UML
(Unified Modelling Language)

von Christian Bartl

Inhaltsverzeichnis

1 UML – Unified Modelling Language

Die Unified Modelling Language (UML) ist eine von der Object Management Group (OMG) entwickelte und standardisierte Sprache für die Modellierung von Software. Die Väter der UML sind Grady Booch, Ivar Jacobson und James Rumbaugh, auch "die drei Amigos" genannt. Dabei kann UML zur Spezifizierung, Visualisierung, Dokumentierung und Konstruktion eines Entwurfs von großen Softwaresystemen genutzt werden. Um diese Aufgaben bewältigen zu können kennt UML verschiedene Diagrammtypen und eine umfangreiche Notation. Die aktuelle Version ist UML2. Auch wenn UML für die grafische Darstellung von Vorgängen in der Softwareentwicklung entwickelt wurde, wird es heute in vielen Bereichen eingesetzt (grafische Darstellung von Datenbanken, Geschäftsmodellen, usw.)

2 Diagrammtypen

Die aktuelle UML2-Definition kennt 13 verschiedene Diagrammtypen für die Softwareentwicklung.

6 Strukturdiagramme

- Klassendiagramm (class diagram)
- Kompositionsstrukturdiagramm (composite structure diagram)
- Komponentendiagramm (component diagram)
- Verteilungsdiagramm (deployment diagram)
- Objektdiagramm (object diagram)
- Paketdiagramm (package diagram)

7 Verhaltensdiagramme

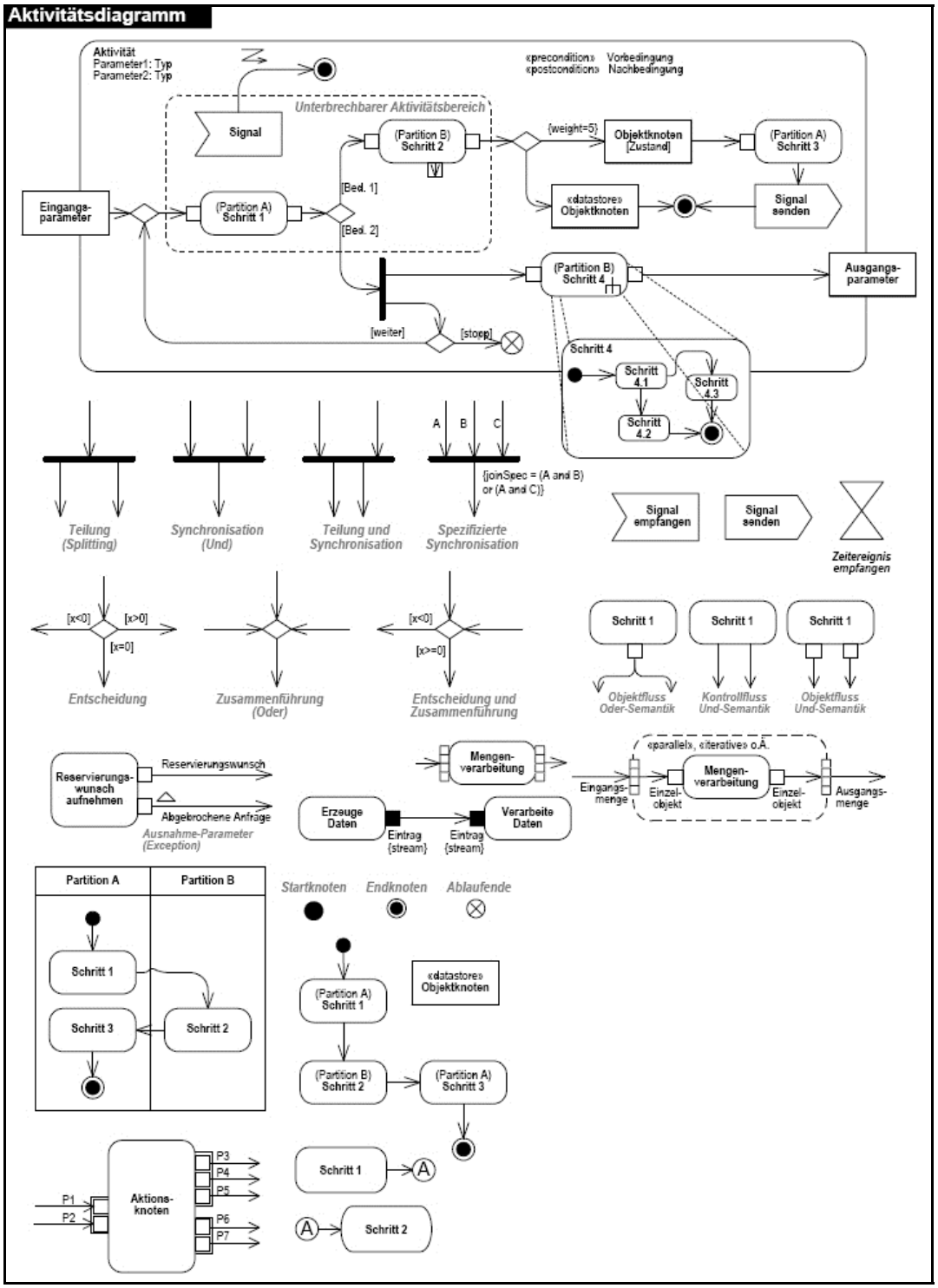
- Anwendungsfalldiagramm (use case diagram)
- Aktivitätsdiagramm (activity diagram)
- Sequenzdiagramm (sequence diagram)
- Kommunikationsdiagramm (communication diagram)
- Interaktionsübersicht (interaction overview diagram)
- Zeitverlaufsdiagramm (timing diagram)
- Zustandsdiagramm (state chart diagram)

Davon sind das Klassen-, Objekt-, Anwendungsfall- und Aktivitätsdiagramm mit Sicherheit die in der Softwareentwicklung am meist genutzten und am öftesten benötigten Diagrammtypen.

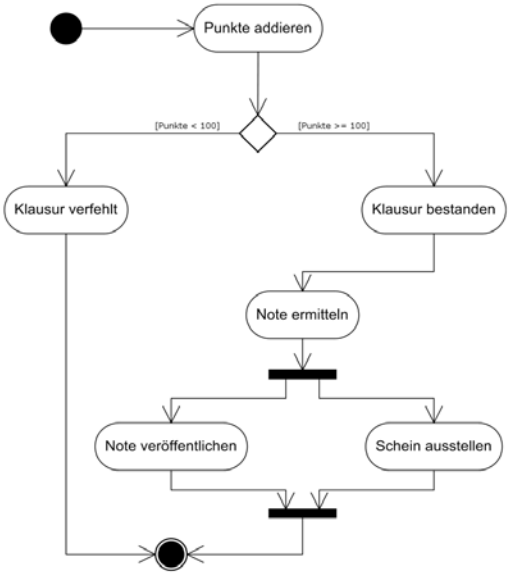
2.1 Aktivitätsdiagramm

Das Aktivitätsdiagramm (engl. activity diagram) ist eines der 7 Verhaltensdiagramme. Es spezifiziert eine Aktivität und stellt diese graphisch dar. Dabei werden die elementaren Aktionen abgebildet und mit Kontroll- und Datenflüssen verbunden. Häufig wird mit diesem Diagrammtyp der Ablauf eines Anwendungsfalls beschrieben, es eignet sich aber auch zur Modellierung aller Aktivitäten innerhalb eines Systems. Das Aktivitätsdiagramm ersetzt den aus der prozeduralen Programmierung bekannten Projektablaufplan (PAP):

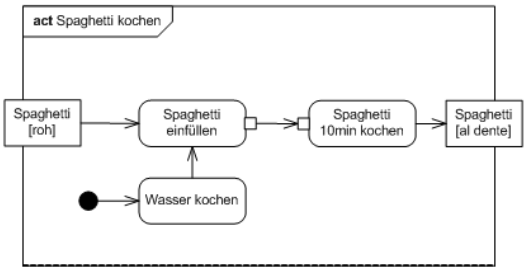
2.1.1 Notation



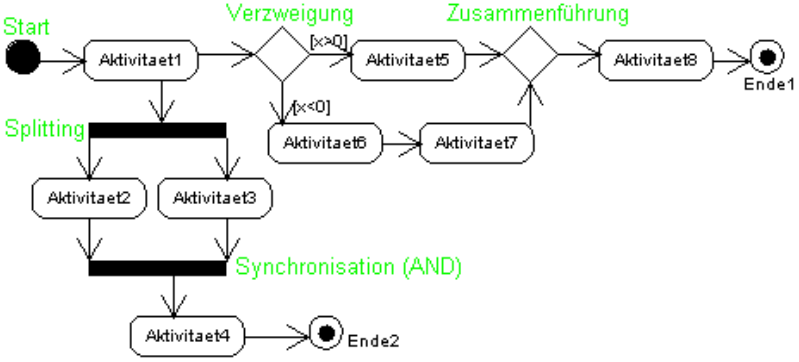
2.1.2 Beispieldiagramm



2.1.3 Beispieldiagramm 2



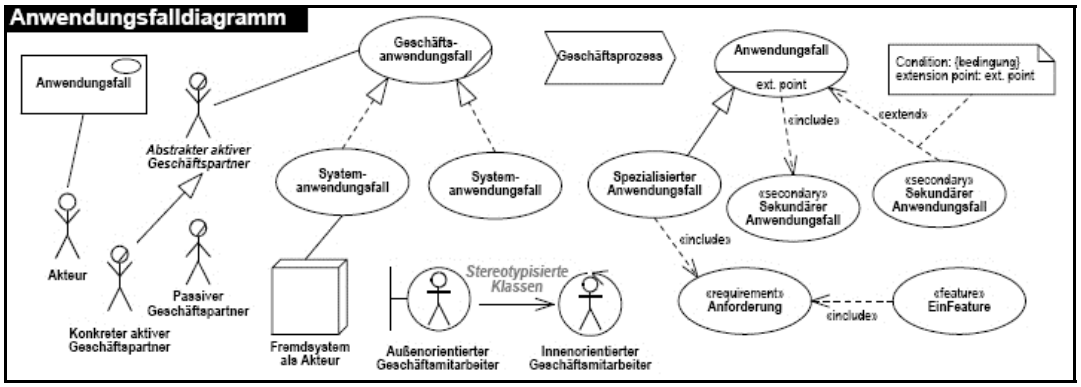
2.1.4 Beispieldiagramm 3



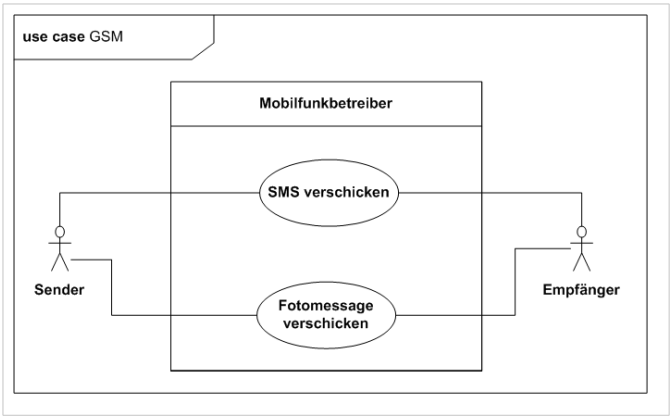
2.2 Anwendungsfalldiagramm

Das Anwendungsfalldiagramm (engl. use case diagramm) gehört zur Kategorie der Verhaltensdiagramme. Ein Anwendungsfalldiagramm stellt keine Ablaufbeschreibung dar, sondern es bildet Szenarien und Akteure mit ihren Abhängigkeiten und Beziehungen ab. Es zeigt eine bestimmte Sicht auf das erwartete Verhalten eines Systems und wird deshalb für die Spezifikation der Anforderungen an ein System eingesetzt.

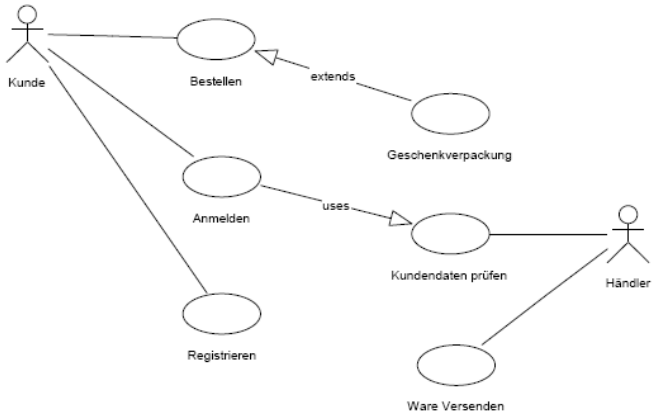
2.2.1 Notation



2.2.2 Beispieldiagramm 1



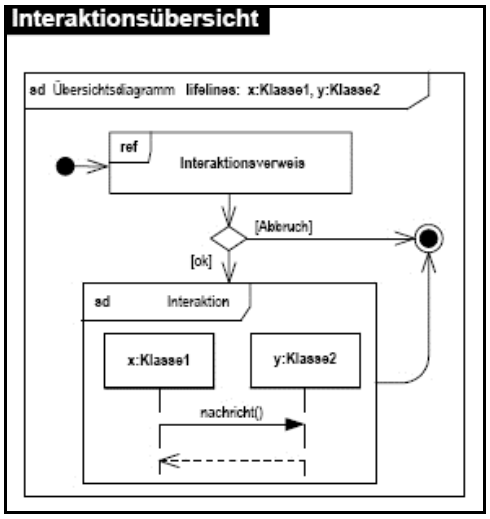
2.2.3 Beispieldiagramm 2



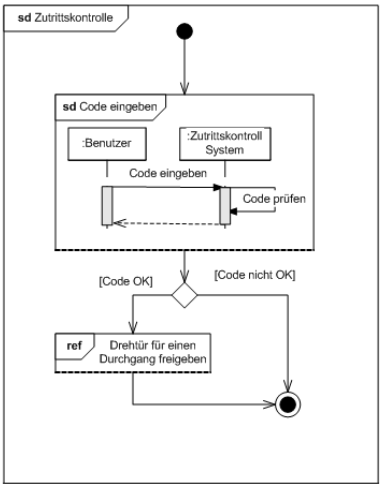
2.3 Interaktionsübersicht

Das Interaktionsübersichtsdiagramm (engl. interaction overview diagram) gehört zu den 7 Verhaltensdiagrammen. Es ist eine grafische Darstellung von Interaktionen. Eine Interaktion bezeichnet das wechselseitig aufeinander Einwirken von Akteuren und Systemen. Für komplexe Systeme können Interaktionsmodelle sehr groß werden und es empfiehlt sich dann einzelne Teile in Sequenzdiagramme auszulagern und die Interaktionsübersicht nur für die Gesamtübersicht zu verwenden.

2.3.1 Notation



2.3.2 Beispieldiagramm



2.4 Klassendiagramm

Das Klassendiagramm (eng. class diagram) gehört in die Kategorie der Strukturdiagramme. Dieses ist die grafische Darstellung von Klassen und deren Beziehungen untereinander. Das UML-Klassendiagramm ist ein wesentlicher Bestandteil der objektorientierten Programmierung und wird seit den 1990er Jahren verwendet.

Das Klassendiagramm stellt im Wesentlichen folgende Informationen dar:

- Klassenname
- Eigenschaften (Attribute)
- Methoden (Operationen)
- Beziehungen (Assoziationen)
- Vererbungen (Generalisierungen)

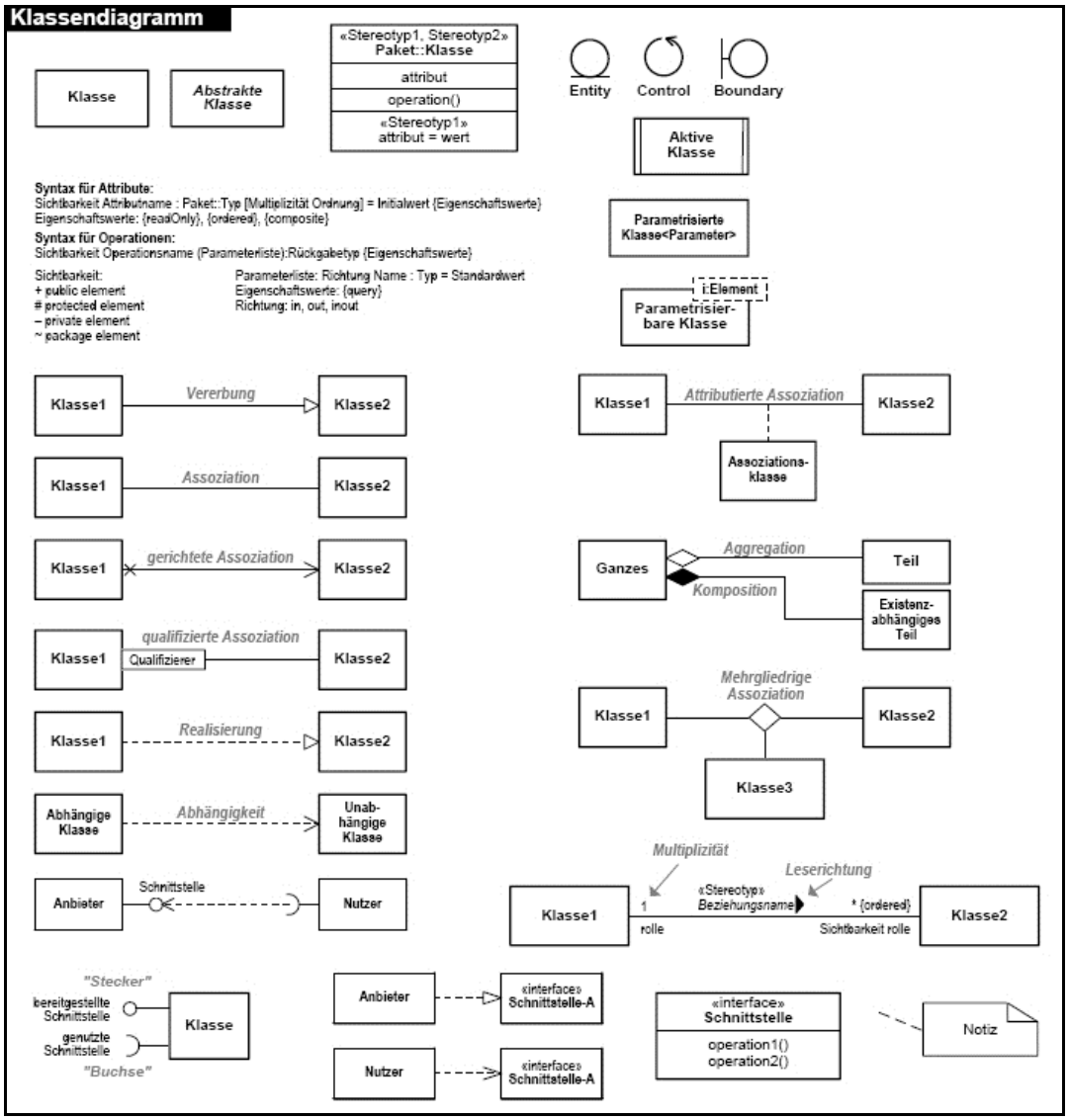
Die Sichtbarkeit von Methoden und Attributen wird wie folgt gekennzeichnet:

+	für public
#	für protected
~	für package
−	für private

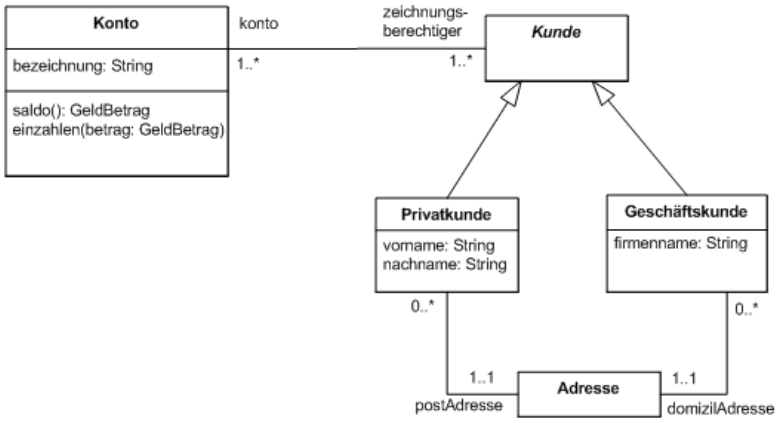
Die Kardialitäten werden wie folgt angegeben:

1	für genau ein Objekt
*	für viele, d.h. Null oder mehr Objekte
0..1	für höchstens 1 Objekt
m..n	für mindesten m aber höchstens n Objekte

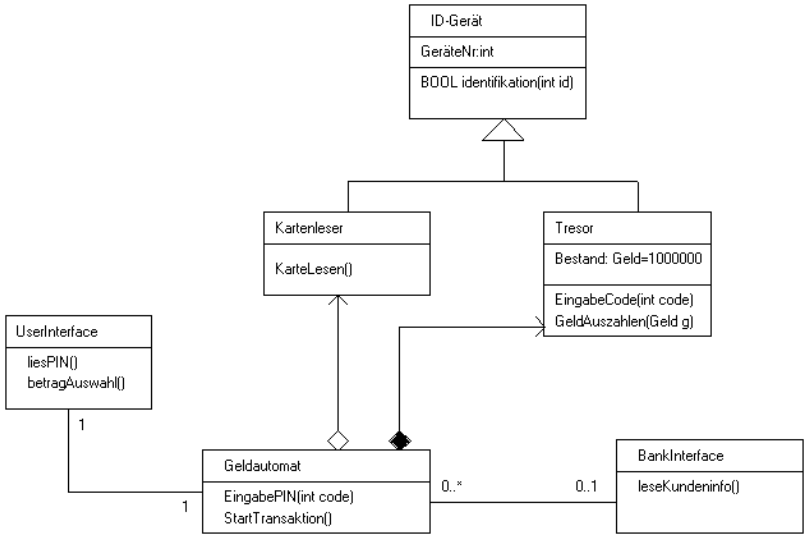
2.4.1 Notation



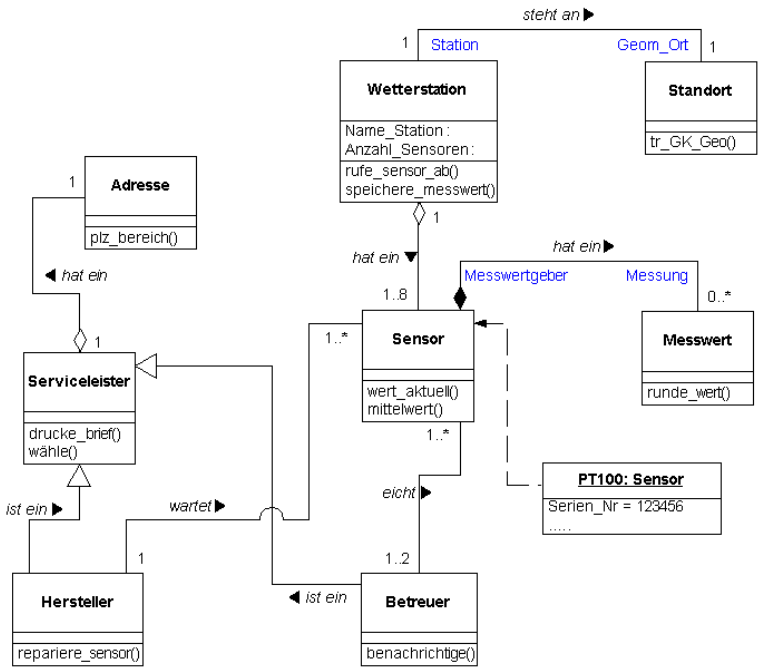
2.4.2 Beispieldiagramm



2.4.3 Beispieldiagramm 2



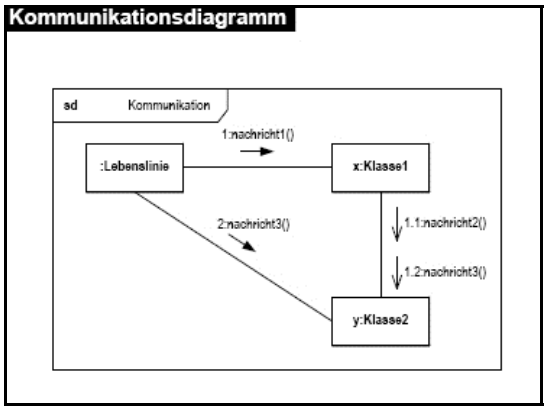
2.4.4 Beispieldiagramm 3



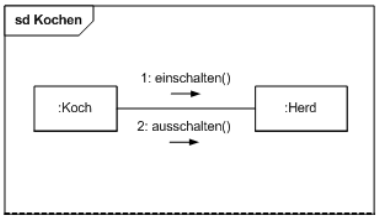
2.5 Kommunikationsdiagramm

Das Kommunikationsdiagramm (engl. communication diagram) gehört zur Kategorie der Verhaltensdiagramme. In älteren UML-Versionen ist das Kommunikationsdiagramm auch unter dem Namen Kollaberationsdiagramm bekannt. Das Kommunikationsdiagramm ist die grafische Darstellung von Interaktionen und spezifiziert den Austausch von Nachrichten zwischen Ausprägungen. Der Nachrichtenfluss wird im Diagramm als Lebenslinie dargestellt.

2.5.1 Notation



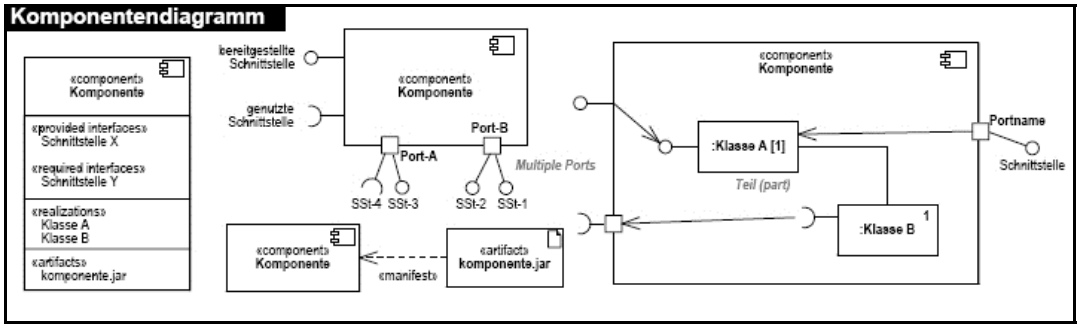
2.5.2 Beispieldiagramm



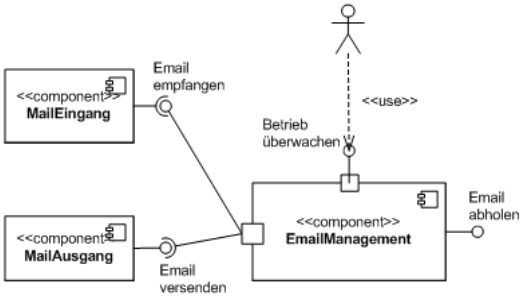
2.6 Komponentendiagramm

Das Komponentendiagramm (engl. component diagramm) ist eines der 6 Strukturdiagramme. Das Komponentendiagramm wird vor allem in der Softwareentwicklung für die Modellierung von komponentenbasierten Softwaresystemen eingesetzt. Es werden einzelne Komponenten deren „Andockstellen“, Schnittstellen und Ports dargestellt. Es zeigt außerdem wie Komponenten über Beziehungen und Konnektoren miteinander verbunden sind. Um das Innere der Komponenten abzubilden werden Notationselemente aus Klassendiagrammen verwendet.

2.6.1 Notation



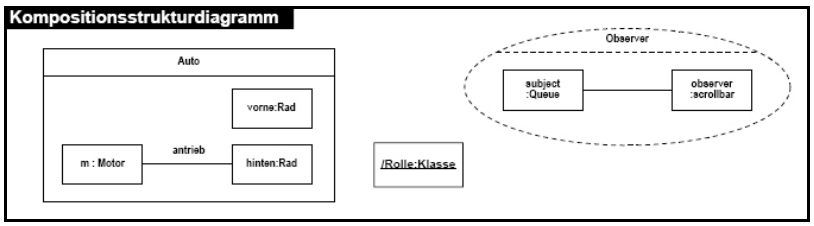
2.6.2 Beispieldiagramm



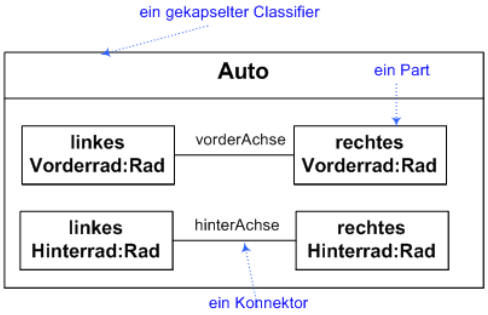
2.7 Kompositionsstrukturdiagramm

Das Kompositionsstrukturdiagramm (engl. composite structure diagram) gehört wie der Name schon vermuten lässt zu den Strukturdiagrammen. Es stell das Innere eines Classifiers und dessen Zusammenarbeit mit seiner Umgebung dar. Eine Classifier (deut. Klassifizierer) ist eine Metaklasse. Eine Metaklasse ist in der Objektorientierung die Klasse einer Klasse. Sie definiert Verfahren zum Erzeugen von Instanzen der Klasse, deren Metaklasse sie ist, sowie statische Methoden, also solche, für deren Ausführung keine Objekte benötigt werden.

2.7.1 Notation



2.7.2 Beispieldiagramm



2.8 Objektdiagramm

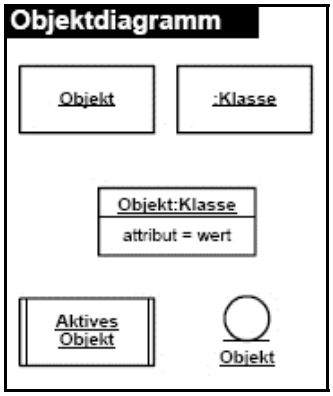
Das Objektdiagramm (engl. object diagram) zählt zu den 6 Strukturdiagrammen der UML2-Spezifikation. Ein Objektdiagramm zeigt den Aufbau von Objekten, deren Assoziationen (Referenzen) und Zustände.

Jedes Objekt (unabhängig vom Objektdiagramm) lässt sich durch folgende 3 Punkte beschrieben:

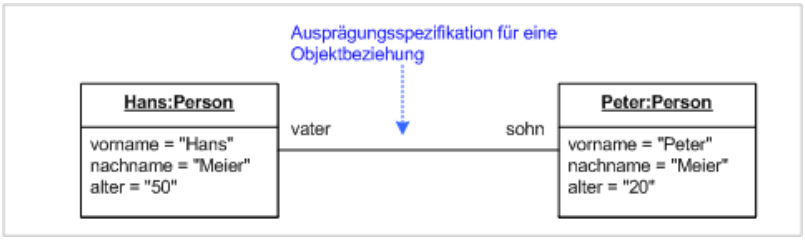
- Identität
  Eindeutiger Name des Objekts, unabhängig vom Zustand und Verhalten.
- Zustand
  Der Zustand wird durch die Werte der Attribute eines Objektes beschrieben.

- Verhalten
  Das Verhalten beschreibt alle Operationen die mit einem Objekt ausgeführt werden können.

2.8.1 Notation



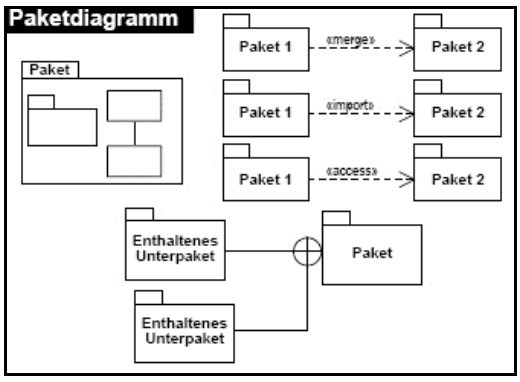
2.8.2 Beispieldiagramm



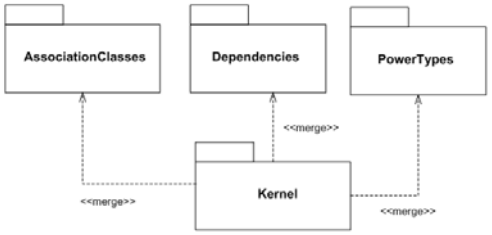
2.9 Paketdiagramm

Das Paketdiagramm (engl. package diagram) ist ein Diagramm aus der Kategorie der Strukturdiagramme. Paketdiagramme werden zur grafischen Darstellung von Paketen und deren Imports sowie Abhängigkeit verwendet. In der Softwareentwicklung stellen Paketdiagramme die Unterteilung der Software in Module dar.

2.9.1 Notation



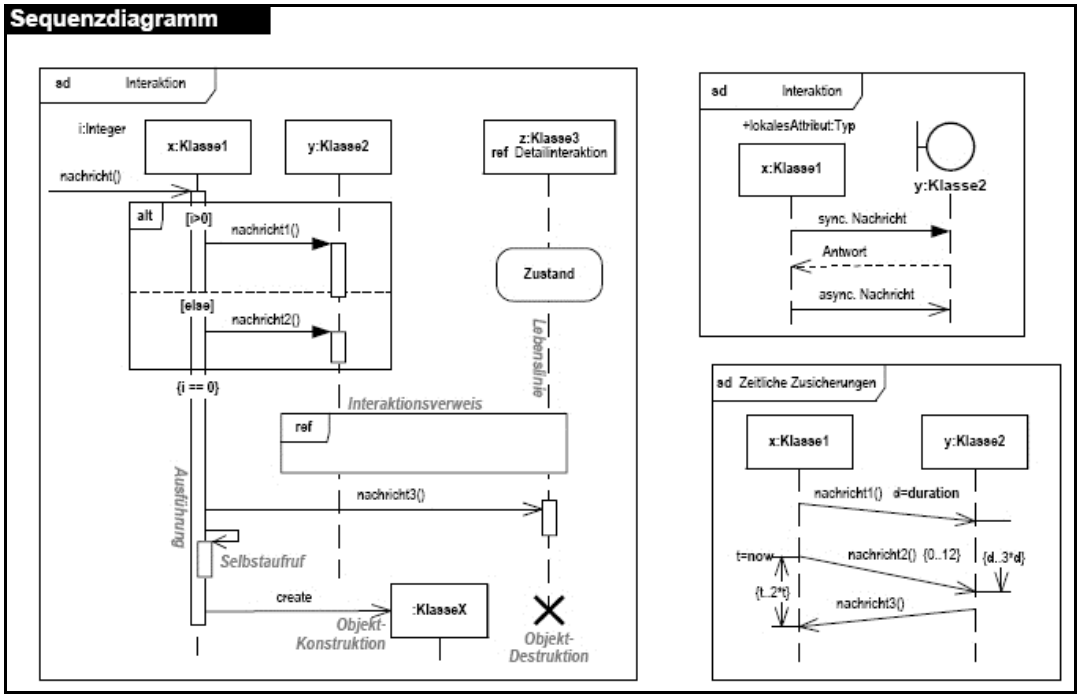
2.9.2 Beispieldiagramm



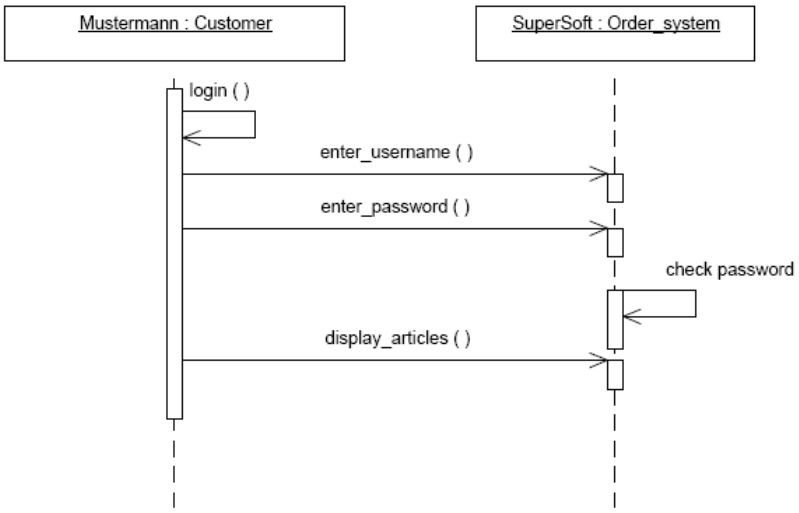
2.10 Sequenzdiagramm

Das Sequenzdiagramm (engl. sequence diagram) gehört zu den Verhaltensdiagrammen. Ein Sequenzdiagramm stellt den zeitlichen Ablauf des Austausches von Nachrichten zwischen einzelnen Ausprägungen dar. Es stellt in der Regel einen Weg durch einen Entscheidungsbaum innerhalb eines Systemablaufes dar. Entscheidungsbäume veranschaulichen aufeinanderfolgende, hierarchische Entscheidungen.

2.10.1 Notation



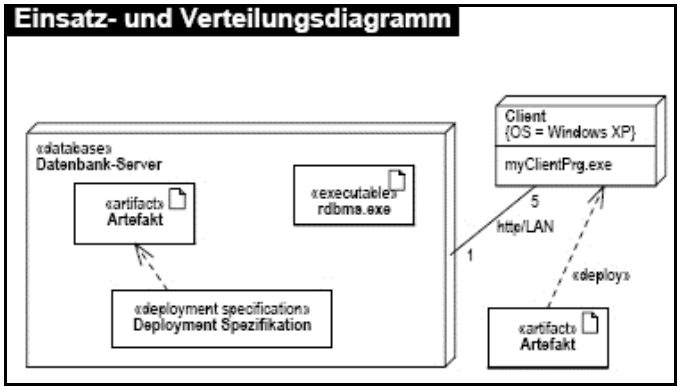
2.10.2 Notation



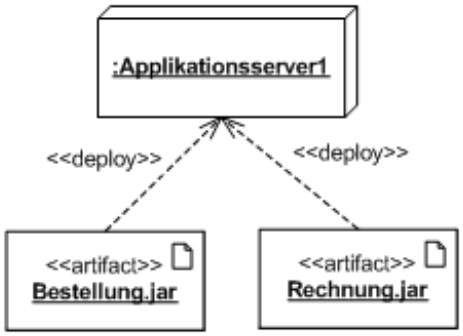
2.11 Verteilungsdiagramm

Das Verteilungsdiagramm (engl. deployment diagram) ist ein Strukturdiagramm. In einem Verteilungsdiagramm werden Knoten und Artefakte sowie deren Verteilungen und Beziehungen dargestellt.

2.11.1 Notation



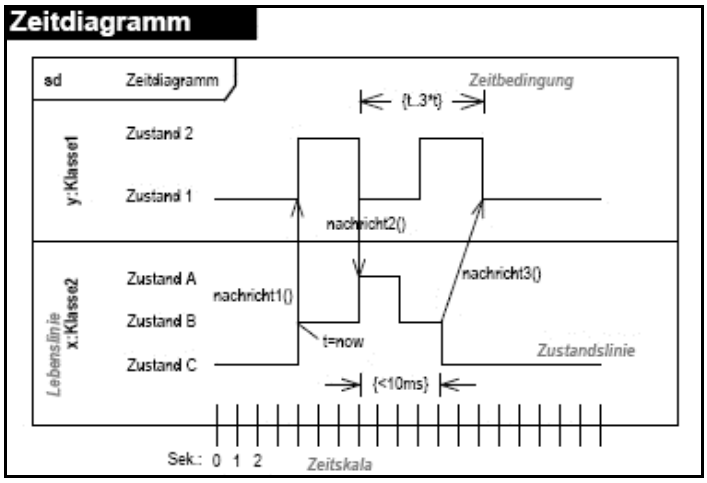
2.11.2 Beispieldiagramm



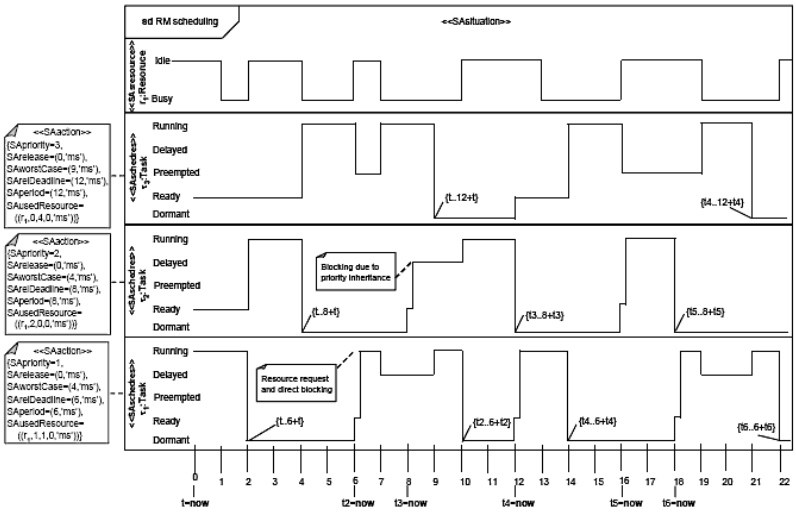
2.12 Zeitverlaufsdiagramm

Das Zeitverlausdiagramm (eng. timing diagram) zählt zu den Verhaltensdiagrammen. Es ist ein zweidimensionales Diagramm, bei dem auf der x-Achse die Zeit und auf der y-Achse die Objekte und deren Zustände dargestellt werden. Dieser Diagrammtyp ähnelt damit der Anzeige eines Oszilloskops, wird also seit langem in der Elektrotechnik verwendet.

2.12.1 Notation



2.12.2 Beispieldiagramm

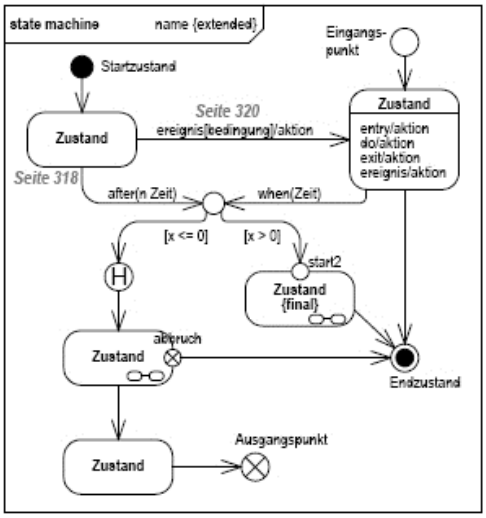


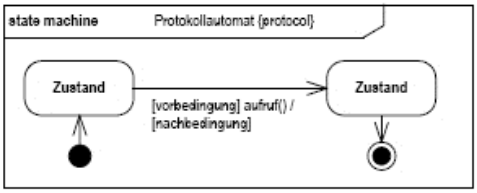
2.13 Zustandsdiagramm

Das Zustandsdiagramm (eng. state chart diagram) gehört zu den 7 Verhaltensdiagrammen der UML2-Spezifikation. Es stellt Statusübergänge innerhalb des Systems dar. Das Diagramm hat einen Start- und einen Endpunkt und dazwischen gibt es unterschiedliche Zustände. Weiters werden die Übergänge zwischen den Zuständen sichtbar.

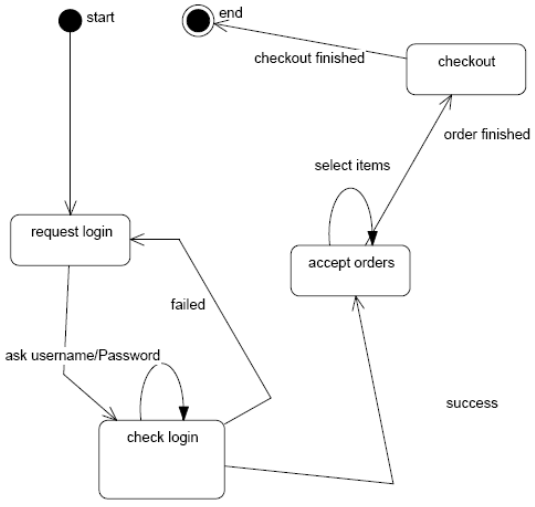
2.13.1 Notation

Zustandsdiagramm





2.13.2 Beispieldiagramm



3 UML und Programmierparadigma

Mit Hilfe von UML ist es möglich, objektorientierte Entwicklungsprozesse durch grafische Darstellung zu beschreiben und zu unterstützen. Neben der Nutzung als Darstellungsmittel kann UML auch direkt zur Erzeugung von Programmcode genutzt werden. Mit den geeigneten Werkzeugen wie Rational Rose, können aus Diagrammen die entsprechenden Klassen in C++ oder Java erzeugt werden, sowie natürlich auch umgekehrt aus Quellcode die entsprechenden Diagramme.

UML dient also vor allem für die Darstellung des Programmierparadigmas der Objektorientierung.