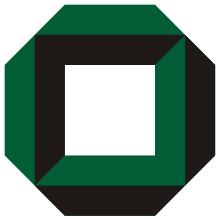


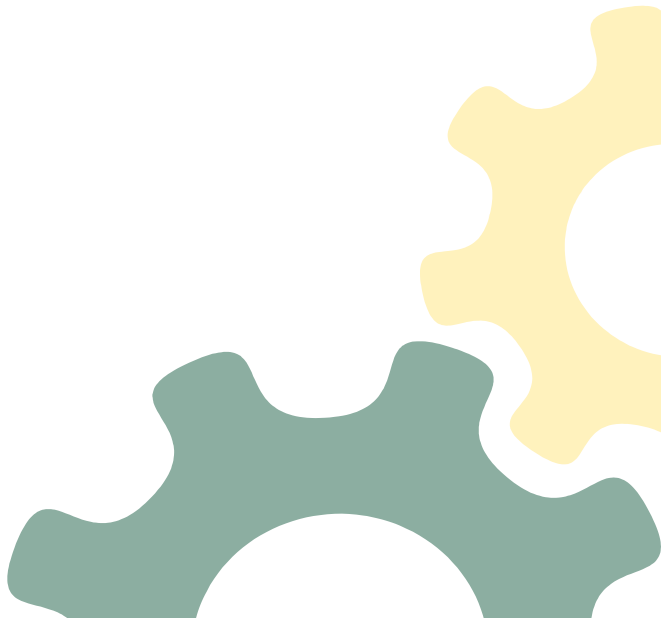
Universität Karlsruhe (TH)
Forschungsuniversität · gegründet 1825

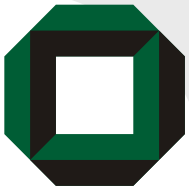
Kapitel 2.3
Objektmodellierung
Wie komme ich zu einem guten Modell?

SWT I – Sommersemester 2009

Prof. Dr. Walter F. Tichy

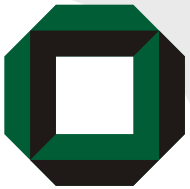
Fakultät für **Informatik**
Lehrstuhl für Programmiersysteme





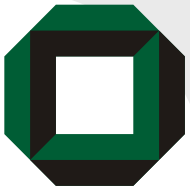
Wo sind wir überhaupt? Ich hab die Orientierung verloren...

- Planungsphase: funktionales Modell
  - Was sind die Szenarien, Anwendungsfälle?
  - Dokumentiert im Pflichtenheft
- Wie macht man das?
  - Frage Kunden, Auftraggeber
  - Studiere Arbeitsprozesse, Formulare, existierende Systeme
  - Studiere Anwendungsbereich, benutze eigenes Weltwissen



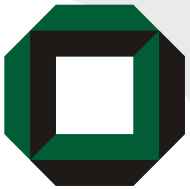
Wir sind doch jetzt in der Definitionsphase!

- Jetzt kommt die **Objektmodellierung** dazu:
  - Was sind die Klassen?
  - Welche Assoziationen bestehen? (einschl. Kardinalitäten)
  - Welche Attribute werden gebraucht?
  - Welche Methoden?
  - Zuletzt: entwickle Vererbungsstrukturen
    - Gemeinsamkeiten von Klassen herausziehen und in Oberklassen legen
    - Substitutionsprinzip beachten.
- Objektmodell ist ein **statisches Modell**



Und wozu braucht man die übrigen Diagrammtypen?

- **Dynamisches Modell**
  - Aktivitätsdiagramm
    - Beschreibe parallele und sequenzielle Abläufe
  - Sequenzdiagramm
    - Identifiziere Methoden gesendet zwischen Objekten bestimmter Klassen
    - (ergänze Klassendiagramm um neue Methoden)
    - Zeige Aufrufsabläufe zwischen mehreren Objekten
  - Zustandsdiagramm
    - Zustandsübergänge innerhalb eines *einzelnen* Objektes



»So finden Sie Klassen« Finden der Kandidaten

- Lassen sich konkrete Objekte identifizieren?
  - Bei technischen Systemen bieten sich die realen Objekte als Ausgangsbasis an
  - Bei kommerziellen Systemen finden sich Objekte oft in Formularen.
- Welche Abstraktionen gibt es im bisherigen (Software-)System?
  - Formulare von Verträgen oder Altsystemen analysieren
  - Attribute entnehmen und zu Klassen zusammenfassen
  - (Das ist eine „bottom-up“-Vorgehensweise)

Beispiel »Seminarorganisation«

Anmeldung zu TEACHWARE-Seminaren

Als Teilnehmer zu nachfolgenden TEACHWARE-Seminaren wird angemeldet:

Titel | Vorname | Name

Veranstaltungs-Nr. | Seminarbezeichnung | vom | bis

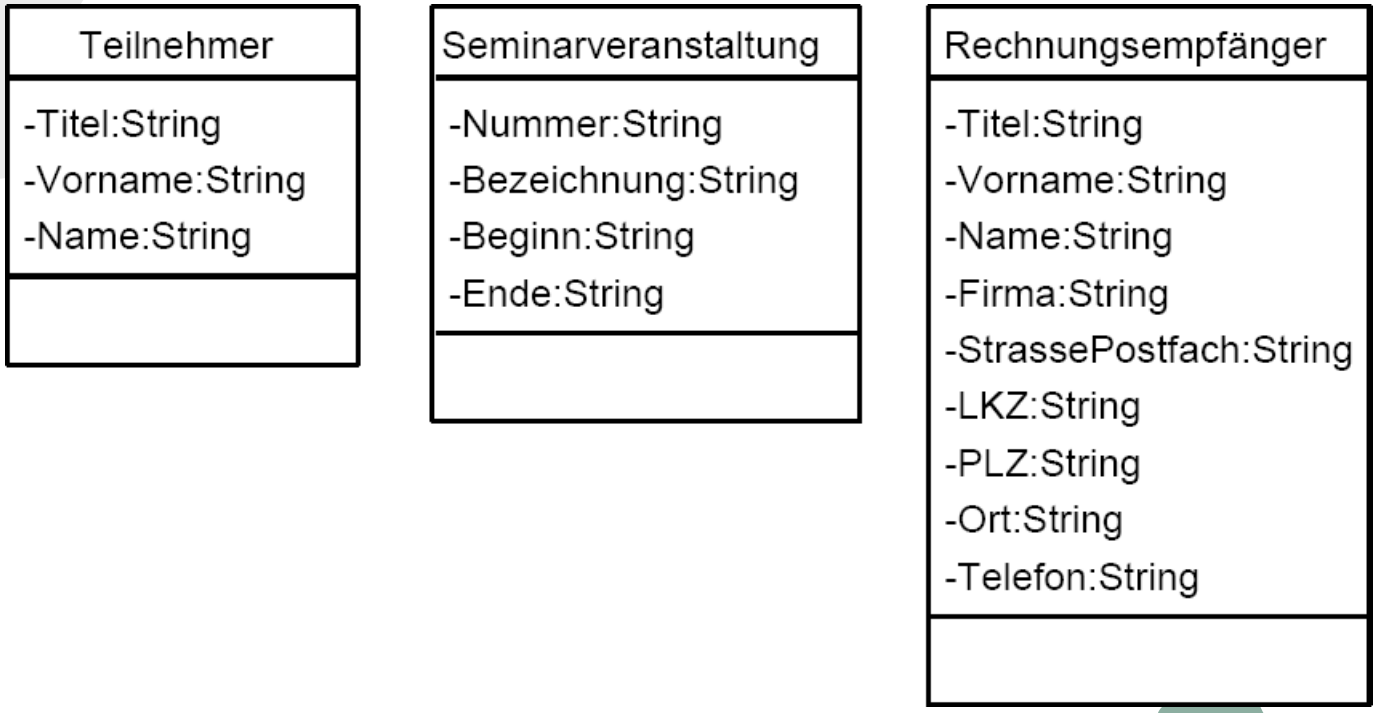
Anmeldebestätigung und Rechnung erbeten an:

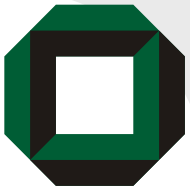
Titel | Vorname | Name

Firma | Straße/Postfach | LKZ

PLZ | Ort | Telefon

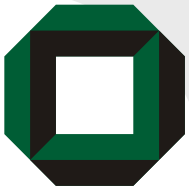
Beispiel »Seminarorganisation«





»So finden Sie Klassen« Finden der Kandidaten

- Dokumentanalyse: Klassen aus Szenarien, Anforderungen identifizieren (top-down)
  - Syntaktische Analysen
    - Nomen-Verb-Analyse (Abbott)
  - Linguistische Analyse nach thematischen Rollen (siehe 2.5)
  - Inhaltliches Durchforsten nach
    - Attributen für potenzielle Klassen
    - Akteuren, über die man sich etwas merken muss (sind potentielle Klassen)



Beispiel

„Der Aufzug schließt seine Tür bevor er zu einem anderem Stockwerk fährt.“

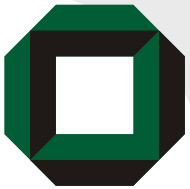
- Potentielle Klassen: Aufzug, Tür, Stockwerk
  - Modelliere Aufzugstür als Klasse, wenn sie eigene Operationen besitzt
  - Wenn Operationen von anderen Operationen anderer Datentypen abgedeckt sind, z.B. Aufzug, dann keine eigenständige Klasse
  - Untersuche Beschreibung nach weiteren Hinweisen

Zur linguistischen Analyse

- Abbott (1983): Wortarten indizieren Modellelemente

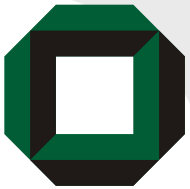
Wortart	Modellelement	Beispiel
Nomen	Klasse	Auto, Hund
Namen	Exemplar	Peter
Intransitives Verb	Botschaft	laufen, schlafen
Transitives Verb	Assoziation	*etw.* essen, *jmd.* lieben
Verb „sein“	Vererbung	ist eine (Art von)…
Verb „haben“	Aggregation	hat ein…
Modalverb	Zusicherung	müssen, sollen
Adjektiv	Attribut	3 Jahre alt

- Nur als erste Annäherung brauchbar!



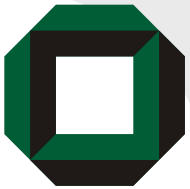
»So finden Sie Klassen« Bewertung der Kandidaten

- Wann liegt (wahrscheinlich) keine Klasse vor?
  - Es lassen sich weder Attribute noch Operationen identifizieren.
  - Eine Klasse enthält dieselben Attribute, Operationen, Restriktionen und Assoziationen/Aggregationen wie eine andere Klasse.
  - Eine Klasse enthält nur Operationen, die sich anderen Klassen zuordnen lassen.
  - Eine Klasse modelliert Implementierungsdetails.
  - Besitzt eine Klasse nur ein einziges oder wenige Attribute, so ist zu prüfen, ob diese Attribute einer anderen Klasse zugeordnet werden können.



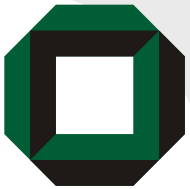
»So finden Sie Klassen« Auswählen und Festlegen

- Liegt ein aussagefähiger Klassenname vor?
- Der Klassenname soll
  - ein Substantiv im Singular sein,
  - so konkret wie möglich gewählt werden,
  - Gesamtheit der Attribute und Operationen darstellen
  - nicht die Rolle beschreiben, die diese Klasse in einer Beziehung zu einer anderen Klasse spielt.



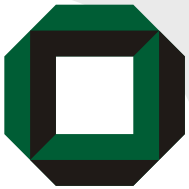
»So finden Sie Klassen« Auswählen und Festlegen

- Allgemeine Hinweise
  - Noch keine Vererbungsstrukturen bilden
  - Eine Klasse kann auch nur ein Objekt besitzen
  - Im Zweifelsfall kleinere Klassen bilden.



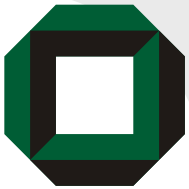
Schritt 2: »So finden Sie Assoziationen«

- Liegen zwischen Objekten dauerhafte Beziehungen vor?
  - Existieren sie für einen längeren Zeitraum?
  - Sind die Assoziationen problemrelevant?
  - Existiert die Beziehung unabhängig von allen nicht beteiligten Klassen?



»So finden Sie Assoziationen«

- Verben in der Problembeschreibung?
  1. räumliche Nähe (wohnt in)
  2. Aktionen (fährt)
  3. Kommunikation (redet mit)
  4. Besitz (hat)
  5. allgemeine Beziehungen (verheiratet mit).

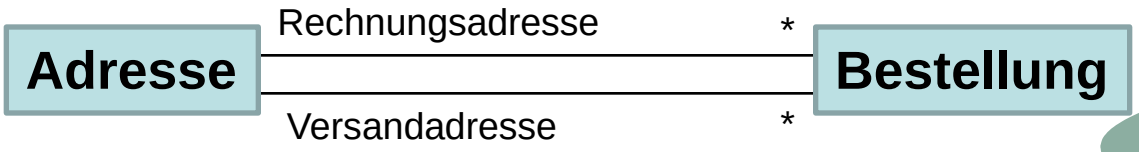


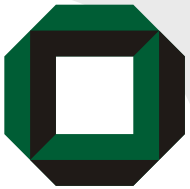
»So finden Sie Assoziationen«

- Sind die beteiligten Klassen gleichrangig?
  - Falls Über/Unterordnung, Aggregation prüfen
  - Im Zweifelsfall Assoziation wählen
- Müssen die Objekte der Klassen miteinander kommunizieren?
  - Ist der Informationsfluss uni- oder bidirektional?

»So finden Sie Assoziationen«

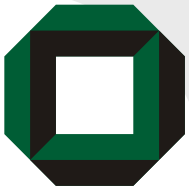
- Mehrere Assoziationen zwischen zwei Klassen?
- Prüfen, ob die Assoziationen
  - eine unterschiedliche Bedeutung besitzen
  - unterschiedliche Kardinalitäten haben
- Assoziationen, die keine neue Information hinzufügen, vermeiden.





»So finden Sie Assoziationen«

- Welche Rollen spielen die beteiligten Klassen?
- Je allgemeiner der Klassenname, desto wichtiger der Rollenname!
- Rollennamen angeben, wenn
  - Mehrere Assoziationen zwischen zwei Klassen existieren
  - Eine Assoziation zwischen Objekten derselben Klasse existiert
  - eine Klasse in verschiedenen Assoziationen auch verschiedene Rollen spielt
  - durch den Rollennamen die Bedeutung der Klasse in der Assoziation genauer spezifiziert werden kann.

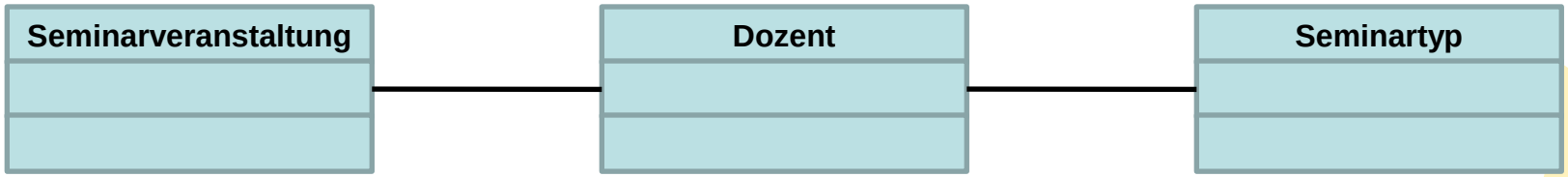


Zulässige Kardinalitäten

- Liegt eine Muss-Beziehung vor?
  - Sobald das Objekt erzeugt ist, muss auch die Beziehung zu dem anderen Objekt aufgebaut werden
- Liegt eine Kann-Beziehung vor?
  - Die Beziehung kann zu einem beliebigen Zeitpunkt nach dem Erzeugen des Objekts aufgebaut werden.
- Ist die Obergrenze fest oder variabel?
  - Ist eine Obergrenze vom Problem her zwingend vorgegeben?
  - Im Zweifelsfall mit variablen Obergrenzen arbeiten

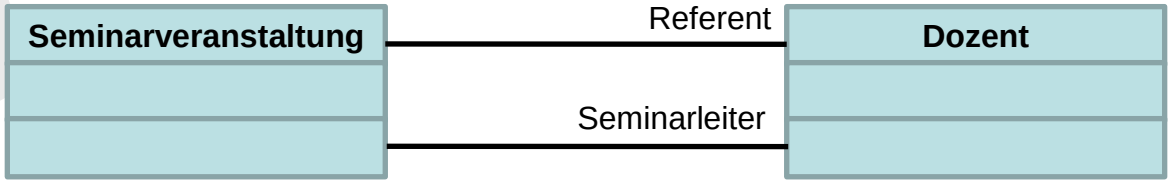
Beispiel: Finden von Assoziationen

- Anhand des Pflichtenheftes lassen sich folgende Beziehungen ermitteln:
  - Ersterfassung, Änderung und Löschung von Dozenten sowie Zuordnung zu Seminarveranstaltungen und Seminartypen /F180/

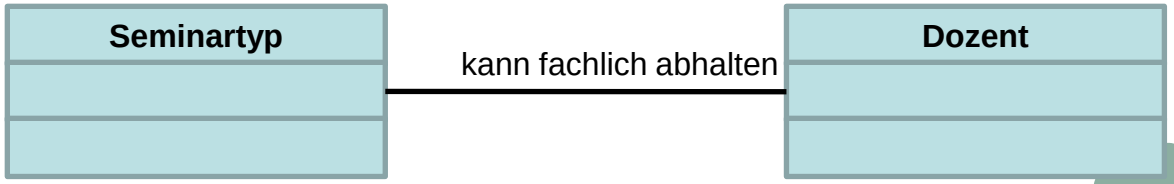


Beispiel: Finden von Assoziationen

- In Bezug auf Seminarveranstaltungen spielt kann ein Dozent folgende Rollen einnehmen:

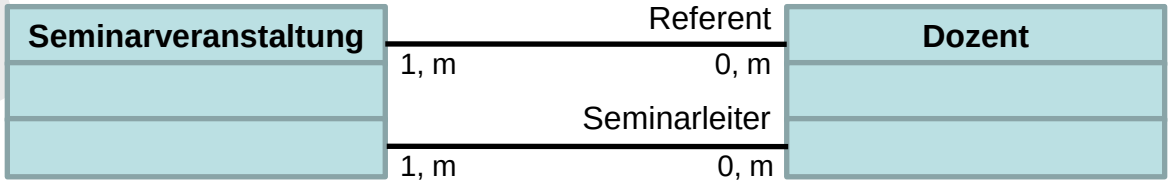


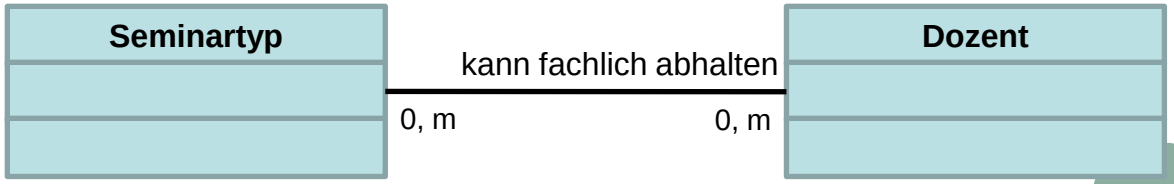
- Der Zusammenhang zwischen Dozent und Seminartyp kann durch eine Benennung verständlicher gemacht werden:

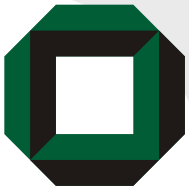


Beispiel: Finden von Assoziationen

- Für diese Beziehungen können folgende Kardinalitäten ermittelt werden:

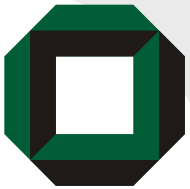






»So finden Sie Aggregationen«

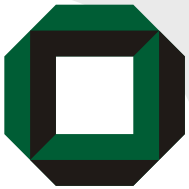
- Existieren eine Rangordnung und ein enger semantischer Zusammenhang?
  - Lässt sich die Beziehung durch „besteht aus“ oder „ist Teil von“ beschreiben? (Flotte besteht aus Schiffen, graph. Anzeige besteht aus Komponenten, Mengen oder Gruppen bestehen aus Einzelteilen)
  - Kann problemlos angegeben werden, ob eine Klasse Aggregat oder Teil in der Beziehung ist?
  - Wenn der Zugriff auf die Teilobjekte ausschließlich über das Aggregat-Objekt erfolgt, dann liegt eindeutig eine Aggregation vor.



3. Schritt: »So finden Sie Attribute«

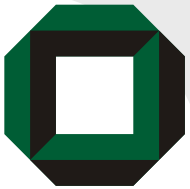
- Attribut-Kandidaten lassen sich ebenfalls mittels
  - Dokumentanalyse,
  - Analyse existierender Formulare oder
  - Suche nach realen Eigenschaften („technische Daten“)

  identifizieren.



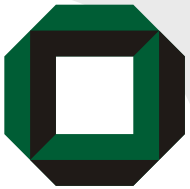
»So finden Sie Attribute«

- Ist ein Attribut problemrelevant im Sinne der Systemanalyse?
  - Kann jedes Attribut im Laufe seines „Lebens“ einen Wert annehmen?
  - Ist der Wert des Attributes jemals an der Benutzerschnittstelle zu sehen? (Wenn nicht, dann wahrscheinlich Implementierungsdetail—weglassen!)
  - Ist jedes Attribut relevant für die zu modellierende Anwendung?



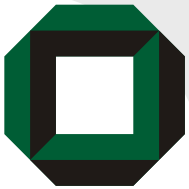
»So finden Sie Attribute«

- Gehört das Attribut zu einer Klasse oder Assoziation?
  - Muss das Attribut auch dann noch zu jedem Objekt der Klasse gehören, wenn die betreffende Klasse isoliert von allen anderen Klassen betrachtet wird, mit denen sie in Beziehung steht?
    - Wenn ja, dann gehört das Attribut zu einer Klasse
    - Wenn nein, dann ist zu prüfen, ob es sich einer Assoziation zuordnen läßt
    - Ist keine Zuordnung möglich, so spricht viel für eine vergessene Klasse oder Beziehung.



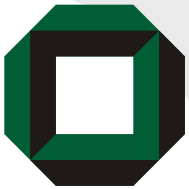
»So finden Sie Attribute«

- Ist der Attributname geeignet?
- Der Attributname soll...
  - Ein Substantiv oder eine Adjektiv-Substantiv-Kombination sein
  - eindeutig und verständlich im Kontext der Klasse sein
  - den Namen der Klasse nicht wiederholen (Ausnahmen sind feststehende Begriffe)
- Typ des Attributs festlegen
  - Beispiel: Telefonnummer
    - Ganzzahl? +49 (721) 608-3934
    - Spezielle Zeichenkette aus +-()0..9? 0700-RUFMICHAN
    - Allgemeine Zeichenkette? Besser!



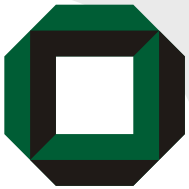
»So finden Sie Attribute«

- Liegen Klassenattribute vor?
  - Ein Klassenattribut liegt vor, wenn gilt:
    - Alle Objekte der Klasse besitzen dieselbe Eigenschaft
      Beispiel:
      Die Anzahl der Ecken aller Rechtecke ist 4.
    - Für die Erzeugung aller Objekte wird eine bestimmte Information benötigt
      Beispiel:
      Die Nummer eines Geschäftskontos beginnt mit einer »3«.
    - Informationen über die Gesamtheit der Exemplare einer Klasse
      Beispiel: *Anzahl der Konten*.



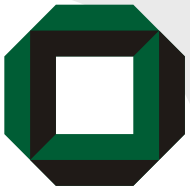
»So finden Sie Attribute«

- Wann wird ein Attribut nicht eingetragen?
  - Ein Attribut dient ausschließlich zum Identifizieren der Objekte
  - Ein Attribut dient lediglich dazu, eine andere Klasse zu referenzieren (benutze statt dessen eine Assoziation)
  - Ein Attribut beschreibt Entwurfs- oder Implementierungsdetails
  - Ein Attribut kann aus anderen Attributen abgeleitet werden (Ausnahme: Redundanz verbessert Lesbarkeit; Achtung: dann aber Konsistenz erhalten!)
    - Beispiel: Alter oder Geburtsdatum oder beides?



»So finden Sie Attribute«

- Allgemeiner Hinweis
  - Bestehen Zweifel darüber, ob Attribute einer bestehenden Klasse zugeordnet werden, oder ob eine neue Klasse gebildet wird, dann eine neue Klasse bilden.
  - Statt mehrerer atomarer Attribute, die untereinander einen deutlich größeren semantischen Zusammenhang besitzen als zu dem Rest der Attribute, lieber eine Assoziation zu einer neuen Klasse, die diese Attribute kapselt
    - Beispiel: „Person“ und ihre „Adresse“

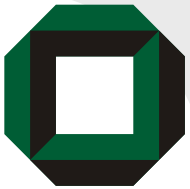


4. Schritt: »Erstellen von Vererbungsstrukturen«

- Da das Substitutionsprinzip gelten muss:

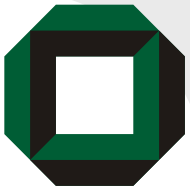
> Mache eine Klasse B erst zur Unterklasse einer Klasse A, wenn gezeigt werden kann, dass jedes Exemplar von B auch als ein Exemplar von A gesehen werden kann.

- Hinweis: im objektorientierten Analysemodell gibt es in der Regel viele Assoziationen, aber erstaunlich wenig Vererbungsbeziehungen!



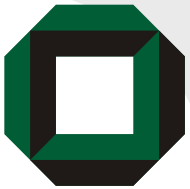
»Erstellen von Vererbungsstrukturen«

- Grundlage: alle bisher identifizierten Klassen
- Bilden einer neuen Oberklasse aus Klassen, die gemeinsame Attribute und Operationen besitzen
  - Ist die neue Oberklasse eine abstrakte Klasse?
- Auswählen einer existierenden Oberklasse
  - Ist ein Teil der Attribute oder Operationen einer Klasse bereits in einer anderen Klasse definiert?
- Zerlegung: Können aus Klassen, die sehr viele Attribute/Operationen besitzen, spezialisierte Klassen gebildet werden?
- Immer überprüfen: Ist die „Ist-ein“ Beziehung erfüllt?



»Erstellen von Vererbungsstrukturen«

- Die Klassen Sub1 und Sub2 sind Unterklassen der Klasse Super, wenn gilt:
  - Es ist notwendig, die Unterklassen Sub1 und Sub2 in dem zu modellierenden System zu unterscheiden
  - Die Unterklassen Sub1 und Sub2 besitzen zwar die Attribute/Operationen von Super gemeinsam, besitzen aber zusätzlich eigene (unterschiedliche!) Attribute/Operationen
  - Zwischen Sub1 bzw. Sub2 und der Oberklasse Super existiert eine Ist-ein-Beziehung (Substitutionsprinzip).



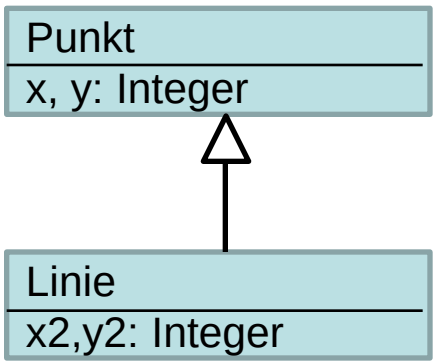
»Erstellen von Vererbungsstrukturen«

- Wann liegt keine Vererbung vor?
  - Die Klassen Sub1 und Sub2 enthalten keine (zusätzlichen) Attribute oder Operationen, keine Operation ist überschrieben.
  - Die Klassennamen von Sub1 und Sub2 bezeichnen verschiedene Typen der Klasse Super.

Achtung: nur Indizien!

»Erstellen von Vererbungsstrukturen«

- Wann liegt keine Vererbung vor?
  - Eine neue Klasse wird unter eine oder mehrere existierende Klassen gehängt, um die Vererbung auszunutzen, obwohl sie keine Spezialisierung darstellt
    - Hier kann nur durch eine Umstrukturierung der Klassen eine echte Generalisierung/Spezialisierung erreicht werden.

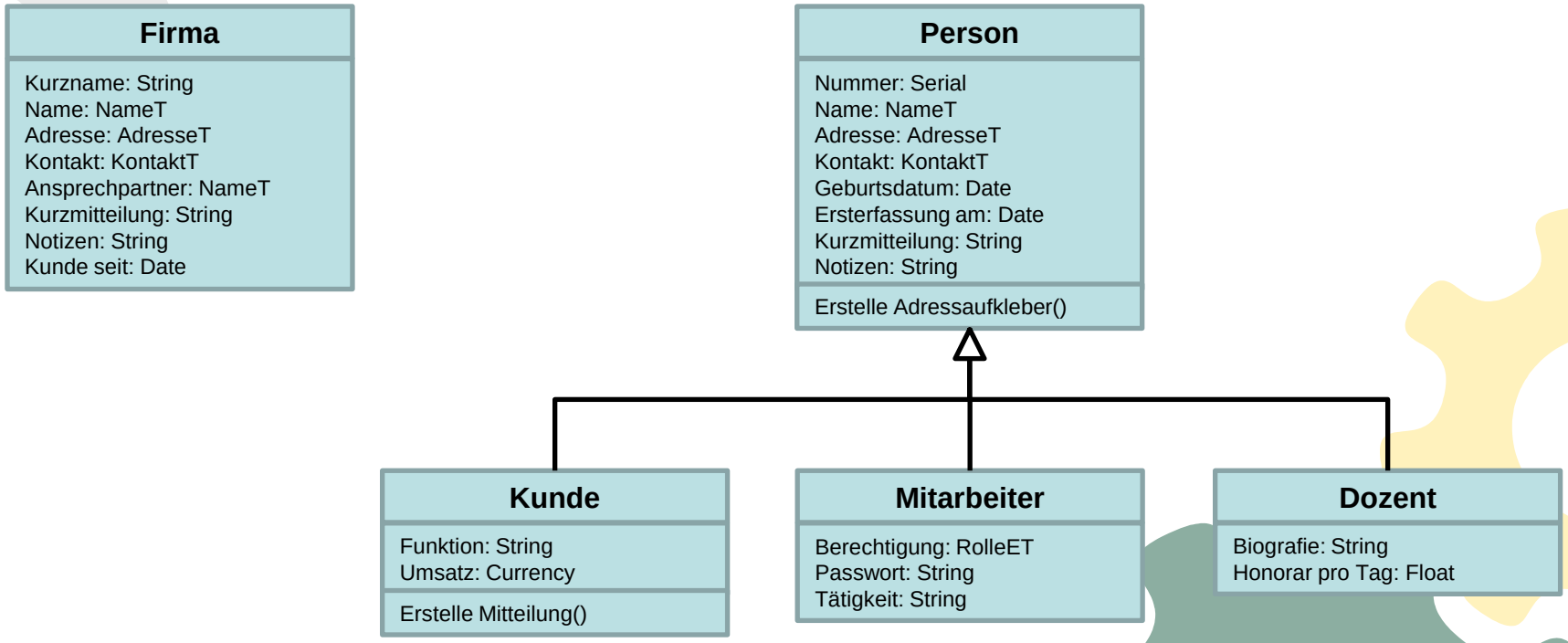


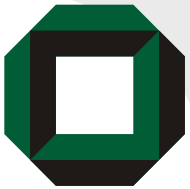
echter Mangel!

Warum sollte hier keine Vererbung benutzt werden?

Beispiel: Seminarorganisation

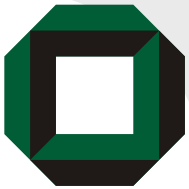
- Firma und Person haben zwar viele Gemeinsamkeiten, aber auch wichtige Unterschiede (z.B. hat eine Firma kein Geburtsdatum), weshalb Firma keine Unterklasse von Person ist.





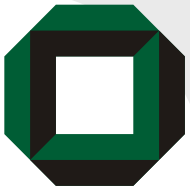
5. Schritt: Dynamisches Modell erstellen

- Quelle: Szenarien, Anwendungsfälle
- Ergebnis: Sequenz- und Aktivitätsdiagramm
- Zweck:
  - Operationen der Klassen identifizieren
  - Fluss der Botschaften durch das System definieren
  - Vollständigkeit und Korrektheit des statischen Systems prüfen
  - Grundlage für Systemtests schaffen
- Aus jedem Geschäftsprozess mehrere Sequenzdiagramme ermitteln
  - Gibt es Varianten im Geschäftsprozess, dann verschiedene Diagramme: für jede Variante eines
  - Positive und negative Fälle unterscheiden



Analyse und Prüfung

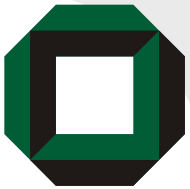
- Anwendungsfälle in Operationen zerlegen
- Empfänger-Objekte erreichbar?
  - Assoziationen vorhanden (dauerhafte Verbindung)
  - «uses»-Beziehung vorhanden (temporäre Verbindung, das gewünschte Exemplar kann dynamisch ermittelt werden)
- Konsistenz mit Klassendiagramm?
  - Existieren alle benötigten Klassen?
  - Existieren alle benötigten Operationen?



6. Schritt: Objektlebenszyklus bestimmen

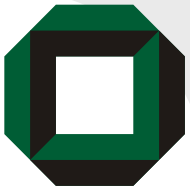
- Ergebnis: Zustandsdiagramm für Klassen, wo nötig.
- Besitzt das Objekt einen „nicht-trivialen“ Lebenszyklus?
  - IBM* gibt für typische Informationssysteme an, dass nur 1-2% der Klassen einen nichttrivialen Lebenszyklus haben.
  - Bei eingebetteten Systemen ist das anders - hier enthalten Steuerungsobjekte komplexe Zustandsdiagramme.
- Ein trivialer Lebenszyklus liegt vor, wenn zwischen Initialisierung und Destruktion nur ein einziger Zustand existiert.
- In diesem Falle nicht in „Analyse-Paralyse“ fallen!

*Developing Object-Oriented Software – An Experience-Based Approach, Prentice Hall, 1997



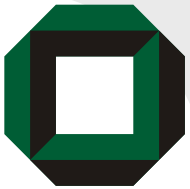
Wenn ein nicht-trivialer Objekt-Lebenszyklus vorliegt

- Erinnerung:
  Def. **Zustand**: Solange sich ein Objekt in einem Zustand befindet, reagiert es im gleichen Kontext immer gleich auf seine Umwelt. Ändert sich der Zustand, reagiert das Objekt in mindestens einem Kontext anders als zuvor.
- Objekt-Lebenszyklus spezifizieren, wenn gilt:
  - Das gleiche Ereignis kann – abhängig vom Objekt-Zustand – unterschiedliche Aktionen auslösen
  - Die Operationen können nur in bestimmten Situationen auf ein Objekt angewendet werden.



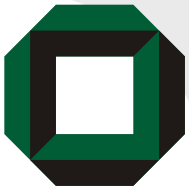
Analyse und Prüfung der Zustandsdiagramme

- Prüfe für alle Operationen der Klasse: Ist für jeden Zustand spezifiziert, was passiert, wenn die Operation aufgerufen wird?
- Prüfe für alle für ein Exemplar zu erwartenden Ereignisse: Ist ein entsprechender Übergang vorgesehen?
  - Externe Ereignisse: vom Benutzer, von anderen Komponenten
  - Zeit-Ereignisse
  - Interne Ereignisse: insbesondere Ausnahmen
- Ist der Automat vollständig und minimal?



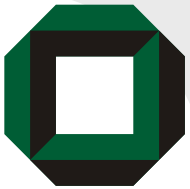
7. Schritt: Operationen festlegen

- Operationen aus Sequenzdiagrammen und Objektlebenszyklen übernehmen
- Operationen so hoch wie möglich in der Vererbungshierarchie eintragen
- Beschreibung erstellen



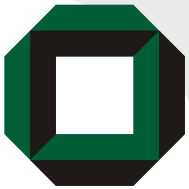
Analytische Schritte

- Besitzt die Operation geeigneten Namen ?
  - Beginnt mit einem Verb
  - Beschreibt, was gemacht wird
- Angemessener Umfang
  - vor allem nicht zu Umfangreich
- Funktionale Bindung
  - Jede Operation realisiert eine (1!) in sich abgeschlossene Funktion



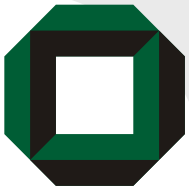
Subsysteme: Modellierung im Großen

- Zusammenfassen von einzelnen Klassen mit gemeinsamen Bezug zu einem Subsystem oder Paket.
- Innerhalb eines Subsystems: „starke Kohäsion“ (engl. strong cohesion)
- Zwischen den Subsystemen: „schwache Kopplung“ (weak coupling)



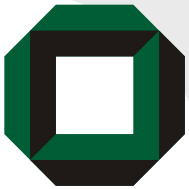
Starke Bindung

- Eine starke Kohäsion liegt vor, wenn das Subsystem
  - den Leser durch das Modell führt
  - einen Themenbereich enthält, der für sich allein betrachtet und verstanden werden kann
  - eine wohl definierte Schnittstelle zur Umgebung besitzt
  - nicht einfach aus einer Menge von Klassen besteht, sondern die Subsysteme sollen eine Betrachtung des Systems auf einer höheren Abstraktionsebene ermöglichen
  - einen aussagefähigen Namen besitzt, der der Gesamtheit der Klassennamen entspricht



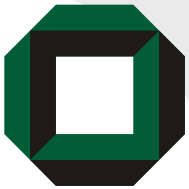
Schwache Kopplung

- Für die Schnittstelle zwischen zwei Subsystemen soll gelten:
  - Vererbungsstrukturen sind nur in vertikaler Richtung zu schneiden, d.h. zu jeder Unterklasse sollen möglichst alle Oberklassen in dem Subsystem enthalten sein, oder ein ganzer Ast des Vererbungsbaumes.
  - Aggregationen sollten nicht durchtrennt werden, d.h. Aggregat und Komponenten in einem Subsystem.
  - Die Schnittstelle zw. Subsystemen soll möglichst wenig Assoziationen enthalten.



Subsystem: Umfang

- Welchen Umfang soll ein Subsystem besitzen?
- Kriterien für eine sinnvolle Größe sind:
  - ca. 10 bis 15 Klassen
  - eine DIN A4-Seite.



Literatur

- Brügge, Dutoit: Object-Oriented Software Engineering, Kap. 5 (**lesen!**)
- Für weitere Hilfe empfohlen (optional!):
  - Helmut Balzert: Lehrbuch der Softwaretechnik, 2. Auflage, Lehreinheit 13
  - Heide Balzert, Lehrbuch der Objektmodellierung, Spektrum-Verlag,1999
  - Meyer: Object-oriented software construction, 2. edition, Prentice Hall, 1997