Technology
Arts Sciences
TH Köln

ST1

Softwaretechnik 1 (ST1) im SoSe 2022
Objektorientierte Modellierung und Entwicklung

Kapitel 1: Einführung in die Modellierung, UML, Funktionsmodellierung

Lernziele: Nach dieser Vorlesung sollten Sie ...

- Wissen, was ein **Modell** ist und welche Eigenschaften gute Modelle haben
- Die drei wesentlichen Modellierungssichten **Funktion**, **Struktur** und **Verhalten** bei der Entwicklung von Software kennen und voneinander abgrenzen können
- Wissen, dass die UML eine **Sprache**, nicht aber eine Methode zur Modellierung in der Softwareentwicklung darstellt
- Verstehen, was die Modellierungskonzepte **Akteur** und **Anwendungsfall** darstellen und wozu es sie in der UML gibt
- Die Funktionalität einfacher interaktiver Systeme mit **Anwendungsfall-diagrammen** skizzieren und **textuell** spezifizieren können
- Die **include-Beziehung** und die **extend-Beziehung** sowie **Generalisierungen** zwischen Anwendungsfällen modellieren und voneinander abgrenzen können
- Mit **Aktivitätsdiagrammen** die Ablaufmöglichkeiten von Anwendungsfällen modellieren können

Überblick

- **Modellierung und UML**
- Akteure und Anwendungsfälle
- Beziehungen zwischen Anwendungsfällen
- Aktivitätsmodellierung

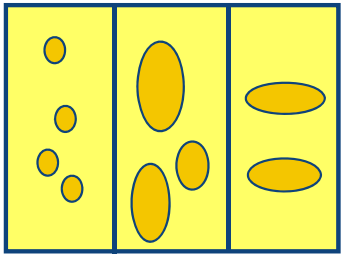
Zur Erinnerung: Warum Softwaretechnik für große Software?

- 1538 Klassen im Java JDK 1.4.2
- 8032 compile-time dependencies in den java.* packages
- Aber: Menschliches “Arbeitsgedächtnis” 7 ± 1

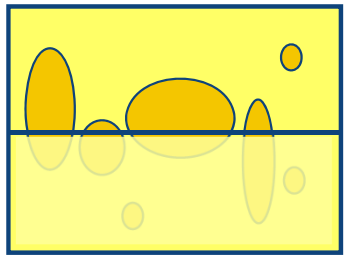
George A. Miller: The Magical Number Seven, Plus or Minus Two – Some Limits on our Capacity for Processing Information; Psychological Review, Vol. 63, 1956, S. 81-97

[Pich et al.: Visual Analysis of Importance and Grouping in Software Dependency Graphs. Proc. SOFTVIS 2008]

Konzepte zur Komplexitätsbeherrschung



- Teile und Herrsche – „Vertikale“ Problemzerlegung
  - Oft nach Teilfunktionen (Modularisierung)
  - Gut geeignet bei wohlstrukturierten Problemen mit abgrenzbaren Teilen
- Abstraktion – „Horizontale“ Problemzerlegung
  - Vereinfachte Beschreibung der Realität, die einige der Details oder Eigenschaften herausstellt, während sie andere unterdrückt
  - Bezogen auf die Perspektive des jeweiligen Betrachters werden jene relevanten Charakteristika herausgearbeitet, die den Betrachtungsgegenstand von anderen unterscheiden und ihn möglichst scharf abgrenzen
  - Eine gute Abstraktion lenkt die Aufmerksamkeit des Betrachters auf die im Kontext wesentlichen Dinge und verzichtet auf bedeutungslose und ablenkende Details
  - Oft in mehreren Stufen (Hierarchisierung)

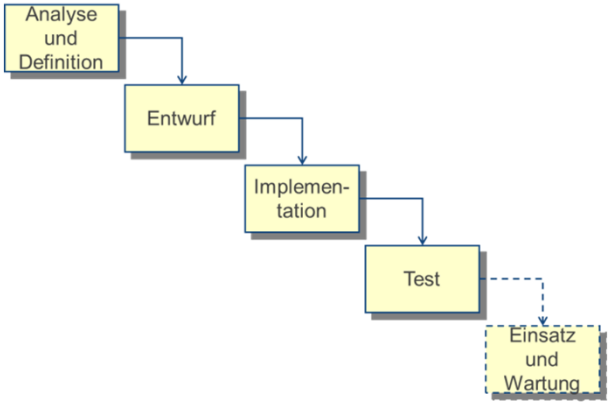


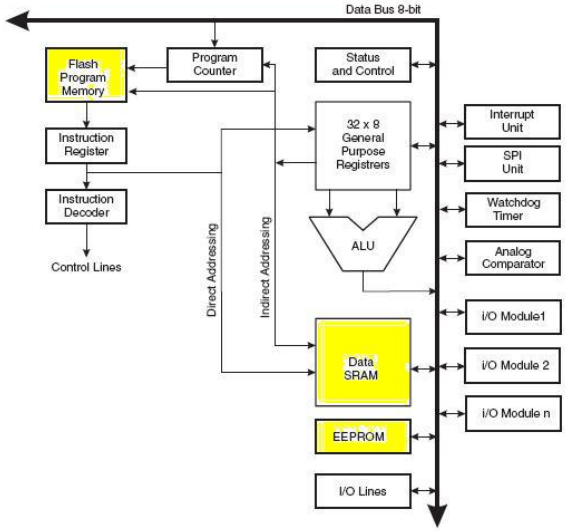
- Prinzip des geringsten Erstaunens (principle of the least astonishment)
  - Es werden genau die im Kontext benötigten Eigenschaften betrachtet; es gibt keine darüber hinausgehenden „überraschenden“ Eigenschaften

- Diese Konzepte bilden die Grundlagen der **Modellierung**

Wer modelliert wo(für)? Beispiele

- Zerlegung





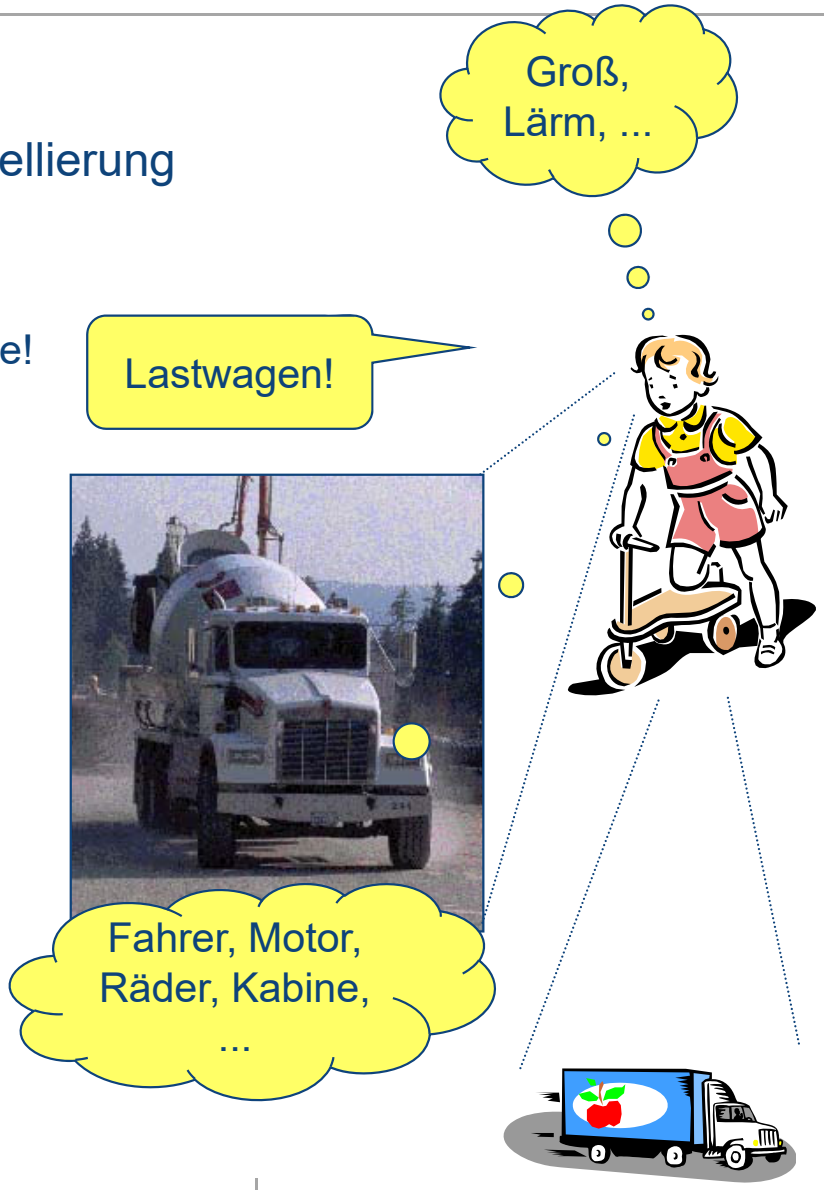
- Abstraktion

http://www.google.de/imghp

Was ist und wie entsteht ein Modell?

- Wir erkennen und verstehen die Welt durch Modellierung
- Zuerst: Zeigehandlung/Sinnliche Wahrnehmung
  - Schemen- bzw. Bild-(Wieder-)Erkennung
  - Szenen- und Bildzerlegung ⇒ Teile und Herrsche!
- Dann: Benennung
  - Symbolerzeugung
- Dann: Prädikatoren- und Begriffsbildung
  - Eigenschaften und Fähigkeiten des zu identifizierenden Objektes bzw. dessen Beziehungen zu anderen Objekten werden identifiziert; Begriffsbildung, Abstraktion, Objektifizierung
  - Das ist bereits (sprachliche) Modellierung!

Ein Modell ist eine aufgaben-angemessene, abstrahierende Sicht auf einen Gegenstand oder Sachverhalt



Drei Modell-Kriterien

- Abbildung
  - Es gibt ein (existierendes oder projektiertes) Ding oder einen Sachverhalt, der im Modell abgebildet wird
  - Entsprechend unterscheidet man deskriptive (beschreibende) und präskriptive (vorschreibende) Modellierung
  - Ding oder Sachverhalt werden oft als “das Original” bezeichnet

- Abstraktion (Verkürzung)
  - Nicht alle Eigenschaften des Originals sind im Modell repräsentiert, sondern das Modell ist eine “Verkürzung” (Abstraktion, Reduktion, Vereinfachung, …) des Originals
  - Auf der anderen Seite muss das Modell zumindest einige Eigenschaften des Originals widerspiegeln

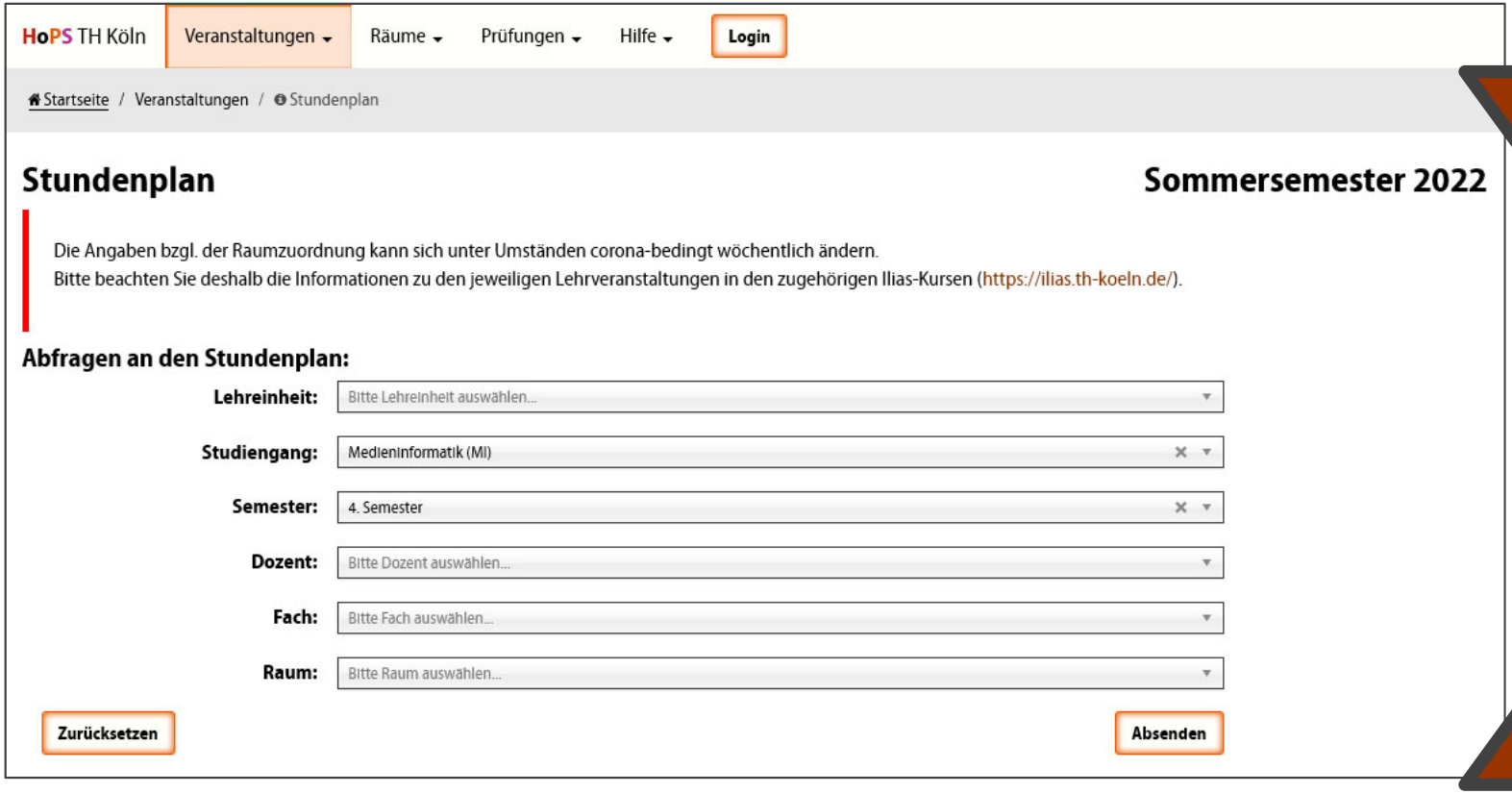
- Pragmatik
  - Das Modell ersetzt/beschreibt das Original für einen bestimmten Zweck. D.H., das Modell ist nützlich …
    - … für jemanden (ein wahrnehmendes oder agierendes Subjekt) …
    - … in einem bestimmten Zeitraum und einem bestimmten Kontext …
    - … eingeschränkt für bestimmte Interpretationen oder Operationen

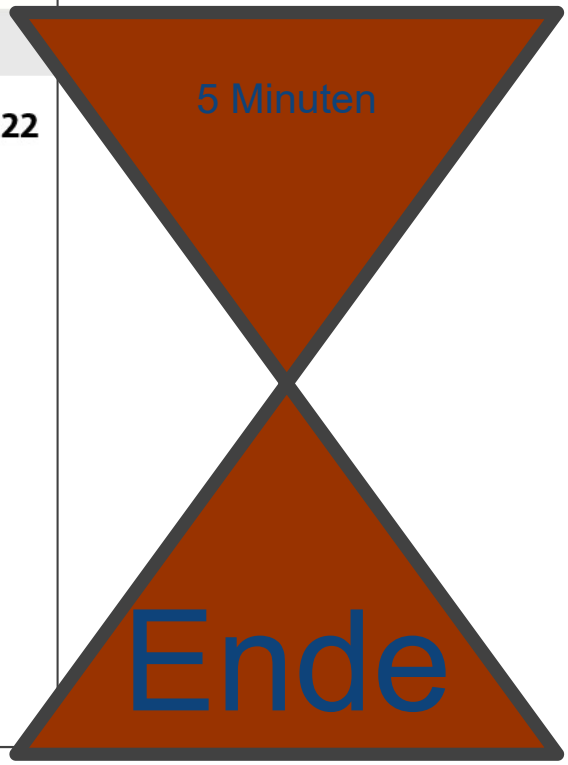
Nach: Stachowiak, H. Allgemeine Modelltheorie Springer, Wien, 1973

Max. 5 Minuten!

Aufgabe 0: Modellierung

- Erstellen Sie ein Modell eines Stundenplans im HoPS

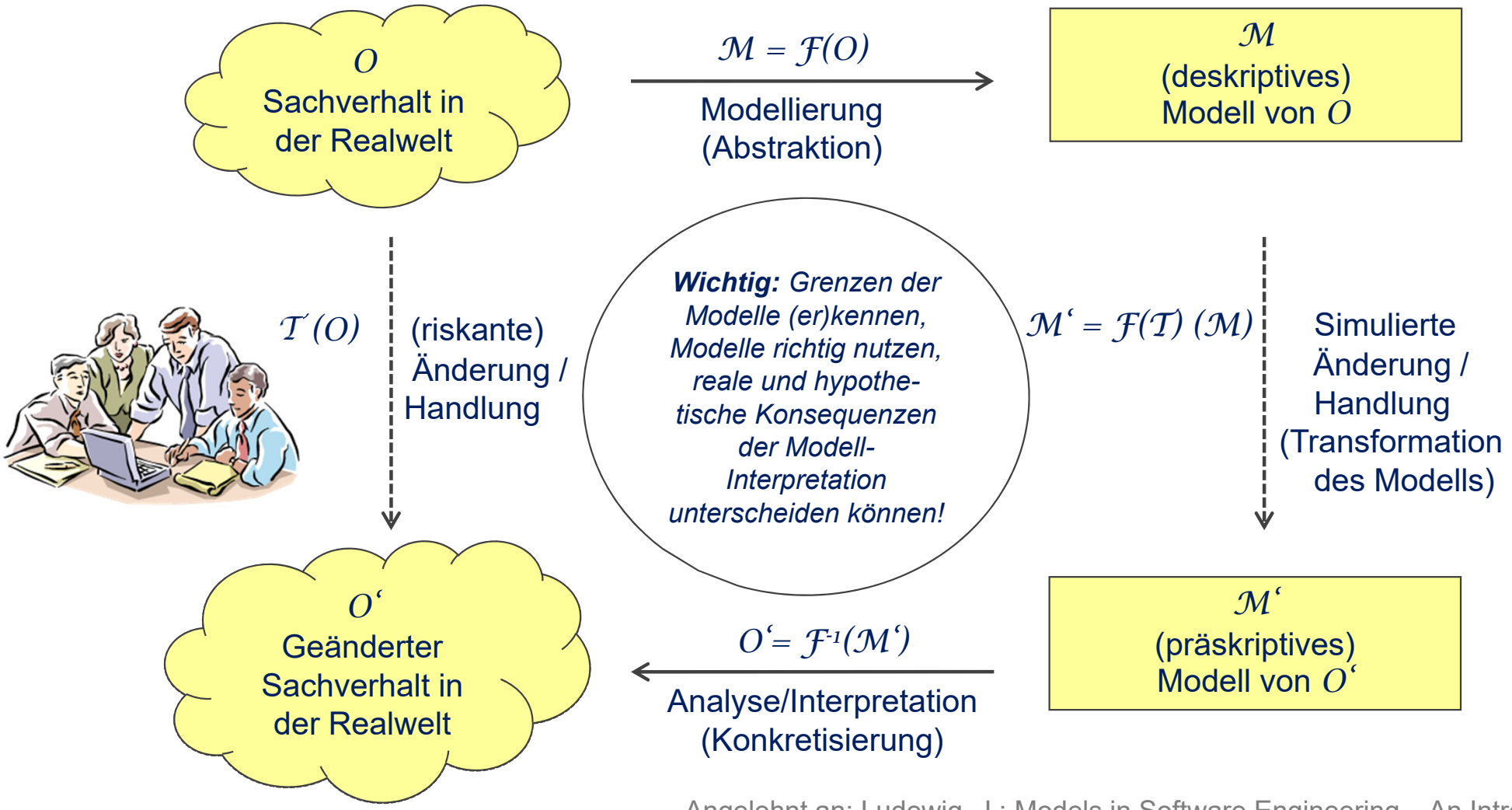




Lösungshinweis Aufgabe 0: Modellierung

- Erstellen Sie ein Modell eines Stundenplans im HoPS

Wofür Modelle? Pragmatik, der Zweck der Modellierung!



Angelehnt an: Ludewig, J.: Models in Software Engineering – An Introduction. Software Systems Modelling, Springer, (2003) 2: 2002. S. 5–14

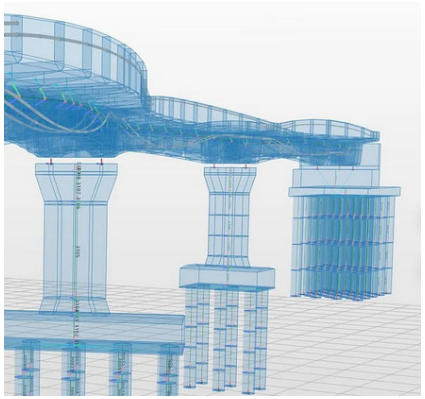
Beispiel einer Modellierung

O

$\mathcal{M} = \mathcal{F}(O)$

Modellierung
(Abstraktion)

$\mathcal{M}$



$\mathcal{T}(O)$

Hält die Brücke eine Last L?

Wichtig: *Grenzen der Simulation (er)kennen, Modelle validieren!*

$\mathcal{M}' = \mathcal{F}(\mathcal{T})(\mathcal{M})$

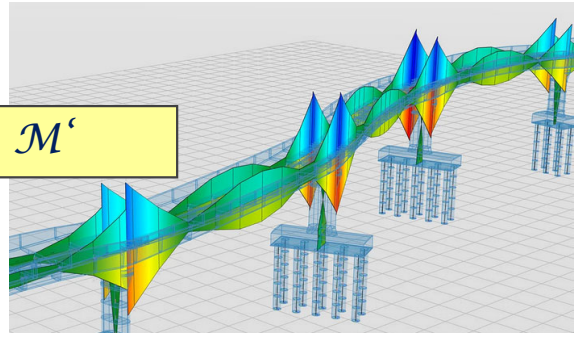
Simulation der Belastung

O'
Ja, Last L ist OK

$O' = \mathcal{F}^{-1}(\mathcal{M}')$

Analyse/Interpretation
(Konkretisierung)

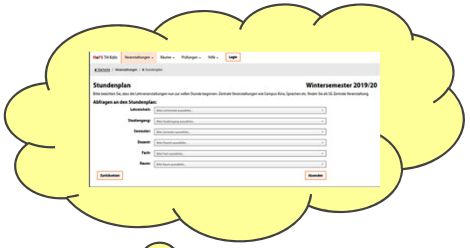
$\mathcal{M}'$

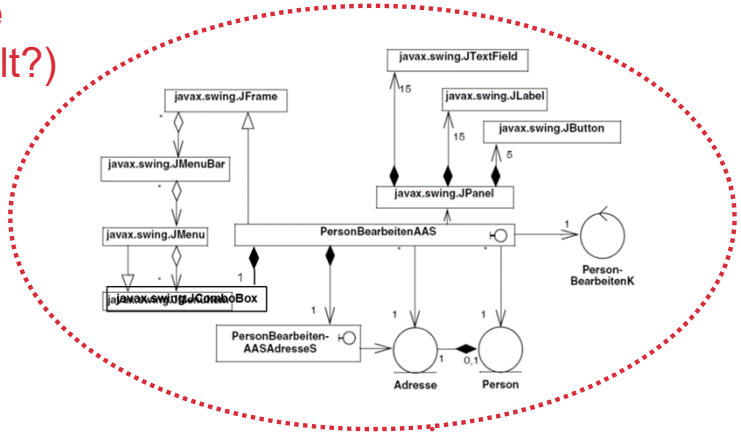


Abbildungen: https://www.baunetzwissen.de

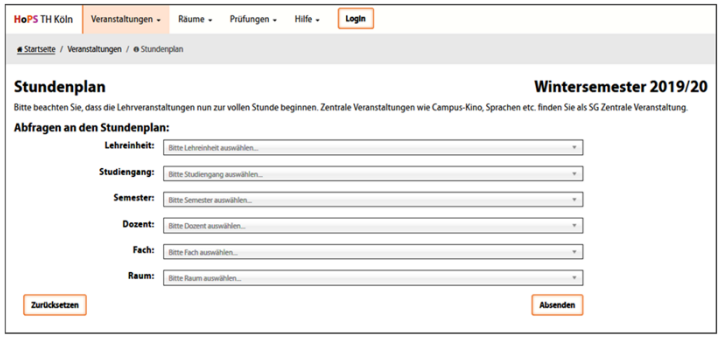
Komplexitätsbeherrschung durch Modellierung

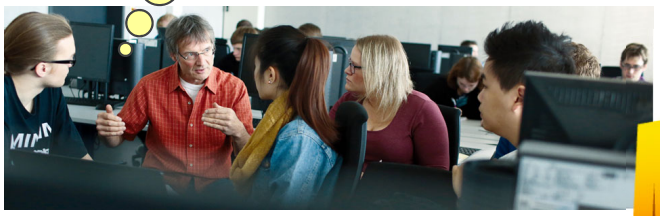
Deskriptive Modelle
(wie wurde es entwickelt?)





Reale Produkte (“Originale”)

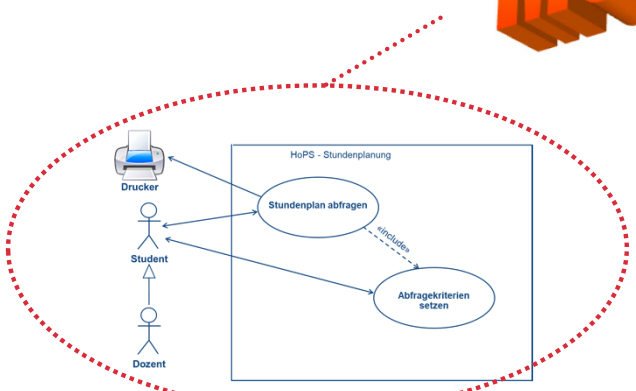




Domänenspezifische Bedarfe

Modellierung

Oft: **Transiente Modelle** (erst präskriptiv, dann nach Entwicklung deskriptiv)

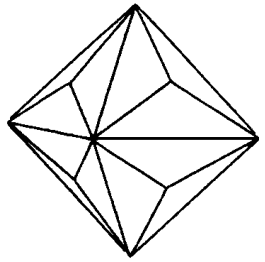


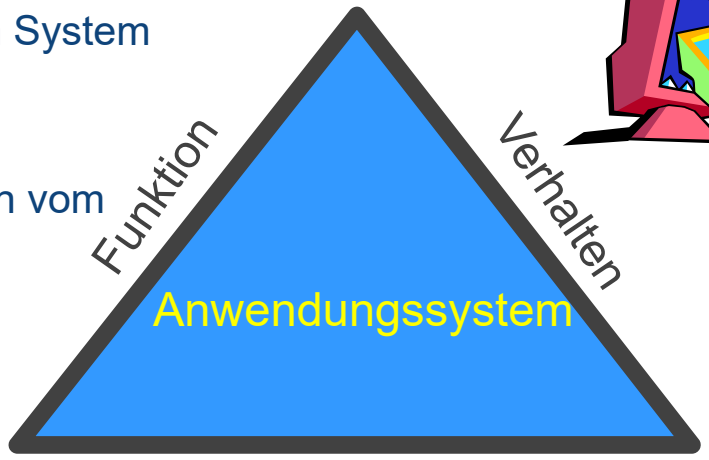
Präskriptive Modelle
(Wie soll es entwickelt werden?)

https://balsamiq.com

Drei wesentliche Modellierungssichten

- Funktion (Nutzung)
  - In welchem Kontext arbeitet das System?
  - Welche Aufgaben soll das Anwendungssystem unterstützen?
- Struktur (Information)
  - Aufbau des Systems aus Teilen sowie deren Eigenschaften und Beziehungen
  - Welche Dinge und Sachverhalte muss das System „kennen“?
  - Womit soll das System „arbeiten“ (Informationen/Daten)?
- Verhalten (Dynamik)
  - Wie soll die Umgebung/Benutzer mit dem System interagieren?
  - Wie reagiert das System auf Ereignisse?
  - Welche (Geschäfts-)Regeln müssen wann vom System beachtet werden?
  - Wie arbeitet das System intern?





Objekte, Modelle, Diagramme

Rene Magritte (* 1898, † 1967, belgischer Maler des Surrealismus)

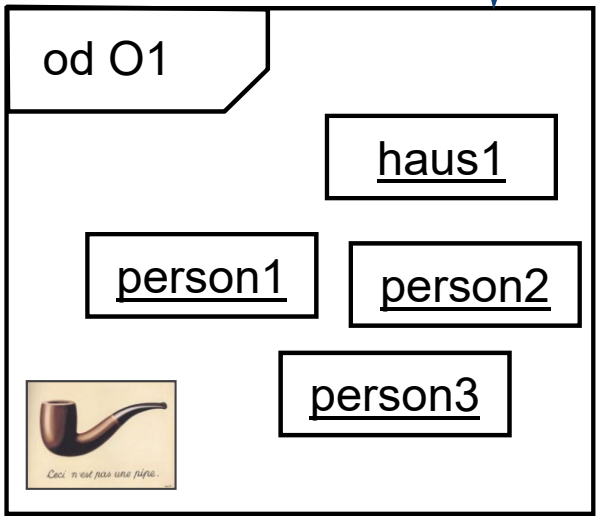
„Trahison des images (Ceci n'est pas une pipe)“

„Der Verrat der Bilder (Dies ist keine Pfeife)“

Objekte, Modelle, Diagramme

- **Objekt**e modellieren einzelne, konkrete „Dinge“ bzw. Sachverhalte
- Ein **Modell** enthält viele Objekte (und andere Modellierungselemente)
  - Zu denselben Dingen gibt es (unendlich) viele Modelle
- Ein **Diagramm** ist eine grafische Darstellung bzw. „Visualisierung“ von einigen (nicht notwendig allen!) Elementen eines Modells
  - Zu einem Modell gibt es (unendlich) viele Diagramme

Diagramm-Rahmen im Skript oft weggelassen



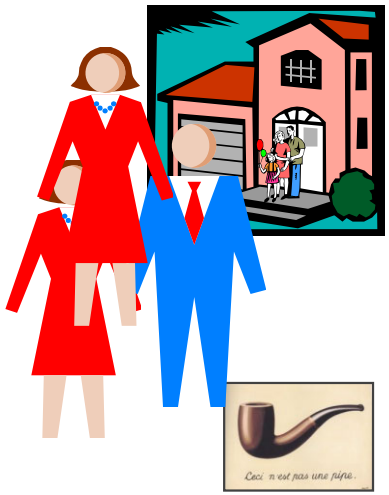
Objekte, Modelle, Diagramme

model.uml (%HOME%\workspace-papyrus\SeminarIS\)

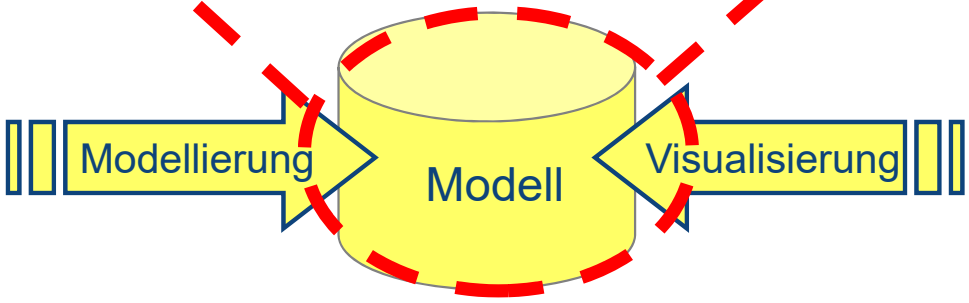
```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xmi:XMI xmi:version="20131001" xmlns:xmi="http://www.omg.org/spec/XMI/20131001"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:MOOS="http:///schemas/MOOS/_cbXLQLr3EeuCuKhzM1QTZw/5"
   xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" xmlns:uml="http://www.eclipse.org/uml2/5.0.0/UML"
   xsi:schemaLocation="http:///schemas/MOOS/_cbXLQLr3EeuCuKhzM1QTZw/5
   ../MOOSProfile/model.profile.uml#_cbg8QLr3EeuCuKhzM1QTZw">
292        <packagedElement xmi:type="uml:InstanceSpecification" xmi:id="_QOvmwBoVEey8997K0Ob4HA" name="haus 1"/>
293        <packagedElement xmi:type="uml:InstanceSpecification" xmi:id="_p7uiUBoVEey8997K0Ob4HA" name="person 1"/>
294        <packagedElement xmi:type="uml:InstanceSpecification" xmi:id="_rNMxQBoVEey8997K0Ob4HA" name="person 2"/>
295        <packagedElement xmi:type="uml:InstanceSpecification" xmi:id="_1jUQgBoVEey8997K0Ob4HA" name="person 3"/>
```

n-
Skript
sen

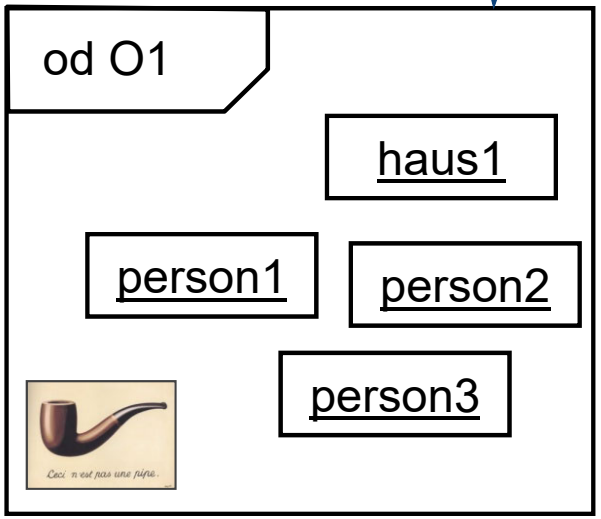
„Dinge“



Modell



(Objekt-)Diagramm

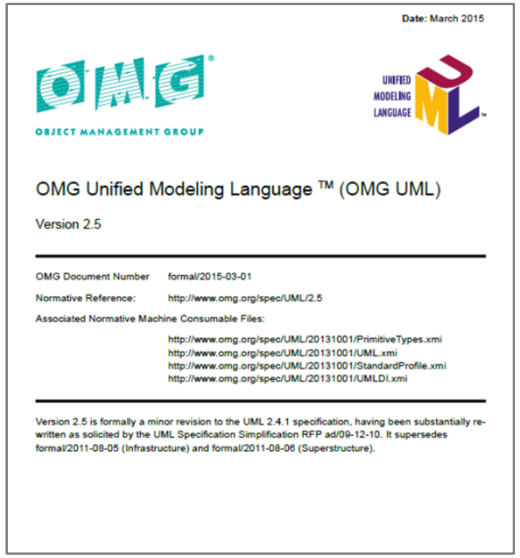


Unified Modelling Language – UML

- Modellierungssprache
- KEINE Methode
- OMG-Standard seit 1997
- Aktuelle Version: V2.5.1, Dezember 2017

- De Facto – Industriestandard und ISO-Standard
  - ISO/IEC 19505-2, Information Technology — OMG Unified Modeling Language (OMG UML) Version 2.4 - Part 2: Superstructure; 2012

- OMG Unified Modeling Language (UML)
  - https://www.omg.org/spec/UML/
  - Definiert UML-Metamodell und grafische Notation/Diagramme
  - 794 Seiten …


Date: March 2015

OMG Unified Modeling Language ™ (OMG UML)

Version 2.5

OMG Document Number: formal/2015-03-01

Normative Reference: http://www.omg.org/spec/UML/2.5

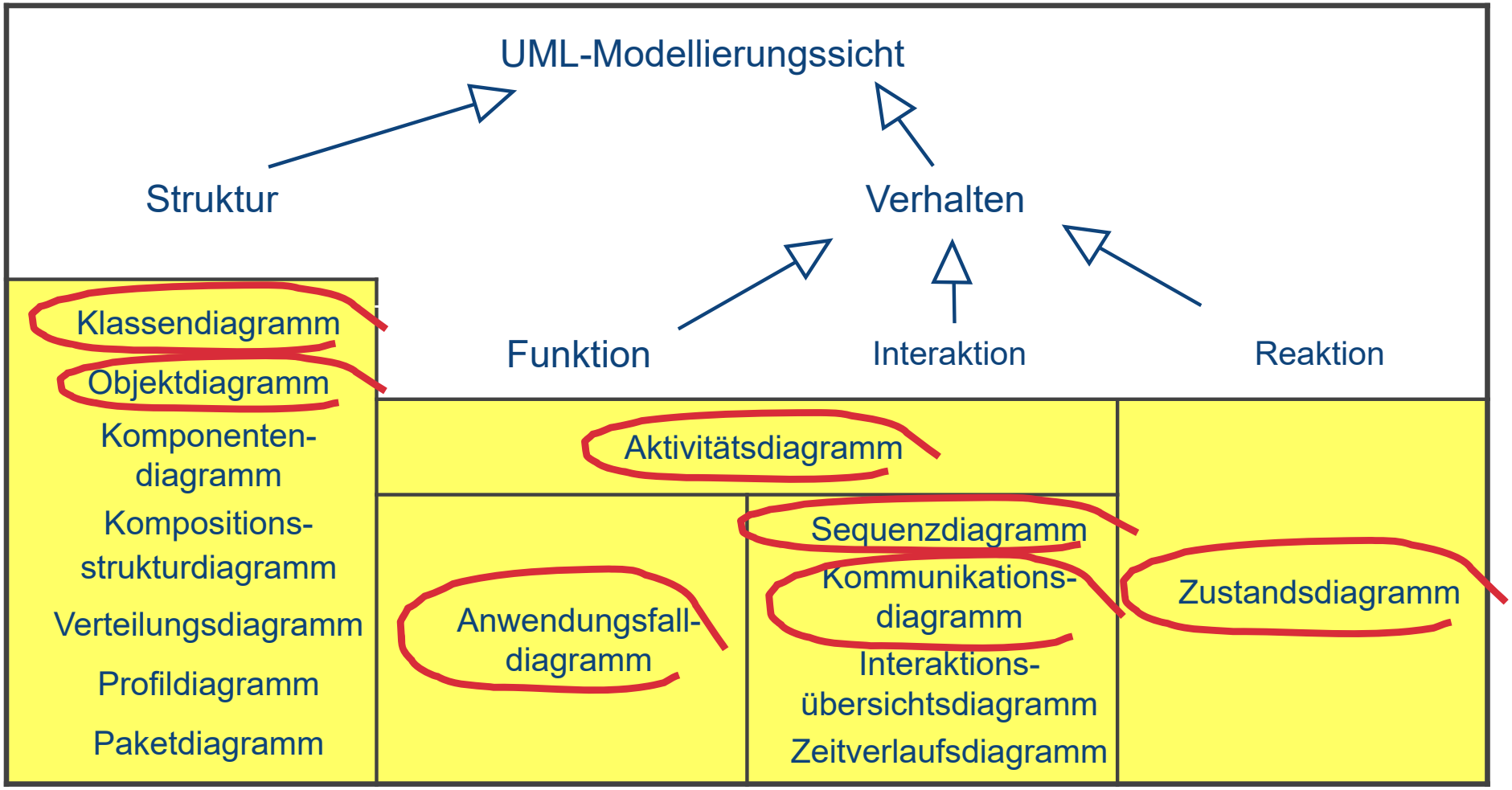
Associated Normative Machine Consumable Files:

http://www.omg.org/spec/UML/20131001/PrimitiveTypes.xmi
http://www.omg.org/spec/UML/20131001/UML.xmi
http://www.omg.org/spec/UML/20131001/StandardProfile.xmi
http://www.omg.org/spec/UML/20131001/UMLDI.xmi

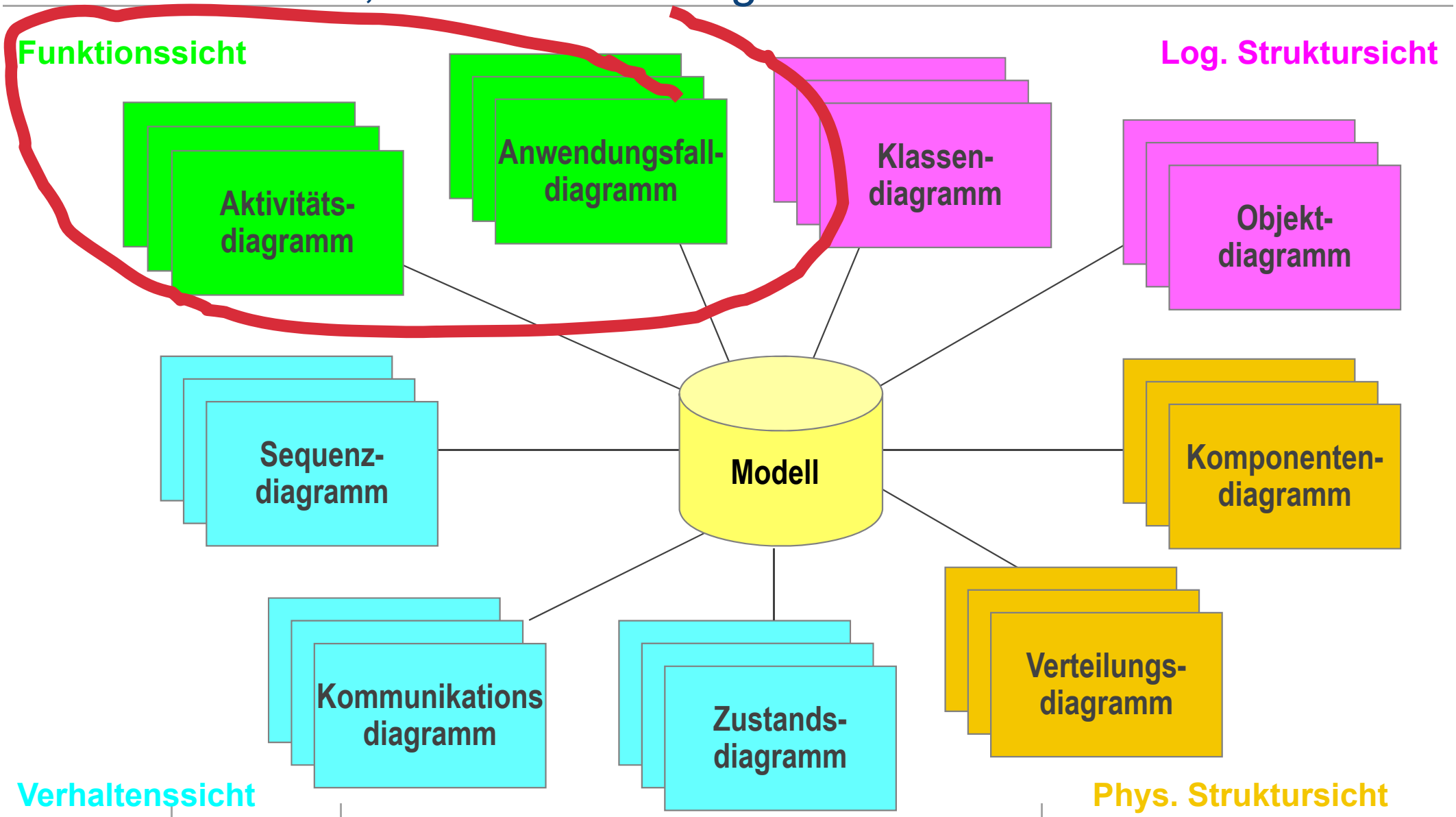
Version 2.5 is formally a minor revision to the UML 2.4.1 specification, having been substantially re-written as solicited by the UML Specification Simplification RFP ad/09-12-10. It supersedes formal/2011-08-05 (Infrastructure) and formal/2011-08-06 (Superstructure).

https://www.oose.de/nuetzliches/uml-unified-modeling-language/
https://www.omg.org/spec/UML/

Die 14 (!) Diagramme der UML 2.x



UML: Modell, Sichten und Diagramme

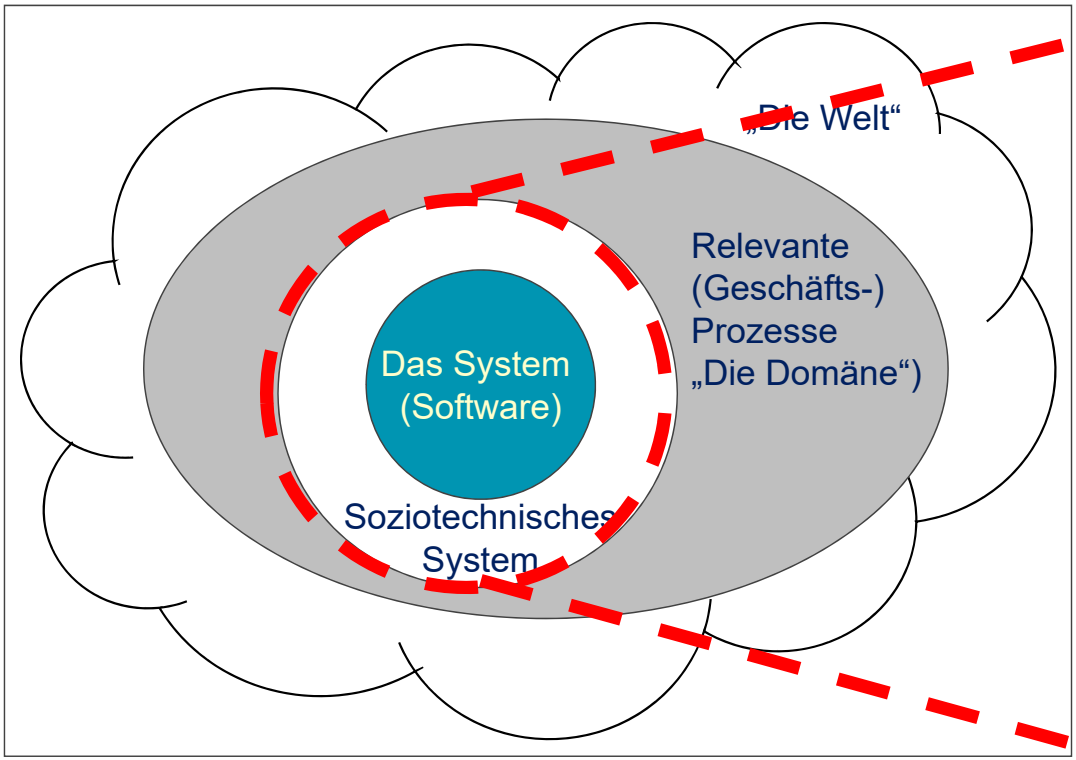


Wo sind wir?

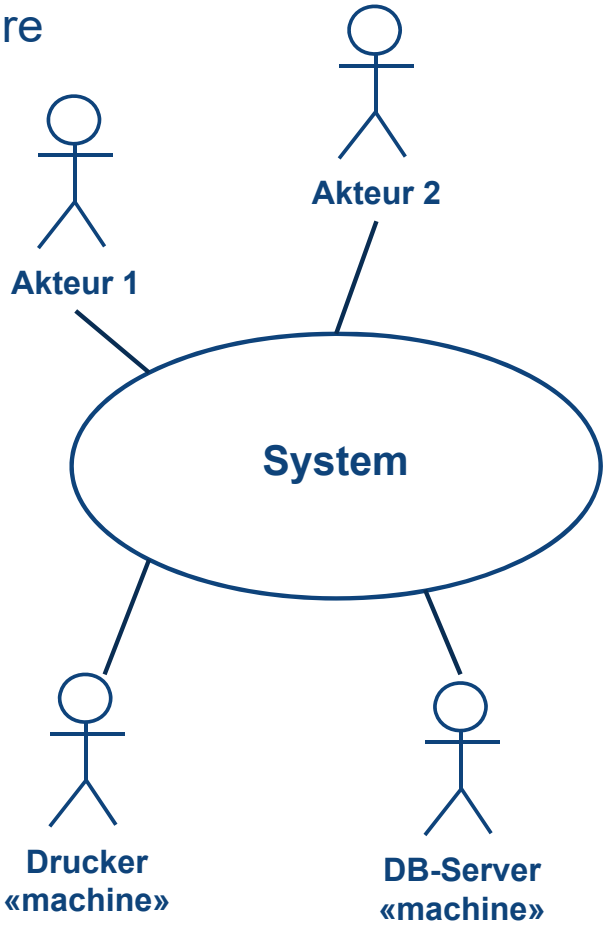
- Modellierung und UML
- **Akteure und Anwendungsfälle**
- Beziehungen zwischen Anwendungsfällen
- Aktivitätsmodellierung

Festlegen des Systemkontexts

- Was gehört zum System? Funktionalität -> Anwendungsfälle
- Wer interagiert mit dem System? Systemkontext -> Akteure
- Wen interessiert das System? Stakeholder $\supsetneq$ Akteure



Ebenen des Systemkontexts (nach [Pohl 2008]

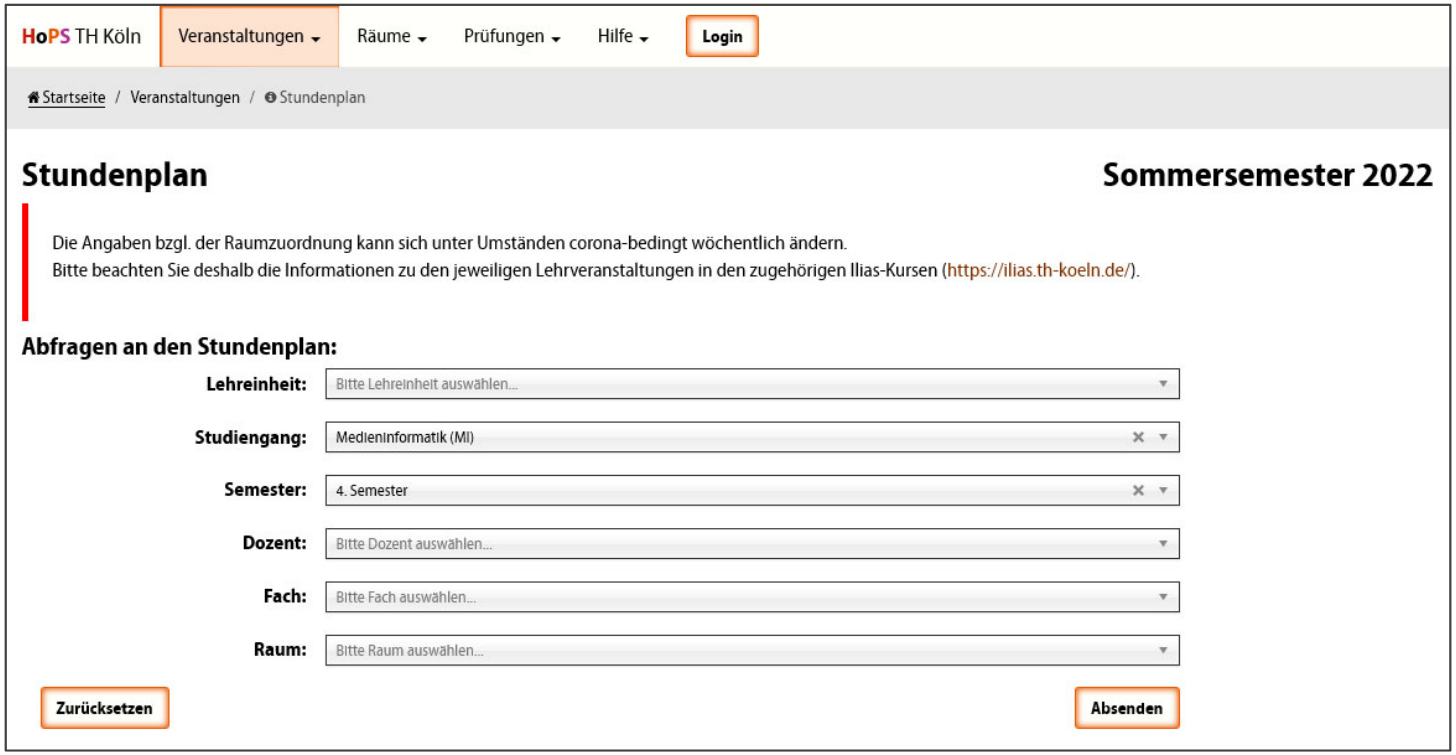


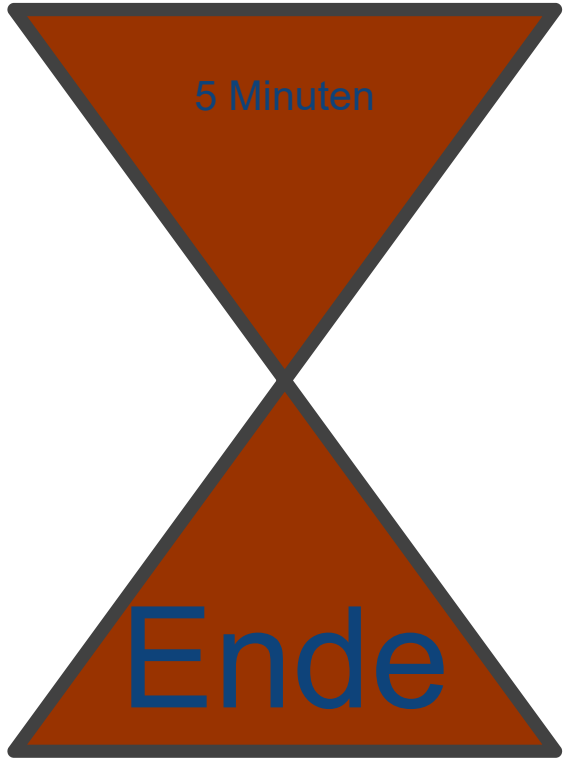
Kontextdiagramm

Max. 5 Minuten!

Aufgabe 1: Kontextmodellierung

- Erstellen Sie ein Kontextdiagramm der Stundenplanung im HoPS





Lösungsidee Aufgabe 1: Kontextmodellierung

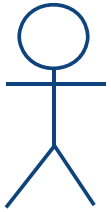
- Erstellen Sie ein Kontextdiagramm der Stundenplanung im HoPS

Funktionsmodellierung mit der UML

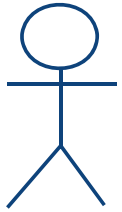
- Ziel: Die wesentlichen Funktionen des Anwendungssystems identifizieren und modellieren
- Wer? Akteure
  - Repräsentieren die Umgebung des Systems; abstrahieren von menschlichen und maschinellen „Benutzern“
- Was? Anwendungsfälle
  - Repräsentieren Systemfunktionen zur Unterstützung z.B. von (Teilen von) Geschäftsprozessen
- Logisch-fachliche Beziehungen zwischen Anwendungsfällen
  - «include» dient zur Redundanzvermeidung
  - «extend» erlaubt eingebaute Erweiterungsstellen
  - Generalisierung erlaubt abstrakte Funktionsbeschreibungen und spezielle Realisierungen
- Ablaufbeschreibungen
  - Beschreiben die von außen beobachtbaren Abläufe und Interaktionen zwischen den Akteuren und dem System
  - In textueller Form (Anwendungsfall-Spezifikationsschablone) und/oder mit Aktivitätsdiagrammen oder (seltener) Sequenzdiagrammen

Akteure – Schnittstellen zur Umwelt

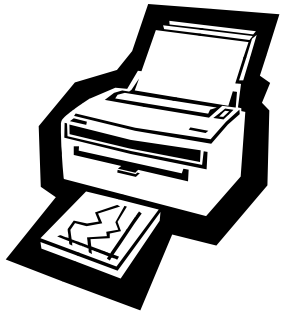
- Innerhalb eines Geschäftsprozesses interagieren menschliche oder maschinelle “Benutzer“ mit dem Anwendungssystem, um bestimmte **Aufgaben** durchzuführen und konkrete **Ziele** zu erreichen
- Im Normalfall spielen mehrere Benutzer in Bezug auf das Anwendungssystem eine bestimmte **Rolle**. Mit jeder solchen Rolle sind bestimmte Aufgaben verknüpft
- Ähnlich wie eine Klasse eine Abstraktion konkreter Objekte darstellt, abstrahiert ein **Akteur** (actor) eine bestimmte Rolle konkreter Benutzer der realen Welt – der Akteur-Name wird daher fett gedruckt
- Ein konkreter Benutzer kann dabei durchaus mehrere Rollen spielen und daher durch mehrere Akteure modelliert werden



Privatkunden Sachbearbeiter



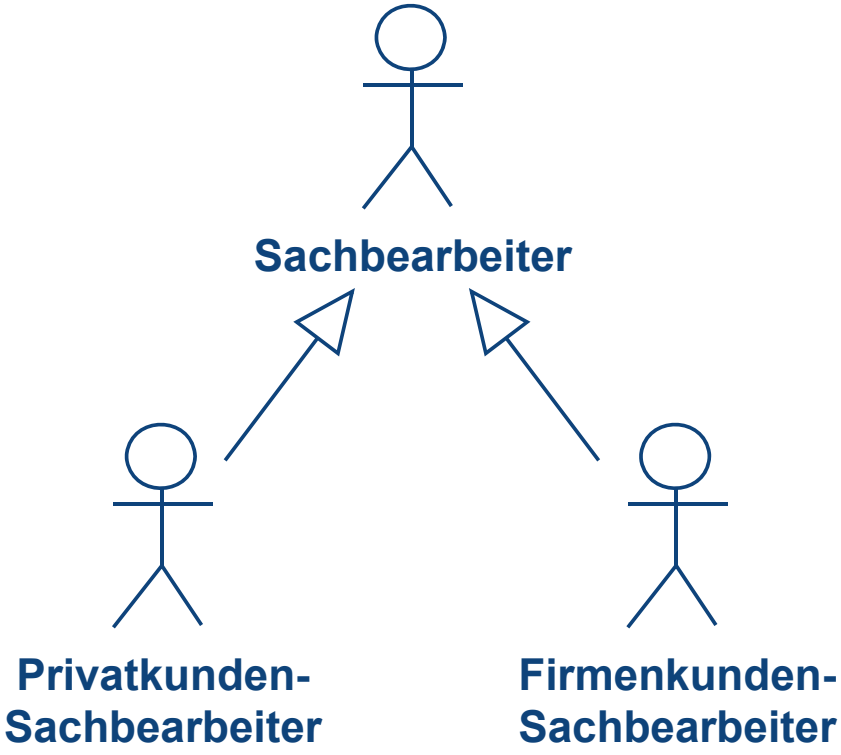
Firmenkunden Sachbearbeiter



Drucker

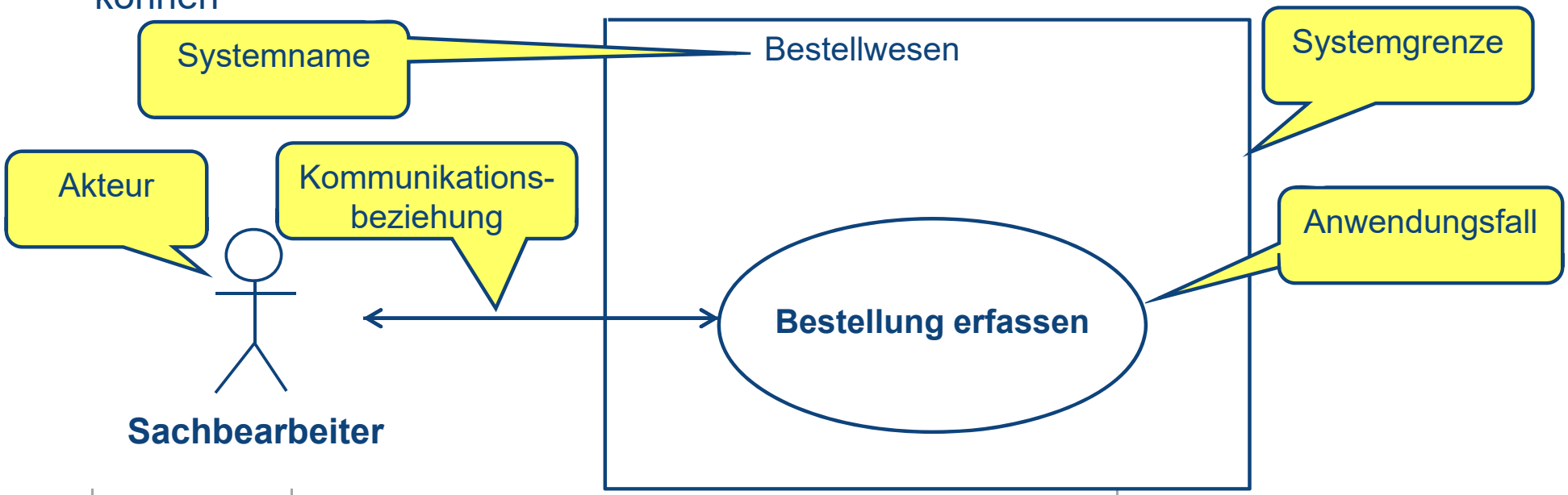
Generalisierung von Akteuren

- Unterschiedliche Akteure können gemeinsame Eigenschaften haben und somit — ebenso wie unterschiedliche Klassen — in einer Generalisierungsbeziehung („Vererbung“) zueinander stehen
- Analog zur Generalisierung („Vererbung“) zwischen Klassen können die spezialisierten Akteure mindestens die Rollen der allgemeineren Akteure spielen



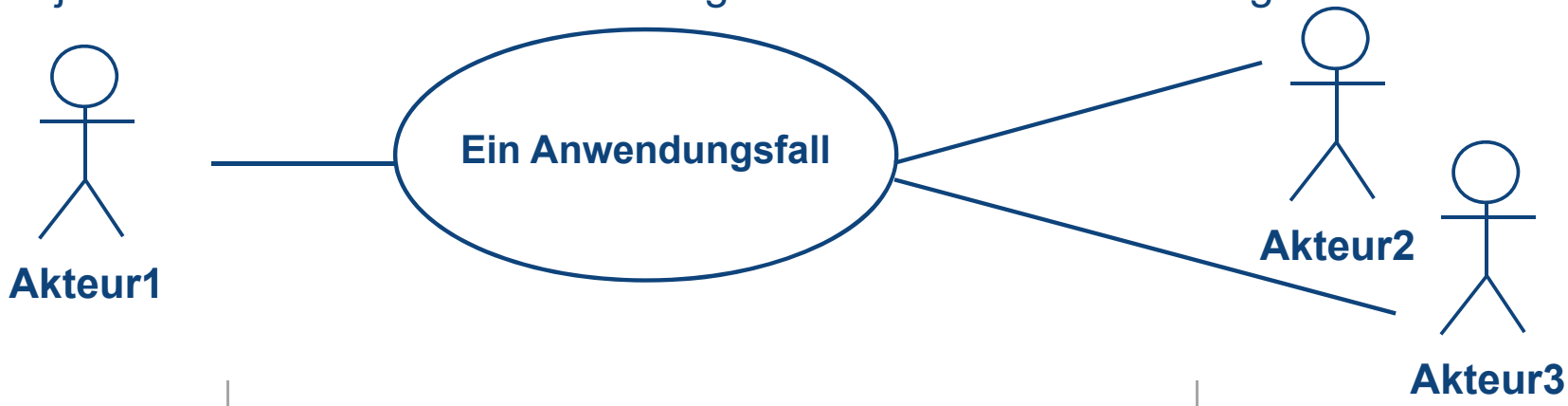
Anwendungsfalldiagramm

- Im Anwendungsfalldiagramm wird normalerweise durch eine Systemgrenze angedeutet, dass die Anwendungsfälle vom Anwendungssystem „erbracht“ werden und die Akteure außerhalb des Systems stehen
- Die Kommunikation der Akteure mit dem System wird durch Assoziationen (Kommunikationsbeziehungen) zwischen Akteuren und Anwendungsfällen modelliert, wobei einseitig gerichtete Informationsflüsse durch entsprechende Navigierbarkeiten der Assoziationen modelliert werden können



Anwendungsfälle - Systemfunktionen

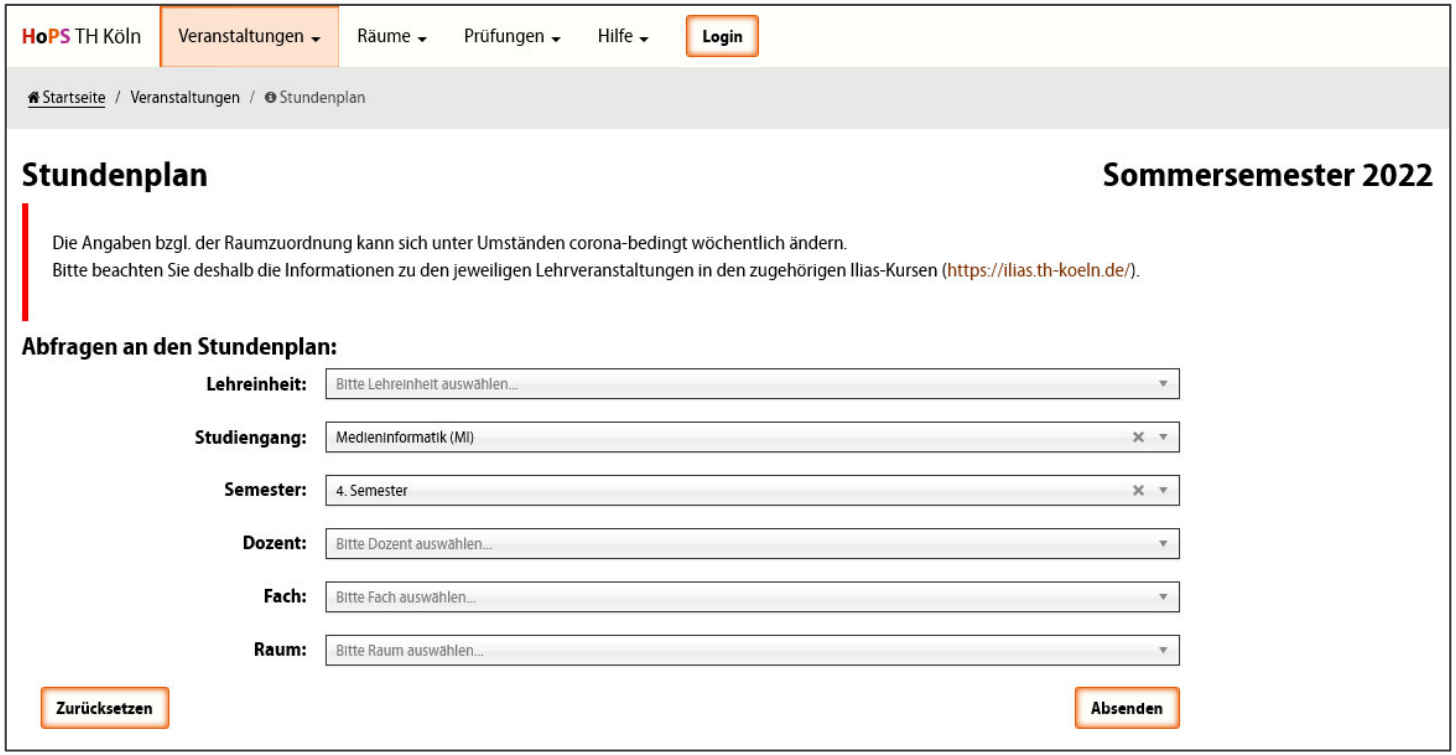
- Anwendungsfälle beschreiben die Funktionen des Anwendungssystems, welche die Akteure zur Durchführung ihrer Aufgaben benötigen
- Jeder Anwendungsfall (use case) formuliert eine in sich abgeschlossene Teilfunktionalität des Anwendungssystems, die für mindestens einen Akteur ein bestimmtes Ergebnis innerhalb des Geschäftsprozesses erbringt
- Darüber hinaus können weitere Akteure an der Durchführung eines Anwendungsfalls beteiligt sein
- Ein Anwendungsfall beschreibt alle Möglichkeiten, die bei einer Funktionsnutzung auftreten können, ähnlich wie eine Klasse viele konkrete Objekte beschreibt – der Anwendungsfall-Name wird daher fett gedruckt

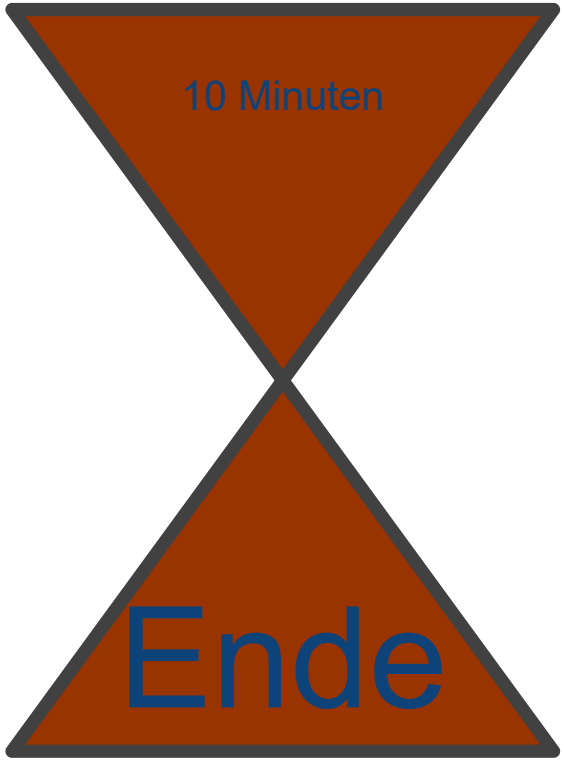


Max. 10 Minuten!

Aufgabe 2: Anwendungsfallmodellierung

- Erstellen Sie ein einfaches Anwendungsfalldiagramm für eine Stundenplan-Abfrage im HoPS





Lösungsidee Aufgabe 2: Anwendungsfallmodellierung (1)

Lösungsidee Aufgabe 2: Anwendungsfallmodellierung (2)

Textuelle Beschreibung von Anwendungsfällen

- Anwendungsfälle spielen eine zentrale Rolle bei der Kommunikation zwischen Anforderungsermittler (Analytiker) und Benutzer
- Textuell werden sie daher zunächst mehr oder weniger “prosaisch“ aus der Sicht der Akteure beschrieben, wobei das Anwendungssystem als “Black-Box“ betrachtet wird
- Im Zuge der fortschreitenden Präzisierung der Beschreibung können sog. Vor- und/oder Nachbedingungen ergänzt werden
- **Vorbedingung** (precondition) des Anwendungsfalls grenzt die Situationen ein, in denen der Anwendungsfall begonnen werden darf
  - Beschreibt „Systemzustand“, muss vom System „geprüft“ werden können
  - Vorbedingung nicht erfüllt $\Rightarrow$ Anwendungsfall darf nicht beginnen!
- **Nachbedingung** (postcondition) präzisiert Ergebnis des Anwendungsfalls
  - Setzt erfüllte Vorbedingung voraus!
  - Für Erfolgsfall und Misserfolgsfall i.d.R. unterschiedliche Nachbedingungen
- Bei erfüllter Vorbedingung garantiert das System die Nachbedingung(en)
- Vor- und Nachbedingungen implizieren oft bestimmte Ausführungs-Reihenfolgen der Anwendungsfälle

Beispiel

use case Kunde deaktivieren
actors Sachbearbeiter
trigger Ein Kunde möchte aus dem System gelöscht werden **ODER** ist nicht mehr erreichbar
precondition Sachbearbeiter angemeldet **UND** mindestens ein Kunde aktiviert
main flow
1. Der Sachbearbeiter bestimmt den zu deaktivierenden Kunden anhand der Kundennummer.
2. Die Deaktivierung muss vom Sachbearbeiter bestätigt werden.

alternative flow Kundennummer unbekannt
1.a Der Sachbearbeiter wählt den zu deaktivierenden Kunden aus einer Liste aller Kunden.

postcondition Ein vorher aktiver Kunde ist deaktiviert
exceptional flow Offene Rechnung
2.a Hat der ausgewählte Kunde noch offene Rechnungen, so darf er nicht deaktiviert werden.

postcondition Keine Deaktivierung durchgeführt
end Kunde deaktivieren

Schablone zur textuellen Anwendungsfallbeschreibung

Anwendungsfälle beschreiben die möglichen Abläufe und Interaktionen, die zwischen Akteuren und Anwendungssystem im Rahmen einer (Teil-)Funktionalität stattfinden können

Vorbedingung (precondition): überwacht, ob der Anwendungsfalls starten darf, muss vom System prüfbar sein!

Auslöser (trigger): gibt an, wann/warum der Anwendungsfall ausgeführt werden soll

“**Normaler“ Ablauf** (main flow): Einfach halten! Mögliche Abweichungen vom normalen Ablauf in alternative flow-Abschnitte!

Alternative Abläufe (alternative flows): Anwendungsfall kann noch wie beim normalen Ablauf erfolgreich enden. Es gilt weiterhin die Nachbedingung des normalen Ablaufs.

Nachbedingung (postcondition) des main flow: spezifiziert das garantierte Ergebnis jedes erfolgreichen Ablaufs

Ausnahmeabläufe (exceptional flows): erfolgreicher Ablauf nicht mehr möglich, daher separate Nachbedingungen

Separate Nachbedingungen der Ausnahmeabläufe

use case Anwendungsfallname
actors AkteurA, AkteurB, ...
trigger …
precondition ...
main flow
....
alternative flow AF1
...
alternative flow AFn
...
postcondition ....

exceptional flow EF1
...
postcondition (von EF1)...

exceptional flow EFm
...
postcondition (von EFm)...
...
end Anwendungsfallname

Wo sind wir?

- Modellierung
- Akteure und Anwendungsfälle
- **Beziehungen zwischen Anwendungsfällen**
- Aktivitätsmodellierung

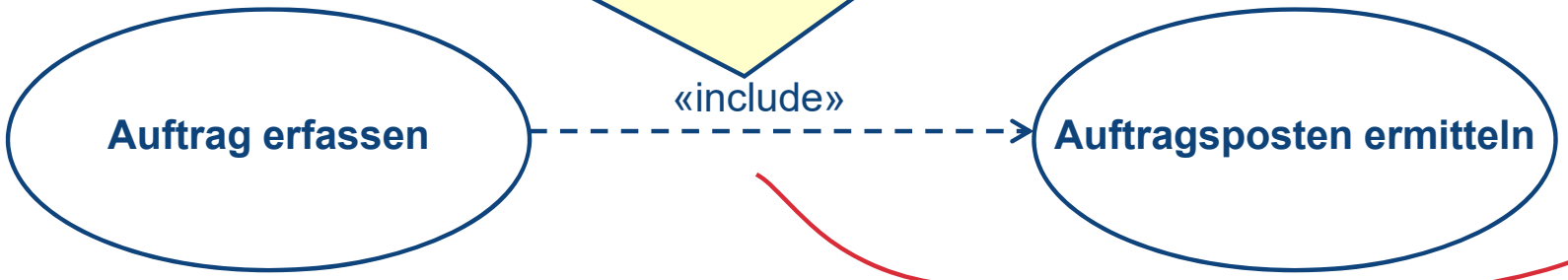
Die include-Beziehung

- Die include-Beziehung wird verwendet, wenn verschiedene Anwendungsfälle dieselbe Teilfunktionalität beinhalten
- Diese **Teilfunktionalität** wird zur Redundanzvermeidung in einem separaten Anwendungsfall beschrieben, der von den “benutzenden“ Anwendungsfällen (Basisanwendungsfälle) verwendet wird
- Eine include-Beziehung von Anwendungsfall A zu Anwendungsfall B bedeutet also, dass A die durch B spezifizierte Teilfunktionalität benutzt
- In der textuellen Spezifikation des Basisanwendungsfalls wird die include-Beziehung durch das Schlüsselwort `include` gefolgt vom Namen des benutzten Anwendungsfalls dargestellt

Beispiel: include-Beziehung

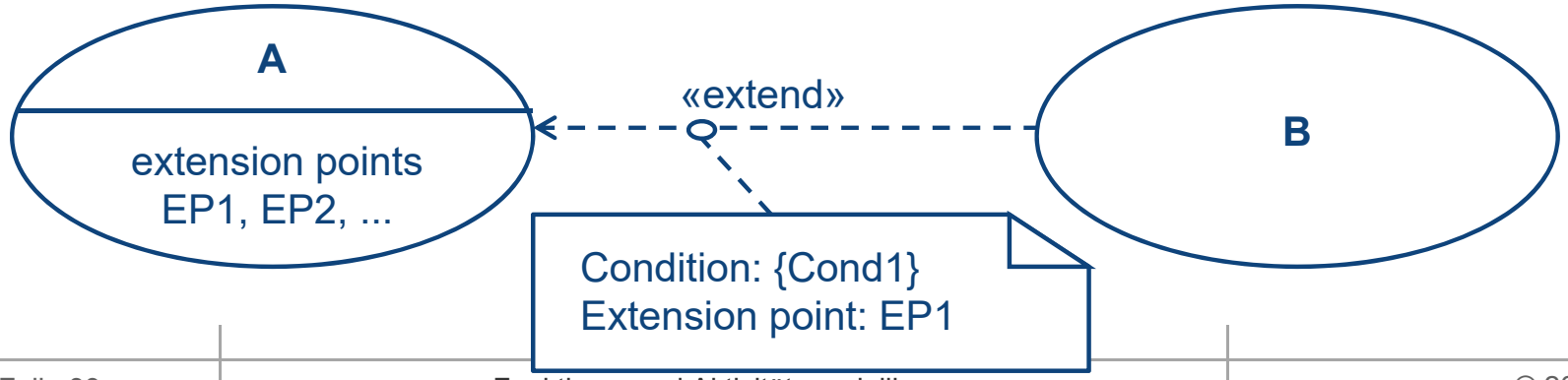
use case Auftrag erfassen
actors Sachbearbeiter
trigger Ein neuer Auftrag ist eingegangen
precondition keine (true)
main flow
Der Sachbearbeiter nimmt den Auftrag (per Telefon, Fax, etc.) entgegen, identifiziert den Kunden im System und nimmt dann alle Auftragsposten mit den jeweiligen Artikeln und Auftragsmengen auf (include "Auftragsposten ermitteln").
...

Beim Ablauf des Anwendungsfalls „Auftrag erfassen“ kann der Anwendungsfall „Auftragsposten ermitteln“ aktiviert werden

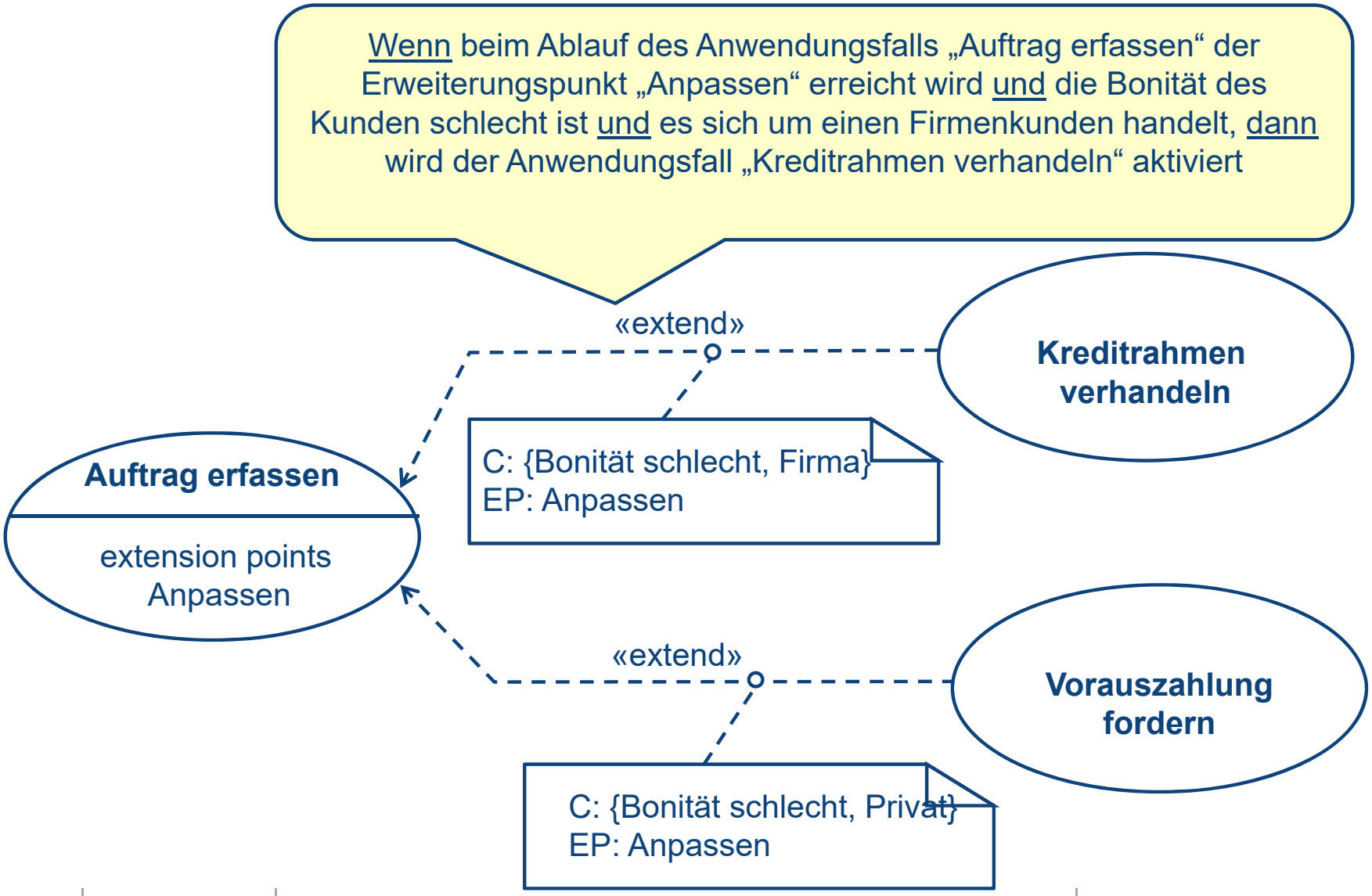


Die extend-Beziehung

- Bei einer extend-Beziehung von einem Anwendungsfall **B** (Erweiterung) zu einem Anwendungsfall **A** (Basisanwendungsfall) kann die Funktionalität von **A** unter bestimmten Bedingungen durch die in **B** beschriebene Funktionalität erweitert werden
- Die Funktionalität des erweiternden Anwendungsfalls **B** ist somit eine optionale Ergänzung des Basisanwendungsfalls **A**
- In der textuellen Spezifikation von Basisanwendungsfall **A** werden die Stellen, an den Funktionalitäten erweiternder Anwendungsfälle eingefügt werden kann, mit `extension point` als Erweiterungspunkte gekennzeichnet
- Für die extend-Beziehung können (in einer Notiz) sowohl die Bedingung, bei deren Eintreten der Anwendungsfall B einzufügen ist, als auch der oder die entsprechenden Erweiterungspunkte angegeben werden



Beispiel: extend-Beziehung



Beispiel: extend-Beziehung

use case Auftrag erfassen

actors Sachbearbeiter

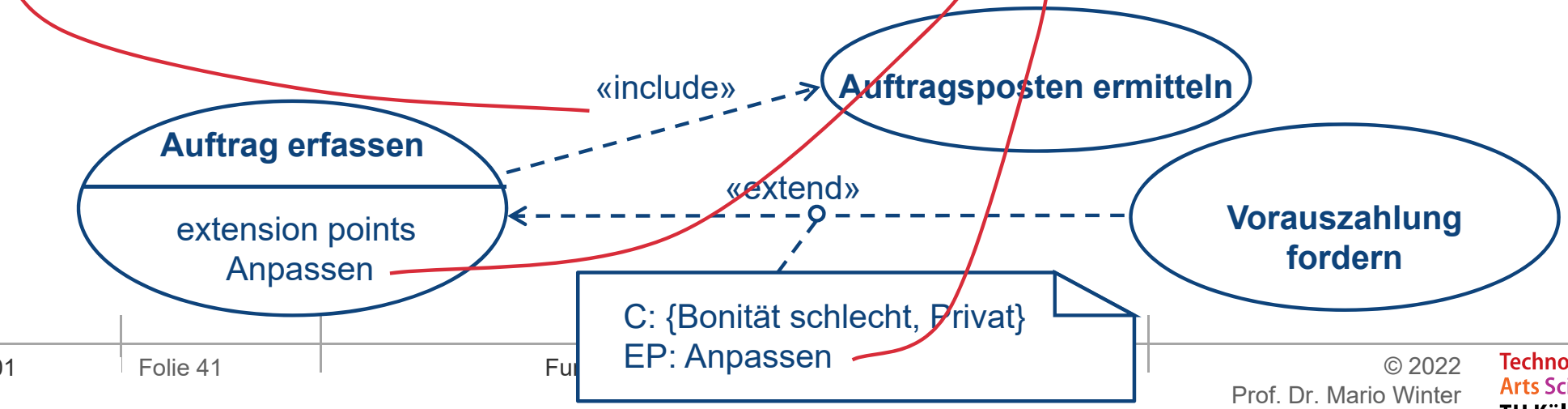
trigger Ein neuer Auftrag ist eingegangen
precondition -

main flow

Der Sachbearbeiter nimmt den Auftrag (per Telefon, Fax, etc.) entgegen, identifiziert den Kunden im System und nimmt dann alle Auftragsposten mit den jeweiligen Artikeln und Auftragsmengen auf (include "Auftragsposten ermitteln").

Bei schlechter Bonität kann der Auftrag noch an die spezielle Situation angepasst werden (extension point: Anpassen)

...



Wann include, wann extend?

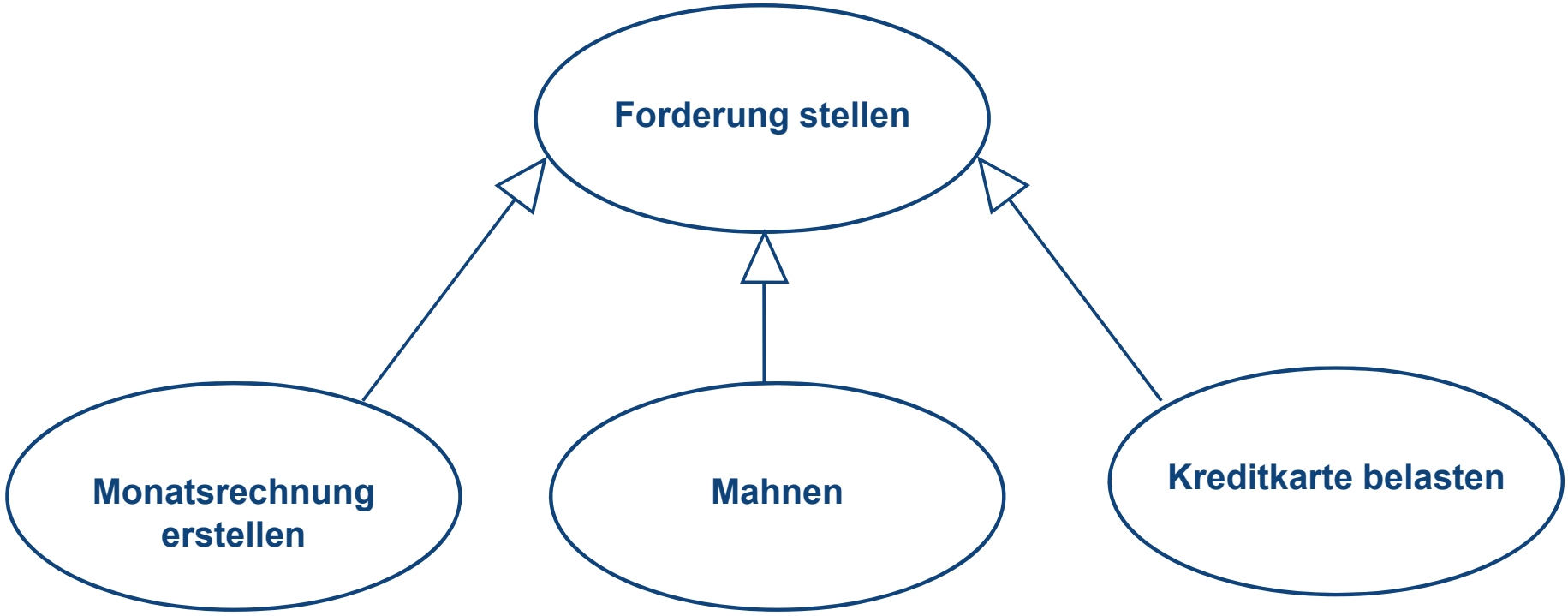
- Sowohl die include- als auch die extend-Beziehung fügen zusätzliche Funktionalität in eine Basisfunktionalität ein
- Im Falle der **include-Beziehung** komplettiert die zusätzliche Funktionalität die des Basisanwendungsfalls
  Beachte: Pfeil zeigt zum inkludierten Anwendungsfall
- Im Falle der **extend-Beziehung** ist sie optional, d.h. der Basisanwendungsfall ergibt auch ohne die zusätzliche Funktionalität ein sinnvolles Ergebnis
  Beachte: Pfeil zeigt zum Basis-Anwendungsfall
- Die include-Beziehung vermeidet Redundanz und erlaubt die Wiederverwendung von Teilfunktionalitäten, während die extend-Beziehung die Flexibilität und einfache Erweiterbarkeit des Anwendungsfallmodells sicherstellen soll
- In der UML bleibt die präzise Semantik der extend-Beziehung etwas unklar, in jedem Fall ist ihre Beschreibung (unangemessen) kompliziert

Generalisierung von Anwendungsfällen

- Anwendungsfälle können auch durch Generalisierungs-beziehungen verknüpft werden
  - Der spezialisierte Anwendungsfall umfasst die Aufgabe des allgemeinen Anwendungsfalls, bearbeitet aber einige Teilaufgaben auf eine spezielle Art und Weise
  - Der spezialisierte Anwendungsfall kann mindestens überall dort Anwendung finden, wo auch der allgemeine Anwendungsfall angewendet wird
- Bei der Generalisierungsbeziehung werden die Kommunikations-Beziehungen zwischen Akteuren und dem allgemeinen Anwendungsfall an die spezialisierten Anwendungsfälle “vererbt“
  - Absprache: Das gilt in MOOS auch bei include-Beziehungen und – oft, aber nicht immer – auch bei extend-Beziehungen
- include-Beziehung statt Generalisierung, wenn mehrere Anwendungsfälle gemeinsam konkrete Teil-Abläufe haben
  - Die Funktionalitäten der gemeinsamen konkreten Teil-Abläufe in eigene Anwendungsfälle fassen, die dann von den ehemaligen spezialisierten Anwendungsfällen inkludiert werden

Beispiel: Anwendungsfall-Generalisierung

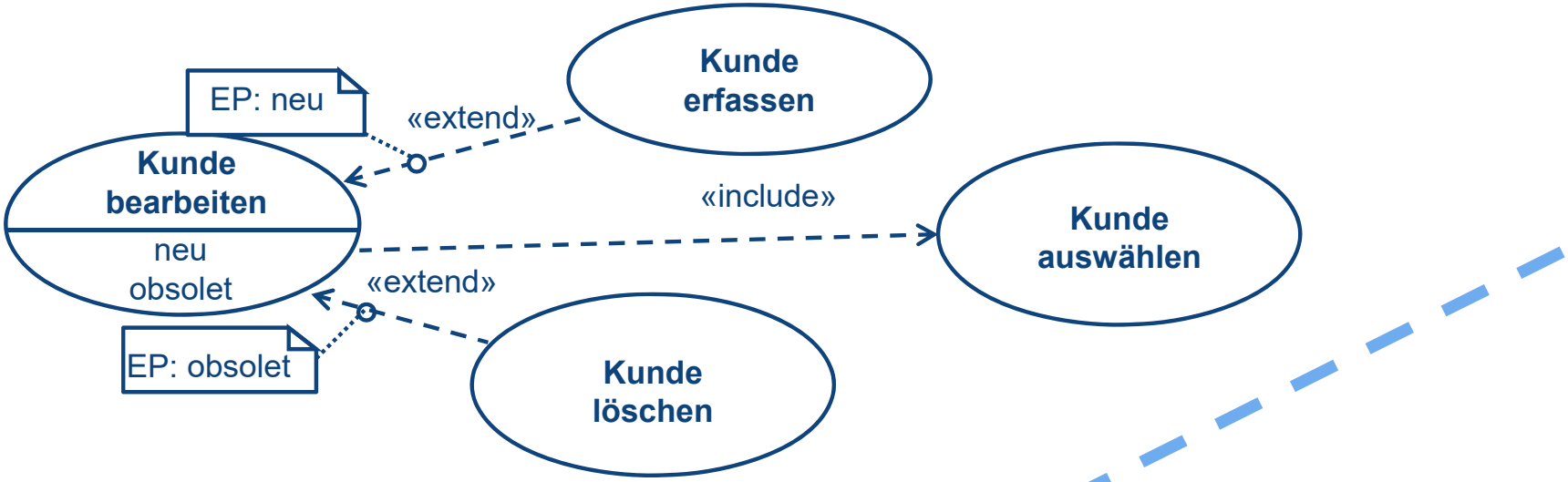
- In der Auftragsverwaltung wird die allgemeine Aufgabe “Forderung stellen“ durch die spezialisierten Aufgaben “Monatsrechung erstellen“, “Mahnung erstellen“ und “Kreditkarte belasten“ konkretisiert



Wo sind wir?

- Modellierung
- Akteure und Anwendungsfälle
- Beziehungen zwischen Anwendungsfällen
- **Aktivitätsmodellierung**

Granularität der Anwendungsfälle



So …

… oder so?

Es geht beides, aber …

- … die textuelle Beschreibung des grobgranularen Anwendungsfalls “Kunden verwalten” muss alle Möglichkeiten der Bearbeitung widerspiegeln, insbes.
  - Neuanlegen eines Kunden
  - Löschen eines Kunden
- Einfachheit des Diagramms auf Kosten einer komplizierten textuellen Beschreibung mit vielen Fallunterscheidungen und komplizierten unterschiedlichen Nachbedingungen
- Hier helfen **Aktivitätsdiagramme**!

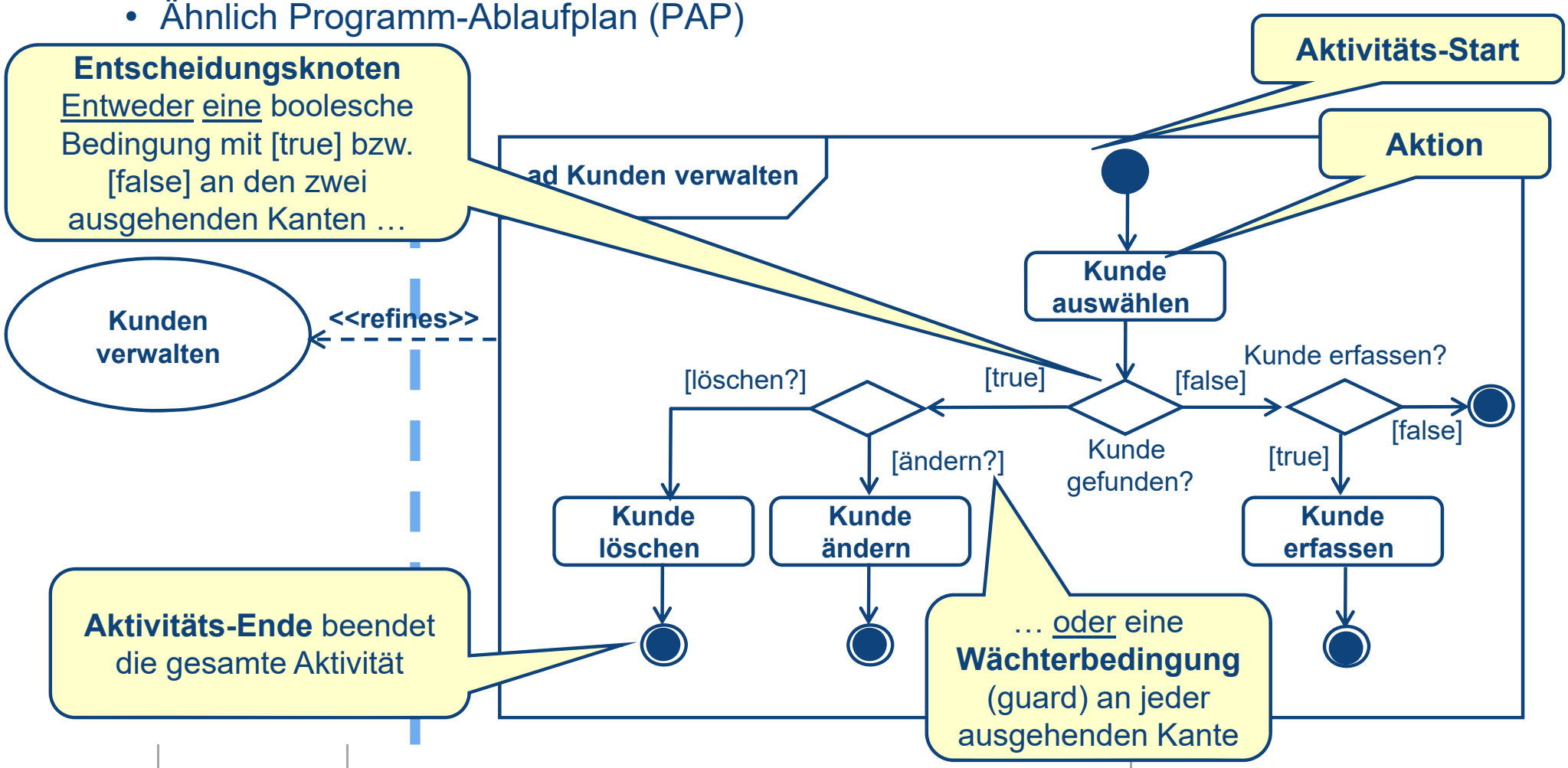
use case Kunden verwalten
actors AkteurA, AkteurB, ...
trigger …
precondition ...
main flow
....
alternative flow AF1
...
alternative flow AFn
...
postcondition ....

exceptional flow EF1
...
postcondition (von EF1)...

exceptional flow EFm
...
postcondition (von EFm)...
...
end Anwendungsfallname

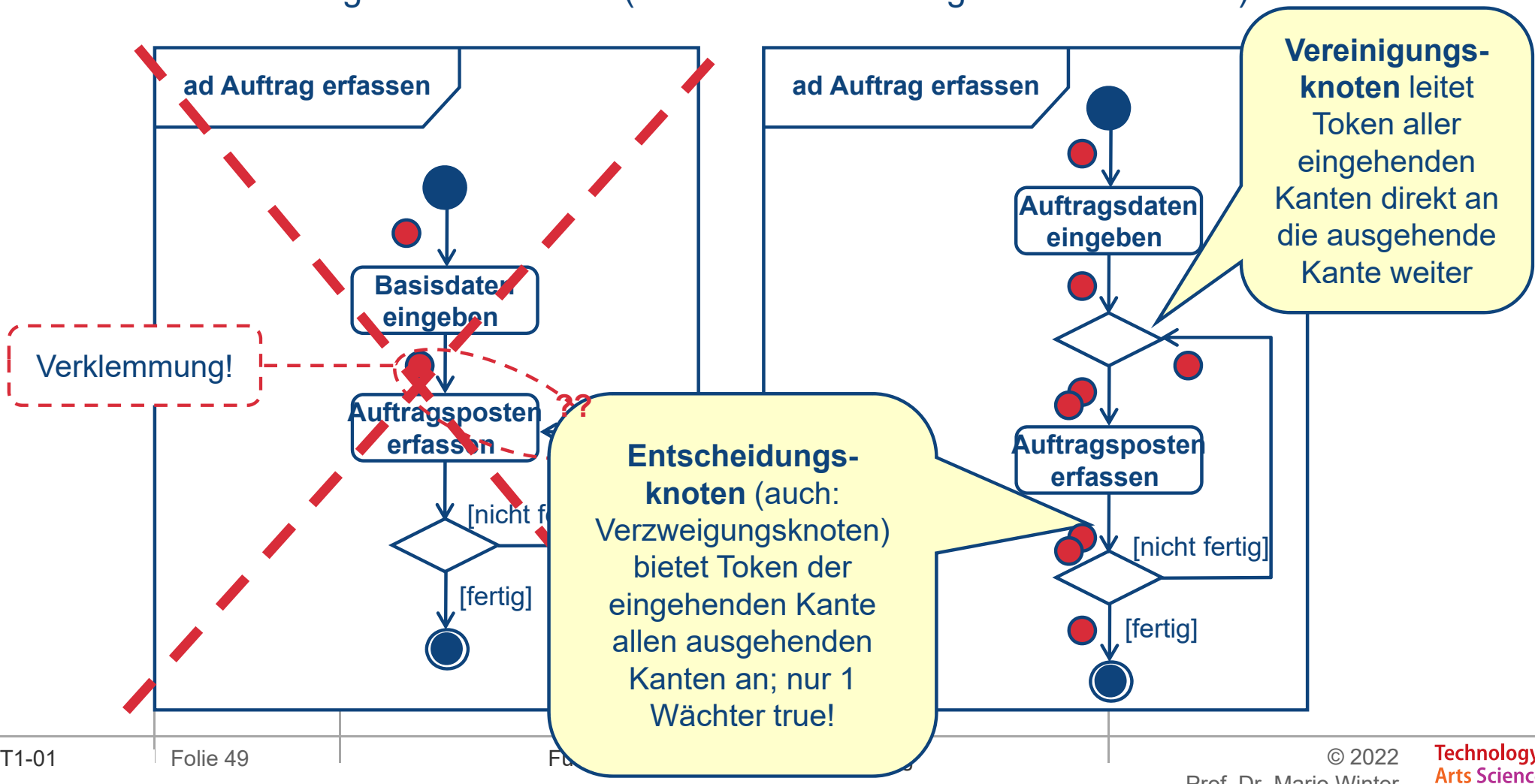
Aktivitätsdiagramm

- Modelliert Ablaufmöglichkeiten (z.B. von komplexen Anwendungsfällen)
- Ähnlich Programm-Ablaufplan (PAP)



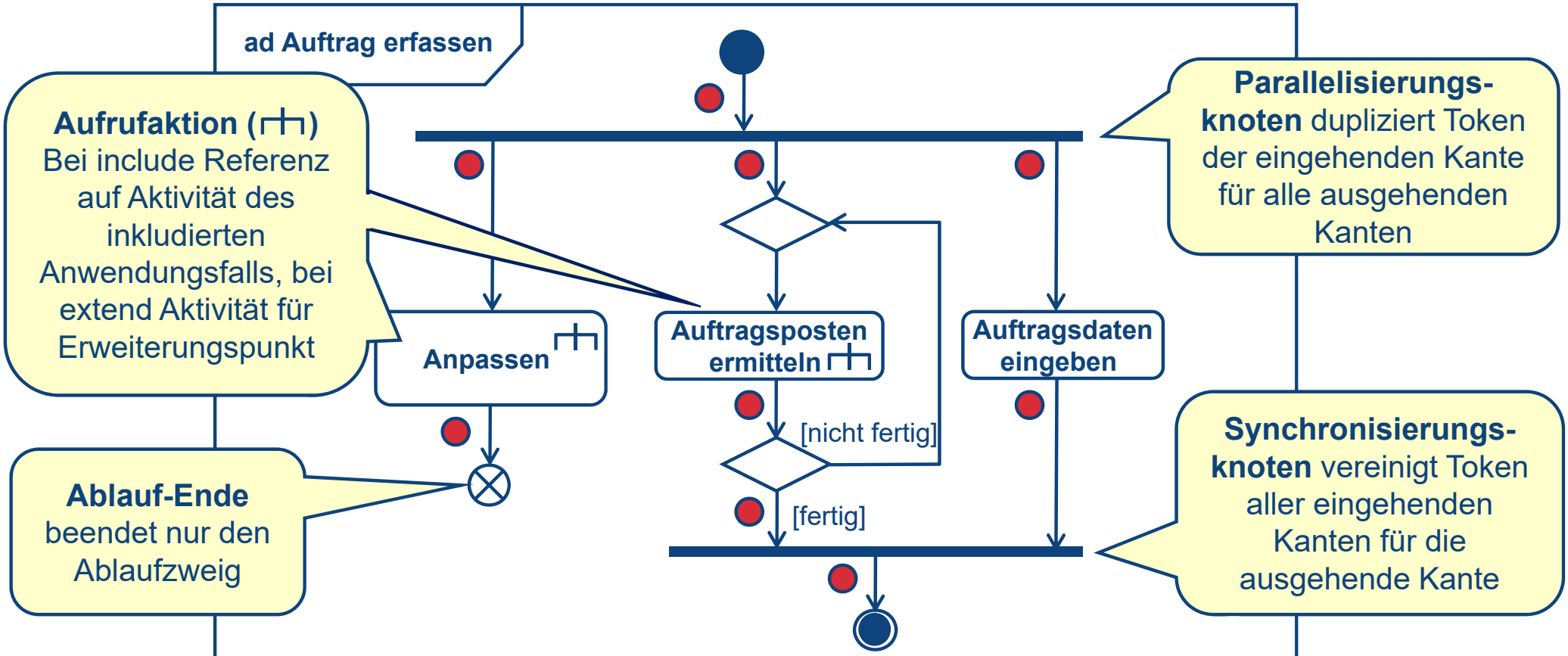
Aktivitätsdiagramm, Schleifen und „Token-Semantik“

- Eine Aktion startet nur dann, wenn alle unmittelbar vorangehenden Aktionen abgeschlossen sind (ähnlich der Schaltregel in Petri-Netzen)

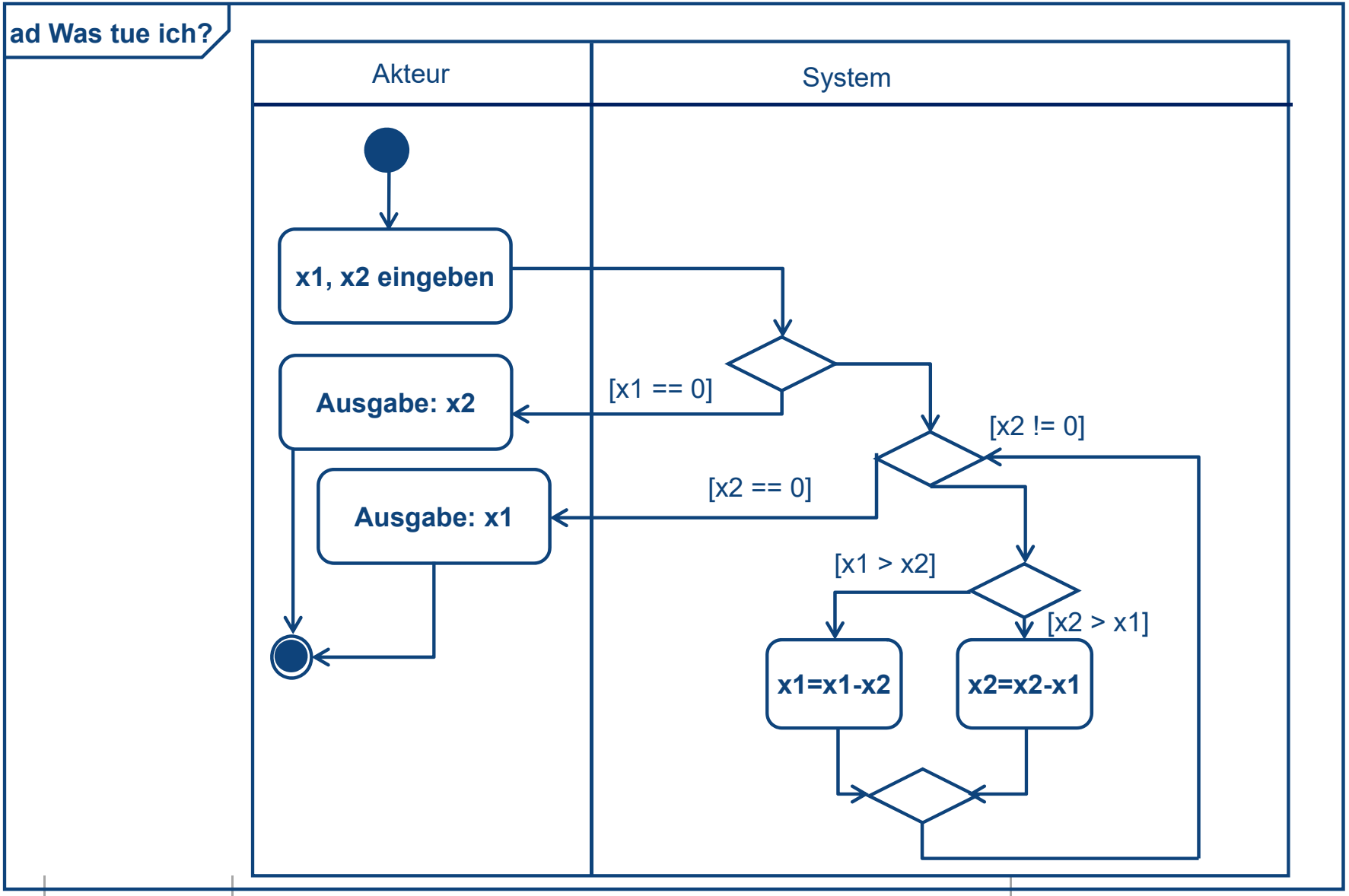


Aktivitätsdiagramm und parallele Abläufe

- Parallelisierungsknoten ermöglichen die Aufspaltung eines Ablaufs in mehrere parallel abarbeitbare Zweige



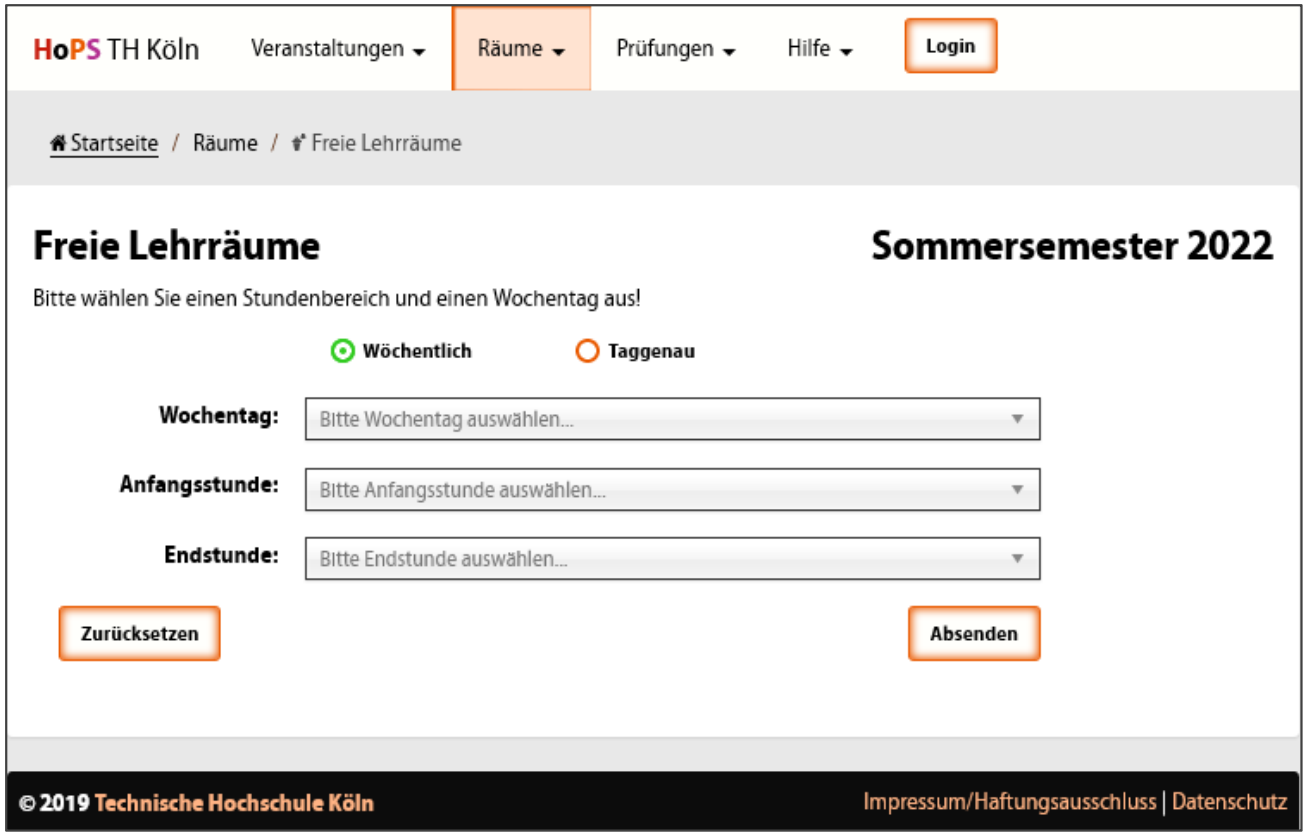
Was „tut“ dieses Aktivitätsdiagramm?

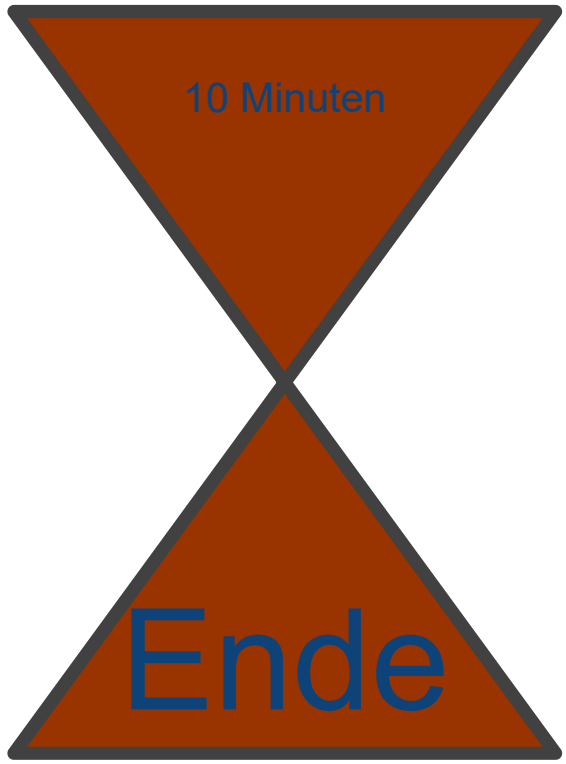


Max. 10 Minuten!

Aufgabe 3: Aktivitätsmodellierung

- Erstellen Sie ein einfaches Aktivitätsdiagramm für die Suche nach einem freien Lehrraum im HoPS

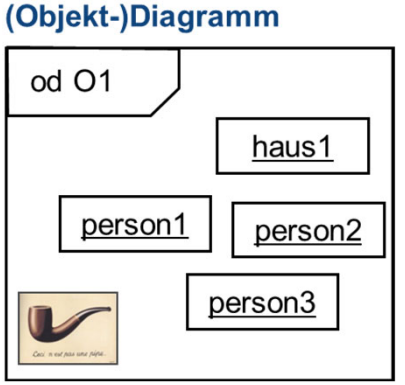




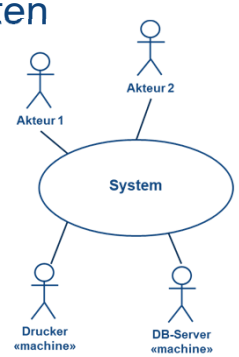
Lösungsidee Aufgabe 3: Aktivitätsmodellierung

Ein Modell ist eine aufgaben-angemessene, abstrahierende Sicht auf einen Gegenstand oder Sachverhalt

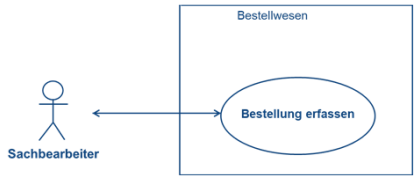
Zusammenfassung



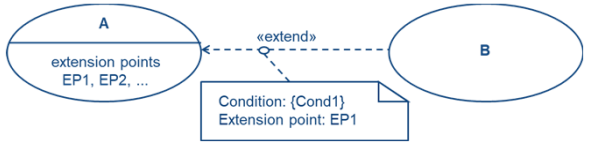
- Modellierung
  - Drei Modellkriterien: Abbildung, Abstraktion, Pragmatik
  - Drei Modellierungssichten: Funktion, Struktur und Verhalten
  - Modell vs. Diagramm
  - Unified Modelling Language
- Akteure und Anwendungsfälle
  - Systemkontext
  - Akteure (direkte Systemumgebung), kommunizieren mit
  - Anwendungsfällen (Funktionen des Systems)



Kontextdiagramm



- Beziehungen zwischen Anwendungsfällen
  - include und extend zwischen Anwendungsfällen
  - Generalisierung von Akteuren sowie Anwendungsfällen



- Aktivitätsmodellierung
  - Eine Aktivität beschreibt mögliche Ablaufreihenfolgen von Aktionen
  - Aktivitätsdiagramme u.A. zur Präzisierung komplexer Anwendungsfälle
  - Ähnlich Programm-Ablaufplan (PAP), aber Token-Semantik