

## Softwareengineering

- 1 Nutzwertanalyse
- 2 IT-Projekte
- 3 Vorgehensmodelle
- 4 Geschäftsprozessmanagement
- 5 UML Diagramme

## Problemstellung und Idee

Es müssen Entscheidungen zur Auswahl von verschiedenen Lösungen/Alternativen getroffen werden.

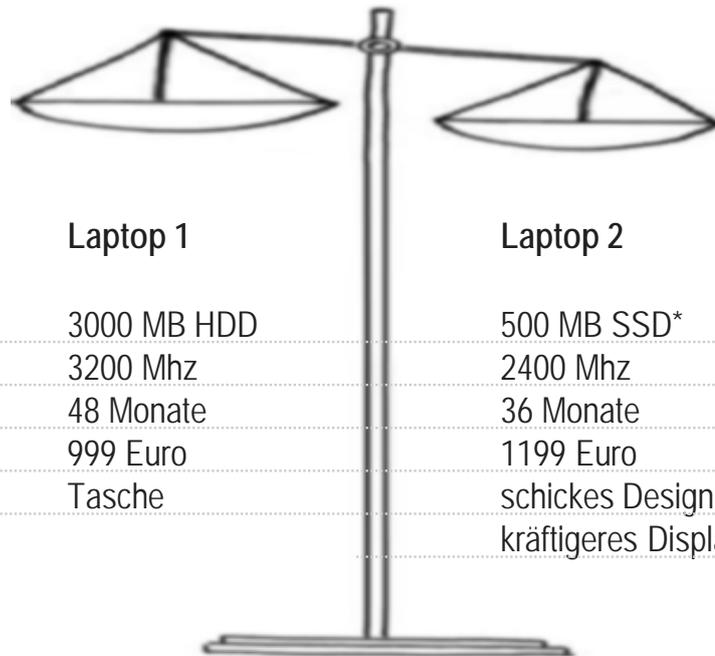
- Unbewusst / bewusst
- Privat / im Unternehmen

Lösungen können sein:

- Produkte (z.B. Auswahl einer Festplatte, einer Maschine, eines Kredites)
- Systeme (z.B. Auswahl eines ERP-Systems)
- Dienstleistungen (z.B. Beratung, Reinigungsfirma)

## Problemstellung und Idee

Die zentrale Frage lautet:  
Welche Lösung ist die „beste“?



HDD =Harddiscdrive: mechanische, aber meist sehr große Festplatte  
SSD =SolidStateDisc: bis zu 10mal schnellere Festplatte mit Chip-Technologie

Welche Probleme bestehen bei der Auswahl?

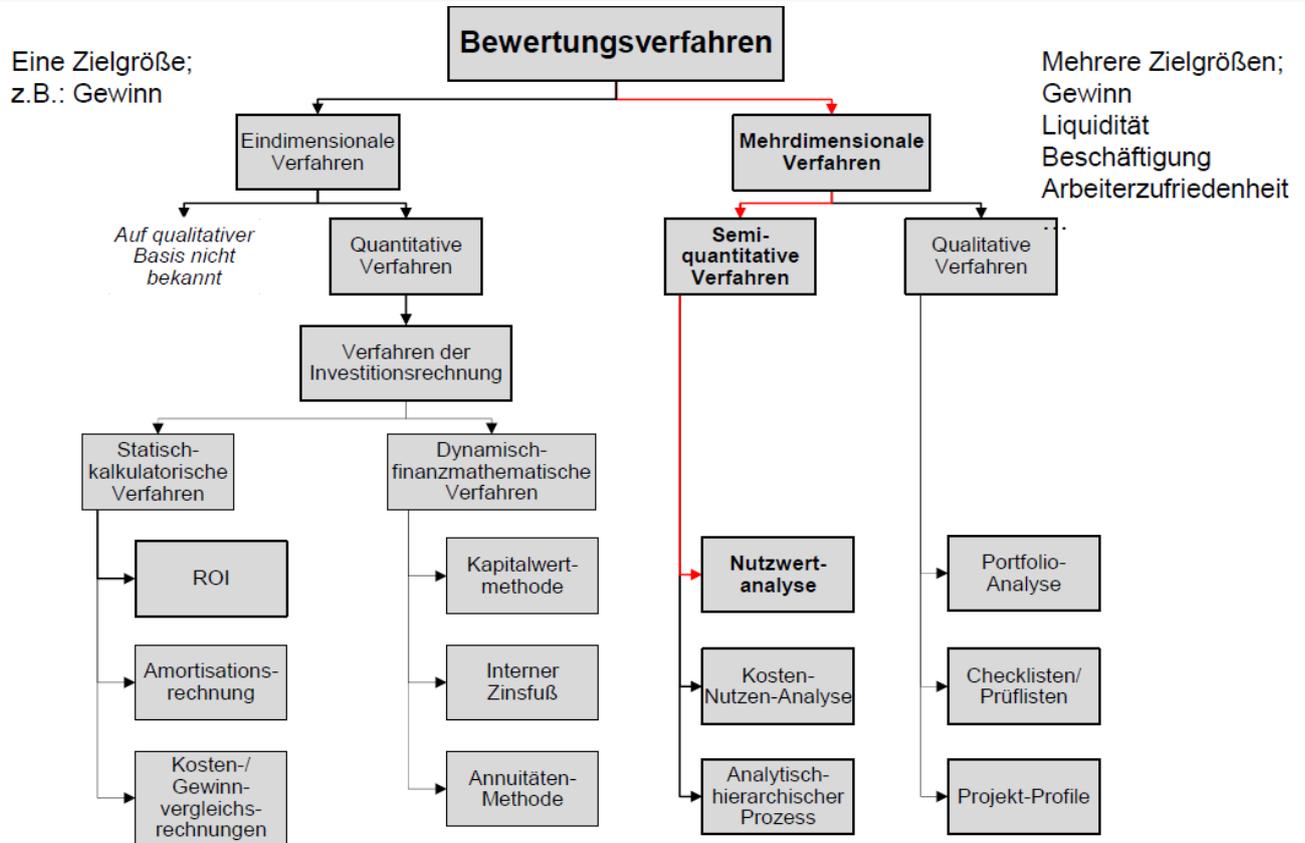
## Problemstellung und Idee

### Lösung: Einsatz eines Verfahrens

- Die **Nutzwertanalyse** (NWA, auch Scoring- Modell, Punktwertverfahren, Multifaktoren-Methode, Utility Analysis) ist ein Verfahren zur Bewertung von Handlungsalternativen
- Sie ermöglicht die Berücksichtigung mehrerer, beliebiger, komplexerer Ziele.
- Einzelnen Kriterien wird ein Nutzen zugewiesen; die Alternative mit dem höchsten Gesamtnutzen wird gewählt

*Definition:* „Analyse einer Menge komplexer Handlungsalternativen mit dem Zweck, die Elemente dieser Menge entsprechend den Präferenzen des Entscheidungsträgers bezüglich eines multidimensionalen Zielsystems zu ordnen“ (C. Zangenmeister, 1976)

# Problemstellung und Idee



Quelle: Endrik Lengwenat, TU München

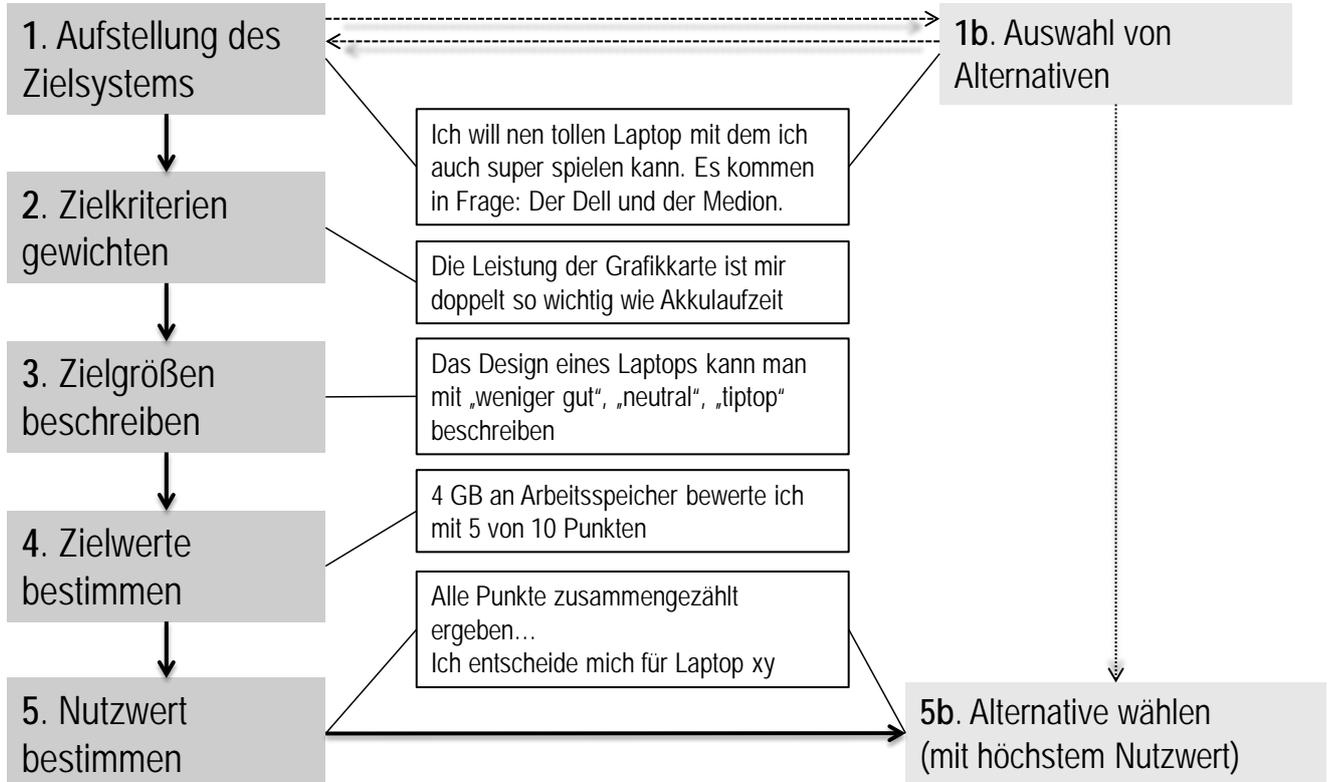
## Problemstellung und Idee

- Sie wurde als Erweiterung der Kosten-Nutzen-Analyse entwickelt, da sie nicht nur monetäre Bewertungen sondern auch die persönlichen Präferenzen des Entscheiders berücksichtigt.
- Die Nutzwertanalyse erlaubt bei einer großen Anzahl von Alternativen mit vergleichsweise geringem Aufwand eine erste, grobe Beurteilung von Investitionen oder Projekte.
- Diese Methode kann für eine Vielzahl von Anwendungsgebieten eingesetzt werden, bei denen quantitative Daten fehlen, wie z. B. die Arbeitsplatzbewertung, die Standortwahl, die Erklärung von Käuferverhalten, die Vorauswahl von Investitionsobjekten etc.

## Vorgehen (Übersicht)

bewusst

unbewusst



Natürlich kann, bei der bewussten Entscheidung durch eine NWA, die Auswahl von Alternativen an beliebiger Stelle vor Schritt 4 erfolgen. In der Regel handelt es sich um wenige Alternativen. Falls nicht, sollte die Auswahl „mit gesundem Menschenverstand“, z.B. anhand von bekannten KO-Kriterien, reduziert werden.

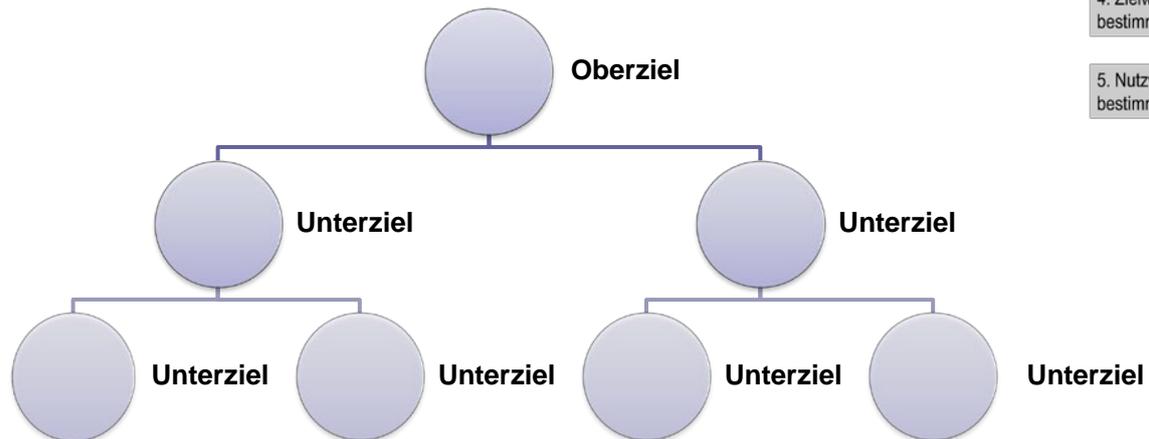
Frage:

Spaghetti Bolognese passt immer, aber es muss schnell gehen.

Handelt es sich bei der Wahl um das Tüten-Fix von entweder Knorr oder Maggie um eine bewusste oder unbewusste Entscheidung? Welche Faktoren spielen eine Rolle?

## Vorgehen: 1. Zielsystem

Ein Zielsystem kann mit Hilfe eines Zielbaums visualisiert werden.



## Vorgehen: 1. Zielsystem

Anforderungen:

**Unabhängigkeit:** Die Kriterien dürfen nicht in einem Zusammenhang stehen. Beispiel für fehlende Unabhängigkeit: Fahrtzeit und verkehrsgünstige Lage.

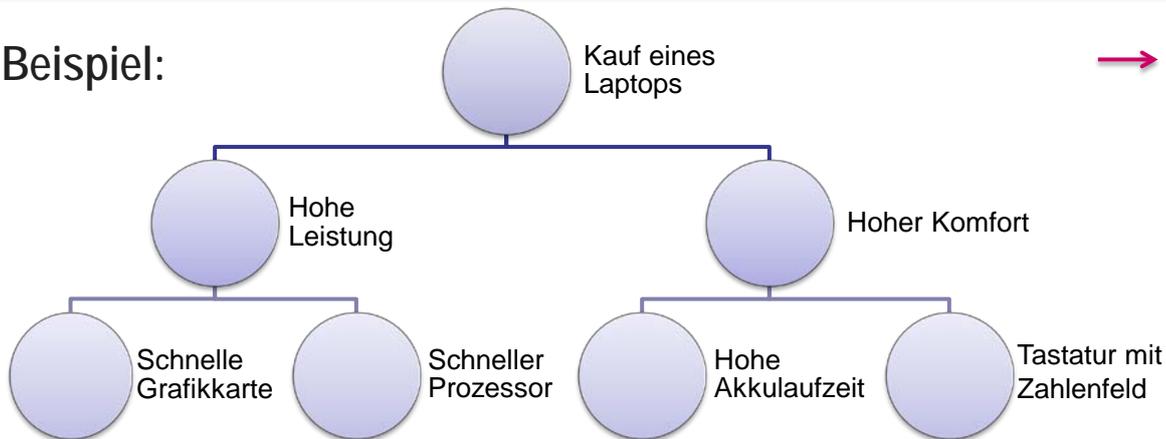
**Genauere Formulierung:** Beispiel: das Kriterium „Erfolg“ ist mehrdeutig und muss entsprechend genauer beschrieben werden. Dazu eignet sich eine Unterkategorie.

**Vollständigkeit:** Die Bewertungskriterien sollen unterschiedliche Objekteigenschaften abfragen, um Einseitigkeit in der Beurteilung zu vermeiden.



## Vorgehen: 1. Zielsystem

Beispiel:

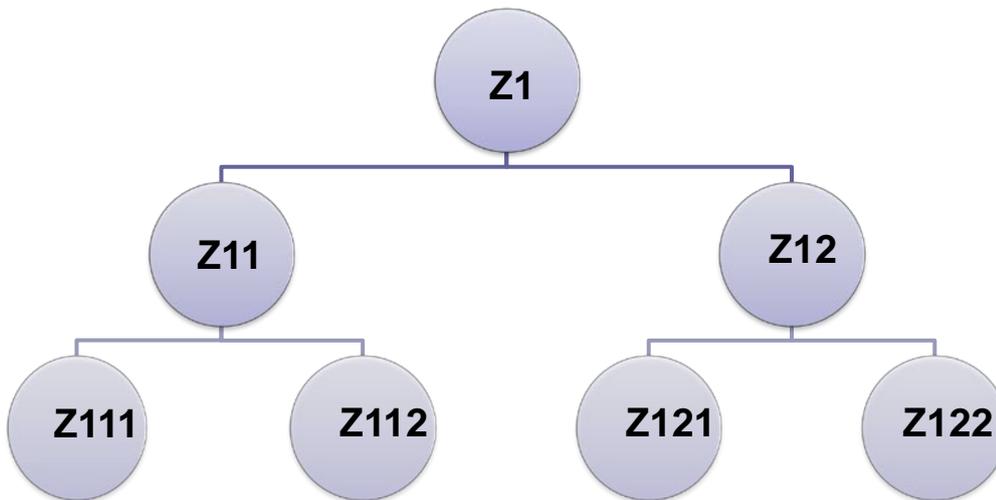
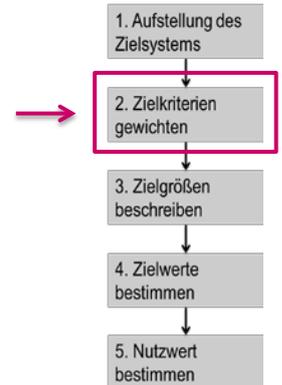


- Die Zielkriterien werden i.d.R. durch mehrere Personen aus unterschiedlichen Unternehmensbereichen festgelegt.
- Bei der Auswahl eines komplexen Systems empfiehlt sich die Erstellung eines Lastenheftes

(Das Lastenheft wird vom Auftraggeber oder in dessen Auftrag erstellt. Hier werden alle Anforderungen, Erwartungen und Wünsche an ein geplantes Produkt (Objekt) in natürlicher Sprache erfasst.)

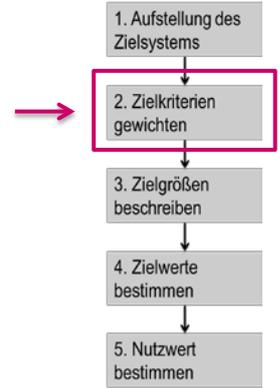
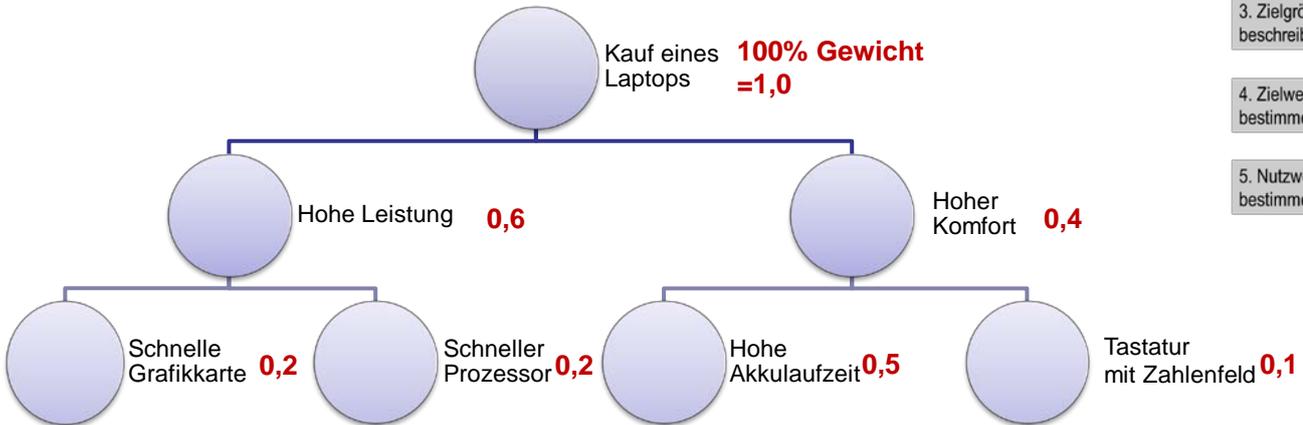
## Vorgehen: 2. Zielkriterien gewichten

die Zielkriterien werden entsprechend ihrem Einfluss auf den Gesamtnutzen gewichtet. In einer Zielstufe ist die Summe der absoluten Kriteriengewichte = 1.



# Vorgehen: 2. Zielkriterien gewichten

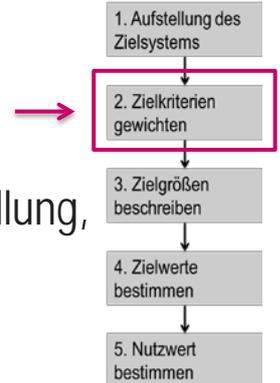
Beispiel:



Problem: wie kann die Rangfolge ermittelt werden?

## Vorgehen: 2. Zielkriterien gewichten

### Lösung: Paarweiser Vergleich



- Vergleich jedes Einzelkriteriums mit jedem anderen und Festlegung einer Wichtigkeit in der direkten Gegenüberstellung, wobei sich die Entscheidungsmenge auf (A wichtiger B), (A gleichwertig B), und (A unwichtiger B) reduziert.
- Soll die Entscheidung einen Zwang zur Reihenfolgenbildung enthalten, kann die Gleichwertigkeit entfallen.
- Die Summe dieser Einzelbewertungen führt auf eine Aussage zur Gesamtwichtigkeit jedes Einzelkriteriums in Form einer Rangziffer.

als wichtiger	Kriterium 1	Kriterium 2	Kriterium 3	Kriterium 4	Kriterium 5	Kriterium 6	Kriterium 7	Kriterium 8	Kriterium 9	Kriterium 10	Summe	%
Kriterium 1	1	1	1	1	1	1	1	1	1	1	9	19,87%
Kriterium 2	0	1	1	1	1	0	0	1	1	1	6	12,84%
Kriterium 3	0	0	1	1	0	0	1	1	0	1	4	8,76%
Kriterium 4	0	0	0	1	1	0	1	0	0	1	3	6,62%
Kriterium 5	0	0	0	0	1	0	1	1	1	0	3	6,62%
Kriterium 6	0	0	1	0	0	1	1	1	1	1	6	10,87%
Kriterium 7	0	1	1	1	1	0	1	0	0	1	6	10,87%
Kriterium 8	0	1	0	0	0	0	1	1	0	1	3	4,34%
Kriterium 9	0	0	0	1	0	0	1	0	1	1	3	6,62%
Kriterium 10	0	0	1	1	1	0	1	1	1	1	6	12,84%
											<b>Prüfsumme</b>	<b>100,00%</b>

Bewertung 1 bedeutet in Beispiel Kriterium 1 ist wichtiger als Kriterium 2

## Vorgehen: 2. Zielkriterien gewichten/Pairweiser Vergleich

als wichtiger	Kriterium 1	Kriterium 2	Kriterium 3	Kriterium 4	Kriterium 5	Kriterium 6	Kriterium 7	Kriterium 8	Kriterium 9	Kriterium 10	Summe	%
Kriterium 1		1	1	1	1	1	1	1	1	1	9	19,57%
Kriterium 2			1	1	1	1	0	0	1	1	6	13,04%
Kriterium 3				1	1	0	0	1	1	0	4	8,70%
Kriterium 4					1	1	0	1	0	0	3	6,52%
Kriterium 5						1	0	1	1	0	3	6,52%
Kriterium 6							1	1	1	1	5	10,87%
Kriterium 7								1	0	0	5	10,87%
Kriterium 8									1	0	2	4,35%
Kriterium 9										1	3	6,52%
Kriterium 10											6	13,04%
											<b>Prüfsumme</b>	<b>100,00%</b>

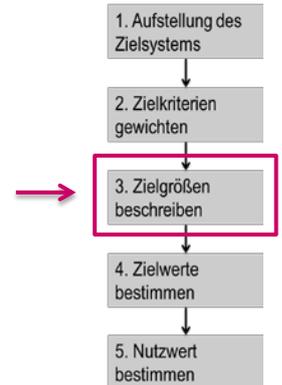
Bewertung 1 bedeutet im Beispiel Kriterium 1 ist wichtiger als Kriterium 2

Quelle: korrigiert von: <http://www.sixsigmalackbelt.de/paarweiser-vergleich/>

☺ Aufgabe:  
Vervollständigen Sie die Tabelle!

## Vorgehen: 3. Beschreibung der Zielgrößen

- Anhand der Zielkriterien getätigte Beschreibung der bekannten qualitativen und quantitativen Eigenschaften (sog. Zielgrößen) der Varianten.
- Punktevergabe (1-10) durch Experte(n)
- Bsp.: Design kann als „schlecht (0 Punkte)“, „mittel“ (5 Punkte) und „super“ (10 Punkte) beschrieben werden
- Messung Anhand von Skalierungen
  - Nominale Skalierungen
  - Ordinale Skalierungen
  - Intervallskalen
  - Verhältnisskalen



## Vorgehen: 3. Beschreibung der Zielgrößen

### Nominale Skalierung

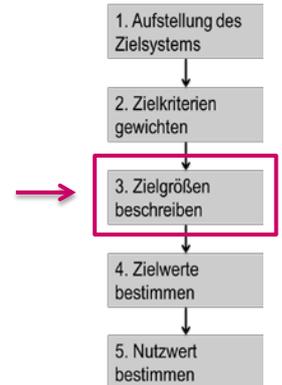
Einfachste Form der Skalierung

- Die Klassifizierung eines Objektes erfolgt darüber, ob einem Bewertungskriterium entsprochen wird oder nicht.
- Beispiel: „ja/nein“, „gut/schlecht“, „+/-“.

### Ordinale Skalierung

differenzieren stärker die Richtung der Nutzenunterschiede.

- Die Rangordnung wird garantiert.
- Beispiel: „größer“, „kleiner“, „gleich“ etc.

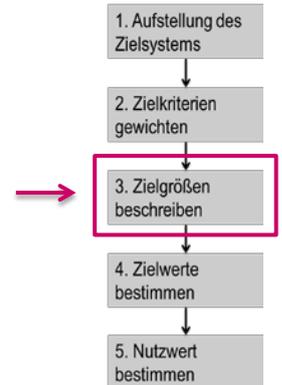


## Vorgehen: 3. Beschreibung der Zielgrößen

### Intervallskala

weisen den Abständen der Rangordnung feste Intervalle zu.

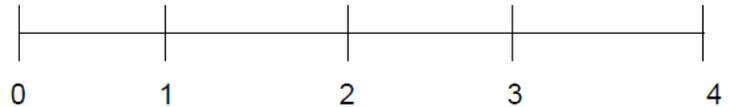
- Die Stufen der Skalen sind immer gleich groß
- Relativer Nullpunkt
- Beispiele: „sehr starke Leistung= 5 Punkte“, „sehr schwache Leistung= 0 Punkte“.  
Temperaturmessung in °C



### Verhältnisskala

fordern neben den festen Intervallen auch die exakte Bestimmung eines absoluten Nullpunktes.

- Beispiel: Gewichtsskala.



## Vorgehen: 4.+5. Ziel- und Nutzwerte bestimmen

### Auch: Wertsynthese

- In einer Zielwertmatrix werden die Alternativen nebeneinander aufgestellt
- Die jeweiligen Kriterien werden untereinander gestellt
- Berechnung



$$N_j = \sum w_k * P_{jk}$$

- $N_j$ : der Nutzwert der Alternative  $j$
- $w_k$ : Gewichtung, die dem Kriterium  $k$  zugewiesen wird
- $P_{jk}$ : Punktwert, den eine Alternative  $j$  beim Beurteilungskriterium  $k$  aufgrund eines Expertenurteils erreicht
- $k$ : das Beurteilungskriterium

Vorgehen: 4.+5. Ziel- und Nutzwerte bestimmen

	Gewichtung	Alternative 1		Alternative 2		Alternative 3		Alternative 4	
		Bewertung	Wert	Bewertung	Wert	Bewertung	Wert	Bewertung	Wert
Kriterium 1	19,57%	10	1,96	10	1,96	7	1,37	3	0,59
Kriterium 2	13,04%	4	0,52	10	1,30	8	1,04	5	0,65
Kriterium 3	8,70%	3	0,26	3	0,26	3	0,26	3	0,26
Kriterium 4	6,52%	7	0,46	4	0,26	3	0,20	4	0,26
Kriterium 5	6,52%	8	0,52	8	0,52	7	0,46	8	0,52
Kriterium 6	10,87%	4	0,43	9	0,98	9	0,98	4	0,43
Kriterium 7	10,87%	7	0,76	2	0,22	1	0,11	1	0,11
Kriterium 8	4,35%	9	0,39	3	0,13	3	0,13	3	0,13
Kriterium 9	6,52%	8	0,52	1	0,07	0	-	1	0,07
Kriterium 10	13,04%	8	1,04	1	0,13	3	0,39	1	0,13
<b>Summe</b>		<b>6,87</b>		<b>5,83</b>		<b>4,93</b>		<b>3,15</b>	

Zielwert für Kriterium 1 der Alternative 4

Nutzwert der Alternative 4

Bewertungszahl von 0 - 10  
 Bewertungszahl 0 entspricht Alternative erfüllt das Kriterium nicht  
 Bewertungszahl 10 entspricht Alternative erfüllt das Kriterium vollständig

Was wäre der maximal zu erreichende Wert gewesen?

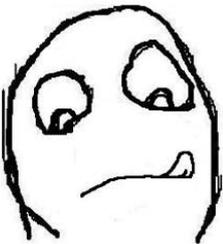
Wieviel Prozent davon erreicht bspw. Alternative 1?

## Zusammenfassung und Kritik

- Häufig eingesetztes Instrument als Entscheidungsgrundlage.
  - Durch die Gliederung in Teilaspekte gute Übersicht über das zu bewertende Problem.
  - Alle Schritte des Verfahrens sind formal transparent
  - Berücksichtigung von mehreren Zielen, die auch nicht monetären Inhalt haben können.
- 
- 
- Vorauswahl der Alternativen sind nicht formalisiert
  - Die Auswahl der Kriterien ist subjektiv und die Bewertung erfolgt anhand grob geschätzter Werte.  
→ Die NWA kann als Täuschungsinstrument dienen.
  - Erheblicher Zeitaufwand, wenn mehrere Personen an der Bewertung beteiligt sind.
- 

## Zusammenfassende Fragen: Abschnitt 2.1

- Wozu dient das Verfahren der Nutzwertanalyse?
- Wie lässt sich die NWA einordnen?
- In welchen Schritten erfolgt eine NWA?
- Wie funktioniert ein paarweiser Vergleich?
- Wie wird der Gesamtnutzen ermittelt?
- Welche Kritik gibt es an der Methode? An welchen Stellen?



## Aufgaben

Kriterium	Gewichtung (Punkte)	Alternative A	Nutzwert A	Alternative B	Nutzwert B	Alternative C	Nutzwert C
<b><u>Standort:</u></b>							
Hauptstraße	7	5		7		3	
Nähe an der Geschäften	9	5		3		10	
Lagernähe	8	10		8		6	
<b><u>Kosten:</u></b>							
Steuerliche Belastung	6	4		5		5	
Bau-/Mietkosten	15	6		7		4	
Werbung	10	7		4		8	
<b><u>Gewinnpotenziale:</u></b>							
Wenig Konkurrenz	15	3		4		3	
Kunden sind offen für Neues	10	5		5		6	
Arbeitslosenquote gering	20	7		5		6	
<b>Summe:</b>	<b>100</b>						

Eine Firma möchte eine weitere Filiale in einem anderen Bundesland aufmachen. Ihr wurden drei Angebote aus verschiedenen Bundesländern gemacht und sie muss sich nun für einen Standort entscheiden. Führen Sie eine Nutzwertanalyse anhand obiger Tabelle durch.

## Softwareengineering

- 1 Nutzwertanalyse 
- 2 IT-Projekte
- 3 Vorgehensmodelle
- 4 Geschäftsprozessmanagement
- 5 UML Diagramme

## Lesen..Nachschlagen..Lernen..

### Literatur zu Softwareengineering (IT-Projekte, Vorgehensmodelle, UML)

- Brandt-Pook, H.; Kollmeier, R.:  
Softwareentwicklung kompakt und verständlich;  
Wiesbaden 2008  
**online via [Springerlink](#)**



## Definition eines Projektes

Definition DIN: „Ein Projekt ist ein zeitlich begrenztes Vorhaben zur Schaffung eines einmaligen Produktes, einer Dienstleistung oder eines Ergebnisses.“



Aus der Definition wird bereits deutlich, dass ein Projekt mit dem Erledigen einer nicht ganz einfachen Aufgabe zu tun hat.

# Kriterien zur Definition eines Projektes



## Kriterium: Ziele

### Ein Projekt hat Ziele.

Anhand der Ziele richten sich Projekte aus.

Bei der Zielfestlegung ist einiges zu beachten:

- Ziele sollten präzise und verständlich formuliert sein.
- Die Ziele sollten vollständig und widerspruchsfrei formuliert sein.
- Ob ein Ziel erreicht wurde, sollte entscheidbar oder messbar sein.
- Sie sollten erreichbar sein.
- Sie sollten ein definiertes Ende haben.



→ SMART

## Kriterium: Anfang und Ende / Einmaligkeit

### Ein Projekt hat einen Anfang und ein Ende.

- Beginn ist ein sog. Kick-Off-Workshop
- Projekte sollten nicht unendlich lange dauern
- Das Projektende ist erreicht, wenn alle Projektziele erreicht wurden.



### Ein Projekt ist einmalig.

- Einmaligkeit ist eine wichtige Abgrenzung zum normalen Arbeitsprozess
- An individuellen Kundenwünsche orientiert
- Keine Routinearbeiten

## Kriterium: Komplexität

### Ein Projekt ist komplex.

- Ist gegeben sobald verschiedene Fachexperten (mit unterschiedlichen Kompetenzen) zur Erreichung eines Ziels benötigt werden.
- kann auch bedeuten, dass eine Fachkraft an einem besonders umfangreichen Projekt arbeitet.
- Anspruchsvolle Projektziele
- Oft hohes Budget
- Lange Projektlaufzeit



## Kriterium: Budget / AuftraggeberIn

Ein Budget ist ein Geldbetrag, der dem Projekt zur Verfügung steht.

- Ist ein festgelegter Geldbetrag für ein Projekt,
- kann aber auch als Arbeits- und Personentage festgelegt werden.



Jedes Projekt hat einen Auftraggeber oder Auftraggeberin

- Ist der wichtigste (mächtigste) Bestandteil eines Projektes.
- gibt genaue Angaben (Aufschluss) über die Projektziele.
- Aushandeln von Verträgen ist ihre Aufgabe
- kann auch eine juristische Person sein

## Kriterium: Projektorganisation

### Jedes Projekt hat eine eigene Projektorganisation

- Einteilung von Rollen
- Zuständigkeiten
- Verantwortlichkeiten



## Kriterium: Risiko

### Jedes Projekt beinhaltet das Risiko des Scheiterns

Bedeutet nicht, dass kein Projektziel erreicht wurde und nicht das „Aus“ des Projektes, sondern lediglich das „Nicht-Erreichen“ aller Projektziele



### Beispiele/Gründe

- Budgetüberschreitung
- Zeitüberschreitung
- Unerfahrene Projektleiter
- Projektziele zu anspruchsvoll
- Mangel an Teamgeist / geringe Identifikation mit den Projektzielen
- Unzureichende Berichtserstattung
- Falsche Projektplanung (Projektteam wurde nicht beteiligt)

## Warum machen wir Projekte?

Projekte sind heutzutage in den meisten Unternehmen eine „klassische“ Organisationsform

- Mittel zur Bewältigung des ständigen Wettbewerbs- und Veränderungsdruckes in der Wirtschaft
- Wettbewerbs- und Veränderungsdruckes besonders in der IT außerordentlich hoch



### Beispiel:

Moore's Law in der IT = 18 Monate vs.

Entwicklungszyklus bei VW Golf V = 5 Jahre

## Arten von IT-Projekten

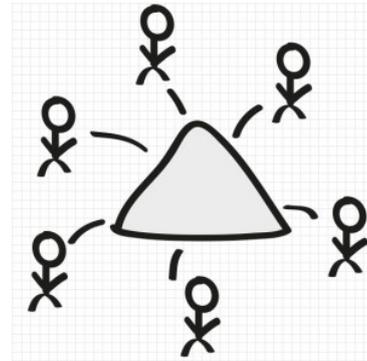
### Projektdimensionen im Überblick:

- Projektaufwand – Kleine oder große Projekte
- Projektstandorte – Zentrales oder verteiltes Projektgeschehen
- Technologie – SAP oder Linuxprojekt
- Charakter der Projektziele – Erforschungsprojekt
- Phasenschwerpunkte – Konzeptionsprojekt, Umsetzungsprojekt, Einführungsprojekt, Wartungsprojekt

## Rollen in einem Projekt

Wichtige Rollen im Projektgeschehen sind:

- Der Kunde
- Der Projektleiter
- Das Teammitglied
- Der Qualitätsmanager
- Gremium Lenkungsausschuss



## Rollen in einem Projekt: Kunde

Die wichtigste Rolle im Projekt, da der Kunde das Projekt in Auftrag gibt:



- Was sind meine Ziele? Was will ich erreichen?
- Wie finde ich einen geeigneten Dienstleister?
- Welchen Beitrag kann mein Unternehmen zum Projekt beitragen?
- prüft und priorisiert (Kosten- Nutzenanalyse).
- Welche Projektaktivitäten sind bereits abgeschlossen?
- Welche Projektaktivitäten sind in Bearbeitung?
- Welche Probleme gibt es derzeit?
- Ist der geplante Endtermin zu halten?

## Rollen in einem Projekt: ProjektleiterIn

Die Projektleiterin bzw. der Projektleiter leitet das Projekt.  
Er/Sie ist verantwortlich für:

- Die Planung eines Projektes
- Die Führung und Steuerung des Projektteams
- Die Kontrolle der erarbeiteten Leistung
- Die Kommunikation mit dem Auftraggeber, dem Lenkungsausschuss und anderen Personen die ein Interesse am Projekt haben → Stakeholder



## Rollen in einem Projekt: Teammitglied

Die Rolle Teammitglied ist ein Oberbegriff für alle fachlichen Rollen im Projekt und beinhaltet folgende Aufgaben:

- sollen die ihnen übertragenen Aufgaben erfüllen (Arbeitspakete abarbeiten).
- sollen ihre Kenntnisse auf dem neusten Stand halten.
- beteiligen sich an der Planung und nutzen ihre Fachkenntnisse.
- ergreifen Eigeninitiativen und sollen dennoch in der Gruppe funktionieren.



## Exkurs: Teambildungsphase

**Forming**

- **Orientierungsphase**  
„Wo ist mein Platz in der Gruppe?“).

**Storming**

- **Konfliktphase**  
Unzufriedenheit, weil die eigene Rolle noch nicht gefunden wurde

**Norming**

- **Teambildungsphase**  
Organisation: Die eigene Rolle im Team wird klar

**Performing**

- **Performancephase**  
Leistung: Alle Beteiligten funktionieren im Team und leisten ihren Beitrag zur Zielerreichung

## 5-Phasen Modell nach Tuckman (1965)

- Gruppen entwickeln eine Dynamik und durchlaufen Phasen, in denen jeweils andere Themen und Bedürfnisse der Gruppenmitglieder aktuell sind
- Stadien lassen sich nicht immer exakt trennen
- Gruppen können in frühere Phasen zurückfallen
- 5. Phase (Adjourning) bezeichnet das Auseinandergehen einer Gruppe

## Rollen in einem Projekt: QualitätsmanagerIn

Die Aufgabe einer QualitätsmanagerIn ist es die Qualität von Prozessen und Produkten sicherzustellen und zu verbessern.

- stellt die Qualität von Prozessen und Produkten sicher und verbessert diese gegebenenfalls.
- gibt Verbesserungsvorschläge zu Vorgehensmodellen und Tests.
- Der Qualitätsmanager ist unabhängig vom Projektleiter
- kann als Rolle auch vom Projektleiter übernommen werden (jeweils abhängig von der Projektgröße).



## Rollen in einem Projekt: Gremium Lenkungsausschuss

Den Lenkungsausschuss kann man als oberste Instanz im Projektgeschehen ansehen.

- muss bei größeren Projekten vorhanden sein.
- besteht aus Abteilungsleitern und/oder anderen Entscheidern (Schlüsselpersonen die Interesse am Erfolg des Projektes haben).
- trifft wichtige strategische Entscheidungen.
- hilft in kritischen Projektsituationen.
- wacht über den Projektfortschritt.



## Projektdokumente

Dokumentieren hilft Informationen nachzuvollziehen und Vorgehensschritte zurückzuverfolgen.

Unterscheidung der Dokumente:



**vertragliche:** Angebot, Dienstleistungsvertrag, Rechnung, Abnahmeprotokolle.

**fachliche:** Testplan, Lastenheft, Pflichtenheft, projektinterne Richtlinien, Benutzerhandbuch, technische Systemdokumentation.

**PM-Dokumente:** Projektauftrag, Projektplan, Projektübersicht, Änderungsprotokoll, Projektstatusbericht.

### Erinnerung

#### Lastenheft:

- wird vom Kunden erstellt.
- beinhaltet die Vorstellung über die Lösung aus der Kundensicht.
- je detaillierter der Einbezug von Projektbestandteilen desto geringer sind die Risiken.
- wird verwendet zum Abarbeiten der Anforderungspunkte (Checkliste).

#### Pflichtenheft:

- wird vom Dienstleister erstellt.
- wird an die Anforderungen des Lastenheftes angepasst.
- beinhaltet alle Anforderungen an die geplante Lösung. (Vollständigkeit)
- hat wohlüberlegte Inhalte (Widerspruchsfreiheit).
- muss im Bezug auf die Anforderungen verständlich für Kunden und Dienstleister sein (Allgemeinverständlichkeit).
- Fachbegriffe werden vermieden oder verdeutlicht (Präzision).
- ist eine Checkliste zur Abnahme des Kunden.

## PM-Dokumente: Projektauftrag

Der Projektauftrag beschreibt, um was es bei dem Projekt im Kern geht:



- Projektname, **Ziele**, Rahmenbedingungen, Risikofaktoren, Meilensteine, Aufwand und Zeitplan, Abnahmekriterien, rechtliche Aussagen, Organisation, sämtliche Kosten.
- Abgleich mit der SMART-Regel

# PM-Dokumente: Projektübersicht

stellt kurz und knapp wesentliche Inhalte aus dem Projektauftrag und dem Projektplan dar.

<b>Projektübersicht</b>		<b>Projekt: Webshop XXL</b>			
<b>Auftraggeber</b>		<b>Auftragnehmer</b>			
MBZL GmbH		SWHL GmbH			
<b>Projektleiter Auftraggeber</b>		<b>Projektleiterin Auftragnehmer</b>			
Herr Mommert		Frau Beitinger			
<b>Projektziele</b>	- Aufbau Internetpräsenz mit Bestellwesen - Anbindung an bestehendes Kundenverwaltungssystem				
<b>Projektaufwand</b>	Ca. 50 Personentage (PT)	<b>Abrechnungsform</b>	Nach Aufwand		
<b>Phase</b>	<b>Ergebnis</b>	<b>Aufwand</b>	<b>von</b>	<b>bis</b>	<b>wer</b>
Konzeption	Pflichtenheft	8 PT	02.02.	06.02.	Team 1
Design	Systementwurf	7 PT	08.02.	13.02.	Team 2
Realisierung Internetpräsenz	DB läuft	10 PT	16.02.	06.03.	Team 2
Realisierung Schnittstelle	Schnittstelle erprobt	12 PT	08.02.	06.03.	Team 1
Testen	Testplan erledigt; Abnahme	6 PT	09.03.	20.03.	Team 1 + 2

Quelle: Brandt-Pook; Softwareentwicklung kompakt und verständlich

## PM-Dokumente: Projektverzeichnis

- [-] Projektverzeichnis\_WeserTourist
  - [-] Projektergebnisse
    - [-] Phase1\_Konzeption
    - [-] Phase2\_Design
  - [-] Phase3\_Realisierung
    - [-] Software\_Version\_7
    - [-] Software\_Version\_8
  - [-] Phase4\_Testen
    - [-] Abnahmetest
    - [-] InternerTestplan
  - [-] Phase5\_Einführung
  - [-] Phase6\_Dokumentation
    - [-] Benutzerhandbuch
    - [-] technische Systemdokumentation
  - [-] Projektlogbuch
  - [-] Projektmanagement
    - [-] Projektauftrag
  - [-] Projektplanung
    - [-] Projektpläne
  - [-] Protokolle
    - [-] Protokolle\_intern
    - [-] Protokolle\_mitAuftraggeber



senkt den Verwaltungsaufwand

Bildung von Hauptkategorien  
sinnvoll.

## Projektmanagement

Definition: „Projektmanagement bezeichnet die Gesamtheit von Führungsaufgaben, Führungsorganisation, Führungstechniken und Führungsmitteln für die Abwicklung eines Projektes.“

Inhalt:

- planen,
- organisieren und
- kontrollieren von Projekten.



## Phasen im Projektmanagement



In jeder Phase werden bestimmte Prozesse und Aktivitäten durchlaufen.

### Initialisierung:

- Geburtsstunde des Projekts
- Vorstellung beim Kunden und beim Auftraggeber
- Ziel:            Gemeinsames Verständnis von Auftraggeber und Auftragnehmer über die Eckpunkte und wichtigen Rahmenbedingungen des Projektes
- Ergebnis:       Das zentrale Ergebnis der Projektinitialisierung ist der Projektauftrag

## Phasen im Projektmanagement (2)



### Planung:

beinhaltet:

- Die Projektziele
- Angaben zu den wichtigsten Meilensteinen
- Der geplante Aufwand
- Verantwortlichkeiten
- Angaben zum geplanten Vorgehen im Projekt

## Phasen im Projektmanagement (3)



### Steuerung:

#### Voraussetzungen:

- Der Projektauftrag und der Projektplan liegen vor.
- Das Projektteam ist zusammengestellt und über das Projekt informiert.
- Die Infrastruktur für das Projektteam ist gegeben

#### Ziele:

- Geeignete Vorgaben an den Kernbereich des Projekts zu machen
- Messwerte aus dem Kernbereich des Projektes aufzunehmen
- Bedarfsgerecht die Planung anzupassen
- Überarbeitete Vorgaben für den Kernbereich des Projektes zu liefern.

## Phasen im Projektmanagement (4)



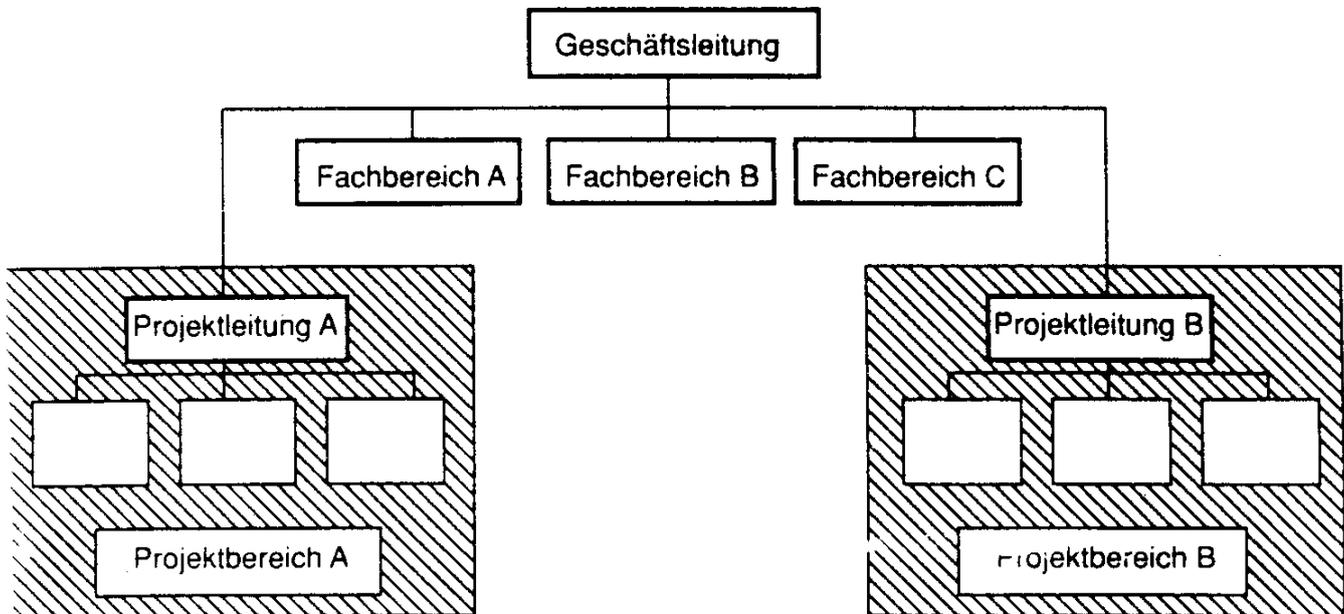
### Abschluss:

Die Voraussetzungen für den Projektabschluss sind, dass die Projektziele erreicht wurden und der Kunde die IT-Lösung abgenommen hat.

- Abschlusssitzung
- Archivierung
- Weitere Zusammenarbeit
- Ergebnisse übertragen
- Team auflösen

# Projektorganisation

## Reine Projektorganisation



### Vorteile:

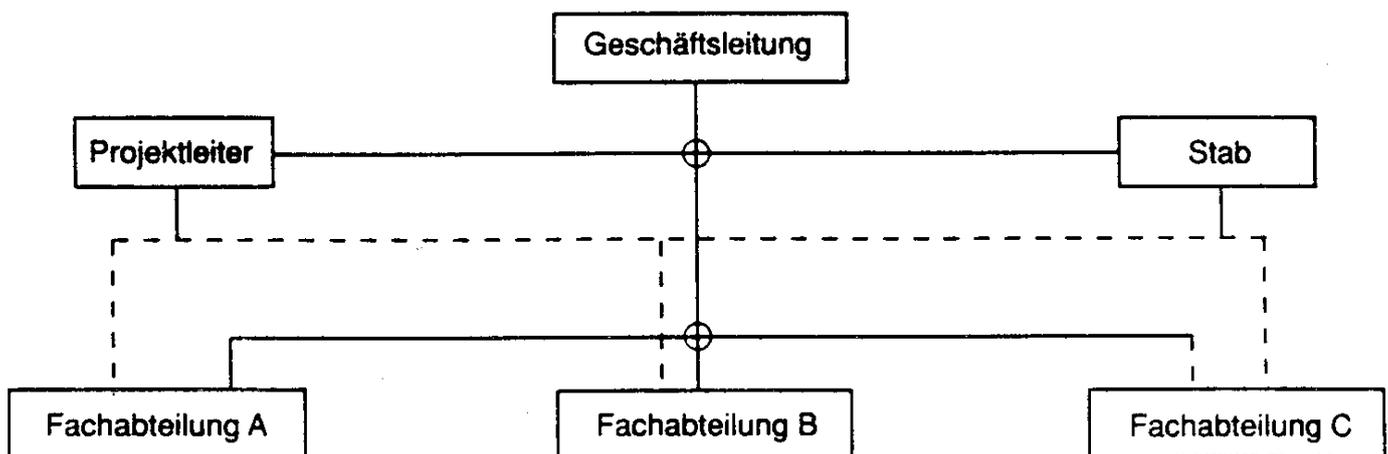
- volle Konzentration der Beteiligten auf die Projektziele
- gute Identifikation mit dem Projekt
- durch eindeutige Weisungsbefugnis ist die Möglichkeit gegeben zur straffen Führung und zu schneller Reaktion auf Störungen

### Nachteile:

- Bereitstellung der erforderlichen Ressourcen
- Auflösung der temporären Organisation am Projektende
- Einsatz von Spezialisten, die nur zeitweise benötigt werden

## Projektorganisation (2)

### Einfluss - Projektorganisation



Die Primärorganisation bleibt unverändert bestehen, lediglich ergänzt durch eine Stabstelle. Der Projektleiter befindet sich im Stab und hat gegenüber den Fachabteilungen nur Informations- und Beratungsbefugnis ohne irgendeine Exekutive

#### Vorteile:

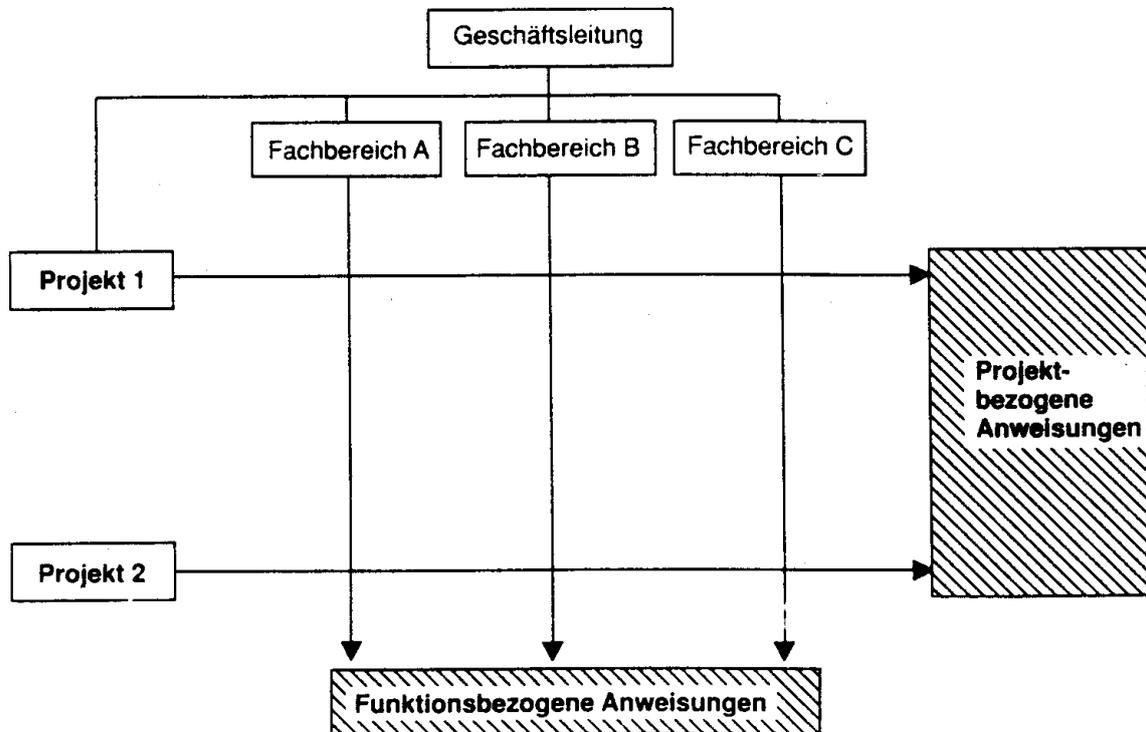
- hohes Maß an Flexibilität hinsichtlich des Personaleinsatzes
- keine organisatorische Umstellung erforderlich
- Sammlung von Erfahrung und deren Austausch über verschiedene Projekte ist relativ einfach

#### Nachteile:

- niemand fühlt sich für das Projekt verantwortlich
- Reaktionsgeschwindigkeit bei Störungen ist kleiner
- es bestehen Probleme, Schwierigkeiten über die Abteilungsgrenzen hinweg gemeinsam zu lösen

## Projektorganisation (2)

### Matrix- Projektorganisation



Jede Organisationseinheit wird zwei Instanzen unterstellt (1. der Fachabteilung, 2. dem Projektleiter)

#### Vorteile:

- Der Projektleiter fühlt sich für sein Projekt verantwortlich
- Es ist eine zielgerichtete Koordination verschiedener Interessen möglich
- Spezialwissen kann gezielt von einem Projekt zum anderen transferiert werden

#### Nachteile:

- die aufwendige Organisation
- Gefahr von Kompetenzkonflikten
- der große Personalaufwand für die Projektleitung

## Projektmanagement: Hilfsmittel

Sehr häufig wird eine Projektmanagementsoftware eingesetzt, die insbesondere bei hoher Komplexität oder vielen Benutzern unterstützend wirkt.

Bekannte Beispiele sind:

MS-Projekt

Redmine (Webbasiertes Open-Source PMS)

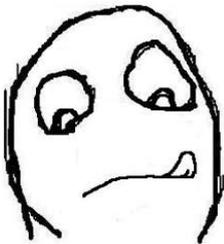
Wikis als Wissensmanagementsoftware

Ein beliebtes grafisches Element zur Unterstützung in Projekten ist der Gantt-Chart



## Zusammenfassende Fragen: Abschnitt 2.2

- Welche Kriterien kennzeichnen ein Projekt?
- Welche Rollen gibt es in einem Projekt?
- Nennen und beschreiben Sie die Teambildungsphasen.
- Welche Phasen hat der Projektmanagementprozess?
- Welche Organisationsformen gibt es und wie sind sie gekennzeichnet?
- Was ist ein typisches Hilfsmittel im Projektmanagement?



## Übungsaufgaben

1) Überlege, ob es sich bei den unten genannten Vorhaben um Projekte handelt. Nutze die Kriterien zur Charakterisierung von Projekten. Wenn es kein Projekt ist dann nenne das fehlende Kriterium.

- Umzug in eine andere Stadt
- Diskobesuch am Samstagabend
- Werbekampagne für ein Handy
- Flughafen BER
- Winterreifen kaufen und montieren
- Planung einer komplett neuen Küche incl. Raumumgestaltung
- Buchung eines Hotelzimmers
- Übernahme von Daten aus dem alten in ein neues IT-System

## Übungsaufgaben

2) Was halten Sie von folgender Aussage: „Alle Projekte funktionieren nach denselben Regeln. Ob ein Haus oder ein Schiff gebaut, eine Standardsoftware in einer großen Firma eingeführt oder ein Wahlkampf organisiert wird: Für einen Projektleiter ist dies alles kein Problem“

3) Diskutieren Sie: Welche Regeln innerhalb eines Team können dem Projekterfolg zuträglich sein?

## Softwareengineering

- 1 Nutzwertanalyse 
- 2 IT-Projekte 
- 3 **Vorgehensmodelle**
- 4 Geschäftsprozessmanagement
- 5 UML Diagramme

Quellen u.a:

Brandt-Pook, H.; Kollmeier, R.: Softwareentwicklung kompakt und verständlich; Wiesbaden 2008

Pichler; Scrum; 2007

[http://winfwiki.wi-fom.de/index.php/Qualit%C3%A4tssicherung\\_im\\_Softwareentwicklungsprozess\\_der\\_Software\\_Life\\_GmbH](http://winfwiki.wi-fom.de/index.php/Qualit%C3%A4tssicherung_im_Softwareentwicklungsprozess_der_Software_Life_GmbH)

## Motivation

- Systeme werden immer komplexer
- Entwicklung wird komplizierter
- Wirtschaftliche Bedeutung von Software steigt
- Wartung wird schwieriger
- Schätzungen werden unübersichtlich
- Überblick geht verloren
- immer kürzere Zeiträume
- immer mehr Anforderungen
- Änderungsmanagement
- Sicherstellung eines hohen Qualitätsniveaus
- Wartung und Weiterentwicklung von Altsystemen
- Guter Programmierstil und Entwicklungswerkzeuge



**CHALLENGE ACCEPTED**

→ Software entwickeln ist mehr als programmieren

## Definition

„**Vorgehensmodelle** in der Systementwicklung und im Softwareengineering beschreiben Folgen bzw. Bündel von Aktivitäten, die zur Durchführung eines (IT)-Projekts erforderlich sind. Üblich ist eine Gliederung in Phasen, Arbeitsabschnitte und Aktivitätenblöcke, durch die der Entwicklungsprozess in planbare und kontrollierbare Einheiten zerlegt wird“,

wobei man sich stark an den logischen und/oder chronologischen Reihenfolge der Einzelaktivitäten orientiert.

(Quelle: Enzyklopädie der Wirtschaftsinformatik).

## Definition (2)

### Kennzeichen:

- *Phasen*: VM legt fest, wie Projekte gleicher Art ablaufen
  - Phasen fassen Tätigkeiten des Gesamtprozesses zusammen
- *Rollen*: VM benennt die Teilnehmer an Projekten und beschreibt ihre Aufgaben
  - Nicht Person XY, sondern die verallg. Benennung und Aufgabenbeschreibung
- *Methoden*: VM stellt Methoden zur Verfügung, die bei der Bewältigung der Aufgaben benutzt werden
  - Methoden sind Verfahren und Anweisungen zu deren Nutzung, um bestimmte Ergebnisse zu erzielen.

(Quelle: Enzyklopädie der Wirtschaftsinformatik).

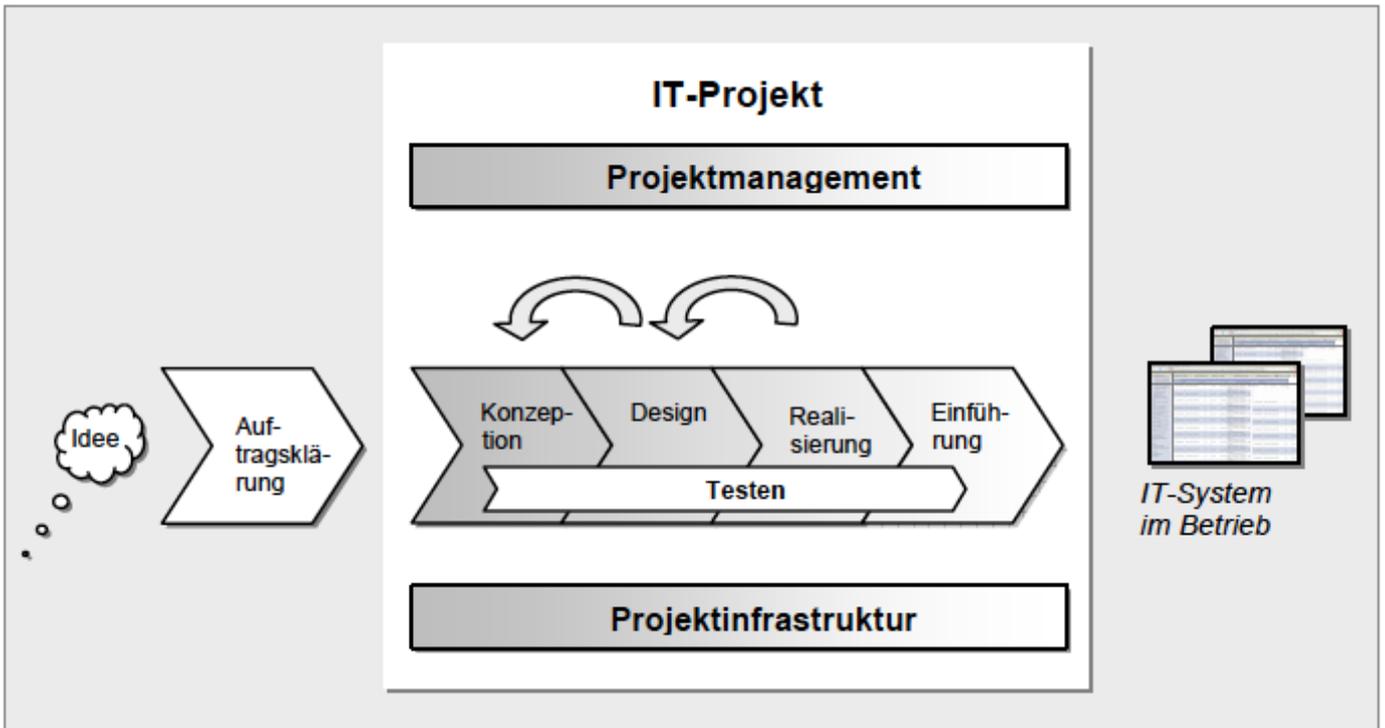
## Arten von Vorgehensmodellen (u.a.)

- **Phasenorientierte VM**
  - Basismodell
  - Wasserfall-Modell
  - V-Modell
  
- **Iterative VM**
  - Spiralmodell
  
- **Agile VM**
  - eXtreme Programming
  - SCRUM

## Phasenmodelle

- **Merkmal:**  
aufeinander folgendes Vorgehen mit klar definierten Phasen und Ergebnissen.
  
- **Vorteile:**
  - Einfach durchzuführen
  - schränkt Freiheitsgrade stark ein, daher auch für sehr große Projekte anwendbar
  - sehr effizient bei bekannten und konstanten Anforderungen
  - ist gut zu vermessen (notwendig z.B. für Prozessverbesserung)
  
- **Nachteile:**
  - Risiken gesammelt am Schluss („Big Bang“)

# Phasenmodelle – Basismodell



Quelle: Brandt-Pook; Softwareentwicklung kompakt und verständlich

## Phasenmodelle – Basismodell

### 1. Phase: Auftragserklärung

- Ziele:

- Ist die Umsetzung der Idee aus der Sicht des Geschäfts sinnvoll? (Wirtschaftlichkeit; Kosten; Geschäftsanalyse)
- Welche IT-Projekte zur Umsetzung der Idee anzuwenden sind (Machbarkeit; politische Interessen; Ausarbeitung der Idee).

## Phasenmodelle – Basismodell

### 2. Phase Konzeption

Leistungsumfang des Systems – WAS soll das beabsichtigte IT-System leisten (Eigenschaften und Fähigkeiten des IT-Systems aus Zielen ableiten)

- Dazu 3 Perspektiven:
  - Welche funktionale Anforderungen gibt es an das System?
  - Welche Daten werden in dem System gehalten?
  - Welche wichtigen Kennzahlen und weitere Anforderungen sind zu beachten?
- **Lösungsalternativen** beschreiben und bearbeiten; Nutzwertanalyse (Bsp.: technische Basis der IT-Lösung (Rechner, Betriebssystem usw.)).

## Phasenmodelle – Basismodell

### 3. Phase Design

„WIE soll das IT-System intern aufgebaut und gestaltet sein?“

▪ Aufgaben:

- Die Hard- und Softwarearchitektur zu klären
- Einzelne Komponenten innerhalb der Softwarearchitektur zu spezifizieren
- Transformation der Komponenten in die geplante IT-Landschaft zu planen.

## Phasenmodelle – Basismodell

### 4. Phase Realisierung (Implementation)

„Build it!!!“

- Ziele:
  - Das IT-System so zu erstellen, dass alle Anforderungen erfüllt sind und das IT-System beim Kunden eingeführt werden kann.
  - Die Konzeption und das Design in einer Programmiersprache zu implementieren.

### 5. Phase Einführung (Roll-Out)

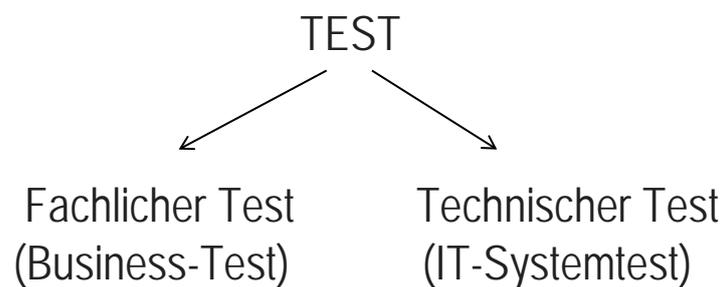
„Go live!!!“

- Aufgaben:
  - Die Abnahme des IT-Systems;
  - Die Überführung in den produktiven Betrieb (Big Bang; Step-by-Step);
  - Die Schulung der Anwender (KeyUser).

## Phasenmodelle – Basismodell

### 6. Phase Testen

- Aufgabe:  
mögliche Fehler in einem IT-System zu finden.  
Wichtig: in jeder Phase zu testen!

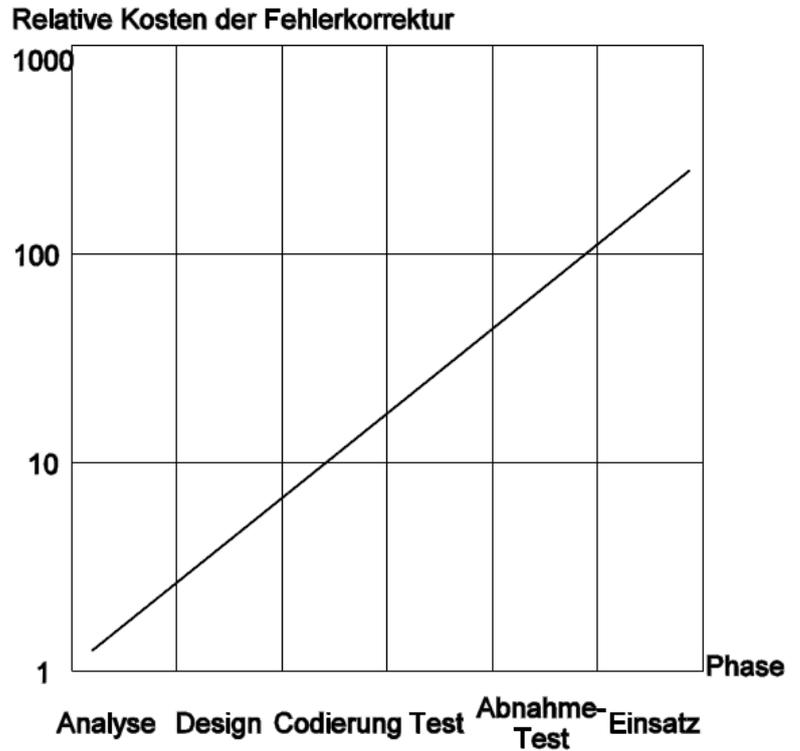


- Das Testen des IT-Systems läuft parallel und wird ständig durchgeführt.
- Durch Testen lassen sich Fehler aus dem System „fischen“.
- Ein Nachweis, dass ein System fehlerlos ist, ist nicht möglich.

## Phasenmodelle – Wasserfallmodell (Royce 1970)

# Phasenmodelle – Wasserfallmodell (Royce 1970)

Korrekturkosten  
nach Boehm



**Fehler im Einsatz: 100x teurer**

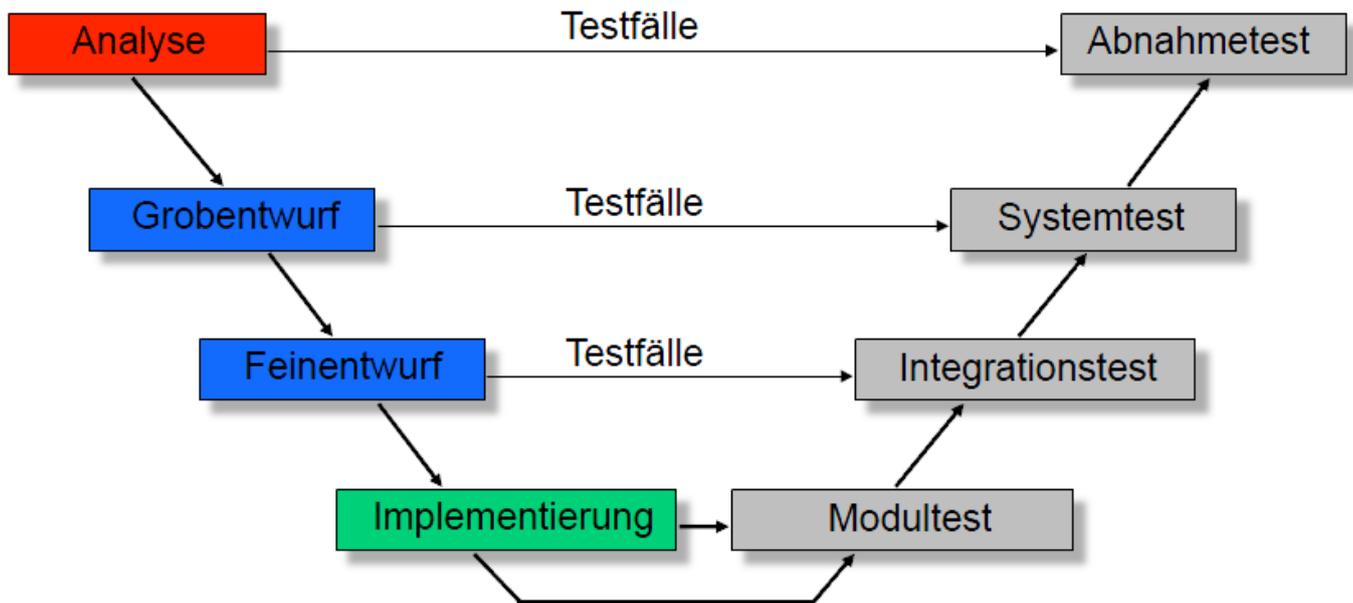
## Phasenmodelle – Wasserfallmodell

### Merkmale:

- Phasenmodell; die Phasen „Inbetriebnahme“ und „Wartung“ beschreiben das IT-System im produktiven Betrieb
- Die Phasen folgen strikt nacheinander. Eine Phase muss erst vollständig abgeschlossen sein, bevor die nächste Phase beginnt
- Ein einfaches und erfolgreiches Modell. Aber in komplizierteren Projekten ist es schwer anzuwenden
- Behandlung des Kernbereiches (ohne Projektinfrastruktur und Projektmanagement)
- Das Modell sagt nichts über Rollen und Methoden
- Notwendige „Kurskorrekturen“ nicht frühzeitig erkennbar

## Phasenmodelle – V-Modell (Boehm 1979)

- Idee: Jede Phase wird überprüft und getestet.
- Erweiterung des Wasserfallmodells



## Phasenmodelle – V-Modell (1986)

### Merkmale:

- Zu jeder Aktivität sind Ergebnisse (= Produkte) definiert, die erbracht werden sollen
- Die Verantwortung für die Produkte tragen definierte Rollen
- Die Anforderungen an die Produkte sind genau festgelegt
- Für große Projekte geeignet
- „Das Vorgehensmodell des Bundes“

## Iterative Modelle

- **Merkmal:**

*Iterativ* heißt, dass sich der Entwicklungsprozess mehrfach wiederholt: statt den „Wasserfall“ einmal zu durchlaufen, werden kleine Wasserfälle hintereinander gesetzt.

- **Vorteile:**

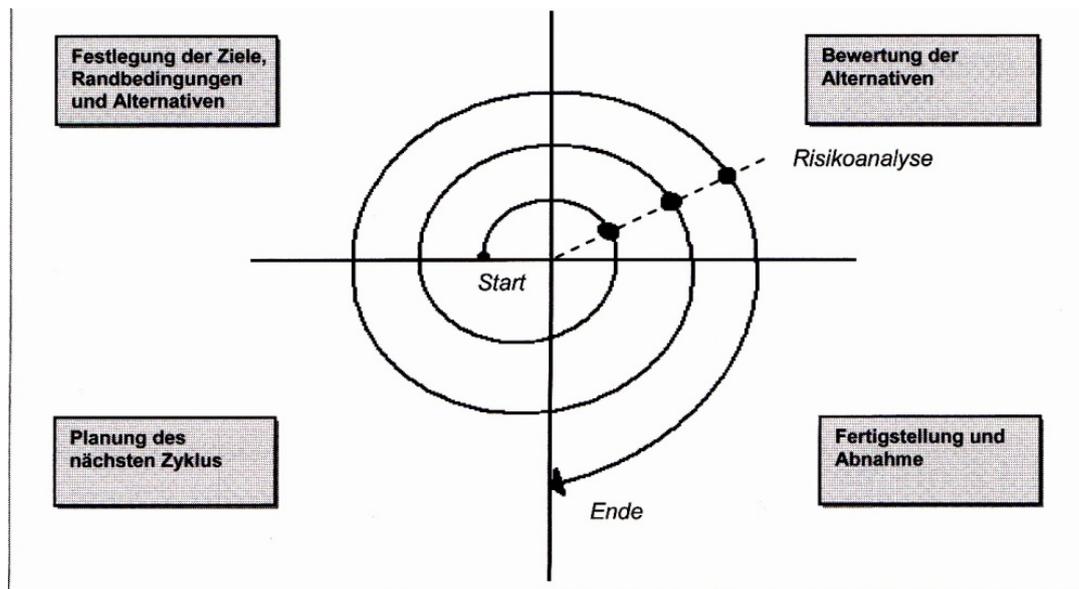
- Risiken können früher erkannt werden
- Schwankende Anforderungen können besser berücksichtigt werden
- Aufeinander aufbauende Auslieferung wird erleichtert.

- **Nachteile:**

- Mehrarbeit
- komplexeres Projektmanagement
- schwerer messbar.

## Iterative Modelle – Spiralmodell

- Idee: Immer wieder gleichartige Phasen durchlaufen während des Projektfortschritts und in jeder Runde wird eine Risikoanalyse ausgeführt.



## Iterative Modelle – Spiralmodell

- Ablauf:
  - Beginn mit der Festlegung der Ziele (grobe Gesamtziele), der Rahmenbedingungen und der Identifikation der möglichen Alternativen
  - Nähere Bewertung und Ausgestaltung der Alternativen; Entscheidung für Alternative
  - Umsetzung, Test und Abnahme bester Alternative
  - Planung nächster Runde
  
- Vorteil:
  - Risikobetrachtung als festgelegte Aufgabe in jeder Phase
  - Chance zur Korrektur!

## Agile Modelle

*Agil* = beweglich, dynamisch.

- **Merkmal:**

- kontinuierliche Anpassung an Änderungen.

- **Ziele:**

- Höhere Flexibilität als bei klassischen Modellen
- Software wird in regelmäßigen, kurzen Abständen dem Kunden präsentiert
- Flexible Reaktion auf Kundenwünsche
- Fokussierung auf die zu erreichenden Ziele
- Angehen von technische **und** sozialen Problemen
- nicht schwergewichtig und bürokratisch vorgehen
- Fail fast - fail cheap - fail early

### Manifest nach Beck 2001 (Auszug)

- Individuen und Interaktionen sind wichtiger als Prozesse und Werkzeuge.
- Funktionierende Programme sind wichtiger als ausführliche Dokumentation.
- Die stetige Abstimmung mit dem Kunden ist wichtiger als die ursprüngliche Leistungsbeschreibung in Verträgen.
- Der Mut und die Offenheit für Änderungen stehen über dem Befolgen eines festgelegten Plans.

## Agile Modelle (2)

### ▪ Vorteile:

- Gut einsetzbar bei unklaren Zielen und sich ändernden Anforderungen/Umgebung
- Verspricht besseres Kosten/Nutzen-Verhältnis
- Bessere Code-Qualität

### ▪ Nachteile:

- Ergebnis ist nicht vorhersagbar
- Qualitätseigenschaften können nicht garantiert werden
- Oft nicht nachvollziehbar, wie eine Funktion zustande kommt

## Agile Modelle (3)

- Basiswerte:
  - Kommunikation (im IT-Team; zwischen IT-Team und Kunden)
  - Einfachheit
  - Feedback (vom Kunden)
  - Mut (mutig, eigenverantwortlich und umsichtig handeln)
  - ➔ Wichtig ist gute Zusammenarbeit im Team!
  
- Methoden und Techniken, u.a.:
  - Pair-Programming
  - Test first (Testen)
  - Kurze Entwicklungszyklen
  - Gemeinsamer Codebesitz u. Programmierstandards
  - Customer on-site (Kundenbeteiligung)

## Agile Modelle – eXtreme Programming

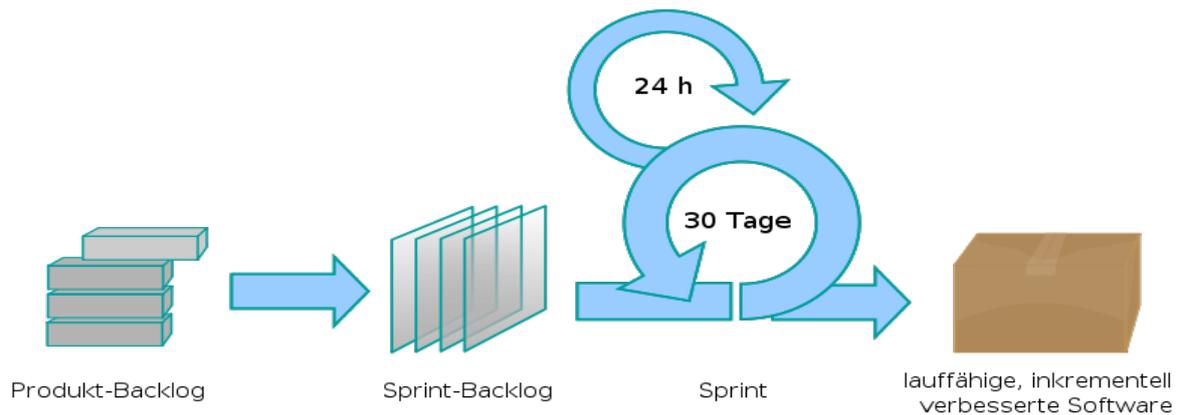
Idee: schnelle und flexible Anpassung an Änderungen; Befreiung vom unnötigen Ballast und Konzentration auf die Programmierung.



Copyright © 2003 United Feature Syndicate, Inc.

## Agile Modelle – SCRUM

- Idee: die Selbstorganisation von Entwicklerteams und die eigenverantwortliche Auswahl der eingesetzten Mittel. Viele Ähnlichkeiten zu XP, etwas strukturierter

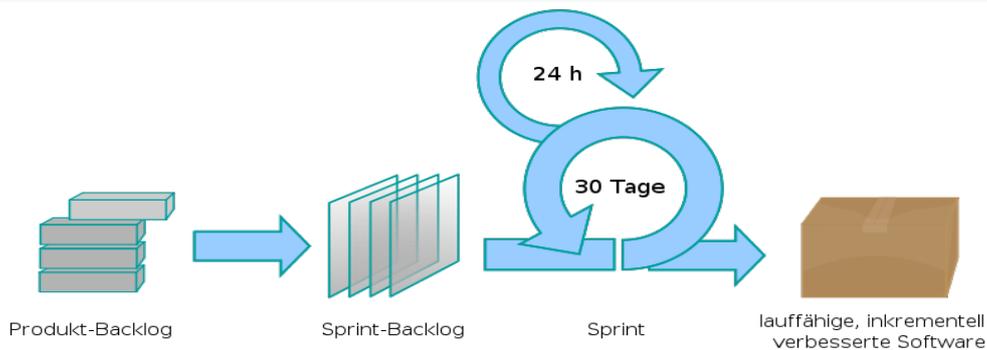


- Scrum-Projekte schreiten in Serien von Sprints iterativ voran
- Ein tägliches Treffen im Team (max. 15 Min.)

Täglich werden jedem Teammitglied folgende Fragen gestellt:

- Bist du gestern mit dem fertig geworden, was du dir vorgenommen hast ?
- Was hast du heute vor ?
- Gibt es ein Problem das dich behindert ?
- Gibt es etwas das wir verbessern können ?

## Agile Modelle – SCRUM

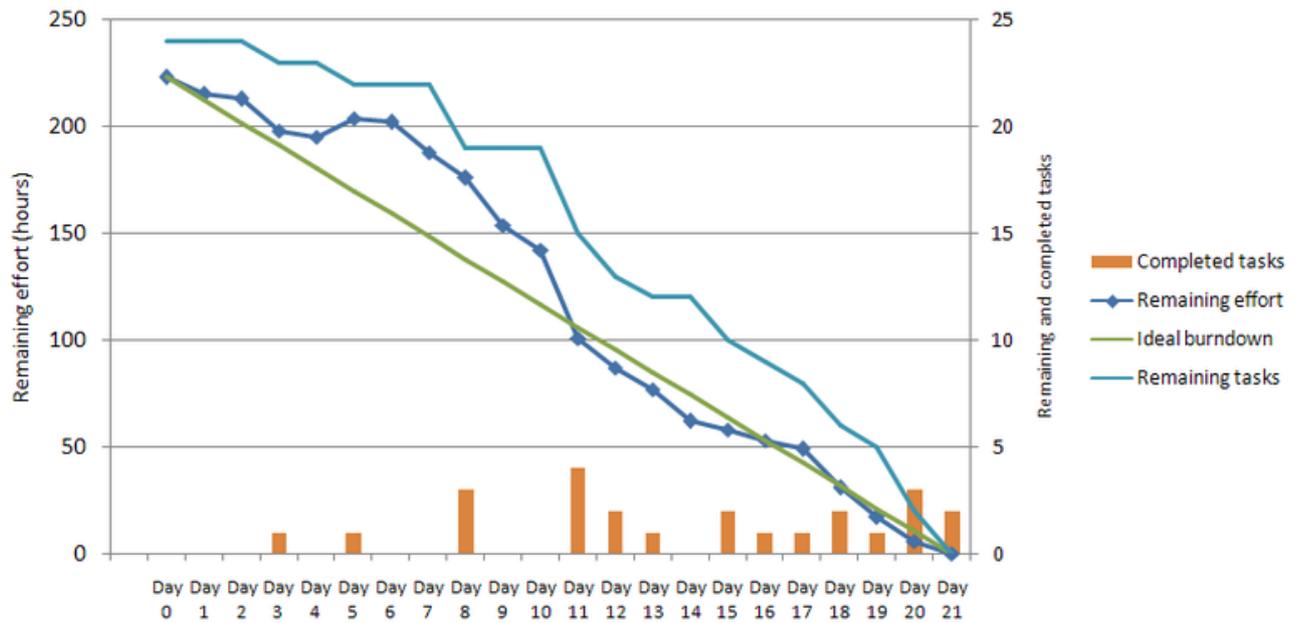


- **Produkt-Backlog:**  
Eine Liste aller gewünschten funktionalen und nicht-funktionalen Anforderungen
- **Sprint-Backlog:**
  - Team-Mitglieder wählen Tasks selbst aus, die Arbeit wird nie zugewiesen
  - Der geschätzte Restaufwand wird täglich aktualisiert
  - Jedes Team-Mitglied kann Tasks hinzufügen, löschen oder ändern

# Agile Modelle – SCRUM

- Graphische Darstellung des Sprintfortschritts.
- Gegenüberstellung von Restaufwand und geleisteter Arbeit

**Sample Burndown Chart**



## Einsatz im Unternehmen

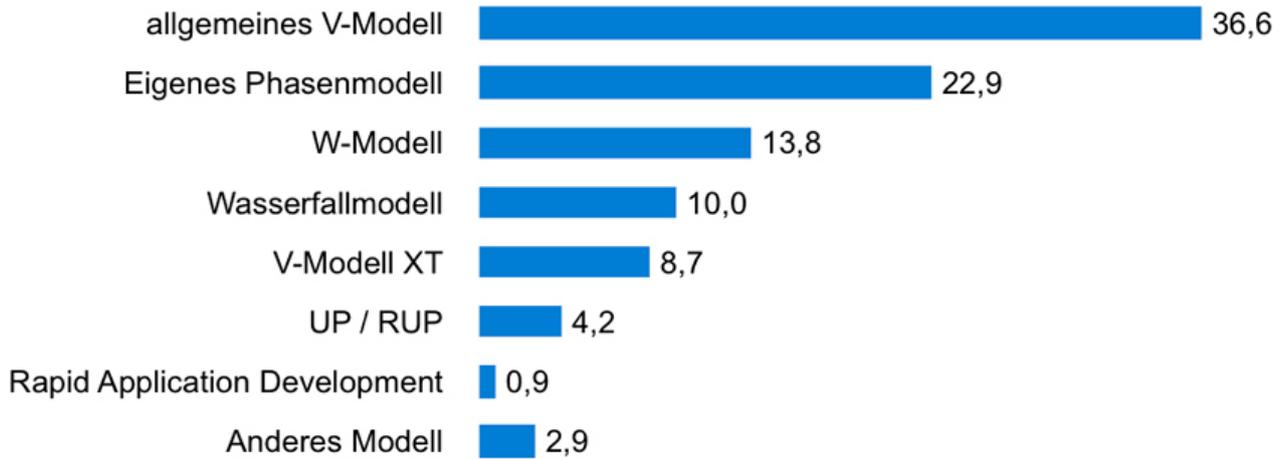
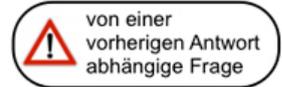
*Wie ordnen Sie Ihr Vorgehensmodell in der Softwareentwicklung ein, eher als ... (in %)*



© Softwaretest-Umfrage 2011: HS Bremen, HS Bremerhaven, FH Köln, ANECON, GTB, STB

## Einsatz im Unternehmen

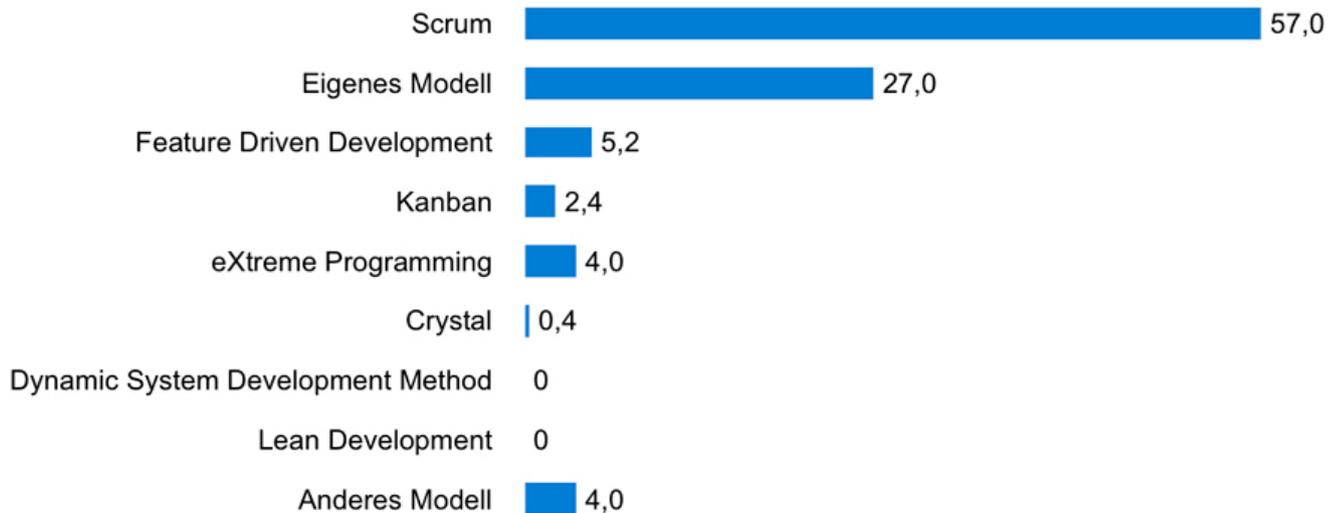
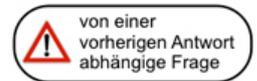
An welchem Phasenmodell zur Softwareentwicklung orientieren Sie sich in Ihren Projekten? (in %)



© Softwaretest-Umfrage 2011: HS Bremen, HS Bremerhaven, FH Köln, ANECON, GTB, STB

## Einsatz im Unternehmen

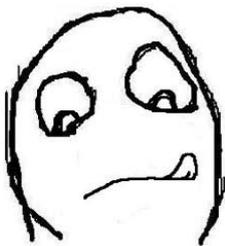
An welchem agilen Vorgehensmodell zur Softwareentwicklung orientieren Sie sich in Ihren Projekten? (in %)



© Softwaretest-Umfrage 2011: HS Bremen, HS Bremerhaven, FH Köln, ANECON, GTB, STB

## Zusammenfassende Fragen: Abschnitt 2.3

- Was ist ein Vorgehensmodell und was ist die Motivation?
- Welche Arten Vorgehensmodellen gibt es?
- Was zeichnet die Arten aus? Merkmale, Vor- und Nachteile?
- Welche Phasen hat das Basismodell? Was zeichnet Phase xy aus?
- Was ist ein Sprint im SCRUM?
- Anhand welcher Kriterien könnte man ein Vorgehensmodell auswählen?



## Übungsaufgaben

1) Wählen Sie für die folgenden Projekte jeweils ein Vorgehensmodell aus und begründen Sie Ihre Auswahl!

- a) Entwicklung eines einfachen Editors  
(Teamgröße 2 Entwickler, ca. 30.000 LOC, Lebensdauer 5 Jahre)
- b) Entwicklung eines Kfz-Steuergerätes  
(Teamgröße 5 Entwickler, ca. 30.000 LOC, Lebensdauer 10 Jahre)
- c) Entwicklung eines Buchhaltungssystems für eine  
Versicherungsgesellschaft  
(Teamgröße 50 Entwickler, ca. 400.000 LOC, Lebensdauer 25 Jahre)

Legende: LOC = Lines of Code

## Übungsaufgaben

	Wasserfallmodell	Spiralmodell	V-Modell	SCRUM
Projektgröße				
Benutzerbeteiligung				
Managementaufwand				
Anpassbarkeit				
zulässiger Änderungszeitpunkt				
Risikominimierung				
Qualität				
Entwicklungszeit				

Malte Wattenberg Datenbanken und Softwareengineering Wintersemester 2014/2015 196

Füllen Sie die obige Tabelle mit einer in 3 Ausprägungen skalierten Bewertung aus. (klein, mittel, groß / kurz, mittel, lang /.,.,.)

Arbeiten Sie im Team und diskutieren Sie!

## Softwareengineering

- 1 Nutzwertanalyse 
- 2 IT-Projekte 
- 3 Vorgehensmodelle 
- 4 **Geschäftsprozessmanagement**
- 5 UML Diagramme

## Lesen..Nachschlagen..Lernen..

### Literatur zu Geschäftsprozessmanagement

- Freund, J.;  
Praxishandbuch BPMN 2.0;  
3. Aufl.; 2013
- Gadatsch, A.:  
Grundkurs Geschäftsprozess-  
Management;  
7. Aufl.; Wiesbaden; 2012  
**online via Springerlink**



## Definition und Motivation

- Def.: Betriebliche Informationssysteme  
„Betriebliche Informationssysteme dienen dazu, die in einem Unternehmen zur Erfüllung des Unternehmenszwecks anfallenden Aufgaben der Informationsverarbeitung zu erfüllen.“ (Dessaules, FH-Bielefeld)
  
- Motivation
  - In (fast) allen Unternehmen ist Informationstechnologie zu einem wesentlichen Bestandteil für erfolgreiche Geschäftsabwicklung geworden
  - Viele Unternehmen können ohne Informationstechnologie nicht mehr erfolgreich ihre Geschäfte abwickeln
  - Betriebliche Informationssysteme sind zu einem wichtigen Wettbewerbsfaktor von Unternehmen geworden
  - Unternehmen wenden viel Zeit und Kosten auf, um ihre betrieblichen Abläufe und Informationssysteme wettbewerbsfähig aufzustellen

## Definition und Motivation

- Def.: Geschäftsprozess

“Ein Geschäftsprozess ist eine Anordnung von **Aktivitäten** über **Zeit** und **Raum** mit definierten **Anfang** und **Ende** und definierten **Eingaben** und **Ausgaben**“. (übersetzt nach Davenport 1993)

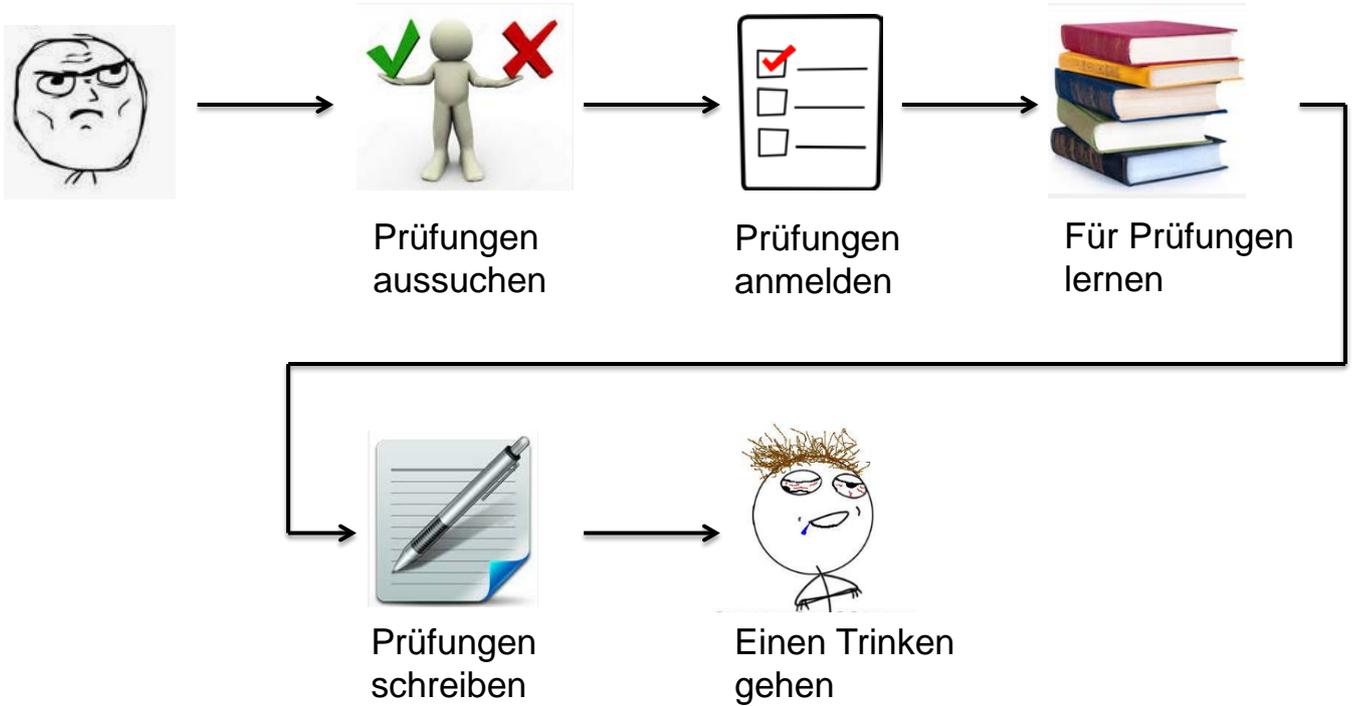
- Motivation

- Viele Beanstandungen, Fehler, Änderungen
- Hohe Produktkosten und Durchlaufzeiten
- Unzureichende Liefertreue und -fähigkeit
- Hohe Bestände
- Geringe Flexibilität

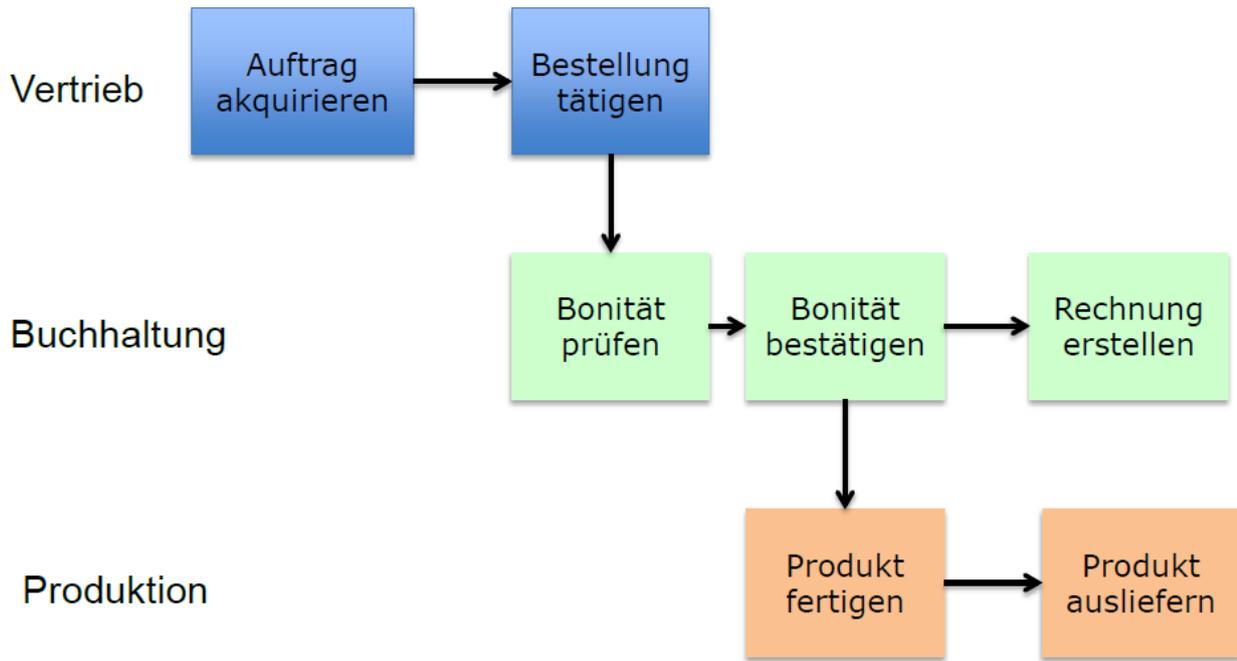


Um Prozesse systematisch in Unternehmen zu beherrschen, ist ein systematisches **Geschäftsprozessmanagement** erforderlich

# Beispielprozess: Prüfung an der HS-OWL



# Beispielprozess in einem ERP-System

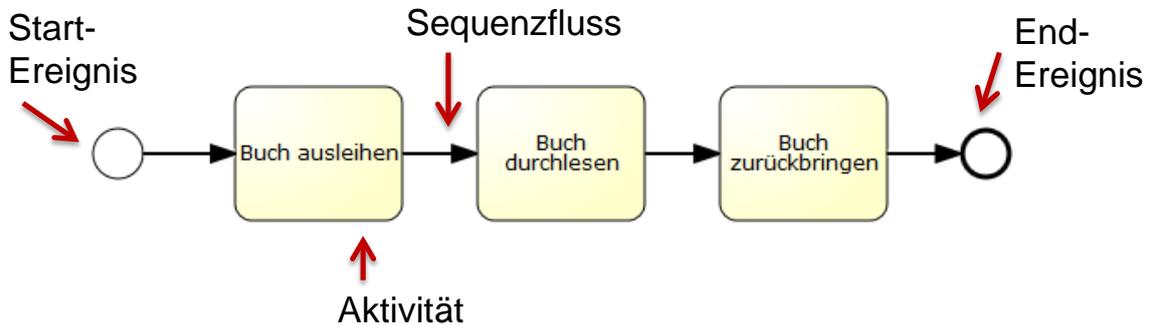


Quelle: Laudon et al.; S. 97

## Geschäftsprozessmodellierung mit BPMN

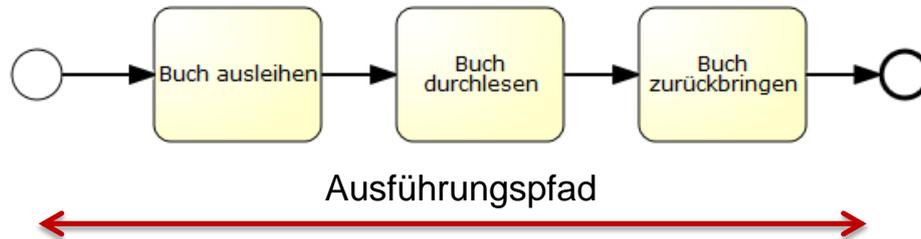
- Motivation:
  - Dokumentation der Prozesse
  - Wiederholte Ausführung der Prozesse an Hand des Prozessmodells
  - Simulation und Analyse von Prozessen
  - Optimierung von Prozessen
  - Automatische und IT-unterstützte Ausführung von Prozessen
  - Wir benötigen eine **eindeutige** Beschreibung der Prozesse
  
- Ein Prozessmodell beschreibt die **wesentlichen Aspekte** eines Prozesses und lässt **unwesentliche Aspekte** aus
  
- Wesentliche Aspekte:  
Eingabe, Ausgabe, Aktivitäten, Ablauf, Ressourcen

## Notation



- Aktivitäten: beschreiben Aufgaben, die im Rahmen des Prozesses ausgeführt werden
- Ereignisse: beschreiben, dass vor, während oder am Ende des Prozesses etwas passiert
- Sequenzfluss: beschreibt die zeitlich-logische Reihenfolge, in der andere Elemente zueinander stehen

## Notation



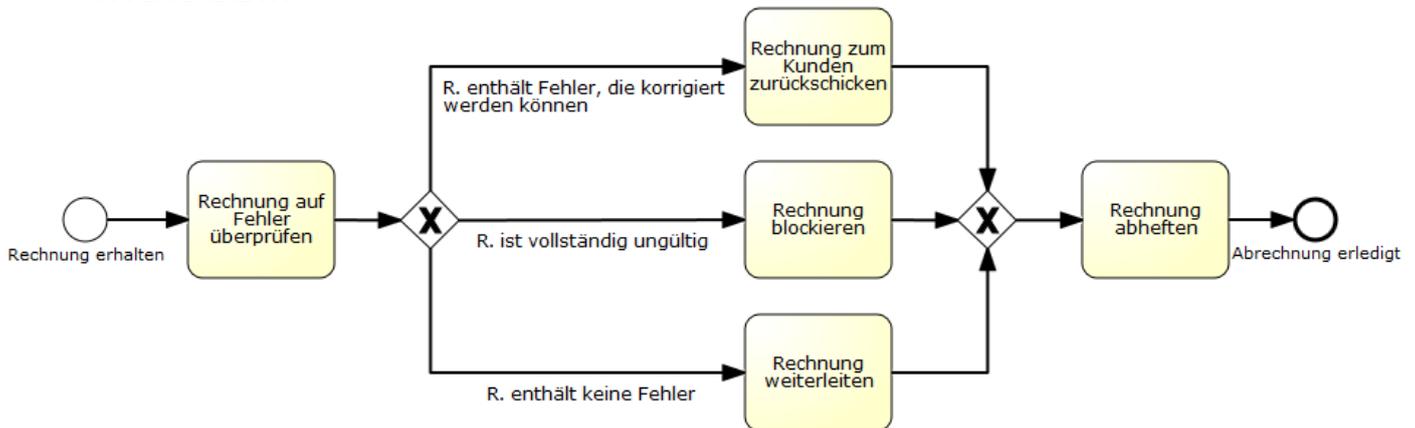
- Ausführungspfade zeigen die Aktivitäten, die in einer bestimmten Prozessinstanz des Prozessmodells durchlaufen worden sind
- Ein Ausführungspfad kann als Sequenz dargestellt werden

Notation: <A,B,C,D>

Bsp.: <Buch abholen, Buch durchlesen, Buch zurückbringen>

## Gateways: XOR-Split und XOR-Join

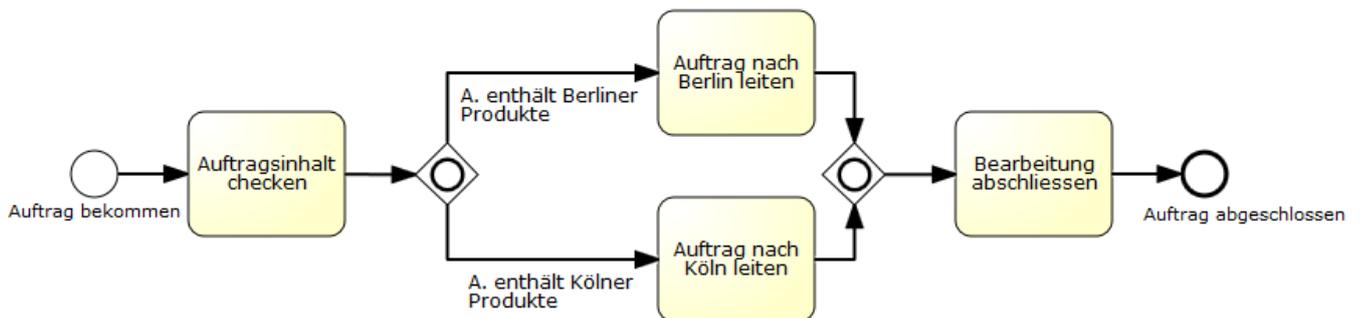
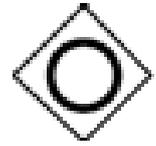
- Eine Entscheidung, bei der der Ausführungspfad in mehrere alleinstehende Ausführungspfade (**nur einer** kann ausgeführt werden) aufgeteilt wird, ist eine **Exklusive Entscheidung (XOR-Split)**



- Wenn in einem Prozess mehrere exklusive Ausführungspfade zusammengeführt werden, benutzt man einen **XOR-Join**.

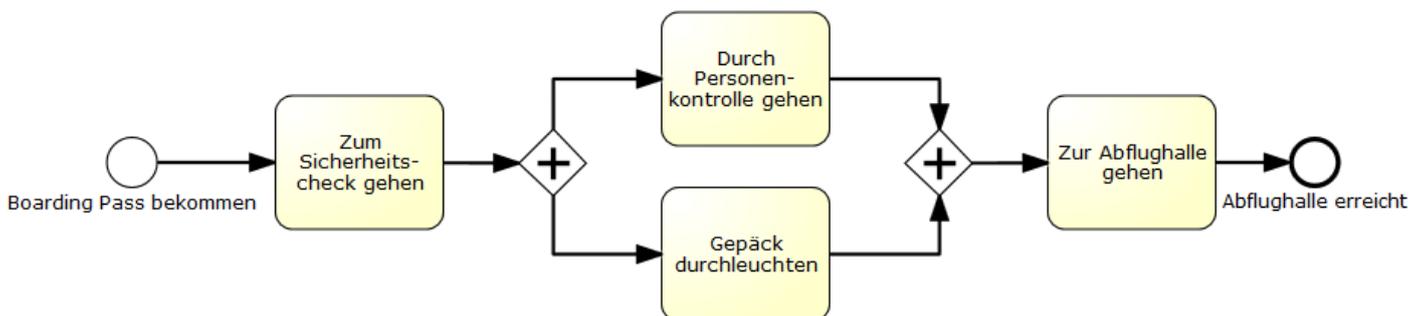
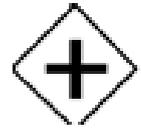
## Gateways: OR-Split und OR-Join

- Eine Entscheidung, bei der **eine oder mehrere** mögliche valide Alternativen **zur gleichen Zeit** ausgeführt werden können, wird als **OR-Split** modelliert.
- Die Zusammenführung wird durch einen **OR-Join** modelliert.



## Gateways: AND-Split und AND-Join

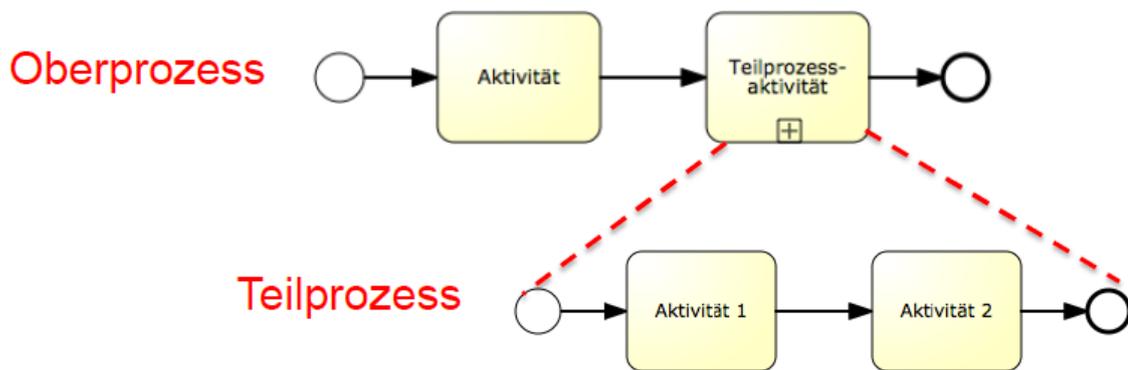
- Wenn zwei oder mehr Aktivitäten **keine Abhängigkeiten** haben, können sie **parallel** ausgeführt und ein einzelner Ausführungspfad in mehrere aufgeteilt werden.
- Ein **AND-Join** wird dazu benutzt, mehrere durch **AND-Split** aufgeteilte Ausführungspfade wieder zusammenzuführen.



## Teilprozesse

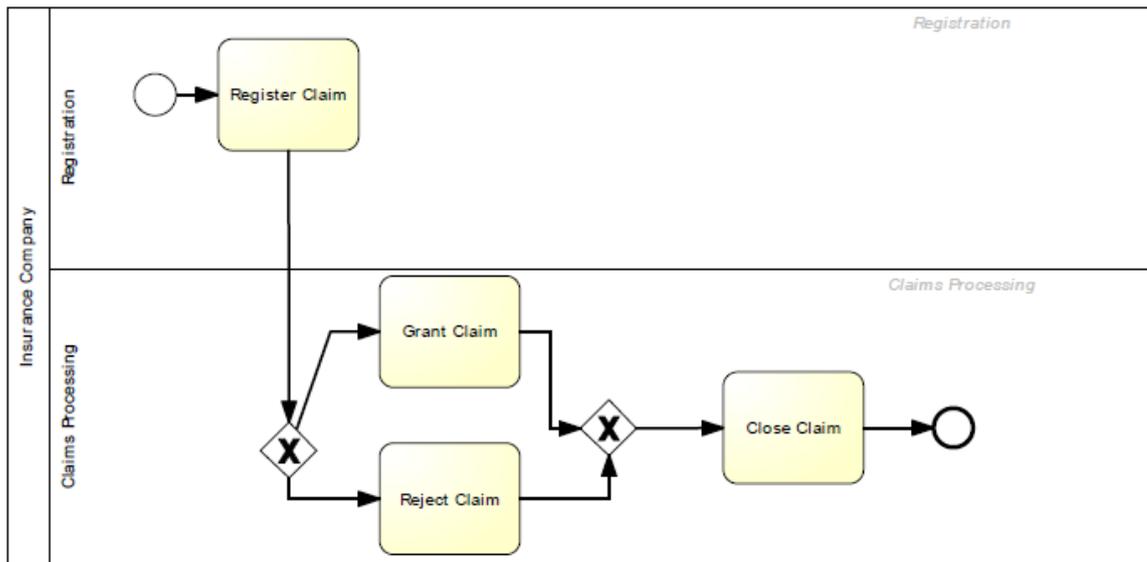
Wenn verwandte Teilaktivitäten das gleiche Ziel erreichen und/oder der Prozess zu groß wird

→ Bildung von Teilprozessen  
(Subprozess / Unterprozess)



## Lanes

- Eine Lane repräsentiert eine bestimmte Ressource (Person, Softwaresystem oder Hardwarekomponente)
- Lanes dienen dazu, in einem Prozess Aktivitäten dieser Ressource zu zuordnen. Sie können zudem verschachtelt sein



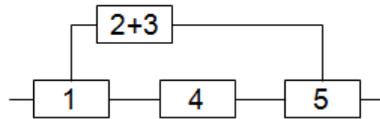
## Prozessoptimierung

- Wichtige Einflussfaktoren für Prozessqualität
  - Durchlaufzeiten der Prozesse
  - Kosten, die Prozesse verursachen
  
- Ein Prozess verursacht bei seiner Ausführung Kosten durch
  - Arbeitseinsatz von Mitarbeitern, die eine Aktivität ausführen
  - IT Systeme, die zur Ausführung benötigt werden
  - Eine Aktivität an sich (z.B. Produktionskosten)



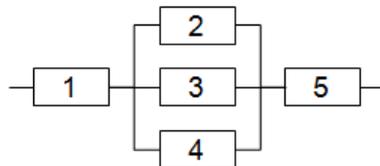
# Prozessoptimierung

**Zusammenfassen**



• **Zusammenlegung von Aktivitäten**

**Parallelisieren**



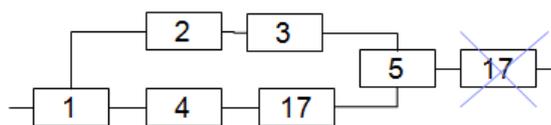
• **Erhöhung der Arbeitsteilung**

**Beschleunigen**



• **Bereitstellung von Arbeitsmitteln zur effizienten Aufgabenerledigung**  
• **Vermeidung von Warte- und Liegezeiten**

**Verlagern**

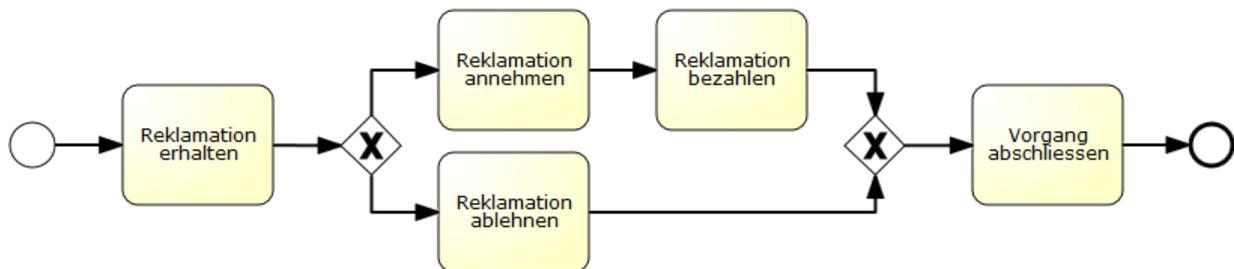


• **Früherer Beginn von bisher nachgelagerten Aktivitäten**

Quelle: Gadatsch, S. 21

## Prozesssimulation

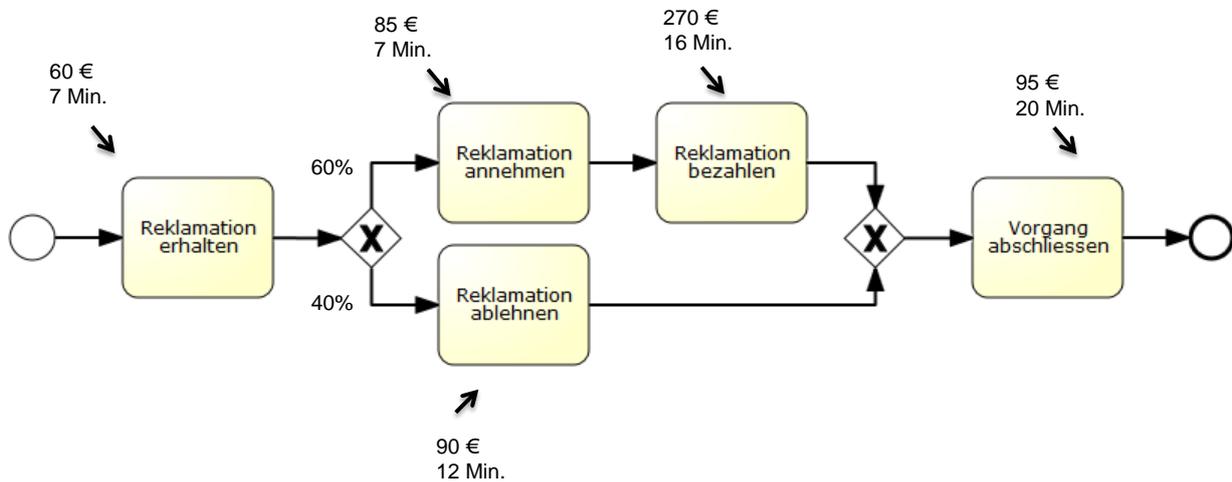
- Die Simulation von Prozessmodellen erlaubt es, Informationen über die Ausführung des Prozesses zu erhalten und zu analysieren
- Typische Fragen:
  - Was kostet die Prozessausführung im Mittel?
  - Wie lange dauert die Prozessausführung im Mittel?



Welche Daten werden von welchen Bestandteilen benötigt?

## Beispielaufgabe

- Berechnen Sie die mittleren Kosten und Dauer der Prozessausführung!



# Werkzeuge: Signavio.com

The screenshot shows the Signavio BPMN editor interface. The title bar includes 'BPMACADEMIC INITIATIVE', 'Prozesskosten', and 'Grafischer Editor Malte Wattenberg'. A left sidebar lists modeling elements under 'BPMN 2.0 / Kernelemente', including Task, Subprocess, Gateways, and various flows. The main workspace displays a BPMN diagram for a complaint handling process:

- Start event (circle) leads to the task 'Reklamation erhalten'.
- 'Reklamation erhalten' leads to an exclusive gateway (diamond with 'X').
- The gateway splits into two paths: 'Reklamation annehmen' and 'Reklamation ablehnen'.
- 'Reklamation annehmen' leads to the task 'Reklamation bezahlen'.
- 'Reklamation bezahlen' leads to a second exclusive gateway (diamond with 'X').
- 'Reklamation ablehnen' also leads to this second gateway.
- The second gateway leads to the task 'Vorgang abschliessen'.
- 'Vorgang abschliessen' leads to an end event (circle).

At the bottom of the interface, the footer contains: 'Malte Wattenberg Datenbanken und Softwareengineering Wintersemester 2014/2015' and a blue box with the number '216'.

[academic.signavio.com](http://academic.signavio.com)

Eine nicht so umfangreiche Alternative ist (ohne Anmeldung) [draw.io](http://draw.io)

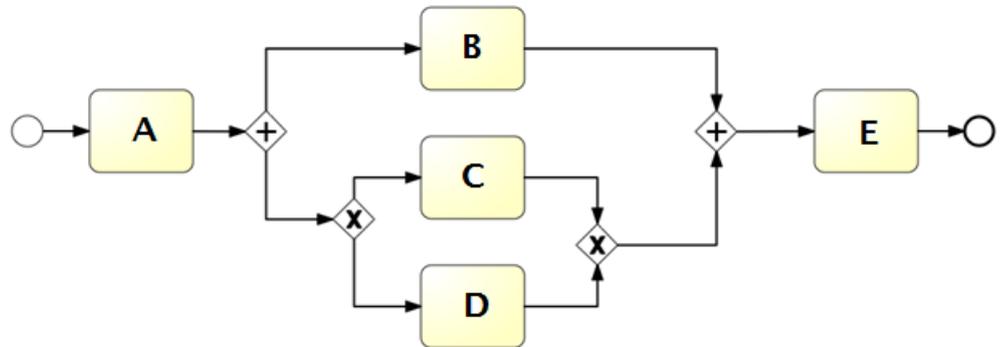
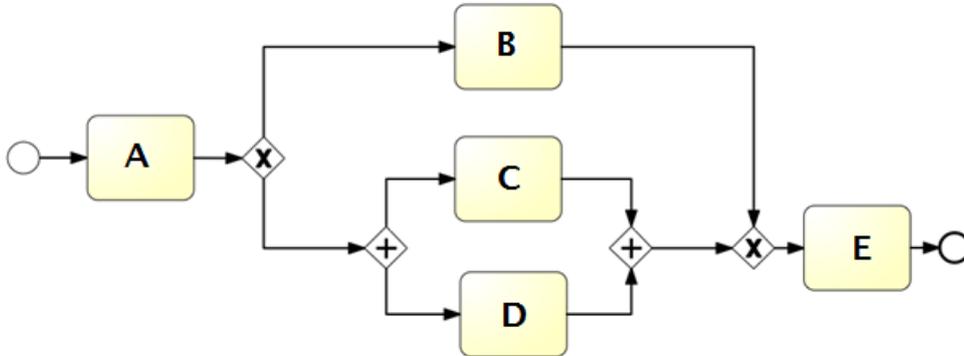
## Übungsaufgaben

### 1. Modellieren Sie folgende Prozesse als BPMN (zunächst per Hand, danach in Signavio)

- Urlaubsantrag: Der Antragssteller füllt ein Formular aus. Der Urlaubsantrag wird an den Weisungsberechtigten weitergeleitet. Er prüft den Antrag. Wenn der Antrag in Ordnung ist wird der Antrag freigegeben, ansonsten zurückgewiesen. Der Antragsteller wird informiert.
- PC-Kauf: Ein Kunde möchte einen PC kaufen. Zunächst wählt er die Marke aus. Danach wählt er die Ausstattung. Anschließend hat er die Möglichkeit, eine Garantie dazu zukaufen. Sofern er das wünscht, kann er die Dauer auswählen. Zuletzt wird das Bargeld für den Kauf übergeben
- Wohnungsabnahme: Zunächst wird eine Mängelliste der Wohnung erstellt. Danach wird die Wohnung gereinigt und parallel dazu gelüftet. Sollten Mängel festgestellt worden sein, werden sie beseitigt. Zuletzt erfolgt die Schlüsselübergabe.

# Übungsaufgaben

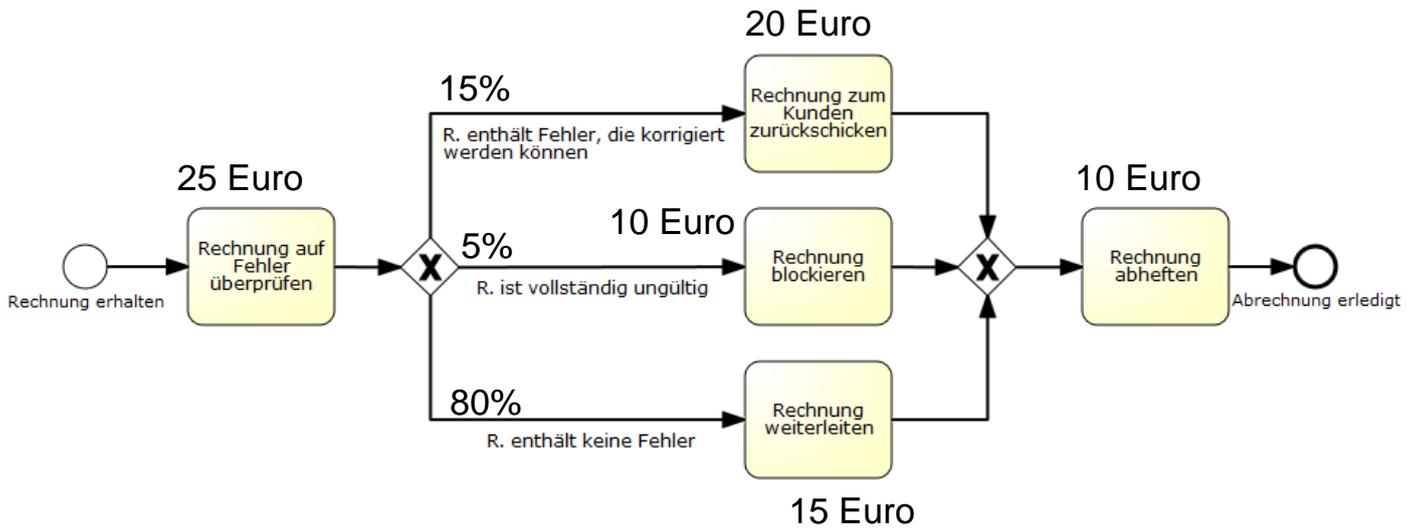
2. Notieren Sie jeweils drei unterschiedliche Ausführungspfade:



# Übungsaufgaben

## 3. Simulieren Sie die Durchschnitte per Hand und mit Signavio

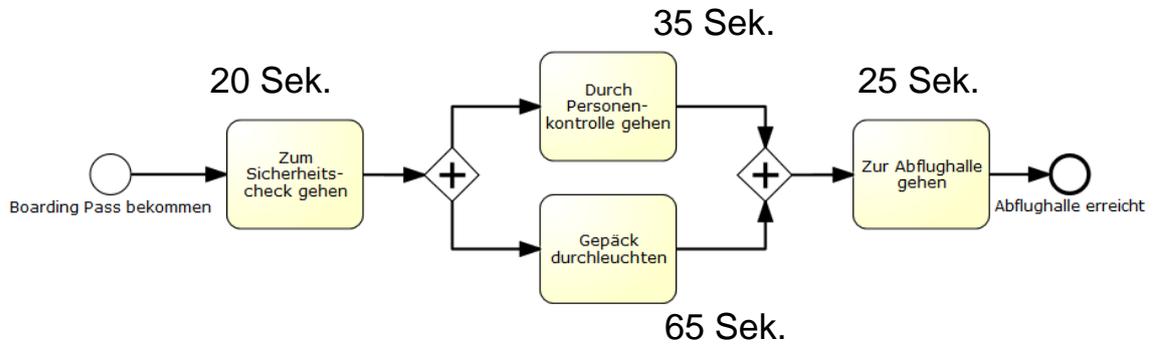
a.



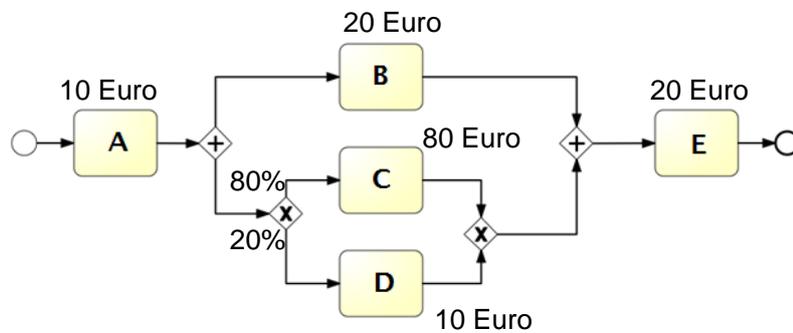
# Übungsaufgaben

## 3. Simulieren Sie die Durchschnitte per Hand und mit Signavio

b.



c.



## Softwareengineering

- 1 Nutzwertanalyse 
- 2 IT-Projekte 
- 3 Vorgehensmodelle 
- 4 Geschäftsprozessmanagement 
- 5 UML Diagramme

## UML Diagramme

- 4.1 **Überblick**
- 4.2 Aktivitätsdiagramm
- 4.3 Use-Case Diagramm
- 4.4 Business-Objekt-Modell  
(in Form eines Klassendiagramms)
- 4.5 Zustandsdiagramm
- 4.6 Objekt-Sequenz Diagramm

## Lesen..Nachschlagen..Lernen..

## Literatur zu UML u.a.

- Rupp, Christiane et. al.  
UML2 glasklar Praxiswissen für die UML Modellierung  
Hanser, 2007
- Brandt-Pook, H.; Kollmeier, R.:  
Softwareentwicklung kompakt und verständlich;  
Wiesbaden 2008  
**online via Springerlink**



## UML = Unified Modeling Language

### Motivation:

- Komplexität verschiedener Systeme ist unterschiedlich groß
- Es gibt unterschiedliche Sichtweisen auf die Systeme

### Beispiel

#### Hundehütte

- kann von einer Person gebaut werden
- Geringe Anforderungen an den Entwurf
- Simpler Prozessablauf
- Einsatz einfacher Hilfsmittel



#### Mehrfamilienhaus

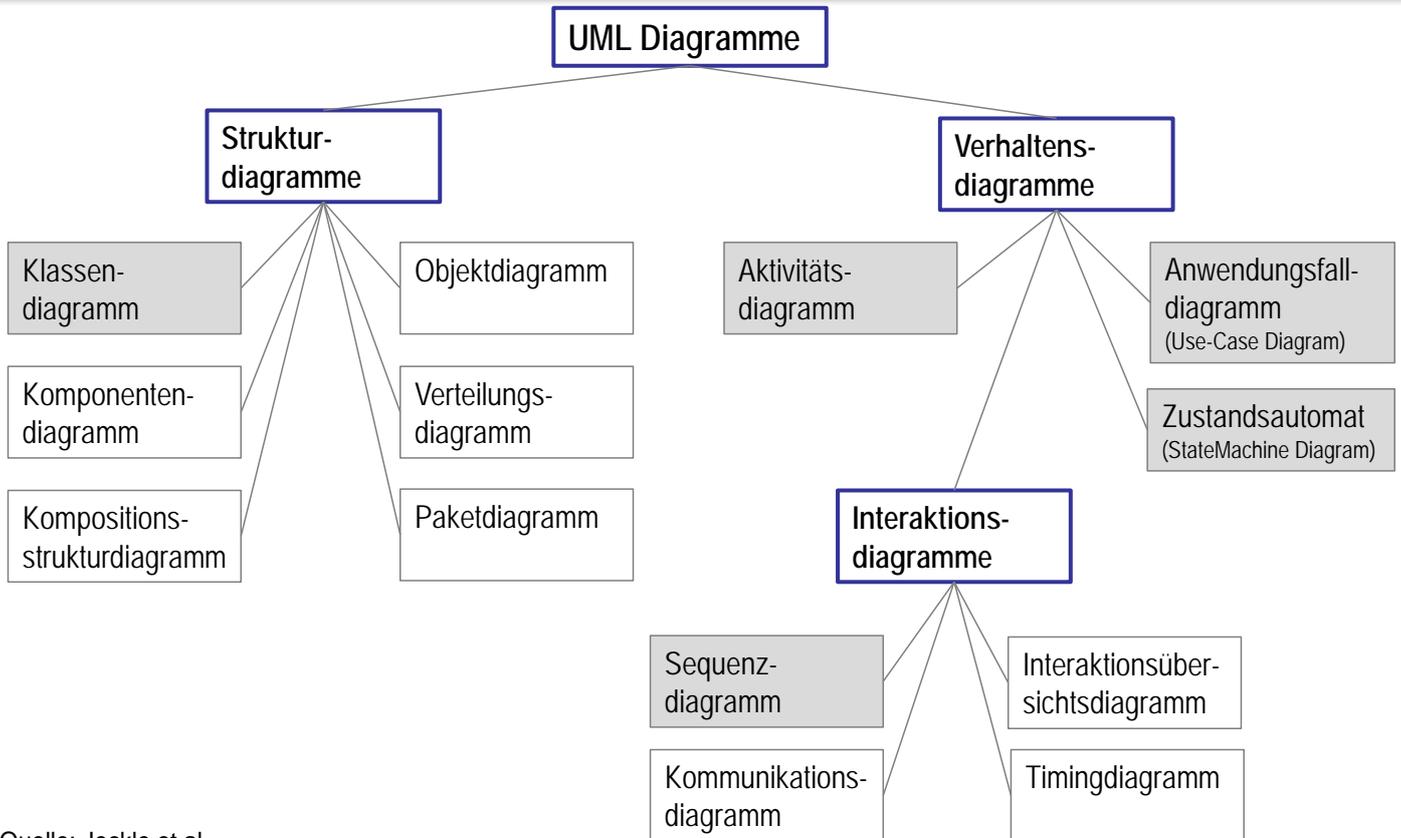
- wird von einem Team erbaut
- Komplexer Entwurf durch Architekten
- Umfassender Prozess mit vielen Rollen und Aufgaben
- Einsatz aufwändiger Werkzeuge



## UML – Idee

- UML ist eine Sprache und Notation zur Spezifikation, Konstruktion, Visualisierung und Dokumentation von Modellen für Softwaresysteme.
- UML ist ein Sammelbegriff für bestimmte standardisierte Diagrammtypen, die Softwaresysteme abbilden.
- Ein System wird häufig durch mehrere Diagramme abgebildet, da die verschiedenen Diagramme verschiedene Sichtweisen (z.B. Ablaufmöglichkeiten, Folgen von Zuständen etc.) auf das System liefern.
- Grafisch und objektorientiert
- Weltweit anerkannt

# UML – Diagrammarten



Quelle: Jeckle et al.

## Strukturdiagramme

- » Modellierung Systemstruktur
- » Modellierung statischer Aspekte des Systems

## Verhaltensdiagramme

- » Modellierung von dynamischen Aspekten (Verhalten) eines Systems

## UML – Idee (2)

### Unterschied zu ER-Modell

- Der wesentliche Unterschied zwischen objektorientierten Konzepten wie UML zu Entity-Relationship Modellen besteht darin, dass Verhalten beim ER-Modell nicht modelliert werden kann. Das ER-Modell enthält keine Operationen oder Nachrichten und es benutzt Schlüssel zur Identifikation von Entitäten.
- Bei UML besitzt jedes Objekt ein einzigartiges Identifikationsmerkmal, welches unabhängig von Attributen ist.

## UML Diagramme

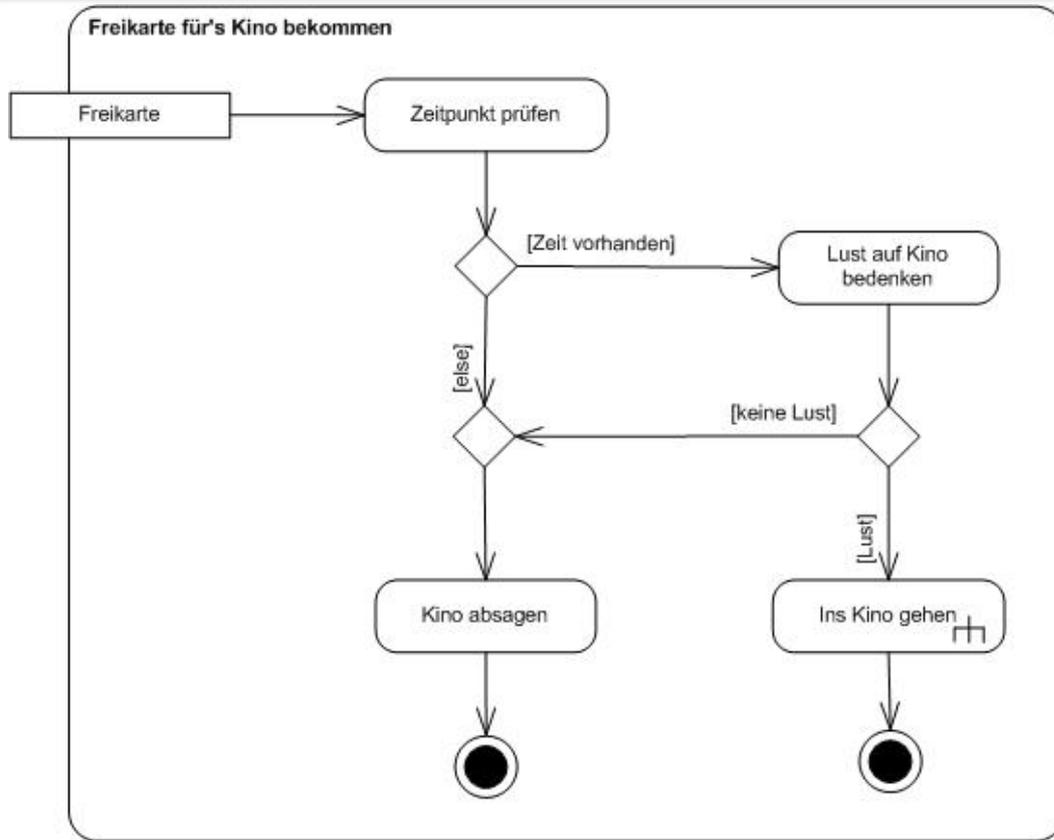
- 4.1 Überblick 
- 4.2 **Aktivitätsdiagramm**
- 4.3 Use-Case Diagramm
- 4.4 Business-Objekt-Modell  
(i.F.e. Klassendiagramms)
- 4.5 Zustandsdiagramm
- 4.6 Objekt-Sequenz Diagramm
- 4.7 Zusammenspiel

## Aktivitätsdiagramm

*Fragestellung: Wie sieht der Geschäftsablauf im Detail aus?*

- Eignen sich zur Darstellung von Abläufen aller Art, vor allem um Geschäftsabläufe in einer frühen Projektphase darzustellen und zu beschreiben.
- Durch die nahezu selbst erklärenden Symbole einfach zu verstehen. Trotzdem lassen sich auch komplizierte Abläufe darstellen.  
→ Einfach, aber ausdrucksstark.
- Das Diagramm wird häufig um einen Text erweitert, der das Diagramm und die einzelnen Funktion erklärt, falls dies nicht selbst erklärend ist.

# Aktivitätsdiagramm – Beispiel



## Aktivitätsdiagramm – Notation

Element	Name	Bedeutung
	Aktivität	<ul style="list-style-type: none"><li>• Alles was man auf dem Diagramm sehen kann. Ein Rahmen für das Diagramm.</li><li>• Oben links steht der Name der Aktivität</li></ul>
	Aktion	<ul style="list-style-type: none"><li>• Beschreibt, was passiert – Ein Prozess oder Funktionalität</li><li>• Kleinster Einzelschritt innerhalb der Aktivität</li></ul>
	Objektknoten	<ul style="list-style-type: none"><li>• An der Aktivität beteiligte Daten oder Objekte, also „anfassbare“ Dinge</li></ul>

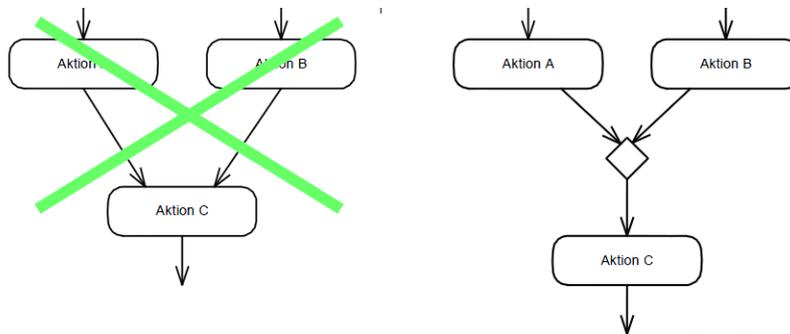
## Aktivitätsdiagramm – Notation

Element	Name	Bedeutung
	Verzweigungs-knoten	<ul style="list-style-type: none"><li>• Unter bestimmten Bedingungen wird hier derjenige Pfad gewählt, auf den die Bedingung zutrifft</li><li>• Nur einer der Wege wird beschriftet</li><li>• Der Verzweigungsgrund muss angegeben werden</li><li>• Exklusive ODER-Verzweigung (<b>XOR</b>)</li></ul>
	Verbindungs-knoten	<ul style="list-style-type: none"><li>• Gegenstück zum Verbindungsknoten</li><li>• Vorher verzweigte Wege werden nun wieder zusammengeführt und vereinigen sich zu einem einzigem Pfad</li></ul>

## Aktivitätsdiagramm – Notation

Element	Name	Bedeutung
	Kante	<ul style="list-style-type: none"> <li>• Verbindung zwischen Aktionen</li> <li>• Pfeile geben die Richtung an</li> </ul>

- Wege, die durch Verzweigungsknoten aufgespalten werden, werden typischerweise\* durch Verbindungsknoten wieder zusammengeführt.
- In eine Aktion läuft genau eine Kante ein und es geht genau eine Kante ab: sonst Verzweigungs- bzw. Verbindungsknoten nutzen:

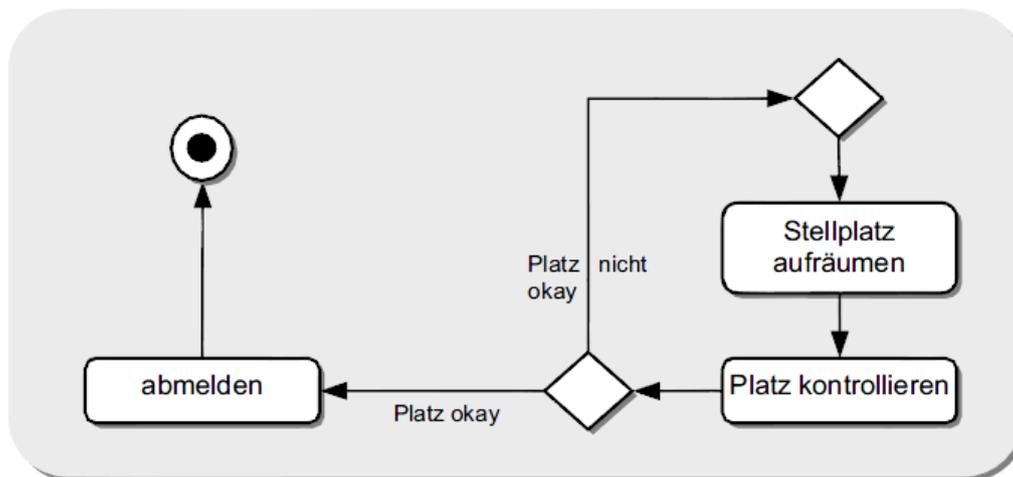


Eine Ausnahme wäre z.B. der Fall, wenn es mehrere Endpunkte im Diagramm gibt.

## Aktivitätsdiagramm – Notation

Modellierung von Wiederholungen:

- Typisches Szenario in der Programmierung
- Endlosschleife möglich
- Modellierung durch Verbindungs- und Verzweigungsknoten



## Aktivitätsdiagramm – Notation

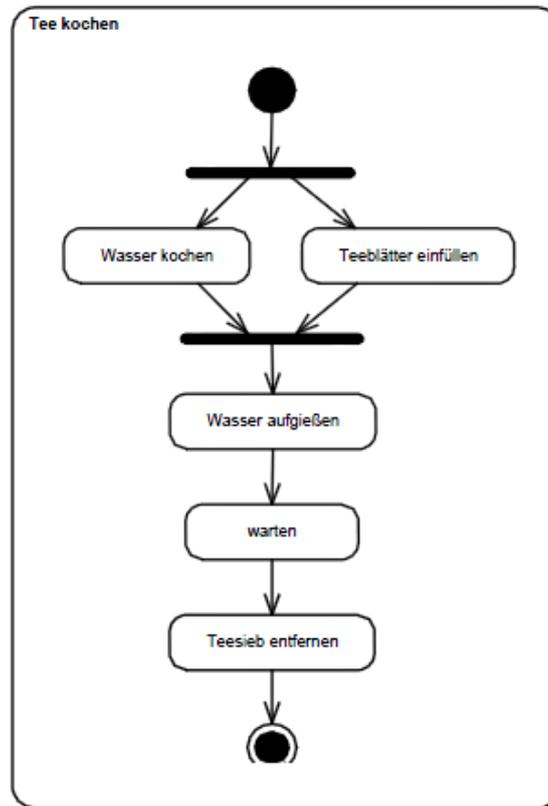
Element	Name	Bedeutung
	Parallelisierung	<ul style="list-style-type: none"><li>• Aufspaltung des Pfades.</li><li>• Ab hier können Aktionen gleichzeitig und unabhängig voneinander erfolgen</li></ul>
	Synchronisation	<ul style="list-style-type: none"><li>• Gegenstück zur Parallelisierung. Mehrere Pfade werden hier zusammengeführt, also „synchronisiert“</li><li>• Implizite <b>UND-Verbindung</b> der Pfade</li></ul>

## Aktivitätsdiagramm – Notation

Beispiel Parallelisierung:

Wasser muss gekocht sein und Teeblätter eingefüllt, bevor Wasser aufgegossen werden kann.

Ob gleichzeitig oder nacheinander ist egal, ebenso dann die Reihenfolge.

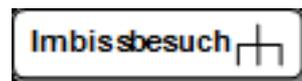


## Aktivitätsdiagramm – Notation

Element	Name	Bedeutung
	Startknoten	<ul style="list-style-type: none"><li>• Anfang des Diagramms</li><li>• Objekt statt Startknoten möglich, wenn die Existenz einer konkreten Instanz (bspw. eine Freikarte für's Kino) eine Aktivität auslöst</li></ul>
	Endknoten	<ul style="list-style-type: none"><li>• Ende des Diagramms</li><li>• Es kann mehrere Endknoten geben</li></ul>

## Aktivitätsdiagramm – Notation

- lässt sich sehr stark verfeinern, wird aber meist vermieden, wenn dies nicht unbedingt erforderlich ist. Das zu verfeinernde Element wird mit einem Gabelsymbol gekennzeichnet und kann in einem Unterdiagramm verfeinert werden.

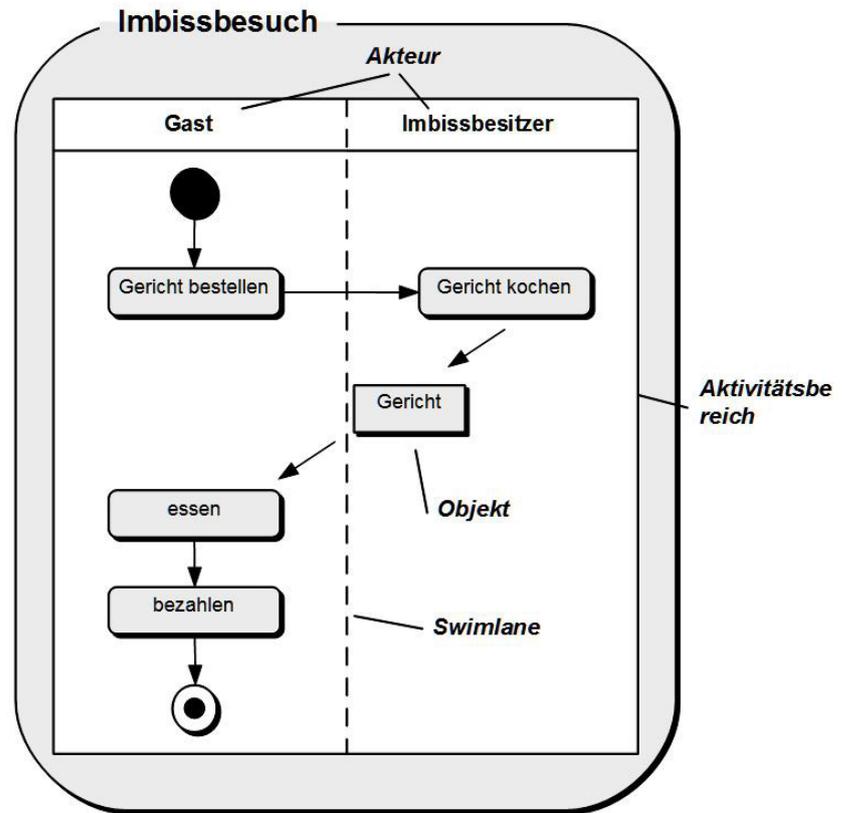


## Aktivitätsdiagramm – Beispiel 2

Aufteilung des Diagramms (und der enthaltenen Aktionen) in unterschiedliche Verantwortungsbereiche.

Welcher Akteur hat welche Aktion?

Trennung der Akteure durch eine „Swimlane“.



Quelle: Brandt-Pook; Softwareentwicklung kompakt und verständlich

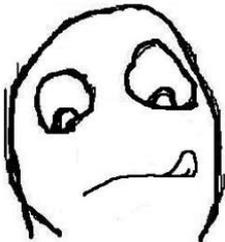
## Aktivitätsdiagramm

Diese Stärken hat das Diagramm:

- Detaillierte Visualisierung von Abläufen mit Bedingungen, Schleifen und Verzweigungen
- Parallelisierung und Synchronisation ist möglich
- Darstellung von Daten- und Kontrollflüssen
- Nicht auf IT-Projekte beschränkt
  
- Alternative zum Programmablaufplan PAP (Flussdiagramm)

## Zusammenfassende Fragen: Abschnitt 2.4.1 / 2.4.2

- Was ist UML?
- Welche Arten von Diagrammen gibt es?
- Wozu eignen sich Aktivitätsdiagramme?
- Woraus bestehen Aktivitätsdiagramme?
- Wie kann ein Aktivitätsdiagramm verfeinert werden?
- Wie kann eine Schleife modelliert werden?



## Übungsaufgaben

Erstellen Sie ein vollständiges (!) Aktivitätsdiagramm unter folgenden Aspekten:

- Inhaltliche Aspekte
  - Es gibt zwei Akteure: einen Käufer und einen Verkäufer
  - Es gibt folgende ungeordnete Aktivitäten: „Kauf tätigen“, „beraten“, „Wunsch äußern“ und „Geschäft verlassen“
  - Es gibt eine Bedingung: für einen Kauf muss die Beratung gut gewesen sein
- Formale Aspekte
  - Ihr Aktivitätsdiagramm nutzt Aktivitätsbereiche

## Übungsaufgaben

Erstellen Sie ein Aktivitätsdiagramm unter folgenden Aspekten:

- Inhaltliche Aspekte

Beschreiben Sie den ganzen Ablauf eines Campingplatzbesuches: Ankunft, Aufenthalt, Abreise, die wie folgt gekennzeichnet sind:

- Anreise: Anmeldung; Stellplatz aussuchen durch die Besucherin, Stellplatz beziehen
- Aufenthalt: Zelt aufbauen, Seele baumeln lassen, Wäsche waschen, Essen kochen, essen, Müll wegbringen.
- Abreise: Abschluss-Check durch Platzwart, Bezahlen, evtl. für's nächste Mal reservieren.

- Der Platzwart hat mehrere Aufgaben, die ebenfalls dargestellt werden könnten, insbesondere Rasenpflege, Müllentsorgung, Abschluss-Check.

Anmerkung: diese inhaltlichen Aspekte sind nur Anregungen. Falls Sie einen Aufenthalt anders erleben, können Sie das auch gerne anders beschreiben!

### Aspekte zu Ihrem UML-Lernprozess

Überdenken Sie, ob Sie alles richtig und geschickt gemacht haben. Vielleicht ist es sinnvoll, eine geschachtelte Aktivitätenstruktur einzuführen?

Nutzen Sie diese Übung, um die verschiedenen Notationselemente des Aktivitätsdiagramms einzusetzen. Die Aufgabenstellung gibt das her!

Aktionen

Objektknoten

Kanten

Kontrollelemente

Start- und Endknoten

Verzweigungs- und Verbindungsknoten

Parallelisierungs- und Synchronisationsknoten

Aktivitätsbereiche

## UML Diagramme

- 4.1 Überblick 
- 4.2 Aktivitätsdiagramm 
- 4.3 **Use-Case Diagramm**
- 4.4 Business-Objekt-Modell  
(i.F.e. Klassendiagramms)
- 4.5 Zustandsdiagramm
- 4.6 Objekt-Sequenz Diagramm
- 4.7 Zusammenspiel

## Use-Case Diagramm (UCD)

*Fragestellung: Wer soll was mit dem System tun können?*

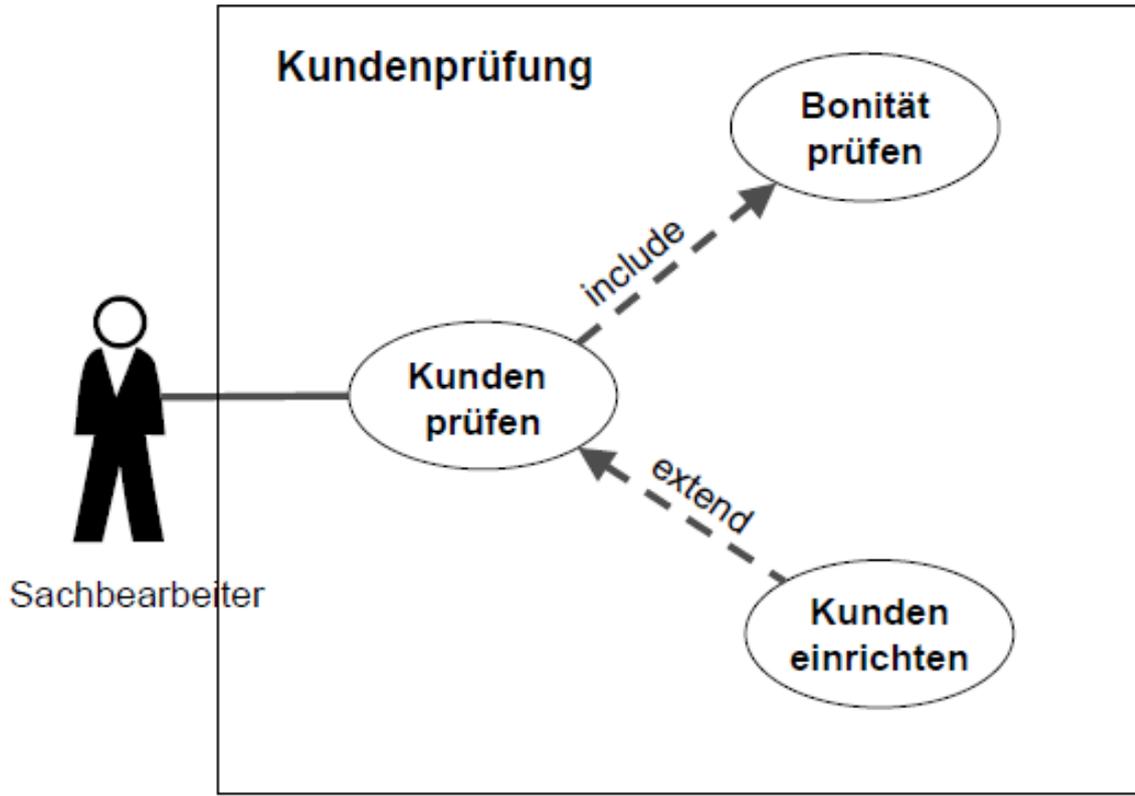
- Beispiele: Datei öffnen, Bild einfügen, Windows herunterfahren
- Ein Use-Case beschreibt
  - in strukturierter Form
  - ein Stück gekapselte\* Funktionalität,
  - die ein Akteur mit einem IT-System durchführen kann.
- Use-Cases beschreiben, was man mit einem IT-System machen kann – nicht wie die Anforderungen im System umgesetzt werden!
- Oft gilt: Use-Cases beschreiben die IT-Unterstützung zur Durchführung der elementaren Geschäftsprozesse.



Synonym: Anwendungsfalldiagramm

\*Gekapselt bedeutet: sinnvoll und in sich geschlossen; ein Aspekt wird voll und ganz beschrieben. „Alles was man wissen muss (gekapselt)“. Es ist egal, wer den Use-Case aufruft.

# Use-Case Diagramm – Beispiel



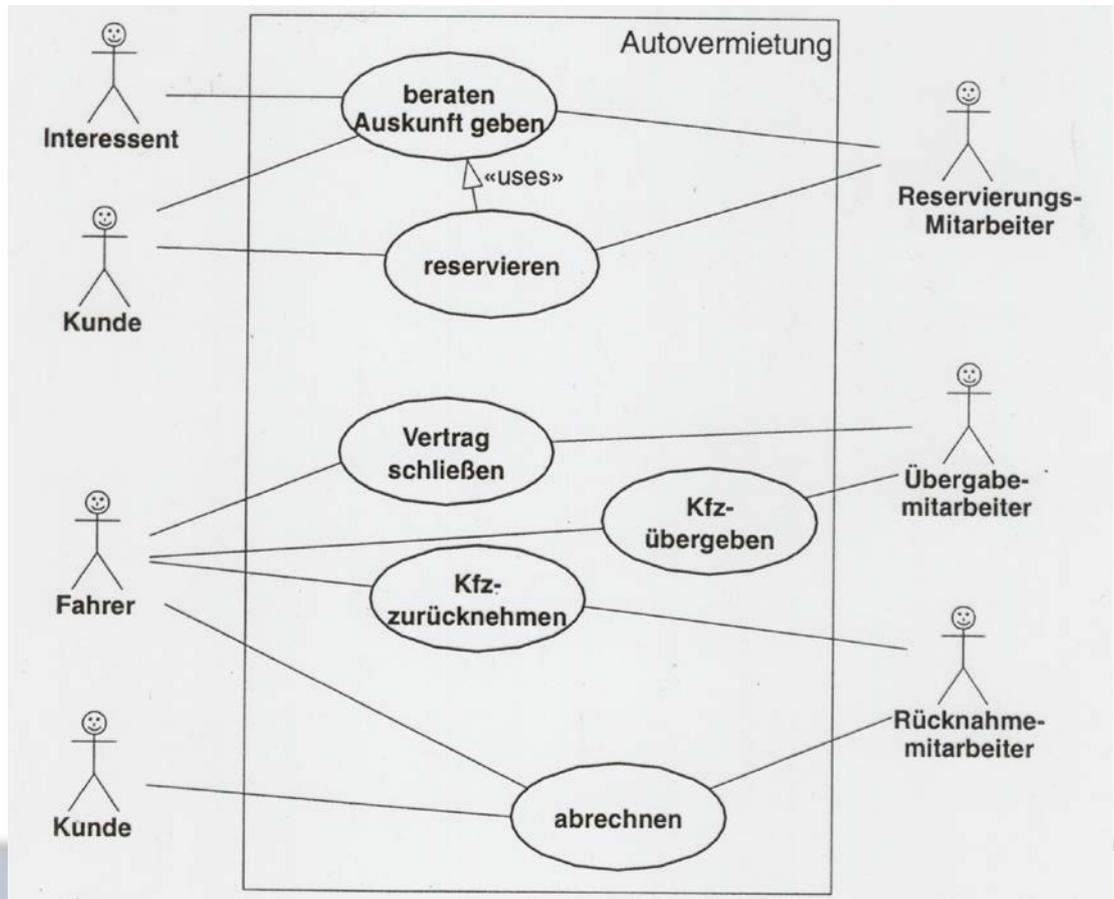
## Use-Case Diagramm – Notation

Element	Name	Bedeutung
	Systemgrenze	<ul style="list-style-type: none"><li>• Alles außerhalb gehört nicht zu System</li></ul>
	Akteur	<ul style="list-style-type: none"><li>• Eine Rolle zur Nutzung des Systems</li><li>• Kann auch wiederum ein System sein</li><li>• Mehrere Akteure können einbezogen werden</li><li>• Benutzt-Beziehung zur Verbindung sagt aus, das UC direkt benutzt werden kann</li></ul>
	Use-Case	<ul style="list-style-type: none"><li>• Use-Case ist ein Stück gekapselte Funktionalität</li><li>• Funktionalität wird immer zusätzlich durch einen Text beschrieben</li></ul>

## Use-Case Diagramm – Notation

Element	Name	Bedeutung
	Beziehung	<ul style="list-style-type: none"><li>• Gestrichelte Pfeile als Verbindung</li><li>• Beziehungen werden an den Pfeilen vermerkt</li> <li>• Include = <b>Muss</b>-Beziehung</li><li>• Der inkludierte Use-Case wird stets aufgerufen</li> <li>• Extend = <b>Kann</b>-Beziehung</li><li>• Der erweiterte Use-Case nutzt unter Umständen den anderen Use-Case</li></ul>

# Use-Case Diagramm – Beispiel (2)



„uses“ als synonym zu:

## Use-Case Beschreibung

Jeder Use-Case erhält eine Beschreibung:



<b>Name</b>	<ul style="list-style-type: none"> <li>• sinnvoll Kombination Substantiv und Verb</li> <li>• Bei großen Softwareprojekten zusätzlich Nummern oder Präfixe</li> </ul>
<b>Ziel</b>	Ziel des Use-Case
<b>Beschreibung</b>	Darstellung was die Funktionalität beinhaltet und wie sie abläuft
<b>Vorbedingung</b>	<ul style="list-style-type: none"> <li>• Keine Selbstverständlichkeiten</li> <li>• Was die Vorbedingung abfängt, braucht in den Alternativen nicht beschrieben werden.</li> <li>• um eine Reihenfolge (zeitliche Abfolge) bei der Ausführung zu erreichen, die i.d.R. nicht erkennbar ist</li> </ul>
<b>Ergebnis</b>	Wie hat sich das IT-System verändert bei erfolgreicher Ausführung
<b>Alternative Abläufe</b>	<ul style="list-style-type: none"> <li>• Beschreibung von Sonderfällen</li> <li>• Sammelstelle für vorläufige Fragen</li> </ul>

\* kein Eintrag ebenfalls möglich

## Use-Case Beschreibung (Beispiel)

<b>Name</b>	Kunde prüfen
<b>Ziel</b>	Kundeninformationen sind geprüft und aktuell
<b>Beschreibung</b>	<ol style="list-style-type: none"><li>1. Der Sachbearbeiter prüft, ob der Kunde vorhanden ist. Falls nicht, wird der Kunde neu angelegt (UC Kunden einrichten)</li><li>2. Die Kundeninformationen werden ggf. aktualisiert</li><li>3. Die Bonität wird geprüft (UC Bonität prüfen)</li></ol>
<b>Vorbedingung</b>	Kundendaten liegen vor
<b>Ergebnis</b>	<ul style="list-style-type: none"><li>• Valide Kundendaten im System gespeichert</li><li>• Kunde freigeschaltet</li></ul>
<b>Alternative Abläufe</b>	Anfrage übersteigt Kreditlimit

## Use-Case Diagramm – Vorgehen

- 1. Use-Cases identifizieren
- 2. Use-Cases systematisch beschreiben
- 3. Use-Cases zu Diagrammen zusammenstellen
  - Beziehungen einzeichnen
  - Akteure identifizieren und eintragen

### Wichtig:

- Schritte werden i.d.R. mehrmals durchlaufen!
- Use-Case i.d.R. Ausschnitt der Gesamtfunktionalität
- Use-Case kann in mehreren Diagrammen vorkommen

Ad1)

Wie groß sollte ein UC sein?

Alles was vor geistigem Auge auf einer Bildschirmmaske untergebracht wird

Ad2)

Wie vorgestellt: strukturiert und step-by-step

Ad3)

Keine Restriktionen in der UML-Spezifikation

Möglichkeiten

Akteurorientiert: alle Use-Cases die einem Akteur zur Verfügung stehen  Fachexperte kann entscheiden ob Funktionen ausreichen

Prozessorientiert: alle Use-Cases die zur Erledigung einer Aufgabe benötigt werden

## Use-Case Diagramm

### *Einsatz und Zusammenfassung*

- Übersicht der Systemfunktionalität und die Einbettung eines Systems in das Geschäft des Auftraggebers
- Anforderungsermittlung aus Sicht der Akteure
- Stellt Anwendungsfälle und Akteure mit ihren jeweiligen Abhängigkeiten und Beziehungen dar
- Fähigkeiten des Systems werden in kleinen Häppchen dargelegt (modularer Aufbau) und stellt in der Regel nur einen Ausschnitt der Gesamtfunktionalität dar

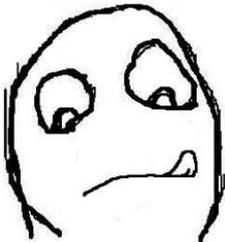
## Use-Case Diagramm

### *Einsatz und Zusammenfassung (2)*

- Wird von Auftraggeber und Auftragnehmer verwendet und dient als Kommunikationsgrundlage
- Use Cases sind das Bindeglied zwischen Geschäft und IT
- Use Cases haben viele Querverbindungen zu anderen Diagrammen
- Use Cases begleiten den Entwicklungsprozess bis zur Realisierung

## Zusammenfassende Fragen: Abschnitt 2.4.3

- Wozu dient ein Use-Case Diagramm?
- Welche Bestandteile hat ein UCD?
- Was ist der unterschied zwischen einem Use-Case und einem UCD?
- Zeichnen Sie ein Beispiel eines UCD.
- Wie erfolgt die Beschreibung eines Use-Case?



## Übungsaufgaben

Diese Übung kommt auf das Campingplatzbeispiel zurück.

1.) Erstellen Sie nun ein Use-Case Diagramm unter folgenden Aspekten:

- Das Diagramm enthält alle notwendigen Bestandteile.
- Das IT-System soll zu Beginn nur den Anreiseprozess unterstützen. Dazu wird die Aktion „Anmeldung“ als zentraler Prozess betrachtet, der abgebildet werden soll.
- Das Diagramm benötigt weiterhin die Use-Cases „Kunden anlegen“, „Platz buchen“, „Platz suchen“.

Dabei gilt:

- Der Use-Case „Anmeldung“ ist direkt mit dem Akteur verbunden.
- Wenn der Kunde bereits im System erfasst ist, muss er nicht neu angelegt werden.
- In jedem Fall kann der Platz nur gebucht werden, wenn die Suche einen freien Platz ergab.

2.) Fertigen Sie eine Beschreibung aller Use-Cases an!

## UML Diagramme

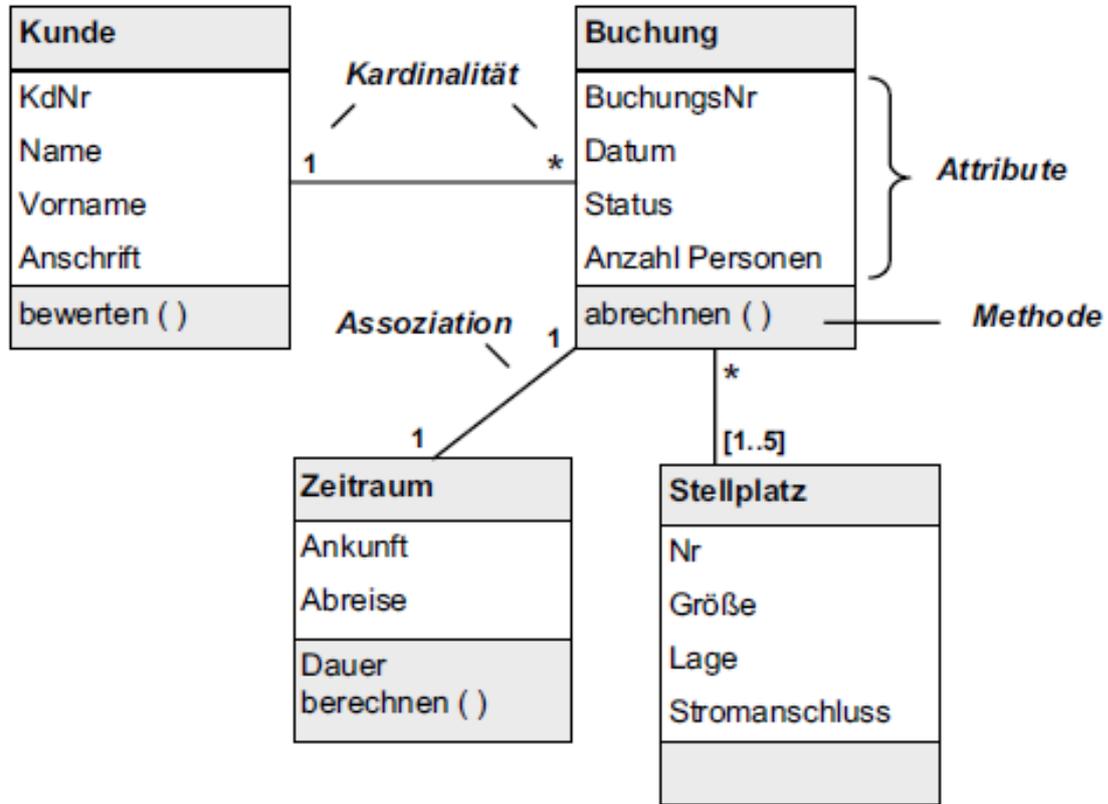
- 4.1 Überblick 
- 4.2 Aktivitätsdiagramm 
- 4.3 Use-Case Diagramm 
- 4.4 **Business-Objekt-Modell  
(i.F.e. Klassendiagramms)**
- 4.5 Zustandsdiagramm
- 4.6 Objekt-Sequenz Diagramm
- 4.7 Zusammenspiel

## Business Object Model (BOM)

*Fragestellung: Wie ist die Beziehung zwischen Objekten?*

- Objekte mit Eigenschaften und Fähigkeiten sowie ihre Beziehungen untereinander werden beim BOM in einem Diagramm dargestellt.
- Objekte sind Begriffe die sich im IT-System widerspiegeln. Meistens werden Dinge, Rollen oder Ereignisse mit Hilfe von Objekten modelliert. Beispiele für typische Objekte sind: Kunde, Rechnung, Auftrag, Storno und Artikel.
- BOM ist oft erste Grundlage für die Arbeit der Programmierer.  
→ Die Objekte finden sich in der Programmierung wieder.
- Es lassen sich Datenbankstrukturen aus dem BOM ableiten.

# Business Object Model – Beispiel



Quelle: Brandt-Pook; Softwareentwicklung kompakt und verständlich; S. 84

## Business Object Model – Notation

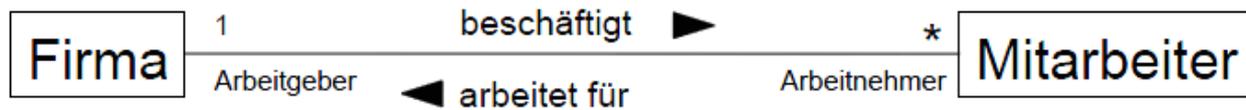
Element	Bedeutung
Objekt	Beschreibt einen Begriff (im Singular). Hier sind die Objekte: Kunde, Buchung, Zeitraum, Stellplatz.
Attribut	Eigenschaften des Begriffs. Beschreiben das Wissen eines Objektes. Attribute müssen vorhanden sein.
Methode	Fähigkeiten des Objektes. Beschreiben das Können eines Objektes. Nicht jedes Objekt benötigt eine Fähigkeit.
Assoziation	Verbindung zwischen zwei Objekten. Es gilt: keine unnötigen Assoziationen einzeichnen.
Kardinalitäten	Geben an, in welcher Häufigkeit die Objekte an der Assoziation beteiligt sind. n → genau n n .. m → eine Zahl zwischen n und m n, m → entweder n oder m * → beliebig viele

Konkret in dem Beispiel heißt das:

- Jede Buchung bezieht sich auf genau einen Kunden. Ein Kunde kann beliebig viele Buchungen tätigen
- Zu jeder Buchung gehört genau ein Zeitraum. Jeder Zeitraum gehört zu genau einer Buchung
- Zu einer Buchung können eins bis fünf Stellplätze gehören (man kann also max. fünf Stellplätze gleichzeitig buchen). Jeder Stellplatz kann in beliebig vielen Buchungen vorkommen.

## Business Object Model – Notation (2)

- Assoziationen können Namen haben, die beteiligten Objekte Rollenbezeichner



## Business Object Model – Notation (3)

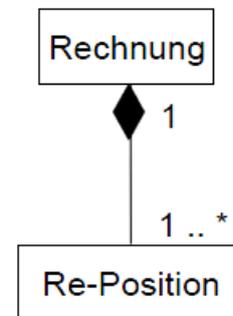
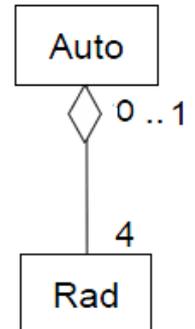
Vereinigung

Zusammenstellung

Aggregation und Komposition

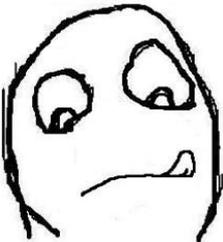
Zwei Sonderformen bei den Assoziationen:

- Aggregation: „hat Teil“:  
Darstellung durch offene Raute  
mit der Multiplizität 0 oder 1
- Komposition:  
„hat existenzabhängiges Teil“:  
Darstellung durch geschlossene Raute  
und der Multiplizität 1



## Zusammenfassende Fragen: Abschnitt 2.4.4

- Welche Arten von Kardinalitäten gibt es?
- Welche Arten von Assoziationen gibt es und wie können sie eingesetzt werden?



## Übungsaufgabe

Diese Übung kommt auf das Campingplatzbeispiel zurück.

1.) Erstellen Sie nun ein BOM unter folgenden Aspekten:

- Das Diagramm enthält das zentrale Objekt Anmeldung mit den Attributen (Nr., Datum, AnzahlPersonen)
- Weitere Objekte sind Kunde (Nr., Name, Anschrift), Zeitraum (Ankunftsdatum, Abreisedatum), Abrechnung (Nr.), Stellplatz (Nr., Qm, Art, Größe).
- Es handelt sich um jeweils 1-1 Beziehungen, außer dass natürlich Kunden die Möglichkeit haben, mehrere Stellplätze gleichzeitig zu mieten.
- Zeichnen Sie so wenig Assoziationen wie nötig.
- Jedes Objekt hat eine Methode. Was könnten sinnvolle Methoden sein?

## UML Diagramme

- 4.1 Überblick 
- 4.2 Aktivitätsdiagramm 
- 4.3 Use-Case Diagramm 
- 4.4 Business-Objekt-Modell  
(i.F.e. Klassendiagramms) 
- 4.5 **Zustandsdiagramm**
- 4.6 Objekt-Sequenz Diagramm
- 4.7 Zusammenspiel

## Zustandsdiagramm

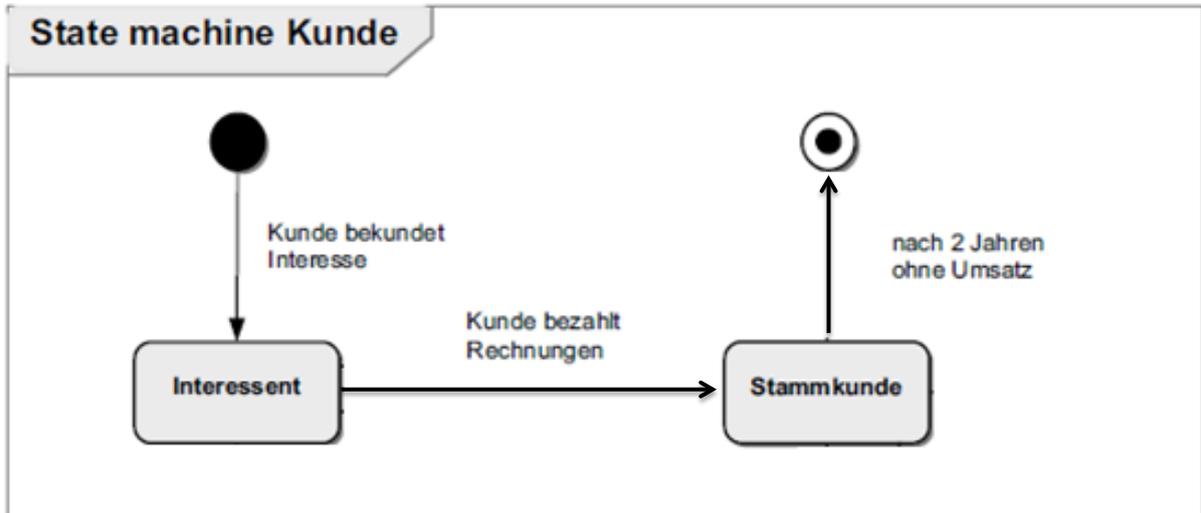
*Fragestellung: Welche Zustände haben die Objekte aus dem BOM?*

- Zeigt in dem Diagramm die möglichen Zustände eines Objektes auf und dokumentiert wie diese gewechselt werden
- Es zeigt anhand von Zuständen den Lebenslauf eines Objektes
- Zustandsübergänge werden von Ereignissen ausgelöst
- Beispiel:  
Ein Objekt „Rechnung“ kann den Status „Bezahlt“ oder „Offen“ erhalten

<b>Kunde</b>
KdNr
Name
Vorname
Anschrift
bewerten ( )

Andere Bezeichnungen sind Zustandsautomat bzw. state maschine (state=Zustand).

# Zustandsdiagramm – Beispiel

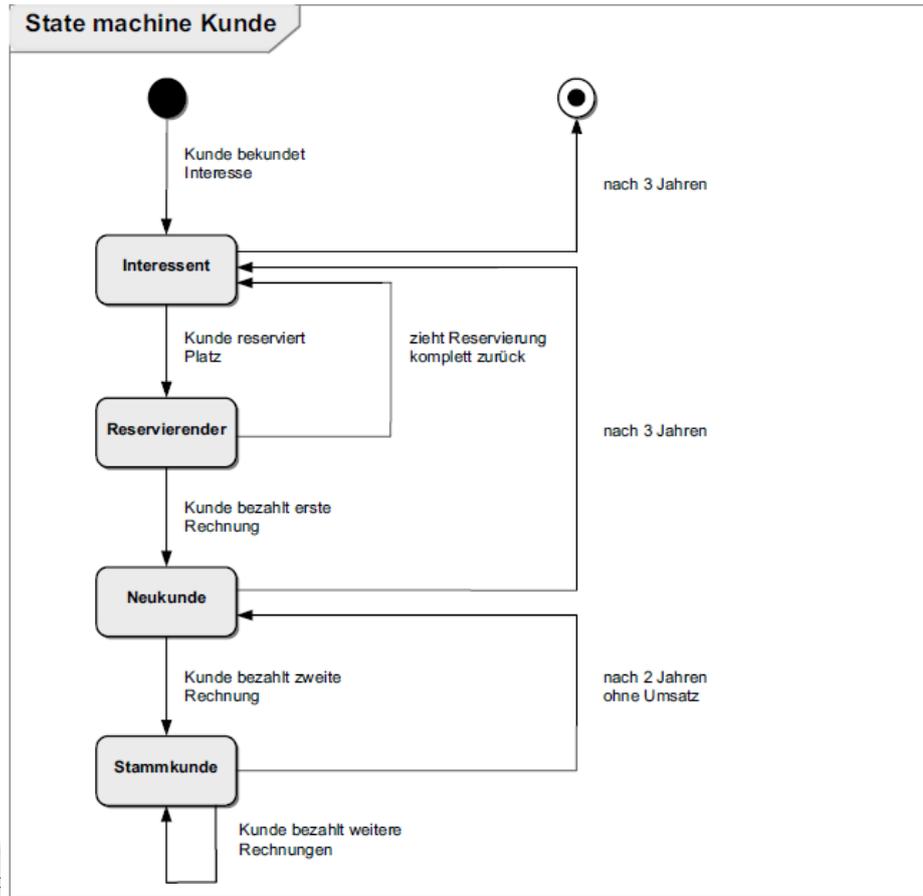


Quelle: Brandt-Pook; Softwareentwicklung kompakt und verständlich; S. 95

## Zustandsdiagramm – Notation

Element	Name	Bedeutung
	Zustandsdiagramm	<ul style="list-style-type: none"><li>• beinhaltet alle Zustände eines Objektes (z.B. Kunde)</li><li>• Begriff State maschine bezeichnet die Art des Diagramms</li></ul>
	Zustand	<ul style="list-style-type: none"><li>• Zustände, die das Objekt annehmen kann</li><li>• Beinhaltet verständliche Bezeichnung (z.B. Neukunde)</li></ul>
	Ereignis	<ul style="list-style-type: none"><li>• Verbindet die Zustände</li><li>• Pfeilspitze gibt Folgezustand an</li><li>• Bezeichnung ist erforderlich</li></ul>
	Startzustand Endzustand	<ul style="list-style-type: none"><li>• Anfang und Ende</li></ul>

# Zustandsdiagramm – Beispiel (2)



## UML Diagramme

- 4.1 Überblick 
- 4.2 Aktivitätsdiagramm 
- 4.3 Use-Case Diagramm 
- 4.4 Business-Objekt-Modell  
(i.F.e. Klassendiagramms) 
- 4.5 Zustandsdiagramm 
- 4.6 **Objekt-Sequenz Diagramm**
- 4.7 Zusammenspiel

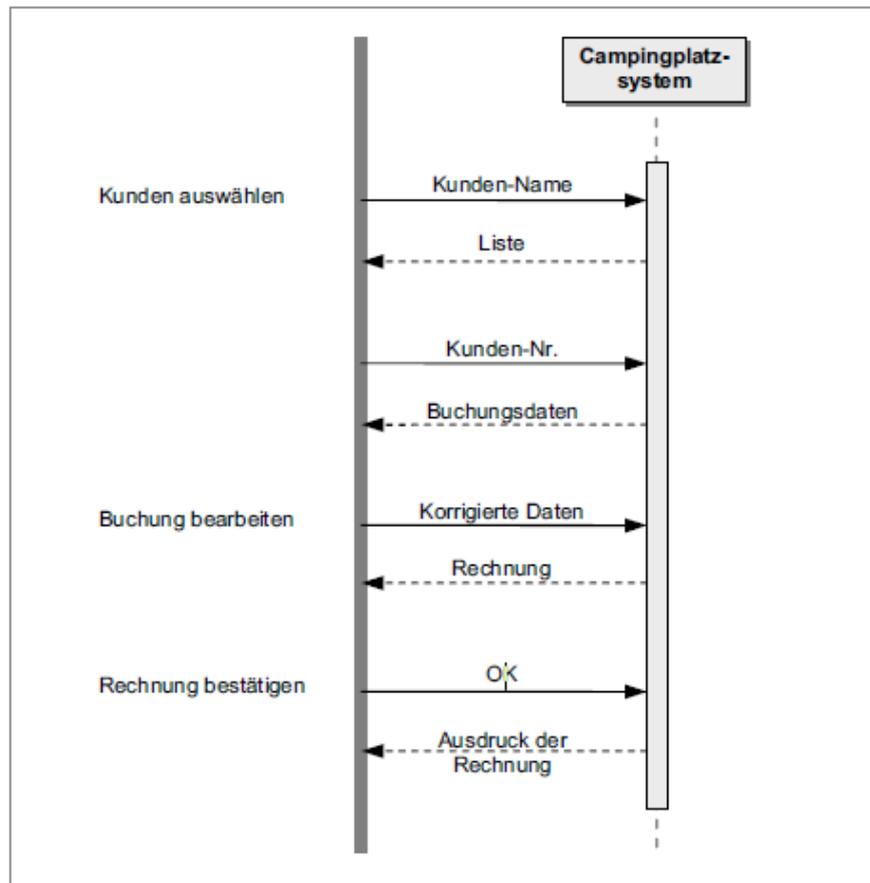
## Objekt Sequenz Diagramm (OSD)

*Fragestellung: Wie arbeiten die Objekte aus dem BOM zusammen?*

- Ein Objekt Sequenz-Diagramm beschreibt das Zusammenwirken von Objekten bei der Erledigung einer Aufgabe
- zeigt Abläufe entlang der Zeitachse
- zeigt die Wechselbeziehung von Objekten bei der Erledigung einer Aufgabe
- stellt dar, wie die Objekte miteinander sprechen und welche Informationen sie dabei austauschen
- wird häufig im Design und in der Realisierung eines Systems benutzt
  
- eignet sich dazu, sehr ins Detail zu gehen  
→ komplizierte OSD nur für erfahrene UML Anwender\*

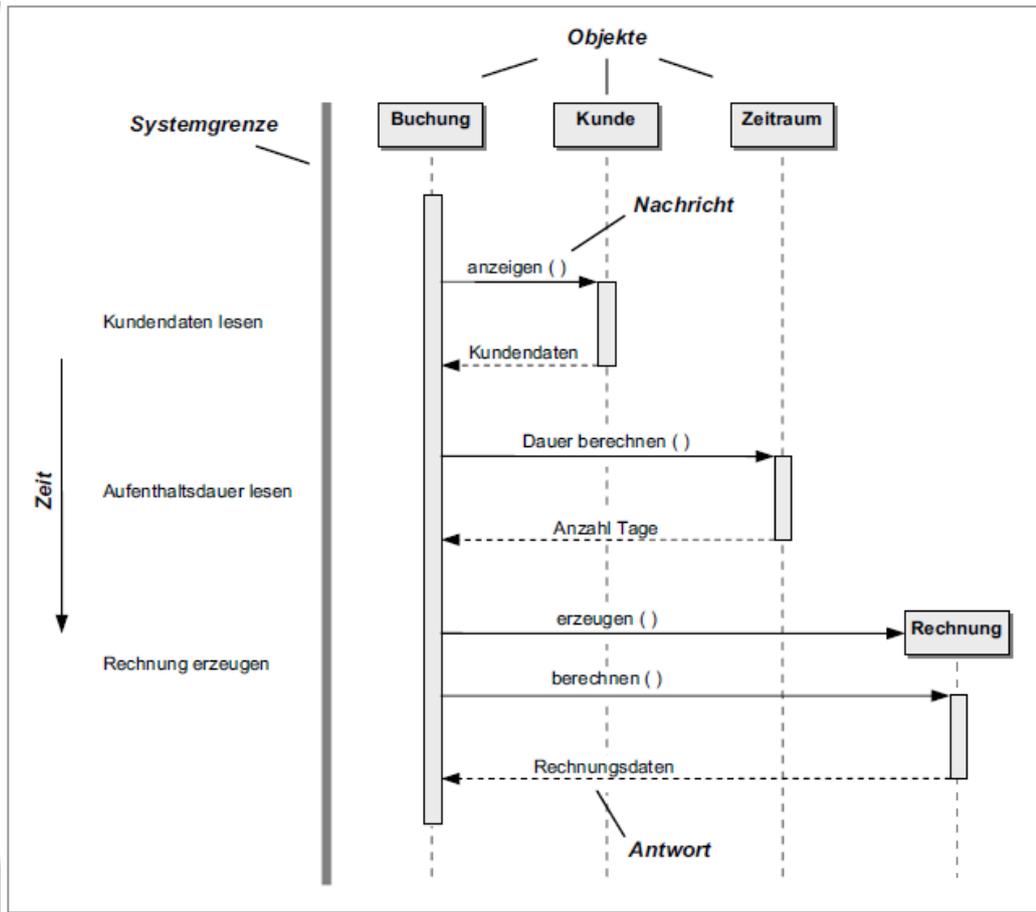
\*Deswegen in der Notation und in Aufgaben nicht weiter vertieft

## Objekt Sequenz Diagramm – Beispiel



In dieser einfachsten Form wird gezeigt, wie ein Benutzer mit dem System arbeitet.

# Objekt Sequenz Diagramm – Beispiel (2)

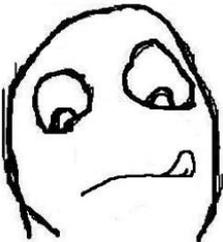


Quelle: Brandt-Pook; Softwareentwicklung kompakt und verständlich; S. 98

Im Regelfall sind dies die Objekte aus dem BOM, aber das ist nicht festgelegt. Das OSD eignet sich ebenfalls dazu, den Ablauf eines Use-Case oder einen Prozess über mehrere Use-Cases hinweg darzustellen.

## Zusammenfassende Fragen: Abschnitt 2.4.5 + 2.4.6

- Wozu dienen das Zustandsdiagramm und das OSD?
- Wie können Zustände dargestellt werden?
- Wie werden Zustände verbunden?
- Zeichnen und erläutern Sie in knapper Form ein Beispiel eines Zustands- und OS-Diagramms.



## UML Diagramme

- 4.1 Überblick 
- 4.2 Aktivitätsdiagramm 
- 4.3 Use-Case Diagramm 
- 4.4 Business-Objekt-Modell  
(i.F.e. Klassendiagramms) 
- 4.5 Zustandsdiagramm 
- 4.6 Objekt-Sequenz Diagramm 
- 4.7 **Zusammenspiel**

## Überblick

- Alle Methoden sind für ihr Einsatzgebiet sinnvoll.
- Methoden greifen in einander, doch gibt es keinen Automatismus.

Häufig aber werden ...:

- interessante Prozesse werden in einem Aktivitätsdiagramm dargestellt,
- den Aktionen des Aktivitätsdiagramms Use-Cases zugeordnet und diese zu UCD's zusammengefasst,
- die Beschreibungen der Use-Cases zur BOM Erstellung herangezogen
- Use-Cases mit dem OSD in konkrete Abläufe umgesetzt,
- die Objekte des BOM mit Zustandsdiagrammen mit ihrem Status dargestellt.

# Überblick (2)



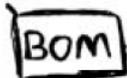
Aktivitätsdiagramm

- wie sieht der Geschäftsablauf im Detail aus?
- einfach, aber ausdrucksstark!



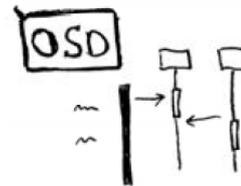
UCD

- „wer“ kann „was“ mit dem System?
- zeigt Funktionalität
- Details in UC-Beschreibung



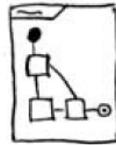
BOM

- wie ist die Beziehung zwischen Objekten?
- Dinge, Rollen, Ereignisse = Objekte (z.B. Kunde)



OSD

- wie arbeiten die Objekte aus dem BOM zusammen?
- Spezialwerkzeug
- viele Details



Zustandsdiagramm

- welche Zustände haben die Objekte aus dem BOM?

Quelle: Brandt-Pook; Softwareentwicklung kompakt und verständlich; S. 105

## Zusammenfassende Fragen: Abschnitt 2.4

- Was ist UML?
- Welche Diagrammarten kennt UML?
- Wo liegt der Fokus der behandelten Diagramme?
- Wie können die Diagramme ineinander greifen?

