

# UML (UNIFIED MODELING LANGUAGE)

Ziel: Einheitliche Darstellung einer Vielzahl von Elementen von Softwaresystemen mittels einer einheitlichen Notation.

## Übersicht

Strukturdiagramme				
Klassendiagramm	Objektdiagramm	Paketdiagramm Package-Diagramm	Implementierungsdiagramme	
			Komponentendiagramm	Verteilungsdiagramm Deployment-Diagramm
Zusammenhang zwischen Elementen	Beziehungsausprägung	Organisation von Elementen	Organisation von Komponenten	Hardwareaufbau, Ausführung des Systems

## Verhaltensdiagramme

Anwendungsfall- Use-Case - Diagramm	Aktivitätsdiagramm	Zustandsautomat Zustandsdiagramm	Interaktionsdiagramm	
			Sequenzdiagramm	Kommunikationsdiagramm Kollaborationsdiagramm
Kommunikation mit Anwender	parallele Prozesse, dynamischer Ablauf	Interne Zustände	Zeitliche Aufrufstruktur, Interner Datenaustausch	Statische Sicht auf dynamische Interaktionen

## Einteilung

<b>Statische Diagramme</b>	Visualisierung der Struktur der Klassen, ohne temporäre Abfolgen
<b>Dynamische Diagramme</b>	Relation der Klassen untereinander, und Verhalten des Systems

## Kommunikationslinien

	<b>Beziehung</b>	A kennt B B kennt A
	<b>Gerichtete Beziehung</b>	A kennt B
	<b>Generalisierung</b>	A kann alles was B kann A kann noch mehr
	<b>Abhängigkeit</b> (schwächer als Beziehung) Keine permanente statische Beziehung	A ist abhängig von B
	<b>Use-Case-Beziehung „&lt;&lt;extend&gt;&gt;“</b> Beziehung zwischen zwei Anwendungsfällen	B kann von A erweitert werden
	<b>Use-Case-Beziehung „&lt;&lt;include&gt;&gt;“</b> Beziehung zwischen zwei Anwendungsfällen	B kann von A benutzt werden
	<b>Use-Case-Beziehung „&lt;&lt;realize&gt;&gt;“</b> Beziehung zwischen zwei Anwendungsfällen	

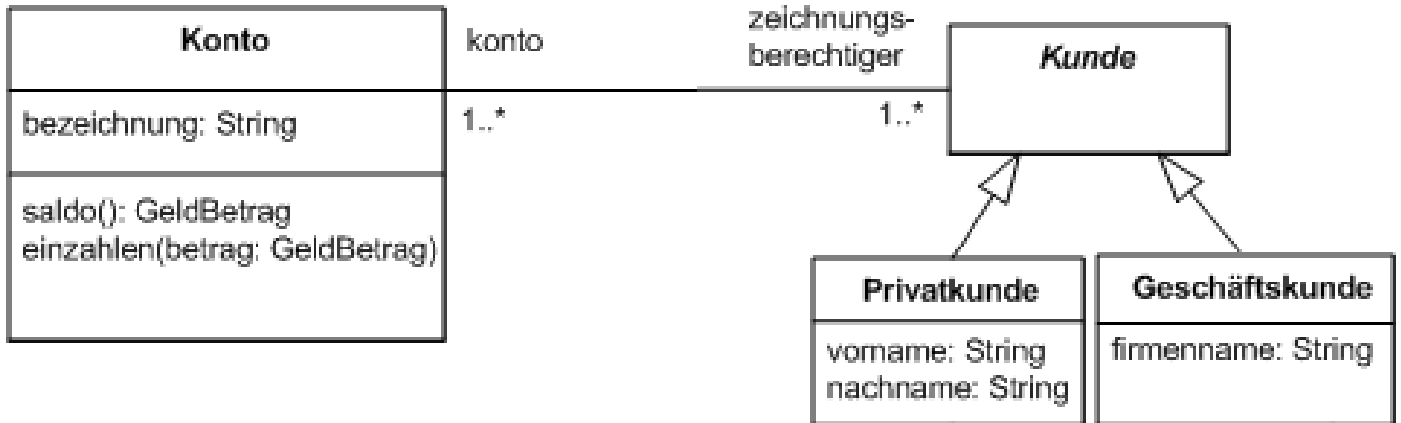
## Allgemeine UML-Objekte

	<b>Aktor</b>	
--	--------------	--

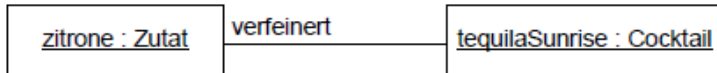
### Beispiele

#### Strukturdiagramme

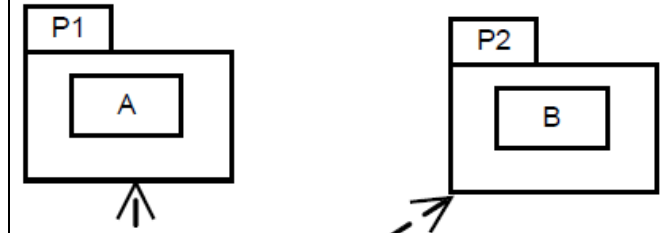
##### Klassendiagramm



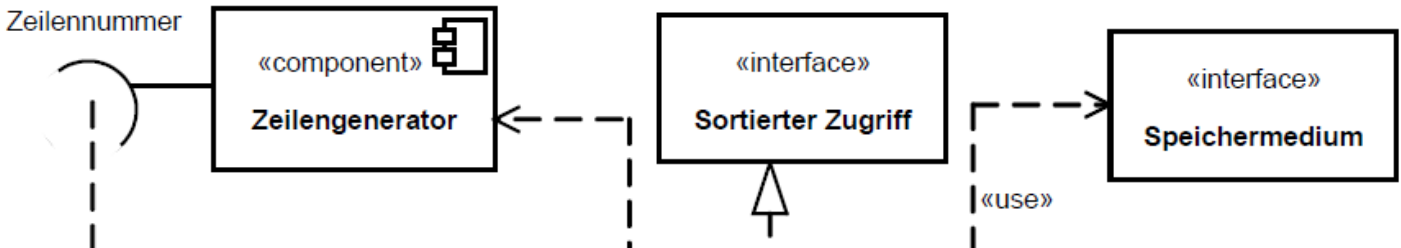
##### Objektdiagramm



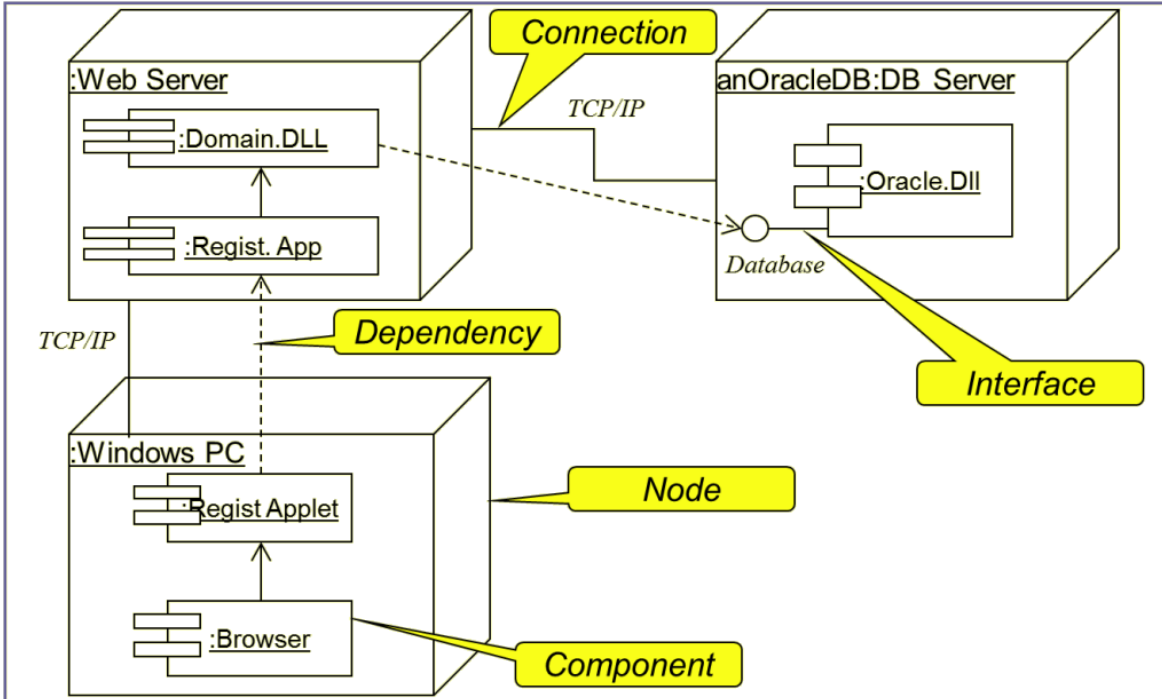
##### Paketdiagramm / Package-Diagramm



##### Komponentendiagramm

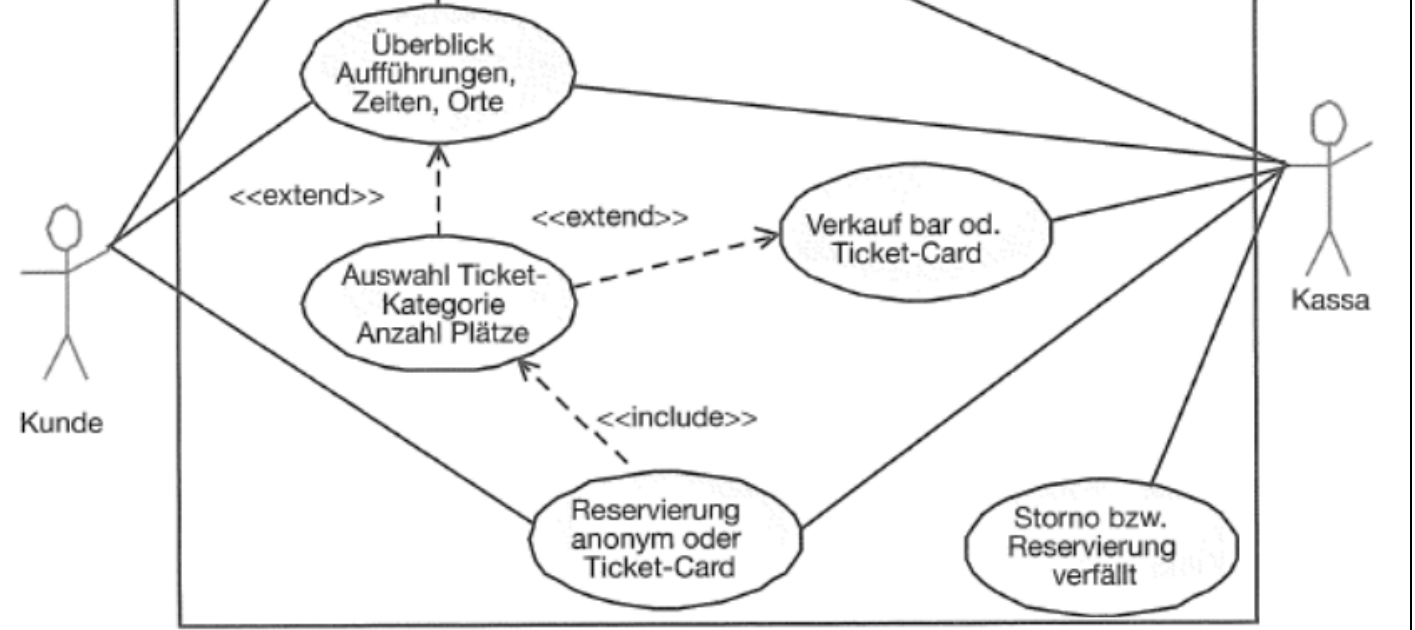


##### Verteilungsdiagramm / Deployment-Diagramm

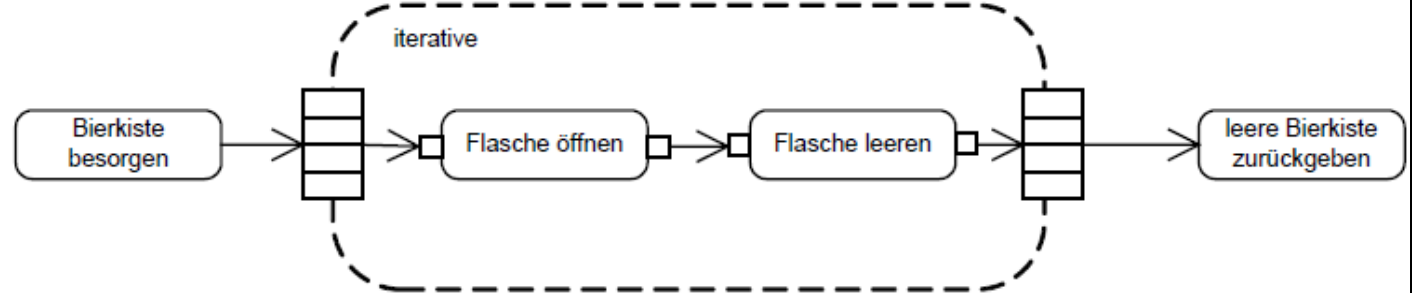


Verhaltensdiagramme

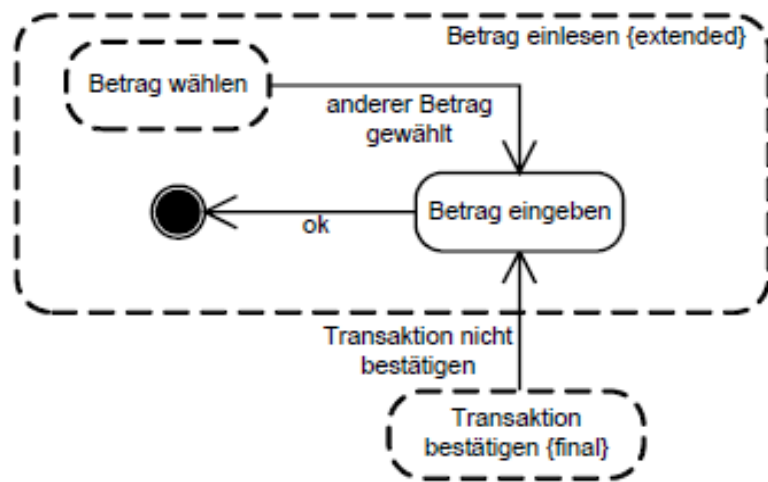
Anwendungsfall- / Use-Case - Diagramm



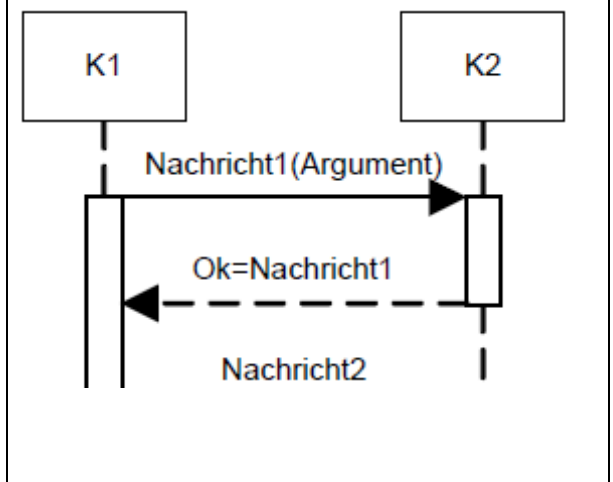
Aktivitätsdiagramm



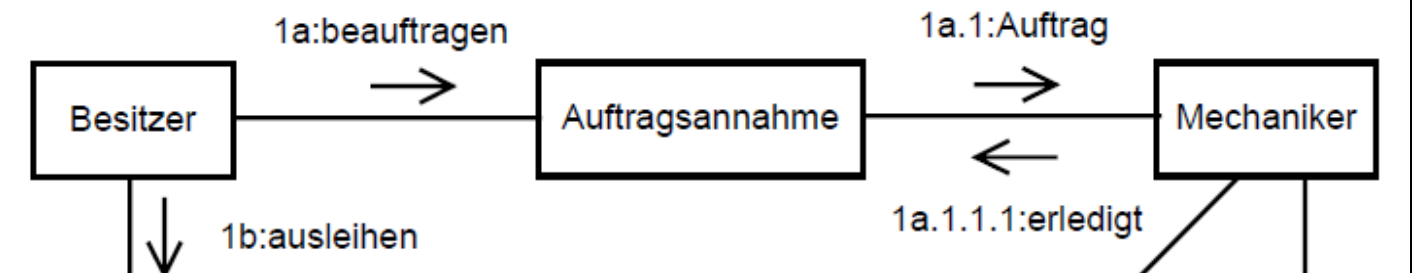
Zustandsautomat / Zustandsdiagramm



Sequenzdiagramm



Kommunikationsdiagramm / Kollaborationsdiagramm



**Theorie**

**Strukturdiagramme**

**Klassendiagramm**

Nur Klassenname	inkl. Attribute	inkl. Attribute, Methoden	Abstrakte Klasse
Aufführungssaal	Aufführungssaal Ort Bezeichnung Art Anzahl_Plätze Kosten_pro_Tag	Aufführungssaal Ort Bezeichnung Art Anzahl_Plätze Kosten_pro_Tag definieren suchen	Ticket {abstract}

persistent = dauerhaft gespeicherte (Attribute)

	<b>Assoziationen</b> (Verb + Kardinalitätszahl) Kardinalität = Anzahl Instanzen, einer Beziehung Dreieck = Leserichtung  Stern * = natürliche Zahl Zahl = einzelner ganzzahliger positiver Wert Wertebereich = 1..3 Mehrere Werte = 1,3..6,9	Ein Künstler wirkt bei 0 bis n Mitwirkungen  Eine Mitwirkung entspricht genau einem Künstler
	<b>Ganzes-Teil-Assoziation - Typ Aggregation</b> Teil existieren auch nach der Zerstörung des Ganzen	Raum existert auch nach Konkurs des Kinos
	<b>Ganzes-Teil-Assoziation - Typ Komposition</b> Teil hören ebenfalls auf zu existieren, nach Zerstörung des Ganzen	Reihen werden mit dem Löschen der Sitzplätze auch aufgelöst

**Objektdiagramm**

-

**Paketdiagramm / Package-Diagramm**

	<b>Pakete</b> Logische Gruppierung + minimierter Arbeitsaufwand + Konsistenz bei der Entwicklung
--	--

**Komponentendiagramm**

-

**Verteilungsdiagramm / Deployment-Diagramm**

<b>Darstellung</b>	von der Konfiguration der Laufzeitelementen Software-Komponenten und Prozessen		
<b>Bestandteile</b>	<b>Knoten</b>	Quader	die eine Verarbeitungs- oder Hardwareeinheit darstellen
	<b>Verbindungen</b>	Linien	die physikalische Kommunikationspfade zwischen den Knoten darstellen
	<b>Softwarekomponenten Objekte</b>	Rechteck	innerhalb der Knoten
	<b>Programme</b>	Artefakte	auch ausserhalb möglich
	<b>Abhängigkeitslinien</b>		sind zwischen Objekten, Komponenten oder Schnittstellen gezeichnet
<b>Deployment-abhängigkeit</b>	→Softwarekomponente XY wird auf diesem Knoten installiert		






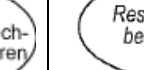
**Verhaltensdiagramme**

**Anwendungsfall- / Use-Case - Diagramm**

Alle vorhandenen Aktoren und Anwendungsfälle und deren Kommunikation untereinander werden dargestellt.

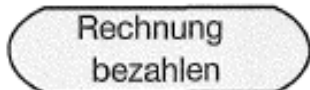
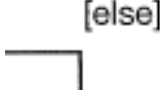

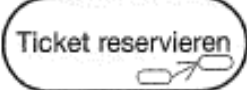
**Anwendungsfälle**

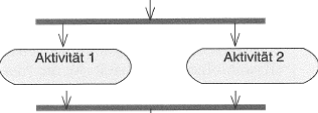



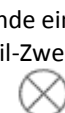
Beschreibt anhand eines zusammenhängenden Arbeitsablauf die Interaktion mit einem System

Geschäftsanwendungsfall	System-anwendungsfall	Sekundärer Anwendungsfall	Abstrakter Anwendungsfall	Anforderung, Feature & Co
geschäftlicher Ablauf	für aussen stehende Akteure	unvollständiger Teilablauf	Verallgemeinerung einer Menge	Einzelne Anforderung Sammlung von Anf. =Feature
 Geschäfts-anw.	 Kern-Geschäfts-a.	 standard	 «secondary» «secondary»	 «requirement» «feature»
			 kursiv	

Stereotyp = Erweiterung eines vorhandenen Modellelementes; für neue Symbole

**Aktivitätsdiagramm**


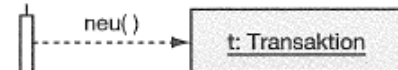












Aktivität	Bedingungen	Entscheidungszustände	Sub-Aktivität
abgerundete Vierecke ununterbrechbar, unverzögliche Durchführung	in Eckigen Klammern Erfüllen = Übergang	Drachenvierecke	(Mehrere Aktivitäten)
			

Parallele Prozesse	Einteilung „Swimlanes“	Startzustand	Endzustand	Ablaufender Zustand
				

**Zustandsautomat / Zustandsdiagramm**





Zustand	Aktionen	Zustand mit Aktionen	Sammelzustand
	Typen: Entry: beim Wechsel in diesen Zustand Exit: beim Verlassen dieses Zustandes Do: während diesem Zustand		

**Sequenzdiagramm**

<b>gestrichelte Linie nach unten</b> Lebensdauer eines Objekts		<b>Objekt-erzeugung</b>							
<b>schmales Rechteck</b> Objekt mit Kontrolle über Programmfluss		<b>Objekt-zerstörung</b>							
<b>Aufruf innerhalb des eigenen Objekts</b>		<b>Pfeile</b>	<table border="1"> <tr> <td></td> <td>Abgebende Nachricht</td> </tr> <tr> <td></td> <td>Rückkehrende Nachricht</td> </tr> <tr> <td></td> <td>Nachricht ohne erwartete Rückkehr</td> </tr> </table>		Abgebende Nachricht		Rückkehrende Nachricht		Nachricht ohne erwartete Rückkehr
	Abgebende Nachricht								
	Rückkehrende Nachricht								
	Nachricht ohne erwartete Rückkehr								

**Kommunikationsdiagramm / Kollaborationsdiagramm**

Stellt Abhängigkeit von Objekten anhand der zwischen ihnen auszutauschenden Nachrichten dar.

	<b>Objekte</b> (Objektname + „:“ + Klassenname)
	<b>Kollaboration</b> (Verbindung zwischen Objekten) zum Austausch von Objekten
	<b>Nachricht</b> , Programmfluss kehrt nach beenden zurück
	<b>Nachricht</b> , ohne eine Rückkehr zu erwarten