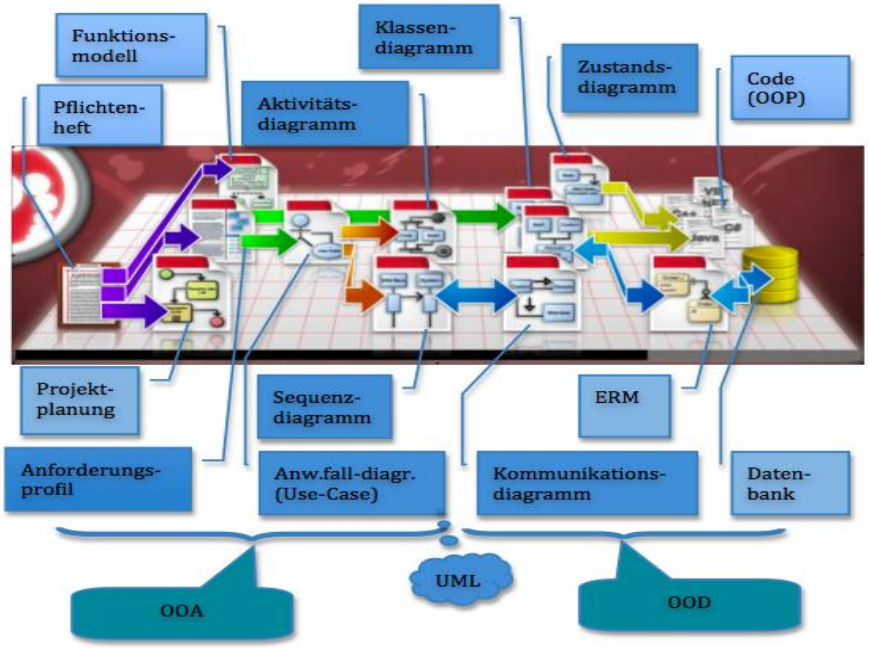
INFORMATIONS-TECHNIK
Ausstellungsstrasse 70
CH-8090 Zürich
Homepage www.tbz.ch
Telefon 044 446 96 60
Telefax 044 446 96 66
E-Mail admin.it@tbz.zh.ch



UML-Diagrammtypen mit Übungen

Version 2.0

D. A. Waldvogel
12.03.2018

TECHNISCHE BERUFSSCHULE ZÜRICH

Inhaltsverzeichnis

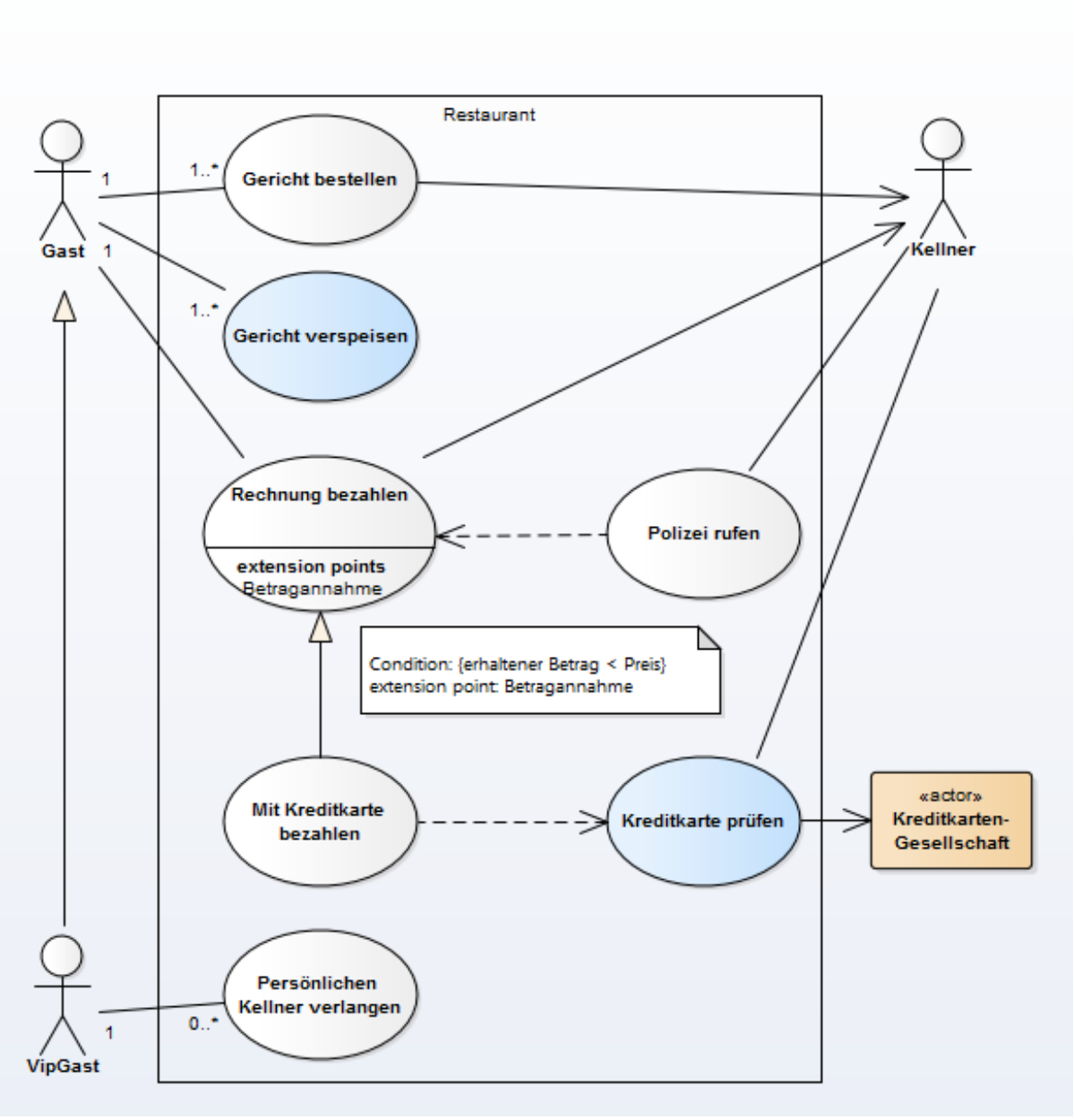
1 Anwendungsfalldiagramme (UseCase)

1.1 Anwendungsbereiche

Anwendungsfalldiagramme modellieren die Funktionalität des Systems auf einem hohen Abstraktionsniveau aus der sogenannten Black-Box Sicht des Anwenders.

Die Modellierung beschreibt, was für Anwendungsfälle das System anbietet, und nicht, wie sie im System realisiert werden.

Anwendungsfälle werden überwiegend während der Analyse und Definition eingesetzt im frühen Stadium eines Softwareprojekts



Ein Anwendungsfall-Diagramm zeigt somit:

- Die Akteure (wer interagiert mit dem System?)
- Das System (die Applikation)
- Die Anwendungsfälle (was „tut“ der Akteur mit dem System?)
- Beziehungen zwischen diesen Elementen

Keinen Ablauf modellieren

Ein Anwendungsfall-Diagramm zeigt nie einen Ablauf. Es zeigt eine Sammlung von Anwendungen. Diese Anwendungsfälle werde sehr allgemein gehalten (man kann dann immer noch einen Anwendungsfall im Detail genauer ausskizzieren, Bsp. als spezialisierter Anwendungsfall).

Ein Ablauf kann mittels eines *Aktivitätsdiagramms* (zeigt nur den Datenfluss) oder eines *Sequenzdiagramms* (zeigt wer macht wann was) abgebildet werden.

<<include>> und <<extend>>

Spezifischere Anwendungsfälle werden mittels der Bezeichnung *include* und *extend* differenziert. Dabei wird ***include*** für Anwendungsfälle verwendet, die zwingend vom verbundenen Anwendungsfall verlangt werden. Es ist zwingend, dass die Kreditkarten überprüft werden, wenn mit Kreditkaten bezahlt wird. s.a. Bsp.

Extend weist auf einen zusätzlichen Anwendungsfall hin, der möglicherweise innerhalb des anderen Anwendungsfalls verlangt wird. In diesem Beispiel kann es sein, dass die **Polizei gerufen** werden muss, wenn es bei der **Rechnung bezahlen** zu Schwierigkeiten kommt.

Die Grenze eines Diagramms

Wie Beispiele aus dem Netz zeigen, sprengen solche Diagramme rasch den Rahmen. Es muss dann abgewogen werden, ob man Diagramme in genauere Diagramme runterbricht („drop-down“) oder ob das Diagramm nur grobe Fälle zeigt, die man dann Bsp. in Anwendungsfall-Szenarien genauer beschreibt.

http://www.andrew.cmu.edu/course/90-754/umlucdfaq.html

Use Case Beschreibungen

Jeder Anwendungsfall wird mit diesem Raster beschrieben.

Gibt es Vorbedingungen?

Was ist das Resultat nach dem Anwendungsfall (Nachbedingung)?

Gibt es Ausnahmen (Beispiel: User gibt etwas Falsches ein, was passiert?)

Use Case	
Vorbedingung	
Beschreibung Anwendungsfall	
Nachbedingung	
Ausnahmen	

Aufgabe: Beschreiben Sie folgenden UseCase: **Fahrzeug ausleihen** mit dem vorgegebenen Raster

Zusammenfassung: *Fahrzeug ausleihen*

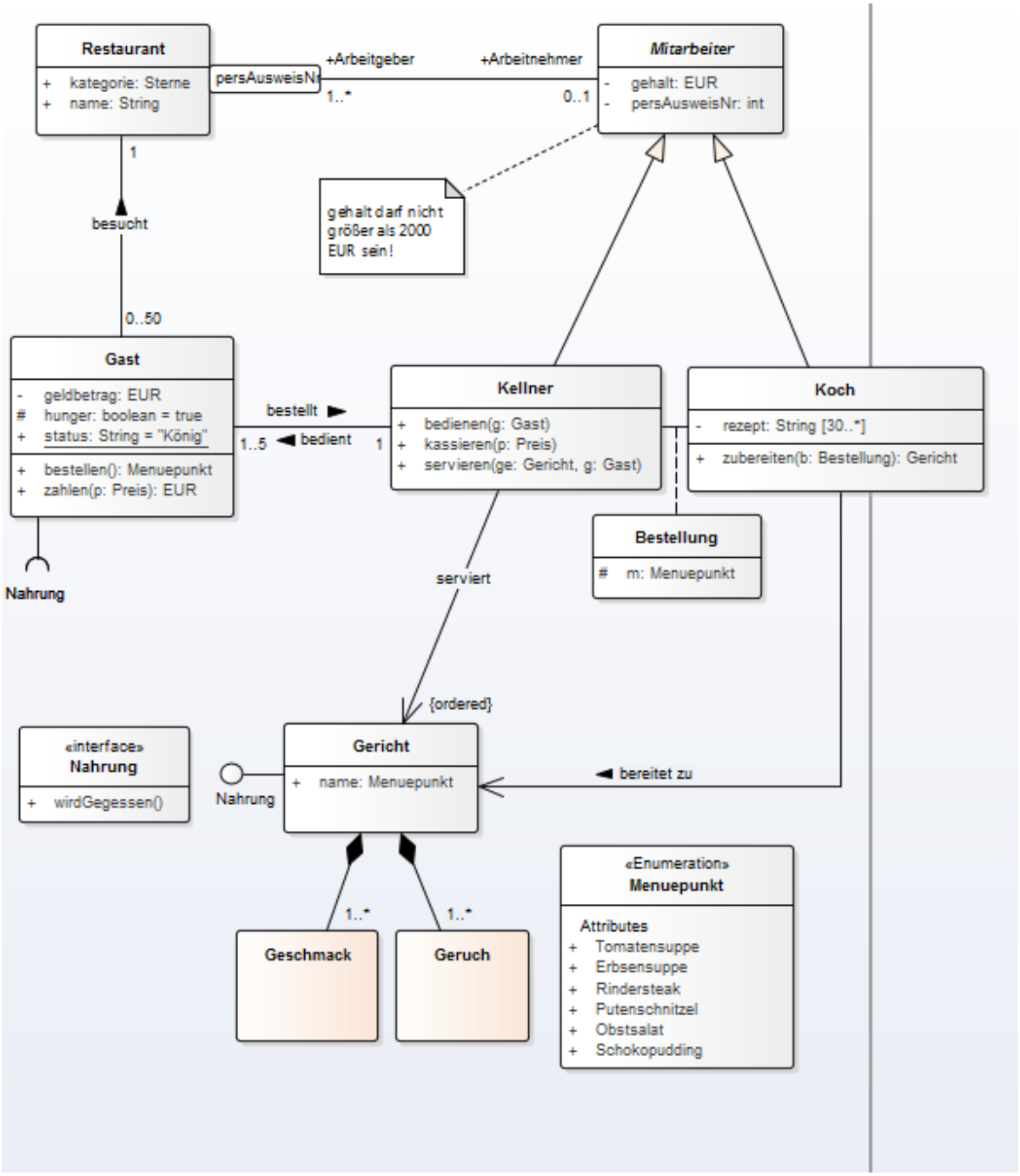
Ein Kunde kommt, leiht sich ein Fahrzeug und fährt damit an den gewünschten Ort. Der Ausleihungsvertrag wird unterschrieben und das Fahrzeug anschliessend dem Kunden ausgehändigt.

2 Klassendiagramme

2.1 Anwendungsbereiche

Klassendiagramme stellen das zentrale Konzept der UML dar. Sie zeigen die statischen Bestandteile und Attribute von Systemen und welche Beziehungen sie untereinander einnehmen können. Klassendiagramme werden üblicherweise in den ersten beiden Phasen Analyse und Entwurf verwendet.

- **Analyse:** Aus Sicht der Anwender bzw. Auftraggeber relevanten Bestandteile eines Systems.
- **Design:** logische Klassen, die in der nachfolgenden Implementation auch tatsächlich realisiert werden.



Details zu Klassendiagrammen

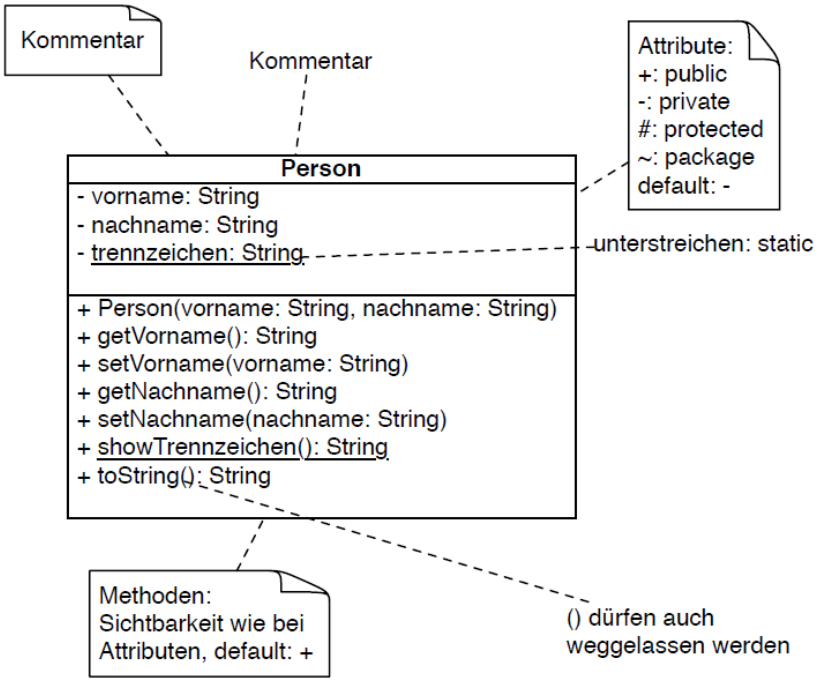
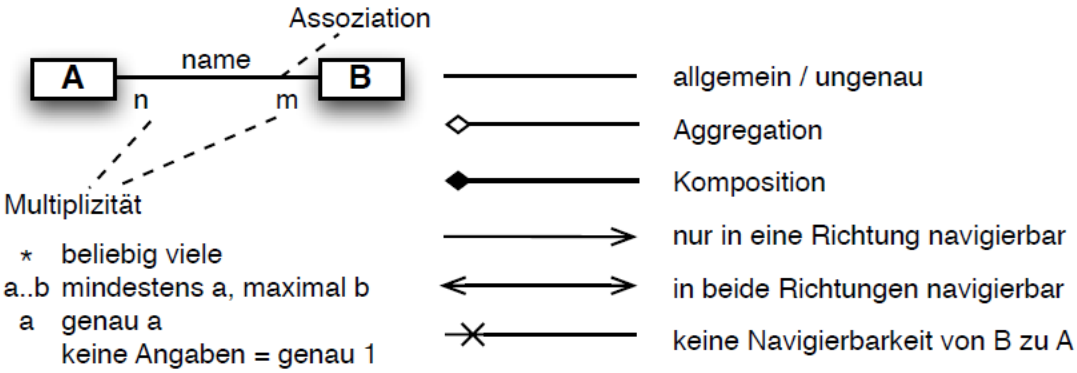


Diagramm zur Klasse Person

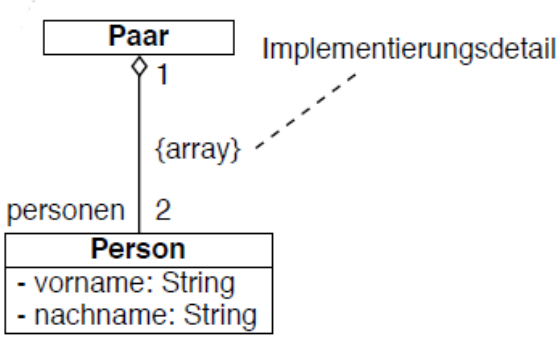
Der Diagrammtyp Klassendiagramm dient dazu, eine oder mehrere Klassen, unabhängig von der Sprache, in der sie implementiert wurden/werden sollen, abzubilden. Es können sowohl die Klasse(n) selbst als auch ihre Beziehungen zu anderen Klassen dargestellt werden.



Assoziationen beschreiben die Beziehungen unter mehreren Klassen

Zur genaueren Spezifizierung können sie mit Multiplizitäten versehen werden. Eine Multiplizität bezeichnet die Anzahl der verbundenen Objekte. Keine Angabe einer Multiplizität entspricht der Angabe 1.

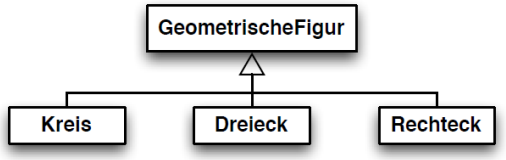
Neben der Multiplizität können sie auch durch einen Namen gekennzeichnet werden. Dies ist sinnvoll, wenn zwischen zwei Objekten zwei unterschiedliche Beziehungen bestehen sollen. Die ungefüllte Raute an einem Ende wird als **Aggregation** bezeichnet und drückt aus, dass Objekte der Klasse B zu Objekten der Klasse A gehören, B-Objekte aber auch allein existieren können. Die gefüllte Raute wird als **Komposition** bezeichnet und drückt aus, dass Objekte der Klasse B nur dann existieren, wenn auch das zugehörige Objekt der Klasse A existiert




```
public class Paar {
    private Person[] personen;

    public Paar(Person a, Person b){
        personen = new Person[2];
        personen[0] = a;
        personen[1] = b;
    }
}
```

Klassendiagramm zur Klasse Paar




```
public class GeometrischeFigur {...}
public class Kreis extends GeometrischeFigur {...}
public class Dreieck extends GeometrischeFigur {...}
public class Rechteck extends GeometrischeFigur {...}
```

Klassendiagramm mit Vererbungsbeziehung

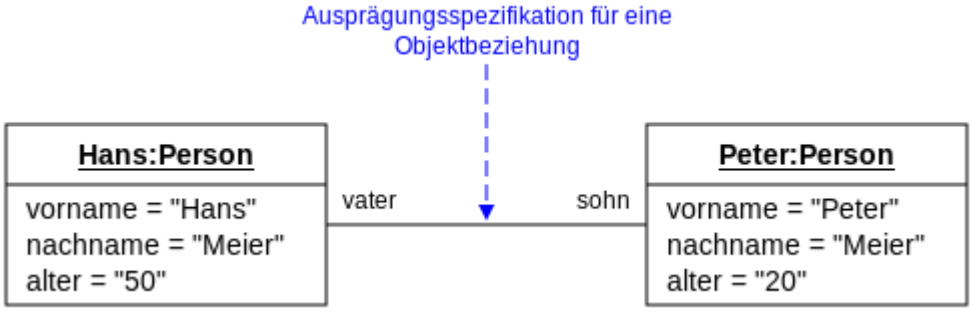
2.2 Aufgabe: Essen im Gasthaus

In diesem kleinen Städtchen gibt es fünf Restaurants, aber egal, ob es sich um ein 2**-Restaurant, oder um ein 5*****-Restaurant handelt, alle haben Gemeinsamkeiten. In jedem Restaurant sind mehrere Mitarbeiter angestellt. Im kleinsten Restaurant nur zwei, aber im grössten verdienen 16 Mitarbeiter ihr Geld. Die Mitarbeiter bekommen in jedem Restaurant ein anderes Gehalt, aber alle machen ihre Arbeit. Die Kellner nehmen Bestellungen entgegen, servieren das Essen und kassieren das Geld von den Gästen. Jeder Kellner gibt die Bestellungen der Gäste an die Köche weiter. Die Köche kennen verschiedene Rezepte, nach denen sie die Menüs dann zubereiten. Jedes Menü hat einen anderen Preis, aber es besteht nach Tradition immer aus einem Salat und einer Hauptspeise. An manchen Tagen ist kein Gast da. Dann werden Sie in ganz familiärem Rahmen bewirtet. Aber auch an guten Tagen gibt es, bei 100 Plätzen im grössten Restaurant, sicher für jeden hungrigen Gast einen Platz. Wer in Gasthausen Essen geht sollte hungrig sein! Natürlich hat jeder Gast unterschiedlich viel Geld zur Verfügung, aber bei entsprechender Wahl des Restaurants reicht es sicher für sein Lieblingsessen.

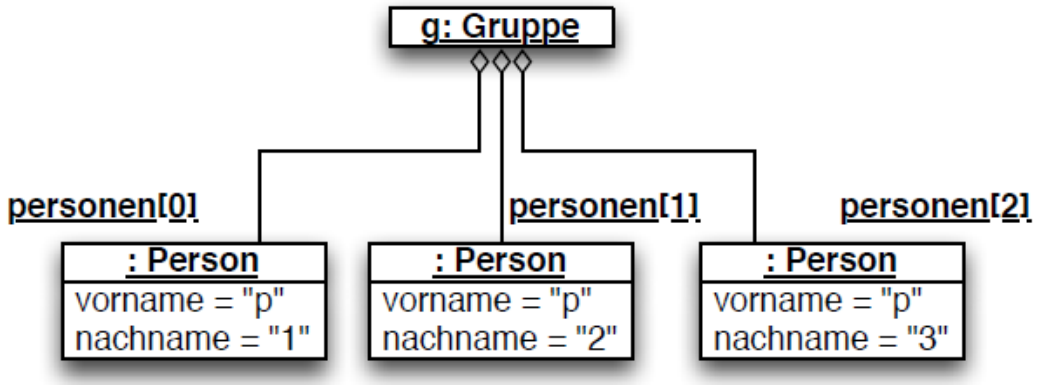
3 Objektdiagramme

3.1 Anwendungsbereich

Objektdiagramme werden eingesetzt um einen Schnappschuss eines Systems zur Laufzeit darzustellen. Es werden die zur Laufzeit existierenden Objekte, Attribute und Beziehungen angegeben.



Vater Sohn Beziehung zwischen zwei Personen



Ein mögliches Objektdiagramm zur Klasse Gruppe

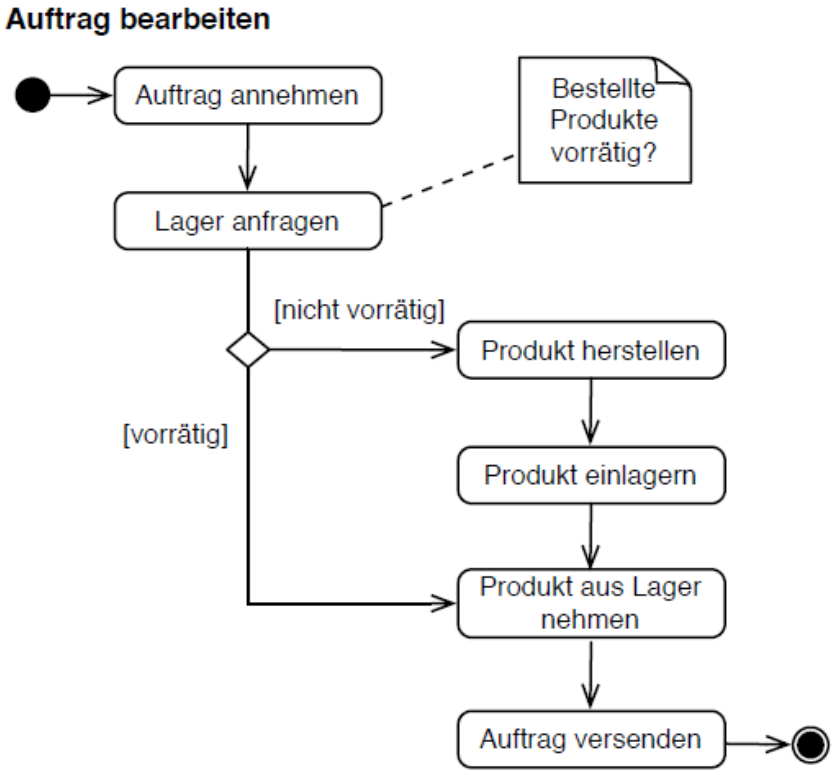
3.2 Aufgabe: François der bestbezahlte Kellner

Peter, der gern Gast in einem kleinen Fischrestaurant um die Ecke ist, hat heute seinen 21-ten Geburtstag. Er hat beschlossen seine 3 besten Freunden Karl, Moritz und Vanessa in sein Lieblingsfischrestaurant einzuladen um diesen Tag mit ihnen zu feiern. Peter mag dieses Restaurant so sehr, weil es ein kleines, gemütliches 3 Sterne Restaurant ist, in dem nur etwa 50 Gäste Platz haben. Es heisst „Zum glitschigen Hering“. Das Tolle an diesem Restaurant ist, dass der Gast den Status eines Königs hat. Das erzählt Peter jedenfalls immer seinen Freunden. Eine weitere spektakuläre Besonderheit des Restaurants ist, dass es nur einen Kellner gibt. Sein Name ist „Francois“, er ist Franzose und sehr penibel. Keiner der ihm je zur Unterstützung dienen sollte, war ihm gut genug und so kam es, dass Francois nach und nach lernte alle Tische gleichzeitig zu bedienen. Dazu stapelte er die Teller immer höher und höher fast so hoch, dass er irgendwann nicht mehr durch die Türen passte. Jeder Gast, der 15 Franken bei sich hat und Hunger mitbringt, kann sich von diesem Spektakelverzaubern lassen. Francois steht schon seit einer Weile im Guinnessbuch der Rekorde, aber nicht, weil er sich 50 Bestellungen gleichzeitig merken kann, sondern weil er mit 10.000 Euro der bestbezahlte Kellner der Welt ist.

4 Aktivitäten Diagramme

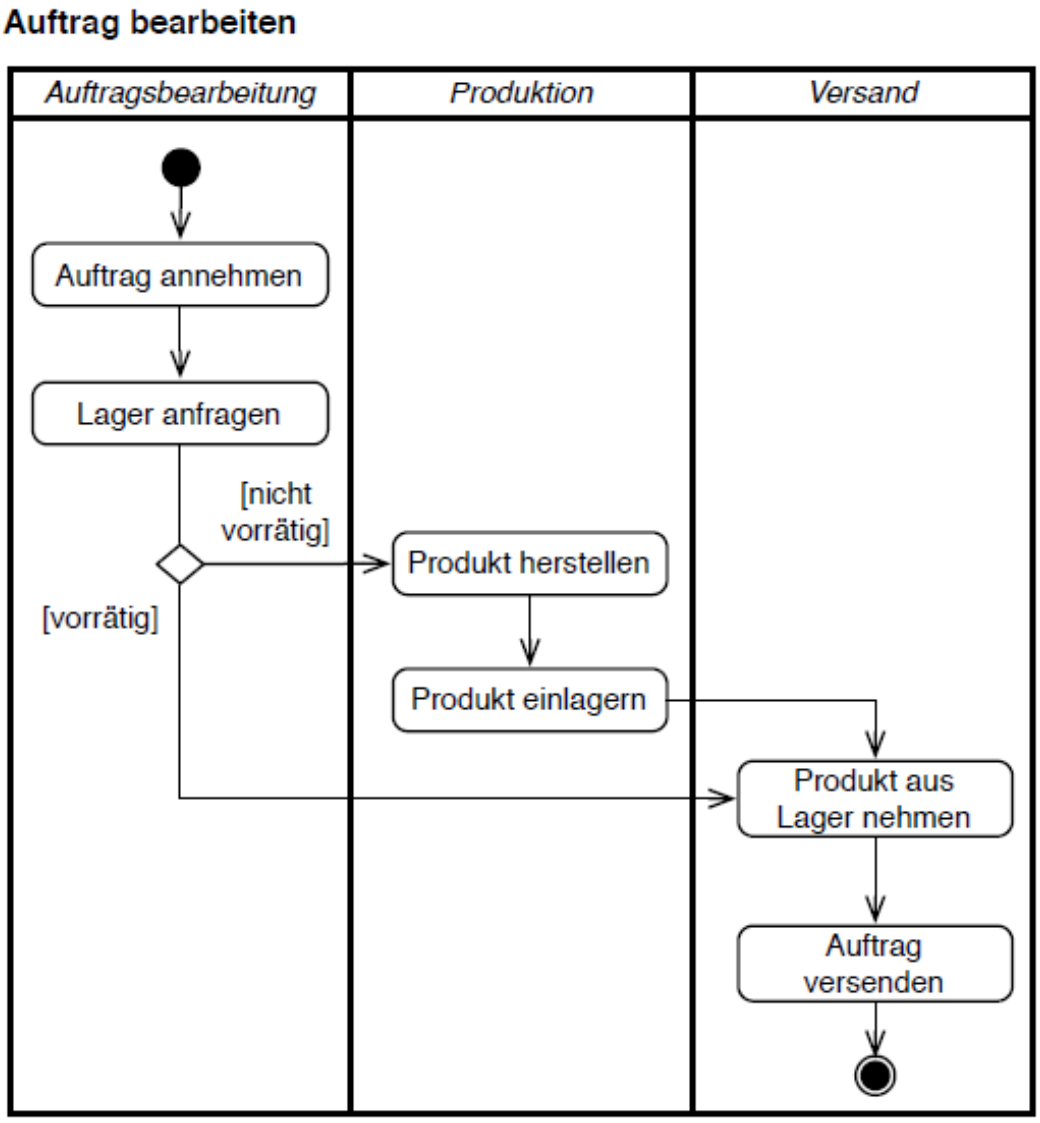
4.1 Anwendungsbereiche

Das Aktivitätsdiagramm zeigt einen Ausschnitt eines Programmablaufs. Aktivitätsdiagramme stellen im Prinzip eine Erweiterung von Flussdiagrammen dar.



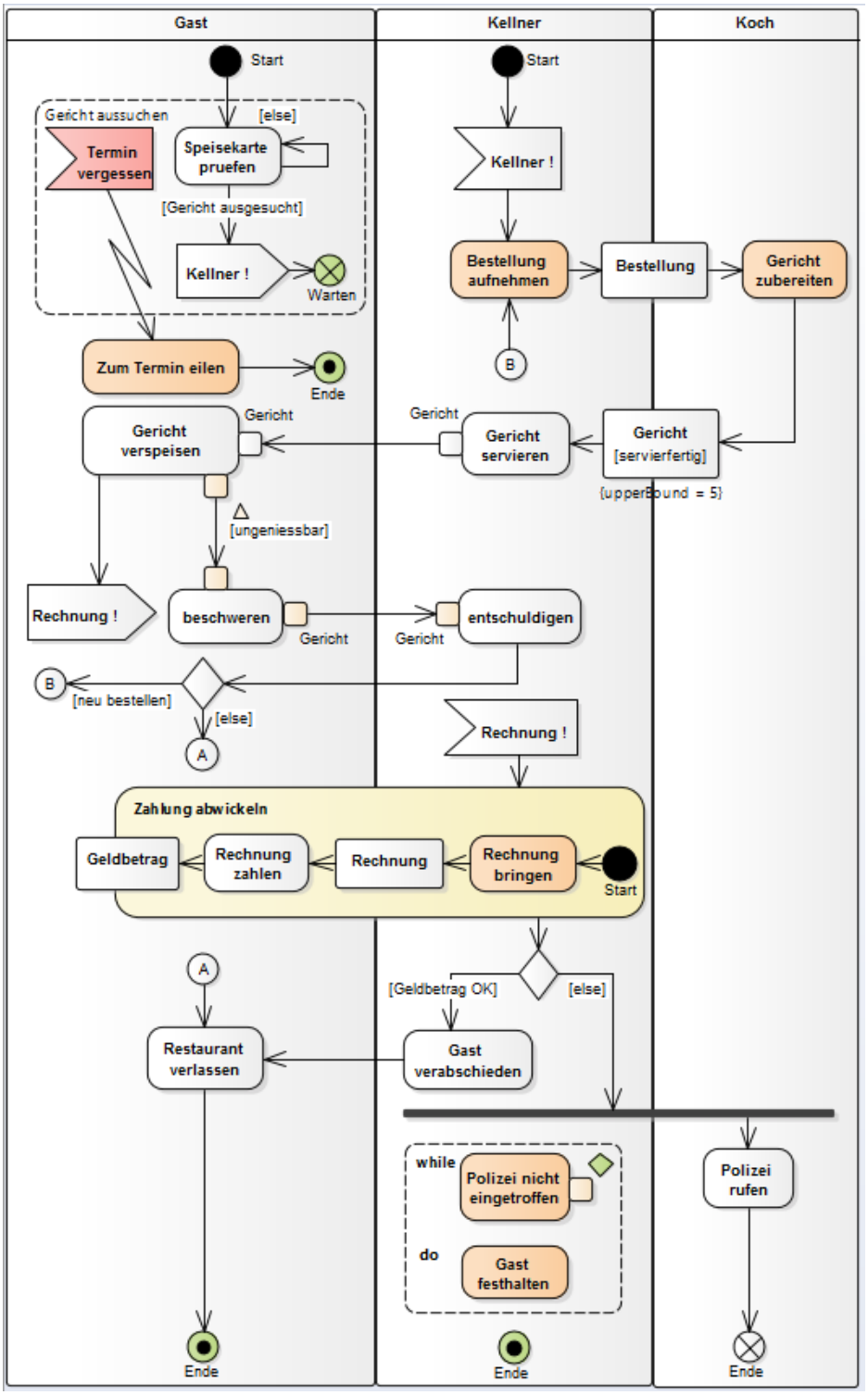
Aktivitätsdiagramm einer Versandfirma

Eine Erweiterung ist z.B. die Möglichkeit paralleles Verhalten oder Verantwortlichkeiten darzustellen. Will man die Verantwortlichkeit für bestimmte Aktivitäten ausdrücken, kann man **Aktivitätsbereiche** (Swim Lanes) einsetzen. Im folgenden Beispiel: Auftragsbearbeitung; Produktion; Versand



Aktivitätsdiagramm mit Aktivitätsbereichen einer Versandfirma

Parallele Aktivitäten synchronisieren



Der **Kellner** wartet bis der **Gast** das Gericht ausgesucht hat und den Kellner zur Bestellungsaufnahme ruf.
Einzelne detaillierte Aktivitäten können in einer eigenen Aktivität zusammengefasst werden. Siehe Beispiel: ***Zahlung abwickeln***

4.2 Aufgabe: Karte laden in der Berufsschule Sursee

In der Kantine der Berufsschule Sursee wird aus hygienischen Gründen mit bargeldlosen speziellen Karten bezahlt. Es ist allerdings Voraussetzung, dass auf diesen Karten immer ein genügender Geldbetrag gespeichert ist.

Bei der Bezahlung des Essens kann die Frau an der Kasse sehr einfach den Essensbetrag von der Karte abbuchen. Damit das reibungslos funktioniert, kann der Kunde selbständig am Eingangsautomat überprüfen, ob noch genügend Geld auf der Karte vorhanden ist und allenfalls die Karte wieder mit Bargeld aufladen.

Er schiebt die Karte in dem Automaten, anschliessend wird der noch vorhandene Betrag auf der Karte angezeigt. Ist zu wenig darauf, kann er die Karte mit einem gewünschten Betrag aufladen.

Beschreiben Sie den Ablauf. **Überprüfen und Laden der Karte** mit einem Aktivitätsdiagramm, bedenken Sie, dass nicht mit allen Geldscheinen bezahlt werden kann, und dass die Scheine geprüft werden müssen. Ein Bezahlen mit Kreditkarten ist bis jetzt nicht möglich.

5 Zustandsdiagramme

5.1 Anwendungsbereich

Zustandsautomaten beschreiben das Verhalten der Elemente während ihres Lebenszyklus durch Darstellung der möglichen Zustände und Zustandsübergänge.

Einfacher Zustand (simple state)

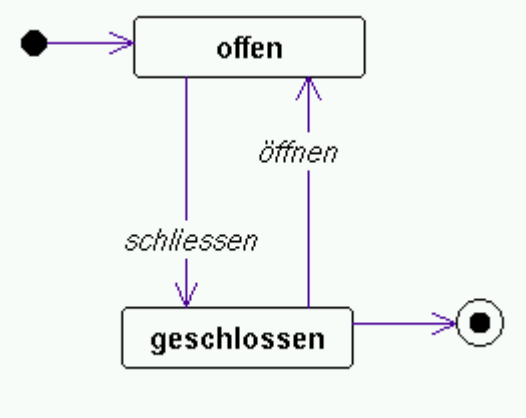
Ein einfacher Zustand hat keine Unterzustände. Ein einfacher Zustand kann entry, exit und do-Aktivitäten enthalten und interne Transitionen. Nach Erreichen des Zustands wird sofort die entry-Aktivität ausgeführt. Vor Verlassen des Zustands wird die exit-Aktivität ausgeführt. Die do-Aktivität wird nach Beendigung der entry-Aktivität ausgeführt.

Zustandsübergang (Transition)

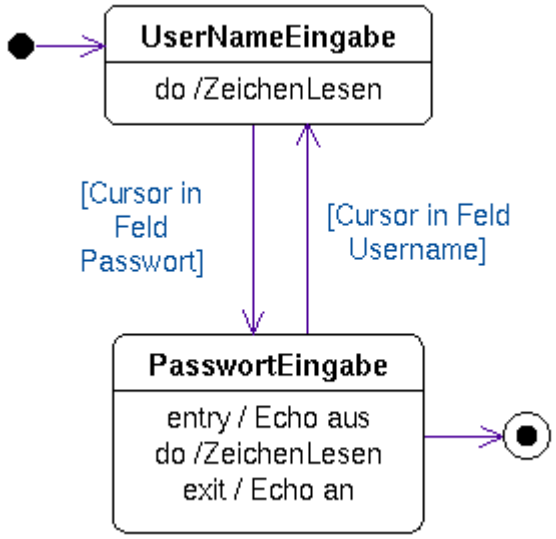
Eine Transition wird durch einen Auslöser (Trigger) verursacht. Der Trigger kann intern sein oder extern; er kann von dem Objekt kommen, auf das die state machine sich bezieht, oder von einem anderen Objekt. Für einen Zustandsübergang kann eine Bedingung (guard) angegeben werden, die erfüllt sein muss, damit die Transition ausgeführt wird. Bei einer Transition kann ein Objekt eine Aktivität ausführen, die zu dem Aufruf einer Methode führt.

Einfacher Zustandsautomat

Zustandsautomaten beschreiben den Lebenszyklus von Objekten, können sich aber auch auf komplexere Elemente beziehen, wie z. B. Komponenten, Subsysteme oder Use Cases. Sie stellen die Zustände dar, die Objekte, Komponenten usw. im Laufe ihrer Existenz annehmen können und sie geben die Zustandsübergänge an, die möglich sind. Ein Zustandsautomat gibt auch an, in welchem Zustand ein Objekt auf welche Ereignisse reagiert.



Diese Abbildung stellt einen einfachen Zustandsautomaten einer Tür dar. Die Tür kann die beiden Zustände offen und geschlossen annehmen. Sie reagiert auf die Ereignisse schliessen und öffnen. In dem Zustand offen kann jedoch nur das Ereignis schliessen angewandt werden, in dem Zustand geschlossen kann nur das



Dieses Beispiel modelliert eine Loginmaske eines Rechners. Sie hat die Zustände "UserNameEingabe" und "PasswortEingabe". In dem Zustand "UserNameEingabe" führt sie die Aktivität "ZeichenLesen" aus. Ist die Bedingung "[Cursor in Feld Pass-wort]" erfüllt, so wechselt sie zu dem Zustand "PasswortEingabe". Sobald die Mas-ke den Zustand betritt führt sie die entry-Aktivität "Echo aus" aus, sie schaltet das Echo aus und zeigt das Passwort nicht auf dem Bildschirm an. Danach liest sie mit der Aktivität "ZeichenLesen" das Pass-wort ein und schaltet bei Verlassen des Zu-stands mit der exit-Aktivität "Echo an" das Bildschirmecho wie-der ein.

5.2 Auftrag: Restaurant fleissige Bienchen

Max und Sarah haben ein Ziel. Jeder Gast, der ihr Restaurant, das „fleissige Bienchen“ besucht, soll so schnell wie möglich satt werden. Um das zu erreichen arbeiten sie auf Hochtouren. Wenn ein Gast sein Es-sen bestellt, wird das sofort in der Küche gehört. Max fängt direkt an das Gericht zuzubereiten. Man hat es noch nicht bewiesen, aber wahrscheinlich ist er der schnellste Koch der Welt und dementsprechend dauert das Kochen nur wenige Minuten. Ist das Gericht fertig rennt Sarah zu den Gästen und serviert es. Manchmal verliert sie dabei, weil sie so schnell rennt, Teile des Essens. Dann muss sie zurück in die Küche, in der das Gericht neu zubereitet wird. Wenn aber alles gut gegangen ist, kann der Gast schnell essen. Gäste mit klei-nem Hunger sind nach dem Essen satt. Hungrige Gäste bestellen noch einen Nachtisch, der schon bereitsteht, damit keine Zeit verloren geht. Ist dieser aufgegessen sind auch die hungrigsten Gäste satt. Sarah und Max haben herausgefunden, dass die Gäste beim Bezahlen der Rechnung viel mehr Trinkgeld geben, wenn sie innerhalb von einer halben Stunde satt sind. Aber egal, wie viel Geld sie bekommen. Max und Sarah sind glücklich über jeden Gast, der sie besucht.

6 Sequenzdiagramme

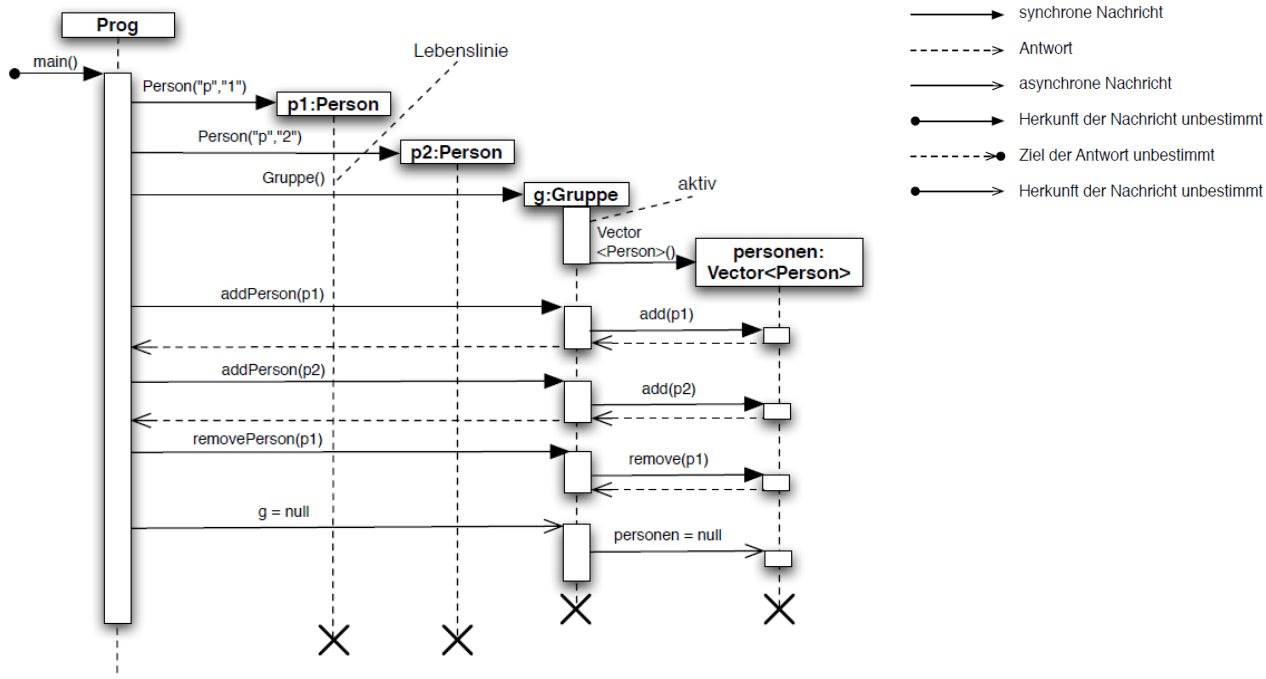
6.1 Anwendungsbereich

Das Sequenzdiagramm stellt den zeitlichen Ablauf kommunizierenden Objekte untereinander dar. Dabei werden nur die von besonderem Interesse stehen Objekte dargestellt.

Es ist die aktive Form des Objektdiagramms, d.h. es wird nicht nur der momentane Zustand der Objekte abgebildet, sondern ein ganzer Ablauf. Die Zeit läuft von oben nach unten. Sequenzdiagramme dürfen als Beteiligte Objekte und weitere Akteure beinhalten.

```
public class Prog{
  public static void main(String args[]) {
    Person p1 = new Person("p","1");
    Person p2 = new Person("p","2");
    Gruppe gruppe = new Gruppe();
    gruppe.addPerson(p1);
    gruppe.addPerson(p2);
    gruppe.removePerson(p1);
  }
}
```

Beispielhafte Verwendung der Klasse Gruppe



Sequenzdiagramm zur Verwendung der Klasse Gruppe

Die Beteiligten kommunizieren über Nachrichten. Sie werden durch ein Rechteck mit Beschriftung, an dem unten eine vertikale, gestrichelte Linie angebracht ist, dargestellt. Die Gesamtheit aus Rechteck und gestrichelter Linie wird als Lebenslinie bezeichnet.

6.2 Aufgabe: Pizzeria in Verona

Mario betreibt mit seinem Bruder Luigi eine kleine Pizzeria in Verona. Mario, der von sich selbst glaubt der besser Pizzabäcker zu sein, schmeisst als Koch die Küche und Luigi ist als Kellner im Service tätig. Das funktioniert sowieso besser, wenn man bedenkt, dass Mario fast das Doppelte auf die Waage bringt als Luigi. Jeder hungrige Gast, der nach einem Kellner ruft und das Glück hat, von Luigi bedient zu werden, kann sich auf einen kulinarischen Hochgenuss freuen. Luigi gibt zwar die Bestellung an seinen Bruder Mario weiter, doch was die wenigsten wissen, ist, dass Luigi die Pizza die sein Bruder Mario bäckt, vor dem Servieren mit den erlesensten Kräutern aus ganz Italien verfeinert. Und so kommt es, dass wenn die Gäste wieder mal zufrieden zu Luigi sagen, er solle die Küche grüssen und loben, was für ein ausgezeichnete Pizza es hier doch gibt, freut er sich nur bescheiden und sagt mit einem Grinsen auf den Lippen, dass er das Lob gerne weitergebe.

7 Anhang (Lösungen und Hinweise)

Diese Lösungen erheben keinen Anspruch auf Vollständigkeit. Der Lernende sollte zuerst selber versuchen die Aufgabenstellungen zu lösen. Diese hier vorgestellten Lösungen zeigen vielmehr einen möglichen Lösungsansatz.

Fahrzeug ausleihen (Use Case)

Zusammenfassung:

Ein Kunde kommt leiht sich ein Fahrzeug und fährt damit an den gewünschten Ort. Der Ausleihungsvertrag wird unterschrieben und das Fahrzeug dem Kunden ausgehändigt.

Actors: Ausleihungsbeamter, Kunde

Vorbedingugen: Das gewünschte Auto wurde vom Kunden vorbestellt

Beschreibung ("Sunny Day Scenario"):

1. A customer comes to the office to acquire a vehicle.
2. The clerk locates the vehicle reservation contract by means of the reservation number and/or customer name. [Exception: Required vehicle type is not available due to late arrivals.]
3. The customer signs the contract and the clerk gives the keys to the vehicle.
4. The clerk then marks the contract active by entering the vehicle release date (today's date) onto the vehicle reservation contract. The use case terminates at this point.

Ausnahmen ("Rainy Day Scenario"):

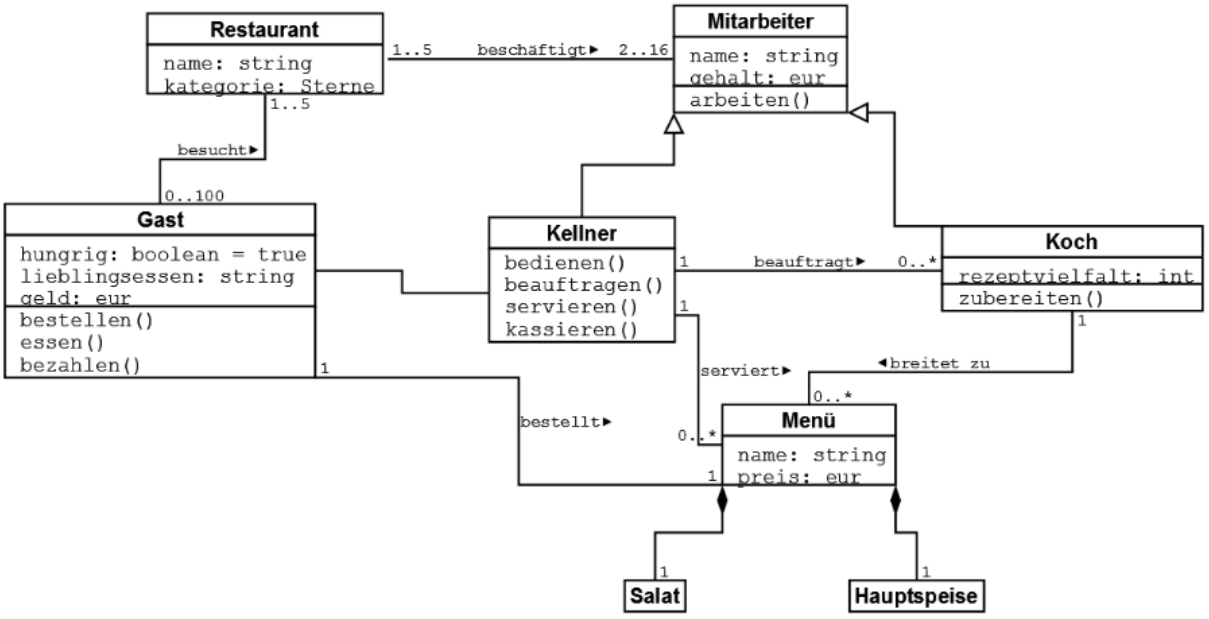
Required vehicle type is not available due to late arrivals:

Raised when the reserved vehicle is not available due to late returns. The customer is informed of the situation and told about the other vehicle types that are available. The customer is offered an incentive to accept another vehicle type. If the customer is not satisfied, the reservation is cancelled without penalty charges. The customer either accepts another vehicle type or cancels the reservation.

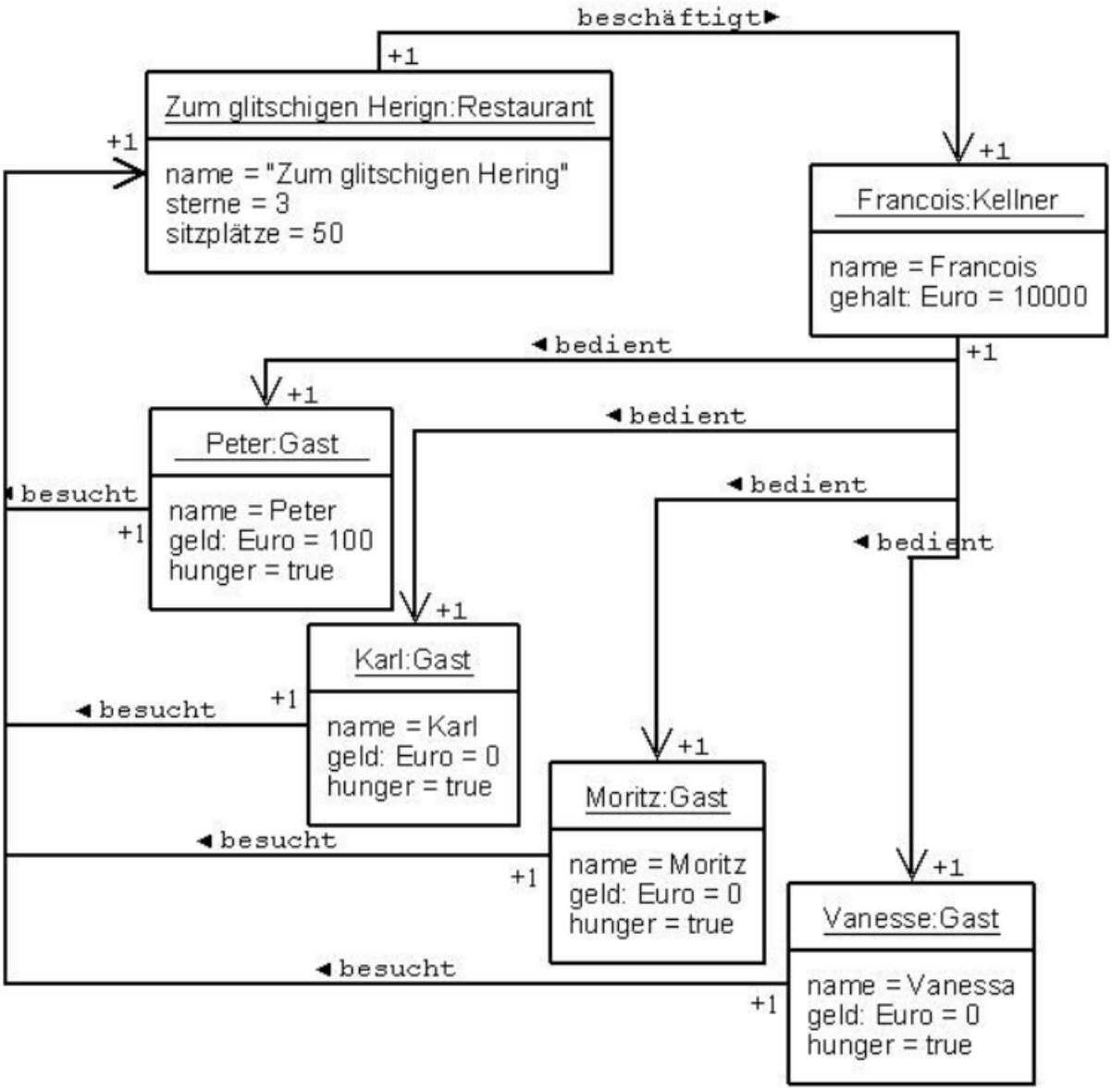
Nachbedingungen:

Der Kunde fährt mit dem Auto weg und der Ausleihevertrag ist aktiviert, oder die Reservation wird gelöscht.

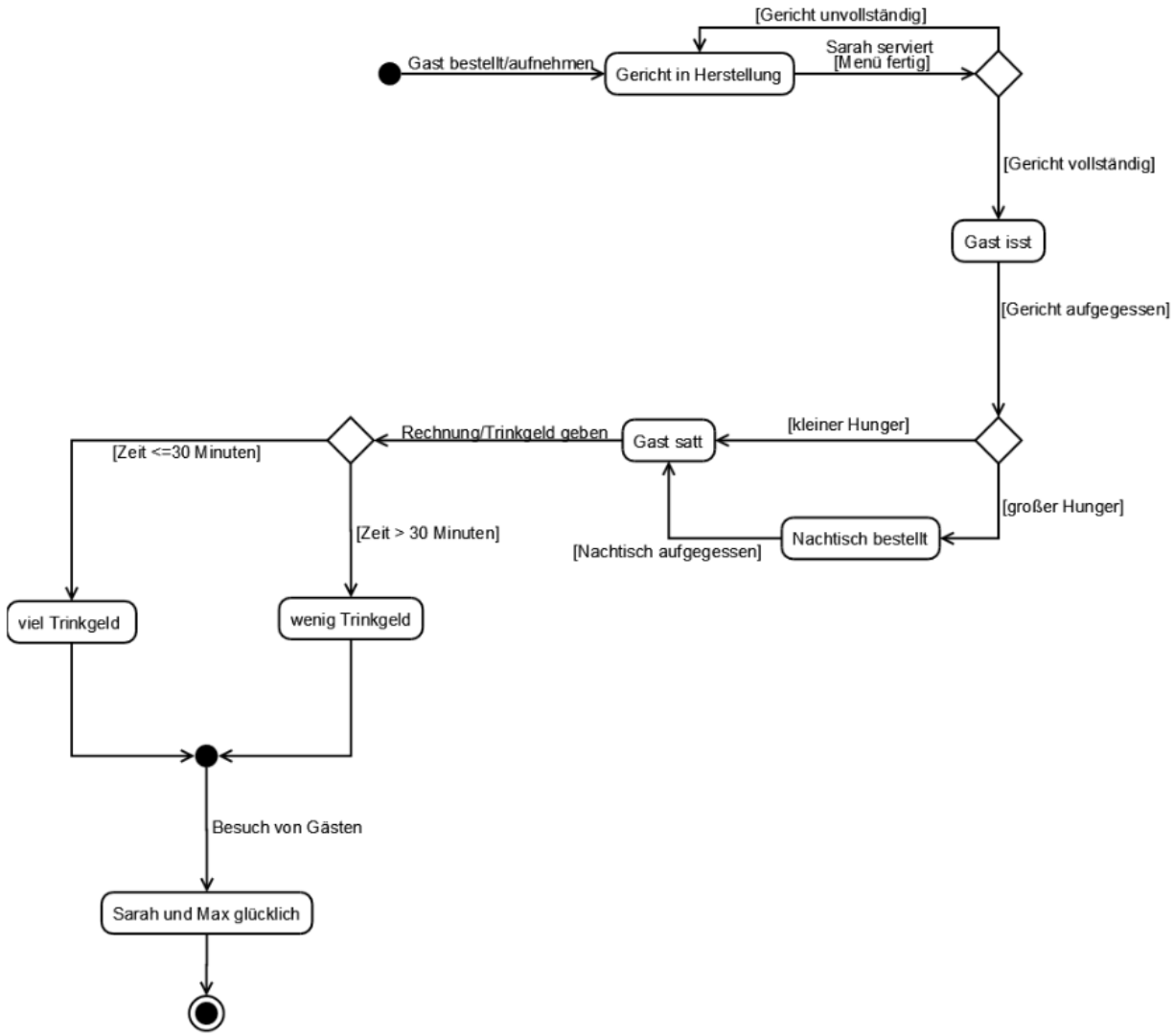
Essen im Restaurant (Klassendiagramm)



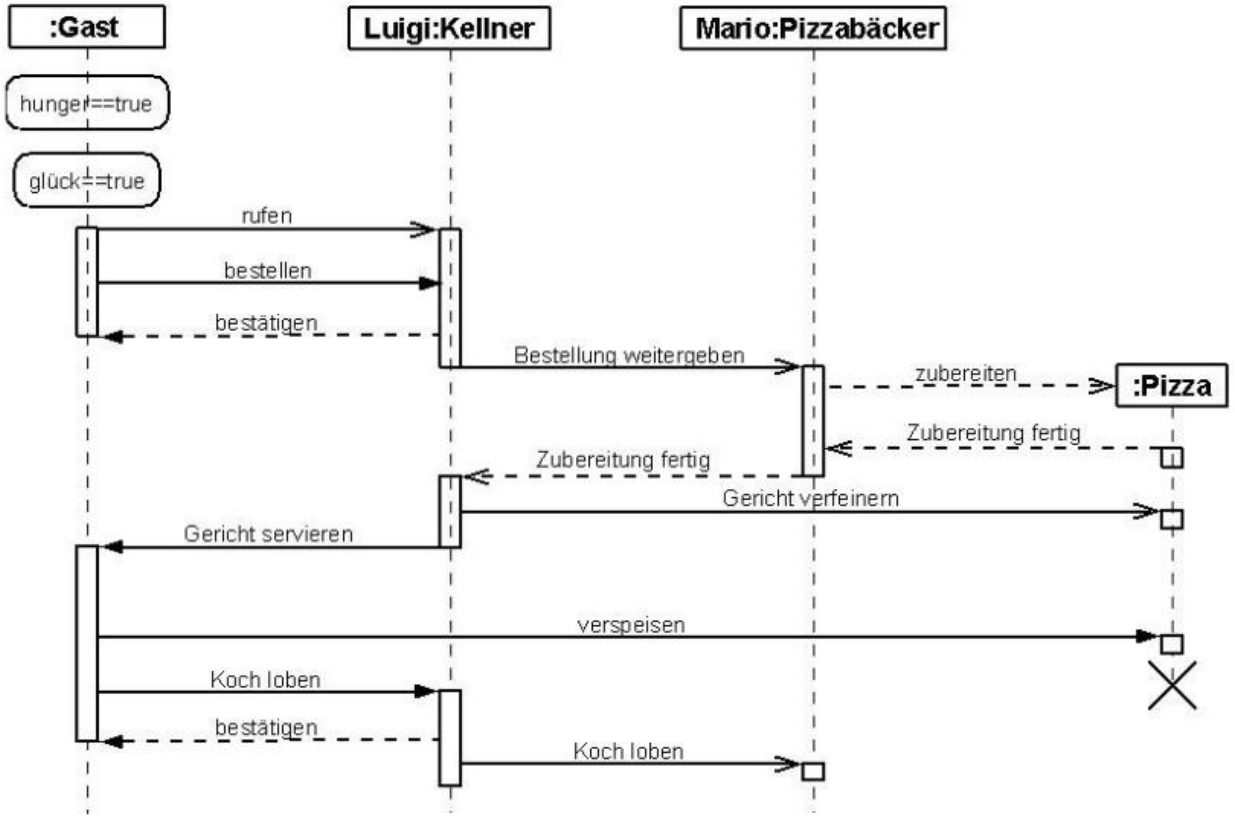
François der bestbezahlte Kellner (Objektdiagramm)



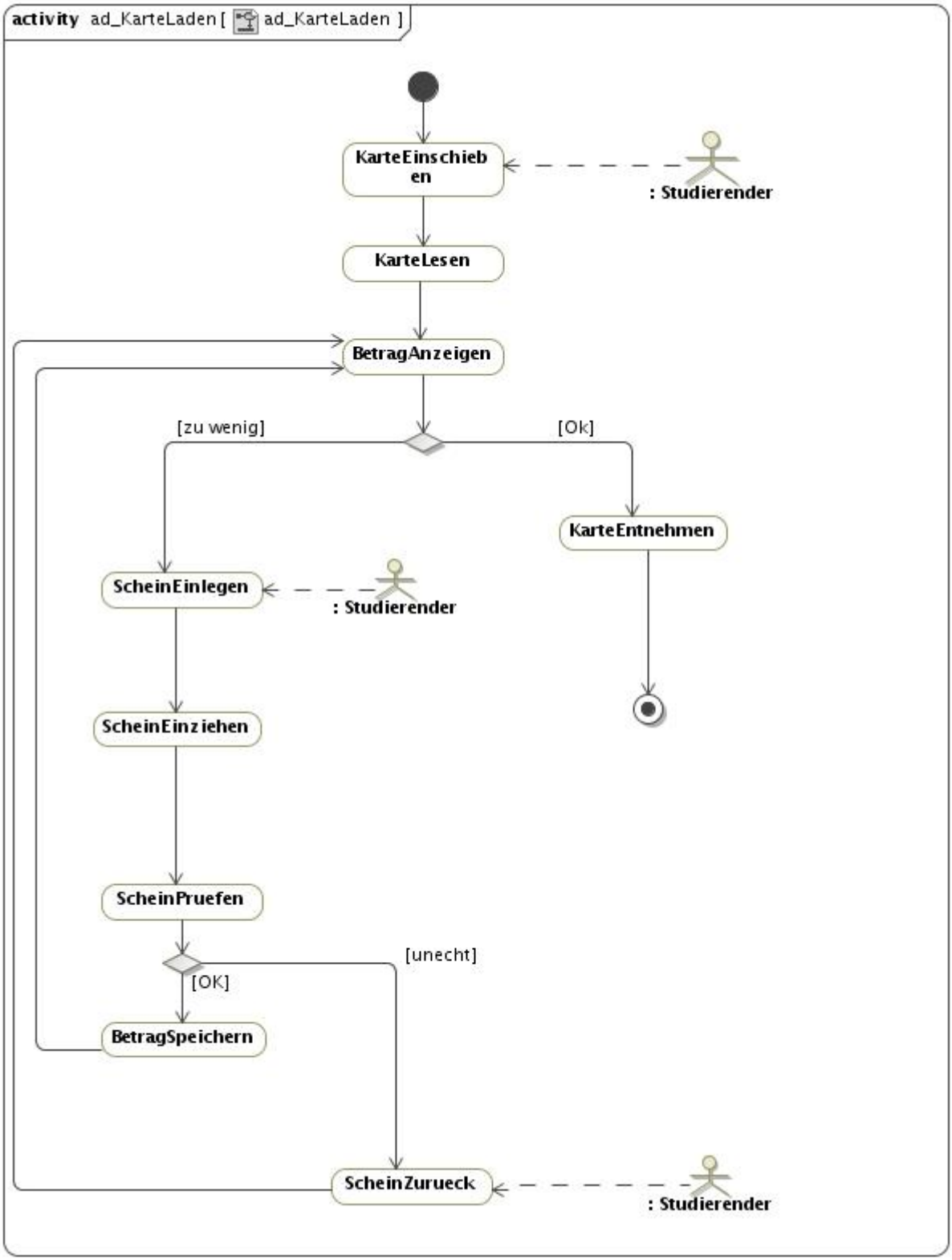
Restaurant fleissige Bienchen (Zustandsdiagramm)



Pizzeria in Verona (Sequenzdiagramm)



Karte laden (Aktivitätsdiagramm)



Hinweise:

https://de.wikipedia.org/wiki/Klassendiagramm

https://de.wikipedia.org/wiki/Objektdiagramm

https://de.wikipedia.org/wiki/Sequenzdiagramm

https://de.wikipedia.org/wiki/Zustandsdiagramm

UML_Zeichnungstool:

https://www.visual-paradigm.com/shop/vp.jsp?license=perpetual

http://alexdp.free.fr/violetumleditor/page.php

http://argouml.tigris.org/ (Standard ist nicht aktuell)

https://www.draw.io/ (online)

Quellen:
(Teile von: https://de.wikiversity.org/wiki/Kurs:Fachdidaktik_Informatik/ldl_uml)

Modeling-business-systems

https://sourcemaking.com/uml/modeling-business-systems

UML 2.5
Das umfassende Handbuch
von Christoph Kecher, Alexander Salvanos, Ralf Hoffmann-Elbern

450 Seiten, gebunden,
E-Book Formate: PDF, EPUB, MOBI, Online

€ 34,90 Sofort lieferbar
Buch | E-Book | Bundle

- UML lernen und effektiv in Projekten anwenden
- Alle Diagramme und Notationselemente im Überblick
- Mit zahlreichen Praxisbeispielen in Java und C#
- Programmierkenntnisse erwünscht