	UML – Use Case Diagramm		AnPr
	Name	Klasse	Datum

1 Allgemeines

Jedes System soll eine Lösung für bestimmte Problemstellungen des Users bieten. Das Usecasediagramm (oder auch Anwednungsfalldiagramm) wurde dafür geschaffen, aus Usersicht die Interaktionsmöglichkeiten eines Systems auf einem relativ hohen Level zu modellieren. Hierbei soll gezeigt werden:

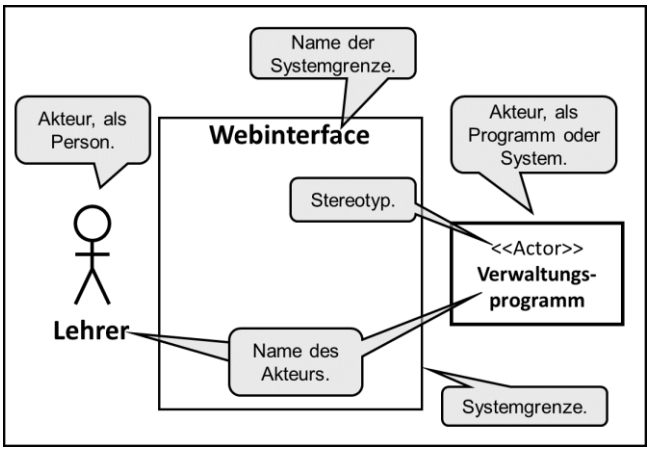
- wo die Systemgrenzen liegen, um systeminterne systemexterne Elemente voneinander unterscheiden zu können
- welcher Akteur welche Prozesse anstoßen kann
- und welche Prozessschritte wie zusammenhängen und ggf. mehrfach verwendet werden können

Das Use Case Diagramm wird relativ früh im Prozess der Softwareentwicklung erstellt. Von dort aus werden mit Hilfe von weiteren UML Diagrammen das System Schritt für Schritt weiter detailliert.

2 UML-Use Case Diagram, Notation

Die wichtigsten Elemente des Use Case Diagramms.

2.1 Akteur und Systemgrenze

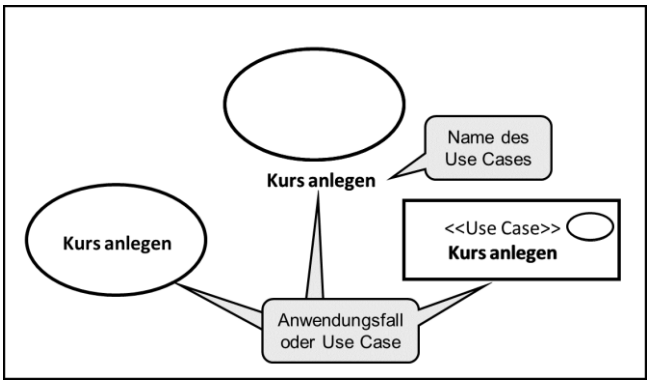


Das zu modellierende System muss sich von den Akteuren abgrenzen, weshalb man die Akteure **immer außerhalb** der Systemgrenze zeichnet.

Bei den Akteuren unterscheidet man die **Person** (im Sinne eines „Menschen“) und **Systemakteure**, meist Programme. Letztere zeichnet man als Rechteck mit einer Stereotypbezeichnung „<<Actor>>“. Beide tragen einen Namen, der den Akteur sinnvoll einordnet.

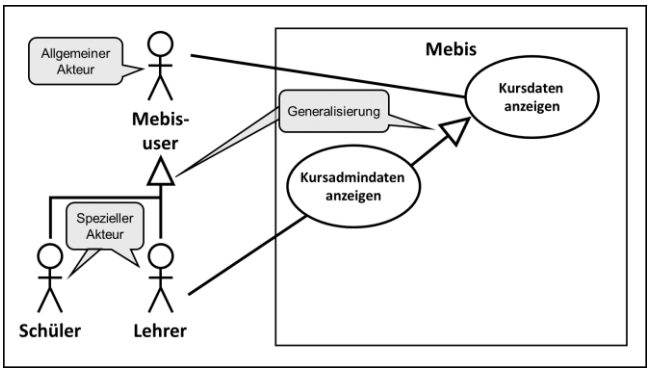
Es darf mehrere Personen – Akteure und mehrere Systemakteure geben, jedoch sollten die Namen einmalig bleiben. Weiterhin ist es erlaubt, Akteure mit einem individuellen Bild (bspw. Firmenlogo) zu zeichnen.

2.2 Anwendungsfall



UML erlaubt drei verschiedene Notationen eines Anwendungsfalls, wobei die linke Notation mit dem Anwendungsfallnamen innerhalb der **Ellipse** die am häufigsten verwendete ist.
Der Name sollte die Intention des Akteurs wiederspiegeln, etwas mit dem System zu tun.

2.3 Generalisierung

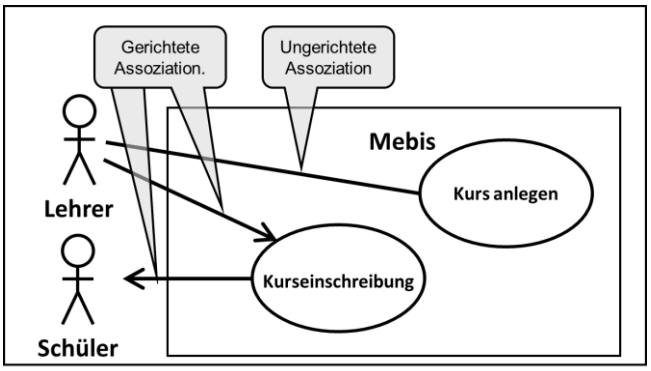


Wie im Klassendiagramm auch, können wir Eigenschaften – bzw. im Fall vom Use Case Diagramm Interaktionsmöglichkeiten **vererben** bzw. sie zu „Generalisieren".

Dies ist sowohl bei den Akteuren, als auch bei den Use Cases möglich. Hier gilt, alles das, was der allgemeine Akteur darf, darf auch der spezielle, aber nicht umgekehrt.

Bei den Use Cases ist die Vererbung so zu sehen, wie im Klassendiagramm.

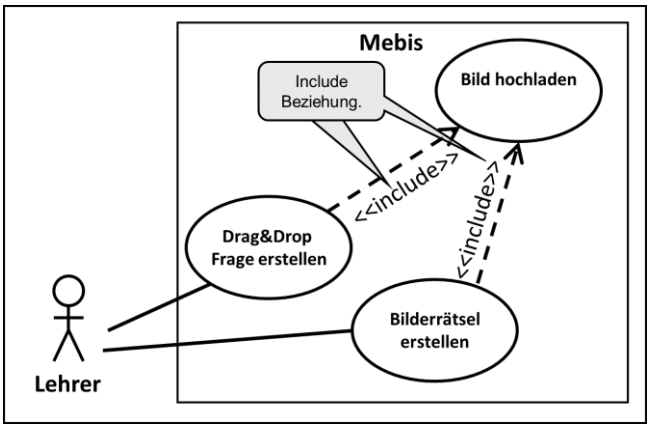
2.4 Assoziation



Assoziationen sind **Interaktionsmöglichkeiten** zwischen Akteuren und Use Cases. Hiermit wird also festgelegt, welcher Use Case von welchem Akteur angelegt wird, bzw. in welchen Use Cases der Akteur eine Rolle spielt – sprich aktiv ist. So ist bspw. beim Kurs Anlegen nur der Lehrer gefragt. Bei der Kurseinschreibung legt der Lehrer den Schüler im Kurs an, welcher dann lediglich vom System informiert wird.

Hierbei können wir die Assoziationen auch **gerichtet** ausführen. Dies bedeutet, dass wir festlegen können, wer den Use Case antriggert und wie der Informationsfluss zu gestalten ist. **Ungerichtete** Assoziationen wiederum erlauben Informationsflüsse in beide Richtungen.

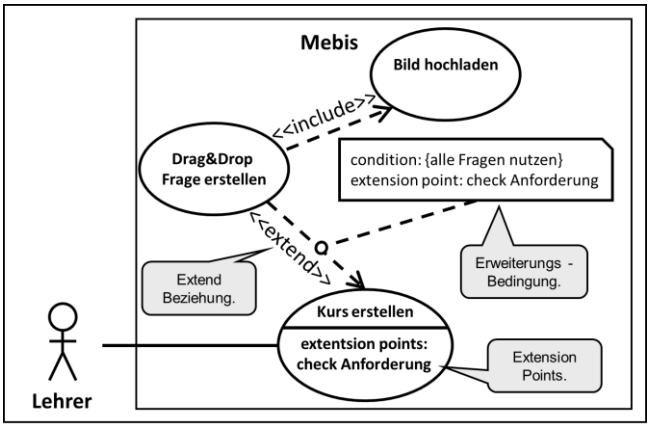
2.5 Include Beziehung



Use Cases können in einzelne Untercases (vergleichbar mit Unterprogrammen) aufgeteilt werden. Wenn ein Use Case einen zweiten **zwingend** benötigt, er also ohne den zweiten nicht vollständig ist, so wird er über eine include Beziehung angebunden. Diese wird mit einer gestrichelten Pfeillinie vom nutzenden zum angebundenen Use Case gezeichnet und mit dem Stereotypen „<<include>>" versehen.

Man kann auch mehrere Use Cases einen Use Case nutzen lassen. Zirkelbezüge sind hier absolut zu vermeiden – dies gilt genauso wie bei Zirkelbezügen in Programmen.

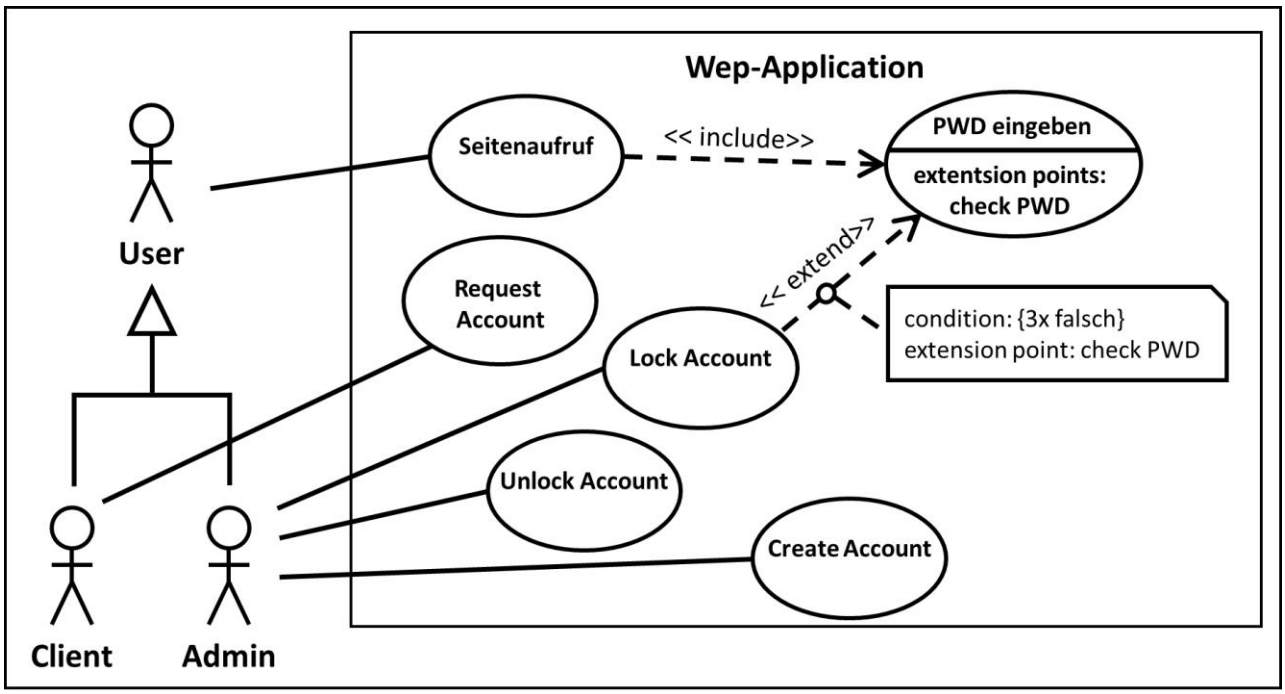
2.6 Extend Beziehung



Optional einzubindenden Use Cases werden mit der Extend Beziehung modelliert. Hierbei geht der gestrichelte Pfeil von dem eingebundenen Use Case zum einbindenden! Weiterhin wird der Stereotyp „<<extend>>" notiert.
Um die Situation einer möglichen Einbindung zu dokumentieren, werden unter dem Use Case Namen die Erweiterungspunkte („Extension Points") eingetragen, welche die Situation darstellen, in der die Entscheidung getroffen wird.

Zusätzlich können wir noch eine Erweiterungsbedingung notieren, in der die eigentliche Bedingungsprüfung hinterlegt ist, so dass klar ist, unter welcher Voraussetzung der erweiterte Use Case einzubinden ist. Dies wird jedoch mitunter weggelassen.

3 Beispiel

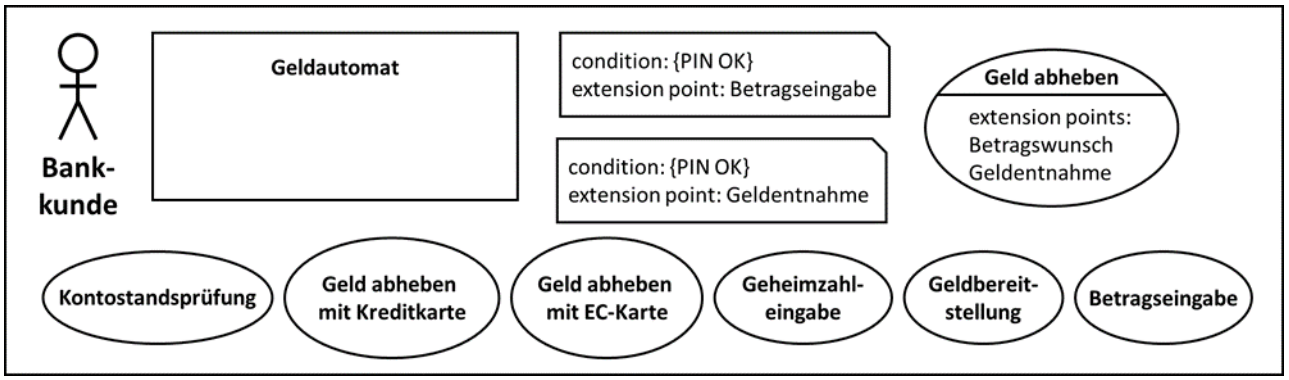


.

4 Aufgabenstellung

Zeichnen Sie ein UML Usecase Diagramm für den Prozess einer ATM (Automated Teller Machine) einr Bank. Hierbei soll der User mit EC(bzw. Bank-) oder Kreditkarte Geld abheben können oder seinen Kontostand anzeigen.

Folgende Elemente existieren – Sie ergänzen dazu die Assoziationen, Erweiterungs- und Include Beziehungen und die Generalisierungen:



5 Lizenz