zustandsdiagramm (Notation)

- Entscheidungen
 - Verzweigt Transitionen dynamisch
 - Ausgehende Transaktion wird erst beim Erreichen der Entscheidung gewählt anhand der Guards

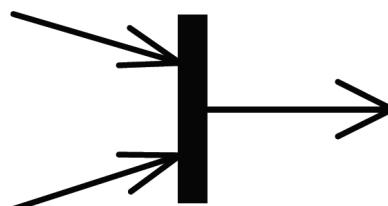


Zustandsdiagramm (Notation)

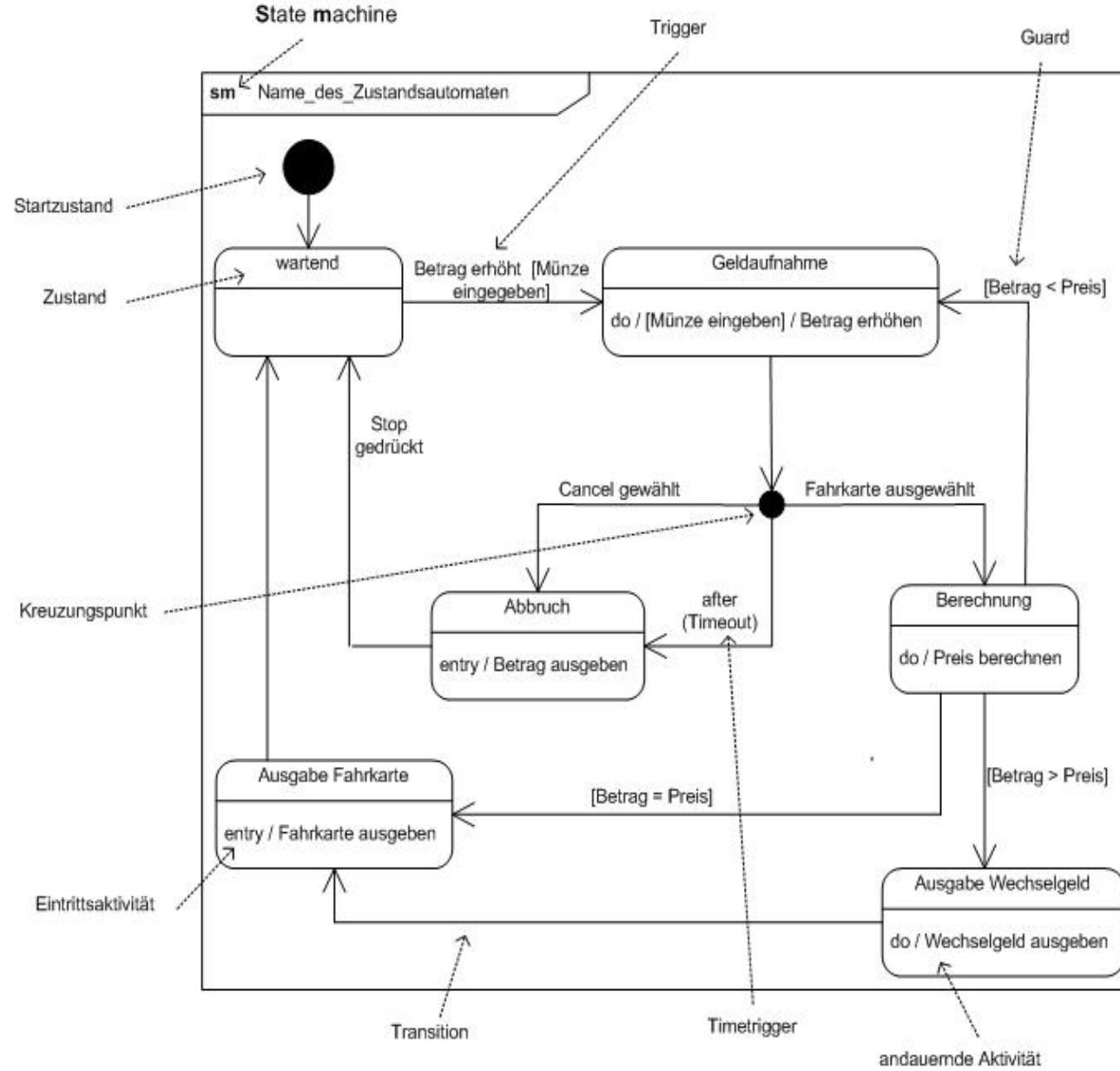
- Gabelung
 - Teilt eine eingehende Transition in mehrere parallel geschaltete ausgehende Transitionen



- Vereinigung
 - Verbindet mehrere parallel geschaltete eingehende Transitionen in eine ausgehende Transitionen

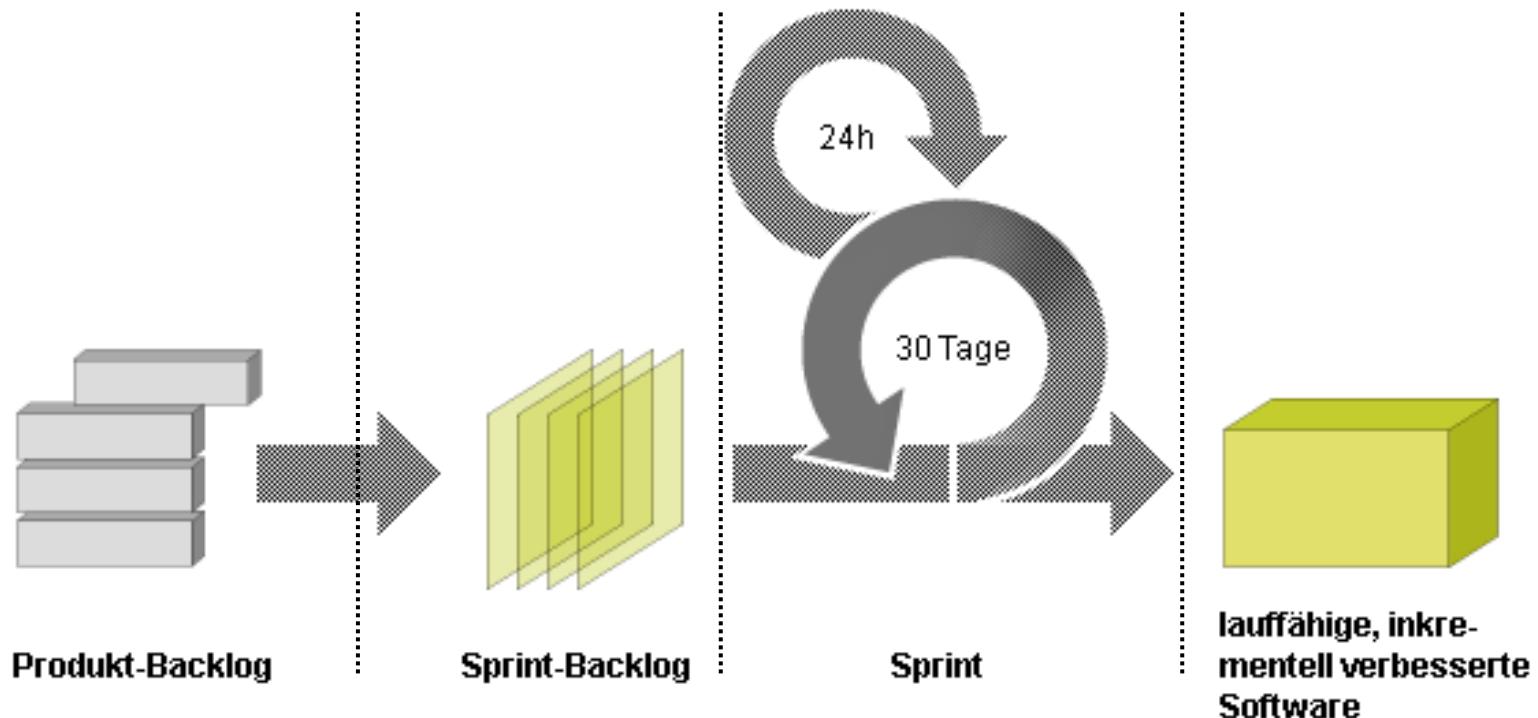


Zustandsdiagramm (Beispiel)



Zusammenfassung UML

Software-Entwicklung im Kontext von SCRUM



Anwendungsmöglichkeiten für UML-Diagramme:

Anwendungsfalldiagramm

K l a s s e n d i a g r a m m

Aktivitätsdiagramm, Sequenzdiagramm,

Zustandsdiagramm, Zeitverlaufsdigramm,

Klassendiagramm, Paketdiagramm, Interaktionsübersichtsdiagramm

Objektdiagramm

m m

Verteilungsdiagramm,

OCL,

Kommunikationsdiagramm

Problemdefinition / Analyse

Spezifikation / Entwurf

Implementierung / Test

Einführung / Wartung

→ Weiterführung: IT-Projektmanagement

Zusammenfassung

- UML2 Standard besteht aus vier Teilen
 - **Superstructure**: (Definition der Notation sowie der Semantik für Diagramme und ihre Modellelemente)
 - **Infrastructure**: (Definition des UML-Metamodells für die Superstructure)
 - **Object Constraint Language (OCL)**: (Spezifikation von Regeln für Modellelemente)
 - **Diagram Interchange (DI)**: (Spezifiziert Layout der Diagramme)

Zusammenfassung

- UML2 besitzt ein eigenes Metamodell
 - Klärung der Bedeutung von Modellierungskonstrukten
- Aktueller Stand
 - UML 1.x wird in Industrie und Forschung verbreitet genutzt
 - Auch mit UML2 verbleiben Schwierigkeiten
 - Werkzeuge unterstützen den neuen Standard oft noch nicht vollständig

Zusammenfassung

- UML2 ist ein ...
 - Standard, der ein Metamodell für die Modellierung von Systemen bereitstellt
 - Standard, der Modellierungskonstrukte und deren Beziehungen untereinander festlegt
- UML2 ist kein ...
 - Standard für eine graphische Modellierungssprache (nur Vorschläge und Empfehlungen)
 - Modell mit einer formal definierten Semantik
 - An vielen Stellen erfolgt eine Interpretation auf Basis des allgemeinen Verständnisses

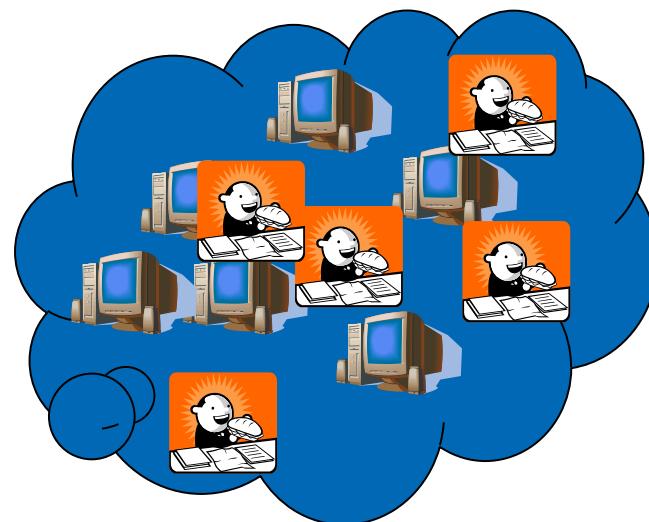
Wiederholung Modellwelt

Mehrstufiges Vorgehen – BIS für die Uni Magdeburg – I

Ziel ist die Erstellung eines Betrieblichen Informationssystem
für die Uni

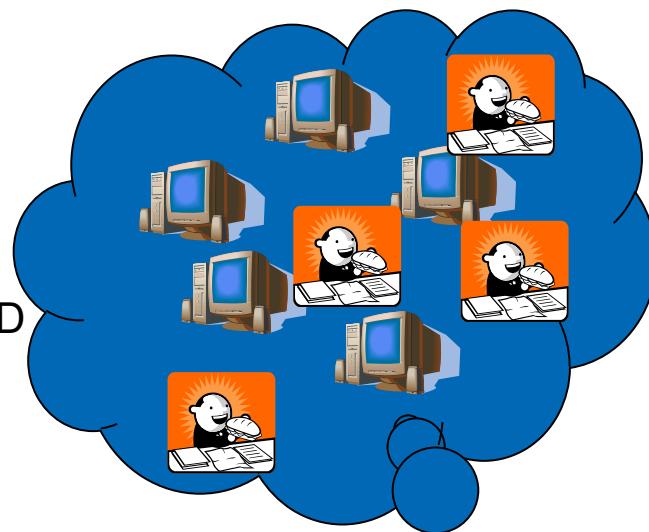
1. Ableitung der Diskurswelt aus der realen Welt

Reale Welt



Computer und Mitarbeiter in **MD**

Handlungs-relevanz
BIS für Uni MD



Computer und Mitarbeiter an der **Uni MD**

Mehrstufiges Vorgehen – BIS für die Uni Magdeburg – II

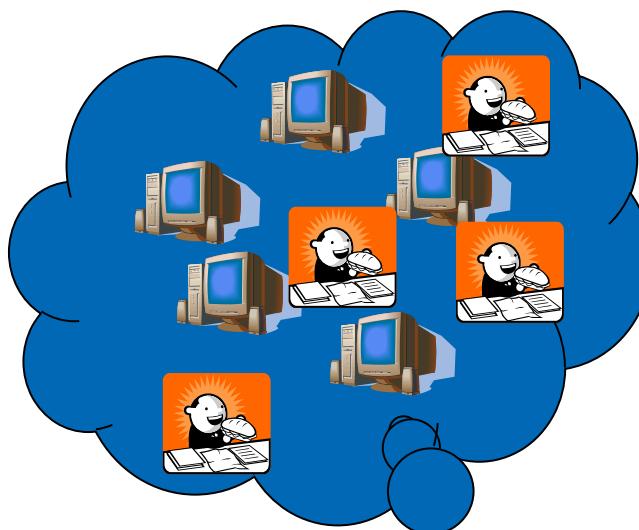
2. Ableitung eines Objektsystems S_o aus der Diskurswelt

- Durch subjektive Interpretation wird aus unserer Diskurswelt ein Objektsystem S_o
 - Objektsystem S_o
 - Elemente des Systems sind abstrakte Repräsentationen realweltlicher Objekte
 - Abstrakte Repräsentation einer realweltlichen Person:
 - Datensatz
 - Aber:
 - Im interpretierten Objektsystem S_o sind noch viele Objekte und Beziehungen erhalten
- Eine Reduzierung der Komplexität wurde noch nicht erreicht.

Mehrstufiges Vorgehen – BIS für die Uni Magdeburg – III

2. Ableitung eines Objektsystems S_o aus der Diskurswelt

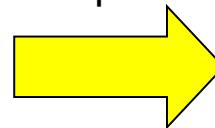
Diskurswelt Uni Magdeburg



Objektsystem S_o Uni
Magdeburg

123567	Geb 19-115	2006	0,3 kW
123568	Geb 19-116	2008	0,5 kW

Subjektive
Interpretation



XXXX	Geb XXXX	20XX	X,X kW
------	----------	------	--------

Computer an der Uni MD

Meier	Hans	01.04.1980	4000,00
Müller	Fritz	01.06.1986	3000,00

Computer und Mitarbeiter an der Uni MD

xxx	xxx	xx.xx.xxxx	xxxxxx,xx
-----	-----	------------	-----------

Mitarbeiter an der Uni MD

Mehrstufiges Vorgehen – BIS für die Uni Magdeburg – IV

3. Ableitung eines Modellsystems S_M aus dem Objektsystem S_O

- Reduzierung der Komplexität durch
 - Überführung des Objektsystems S_O in ein **Modellsystem** S_M
- Modellsystem S_M ist wiederum eine subjektive Interpretation des Objektsystems S_O
 - Eliminierung irrelevanter Sachverhalte
 - Typisierung relevanter Sachverhalte
- Typisierung
 - Mehrere Elemente des Objektsystems S_O sind dann vom gleichen Typ, wenn ihre Eigenschaften übereinstimmen
 - „Peter Meier, Sachbearbeiter“ ; „Fritz Müller, Buchhalter“ vom gleichen Typ Mitarbeiter
- Abbildung $S_O \rightarrow S_M$ ist nur in eine Richtung eindeutig
- Das Modellsystem S_M muss sich struktur- und verhaltenstreu zum Objektsystem S_O verhalten

Mehrstufiges Vorgehen – BIS für die Uni Magdeburg – V

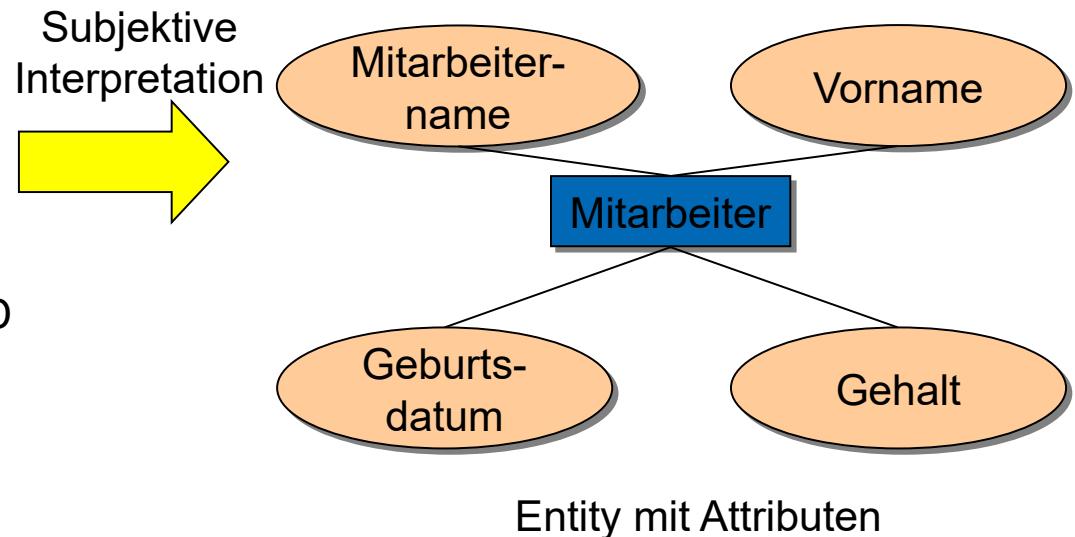
3. Ableitung eines Modellsystems S_M aus dem Objektsystem S_O

Objektsystem S_O Uni Magdeburg

Meier	Hans	01.04.1980	4000,00
Müller	Fritz	01.06.1986	3000,00
xxx	xxx	xx.xx.xxx	xxxxx,xx

Mitarbeiter an der Uni MD

Modellsystem S_M Uni Magdeburg



Meta-Modelle

- Unterschiedliche Zielstellungen für die zu entwickelnden Modellsysteme S_M
 - Modellierung passiver Systemelemente -> Datenmodelle
 - Modellierung von Geschäftsprozessen -> Geschäftsprozessmodelle
 - Modellierung des Verhaltens von realen oder gedachten Systemen -> Simulationsmodelle
- Beschreibung der unterschiedlichen Modelle durch sog. Meta-Modelle
 - Meta-Modelle sind Modelle zur Beschreibung von Modellen
- Beispiele für Meta-Modelle
 - Entity-Relationship-Modell: Meta-Modell für Datenmodelle
 - Ereignisgesteuerte Prozessketten: Meta-Modell für Geschäftsprozessmodelle
 - Diskretes-Ereignismodell: Meta-Modell für Simulationsmodelle

Grundsätze ordnungsmäßiger Modellierung (GoM)

Ein Ordnungsrahmen für die Reduktion der Subjektivität bei der Modellierung

Ziel der Definition von GoM

- Erhöhung der Qualität von Modellen eines Fachkonzepts durch
 - Angabe von Gestaltungsempfehlungen zur bedarfsgerechten Modellerstellung
 - Reduzierung der Vielfalt möglicher Modellierungsvarianten
 - Erhöhung von Vergleichbarkeit und Integrationsfähigkeit von Modellen

Deduktive Herleitung der GoM

- Quellen für die Ableitung
 - Ex definitionem:
 - Ableitung aus den Zielen der Informationsmodellierung
 - Ziele, die von den Modelladressaten vorgegeben werden
 - Ziele, die jedem ökonomischen Handeln zugrunde liegen

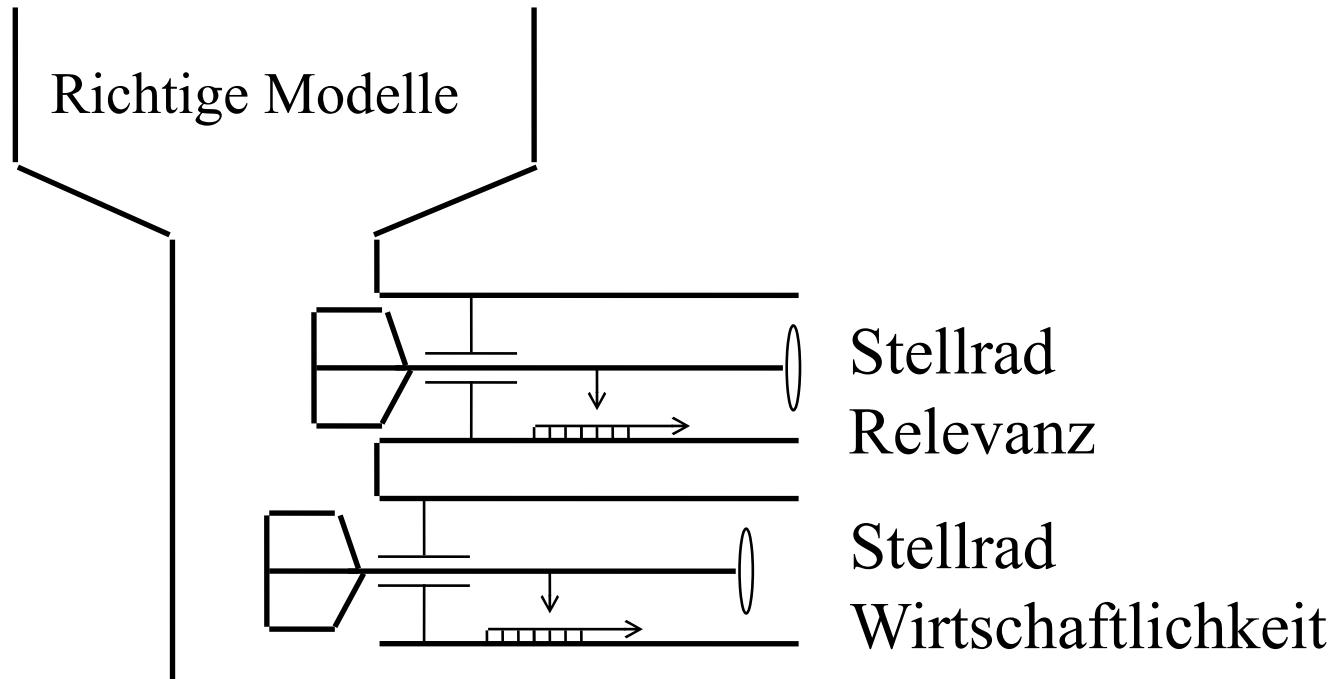
Grundsätze im Überblick (I)

- Grundsatz der Richtigkeit
 - Korrekte Abbildung der Realwelt auf das Modell
- Grundsatz der Relevanz
 - Modell muss subjektiv ziel- und zweckorientiert sein
- Grundsatz der Wirtschaftlichkeit
 - Verhältnismäßigkeit von Kosten und Ertrag ist zu berücksichtigen

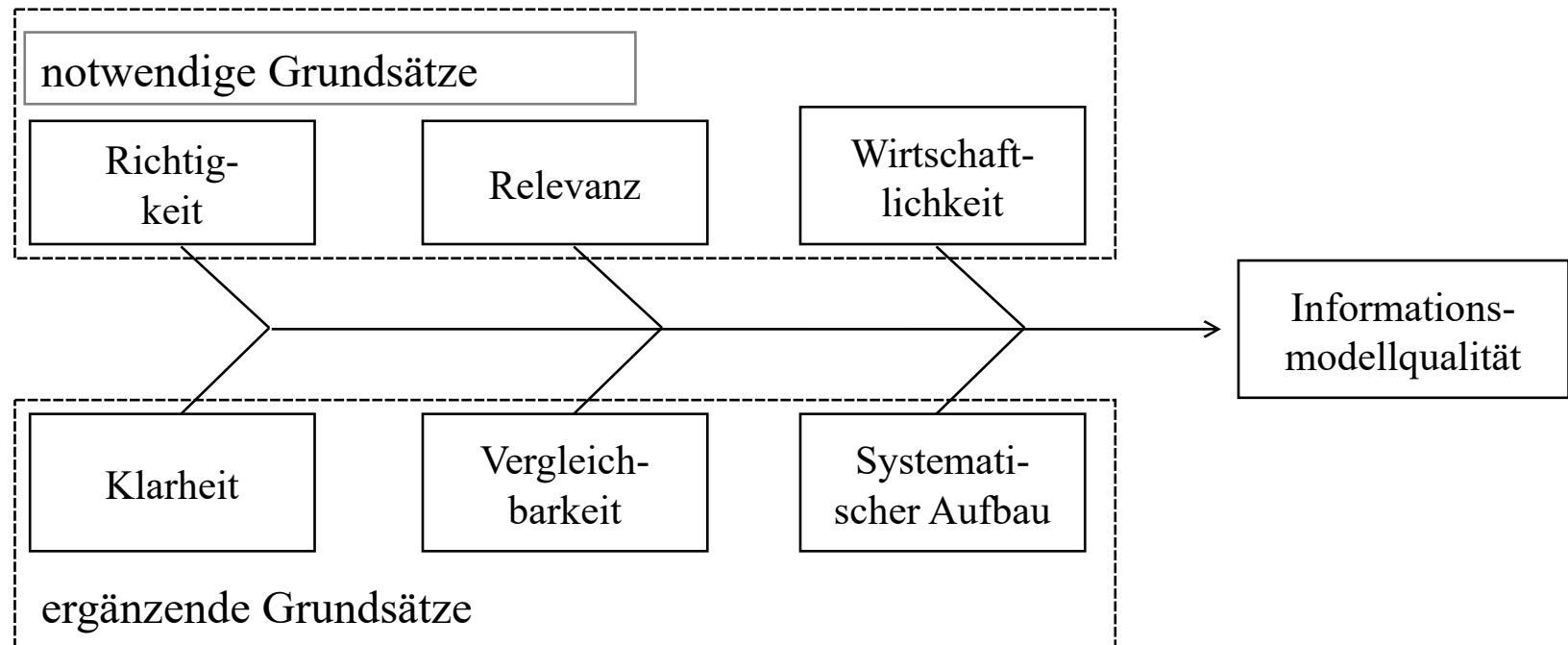
Grundsätze im Überblick (II)

- Grundsatz der Klarheit
 - Modell muss sowohl für Modellierer wie Adressat anschaulich sein
- Grundsatz der Vergleichbarkeit
 - Identitäten, Äquivalenzen und Kompatibilitäten von Modellen müssen erkennbar sein
- Grundsatz des systematischen Aufbaus
 - Informationsarchitekturen umfassen verschiedene Sichten, die zueinander passen müssen

Zusammenhang zwischen Richtigkeit, Relevanz und Wirtschaftlichkeit



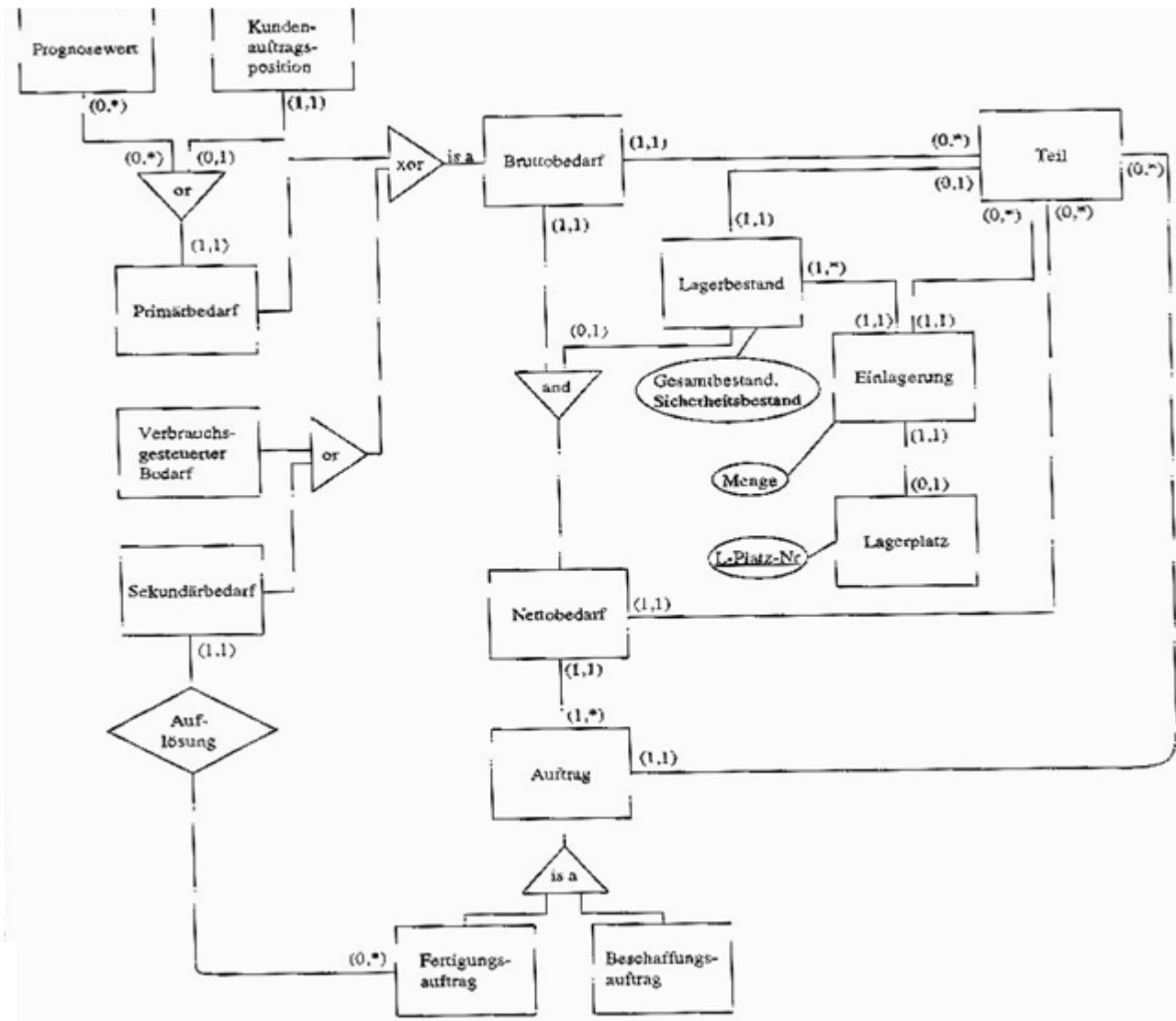
Klassifizierung der GoM



Grundsatz der Richtigkeit

- Syntaktische Richtigkeit
 - vollständig gegenüber zu Grunde liegenden Metamodell
 - Alle methodischen Konstrukte, welche die Modellsyntax erfordert, sind vorhanden.
 - konsistent gegenüber zu Grunde liegendem Metamodell
 - alle im Modell verwendeten Informationsobjekte und Notationsregeln werden im Metamodell erklärt.
- Semantische Richtigkeit umfasst
 - Homomorphismus des Modells gegenüber der Realität
 - Messgrößen: Struktur- und Verhaltenstreue
 - relativ zu Ideal- oder Sollmodellen
 - gemessen an sachlogischen Gegebenheiten und Zusammenhängen
 - Semantische Konsistenz
 - Aktualität

Beispiel



Grundsatz der Relevanz

- Selektion aller aus der Gesamtheit der Realweltphänomene zu modellierenden Sachverhalte
 - Priorisierung nach betriebswirtschaftlichen Kenngrößen oder Zielen
 - Relevanzanalyse
 - sinkt der Nutzeffekt eines Modells für einen Modelladressaten, wenn Teilmodell extrahiert wird, dann ist das Teilmodell relevant
- Relevanz entspricht in etwa Minimalität
- Wichtig: Klare Zielexplizierung
- Relevanz betrifft
 - Objektsystem (modellierte Elemente)
 - Abbildungsbeziehung zwischen Realwelt und Modell (Modellierungsmethode)
 - Modellsystem (Anwendung der Modellierungsmethode)

Grundsatz der Wirtschaftlichkeit

- Modellnutzen ist pauschal nicht bestimmbar, daher ist Untersuchung der Einflussfaktoren erforderlich
 - Erstellungsaufwand
 - Verwendungsdauer
 - Persistenz des Modells
 - Flexibilität

Ansätze zur „wirtschaftlichen“ Modellierung

- Verwendung von Referenzmodellen
- Wiederverwendung von Modellbestandteilen
- Einsatz von rechnergestützten Modellierungswerkzeugen

Grundsatz der Klarheit

- Pragmatik
 - Beziehung zwischen Modell und Modellnutzer
 - subjektive versus intersubjektive Klarheit
- Ästhetische Kriterien
 - Strukturiertheit
 - intuitive Zugänglichkeit
 - Übersichtlichkeit
 - Lesbarkeit

Klare Modelle

- Sind einfach
- Grafisch wohlstrukturiert
 - Positionierung der Modellobjekte in einem Raster
 - Kantenziehung in zwei orthogonalen Dimensionen
 - maximale Gradlinigkeit der Kanten
 - minimale Kantenüberschneidungen
 - grafische Hervorhebung von Korrespondenzen
 - Anordnung der Objekte in Leserichtung
 - Einhaltung von Namenskonventionen

Grundsatz der Vergleichbarkeit

- Relevant besonders bei arbeitsteiliger Modellierung
- Modellübergreifende Wirkung des Grundsatzes
- Zu Grunde liegende Meta-Modelle müssen ineinander überführbar sein
- Erreichung von Modellkonformität durch Einhaltung von Konventionen bezüglich Verwendung von Bezeichnern und Modellkonstrukten

Grundsatz des systematischen Aufbaus

- Erfordert sichtenübergreifendes Meta–Modell
- Schaffung integrationsfähiger Sichten–Modelle