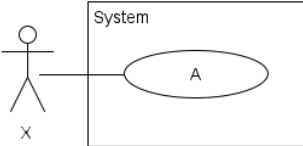
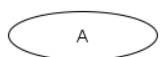

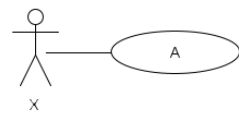
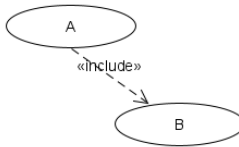
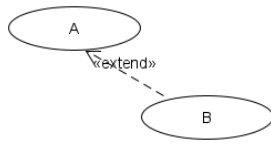
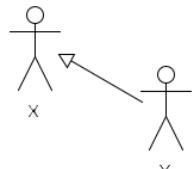


## Cheat Sheet UML

### Die wichtigsten Notationselemente der UML 2

#### Anwendungsfalldiagramm (*Use Case Diagram*)

Name	Notation	Erklärung
System ( <i>system</i> )		Grenzen zwischen dem System und den Akteuren des betrachteten Systems (SuD)
Anwendungsfall ( <i>use case</i> )		Beschreibt ein Verhalten des Systems, das einem Akteur zur Verfügung gestellt wird; ein Use Case beschreibt viele verschiedene Szenarien (Standardablauf, Erweiterungen); ein Szenario ist ein bestimmter Ablauf im Use Case (Instanz)
Akteur ( <i>actor</i> )		Rolle der Systembenutzer, die mit dem Use Case interagieren; für externe Systeme wird die Rechtecknotation verwendet
Assoziation ( <i>association</i> )		Kommunikationsbeziehung zwischen Use Cases und Akteuren; bei mehreren Akteuren kann der initiiierende Akteur mit gerichtetem Pfeil markiert werden
Include-Beziehung ( <i>include relationship</i> )		A include B: notwendiges Verwenden von Use Case B durch Use Case A; in der Use-Case-Spezifikation A wird in einem Szenario der Use Case A aufgerufen (referenziert)
Extend-Beziehung ( <i>extend relationship</i> )		B extend A: optionales Verwenden von Use Case B durch Use Case A; in der Use-Case-Spezifikation A wird in einem Szenario und bestimmten Bedingungen der Use Case B aufgerufen (referenziert)
Aktor Generalisierung ( <i>actor generalization</i> )		Y erbt von X; Y kommuniziert mit allen Use Cases, mit denen X kommuniziert

## Use-Case-Spezifikation (*Use Case Specification*)

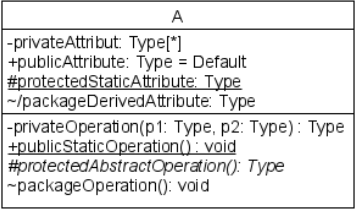
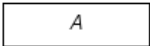
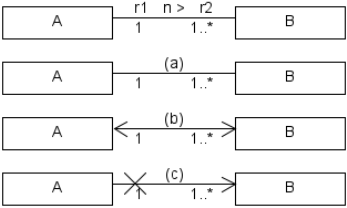
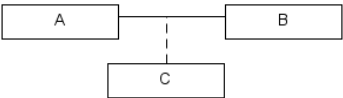
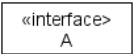
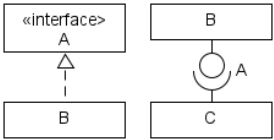
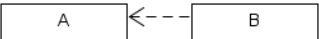
<b>Use-Case-Name</b> ( <i>Use Case Name</i> )	Titel des Use Cases und dann das Verb: Verkauf bearbeiten (im englischen umgekehrt: Process Sale)
<b>Umfang</b> ( <i>Scope</i> )	(fakultativer Abschnitt) Das System unter Design (SuD), das zu entwerfende System
<b>Ebene</b> ( <i>Level</i> )	(fakultativer Abschnitt) «Anwenderziel» oder «Subfunktion»
<b>Primärakteur</b> ( <i>Primary Actor</i> )	Nutzt die Use Cases des Systems
<b>Stakeholder und Interessen</b> ( <i>Stakeholders and Interests</i> )	(fakultativer Abschnitt) Für wen ist der Use Case interessant und welches Interesse hat er daran?
<b>Vorbedingung(en)</b> ( <i>Preconditions</i> )	Was muss am Anfang gewährleistet und für den Leser mitteilenswert sein?
<b>Erfolgsgarantie, Nachbedingungen(en)</b> ( <i>Success Guarantee</i> )	Was muss nach der erfolgreichen Fertigstellung gewährleistet und für den Leser mitteilenswert sein?
<b>Standardablauf</b> ( <i>Main Success Scenario</i> )	Das Szenario für einen typischen, unbedingten erfolgreichen Durchlauf durch den Use Case
<b>Erweiterungen</b> ( <i>Extensions</i> )	Alternative Szenarios für Erfolg und Misserfolg
<b>Spezielle Anforderungen</b> ( <i>Special Requirements</i> )	(fakultativer Abschnitt) Zum System gehörige, nichtfunktionale Anforderungen (Qualitätsanforderungen u.a.)
<b>Liste der Technik- und Datenvariation</b> ( <i>Technology and Data Variations List</i> )	(fakultativer Abschnitt) Alternative I/O-Methoden und Datenformate
<b>Häufigkeit des Auftretens</b> ( <i>Frequency of Occurrence</i> )	(fakultativer Abschnitt) Beeinflusst die Untersuchung, das Testen und die zeitliche Gestaltung der Implementierung
<b>Verschiedenes</b> ( <i>Miscellaneous</i> )	(fakultativer Abschnitt) Beispielsweise offene Probleme


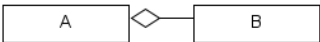
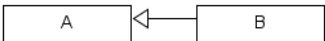
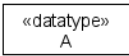
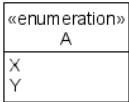
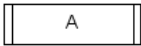
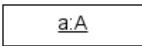
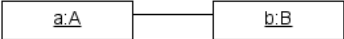
Standardablauf und Erweiterungen sind die beiden Hauptabschnitte.

Fakultative Abschnitte werden beschrieben, wenn es für das Verstehen des Use Cases nützlich und für den Leser mitteilenswert ist.

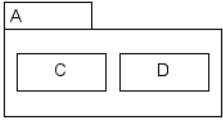
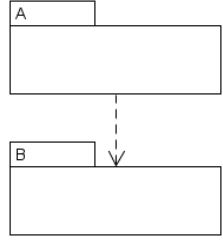
Quelle: Craig Larman, UML 2 und Patterns angewendet, mitp Professional, 2005

## Klassen- und Objektdiagramm (*Class and Object Diagram*)

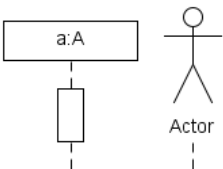
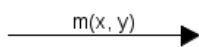
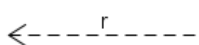
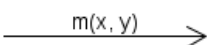
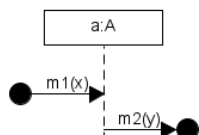
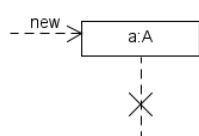
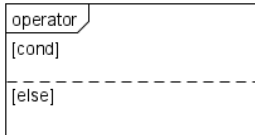
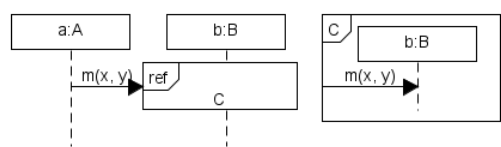
Name	Notation	Erklärung
Klasse ( <i>class</i> )		Beschreibung der Struktur (Attribute) und des Verhaltens (Operationen) einer Menge von Objekten; zusätzliche Spezifikation von Visibilität, abstrakten Operationen und statischen Operationen/Attributen
Abstrakte Klasse ( <i>abstract class</i> )		Klasse, die nicht instanziiert werden kann
Assoziation ( <i>association</i> )		Beziehung zwischen Klassen mit optional Assoziationsname n und Leserichtung >, Rollenbezeichnungen r1, r2 und Multiplizitäten (0, 1, n, n..m, *); keine Angabe über Navigationsrichtung (a), mit Navigationsrichtung (b), in eine Richtung nicht navigierbar (c)
Assoziationsklasse ( <i>association class</i> )		Nähere Beschreibung einer Assoziation; Klasse C mit Attributen und Operationen
Interface ( <i>interface</i> )		Beschreibt eine Menge von öffentlichen Operationen, Merkmalen und «Verpflichtungen», die durch eine Klasse, die die Schnittstelle implementiert, zwingend bereitgestellt werden müssen
Realisierungsbeziehung ( <i>realization relationship</i> )		Klasse B implementiert das Interface A; Alternative Darstellung mit Lollipop- und Socket-Notation für angebotenes und erforderliches Interface
Abhängigkeitsbeziehung ( <i>dependency relationship</i> )		Drückt generell eine Abhängigkeit zwischen Modellelementen aus. B braucht A zur Erfüllung von B (z.B. als Parameter); eine spezielle Abhängigkeit ist die Verwendungsbeziehung (Stereotyp <<use>>)

Komposition ( <i>composition</i> )		Existenzabhängige Teile-Ganzes-Beziehung (B ist Teil von A; wenn A gelöscht wird, werden zugehörige Instanzen von B ebenfalls gelöscht)
Aggregation ( <i>aggregation</i> )		Teile-Ganzes-Beziehung (B ist Teil von A; wenn A gelöscht wird, werden zugehörige Instanzen von B nicht gelöscht)
Generalisierung / Spezialisierung ( <i>generalization / specialization</i> )		Vererbungsbeziehung zwischen Klassen (B erbt von A)
Datentyp ( <i>data type</i> )		Eigener Datentyp mit Attributen und Operationen
Enumeration ( <i>enumeration</i> )		Eigener Aufzählungstyp und allenfalls Operationen
Aktive Klasse ( <i>active class</i> )		Objekte besitzen einen eigenen Thread oder den ausführenden Prozess/Task
Objekt ( <i>object</i> )		Instanz einer Klasse
Link ( <i>link</i> )		Beziehung zwischen Objekten

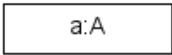
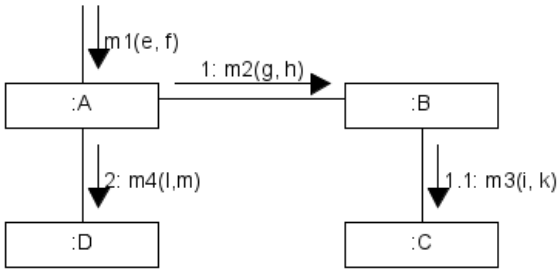
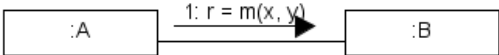
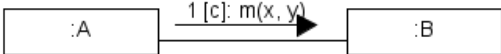
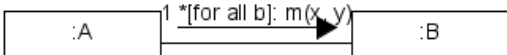
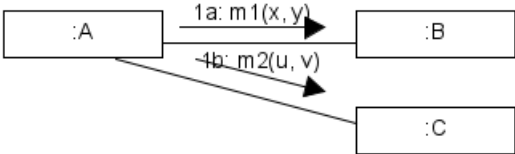
## Paketdiagramm (*Package Diagram*)

Name	Notation	Erklärung
Paket ( <i>package</i> )		Fasst mehrere paketierbare Elemente (Pakete, Klassen, Interfaces, Datentypen etc.) zu einer grösseren Einheit zusammen und bildet einen Namensraum
Abhängigkeitsbeziehung ( <i>dependency relationship</i> )		Beschreibt eine generelle Abhängigkeit zwischen dem Paket A und B; eine Abhängigkeit kommt in der Regel dadurch zustande, dass das abhängige Paket (A) Pakete, Klassen, Interface und/oder Datentypen aus dem unabhängigen Paket (B) importiert; diese Abhängigkeit kann mit dem Stereotyp <<import>> auf dem Pfeil noch markiert werden

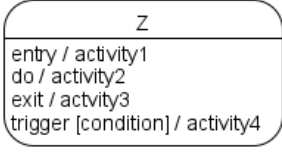
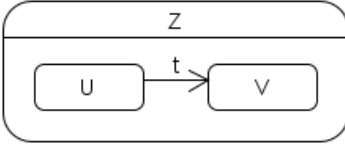
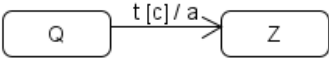


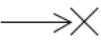

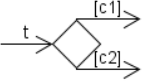
## Sequenzdiagramm (*Sequence Diagram*)

Name	Notation	Erklärung
Lebenslinie ( <i>life line</i> )		An der Kommunikation beteiligte Interaktionspartner; optional mit Aktionssequenz (execution specification), die zeigt, wie lange das Objekt aktiv ist (den Kontrollfluss hat)
Synchrone Nachricht ( <i>synchronous message</i> )		Verschicken der Nachricht m (Methodenaufruf) mit den Parametern x und y; Sender wartet auf eine Antwortnachricht bis er weiterfährt mit der Verarbeitung
Antwortnachricht ( <i>return message</i> )		Antwort mit Rückgabewert r auf synchrone Nachricht
Asynchrone Nachricht ( <i>asynchronous message</i> )		Verschicken der Nachricht m (Methodenaufruf oder Signal) mit den Parametern x und y; Sender setzt nach Absenden der asynchronen Nachricht Verarbeitung sofort fort
Gefundene, verlorene Nachricht ( <i>found, lost message</i> )		Nachricht m1 von unbekanntem Sender («aus dem Nichts») bzw. Nachricht m2 an unbekannten Empfänger («ins Nichts»)
Erzeugungs-, Löschereignis ( <i>create, destroy message</i> )		Zeitpunkt, zu dem ein Interaktionspartner erzeugt (new oder create) bzw. aufhört zu existieren; letzteres kann auch explizit mit einer «destroy»-Nachricht modelliert werden
Kombiniertes Fragment ( <i>combined fragment</i> )		Steuerung des Kontrollflusses mit Operatoren und Bedingungen; Operatoren: loop, alt, opt, break, par
Interaktionsreferenz ( <i>interaction reference/ use</i> )		Aufruf eines anderen Interaktionsdiagramms

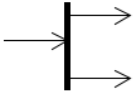
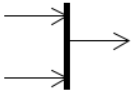
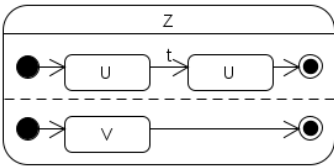
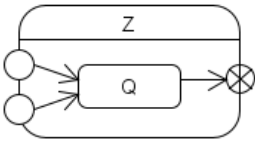
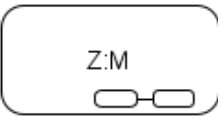
## Kommunikationsdiagramm (*Communication Diagram*)

Name	Notation	Erklärung
Lebenslinie ( <i>life line</i> )		An der Kommunikation betei- ligte Interaktionspartner
Synchrone Nach- richt ( <i>synchronous message</i> )		Sender wartet auf eine Ant- wortnachricht; erste Nachricht m1 ohne Nummer, folgende Nachrichten (m2, m3, m4) werden hierarchisch numme- riert
Antwortnachricht ( <i>return message</i> )		Antwort r auf synchrone Nach- richt m
Bedingte Nach- richt ( <i>conditional mes- sage</i> )		Bedingte Ausführung mit Be- dingung c der Nachricht m
Iteration ( <i>iteration</i> )		Iteration über eine Menge von Objekten; für die Iteration über alle Collection-Elemente einer Collection ist nur die Angabe des «*» auch präzise genug
Parallele Nach- richt ( <i>parallel message</i> )		Nachrichten m1 und m2 wer- den parallel ausgeführt (1a und 1b)

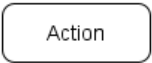
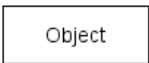


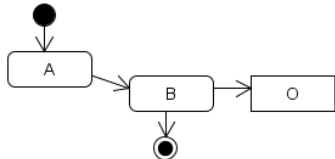
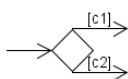
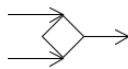
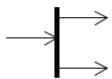
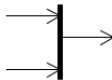
## Zustandsdiagramm (*State Machine Diagram*)


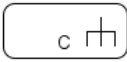
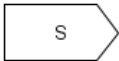

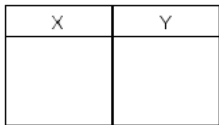
Name	Notation	Erklärung
einfacher Zustand ( <i>simple state</i> )		Beschreibung einer spezifischen «Zeit-spanne», in der sich ein Objekt wäh- rend seines «Lebenslaufs» befindet; im Zustand können Aktivitäten auf dem Objekt ausgeführt werden (entry, do, exit) und weitere «innere» Transitionen definiert werden (trigger)
Zusammengesetzter bzw. geschachtelter Zustand ( <i>composite state</i> )		Zustand Z mit den Subzuständen U und V; beliebige Schachtelungstiefe mög- lich
Transition ( <i>transition</i> )		Zustandsübergang von einem Quellzu- stand Q in einen Zielzustand Z mit Trigger t, Bedingung c und der Aktivität a; Trigger können sein: CallTrigger (Methodenaufruf), SignalTrigger (asyn- chrones Signal), ChangeTrigger (when), TimeTrigger (after)
Startzustand ( <i>initial state</i> )		Beginn des Zustandsautomaten
Endzustand ( <i>final state</i> )		Ende des Zustandsautomaten
Terminator ( <i>terminate state</i> )		Bricht die Ausführung des Zustandsau- tomaten ab (Lebensdauer des Objektes ist beendet)
Flacher/Tiefer History-Zustand ( <i>shallow/deep history</i> )		«Rücksprungadresse» in einen Subzu- stand bzw. geschachtelten Subzustand eines zusammengesetzten Zustands; flache History («H») berücksichtigt nur eine Ebene und tiefe History («H*») be- lieb viele Ebenen
Entscheidungsknoten ( <i>decision node</i> )		Knoten, von dem mehrere alternative Transitionen ausgehen können; an der eingehenden Kante wird der Trigger t und an den ausgehenden Kanten die Bedingungen c1, c2 spezifiziert



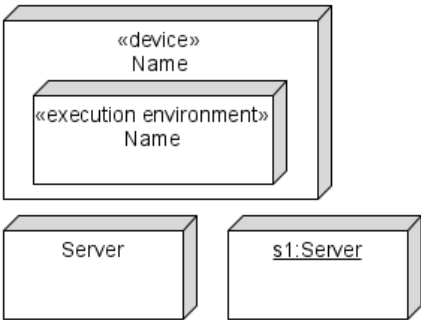
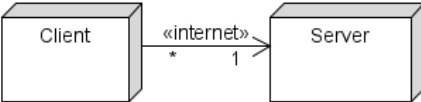
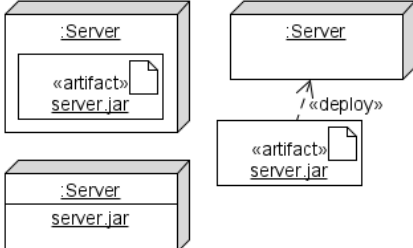
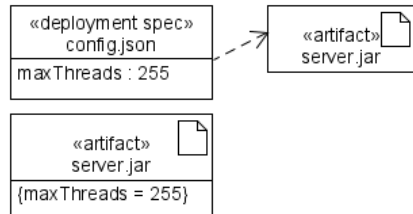
Parallelisierungsknoten ( <i>fork node</i> )		Aufspaltung einer Transition in mehrere parallele Transitionen
Synchronisationsknoten ( <i>join node</i> )		Zusammenführung von mehreren parallelen Transitionen in eine Transition
Orthogonaler Zustand ( <i>orthogonal state</i> )		Zusammengesetzter Zustand mit Regionen, die parallel ausgeführt werden; Regionen können auch noch benannt werden
Einstiegs-/Ausstiegspunkt ( <i>entry/exit point</i> )		Dienen der Übersichtlichkeit in zusammengesetzten Zuständen und Unterzustandsautomaten, um Transitionen zu ordnen und um eine Schnittstelle nach aussen zu definieren
Unterzustandsautomat ( <i>submachine state</i> )		Der Zustand Z steht stellvertretend für den Zustandsautomaten M

## Aktivitätsdiagramm (*Activity Diagram*)

Name	Notation	Erklärung
Aktionsknoten, Aktion ( <i>action node, action</i> )		Aktionen beschreiben beliebiges Benutzer- oder Programm-Verhalten; Aktionen sind atomar, d. h. nicht weiter zerlegbar; eine Aktivität (activity) beschreibt eine Menge von Aktionen und deren Ablauf
Objektknoten, Objekt ( <i>object node, object</i> )		Enthalten Daten und Objekte, die erzeugt, verändert und gelesen werden
Initialknoten ( <i>initial node</i> )		Beginn eines Ablaufs einer Aktivität
Aktivitätsendknoten ( <i>activity final node</i> )		Ende aller Abläufe einer Aktivität
Kante ( <i>activity edge</i> )		Verbindung der Knoten einer Aktivität (Kontrollfluss oder Datenfluss)
Entscheidungsknoten ( <i>decision node</i> )		Aufspaltung eines Ablaufs in alternative Abläufe; an den ausgehenden Kanten sind Bedingungen c1, c2 spezifiziert
Vereinigungsknoten ( <i>merge node</i> )		Zusammenführung von alternativen Abläufen in einen Ablauf
Parallelisierungsknoten ( <i>fork node</i> )		Aufspaltung eines Ablaufs in mehrere parallele Abläufe
Synchronisationsknoten ( <i>join node</i> )		Zusammenführung von mehreren parallelen Abläufen in einen Ablauf

Ablaufendknoten ( <i>flow final node</i> )		Ende von einem Ablauf einer Aktivität (bei parallelen Abläufen)
CallBehavior-Aktion ( <i>call behavior action</i> )		Aktion C verweist auf eine Aktivität gleichen Namens (Unteraktivität)
SendSignal-Aktion ( <i>send signal action</i> )		Asynchrone Übermittlung eines Signals S an einen Empfänger
Asynchrone Ereignis-/Zeitereignisannahme-Aktion ( <i>accept event/time action</i> )		Warten auf ein Ereignis E bzw. ein Zeitereignis T
Partition ( <i>activity partition, swimlane</i> )		Gruppierung von Knoten und Kanten innerhalb einer Aktivität

## Verteilungsdiagramm (*Deployment Diagram*)

Name	Notation	Erklärung
Knoten ( <i>node</i> )		Ein Knoten repräsentiert eine Ressource, die z.B. zur Installation, Konfiguration, Bereitstellung und Ausführung von Artefakten genutzt werden kann; Knoten können sowohl auf der Typ-Ebene als auch auf der Ausprägungs-Ebene definiert werden; zwei vordefinierte Knoten sind: Gerät (device) und Ausführungsumgebung (execution environment)
Kommunikationspfad ( <i>communication path</i> )		Durch Kommunikationspfade können Knoten sowohl auf Typ- als auch auf Instanz-Ebene miteinander verbunden werden, um Nachrichten (Signale oder Operationsaufrufe) auszutauschen; optional können Kommunikationsrichtung, den Typ der Kommunikation sowie Multiplizitäten spezifiziert werden (nur auf Typ-Ebene)
Verteilungsbeziehung ( <i>deployment</i> )		Die Verteilungsbeziehung (deployment) repräsentiert die Beziehung zwischen einem Artefakt und dem Knoten, auf den das Artefakt verteilt ist; es gibt drei verschiedene Darstellungsformen; vordefinierte spezielle Artefakte sind: <<library>>, <<file>>, <<document>>
Einsatzspezifikation ( <i>deployment specification</i> )		Eine Einsatzspezifikation (deployment specification) ist eine spezielle Art eines Artefakts; sie beinhaltet eine Menge von Parametern, die durch Angabe von Werten die Verteilung eines Artefakts auf Knoten regelt (Konfiguration); es gibt zwei verschiedene Darstellungsarten