HSBI

Hochschule Bielefeld

University of Applied Sciences and Arts

Software Engineering

Teil 2
Analyse – Was wird gebaut?

Wintersemester 2023/24

The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements ... No other part of the work so cripples the resulting system if done wrong. No other part is as difficult to rectify later.

Fred Brooks, 1987

ANALYSE UND SPEZIFIKATION

2.1 Systemidee

- Am Anfang steht eine grobe Vorstellung vom gewünschten Softwareprodukt und den Zielen

Car Rental System (CRS)

Das Car Rental System (CRS) unterstützt einen kleinen Autovermieter mit einer Niederlassung bei allen wichtigen Geschäftsprozessen. Es verwaltet den Bestand an Fahrzeugen verschiedenen Typs. Wenn Kunden sich telefonisch nach freien Fahrzeugen und den Preisen erkundigen möchten, kann der Mitarbeiter mit CRS schnell die benötigten Informationen abrufen und ein Fahrzeug eines bestimmten Typs für den gewünschten Zeitraum reservieren. Es muss sichergestellt sein, dass ein Fahrzeug der gewünschten Kategorie für den Mietzeitraum zur Verfügung steht. Reservierungen können von den Kunden geändert oder storniert werden. Innerhalb des Mietzeitraums, spätestens aber an dessen Ende, gibt der Kunde das Fahrzeug ab und zahlt die vom System automatisch erstellte Rechnung.

Spezifikation

Reicht das aus, um mit der Entwicklung zu beginnen?

➔ Viele offene Fragen

- Beispiele:
  - Welche Hard- und Software muss beim Nutzer bereits vorhanden sein?
  - Wie ist der genaue Ablauf bei einer Anmietung?
  - Was muss geprüft werden?
  - Welche Daten muss das System genau speichern können?
  - Gehört ein Abrechnungssystem dazu?
  - ….
- Ziel ist es, ausgehend von der Systemidee eine genauere Beschreibung der **Anforderungen**, die **Spezifikation**, zu entwickeln.

Anforderungen

- Eine Anforderung ist eine Aussage über eine zu erfüllende Eigenschaft oder zu erbringende Leistung des Softwareprodukts

- Die Spezifikation ist die Formulierung der Anforderungen eines Softwaresystems in einem Modell. Es gibt formale Spezifikationen (hier nicht behandelt) und solche in natürlicher Sprache, die oftmals um graphische Notationen erweitert werden.

IEEE-Definitionen

requirement — *(1) A condition or capability needed by a user to solve a problem or achieve an objective.*
(2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.
(3) A documented representation of a condition or capability as in (1) or (2).

specification — *A document that specifies, in a complete, precise, verifiable manner, the requirements, design, behavior, or other characteristics of a system or component, and, often, the procedures for determining whether these provisions have been satisfied.*

IEEE Std 610.12-1990

Funktionale und nichtfunktionale Anforderungen

- Funktion der Software = Beziehung zwischen Ein- und Ausgaben
- Alle Anforderungen, die sich auf diese Funktion beziehen, sind funktionale Anforderungen, alle anderen sind nichtfunktionale Anforderungen (non-functional requirements, NFRs).
- Die Formulierung nichtfunktionaler Anforderungen ist aufwendig und schwierig. Daher werden sie meist nicht mit der nötigen Sorgfalt behandelt, oft ganz weggelassen oder nur durch Schlagwörter ausgedrückt.
- Man sollte versuchen, die nichtfunktionalen Anforderungen so präzise wie möglich zu erfassen.

Präzisierung nichtfunktionaler Anforderungen

Schlagwort	Präzisierte Anforderung
einfach bedienbar	Das Programm soll auch von Laien ohne weitere Einweisung benutzt werden können.
robust	Eine Bedienung über die Tastatur darf unter keinen Umständen zu einem irregulären Abbruch des Programms führen.
portabel	Das Spiel muss auf Windows 7, 8, 10 und 11 ausgeführt werden können, ohne dass sich das Verhalten aus Benutzersicht ändert.
übersichtlich kommentiert	Die Module des Programms müssen einen Kopfkommentar nach Std. xyz enthalten.
plattformunabhängig	Das Programm soll in einer virtuellen Maschine (z.B JVM) auf Windows, Linux und Mac laufen. Eventuelle prozessor-spezifische Teile müssen in einem speziellen Modul liegen.
nicht zu große Module	Klassen dürfen max. 300 Zeilen ausführbaren Code enthalten.
angemessenes Handbuch	Das Benutzungshandbuch wird gemäß Richtlinie ABC aufgebaut. Es wird vom Gutachter N.N. geprüft.
performant	Bei einer Last von 100 Aufrufen pro Minute muss der Server eine Anfrage innerhalb von 200ms verarbeiten.

Der Nutzen der Spezifikation

Die Spezifikation sammelt alle maßgeblichen Anforderungen.

Sie ist notwendig für

1. die Abstimmung mit dem Kunden,
2. den Entwurf und die Implementierung,
3. das Benutzungshandbuch,
4. die Testvorbereitung,
5. die Abnahme,
6. die Wiederverwendung,
7. die Klärung späterer Einwände, Regressansprüche usw.,
8. eine spätere Re-Implementierung.

„Spezifikation im Kopf“ gibt es nicht!

Ermittlung der Betroffenen

- Die Anforderungen stehen nicht allein im Raum, sondern sind die Interessen der Menschen, die direkt oder indirekt mit dem System zu tun haben
- Die Systembetroffenen oder Interessenhalter bezeichnet man auch mit dem englischen Begriff **Stakeholder**
  - Sie sind die wichtigsten Quellen für Anforderungen
  - Das Übersehen eines Stakeholders führt zu lückenhaften Anforderungsdokumenten
- Beispiele für Stakeholder:
  - Benutzer
  - Administrator
  - Management
  - Kunden
  - Betriebsrat
- Einige Stakeholder werden das fertige System nie benutzen!

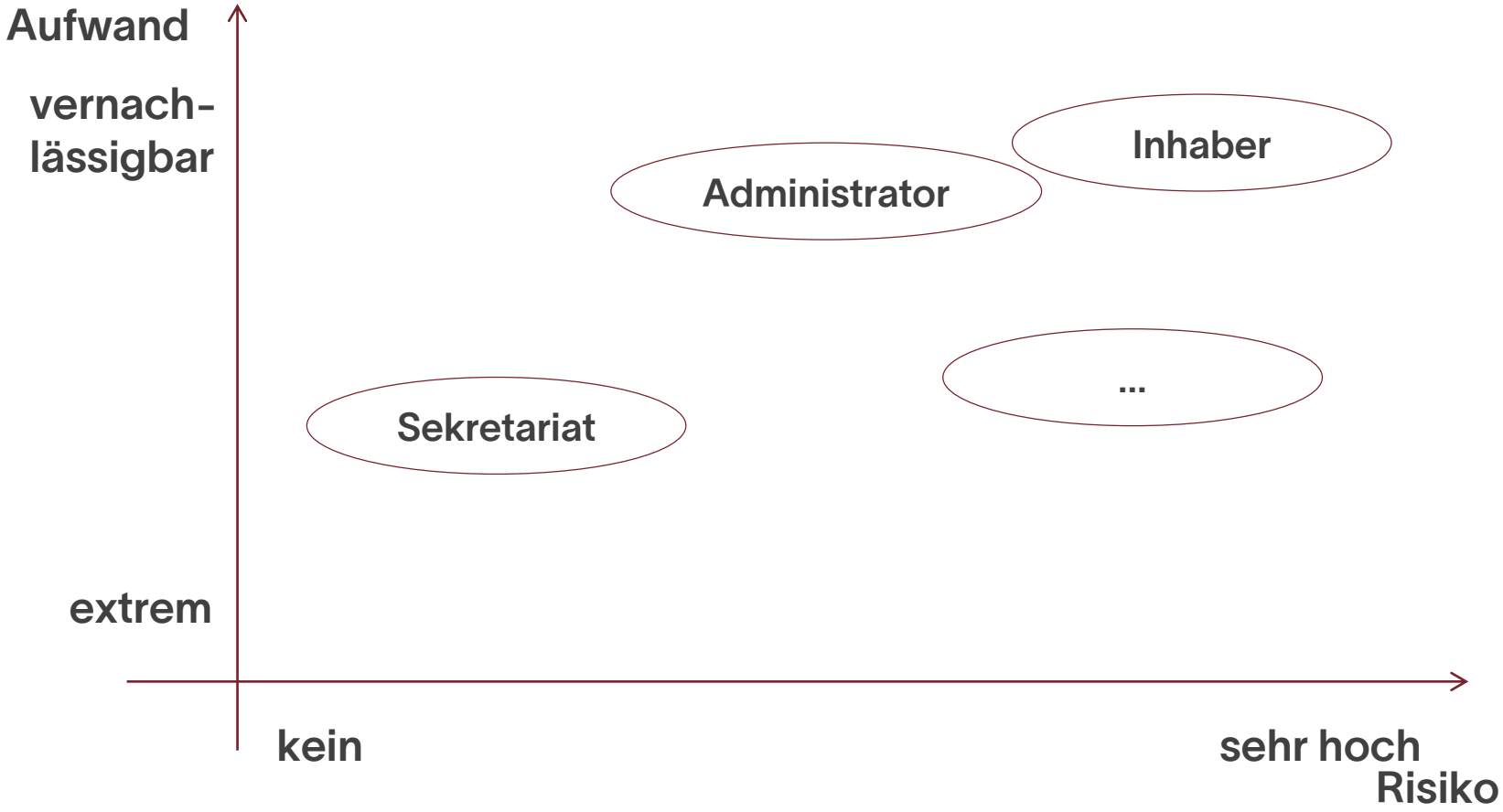
Stakeholder

Stakeholder: Ein Stakeholder eines Systems ist eine Person oder Organisation, die (direkt oder indirekt) Einfluss auf die Anforderungen des betrachteten Systems hat.

(Chris Rupp, Requirements Engineering, 2010)

- Idealerweise sollte man mit allen Stakeholder-Gruppen (natürlich nicht mit allen Einzelpersonen) sprechen, um ihre Anforderungen an das System zu erfragen
- Praktisch ist dies nicht immer möglich (z.B. Kunden, die es noch gar nicht gibt) oder zu aufwändig
- Daher kann man die in Frage kommenden Stakeholder graphisch priorisieren

Risiko-Aufwand-Matrix



(in Anlehnung an Oestereich/Scheithauer: Analyse und Design mit der UML 2.5, Oldenbourg Verlag 2013, S. 50)

Vorgehen bei der Anforderungsanalyse

- Anforderungen **ermitteln**
  - Wichtigste Quelle: Stakeholder
  - Außerdem Dokumente und bestehende Systeme
  - Techniken:
    - Befragung, gemeinsame Workshops
    - Dokumentenanalyse
    - Beobachtung

- Anforderungen **dokumentieren**
  - Basis für die Systementwicklung
  - Rechtlich relevant für Vertrag
  - Verfügbarkeit für alle Projektbeteiligten sicherstellen
  - aktuell halten

Vorgehen bei der Anforderungsanalyse

- Anforderungen **prüfen**

  Qualität der Anforderungen sicherstellen:

  - klar
  - vollständig
  - widerspruchsfrei

- Anforderungen **abstimmen**
  - Mit den Stakeholdern Übereinstimmung darüber erzielen, welche Anforderungen für die Entwicklung zu Grunde gelegt werden sollen
  - Uneinigkeit unter den Stakeholdern (Konflikte beim Kunden) nicht ignorieren

Ist-Analyse

- Eigentliches Ziel der Analyse: Soll-Zustand feststellen
  - Es sollte also genügen, die Stakeholder nach ihren Wünschen zu fragen.
  - Aber: Die Betroffenen konzentrieren sich auf das, was ihnen derzeit nicht gefällt.

➔ Zunächst Ist-Analyse durchführen!

- Wir sind bei jeder Umstellung auf die Schwachpunkte des bestehenden Systems fixiert
  - Stärken nehmen wir erst wahr, wenn sie uns fehlen.
  - Implizit wird erwartet, dass alles, was bisher akzeptabel oder gut war, unverändert bleibt oder besser wird. Die Anforderungen an das neue System bestehen also nur zum Teil aus Änderungswünschen, viele Anforderungen gehen in Richtung Kontinuität.
- Die Ist-Analyse ist eine undankbare Tätigkeit!

Beispiel

Sie öffnen also morgens das Schloss am Haupteingang?

Ja, habe ich Ihnen doch gesagt.

Jeden Morgen?

Natürlich.

Auch am Wochenende?

Nein, am Wochenende bleibt der Eingang zu.

Und während der Betriebsferien?

Da bleibt er natürlich auch zu.

Und wenn Sie krank sind oder Urlaub haben?

Dann macht das Herr X.

Und wenn auch Herr X ausfällt?

Dann klopft irgendwann ein Kunde ans Fenster, weil er nicht reinkommt.

Was bedeutet „morgens“? ...

Die Soll-Analyse

- Sinnvoll:
  - Alle Wünsche sammeln, nicht kritisieren („Wunschzettel“)
  - Widersprüche offenlegen
- Wünsche sind technisch nicht realisierbar oder zu teuer:
  - Anforderungen müssen reduziert oder Mittel aufgestockt werden.
- Gefahr: Der Analytiker wird zum Prellbock zwischen
  - den Klienten (mit großen Wünschen) und
  - den Entwicklern (mit der Sorge, die Wünsche nicht erfüllen zu können, und dem Wunsch, eigene Vorstellungen zu realisieren).
- Konflikte zwischen Wünschen der Klienten:
  - Probleme vor Klienten ansprechen und auf Einigung drängen.

Anwendungsfälle

- Häufig orientiert man sich zu Beginn der Analyse an den **Anwendungsfällen** des Systems
- Ein Anwendungsfall ist ein (abstraktes) Szenario, in dem ein Akteur versucht, mit Hilfe des betrachteten Systems ein bestimmtes fachliches Ziel zu erreichen
- Man benennt Anwendungsfälle danach, wie das Ziel aus Benutzersicht heißt, z.B.
  - Kunden anlegen
  - Auto reservieren
- Wir betrachten hauptsächlich Geschäftsanwendungsfälle, also Abläufe, die direkt oder indirekt einem geschäftlichen Ziel dienen (wie die Beispiele oben).
  Kein Geschäftsanwendungfall wäre:
  - Suchergebnisse anzeigen

Anwendungsfälle: Ergebnisse eines Workshops

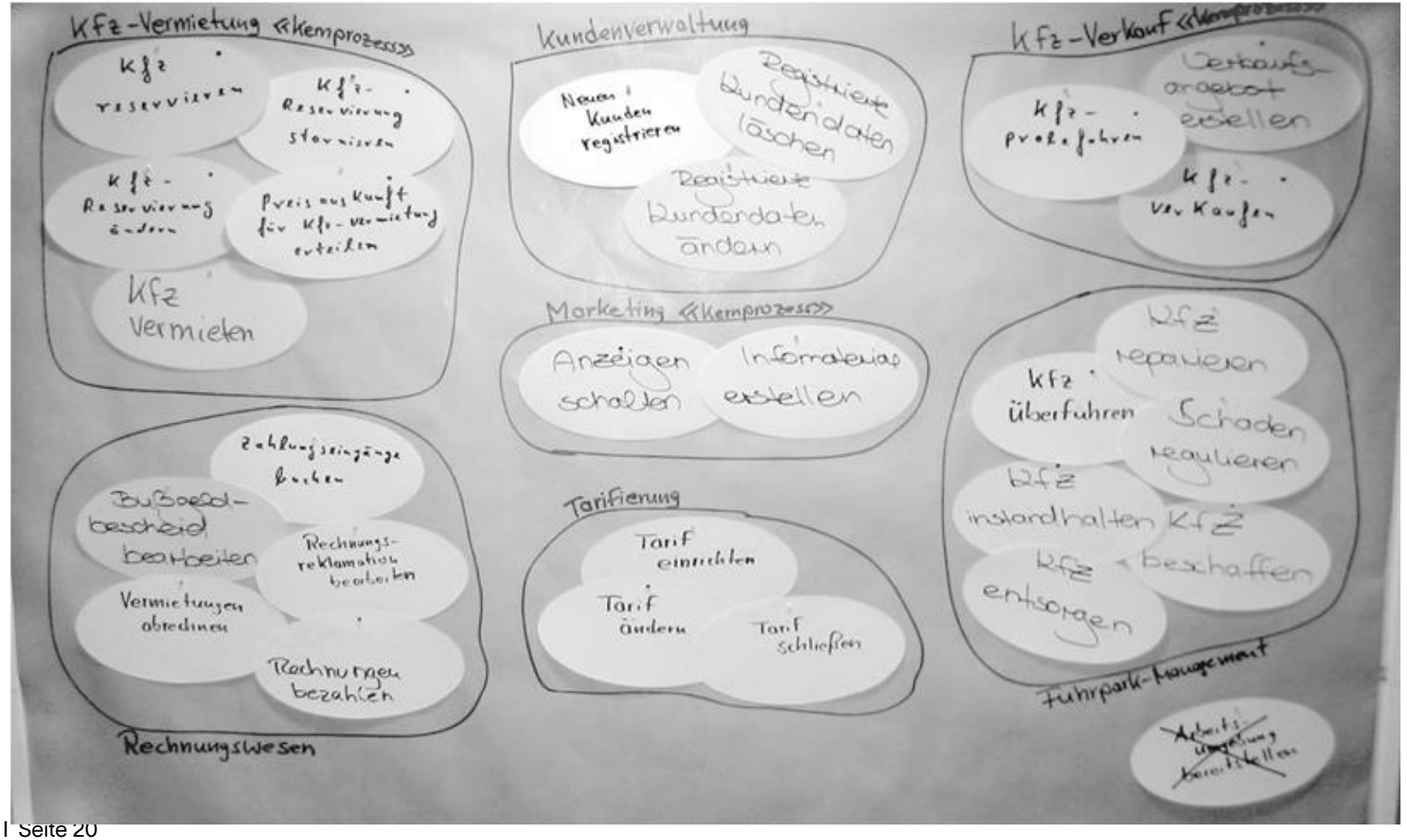
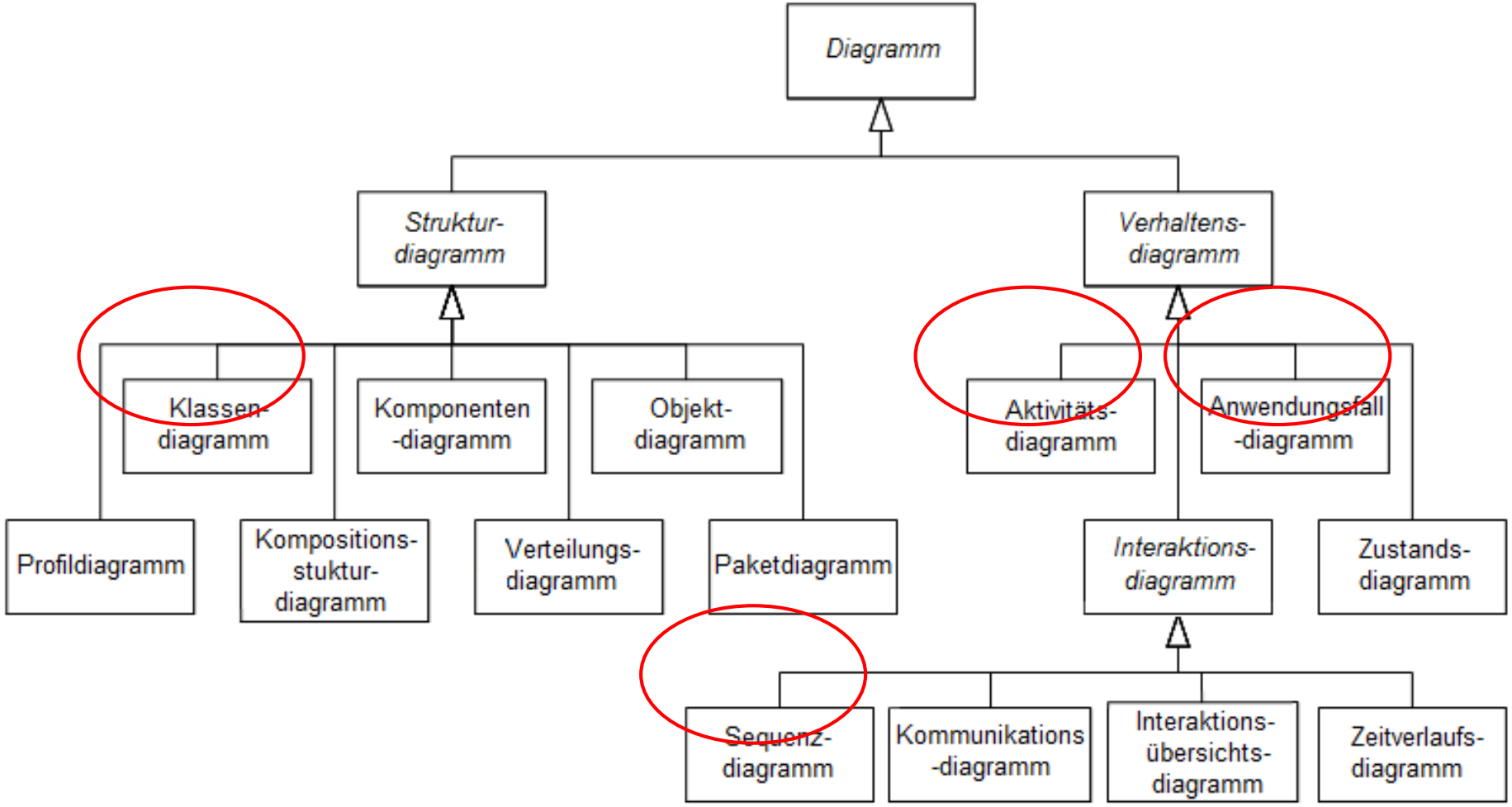


Bild: Oestereich et. al.: Objektorientierte Geschäftsprozessmodellierung mit der UML. dpunkt Verlag 2003 , S.78

2.2 Objektorientierte Analyse mit UML

- Durch die Komplexität von Softwareprojekten ist man auf Softwaremodelle angewiesen.
- Die **Unified Modeling Language (UML)** ist eine graphische Modellierungssprache zur Spezifikation, Konstruktion und Dokumentation von (objektorientierten) Softwaresystemen.
- UML definiert Bezeichner für die meisten bei einer Modellierung wichtigen Begriffe und legt mögliche Beziehungen zwischen diesen Begriffen fest.
- UML ist *die* dominierende Modellierungssprache in der Softwareentwicklung
  - Weltweiter Standard
  - Lingua franca (Verkehrssprache) für den gesamten Softwarelebenszyklus (Anforderungen bis Test), dabei plattform- und bereichsunabhängig

UML im Überblick



Die 80/20 Regel im Umgang mit der UML

"80% of most problems can be modeled with 20% of UML."

Grady Booch *et al., The Unified Modeling Language User Guide*

Was UML nicht ist

- UML ist keine Methode, sondern “nur” eine Notation!
- Die unterschiedlichen Diagramme stellen quasi Werkzeuge in einem Werkzeugkasten dar.
- Wie bei jedem Werkzeug ist der Einsatz eines Diagramms nur dann sinnvoll, wenn es “richtig” eingesetzt wird!

Einsatz der UML in der Analysephase

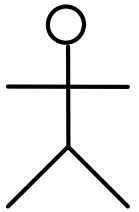
- **Ausgangspunkt** sind die Produktanforderungen des Kunden (entweder als Fließtext oder als strukturiertes Lastenheft)
- **Ergebnis** ist ein (erweitertes) **Pflichtenheft** mit semiformalen Definitionen der Produktfunktionen und -daten als UML-Diagramme
  - zunächst werden typische Anwendungsfälle grob skizziert als “Use Cases”
  - dann werden die Anwendungsfälle durch den Einsatz von Aktivitätsdiagrammen semiformal beschrieben und verfeinert
  - außerdem wird ein Fachklassenmodell erstellt, das die wichtigsten Gegenstände, Konzepte und Personen des Systems und ihre Beziehungen abbildet.

2.2.1 Anwendungsfalldiagramme

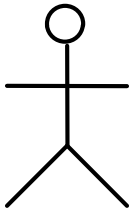
- Die Funktionen des Softwareprodukts werden anhand konkreter Fallbeispiele detailiert beschrieben.
- Vorgehensweise:
  - Skizze ganz konkreter Benutzungsszenarien mit Akteuren (Actors)
  - Zusammenfassung ähnlicher Szenarien zu Anwendungsfällen (Use Cases)
  - Erstellung von Anwendungsfalldiagrammen mit “Use Cases” und “Actors”
  - Anwendungsfälle werden bzgl. ihrer Wichtigkeit für den Kunden kategorisiert

Akteure

- Ein Akteur ist eine externe Entität, die mit dem System interagiert
  - Benutzer, externe Systeme, Geräte
  - Systembusse (z.B. CAN, Ethernet)
  - Sensoren, Motoren
- Der Name des Akteurs beschreibt die Rolle, die er gegenüber dem System spielt
  (User, Administrator = ggf. dieselbe Person)
- Ein Akteur macht etwas „intelligentes“ mit dem System, d.h. ein „button“ ist kein actor.



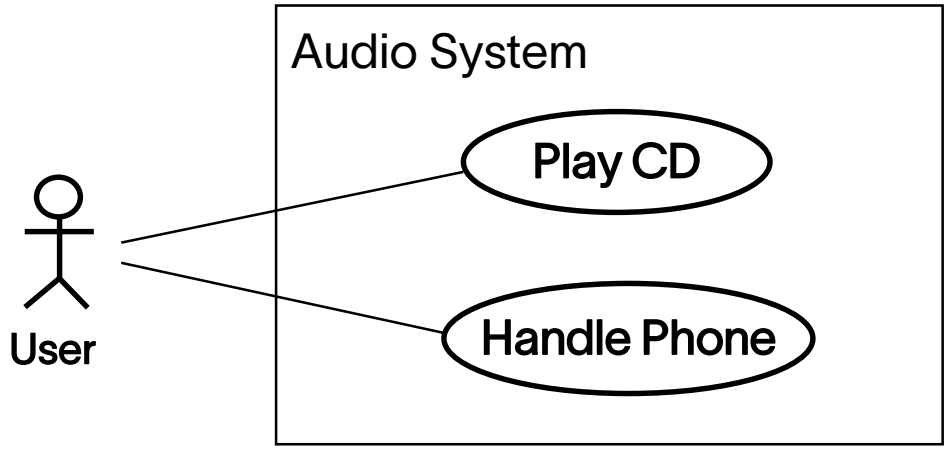
User



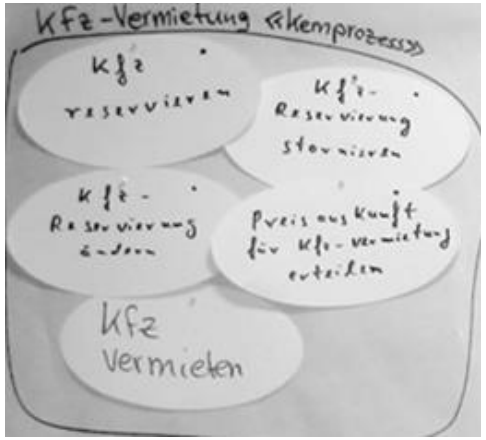
Administrator

Use Case (Anwendungsfall)

- Ein **use case** beschreibt eine zusammenhängende signifikante Teilfunktionalität des Systems
  - Er beschreibt nicht wie diese Funktionalität realisiert wird!
- Jeder Use Case wird von einem Akteur ausgelöst und wird nach dem Ziel des Akteurs benannt
- Beschrieben werden mögliche Interaktionssequenzen
  - die zum Ziel führen
  - aber auch Fehlerfälle und Alternativen
- Das System wird als Kasten dargestellt.



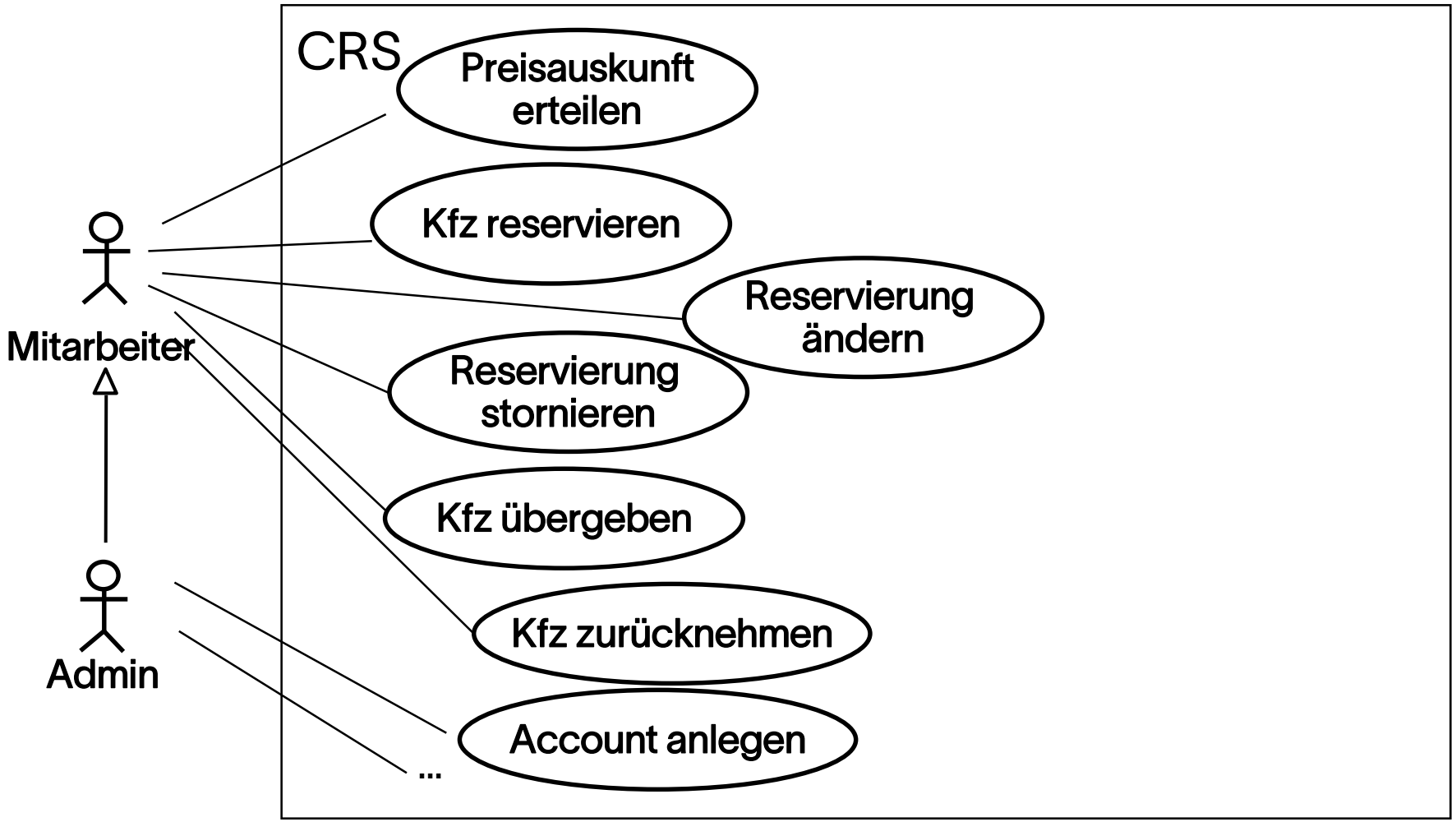
Beispiel: CRS Kernprozess



Car Rental System (CRS)

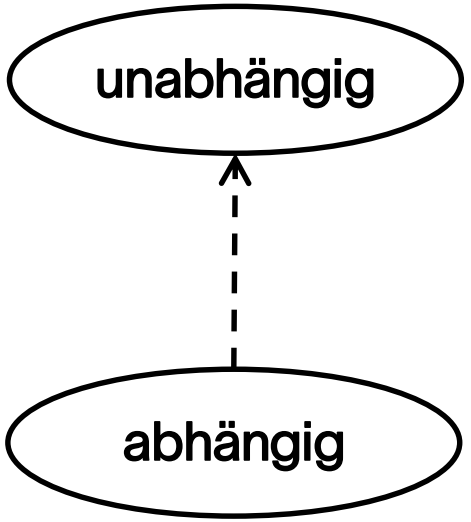
Das Car Rental System (CRS) unterstützt einen kleinen Autovermieter mit einer Niederlassung **bei allen wichtigen Geschäftsprozessen. Es verwaltet den Bestand an Fahrzeugen verschiedenen Typs. Wenn Kunden sich telefonisch nach freien Fahrzeugen und den Preisen erkundigen möchten, kann der Mitarbeiter mit CRS schnell die benötigten Informationen abrufen und ein Fahrzeug eines bestimmten Typs für den gewünschten Zeitraum reservieren. Es muss sichergestellt sein, dass ein Fahrzeug der gewünschten Kategorie für den Mietzeitraum zur Verfügung steht. Reservierungen können von den Kunden geändert oder storniert werden. Innerhalb des Mietzeitraums, spätestens aber an dessen Ende, gibt der Kunde das Fahrzeug ab und zahlt die vom System automatisch erstellte Rechnung.**

Anwendungsfalldiagramm des Kernprozesses



Dependency (Abhängigkeit)

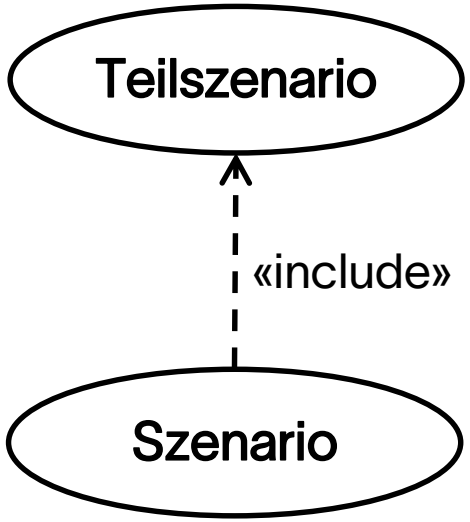
- Eine Abhängigkeitsbeziehung beschreibt die logische Abhängigkeit eines use cases von einem anderen
  - Der abhängige use case kann nicht ohne den unabhängigen betrachtet werden
  - Eine Änderung des unabhängigen use case kann Folgen für die von ihm abhängigen use cases haben
  - Dies sagt nichts über die Art der Abhängigkeit aus



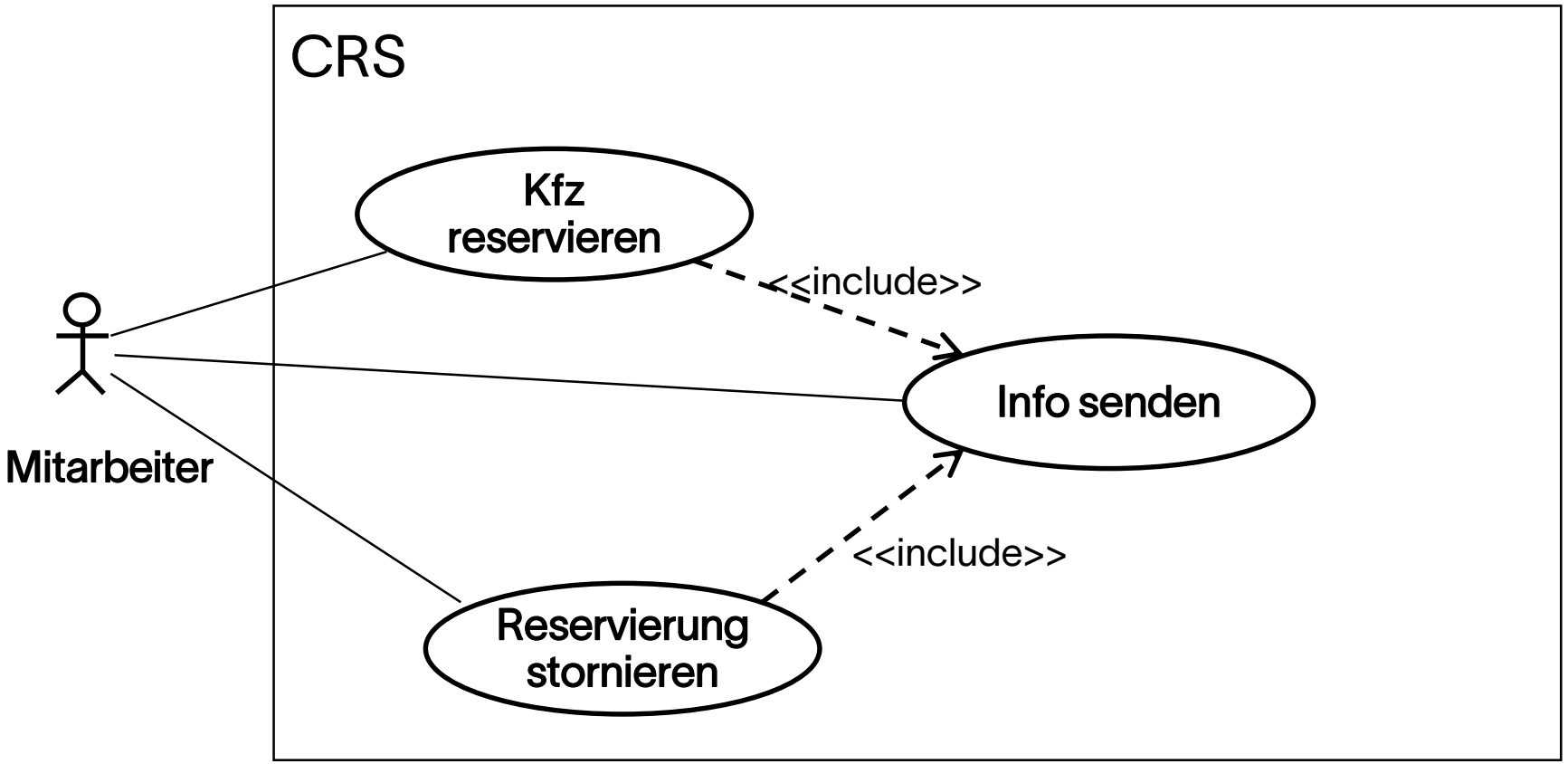
- In der Praxis werden meist die folgenden beiden speziellen dependencies benutzt:

«include»

- Besitzen mehrere Anwendungsfälle gleiche Teilabläufe wie das Verschicken einer Benachrichtigung, so können diese Teilabläufe separat beschrieben werden; es wird also Funktionalität ausgelagert und wiederverwendet (Teilszenario muss ausgeführt werden!)
- Sollte nicht verwendet werden, um Use Cases in Einzelschritte aufzuteilen (sondern nur, wenn der unabhängige use case eine eigenständige Bedeutung hat oder wiederverwendet wird)

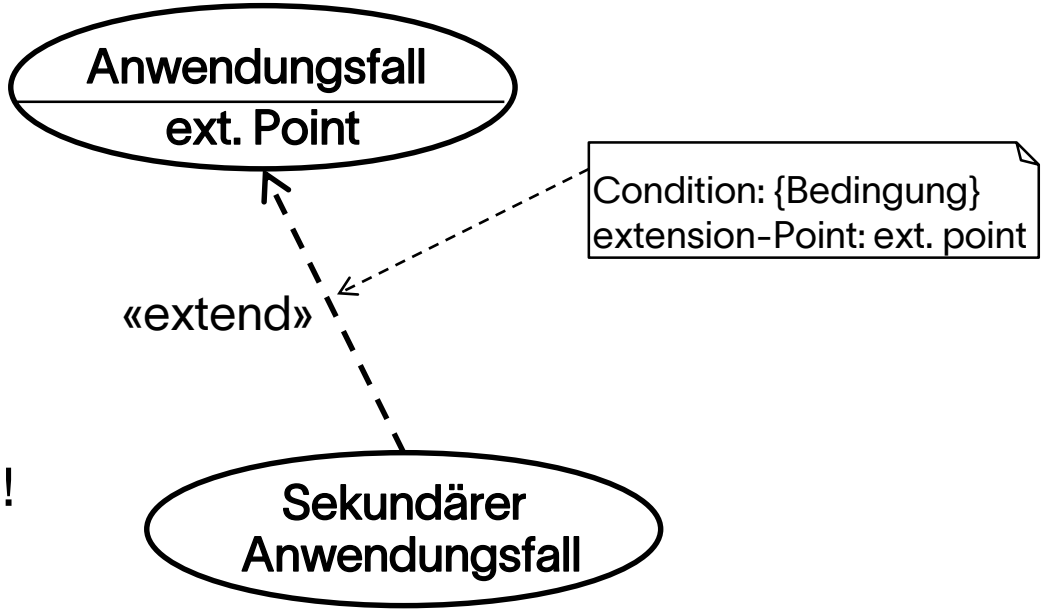


Beispiel für «include»



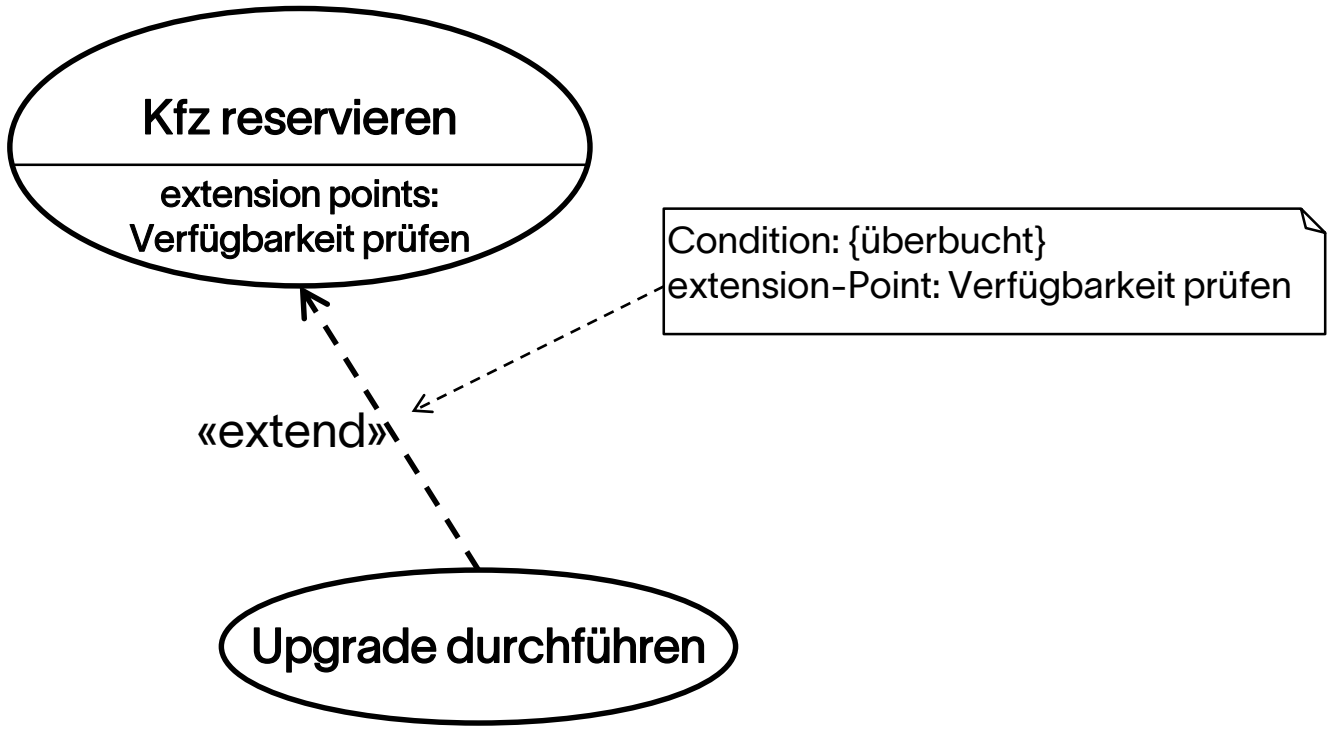
«extend»

- Sonderfälle oder Varianten eines Anwendungsfalles werden manchmal als separate Anwendungsfälle beschrieben
- Das definierte Teilszenario *kann* ausgeführt werden (unter bestimmten Bedingungen)



- Empfehlung: Sparsam benutzen!

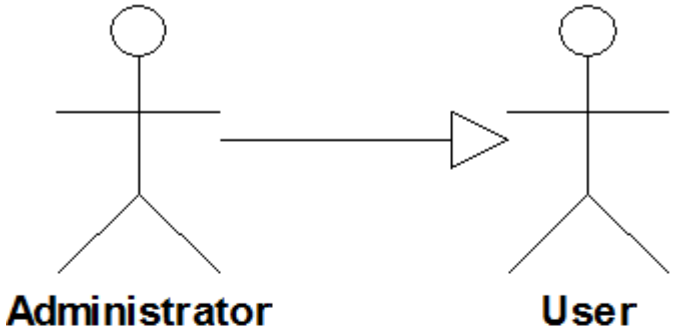
Beispiel für «extend»

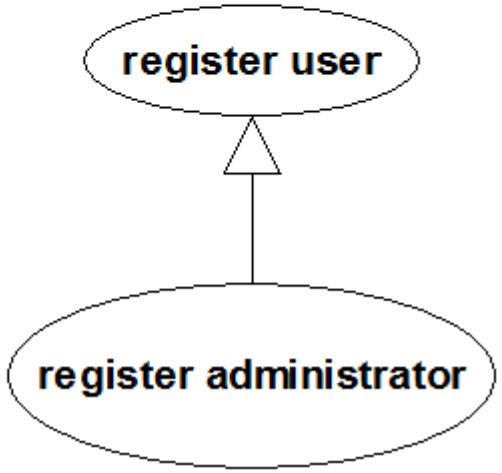


Generalisierung

- Die Generalisierungsbeziehung zwischen zwei Akteuren vererbt alle Beziehungen des “Oberakteurs” mit Anwendungsfällen auf den “Unterakteur”
- Die Generalisierungsbeziehung ist auch auf Anwendungsfällen möglich
- Abgrenzung zu «extend»:

  Die «extend»-Beziehung dient der Verhaltenserweiterung, die Bedingungen unterliegt (das Verhalten eines Use-Cases kann, muss aber nicht erweitert werden). Die Generalisierung hingegen kopiert und überschreibt das Verhalten des Basis-Use-Case.





Textuelle Anwendungsfallbeschreibung

Anwendungsfall	Kfz reservieren
Akteure	Mitarbeiter
Kurzbeschreibung	Fahrzeug der Kategorie C wird für Zeitraum T reserviert.
Auslöser	Kunde wendet sich an das Unternehmen, um ein Fahrzeug zu reservieren
Ergebnis	Für den Kunden wurde ein Fahrzeug reserviert
Vorbedingungen	Keine
Nachbedingungen	Reservierung ist in der Datenbank erfasst
Schritte	- Kunden identifizieren - Reservierungswunsch aufnehmen - Reservierungsmöglichkeit prüfen - Fahrzeug reservieren - Reservierung bestätigen
Priorität	sehr hoch

Kategorisierung (ranking) von Anwendungsfällen

- Die Vergabe von Prioritäten an Anwendungsfälle wird für die Projektplanung benutzt. Anwendungsfälle mit höherer Priorität werden in früheren Iterationszyklen (Produktgenerationen) realisiert als Anwendungsfälle mit niedrigerer Priorität.
- Anwendungsfälle haben hohe Priorität, wenn
  - sie großen Einfluss auf die Architektur des Softwareproduktes besitzen
  - zeitkritische, sicherheitskritische, komplexe, ... Funktionen beschreiben
  - mit Hilfe neuer (riskanter) Technologien zu realisieren sind
  - (über-)lebensnotwendige Geschäftsfunktionen darstellen
  - sie erhöhten Gewinn versprechen

Modellierung von Anwendungsfällen

- Wiederkehrende Teile von Anwendungsfällen werden über die include-Beziehung in eigene Anwendungsfälle ausgelagert (aber nur, wenn wirklich notwendig)
- Die include-Beziehung hat die Semantik eines Prozeduraufrufes und muss in der Beschreibung des aufrufenden Anwendungsfalles aufgeführt werden
- Ein Anwendungsfall beschreibt den Normalfall eines konkreten Ablaufes
- Ein Anwendungsfall abstrahiert von einer Anzahl konkreter Szenarien (wie etwa "Kunde Franklin Turtle reserviert für den 28. April 2023 einen Smart")
- Nur signifikante Teilfunktionalitäten von geschäftlicher Relevanz sind Anwendungsfälle

Was charakterisiert ein Use Case Modell?

- Es ist nicht objektorientiert (trotzdem werden objektorientierte Konzepte verwendet)
- Es ist sehr abstrakt und wenig formal
- Es beschreibt nur WAS das System macht, nicht WIE es das tut (Blackbox Sicht, d.h. es werden keine internen Abläufe beschrieben!)
  - es sollen keine Entwurfsentscheidungen vorweg genommen werden
  - der Kunde soll die Realisierung der Funktionalität nicht erfahren
- Es liegt auf der Grenze zwischen
  - informeller und formaler Beschreibung
  - textueller und graphischer Modellierung
  - Kunde und Entwickler
- Es wird häufig als Bestandteil des Vertrages verwendet.

Checkliste nach Cockburn (gekürzt)

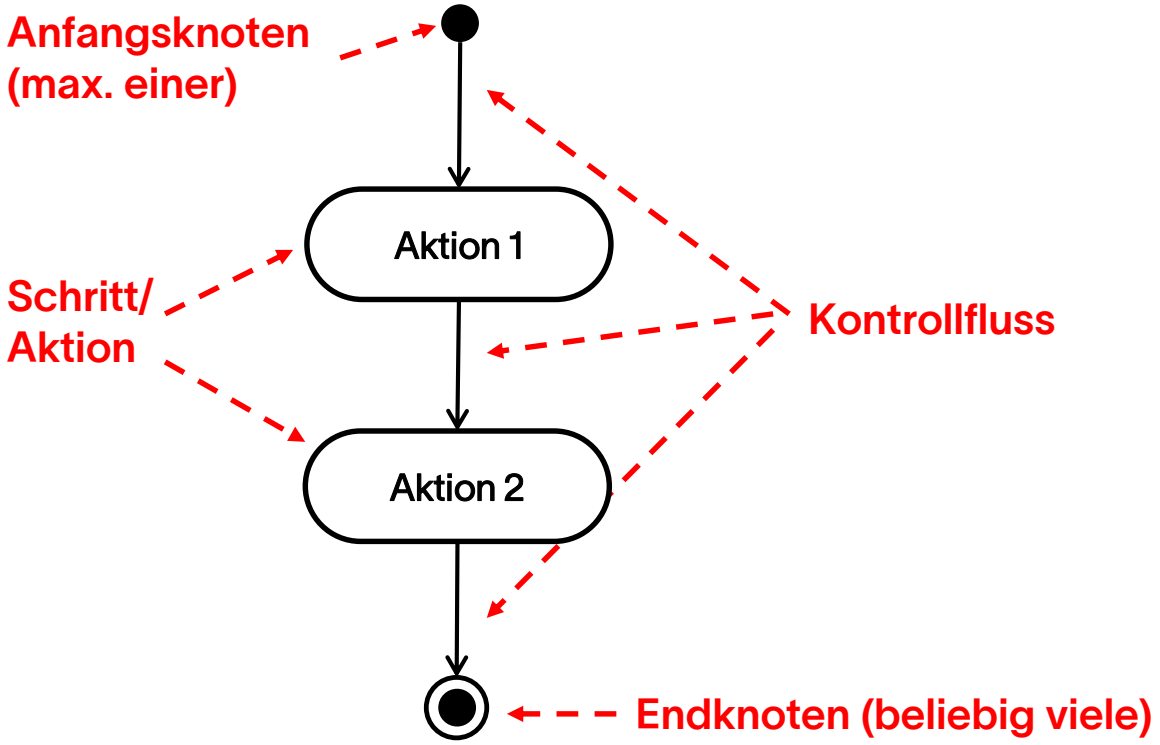
Feld	Frage
Use-Case Titel	• Ist eine Zielformulierung mit aktivem Verb vorhanden, die das Ziel des Primärakteurs nennt? • Kann das System dieses Ziel erreichen?
Akteur	• Liegt ein Verhalten des Akteurs vor? • Verfolgt er ein Ziel, das zum Lieferumfang des Systems gehört?
Vorbedingungen	• Sind sie obligatorisch und können sie vom System eingerichtet werden? • Stimmt es, dass sie im Use Case nicht überprüft werden?
Standardablauf	• Umfasst er 3-9 Schritte? • Zeigt er den Ablauf vom Auslöser bis zur Erfüllung der Nachbedingung im Erfolgsfall?
Nachbedingungen	• Werden alle Interessen der Stakeholder befriedigt?

Adaptiert nach: Alistair Cockburn: Use Cases Effektiv Erstellen, mitp Verlag, 2007
Dort finden sich insgesamt 31 Fragen, die sich aber auf ein anderes Template beziehen

2.2.2 Aktivitätsdiagramme

- Aktivitätsdiagramme finden ihren Einsatz sowohl in der Analyse-Phase als auch in der Entwurfs-Phase eines Projektes.
  - In der Analyse-Phase werden die Aktivitätsdiagramme genutzt, um Geschäftsprozesse genauer zu analysieren und zu modellieren. Dabei wird der zeitliche Ablauf, also das dynamische Verhalten, gezeigt.
  - In der Entwurfs-Phase bieten die Aktivitätsdiagramme die Möglichkeit der Beschreibung technischer Prozesse. (Lösungsansätze, Algorithmen).

Elemente eines Aktivitätsdiagramms



Atomare und nicht-atomare Schritte

Jeder Schritt in einem Aktivitätsdiagramm ist entweder ein

Atomarer Schritt

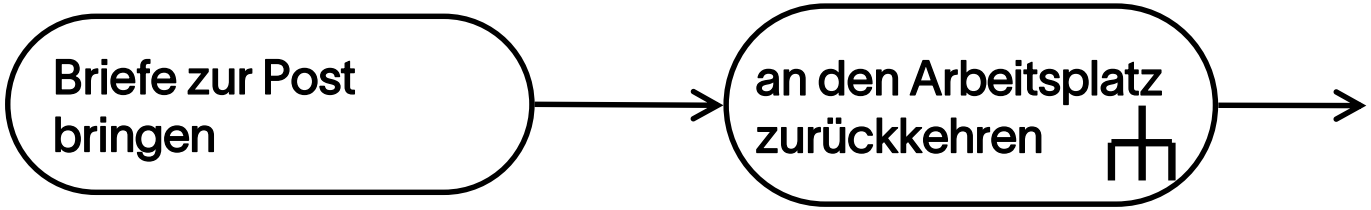
- atomar, keine Substruktur
- wird unmittelbar ausgeführt
- kann nicht unterbrochen werden
- wird bis zu Ende ausgeführt und löst automatisch den Übergang zum nächsten Schritt aus

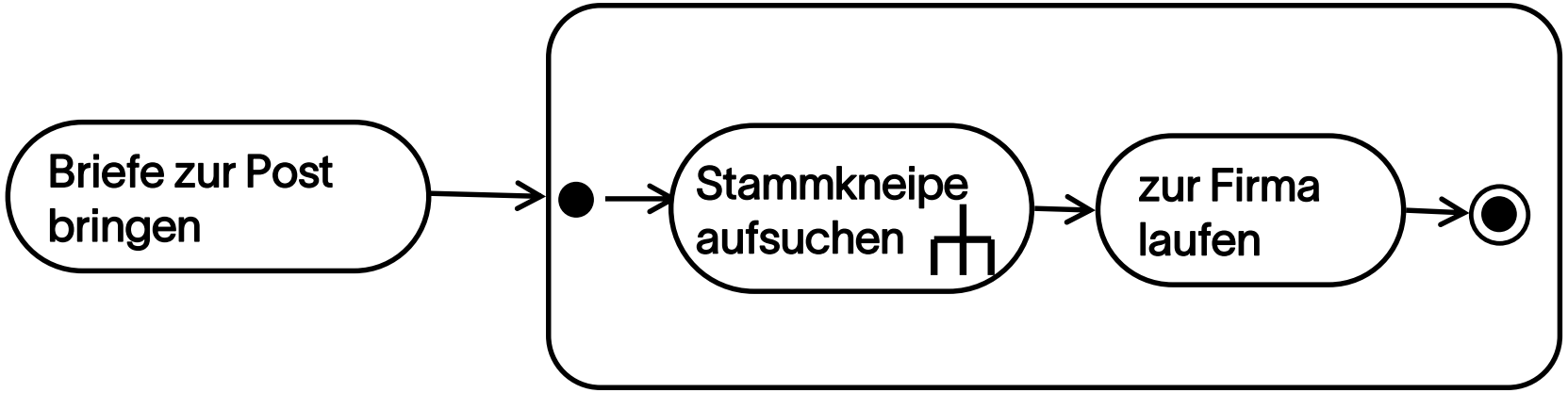
oder ein

Zusammengesetzter Schritt (Gabelsymbol)

- besitzt eine Substruktur, nicht atomar
- führt ein eingebettetes Aktivitätsdiagramm aus
- stellt eine „hierarchische Aktion“ dar

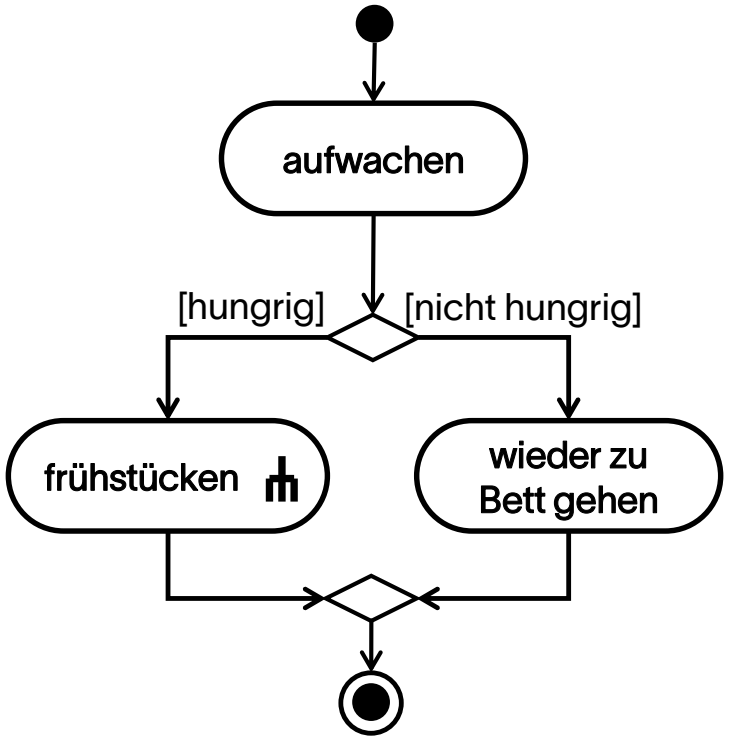
Beispiel


Briefe zur Post bringen
an den Arbeitsplatz zurückkehren


Briefe zur Post bringen
Stammkneipe aufsuchen
zur Firma laufen

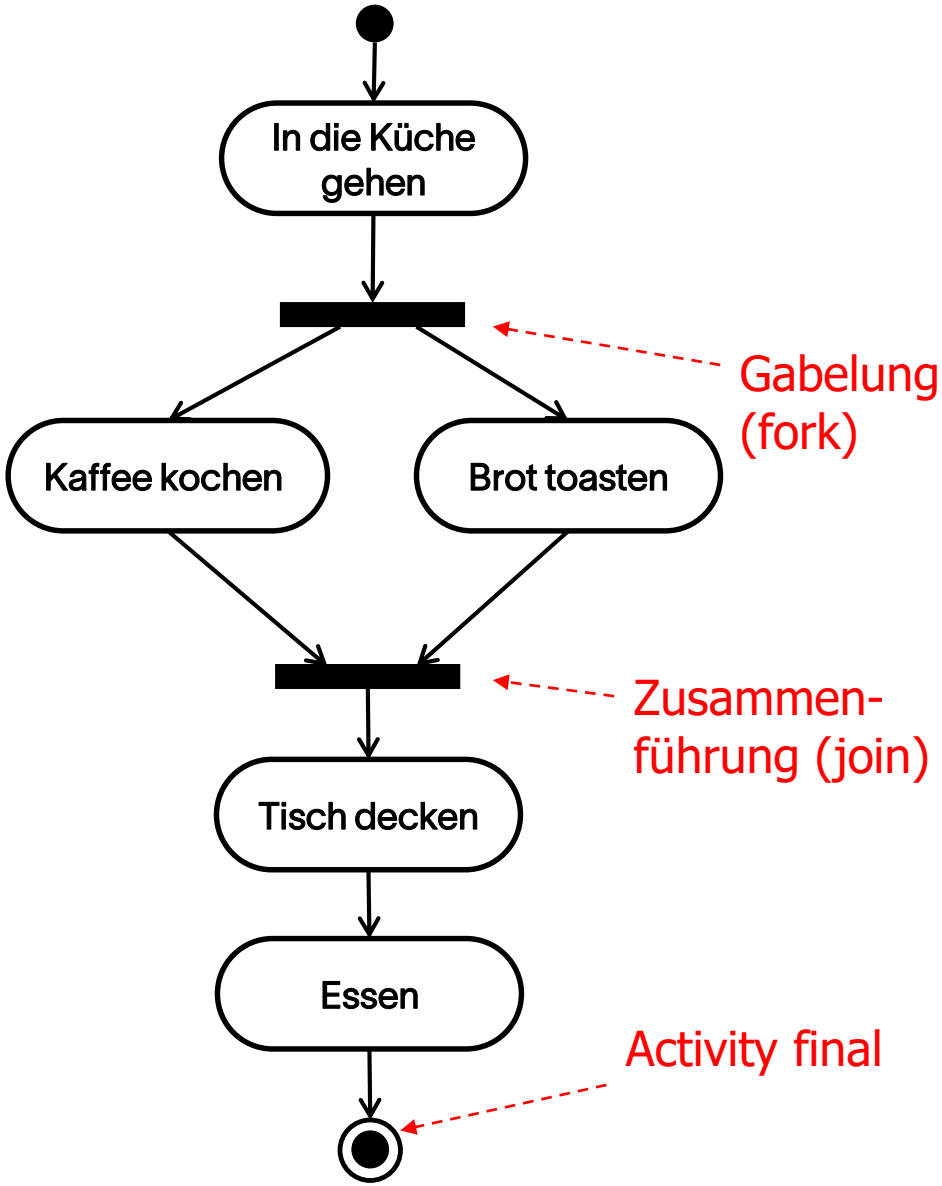
Entscheidungen

- Entscheidungen werden durch Rauten (Diamantsymbol) gekennzeichnet
- Bedingungen werden in eckigen Klammern geschrieben.
- Bedingungen müssen disjunkt sein. Immer muss mindestens eine Bedingung erfüllt sein.



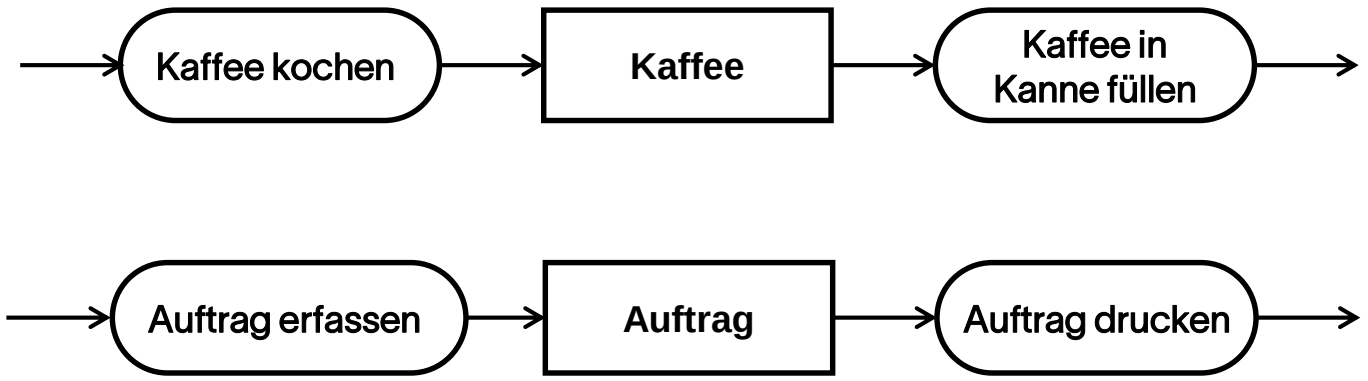
Nebenläufigkeit

- Mit Aktivitätsdiagrammen kann eine Aufteilung des Kontrollflusses modelliert werden (fork).
- Da Aktionen unabhängig voneinander sind, müssen sie explizit wieder synchronisiert werden (join).
  - Der Join wartet auf alle eingehenden Kontrollflüsse, bevor fortgefahren wird
- Nebenläufigkeit bedeutet nicht zwingend Gleichzeitigkeit!
- Alternativ zum *Activity Final* gibt es einen *Flow Final* Knoten ⊗ für das Ablaufende eines Teilkontrollflusses

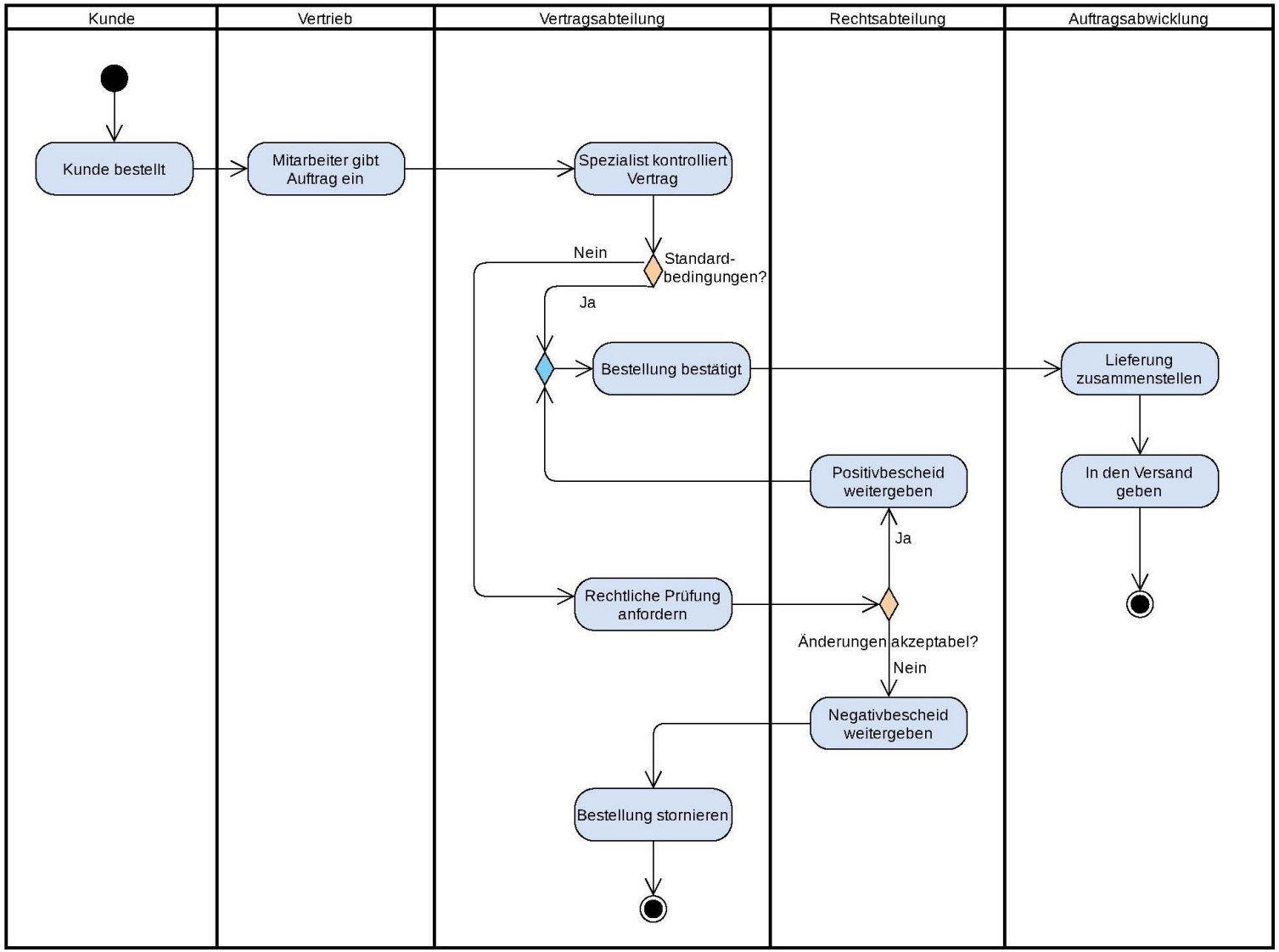


Objektknoten

- Durch Aktionen können Objekte erzeugt und manipuliert werden (Zustandsänderung)
- Die Änderungen im Datenobjekt sind Voraussetzungen für die folgende Aktion

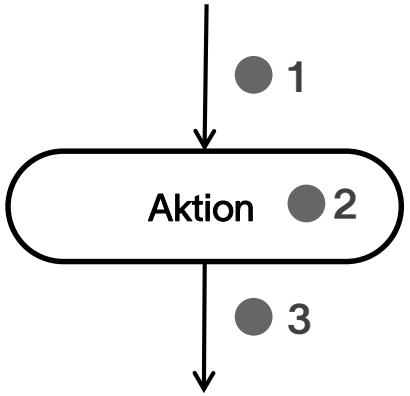


Swimlanes zur Aufteilung der Zuständigkeiten



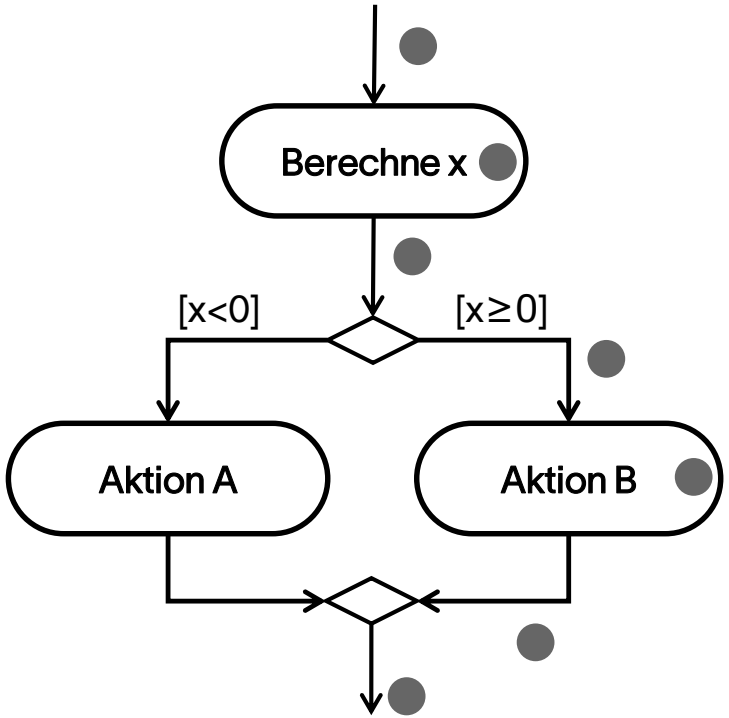
Token-Semantik

- Für ein besseres Verständnis der Abläufe in einem Aktivitätsdiagramm gibt es das Token-Konzept
- Ein Token ist eine Marke, die logisch den Punkt anzeigt, an dem sich der Ablauf gerade befindet.
- Die Wanderung des Tokens durch das Diagramm entspricht der Abarbeitung des Ablaufs
- Grundidee: Eine Aktion startet, wenn über eine Kante ein Token angeboten und aufgenommen wird (1). Es verbleibt während der Verarbeitung in der Aktion (2) und wird danach an die ausgehende Kante abgegeben (3).



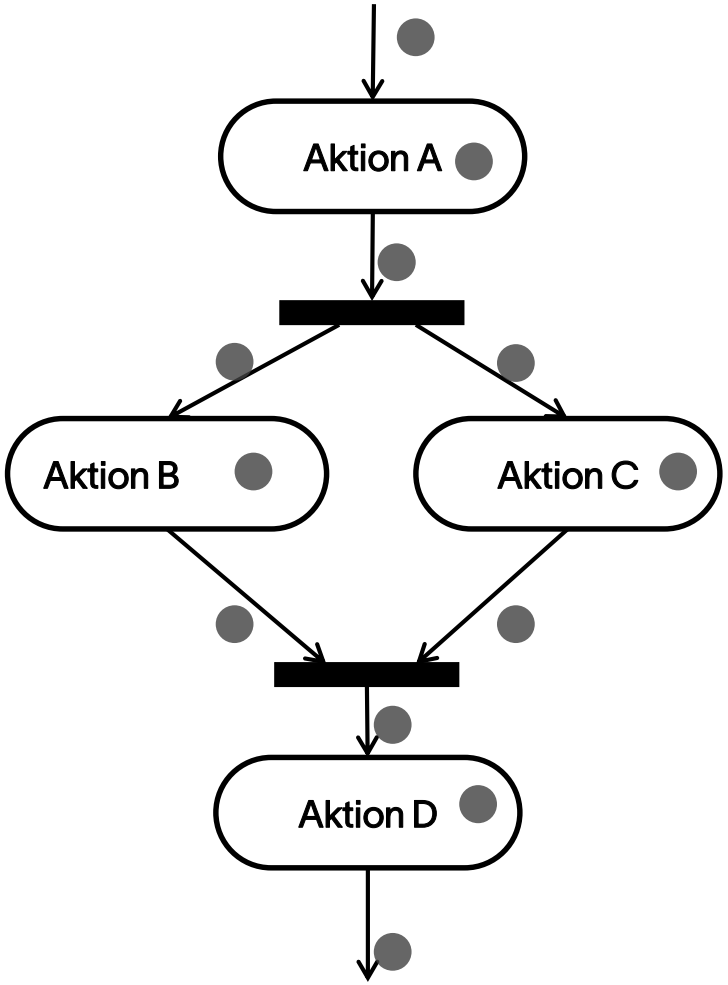
Verzweigung und Vereinigung

- Bei einer Verzweigung nimmt das Token einen der möglichen Wege
- Wenn die Alternativzweige zusammengeführt werden, reicht es, wenn ein Token über einen Zweig an der Vereinigungsraute ankommt



Nebenläufigkeit

- Anders ist es bei der Nebenläufigkeit
- Am Parallelisierungsknoten wird das Token vervielfacht (entsprechend der Anzahl der ausgehenden Kanten)
- Die einzelnen Parallelabläufe finden unabhängig voneinander statt. Über die zeitliche Reihenfolge ist keine Aussage möglich
- Am Synchronisationsknoten wird so lange gewartet, bis an *allen Kanten* Token eingehen. Erst dann werden diese aufgenommen und wieder zu einem Token verschmolzen.



Aktivitätsdiagramme in der Analyse

- Mit Aktivitätsdiagrammen (activity diagrams) wird
  - der zeitliche Zusammenhang der Bearbeitungsschritte modelliert
  - es werden alle Alternativen in einem Diagramm beschrieben,
  - die beteiligten (Geschäfts-)Objekte, sowie deren Verantwortlichkeiten identifiziert.
- Vorgehensweise:
  - Auswahl des zu modellierenden Teilprozesses
  - Identifikation der beteiligten (Geschäfts-)Objekte
  - Bestimmung von Anfangs- und Endzustand des modellierten Ablaufs
  - Modellierung sequentieller/paralleler Teilabläufe
  - Auslagerung komplexer Teilschritte in Unterdiagramme

2.2.3 Klassendiagramme in der Analyse

Mit Klassendiagrammen wird ein fachliches Datenmodell des Anwendungsbereiches **(Fachklassenmodell)** erstellt.

- Es handelt sich dabei in der Regel noch nicht um das interne Datenmodell des später realisierten Softwareproduktes!

Vorgehensweise:

- Bestimmung von Objekt- bzw. bzw. Klassenkandidaten
- Bestimmung wichtiger Assoziationen (Beziehungen) zwischen den identifizierten Klassen
- meist werden noch keine Operationen bei Klassen festgelegt

Kandidaten für Objekte und Klassen

- Kandidaten sind:
  - alle physischen, berührbaren Gegenstände (wie *Fahrzeug*)
  - Personenrollen (wie *Mitarbeiter*, *Kunde*)
  - Institutionen (wie *Firma*)
  - abstrakte Begriffe (wie *Adresse*)
  - durchzuführende Transaktionen (wie *Vermietung*, *Zahlung*)
  - eintretende Ereignisse (wie *Unfall*)
  - …
- Man beachte:
  Auch Operationen, Transaktionen, Ereignisse, Prozesse, … können sinnvolle Objektkandidaten sein. Nicht nur passive Datenbestände werden als Objekte modelliert!

Identifikation von Klassenkandidaten im Text

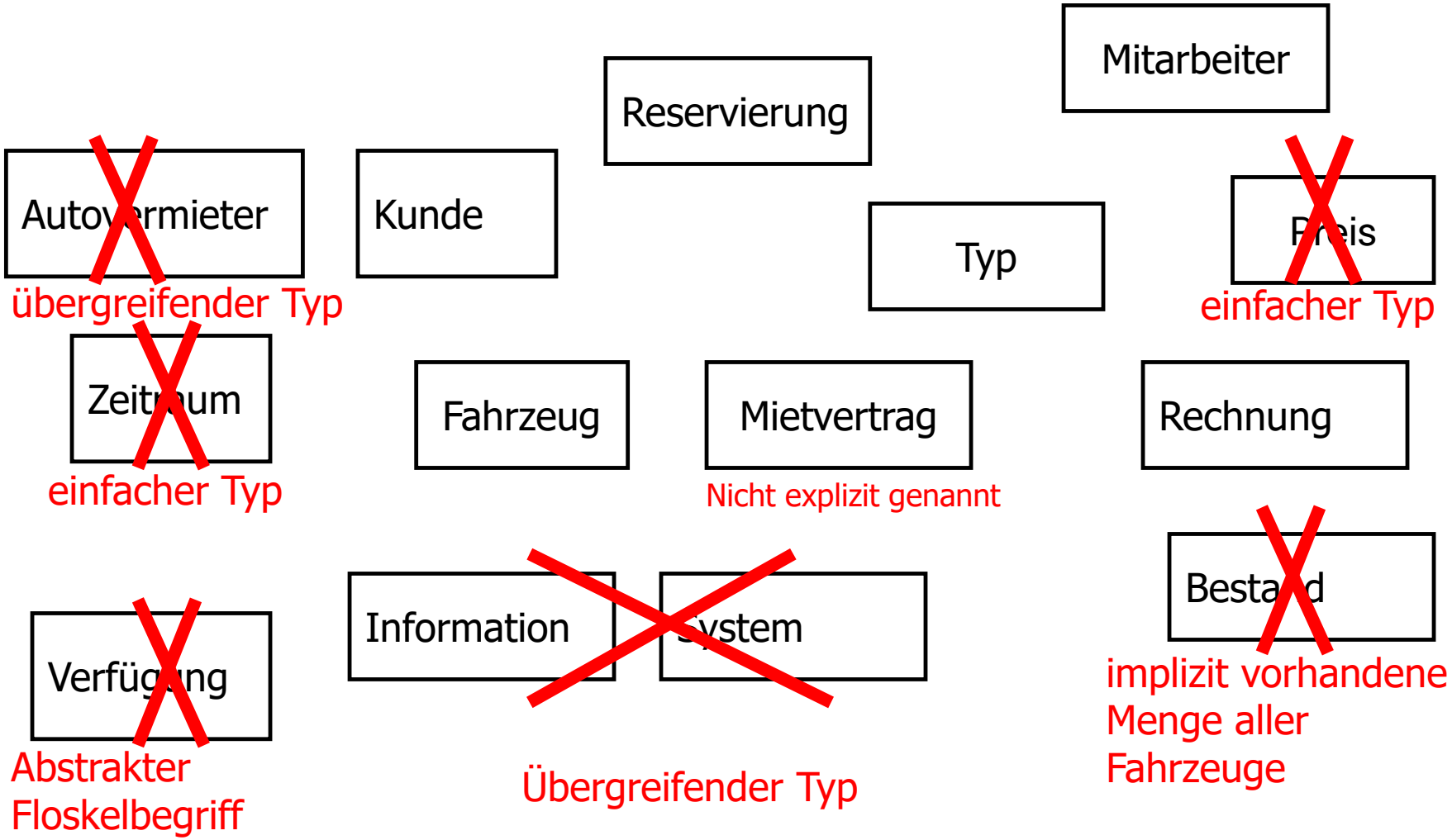
- alle relevanten Substantive markieren:

Car Rental System (CRS)

Das Car Rental System (CRS) unterstützt einen kleinen Autovermieter mit einer Niederlassung bei allen wichtigen Geschäftsprozessen. Es verwaltet den Bestand an Fahrzeugen verschiedenen Typs. Wenn Kunden sich telefonisch nach freien Fahrzeugen und den Preisen erkundigen möchten, kann der Mitarbeiter mit CRS schnell die benötigten Informationen abrufen und ein Fahrzeug eines bestimmten Typs für den gewünschten Zeitraum reservieren. Es muss sichergestellt sein, dass ein Fahrzeug der gewünschten Kategorie für den Mietzeitraum zur Verfügung steht. Reservierungen können von den Kunden geändert oder storniert werden. Innerhalb des Mietzeitraums, spätestens aber an dessen Ende, gibt der Kunde das Fahrzeug ab und zahlt die vom System automatisch erstellte Rechnung.

- Synonyme identifizieren (wie Typ und Kategorie)

Irrelevante Klassenkandidaten streichen



Zusammenarbeit der Klassen

- Die bisher identifizierten Klassen stehen lose nebeneinander
- Es fehlt noch die Beschreibung ihres Zusammenwirkens
- Auf statischer Ebene im Klassendiagramm verwendet man die Konzepte
  - Assoziation
  - Aggregation, Komposition
  - Generalisierung

- Später (in der Entwurfsphase) beschreiben wir den Nachrichtenaustausch zwischen den Objekten verschiedener Klassen im Sequenzdiagramm

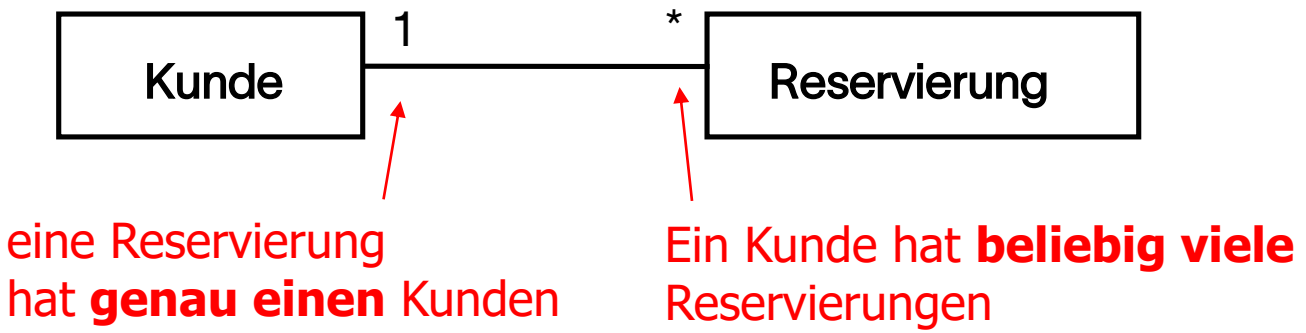
Identifikation von Assoziationen

- Assoziationen sind (in aller Regel zweistellige) Relationen zwischen Klassen. Auf Objektebene spricht man von *Links*, die die Instanzen der vorgegebenen Klassen verbinden.

- Kandidaten für Assoziationen:
  - A ist ein logischer/physikalischer Teil von B (Kunde - Firma)
  - A überwacht/besitzt B (Mietwagenverleih - Fahrzeug)
  - A benutzt B (Kunde - Fahrzeug)
  - A verweist auf B (Versicherung - Mietvertrag)
  - A kommuniziert mit B (Kunde - Mitarbeiter)

Multiplizität von Assoziationen

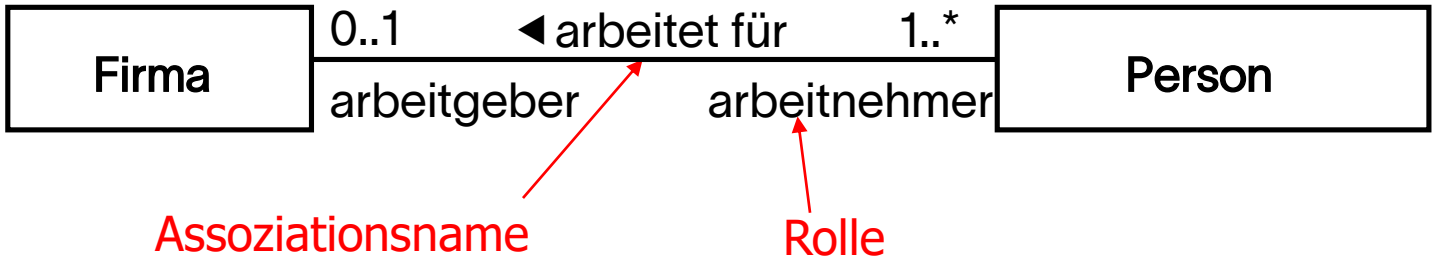
- Die Multiplizität einer Assoziation gibt an, wie viele Objekte zur Laufzeit miteinander zusammenarbeiten können



- Es kann auch eine obere und untere Grenze angegeben werden, z.B.
  - 0..1
  - 1..*
  - 28..31

Ausführliche Notation von Assoziationen

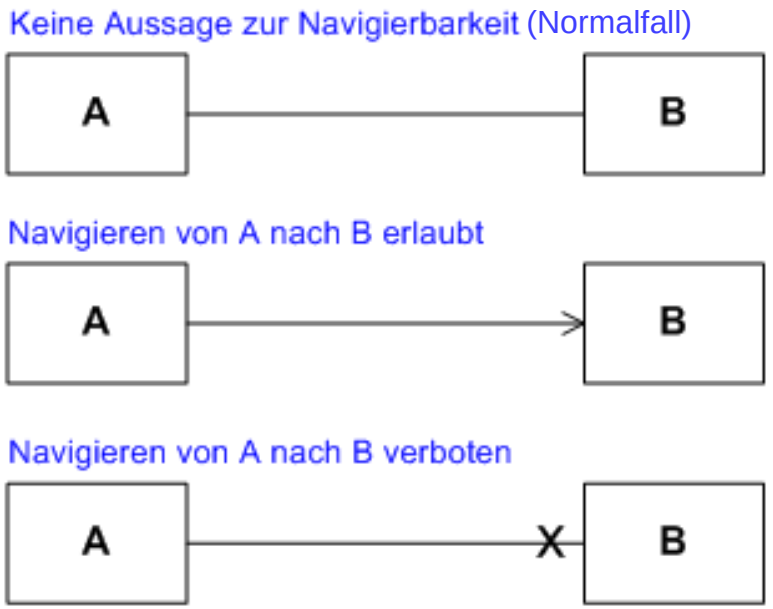
- Assoziationen können einen Namen tragen, der an die Linie geschrieben wird. Falls dieser Name nur in einer Leserichtung Sinn ergibt, kann man die Richtung durch ein kleines Dreieck anzeigen.
- Dies hat nichts mit der Navigierbarkeit (nächste Folie) zu tun!



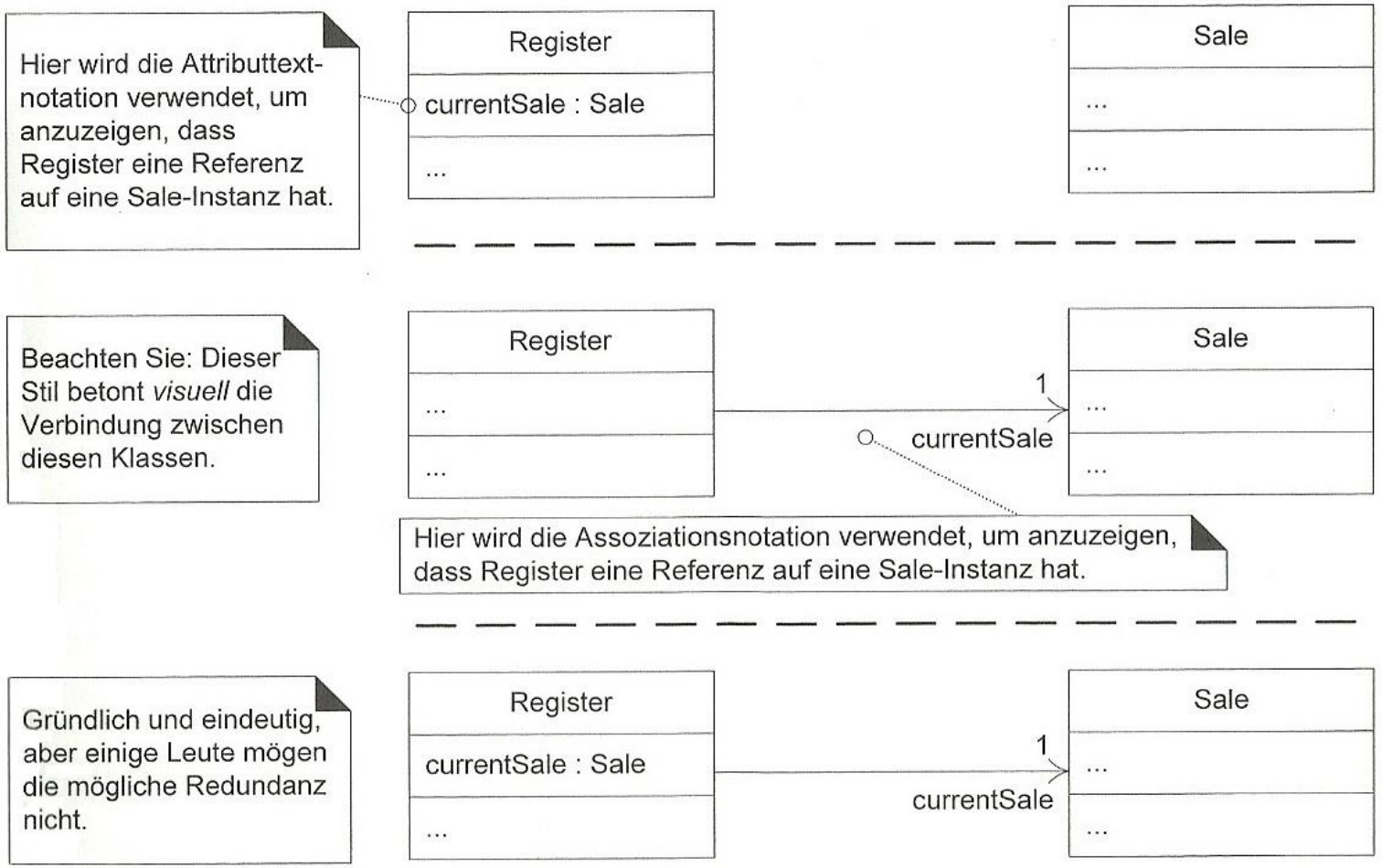
- Zusätzlich können Rollenangaben an den Enden notiert werden.

Gerichtete Assoziationen

- Bei einer gerichteten Assoziation kann von einer Klasse zur anderen navigiert werden, aber nicht umgekehrt
- Gerichtete Assoziationen werden wie gewöhnliche Assoziationen notiert, jedoch zusätzlich mit einer Pfeilspitze
- Navigierbarkeitsaussagen:

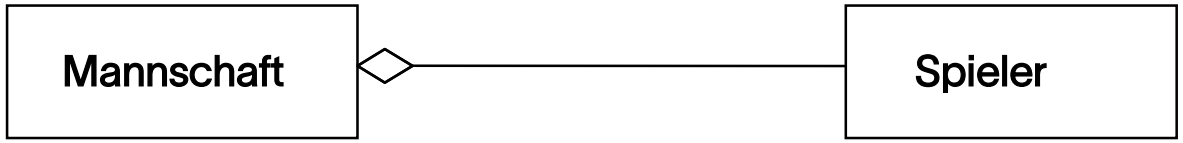


Notation von Assoziationen (Varianten)



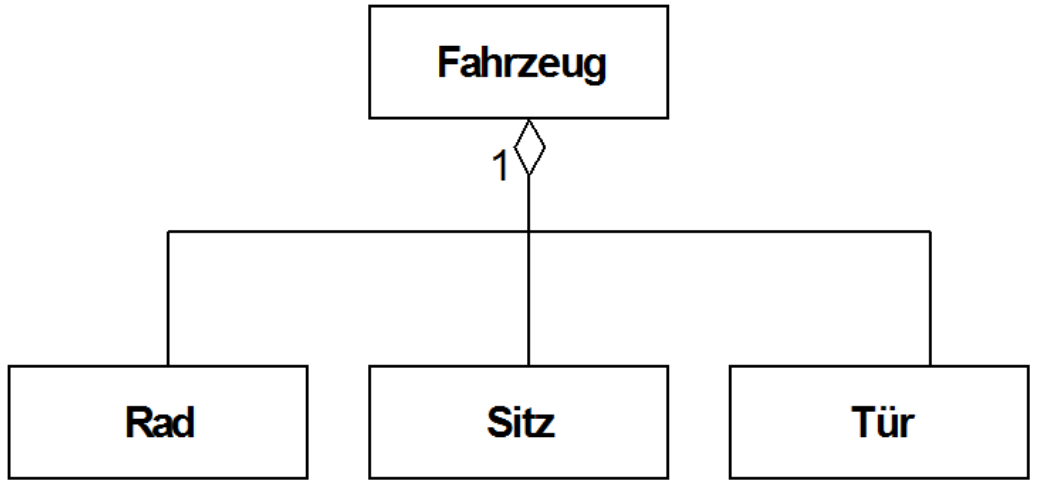
Quelle: UML 2 und Patterns angewendet, mitp-Verlag 2005

Aggregation



- binäre Assoziation mit Zusatzsemantik (ist-Teil-von-Beziehung)
- transitiver Abschluss darf keine Zyklen bilden
- ein Objekt darf Teil mehrerer Elternobjekte sein

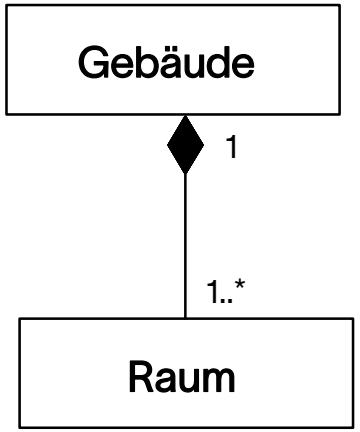
Weiteres Beispiel:



Komposition

- Spezialform („starke Form“) der Aggregation
- Das „Ganze“ ist für den Lebenszyklus seiner „Teile“ verantwortlich
- Wird das Ganze zerstört, so werden auch die Teile zerstört

Beispiel:

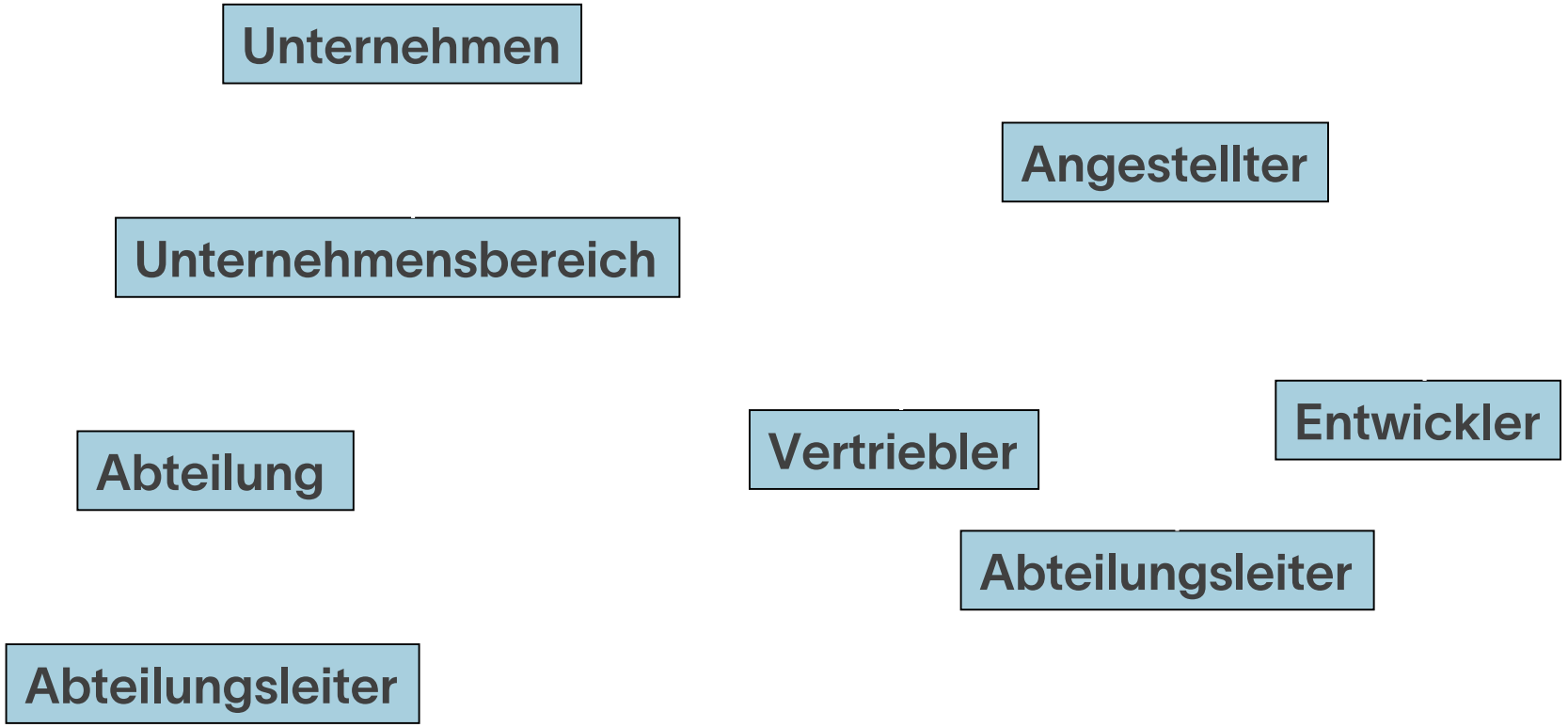


Verwendung von Aggregation und Komposition

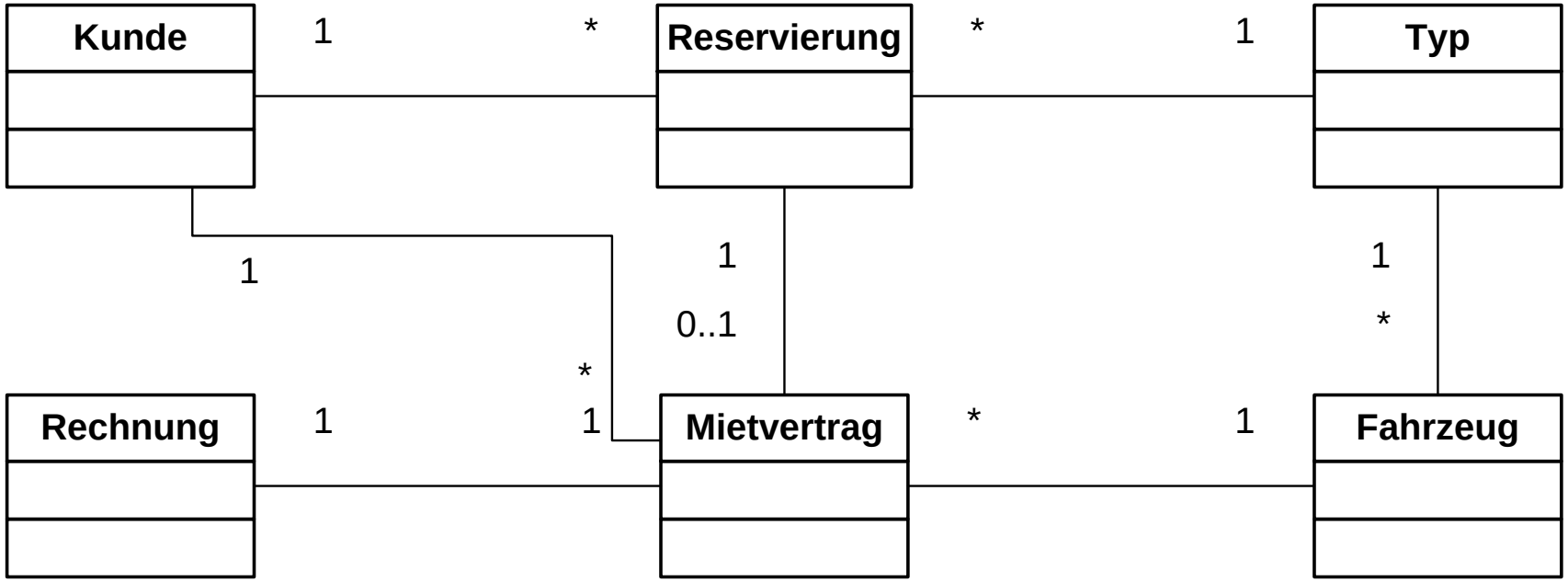
- Aggregation und Komposition werden sparsam verwendet, da sie sich kaum von “normalen” Assoziationen unterscheiden.

- Einsatz von Aggregation/Komposition ggf. wenn:
  - ein Objekt von einem anderen Objekt existenzabhängig ist (Versicherung von Mietvertag)
  - ein Objekt “offensichtlich” logischer/physikalischer Bestandteil eines anderen ist (Gesamtreservierungssystem enthält alle Daten)
  - Eigenschaften des Ganzen sich auf die Teile übertragen (Farbe eines Autos auf seine Karosserie-Bestandteile)
  - Operationen auf dem Ganzen zu den Teilen propagiert werden (so wie etwa löschen, bewegen, …)

Beispiel: Welche Beziehungen gelten hier?



Fachliches Modell des CRS



2.3 Lasten- und Pflichtenheft

- Lasten- und Pflichtenhefte sind in den Ingenieurwissen-schaften verbreitet und in VDI Norm 2519 Blatt 1 definiert
- In der Softwareentwicklung kann man die Analysephase weiter in zwei Phasen untergliedern:
  - Planungsphase
  - Definitionsphase
- In der **Planungsphase** wird durch Voruntersuchungen geprüft, ob ein Produkt entwickelt werden soll
  - Technische Durchführbarkeit (Softwaretechnik, Hardware)
  - Alternative Lösungsansätze (z.B. Kauf von Standardsoftware, Open-Source)
  - Personelle Durchführbarkeit
  - Prüfen der Risiken
  - Aufwands- und Terminschätzung
  - Wirtschaftlichkeitsrechnung

2.3.1 Lastenheft

- Als Ergebnis der Planungsphase entstehen folgende Artefakte
  - Lastenheft (Was und wofür ist etwas zu entwickeln)
  - Projektkalkulation (Personalkosten, Sachkosten)
  - Projektplanung (Personal- und Zeitplanung)
  - Glossar (Definition der wichtigsten fachlichen Begrifflichkeiten)
- Das **Lastenheft** beschreibt die fachlichen Basisanforderungen aus der Sicht des Auftraggebers

VDI 2519: Zusammenstellung aller Anforderungen des Auftraggebers hinsichtlich Liefer- und Leistungsumfang. Im Lastenheft sind die Anforderungen aus Anwendersicht einschließlich aller Randbedingungen zu beschreiben. Diese sollen quantifizierbar und prüfbar sein. Im Lastenheft wird definiert WAS und WOFÜR zu lösen ist. Das Lastenheft wird vom Auftraggeber oder in dessen Auftrag erstellt. Es dient als Ausschreibungs-, Angebots- und Vertragsgrundlage.

Lastenheft

- Im Lastenheft sind Unklarheiten und Lücken noch normal
- Erst die Anforderungsspezifikation (Pflichtenheft) muss vollständig, klar und konsistent sein.
- Obwohl es Normen für das Lastenheft gibt, bleibt der Begriff in der Praxis unscharf, weil der Inhalt des Lastenhefts von Firma zu Firma stark variiert.
- Wir verwenden als Vorlage den Aufbau eines Lastenhefts nach Balzert.

Eigenschaften eines Lastenheftes

- Adressaten: Auftraggeber und Auftragnehmer (Projektleiter, ...)
- Form: übersichtliche Gliederung, prägnante Sätze in natürlicher Sprache
- Inhalt: fundamentale Eigenschaften des Produktes aus Kundensicht
- Erstellungszeitpunkt: Ausschreibungs- und Angebotsphase
- Umfang: **wenige Seiten**

Kategorisierung der Funktionen, Daten und Leistungen:

Bei sämtlichen Funktionen, Daten und Leistungen, die im Lastenheft aufgeführt sind, ist ihre Wichtigkeit anzugeben.

Aufbau eines Lastenheftes nach Balzert

1. Zielbestimmung: welche Ziele sollen mit dem Softwareprodukt erreicht werden.
2. Produkteinsatz: Anwendungsbereiche und Zielgruppen werden genannt.
3. Produktfunktionen: Hauptfunktionen werden aus Anwendersicht (ohne Hilfsfunktionen) beschrieben und in 10er-Schritten durchnummeriert (PF nn).
4. Produktdaten: persistent gespeicherte Hauptdaten werden festgelegt und in 10er-Schritten durchnummeriert (PD nn).
5. Produktleistungen: besondere Anforderungen an Hauptfunktionen oder Hauptdaten (Ausführungszeit, Datenumfang, ...) werden aufgezählt (PL nn).
6. Qualitätsanforderungen: allgemeine Eigenschaften wie gute Zuverlässigkeit, hervorragende Benutzbarkeit, normale Effizienz, ... werden festgelegt.
7. Ergänzungen: alles was nicht in obiges Schema passt und trotzdem wichtig ist.

Das Lastenheft des Softwareprodukts

1. Zielbestimmung:

   Das Programm CRS soll eine kleine Autovermietung mit genau einer Niederlassung in die Lage versetzen, die Buchung und den Verleih ihrer Fahrzeuge zu verwalten.

2. Produkteinsatz:

   Das Produkt wird im Verkaufsraum und im Büro der Firma vom Besitzer der Firma und oft wechselnden Aushilfskräften bedient.

3. Produktfunktionen:

   /PF10/ Ersterfassung, Änderung und Löschung von Fahrzeugen.

   /PF20/ Ersterfassung, ... von Fahrzeugtypen.

   /PF30/ Ersterfassung, ... von Fahrzeugreservierungen.

   /PF40/ Ausgabe verfügbarer Fahrzeuge eines Typs in einem bestimmten Zeitintervall mit folgenden Daten: ... .

   /PF50/ ...

Das Lastenheft des Softwareprodukts

4. Produktdaten:

   /PD10/ Folgende Daten sind zu jedem Fahrzeug zu speichern: Typ, Farbe, gefahrene Kilometer, letzte Inspektion, … .

   /PD20/ Folgende Daten sind bei jeder Fahrzeugreservierung zu speichern: Zeitraum, gewünschter Typ (vorreserviertes Fahrzeug), reservierender Kunde, …

   /PD30/ …

5. Produktleistungen:

   /PL10/ Bei der Listenausgabe der Funktionen /PF40/, …werden zunächst nur die ersten n “Treffer” ausgegeben, weitere Treffer nur auf Wunsch.

   /PL20/ Das System erzwingt die regelmäßige Erstellung von Datensicherungen für die Daten /PD20/, … .

   /PL30/ …

Das Lastenheft des Softwareprodukts

6. Qualitätsanforderungen:

Produktqualität	sehr gut	gut	normal	irrelevant
Funktionalität		x		
Zuverlässigkeit	x			
Benutzbarkeit	x	x		
Effizienz			x	
Änderbarkeit			x	
Portierbarkeit				x

Die Benutzbarkeit der Funktionen /PF10/ und /PF20/ muss gut sein, da sie allein vom Besitzer der Firma verwendet werden. Die Benutzbarkeit aller übrigen Funktionen muss sehr gut sein, da sie von wechselnden Aushilfskräften bedient werden.

7. Ergänzungen (wie z. B. Abgrenzungskriterien):

Die Zuordnung verfügbarer Fahrzeuge zu Reservierungen erfolgt in der ersten Version manuell, Buchhaltungsfunktionen gehören nicht zum Leistungsumfang.

2.3.2 Pflichtenheft

- Nachdem in der Planungsphase das Lastenheft erstellt wurde, ist das Ziel der Definitionsphase, die genauen Anforderungen an ein System zu ermitteln und zu dokumentieren
- Ergebnis der Definitionsphase ist das **Pflichtenheft**
  - Es entsteht auf der Basis des Lastenhefts und muss genauer und detaillierter als dieses sein
  - Es darf keine Widersprüche enthalten
  - Es bestimmt den Lieferumfang im Sinn eines juristischen Vertrages und ist Grundlage für die Implementierung
- Auch für das Pflichtenheft gibt es keinen allgemeinverbindlichen Standard
  - Wir verwenden hier beispielhaft den Aufbau nach Balzert
  - Die Punkte aus dem Lastenheft finden sich detaillierter wieder

Pflichtenheft

VDI 2519: Beschreibung der Realisierung aller Anforderungen des Lastenheftes. Das Pflichtenheft enthält das Lastenheft. Im Pflichtenheft werden die Anwendervorgaben detailliert und die Realisierungsanforderungen beschrieben. Im Pflichtenheft wird definiert WIE und WOMIT die Anforderungen zu realisieren sind. [...] Das Pflichtenheft wird in der Regel nach Auftragserteilung vom Auftragnehmer erstellt, falls erforderlich unter Mitwirkung des Auftraggebers. Der Auftragnehmer prüft bei der Erstellung des Pflichtenheftes die Widerspruchsfreiheit und Realisierbarkeit der im Lastenheft genannten Anforderungen. Das Pflichtenheft bedarf der Genehmigung durch den Auftraggeber. Nach Genehmigung durch den Auftraggeber wird das Pflichtenheft die verbindliche Vereinbarung für die Realisierung und Abwicklung.

Pflichtenheft nach Balzert

1. **Zielbestimmung**
   - **Musskriterien**: Produktfunktionen, die für ein Funktionieren des Produktes unverzichtbar sind.
   - **Wunschkriterien**, sind zwar nicht unentbehrlich, werden aber ausdrücklich vom Auftraggeber gefordert. Ihre Umsetzung im Produkt ist also ebenfalls Pflicht, das Produkt würde aber auch ohne sie seinen Betrieb korrekt ausführen.
   - **Abgrenzungskriterien** definieren die Grenzen des Produktes (Punkte, die in der Analyse auftauchten, deren Umsetzung aber abgelehnt oder aufgeschoben wurde)
2. **Produkteinsatz**
   - **Anwendungsbereiche, Zielgruppen und Betriebsbedingungen** (physikalische Umgebung, tägliche Betriebszeit und ständige Beobachtung oder unbeaufsichtigter Betrieb). Bei den Zielgruppen sollte auch auf das Qualifikationsniveau der User eingegangen werden.

Pflichtenheft nach Balzert

3. Produktübersicht

- Übersicht über alle Geschäftsprozesse (Anwendungsfälle) in Form eines Anwendungsfall-Diagramms.

4. Produktfunktionen

- Die einzelnen Funktionen werden nach dem Schema *"/PFxx/"* durchnummeriert (in 10er-Schritten). Wunschkriterien werden mit *"/PFWxx/"* gekennzeichnet. Die einzelnen Funktionen werden detailliert beschrieben (Bedingungen, Akteure, Ablauf, Auswirkungen). Gliederung nach Geschäftsprozessen und weiteren Produktfunktionen.

5. Produktdaten

- Hier sind nur die langfristig (persistent) zu speichernden Daten gemeint. Die Punkte dieses Kapitels sind nach dem Schema *"/PDxx/"* durchzunummerieren.
  - Keine kryptischen Bezeichner
  - Art der Speicherung

Pflichtenheft nach Balzert

6. Produktleistungen

- Leistungsanforderungen an die Produktfunktionen (Zeit zur Ausführung, Genauigkeit der Berechnungen oder Datentransfervolumen und -dauer bei netzwerkfähigen Anwendungen). Nummerierung nach dem Schema: *"/PLxx/„*.

7. Qualitätsanforderungen

- Qualitätsmerkmale *Funktionalität*, *Zuverlässigkeit*, *Benutzbarkeit*, *Effizienz*, *Änderbarkeit* und *Übertragbarkeit* werden mit einer Qualitätsstufe aus *sehr gut, gut, normal* und *nicht relevant* bewertet.

8. Benutzungsoberfläche

- Beschreibung der grundlegenden Anforderungen (Art des Fensterlayouts, Dialogstruktur, Bedienung). Zusätzlich wichtig ist die Art der Zugriffsrechte der einzelnen Charaktere (Endbenutzer, Akteure).

Pflichtenheft nach Balzert

9. Nichtfunktionale Anforderungen

- einzuhaltende Gesetze und Normen
- Testat durch externe Prüfgesellschaft
- Revisionsfähigkeit
- Performance
- Sicherheitsanforderungen, wie Passwortschutz, Mitlaufen von Protokollen und sichere Übertragungen
- Plattformunabhängigkeiten

10. Technische Produktumgebung

- 10.1 **Software**: alle Software-Systeme (Betriebssystem, Datenbanken, diverse Fenstersysteme, ...), die auf dem Zielsystem der Anwendung installiert sein müssen, damit sie korrekt funktionieren kann.
- 10.2 **Hardware**: Hardware-Konfiguration, wenn das Produkt mit maximaler Last läuft.

Pflichtenheft nach Balzert

- 10.3 **Orgware**: alle organisatorischen Randbedingungen wie z.B. die Verfügbarkeit eines Netzwerkanschlusses. Soll das zu erstellende Produkt zur Laufzeit mit anderen Anwendungen in Verbindung treten, sollten diese und die genutzten Schnittstellen hier genannt werden.

11. **Spezielle Anforderungen an die Entwicklungsumgebung**
    - **11.1 Software**: die zur Entwicklung des Produktes nötigen Software-Werkzeuge
    - **11.2 Hardware** sowie
    - **11.3 Orgware** sind nur dann nötig, wenn an den Rechner für die Entwicklung des Produktes andere Anforderungen gestellt werden als an den für den Einsatz.
    - **11.4 Entwicklungs-Schnittstellen** beschreiben, über welche einzuhaltenden Hard- und Software-Schnittstellen Entwicklungs- und Zielmaschine gekoppelt sind.

Pflichtenheft nach Balzert

12. Gliederung in Teilprodukte

- Entwicklungen größeren Umfangs werden oft noch einmal in kleinere, überschaubarere Einheiten unterteilt.

13. Ergänzungen

- Anforderungen an das Produkt, die in den vergangenen Kapiteln keinen Platz gefunden haben, werden hier aufgeführt. Dazu zählen beispielsweise spezielle Installationsbedingungen, zu berücksichtigende Normen, Vorschriften, Patente und Lizenzen.

Lasten- und Pflichtenheft im Vergleich

Lastenheft	Pflichtenheft
1. Zielbestimmung (allgemein)	1. Zielbestimmung • Musskriterien • Wunschkriterien • Abgrenzungskriterien
2. Produkteinsatz (allgemein)	2. Produkteinsatz • Anwendungsbereich • Zielgruppe • Betriebsbedingungen
-	3. Produktübersicht
3. Produktfunktionen (funktionale Anforderungen)	4. Produktfunktionen (technische Umsetzung der funktionalen Anforderungen)
4. Produktdaten	5. Produktdaten
5. Produktleistungen	6. Produktleistungen
6. Qualitätsanforderungen	7. Qualitätsanforderungen (genauere Zielbestimmung)
-	8. Benutzeroberfläche
-	9. Nichtfunktionale Anforderungen
-	10. Technische Produktumgebung
-	11. Spezielle Anforderungen an die Entwicklungsumgebung
-	12. Gliederung in Teilprodukte
7. Ergänzungen	13. Ergänzungen

Hinweise zum Sprachstil

- Keine Schachtelsätze:
  Als Vorbedingung der Funktion gilt, dass alle übergebenen Argumente, wobei zu berücksichtigen ist, dass das letzte Argument, der Prüfmodus, optional ist mit einer Vorgabe von false, im angegebenen Wertebereich liegen müssen.

  wird ersetzt durch

  Als Vorbedingung müssen alle Argumente im vorgegebenen Wertebereich liegen. Der Prüfmodus ist das letzte Argument und kann entfallen. Dann gilt er als abgeschaltet.
- Überlegte Verwendung englischer Begriffe:

  Nach jeder cursor Bewegung wird das Access-File geupdatet.

  wird ersetzt durch

  Nach jeder Cursor-Bewegung wird die Zugriffsdatei aktualisiert.
- aber: lieber englische Begriffe, als falsche Übersetzungen:

  “Schnörkel rückplatz” löscht die letzte Rolle vor dem Schnörkel.

  hieß im Original vermutlich (echtes Fragment einer Gebrauchsanweisung!)

  “Cursor backspace” deletes the last character before the cursor

Hinweise zum Sprachstil

- Redundanzen und Füllwörter vermeiden:

  Die Klasse A ist hier eine Unterklasse von B, sie erbt deshalb alle Attribute von B.

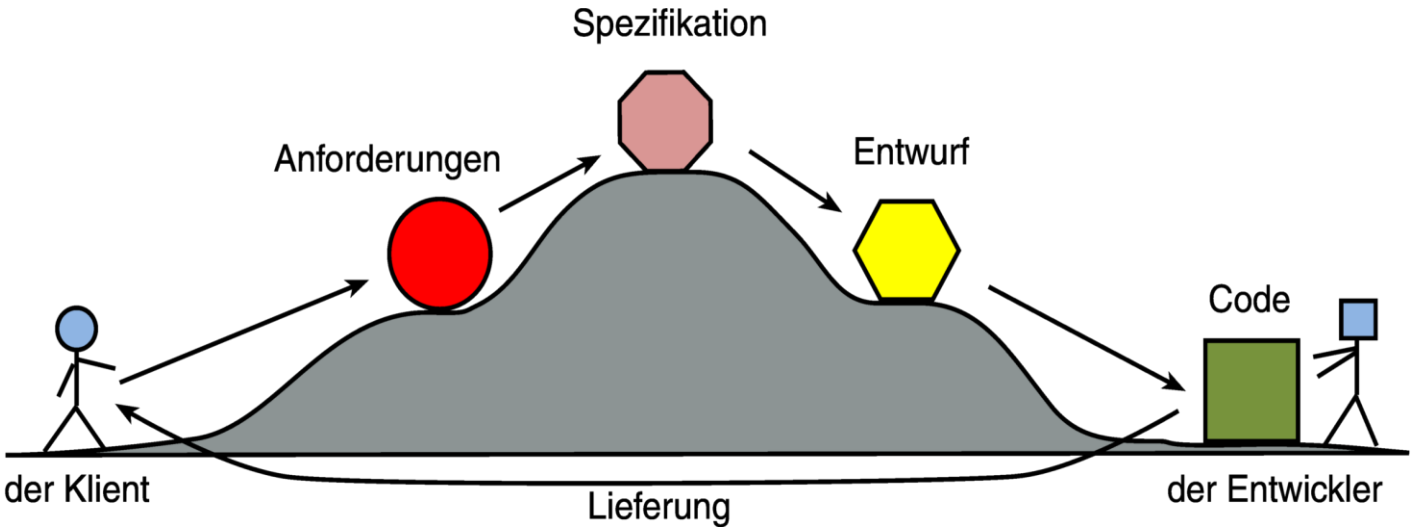
  wird ersetzt durch

  A ist eine Unterklasse von B.

- Keine Synonyme für Fachbegriffe verwenden. Dieselbe Sache muss stets mit demselben Ausdruck bezeichnet werden, auch wenn das zu Wiederholungen führt.
  Beispiel: Sind Typ und Kategorie bei Fahrzeugen der Autovermietung dasselbe oder nicht?

2.4 Fazit

- Mit der Fertigstellung einer sorgfältig geprüften, vollständigen Spezifikation ist in den meisten Projekten bereits die größte Hürde genommen!



Fazit

- Die Bedeutung der Analyse für die erfolgreiche Entwicklung von Softwaresystemen ist kaum zu überschätzen
  - Je später Fehler in den Anforderungen im Verlauf des Projekts behoben werden, umso höher sind die damit verbundenen Kosten
  - Viele Projekte scheitern aufgrund von falsch kommunizierten Anforderungen

➔Der Aufwand für die Analyse ist „gut angelegtes Geld“

Beispiel: Welche Beziehungen gelten hier?