Methodische und Praktische Grundlagen der Informatik (MPGI 3) WS 2008/09

Softwaretechnik

Steffen Helke

Andreas Mertgen (Organisation)

Rojahn Ahmadi, Georgy Dobrev, Daniel Gómez Esperón, Simon Rauterberg, Jennifer Ulrich (Tutoren)

Organisatorisches: Prüfungsanmeldung

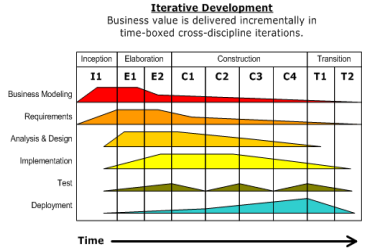
- **Anmeldezeitraum**
  01.11.08 bis 01.12.08
- **Letzte Rücktrittsmöglichkeit**
  30.11.08
- **Anmeldeverfahren**
  - **Bachelor**-Studierende über **QISPOS**
  - **Diplom**-Studierende über **Prüfungsamt**
    (falls offizielle Anmeldung erforderlich)
  - Anmeldungen in den Tutorien vorlegen

Was machen wir heute?

- **Objektorientierung**
  - vom Code zur Modellierung
  - zentrale Prinzipien von OO
- **Unified Modeling Language (UML)**
  - Welche Modelle gibt es?
  - Klassenmodelle
  - Anwendungsbeispiel
- **Requirements Engineering**

Wiederholung aus der letzten Vorlesung

- Eigenschaften von Software
  - essentiell und akzidentiell
- Eigenschaften von Prozessen
  - Reifegrade (CMMI)
- Beispiele für Prozesse
  - Wasserfallmodell
  - Spiralmodell
  - V-Modell-XT
  - **Rational Unified Prozess**



Komplexitätsreduktion durch den Einsatz von Objektorientierung

Abstraktion durch Struktur im Code

Selbstdefinierte Datenstrukturen

- **Klassen** als ein Datensatz (Records)
- Sammlung benannter Felder („Attribute“)

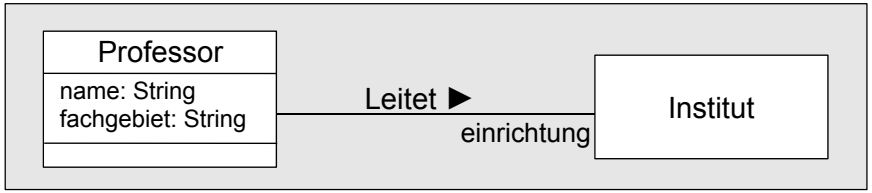
```
class Professor {
String name;
String fachgebiet;
Institut geleiteteEinrichtung;
}
class Institut {
// was steht hier alles?
}
```


Anwendung von Klassen

- Deklaration von Variablen
- Erzeugung von Instanzen/Objekten

```
class Professor {
String name;
String fachgebiet;
Institut geleiteteEinrichtung;
}
...
Professor sj = new Professor();
sj.name = "Jähnichen";
sj.fachgebiet = "Softwaretechnik";
```


Referenzen auf Objekte anderer Klassen




```
class Professor {
String name;
String fachgebiet;
Institut geleiteteEinrichtung;
}
...
Professor sj = new Professor();
sj.name = "Jähnichen";
sj.fachgebiet = "Softwaretechnik";
```

Prozeduren in Klassen

Operationen, „Methoden“

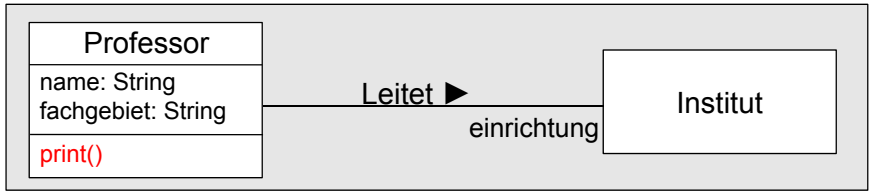
(Achtung: hat nichts mit Methodik zu tun!)

– implementieren das Verhalten der Objekte dieser Klasse

– haben ein verstecktes Argument
`this, self, Current` (je nach Sprache)

```
class Professor {
  void print() {
   System.out.println("Prof. " + this.name);
   System.out.println("Fachgebiet"+ this.fachgebiet);
  }
}
```


Methoden in Klassenbeschreibungen




```
class Professor {
  void print() {
    System.out.println("Prof. " + this.name);
    System.out.println("Fachgebiet"+this.fachgebiet);
  }
}
```


Komplexitätsreduktion durch ...

1. Zerlegen in **kleine Einheiten**
   (Zuständigkeiten)
2. Definieren von **Schnittstellen** zwischen Einheiten (Sichtbarkeiten)
3. **Wiederverwenden** vorgefertigter Teile (Baukastenprinzip)
4. Ausnutzen von **Abstraktionsebenen**
   (Vererbungshierarchien, Komposition)

Prinzipien der Objektorientierung

Ein OO-System ...

- ist eine dynamische Ansammlung miteinander kommunizierender Objekte

Ein Objekt ...

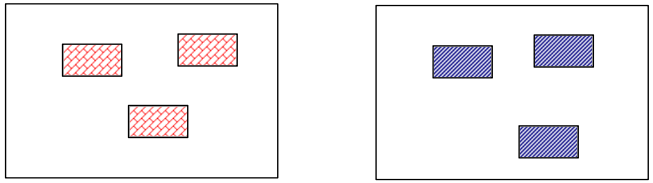
- hat inneren Zustand
  - dient der Datenkapselung –
    Zustand ist Privatsache des Objekts
  - Resultat von Operationen des Objekts
    hängt vom aktuellen Zustand ab.

Weitere Eigenschaften der Objekte

- **Definiertes Verhalten**
  - Festlegung durch Methoden des Objekts
  - Methodenaufruf durch empfangene Nachrichten
- **Eindeutige Identität**
  - Identität unabhängig von anderen Eigenschaften
  - verschiedene Objekte mit identischem Verhalten und identischem inneren Zustand möglich
- **Lebenszeit**
  - wird erzeugt und vernichtet

Typen, Klassen und Objekte

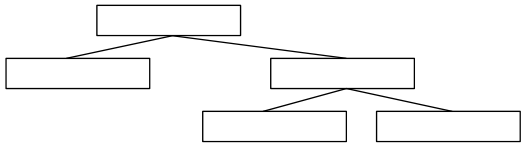
- Ein Objekt gehört zu einer **Klasse**
- Mehrere Objekte einer Klasse sind identisch bzgl. Verhalten und innerer Struktur



Eine Klasse bezeichnet den „Typ“
gleichartiger Objekte

Vererbungskonzept für Klassen

- Klassen besitzen einen „Stammbaum“
- Weitergabe von Eigenschaften durch **Vererbung**



Vererbung
- Spezialisierung einer Klasse zu einer Teilklasse bzw.
- Generalisierung zweier Teilklassen zu einer Klasse

Sammlungen von Klassen: Packages

- Packages
  – helfen Sichtbarkeiten zu regeln
  – bilden Namensräume
  – korrellieren mit Verzeichnissen

- Für große Projekte unverzichtbar
- Konzeptionell eher unbedeutend

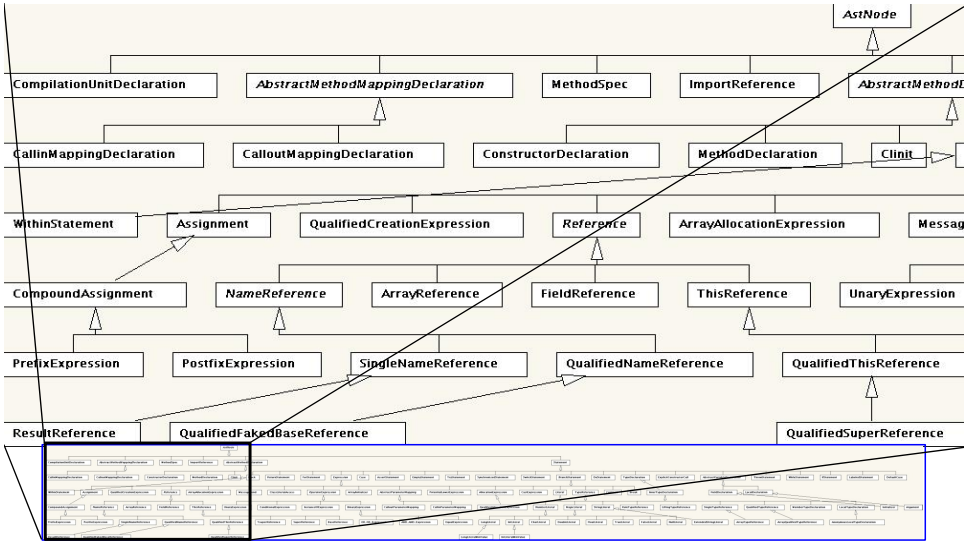
Beispiel: aktuelles Projekt

- **Forschungsaufgabe bei SWT**

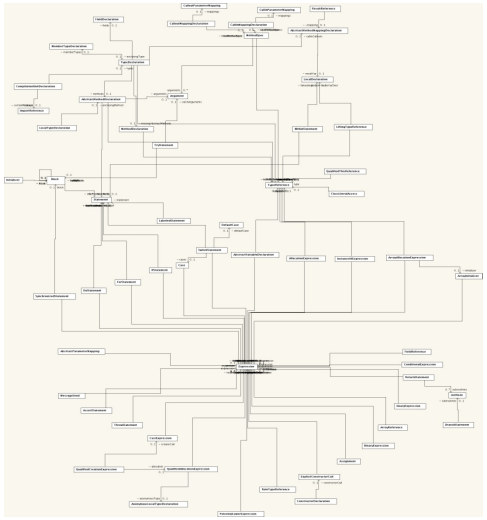
  Erweiterung eines Java-Compilers für die Sprache ObjectTeams/Java

- **Umfang (Java-Quelltext):**
  - > 120 kLOC
  - ca. 7,8 MByte Source
  - 374 Klassen (Dateien)
  - derzeit 1172 Testfälle (allein für Erweiterung)

Abstract Syntax Tree: Vererbung



Abstract Syntax Tree: Assoziationen



„Ein Bild sagt mehr als 1000 Worte“

73560 Worte passen nicht in ein Bild!

- **Zerlegung**
  - abstrakte Übersicht
  - Verfeinerung
- **Sichten**
  - beliebige Ausschnitte
  - thematischer Fokus

Modellierung mit der Unified Modeling Language

Unified Modeling Language (UML)

Sprache zur Modellierung objektorientierter Systeme

- entstanden als Vereinheitlichung älterer Notationen, wie OMT, OOSE und Booch
- Sprachmittel zum Bezeichnen von ...
  - **Dingen** (Struktur, Verhalten, Gruppierung, Annotation)
  - **Beziehungen** (Abhängigkeit, Assoziation, Generalisierung, Realisierung)
  - **Diagrammen** (Klassen, Objekte, Use Cases, Abfolgen, Zusammenspiel, Statecharts, Aktivitäten, Komponenten, Verteilung auf Hardware)

Geschichte der Unified Modeling Language

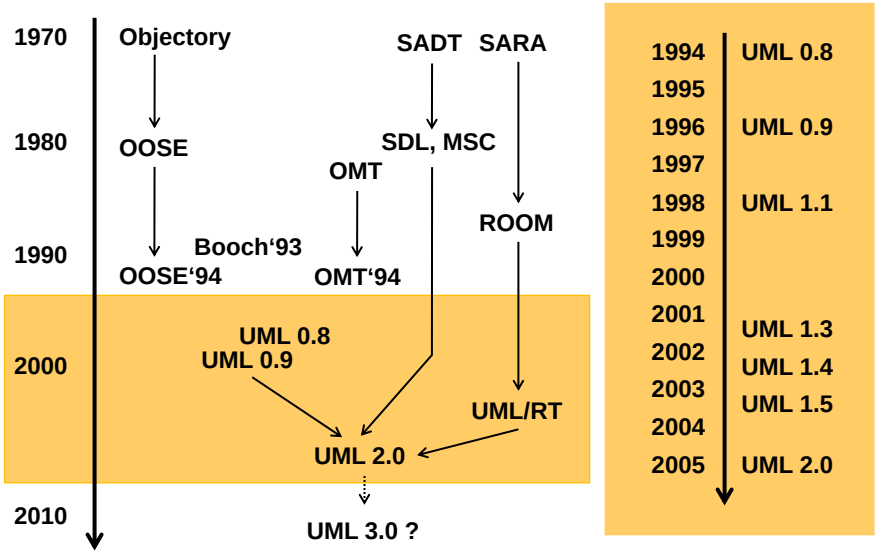


Abbildung nach Störle: *UML für Studenten*, Pearson-Verlag, 2005.

13 Diagramme in der Unified Modeling Language

- **Klassendiagramme**
- **Anwendungsfalldiagramme (Use-Case)**
- **Paketdiagramme**
- **Nutzfalldiagramme**
- **Montagediagramme**
- **Kollaborationen**
- **Installationsdiagramme**
- **Aktivitätsdiagramme**
- **Zustandsautomaten**
- **Zeitdiagramme**
- **Kommunikationsdiagramme**
- **Sequenzdiagramme**
- **Interaktionsdiagramme**

Einsatzgebiete der UML-Diagramme

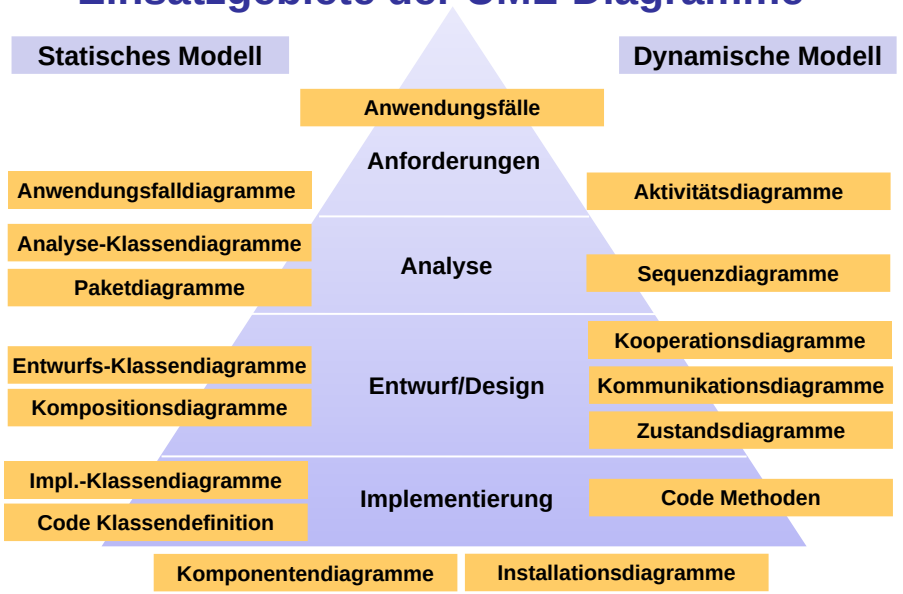
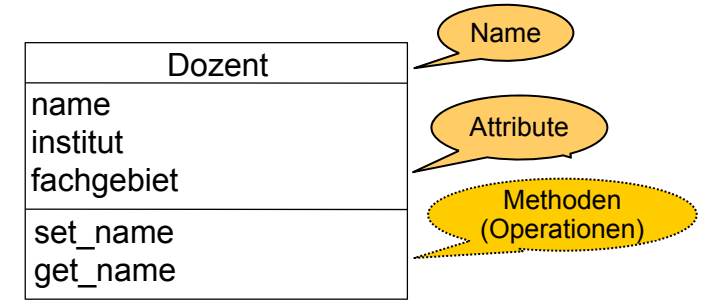


Abbildung nach Seemann, Wolff von Gudenberg: *Software Entwurf mit UML 2,* Springer-Verlag, 2006.

Notationen für UML-Klassendiagramme

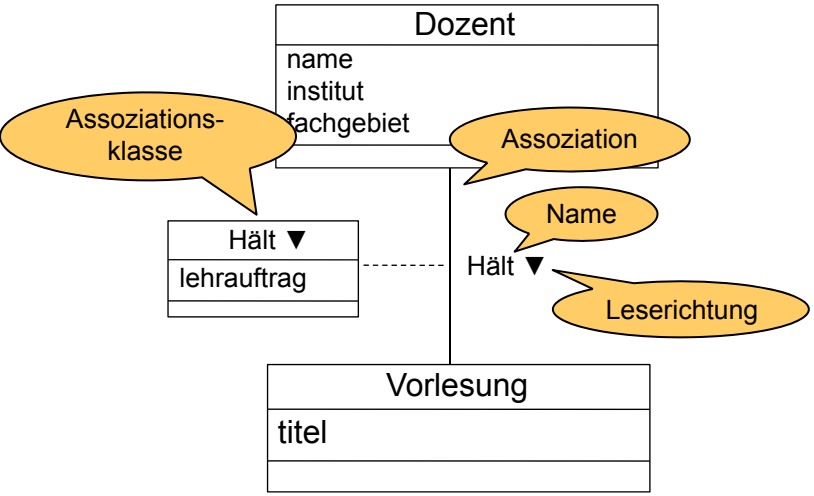
Modellierung von Klassen



Datenbanksicht auf Objekte (kein UML):

	Dozent		
id	name	institut	fachgebiet
..	Jähnichen	ISTI	SWT
..	Nestmann	ISTI	SWT
..	Helke	ISTI	SWT
..	Kondak	ITI	PDV

Klassendiagramme: Assoziationen

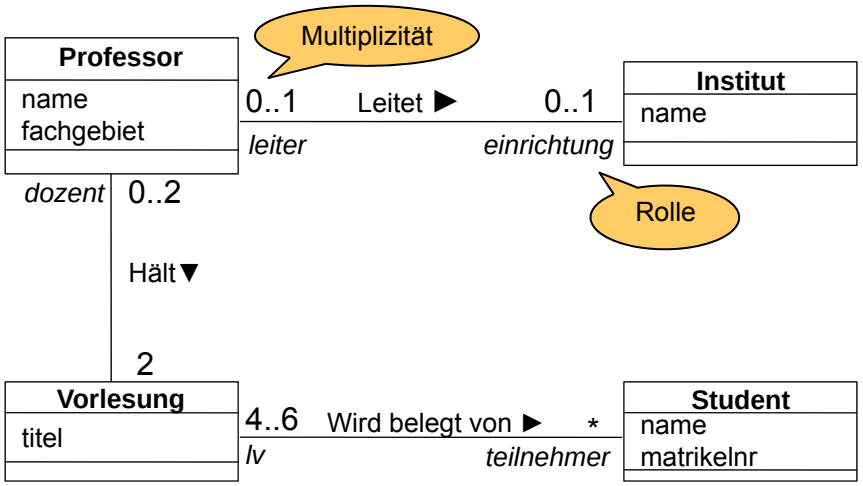


Assoziation als Relation

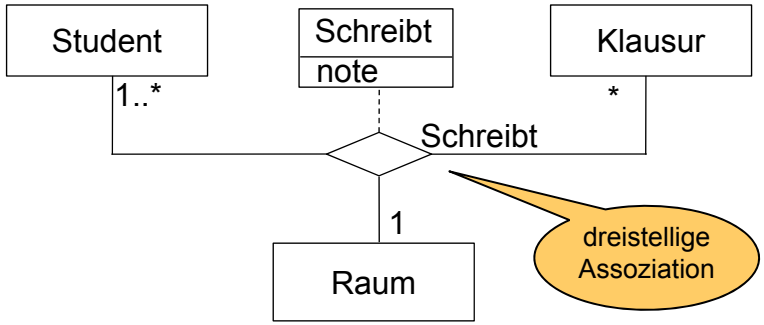
Hält	
Helke	MPGI 3
Herrmann	MWSE
Mosconi	SWT-Projekt
Helke	SASWT
Pepper	MPGI4

Hält $\in$ P (Dozent $\times$ Vorlesung)

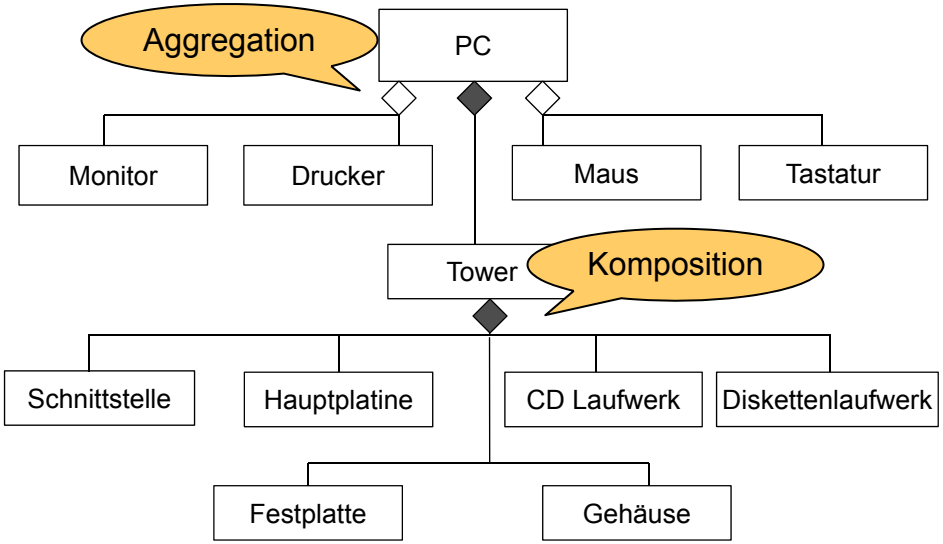
Multiplizitäten und Rollen



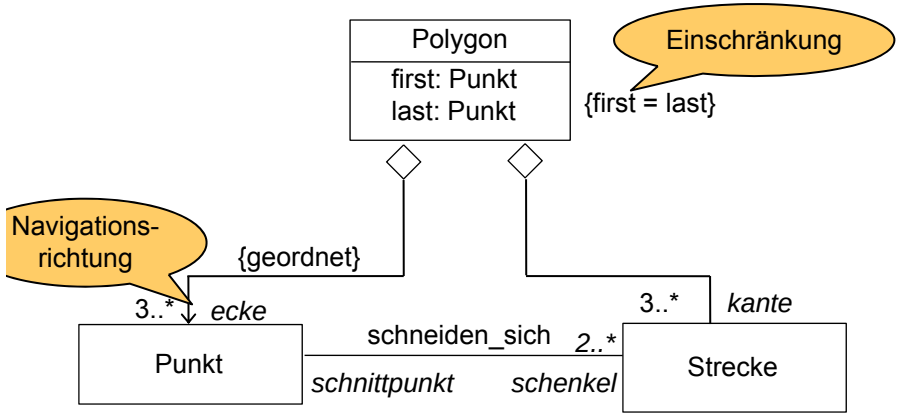
Mehrstellige Assoziation



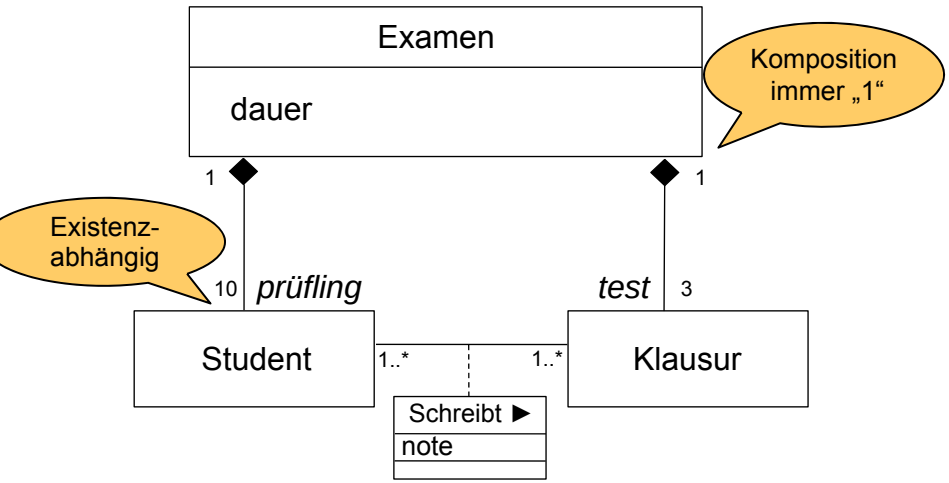
Aggregation und Komposition



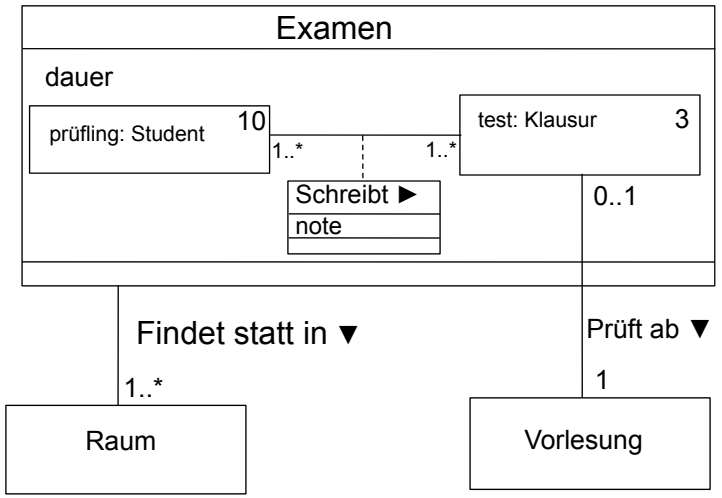
Einschränkungen und Navigation



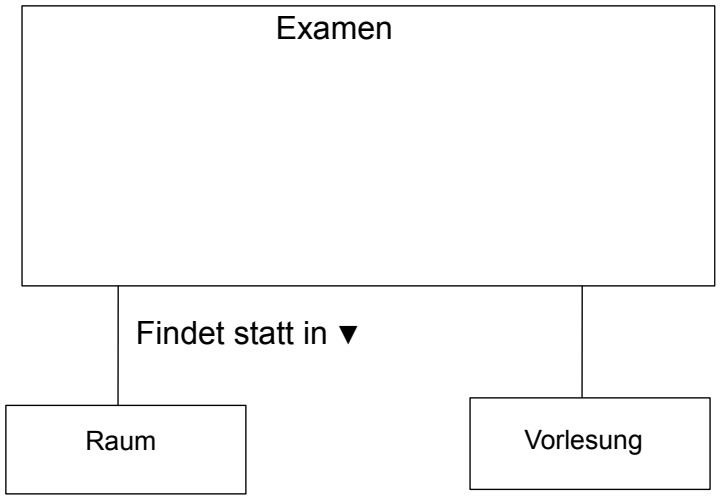
Komposition in der Normalform



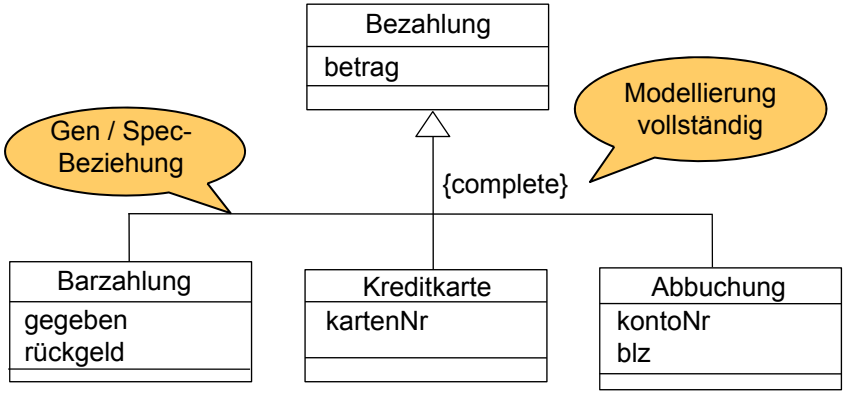
Komposition als Schachtelung



Komplexitätsreduktion durch Vergröberung

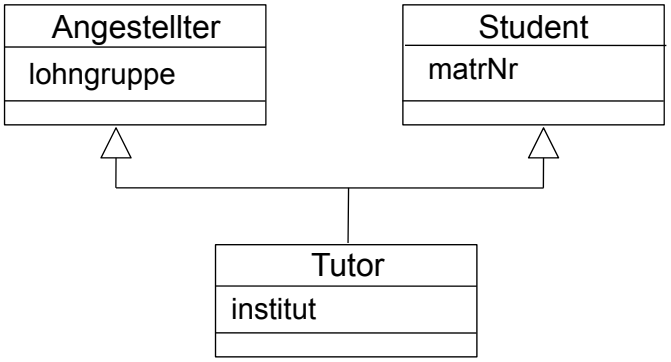


Generalisierung / Spezialisierung



Gen/Spec ist eine Beziehung zwischen ***Klassen***, nicht zwischen Objekten verschiedener Klassen!

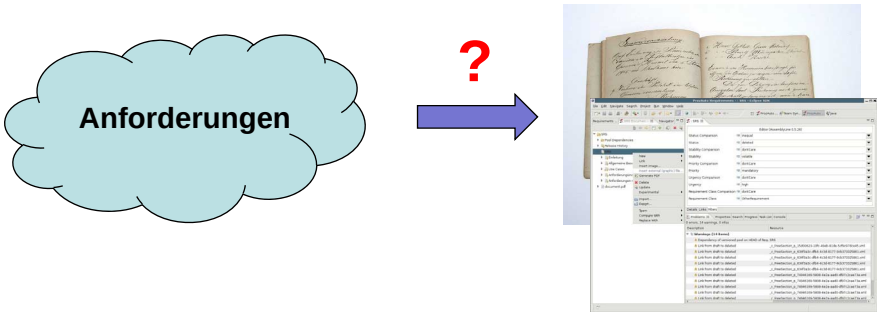
Mehrfachgeneralisierung



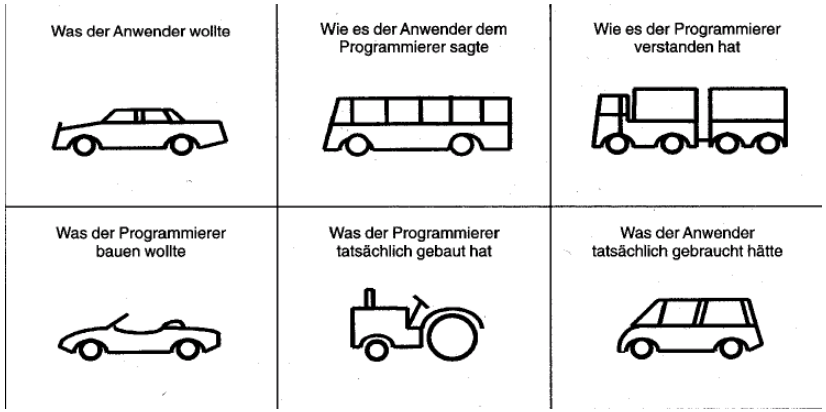
Was haben wir bis jetzt erreicht?

- Überblick Objektorientierung
- Einführung in die UML
- Abstraktion und Sichten
- Ausdrucksmittel für UML-Klassendiagramme

Wie kommt man zu seinen Anforderungen?



Entwicklung von Softwareprojekten



Wie kommt man zu Anforderungen?

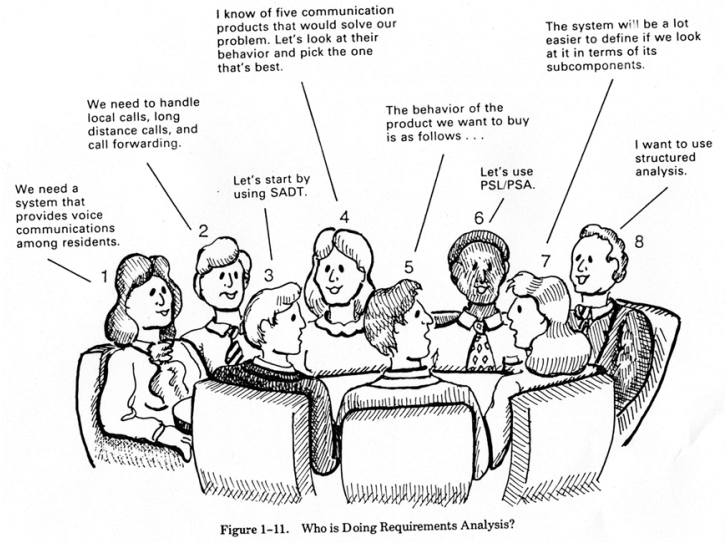


Figure 1–11. Who is Doing Requirements Analysis?

Definition Requirements

A ***user need*** or

a ***necessary feature, function or attribute***

of a system that can be sensed

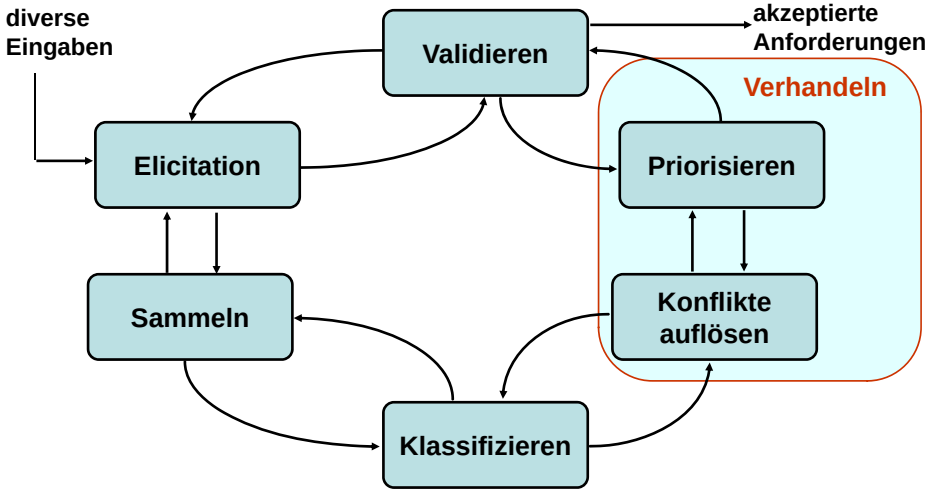
from a ***position external*** to that system

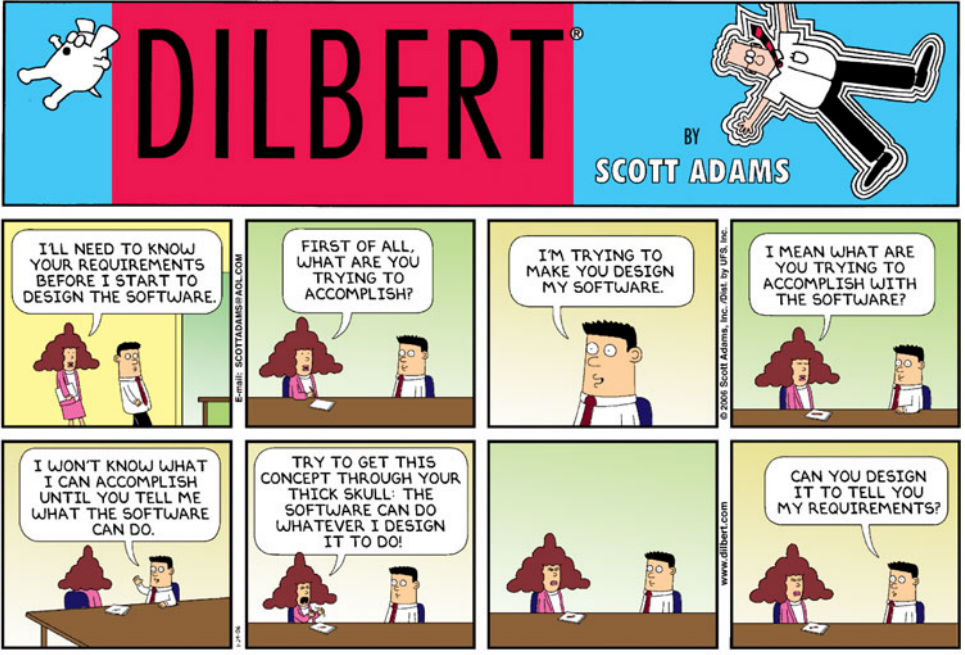
(Alan Davis)

Arten von Anforderungen

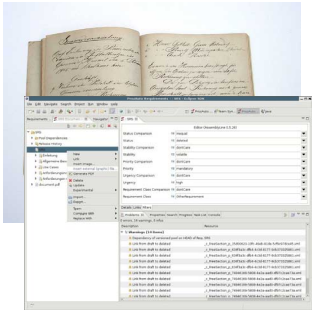
- **Funktionale** Anforderungen
  Aussagen über Dienste und Funktionalität, die das System leisten soll.
- **Nichtfunktionale** Anforderungen
  Anforderungen, die nicht die Funktionalität des Systems betreffen, sondern das gesamte System oder die Umgebung in dem das System eingebettet ist.
- **Problembereich**srequirements
  Kommen aus dem Anwendungsbereich des Systems.

Aktivitäten während der Anforderungsermittlung

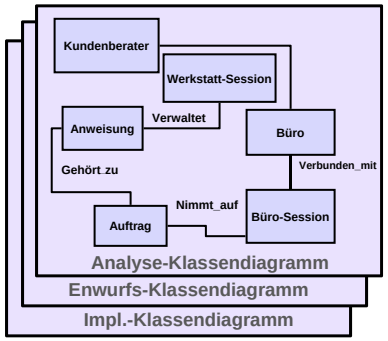




Wie kommt man zu Klassenmodellen?



?



Analyse ↔ Entwurf

Analyse:

„Was macht das System — und was macht es *nicht*?“

- Gegenstandsbereich
- Benutzungsdynamik
- Systemgrenzen

Entwurf:

„Wie erfüllt das System seine Aufgaben *intern*?“

- Kommunikation zwischen Objekten
- Zugriff auf Objekte
- (interne!) Klassenschnittstellen
- Vererbung

Modelle der Analyse-Phase

Klassenmodell (Gegenstandsbereich)

- Erfassung des (statischen) Gegenstandsbereichs (*domain of discourse*)
- Modellierung der Gegenstände, Eigenschaften und Beziehungen des Problems

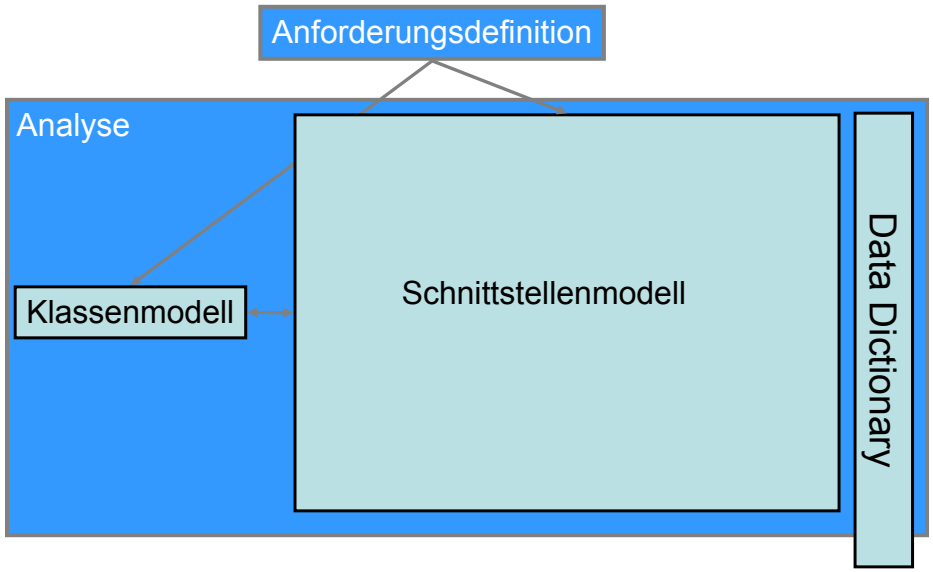
Schnittstellenmodell (z.B: Use-Case)

- Erfassung der Benutzungsdynamik
  - „Mit wem interagiert das System?“
  - „Was tut das System ?“

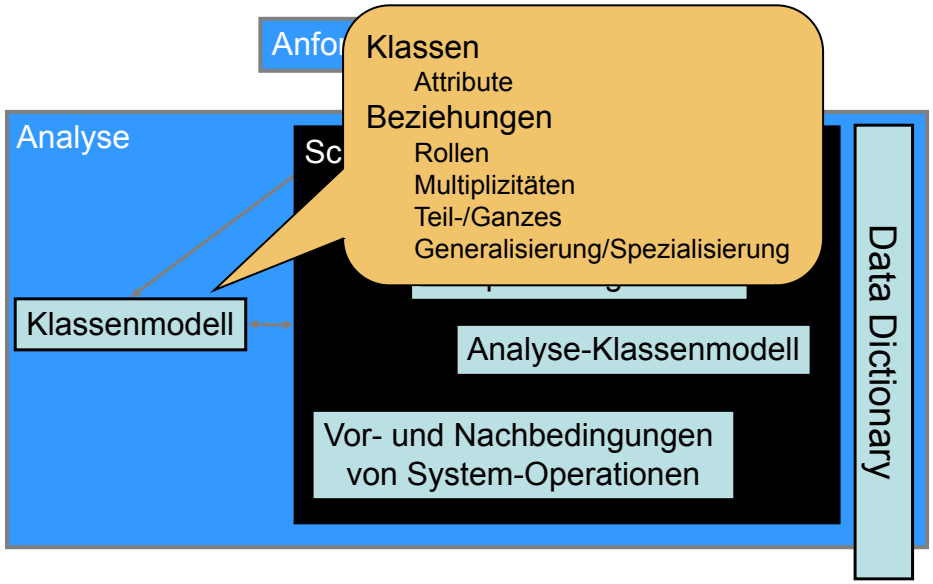
Analyse-Klassenmodell (Systemklassenmodell)

- Fixierung der Systemgrenzen

Analysemodelle



Analyse, Analysemodell: Klassenmodell



Aufstellen eines Klassenmodells

- **Modellierungselemente für das Klassenmodell**
  - Klassen, evtl. Attribute
  - Assoziationen, evtl. als Aggregation / Komposition
  - Generalisierung / Spezialisierung
  - Rollen
  - Multiplizitäten
- **Darstellungshilfen**:
  - Verfeinerung / Vergröberung
  - Mehrfachreferenzen in verschiedenen Diagrammen

Heuristiken zur Aufstellung eines Klassenmodells

- **Objekte und Klassen**
  - physische Objekte
  - Personen, Organisationen, Rollen
  - Vorgänge, Prozesse u.ä. Abstraktionen
- **Assoziationen**
  - Kommunikationen
  - Behälter-Inhalt-Beziehungen
  - Einschränkungen
  - Abstraktionen von Aktionen

Heuristiken zur Aufstellung eines Klassenmodells

– Klassen zunächst ohne Attribute

– Assoziationen und Klassen mit Multiplizität zuerst

– Relationen vor Attributen

– Invarianten ins Data Dictionary

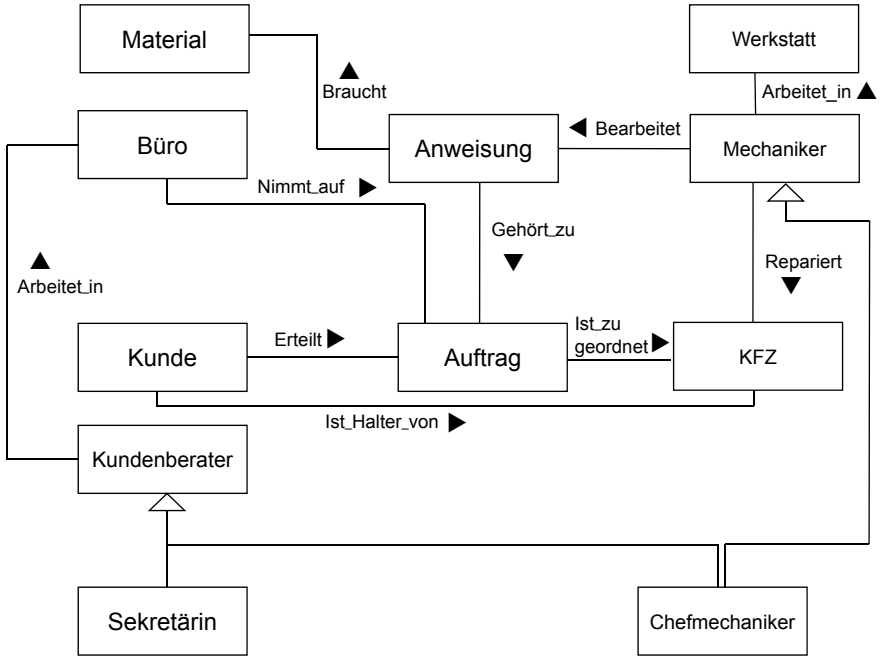
– Generalisierung zum Schluss

Beispiel: KFZ-Werkstatt

- Es soll ein Softwaresystem entwickelt werden, das die Geschäftsabläufe in einer Autowerkstatt verwaltet.

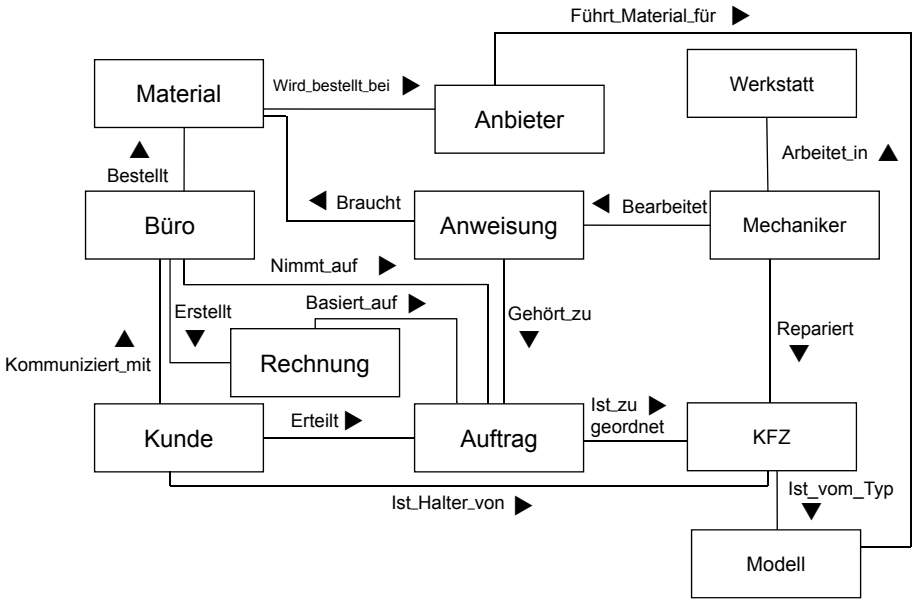
KFZ-Werkstatt : Anforderungen (1)

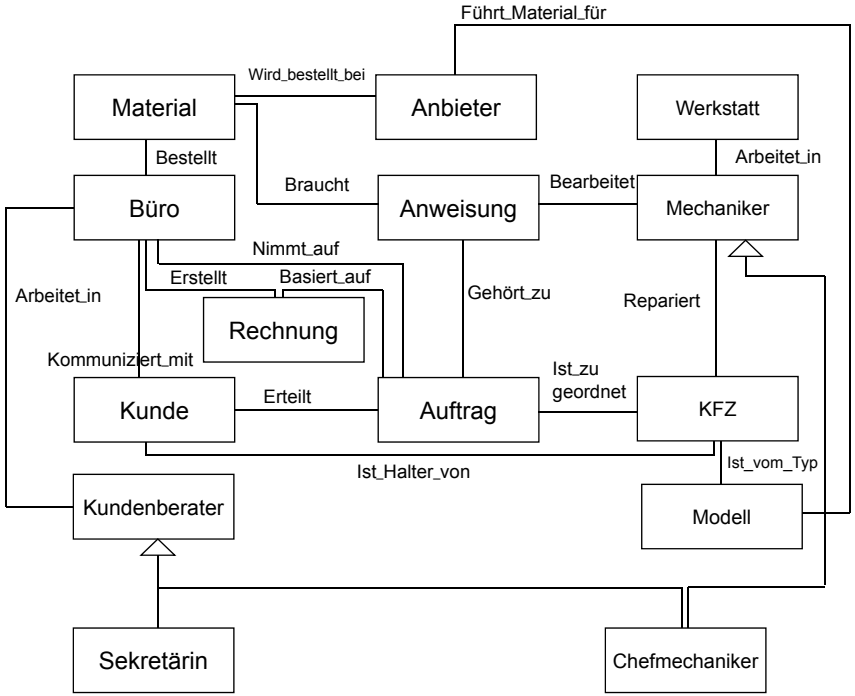
- Ein Kunde erteilt einen Reparaturauftrag für sein Auto. Der Auftrag wird im Büro von einer Sekretärin oder einem Chefmechaniker aufgenommen.
- Für jeden Auftrag wird zunächst eine Reparaturanweisung erstellt. Ein freier Mechaniker druckt diese an einem Terminal in der Werkstatt aus und übernimmt damit die Reparatur.
- Nach erfolgter Reparatur ergänzt der Mechaniker die Anweisung am Terminal um die Schadensbehebung (Arbeitsstunden und verbrauchtes Material). Zusätzlich festgestellte Schäden oder fehlendes Material werden in der Anweisung vermerkt und die weitere Vorgehensweise wird ans Büro verwiesen.

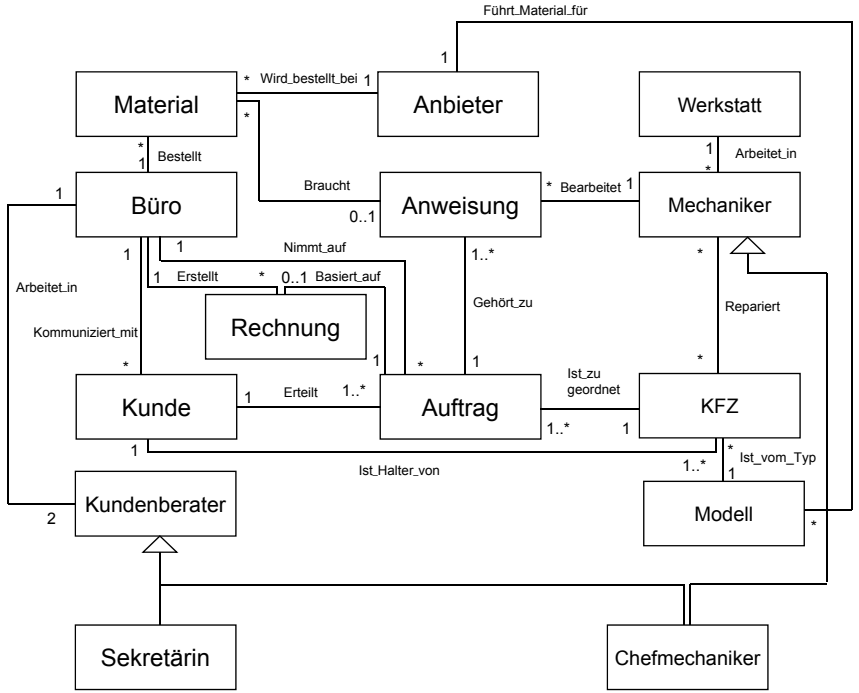


KFZ-Werkstatt : Anforderungen (2)

- Zusätzlich festgestellte Schäden werden nur mit der Zustimmung des Kunden repariert. Der Auftrag wird dabei um eine neue Reparaturanweisung erweitert. Der Prozess der Bearbeitung eines Auftrags kann so mehrere Zyklen durchlaufen.
- In der Werkstatt können Autos verschiedener Modelle bearbeitet werden. Bei fehlendem Material und nach Bestätigung durch einen Kundenberater wird eine Bestellung bei dem jeweiligen Anbieter über das Internet aufgegeben.
- Nach erfolgreichem Abschluss des Auftrags, wird der Kunde benachrichtigt und es wird eine Rechnung auf Basis der gesamten Arbeitsstunden und des verbrauchten Materials erstellt.







Konventionen

- **Klassen**:
  - Großer Anfangsbuchstabe,
  - Substantiv
- **Assoziationen**:
  - Großer Anfangsbuchstabe,
  - Zusammengesetzte Bezeichner durch „_“ trennen,
  - Verben im Singular verwenden

Überprüfen der Lesbarkeit

Klasse & Assoziation & Klasse = Satz