



Modellierung

2. Semester

Prof. Dr. Carsten Dorrhauer

Zeichnungen in den Folien stammen zum Teil aus den angegebenen Büchern.

Teile der Folien stammen aus den Begleitmaterialien zu den Büchern sowie aus dem Foliensatz "Modellierung" von Prof. Dr. Döringer, FH Ludwigshafen



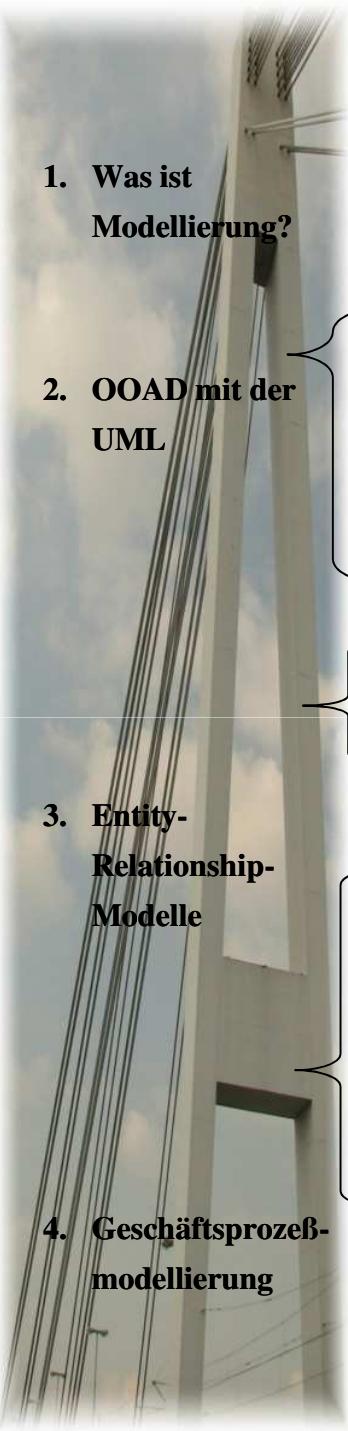
Voraussetzungen und Lernziele

Voraussetzungen

- Sie haben erste Erfahrungen mit der Programmierung in einer objektorientierten Sprache, vorzugsweise Java.

Lernziele

- Sie kennen den Zweck der Modellierung und ihre Rolle bei der Softwareentwicklung.
- Sie verstehen die wichtigsten Konzepte von objektorientierter Analyse und objektorientiertem Design.
- Sie kennen die UML und können ihre wichtigsten Diagrammtypen anwenden.



Literaturempfehlungen

1. Was ist
Modellierung?

2. OOAD mit der
UML

3. Entity-
Relationship-
Modelle

4. Geschäftsprozeß-
modellierung

McLaughlin, Brett/Pollice, Gary/West, David:
Head first object-oriented analysis and design, Beijing
2007

Kecher, Christoph:
UML 2.0 - das umfassende Handbuch, Bonn 2007

Kemper, Alfons/Eickler, André: Datenbanksysteme - eine
Einführung, 6. Aufl., München 2006

Allweyer, Thomas: Geschäftsprozessmanagement -
Strategie, Entwurf, Implementierung, Controlling,
Herdecke 2007

Staud, Josef: Geschäftsprozessanalyse, 3. Aufl., Berlin
2006



Gliederung

1. Was ist Modellierung?
2. Objektorientierte Analyse und Design (OOAD) mit der Unified Modeling Language (UML)
 1. Kurze Wiederholung: Grundbegriffe der Objektorientierung
 2. Wozu OOAD?
 3. Was ist die UML?
 4. OOA: Use-Case-Dokumente und Use-Case-Diagramme
 5. OOD: Klassendiagramme im Detail
 6. Sequenz- und Aktivitätendiagramme
 7. Weitere UML-Diagramme im Überblick
3. Modellierung relationaler Datenbanken mit Entity-Relationship-Modellen
4. Geschäftsprozeßmodellierung mit Ereignisgesteuerten Prozeßketten (EPK)



1. Was ist Modellierung?

Dokument öffnen mit
ufspubcpvusdrelsdrimcbgstba



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Modelle

- ... bilden die Wirklichkeit ab,
- ... dürfen aber nicht mit ihr verwechselt werden.
- ... stellen Informationen über sie zur Verfügung.
- ... reduzieren die Komplexität auf die in einem bestimmten Kontext relevanten Informationen.



- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Modelle dürfen nicht mit der Realität verwechselt werden



"The map is not the territory"

(Alfred Habdank Skarbek Korzybski)

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Modelle reduzieren Komplexität

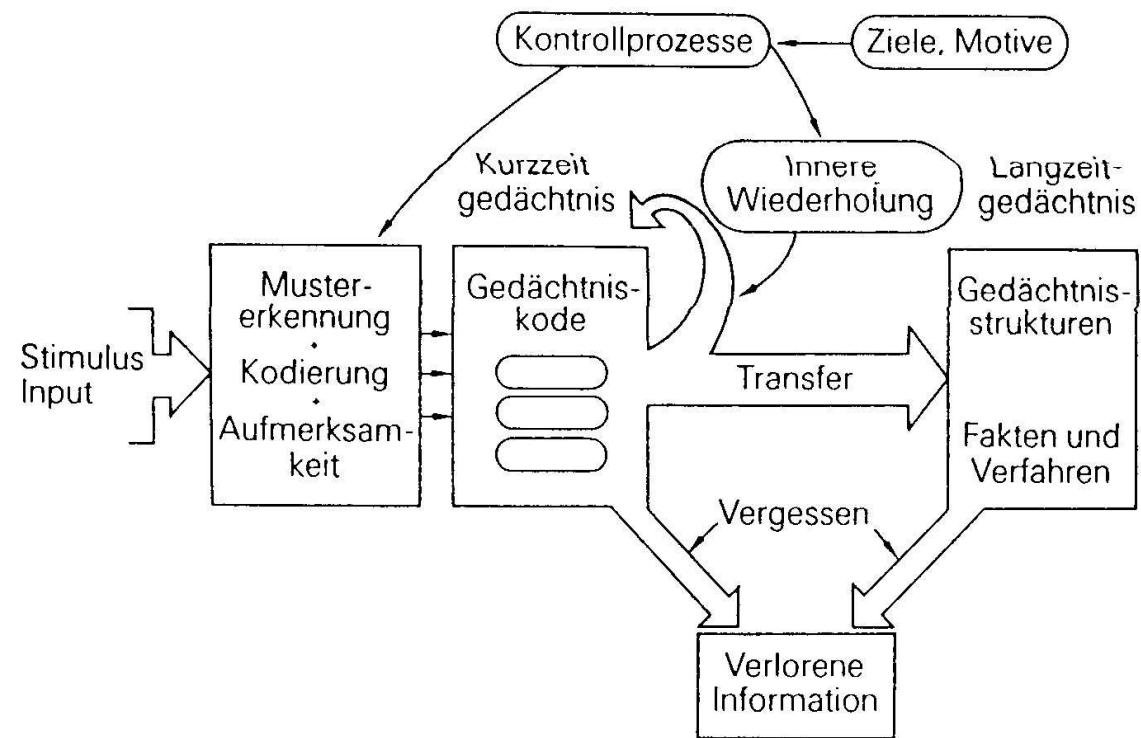


"A model which took account of all the variegation of reality would be of no more use than a map at the scale of one to one. "

(Joan Robinson)

1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeßmodellierung

Bsp.: Ein Modell aus der Pädagogik



Gudjons H., Pädagogisches Grundwissen, Bad Heilbrunn 1993, S. 205

1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeßmodellierung

Bsp.: Ein Modell aus der BWL

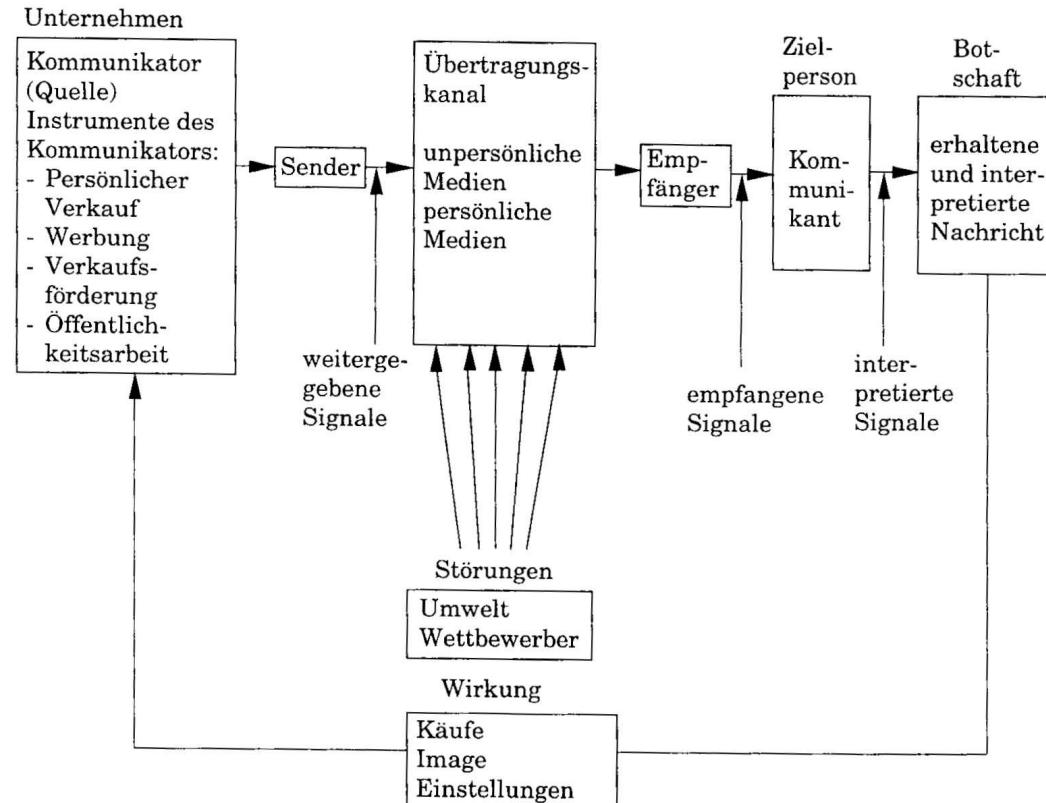


Abb.: Marketingkommunikationssystem

Weis, H. C., Marketing, 12. Aufl., Ludwigshafen 2001, S. 418

- 1. Was ist Modellierung?
- 2. OOAD mit der UML
- 3. Entity-Relationship-Modelle
- 4. Geschäftsprozeß-modellierung

Bsp.: Ein Modell aus der VWL

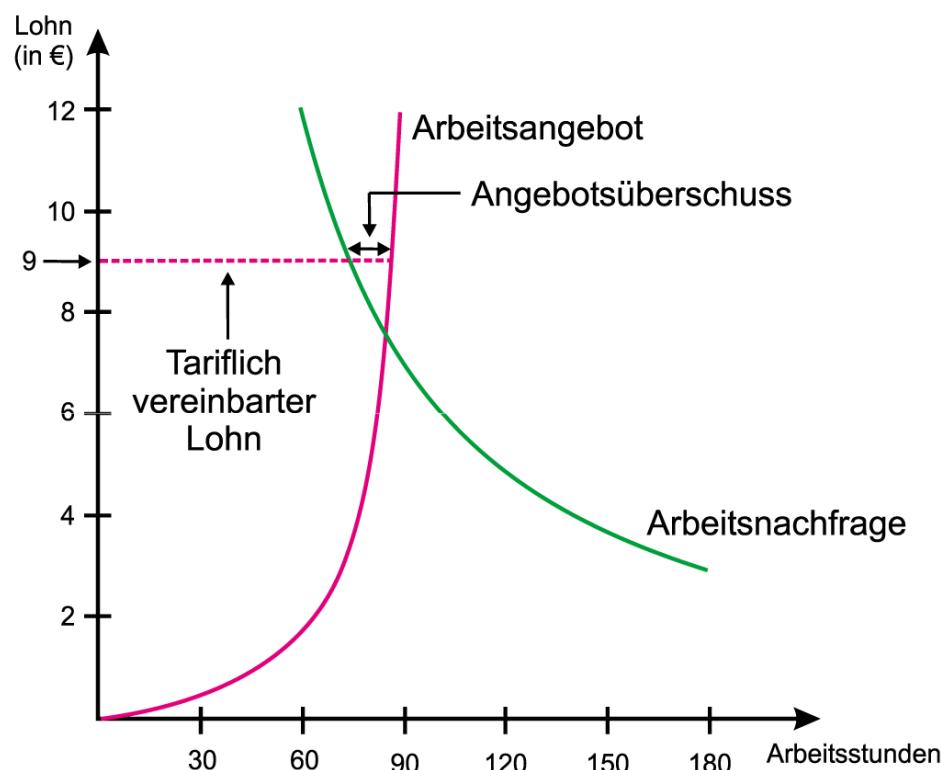


Schaubild 9.8: Zu hohe Löhne können zu Arbeitslosigkeit führen

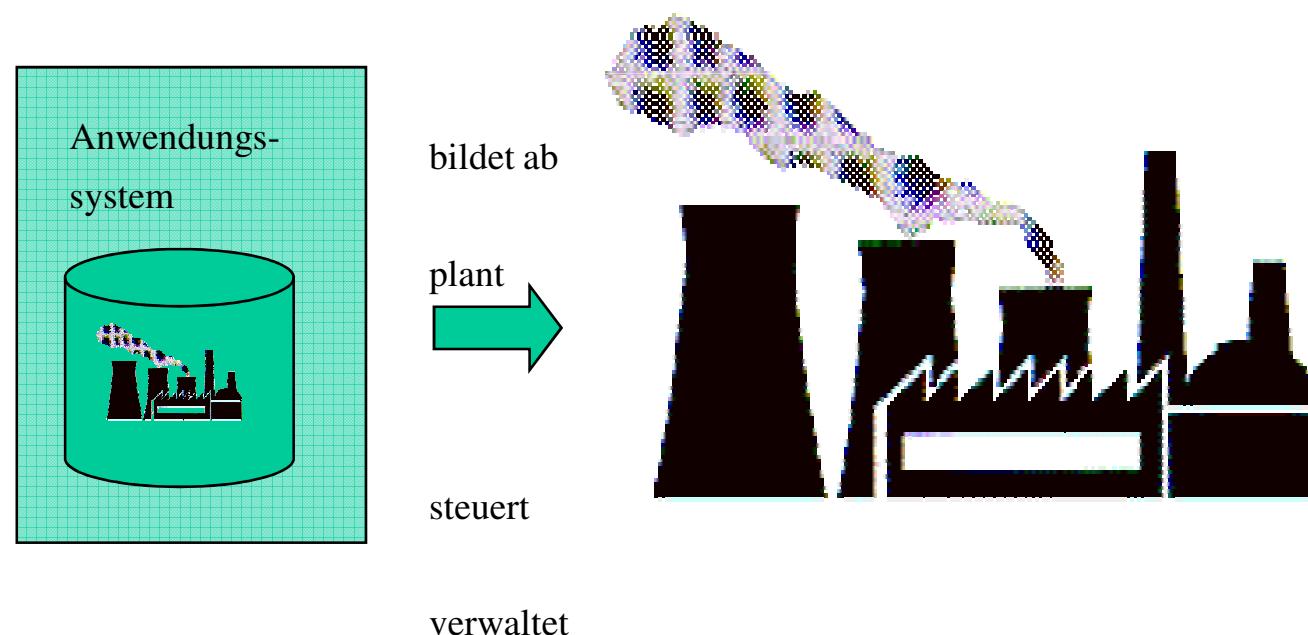
Bofinger, P., Grundzüge der Volkswirtschaftslehre, München 2003, S. 164



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Zweck von Modellen in der Wirtschaftsinformatik

- Ein IT-System, das einen Ausschnitt der realen Welt verwalten soll, muß dessen Struktur kennen.
- Dazu braucht Software ein Modell.





1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Zweck von Modellen in der Wirtschaftsinformatik

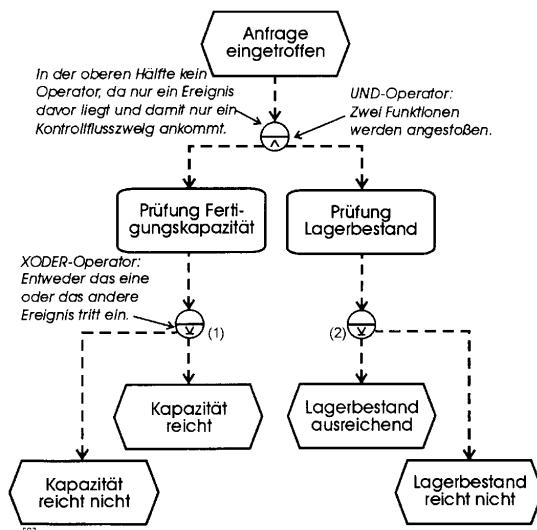
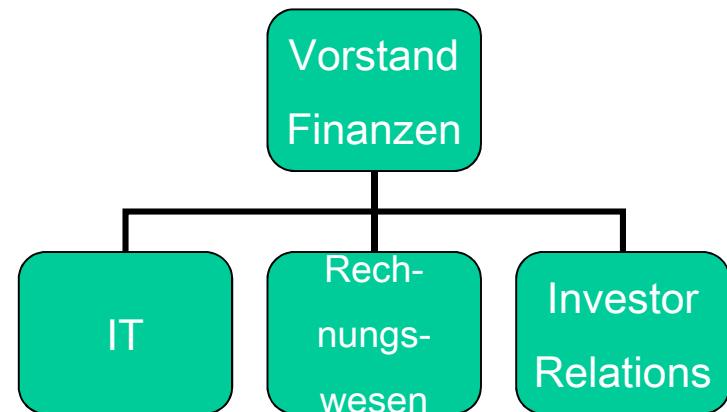
- Modelle vereinfachen das Verständnis von Zusammenhängen, indem sie Komplexität reduzieren.
- Fehler im Modell zu erkennen und zu beheben, ist viel billiger als im fertigen System.
- Um unterschiedliche Aspekte zu betrachten, kann man verschiedene Sichten auf die Realwelt in mehreren Modellen abbilden.



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeßmodellierung

Zweck von Modellen in der Wirtschaftsinformatik

- Modelle eignen sich als Grundlage zur Diskussion über den Realweltausschnitt, den sie darstellen.
 - Bsp.: Ein Organigramm modelliert die Aufbauorganisation eines Unternehmens.
 - Bsp.: Ein Prozeßmodell modelliert die Ablauforganisation eines Unternehmens.
(wird noch ausführlich behandelt)

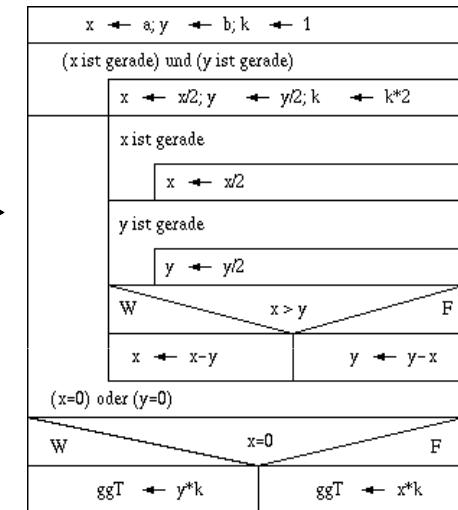
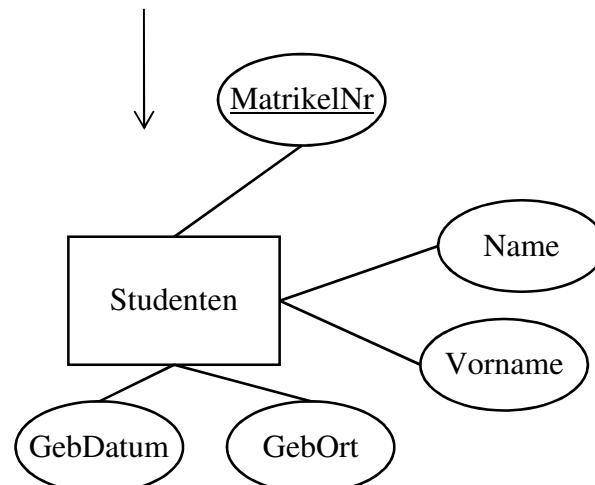




1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Zweck von Modellen in der Wirtschaftsinformatik

- Modelle eignen sich auch als Vorlage für zu erstellende Software. Sie beschreiben z.B.
- einen Algorithmus (Bsp.: Nassi-Shneiderman-Diagramm, auch Struktogramm genannt) →
- Datenstrukturen (Bsp.: Entity-Relationship-Modell)
(wird noch ausführlich behandelt)





1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Zweck von Modellen in der Wirtschaftsinformatik

- Nach Fertigstellung der Software eignen sich Modelle als Grundlage für ihren Test: Die Software wird mit dem Modell verglichen.



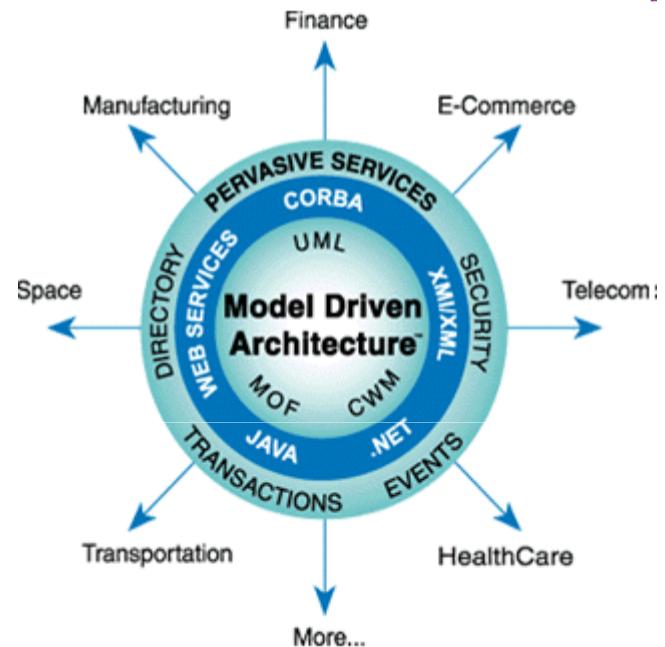
- Spezialisierte Testwerkzeuge können zumindest teilweise automatisiert auf Basis von Modellen eingerichtet werden.



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

MDA

- Model driven architecture (MDA) ist ein Begriff der Object Management Group.
- Man versucht, den Aufbau eines Softwaresystems zunächst unabhängig von einer Plattform oder Programmiersprache in Form von Modellen zu entwickeln. (Platform-independent model, PIM)
- Anschließend werden Teile des Quellcodes automatisiert erzeugt.



<http://www.omg.org/mda/>



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Qualität von Modellen

- Es kommt nicht auf die Realitätsnähe, sondern auf die Zweckmäßigkeit eines Modells an.
- Die Güte eines Modells kann man deshalb nur beurteilen, wenn man Zweck und Zielgruppe kennt.
- Für eine Modellierungsaufgabe gibt es nicht eine, sondern viele "richtige" Lösungen. Modellierung ist ein kreativer Akt.
- Natürlich kann es auch falsche Lösungen geben:
 - Sie verstößen gegen die Notationsstandards.
 - Sie modellieren Dinge, die der Realität widersprechen.
 - Sie sind für Zweck und Zielgruppe zu sehr oder zu wenig detailliert.





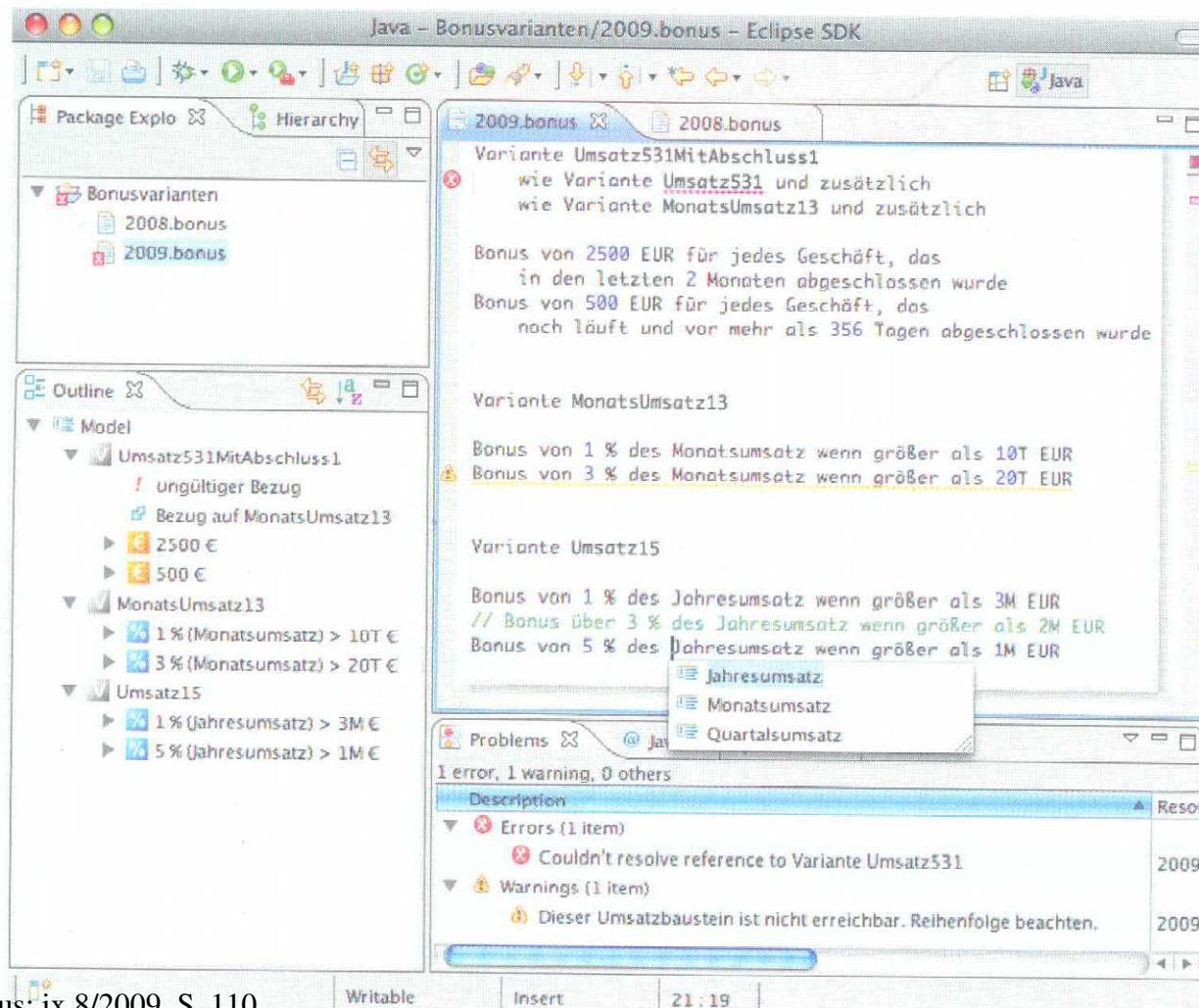
1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Domain Specific Language (DSL)

- Eine DSL ist eine Sprache, die speziell für einen Aufgabenbereich (eine Domain) konzipiert wurde.
- Aus der Problemdefinition in der DSL wird der Programmcode generiert.
- Vorteil: Die Sprache soll von den Fachleuten des Aufgabenbereiches ohne weitere Erläuterung verstanden werden.
- Nachteil: Für jede Domain (oder sogar für jedes Softwareprojekt?) muß zunächst eine eigene DSL konzipiert werden.
- Nachteil: Die Entwickler müssen sich in diese DSL zunächst einarbeiten.
- Es wird diskutiert, ob DSL standardisierte Modellierung ablösen. Vermutlich werden sie eher ergänzen.

1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Beispiel DSL





1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Übungsaufgabe

1. Wobei handelt es sich um ein Modell?
 - a) Eine Explosionszeichnung
 - b) Ein Kochrezept
 - c) Eine Partitur
2. a) Entwerfen Sie mit MS Powerpoint oder Openoffice Draw ein Organigramm für die Hopfengold Brauerei GmbH.
b) Nach welchen Kriterien ist die Aufbauorganisation gegliedert?
c) Welches weitere Kriterium wäre möglich?

Geschäftsführer Erwin Scheff steht vier Bereichsleitern für *Produktion*, *Vertrieb*, *Marketing* und *Finanzen* vor. Produktionschef Meister hat den einzigen Einkäufer unter sich, außerdem einen Mitarbeiter für die Qualitätssicherung und die Abteilung Herstellung, die für den eigentlichen Brauprozess sorgt. Der Vertriebschef steht den drei Abteilungen *Vertrieb Süddeutschland*, *Vertrieb Norddeutschland* und *Export* vor. Die Abteilung Marketing hat einen Abteilungsleiter und je eine Gruppe für *Pils*, *Weizen* und *Mischgetränke*. Die Abteilung Finanzen besteht aus je einem Mitarbeiter für Buchführung, Kostenrechnung und IT sowie einer Halbtagskraft mit Zuständigkeit für Personalverwaltung und Lohnbuchhaltung. Franz Fröhlich besetzt die Stabsstelle *Verkostung* und berichtet direkt an Erwin Scheff.



2. Objektorientierte Analyse und Design mit der Unified Modeling Language



2.1 Kurze Wiederholung: Grundbegriffe der Objektorientierung



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeßmodellierung

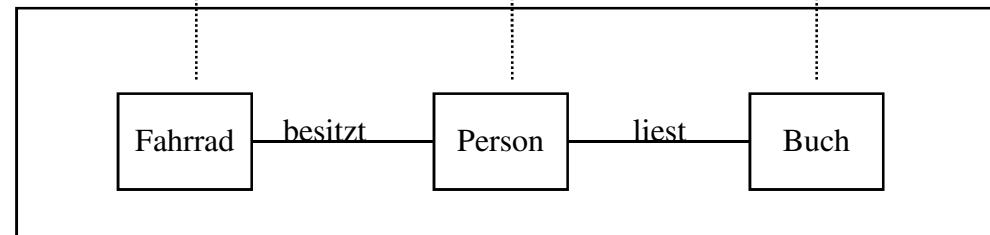
Kurze Wiederholung: Grundbegriffe der Objektorientierung - Objekte

Objektorientierung: Gegenstände der Realität werden als Objekte abgebildet

Realität



Modell





1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

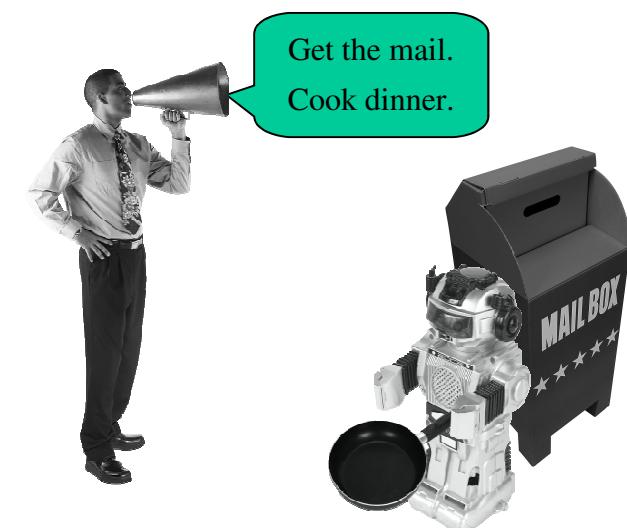
Kurze Wiederholung: Grundbegriffe der Objektorientierung - Objekte

- Ein Objekt hat einen Zustand, der durch die Werte seiner Attribute beschrieben wird.



Dave
Age: 32
Height: 6' 2"

- Ein Objekt hat ein Verhalten: Seine Methoden beschreiben, wie es in einem bestimmten Zustand auf bestimmte Nachrichten reagiert.





1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Kurze Wiederholung: Grundbegriffe der Objektorientierung - Objekte

- Ein Objekt hat eine Identität: Wenn zwei Objekte in allen Attributen identische Werte haben, bleiben es doch zwei verschiedene, jeweils einzigartige Objekte.

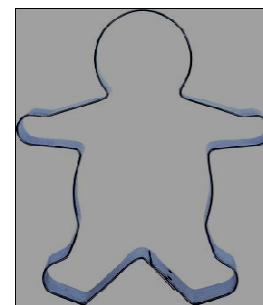




1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Kurze Wiederholung: Grundbegriffe der Objektorientierung - Klassen

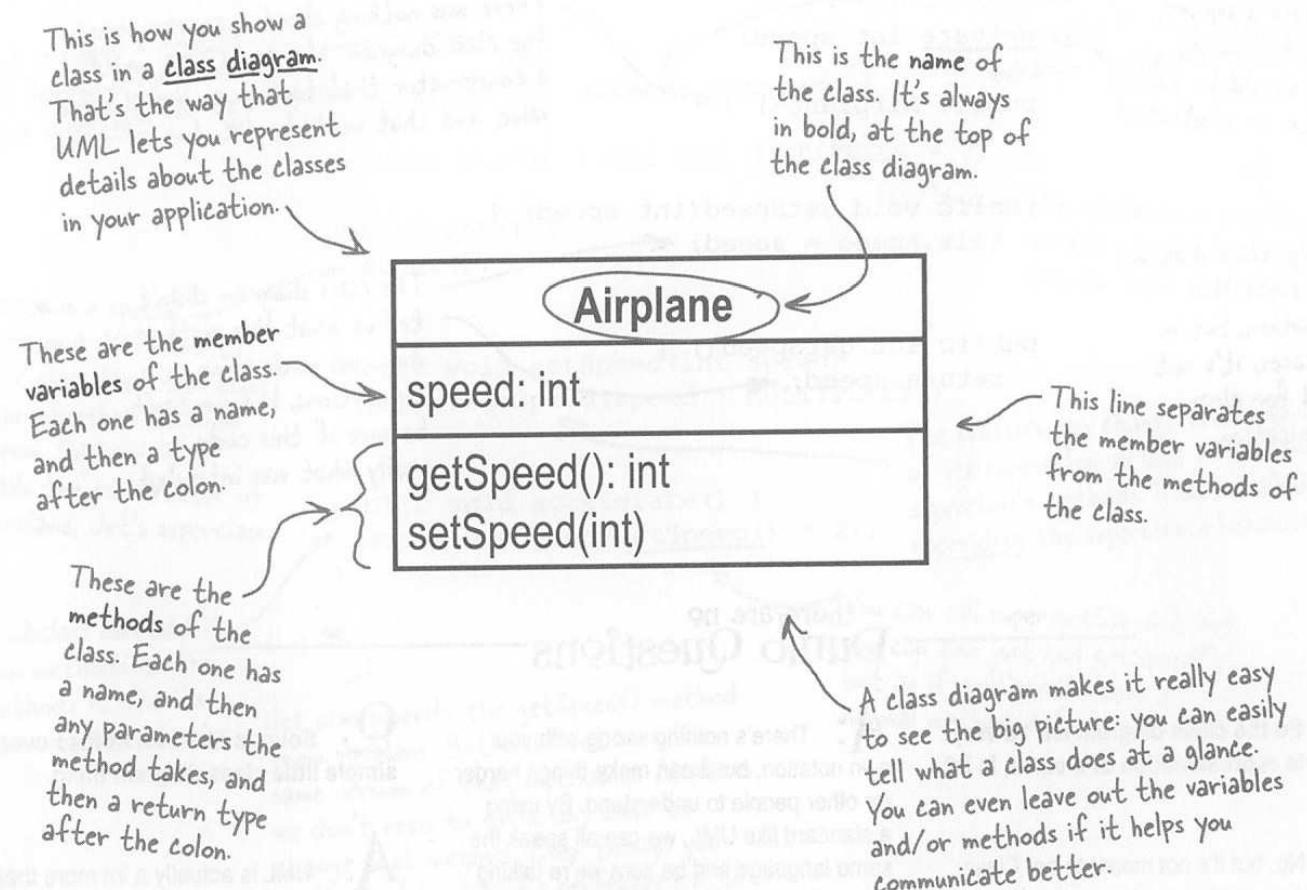
- Klassen definieren Attribute und Verhalten ihrer Objekte.
 - Sie können als Schablone zur Erzeugung von Objekten interpretiert werden.
 - Sie können als Menge von Objekten interpretiert werden.
- Klassen können eigene Attribute und ein eigenes Verhalten haben
- In einem Banksystem gibt es z.B. eine Klasse "Girokonten". Ein Objekt dieser Klasse ist das Konto mit der Nummer 3146883.



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Kurze Wiederholung: Grundbegriffe der Objektorientierung - Klassen

Ein erstes Diagramm:





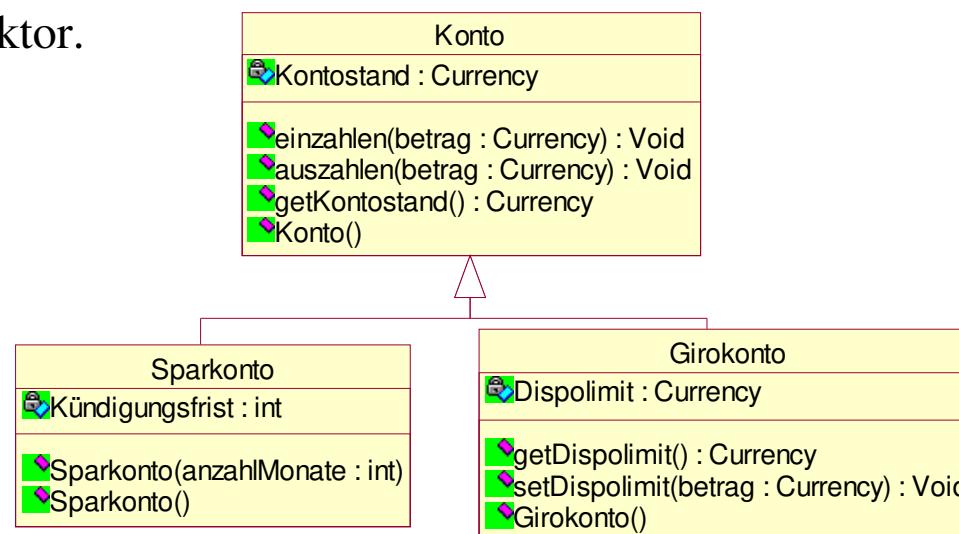
1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Kurze Wiederholung: Grundbegriffe der Objektorientierung - Vererbung

- Klassen können voneinander Verhalten und Eigenschaften erben.

Die Klasse "Girokonto" erbt von der Klasse "Konto" das Datenfeld "Kontostand", erweitert aber den Erblasser "Konto" um das Feld "Dispolimit".

Die Klasse "Sparkonto" erbt von der Klasse "Konto" ebenfalls das Datenfeld "Kontostand", erweitert aber den Erblasser "Konto" um das Feld "Kündigungsfrist" und einen eigenen Konstruktor.

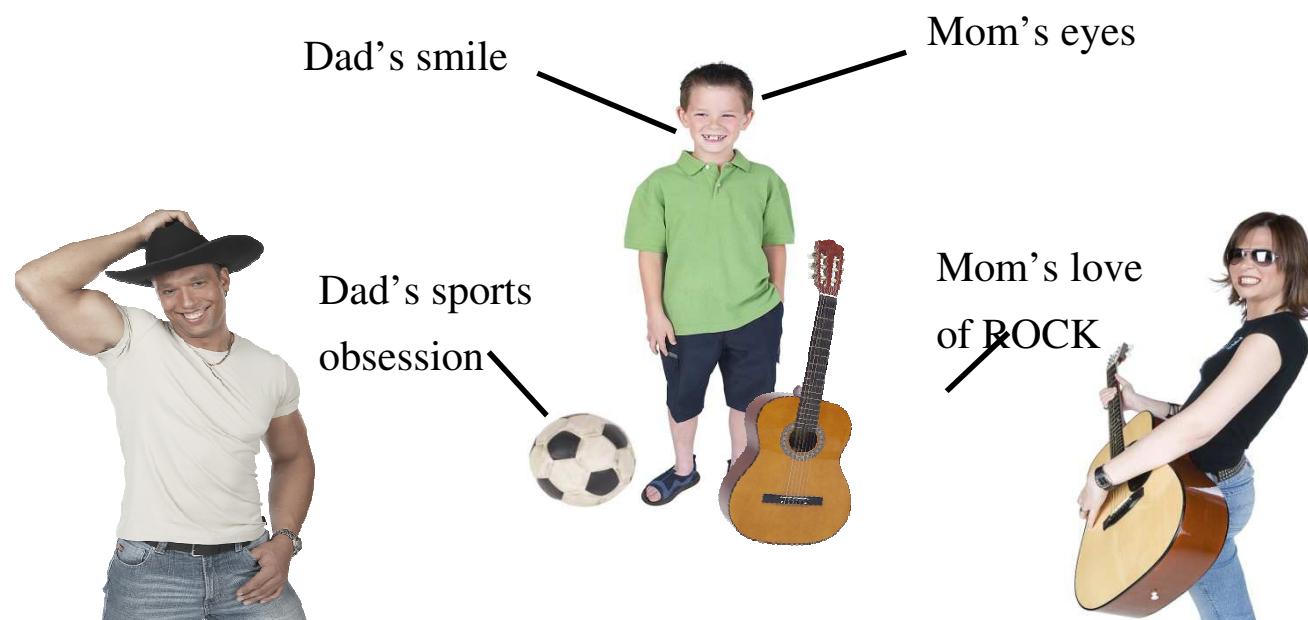




1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Kurze Wiederholung: Grundbegriffe der Objektorientierung - Vererbung

- In Java kann eine Klasse nur von genau einer anderen Klasse erben, in anderen Sprachen von mehreren anderen Klassen.

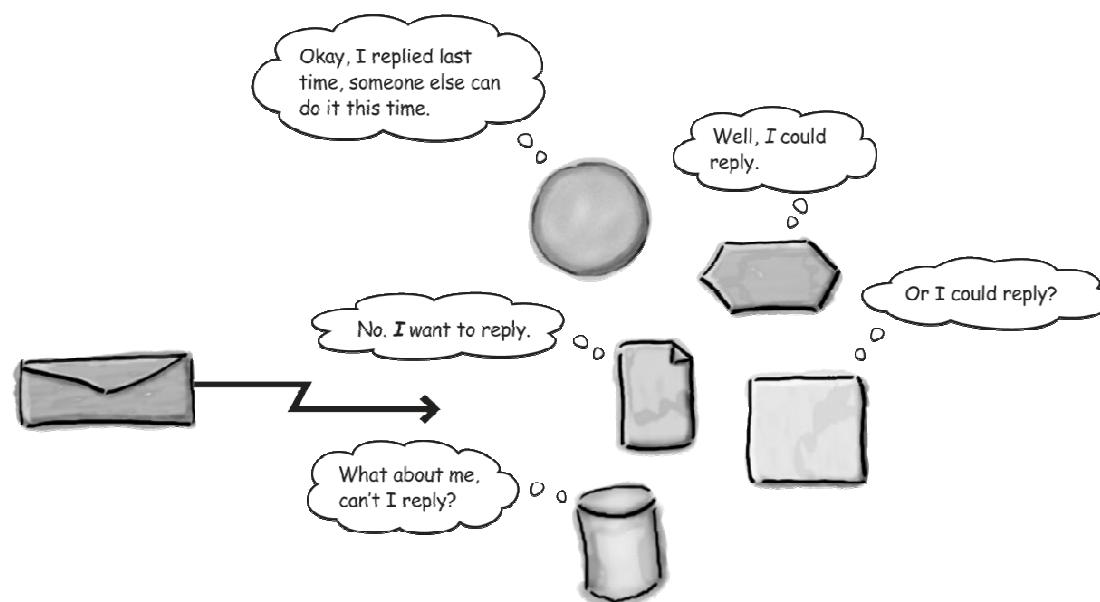




1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Kurze Wiederholung: Grundbegriffe der Objektorientierung - Polymorphismus

- Verschiedene Arten von Objekten können auf die gleiche Nachricht reagieren.
 - Welche Methode tatsächlich ausgeführt wird, entscheidet sich erst zur Laufzeit (Late binding).





1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Kurze Wiederholung: Grundbegriffe der Objektorientierung - Kapselung

- Information wird vor einem Teil der Applikation verborgen gehalten.
- Häufiges (aber nicht einziges) Beispiel: Die Attribute eines Objektes dürfen nur von seinen eigenen Methoden geändert werden, nicht von beliebigen anderen. Von außen sind sie nicht sichtbar.

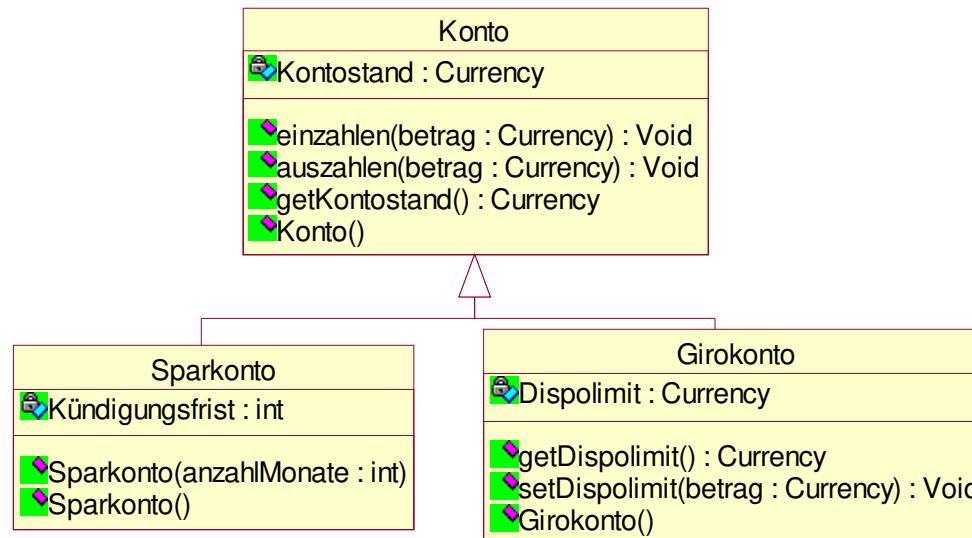




1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Übungsaufgabe

1. Schreiben Sie ein Codefragment für die (unvollständige) Klasse "Sparkonto".
2. Erstellen Sie das Klassendiagramm mit Rational Rose.
3. Erzeugen Sie ein Codefragment mit Rational Rose.
4. Vergleichen Sie Ihr Codefragment mit dem erzeugten Codefragment.



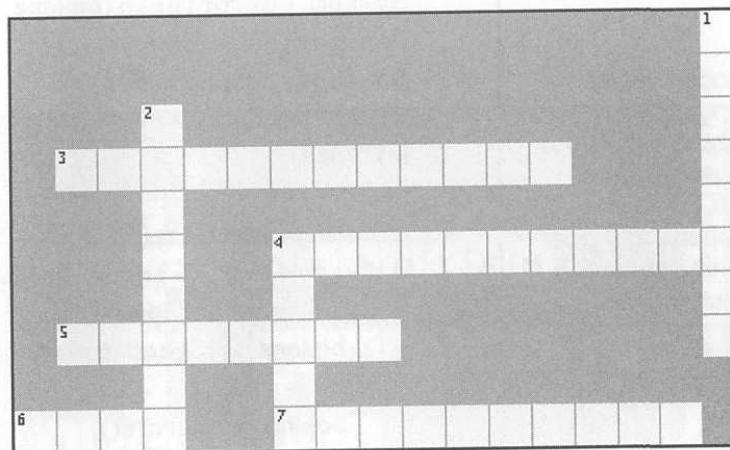
1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Übungsaufgabe



OO&D Cross

Take a moment to review the concepts in this appendix, and then you're ready to step firmly into the world of analysis, design, OO programming, and great software.



Across

3. This is when a subclass can substitute for its superclass.
4. Class diagrams don't show details about this part of our class.
5. A class that inherits from another class is this.
6. Encapsulation lets you do this to information.
7. A class that is inherited from is called this.

Down

1. Another term for encapsulation is separation of this.
2. Polymorphism helps make your code
4. Use this kind of diagram to avoid wading through lots of code.



2.2 Wozu OOAD?

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Wozu OOAD?

- Was macht den Unterschied zwischen "normaler" Software und *wirklich guter* Software aus?



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Beispiel: Ricks Guitars

- Anforderung: Lager verwalten und den Kunden das passende Instrument empfehlen.
- Die Vorgängerapplikation:



Guitar
serialNumber : String
price : double
builder : String
model : String
type : String
backWood : String
topWood : String
+getSerialNumber() : String
+getPrice() : double
+setPrice(newPrice : double)
+getBuilder() : String
+getModel() : String
+getType() : String
+getBackWood() : String
+getTopWood() : String

Inventory
guitars : List
+addGuitar(serialNumber : String, price : double, builder : String, model : String, type : String, backWood : String, topWood : String) : void
+getGuitar(serialNumber : String) : Guitar
+searchGuitar(searchedFor : Guitar) : Guitar



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Beispiel: Ricks Guitars

- Das größte Problem mit der Vorgängerapplikation:
fender ≠ Fender



Rick's Guitars
Back Room

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Beispiel: Ricks Guitars

- Ein Blick auf den Quellcode: Klasse *Guitar*

```
public class Guitar {  
  
    private String serialNumber, builder, model, type, backWood, topWood;  
    private double price;  
  
    public Guitar(String serialNumber, double price,  
                  String builder, String model, String type,  
                  String backWood, String topWood) {  
        this.serialNumber = serialNumber;  
        this.price = price;  
        this.builder = builder;  
        this.model = model;  
        this.type = type;  
        this.backWood = backWood;  
        this.topWood = topWood;  
    }  
    public String getSerialNumber() {return serialNumber;}  
    public double getPrice() {return price;}  
    public void setPrice(float newPrice) {  
        this.price = newPrice;  
    }  
    public String getBuilder() {return builder;}  
    public String getModel() {return model;}  
    public String getType() {return type;}  
    public String getBackWood() {return backWood;}  
    public String getTopWood() {return topWood;}  
}
```

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Beispiel: Ricks Guitars

- Ein Blick auf den Quellcode: Klasse *Inventory* (1)

```
public class Inventory {  
    private List guitars;  
    public Inventory() { guitars = new LinkedList(); }  
    public void addGuitar(String serialNumber, double price,  
                          String builder, String model,  
                          String type, String backWood, String topWood) {  
        Guitar guitar = new Guitar(serialNumber, price, builder,  
                                   model, type, backWood, topWood);  
        guitars.add(guitar);  
    }  
    public Guitar getGuitar(String serialNumber) {  
        for (Iterator i = guitars.iterator(); i.hasNext(); ) {  
            Guitar guitar = (Guitar)i.next();  
            if (guitar.getSerialNumber().equals(serialNumber)) {  
                return guitar;  
            }  
        }  
        return null;  
    }  
}
```

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Beispiel: Ricks Guitars

- Ein Blick auf den Quellcode: Klasse *Inventory* (2)

```
public Guitar search(Guitar searchGuitar) {  
    for (Iterator i = guitars.iterator(); i.hasNext(); ) {  
        Guitar guitar = (Guitar)i.next();  
        String builder = searchGuitar.getBuilder().toLowerCase();  
        if ((builder != null) && (!builder.equals("")) &&  
            (!builder.equals(guitar.getBuilder().toLowerCase())))  
            continue;  
        String model = searchGuitar.getModel().toLowerCase();  
        if ((model != null) && (!model.equals("")) &&  
            (!model.equals(guitar.getModel().toLowerCase())))  
            continue;  
        String type = searchGuitar.getType().toLowerCase();  
        if ((type != null) && (!searchGuitar.equals("")) &&  
            (!type.equals(guitar.getType().toLowerCase())))  
            continue;  
        String backWood = searchGuitar.getBackWood().toLowerCase();  
        if ((backWood != null) && (!backWood.equals("")) &&  
            (!backWood.equals(guitar.getBackWood().toLowerCase())))  
            continue;  
        String topWood = searchGuitar.getTopWood().toLowerCase();  
        if ((topWood != null) && (!topWood.equals("")) &&  
            (!topWood.equals(guitar.getTopWood().toLowerCase())))  
            continue;  
        return guitar;  
    }  
    return null;  
}
```

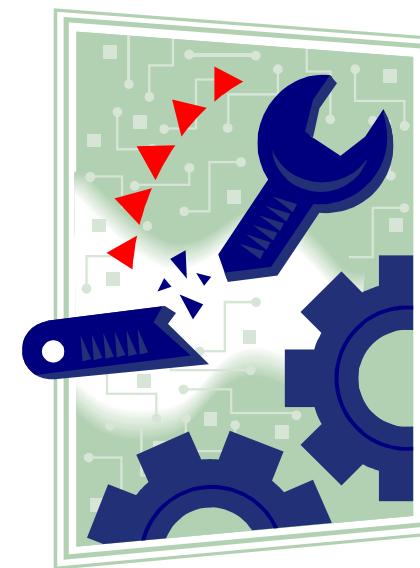
Ihre erste Verbesserung

Vorsicht: Fehler im Lehrbuch!

1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Beispiel: Ricks Guitars

- Welche Probleme gibt es mit diesen beiden Klassen?





1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Drei (mögliche!) Schritte zu guter Software

well-designed apps rock

Great software in 3 easy steps

It may not seem easy now, but we'll show you how OOA&D and some basic principles can change your software forever.

1. Make sure your software does what the customer wants it to do.

This step focuses on the customer. Make sure the app does what it supposed to do FIRST.

2. Apply basic OO principles to add flexibility.

Once your software works, you can start to remove duplicate code and use good OO practices.

3. Strive for a maintainable, reusable design.

Got a good object-oriented app that does what it should? It's time to apply patterns and principles to make sure your software is ready to use for years to come.

you are here ▶ 11



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Drei (mögliche!) Schritte zu guter Software

- Das ist natürlich nur eine von mehreren möglichen Vorgehensweisen!
- Es wäre auch ein Mißverständnis, loszuprogrammieren und erst hinterher über das Design der Software nachzudenken. Im Beispiel gibt es bereits eine (schlechte) Vorgängerapplikation.
- Idealerweise beziehen sich alle drei Schritte auf das Design, nicht auf die Implementierung.
- Manchmal ist es aber hilfreich, sich beim Design schon Codefragmente in der Zielprogrammiersprache zu überlegen, um Alternativen klarer zu sehen.

1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Schritt 1: Den Kunden befragen

- Welche Fragen stellen Sie Rick, bevor Sie seine Software verbessern?



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Schritt 1: Den Kunden befragen

- Mögliche Antworten Ihres Kunden:

Customers don't always know all of the characteristics of the guitar they want.

There's often more than one guitar that matches the customer's needs.

Customers often look for a guitar in a specific price range

I need reports and other capabilities in my inventory, but my #1 problem is finding the right guitar for the customer.



Was gibt es also zu tun?

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Schritt 1: Den Kunden befragen

Erste Verbesserung: Tippfehler ausschließen

```
public class Guitar {  
  
    private String serialNumber,  
        model;  
    private double price;  
    private Builder builder;  
    private Type type;  
    private Wood backWood,  
topWood;  
  
    ...  
}
```

```
public enum Type {  
  
    ACOUSTIC, ELECTRIC;  
  
    public String toString() {  
        switch(this) {  
            case ACOUSTIC: return "acoustic";  
            case ELECTRIC: return "electric";  
            default:         return "unspecified";  
        }  
    }  
}
```

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Schritt 1: Den Kunden befragen

Zweite Verbesserung: Bei Bedarf mehrere Gitarren finden

```
public List search(Guitar searchGuitar) {  
    List matchingGuitars = new LinkedList();  
    for (Iterator i = guitars.iterator(); i.hasNext(); ) {  
        Guitar guitar = (Guitar)i.next();  
        // Ignore serial number since that's unique  
        // Ignore price since that's unique  
        if (searchGuitar.getBuilder() != guitar.getBuilder())  
            continue;  
        String model = searchGuitar.getModel().toLowerCase();  
        if ((model != null) && (!model.equals("")) &&  
            (!model.equals(guitar.getModel().toLowerCase())))  
            continue;  
        if (searchGuitar.getType() != guitar.getType())  
            continue;  
        if (searchGuitar.getBackWood() != guitar.getBackWood())  
            continue;  
        if (searchGuitar.getTopWood() != guitar.getTopWood())  
            continue;  
        matchingGuitars.add(guitar);  
    }  
    return matchingGuitars;  
}
```



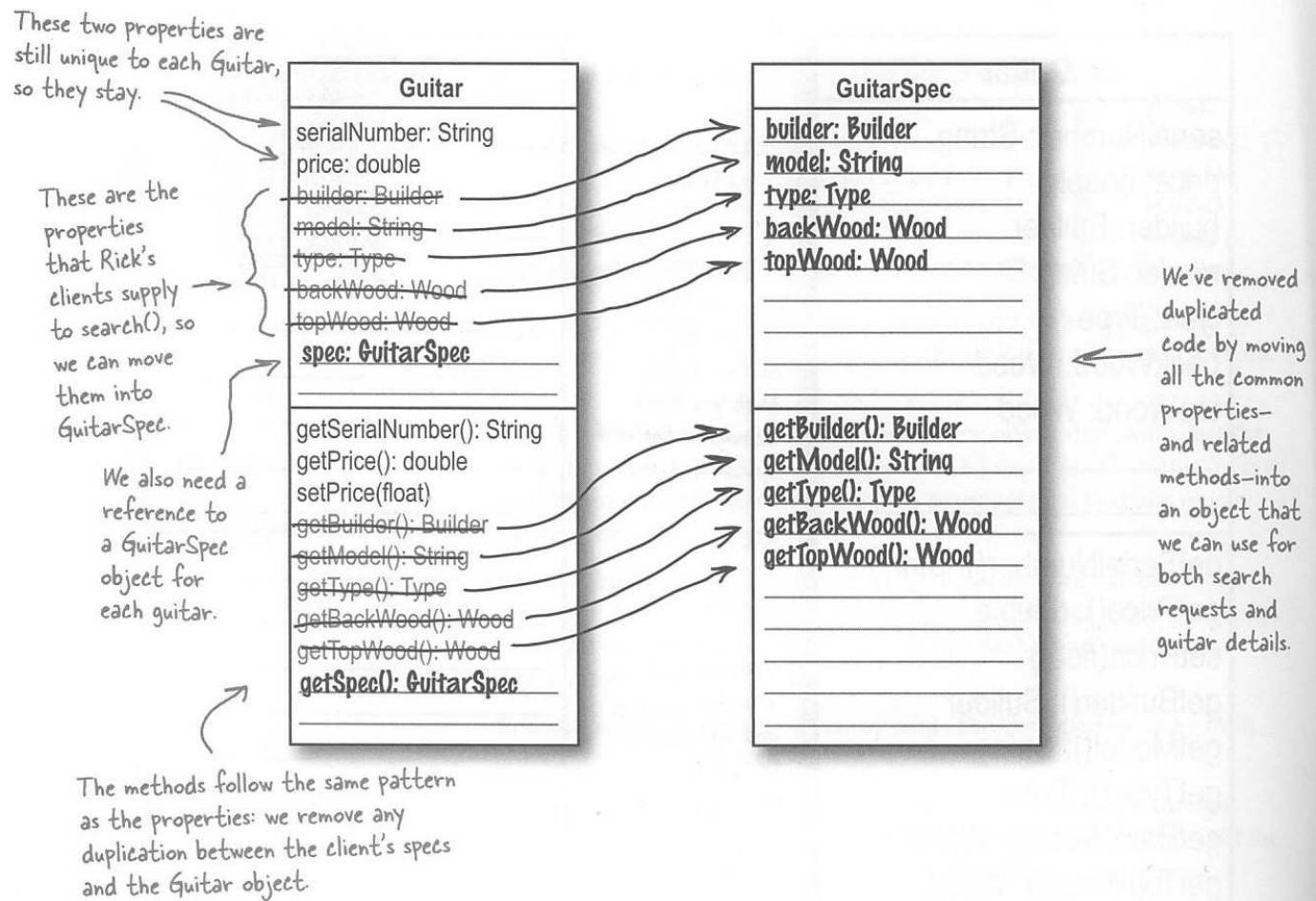
1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Schritt 2: OO-Prinzipien anwenden

- Objekte sollen das sein und tun, was ihre Bezeichnung erwarten läßt.
- Ein Konzept pro Klasse.
 - Objekte, die häufig mit leeren Attributen verwendet werden, deuten darauf hin, daß eine Klasse mehreren Zwecken dient.
- Wie könnte man dieses Design in unserem Beispiel verbessern?

1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

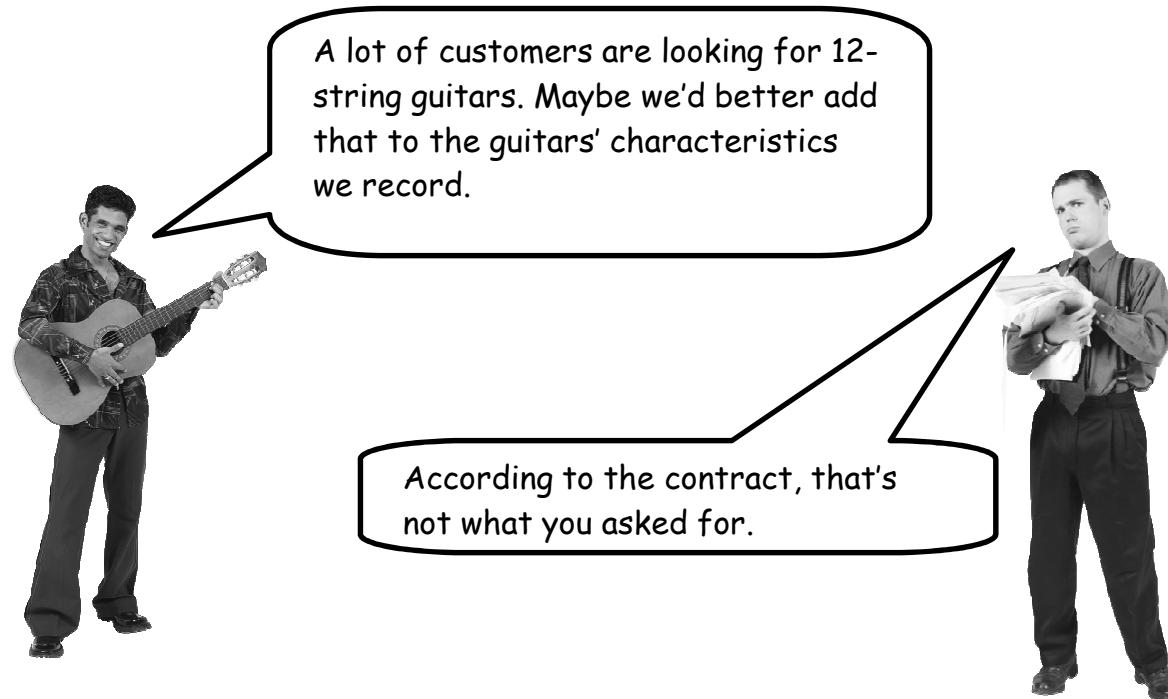
Schritt 2: OO-Prinzipien anwenden



- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Schritt 2: OO-Prinzipien anwenden

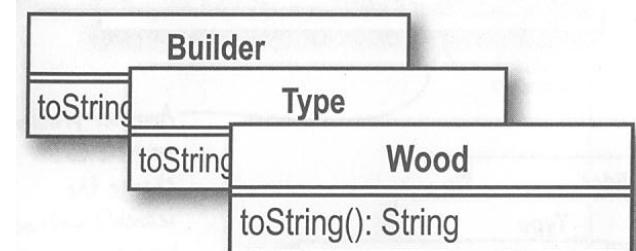
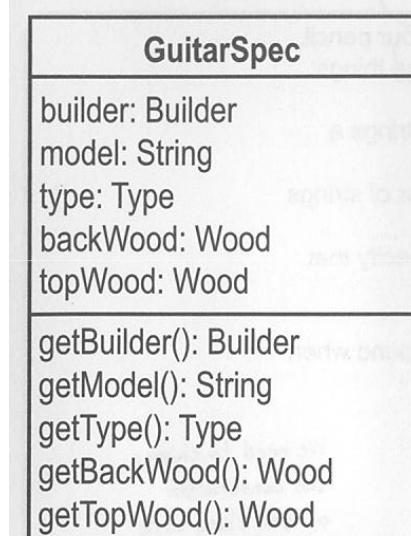
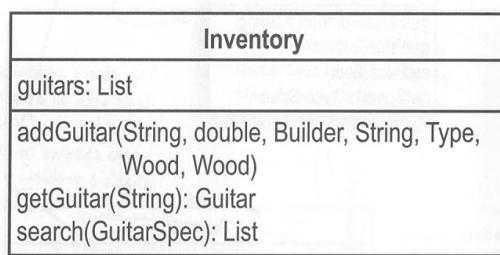
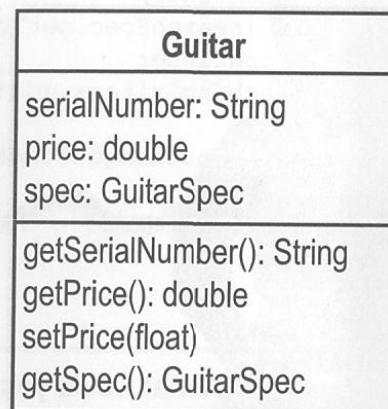
Kapselung erleichtert spätere Veränderungen.



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Schritt 2: OO-Prinzipien anwenden

Was muß geändert werden, um 12-saitige Gitarren verkaufen zu können?

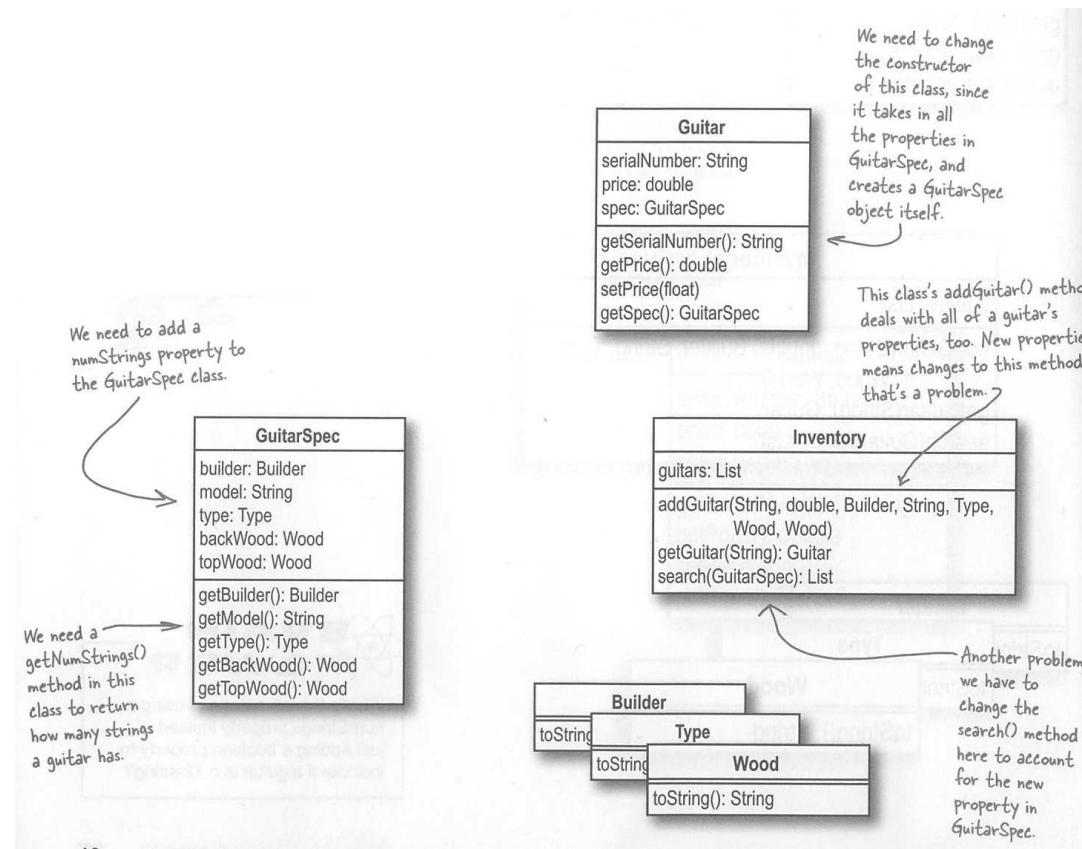




1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Schritt 3: Wiederverwendbarkeit und Wartbarkeit im Auge behalten

- Schlecht: Zusätzliche Attribute (hier: Anzahl der Saiten) müssen in 3 Klassen angepaßt werden.





- 1. Was ist Modellierung?**
- 2. OOAD mit der UML**
- 3. Entity-Relationship-Modelle**
- 4. Geschäftsprozeß-modellierung**

Schritt 3: Wiederverwendbarkeit und Wartbarkeit im Auge behalten

- Wie könnte man die Klassen so modellieren, daß bei zukünftigen neuen Attributen nur noch eine Klasse geändert werden muß?
- 1. Hinweis: Die Klasse, die nicht zuständig ist, sollte die Aufgabe auf die zuständige Klasse abwälzen. (Das nennt man Delegation)
- 2. Hinweis: Im Ergebnis sind die Klassen dann nicht mehr so eng voneinander abhängig. (Man sagt dann, sie sind nur noch lose miteinander verbunden. "loosely coupled")



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Fazit zum Thema "Wozu OOAD?"

- Modelle sind kein Selbstzweck, sondern dienen dazu,
 - den Kunden zufriedenzustellen: Das Programm tut, was es soll. Das Programm kann mit vertretbarem Aufwand verändert werden.
 - Der Entwickler profitiert auch: Teile des Programms können für den nächsten Kunden wiederverwendet werden.

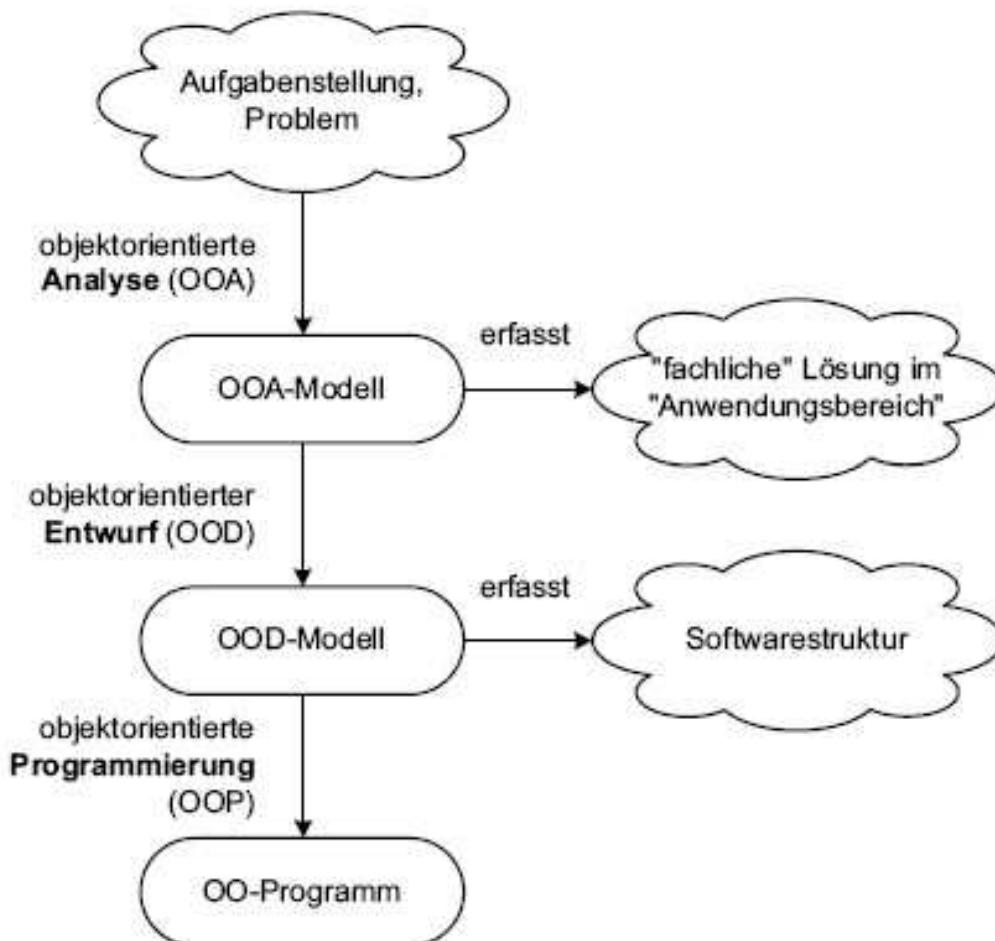


2.3 Was ist die UML?



Was bedeutet OOAD?

1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung



Aus: Tabeling, P., Softwaresysteme und ihre Modellierung, Berlin Heidelberg 2006, S. 332

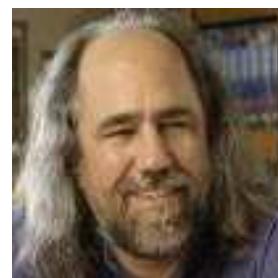


1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Was ist die UML?

- Um die Anforderungen an ein Softwaresystem und den technischen Aufbau des Systems besser zu verstehen und darüber diskutieren zu können, ist ein graphisches Modell hilfreich.
- Als in den 90er Jahren die Objektorientierung an Bedeutung gewann, gab es viele konkurrierende Modellierungssprachen. Drei der wichtigsten stammten von den sogenannten "Amigos":

Grady Booch



Ivar Jacobson



Jim Rumbaugh





1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Was ist die UML?

- Die Rational Software Corporation hat 1994
 - alle drei eingestellt,
 - sie gemeinsam eine Modellierungssprache entwerfen lassen,
 - das Ergebnis UML genannt,
 - es von der OMG standardisieren lassen,
 - es frei verfügbar gemacht und
 - unter der Marke *Rose* Software zur Erstellung von solchen UML-Modellen produziert.
- Rational gehört inzwischen zur IBM.



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Wie erstellt man UML-Diagramme?

- Andere Software zur Erzeugung von UML-Diagrammen ist z.B.:
 - umbrello (freie Software),
 - MS Visio
 - Enterprise Architect
- Einige Produkte können auch aus UML-Modellen Programmfragmente generieren und umgekehrt.
- Inzwischen gibt es auch Webseiten, die online die Erstellung von UML-Diagrammen erlauben, z.B. creately.com



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Welche Vorteile hat die UML?

- Eindeutigkeit: Wer zeichnet, muß sich festlegen und ist gezwungen, sich über die Alternativen Gedanken zu machen.
- Verständlichkeit: Wie in allen Lebensbereichen erleichtern graphische Modelle das Verständnis komplexer Zusammenhänge.
- Ausdrucksstärke: Die vielen verschiedenen UML-Diagramme betonen verschiedene Aspekte. Sie erlauben es, sich auf einen Teilaспект zu konzentrieren, ergeben zusammen aber auch einen umfassenden Blick auf ein System.



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Welche Vorteile hat die UML?

- Standardisierung: Die UML ist weltweit anerkannt und verbreitet. Da fast jeder, der sich mit dem Thema objektorientierte Softwareentwicklung auskennt, auch die UML kennt, vereinfacht sie die Kommunikation in den Projekten.
- Plattformunabhängigkeit: Man legt sich nicht auf eine Programmiersprache oder Plattform fest.

Naja zugegeben: Für die Wartung eines 30 Jahre alten Cobol-Systems unter MVS kommt die UML kaum zum Einsatz.



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Wie verwendet man die UML?

- Die aktuelle Version ist UML 2
- Die UML definiert 14 Diagrammtypen, von denen 13 für Anwender der UML relevant sind.
- Elemente der Diagrammtypen dürfen auch gemischt verwendet werden, wenn es sinnvoll ist.
- Üblicherweise werden in einem Projekt nicht alle 13 Diagrammarten genutzt, sondern eine Auswahl getroffen.
- Diese Auswahl ist immer subjektiv. Sie muß sich an Fragen orientieren wie:
 - Wozu dient das Modell? (Zweck)
 - Für wen ist es gedacht (Zielgruppe)
 - In welchem Verhältnis stehen Kosten und Nutzen eines Modells?



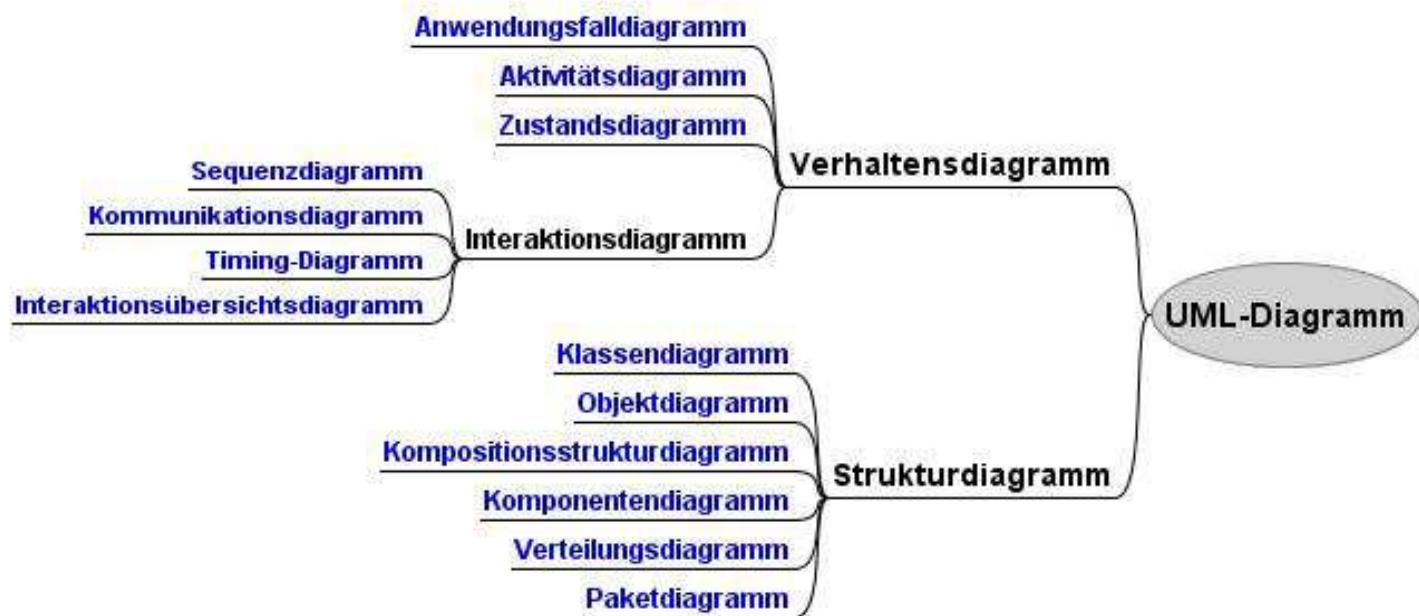
1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Wie verwendet man die UML?

- UML-Diagramme können sowohl ein Geschäftssystem (eine Organisation) als auch ein IT-System beschreiben. Vor Beginn der Modellierung steht immer die Frage: Was genau ist das zu modellierende System und welchen Zweck soll das Modell erfüllen?
- Bsp.:
 - Modelliere ich, wie ein Kunde die Organisation sieht oder wie ein Benutzer das IT-System sieht?
 - Modelliere ich Abläufe im System (interne Sicht als Vorlage für die Entwickler) oder nur die Interaktion des Benutzers mit dem System (externe Sicht als Diskussionsgrundlage für die Spezifikation der Funktionalität, später auch als Dokumentation des Systems)?

1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Welche UML-Diagramme gibt es?



Das 14. Diagramm, das Profildiagramm, gehört zu den Strukturdiagrammen. Es wird nur zur Metamodellierung verwendet. Für praktische IT-Projekte ist es nicht relevant.



- 1. Was ist Modellierung?**
- 2. OOAD mit der UML**
- 3. Entity-Relationship-Modelle**
- 4. Geschäftsprozeß-modellierung**

Weitere Methoden, die bei Analyse und Design zum Einsatz kommen

In den meisten Projekten zur Objektorientierten Softwareentwicklung kommen in der Analysephase zusätzlich zu den UML-Diagrammen zwei Verfahren zur Anwendung, die nicht standardisiert sind:

- Screen-Prototypes
- Use-Case-Dokumente



- 1. Was ist Modellierung?**
- 2. OOAD mit der UML**
- 3. Entity-Relationship-Modelle**
- 4. Geschäftsprozeß-modellierung**

Weitere Methoden, die bei Analyse und Design zum Einsatz kommen: Screen-Prototypes

- Endbenutzer sind daran interessiert, frühzeitig Bildschirm-Prototypen zu sehen. Anhand zeichnerischer Darstellungen der GUI oder anhand einer GUI ohne Funktionalität können sie sich ein Bild vom zu entwickelnden System machen. Dadurch klären sich Mißverständnisse bei der Anforderungsanalyse auf.



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Weitere Methoden, die bei Analyse und Design zum Einsatz kommen: Use-Case-Dokumente

- Anwendungsfälle können mit UML-Use-Case-Diagrammen übersichtlich dargestellt werden, die auch von Endbenutzern recht schnell verstanden werden.
- Anwendungsfälle können z.B. mit UML-Aktivitäts- und Sequenzdiagrammen auch detailliert beschrieben werden. Diese Modelle eignen sich als Vorlage für die Entwickler.
- Endbenutzer kennen aber normalerweise die UML nicht. Daher werden die Anwendungsfälle oft zusätzlich detailliert verbal in Dokumenten beschrieben.

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Übungsaufgabe

Der Geldautomat einer Bank soll um eine Funktion erweitert werden: Außer *Auszahlung* und *Kontostandabfrage* kann man auch *Spenden* wählen. Auf dem Folgebildschirm kann man dann zwischen 4 gemeinnützigen Organisationen wählen und einen Betrag eingeben.

Erstellen Sie Screen Prototypes für den Touchscreen des GAA.



2.4 OOA: Use-Case-Dokumente und Use-Case-Diagramme



2.4.1 Use-Case-Dokumente



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Objektorientierte Analyse (OOA)

- Es ist die Verantwortung des Entwicklers, die Anforderungen der Kunden verstanden zu haben.
- Es klingt banal, wird aber oft übergangen:
 - Mit dem Kunden sprechen
 - Mit anderen Beteiligten sprechen.
 - Fragen stellen.
 - Im Team diskutieren
 - Die Aufgabe aus unterschiedlichen Blickwinkeln betrachten



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeßmodellierung

Anwendungsfälle (Use-Cases)

- Use-Case: Ein Akteur interagiert mit dem zu entwerfenden System mit einer bestimmten Absicht.
- Dies als Fließtext zu beschreiben, wäre zu unübersichtlich.
- Ein Diagramm wäre dafür oft zu detailliert und doch zu schwer verständlich für die Anwender.
- Deshalb verwendet man eine Semi-Strukturierte Methode: Dokumente, die einer gegebenen Struktur folgen.
- Diese Struktur sollte für das Projekt einheitlich sein für alle Use-Case-Dokumente.



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Was kennzeichnet einen Use-Case?

- Ein Use-Case bringt dem Anwender einen bestimmten Nutzen.
- Er hat einen definierten Anfang und ein definiertes Ende.
- Er wird von einem Akteur außerhalb des Systems initiiert.
- Ein Use-Case muß alle Pfade vom Ausgangspunkt bis zum Ziel enthalten.





- 1. Was ist Modellierung?**
- 2. OOAD mit der UML**
- 3. Entity-Relationship-Modelle**
- 4. Geschäftsprozeß-modellierung**

Inhalt von Use-Case-Dokumenten

- In jedem Use-Case-Dokument wird mindestens beschrieben,
 - welche Akteure mit dem System
 - unter welchen Bedingungen
 - mit welchem Ziel interagieren,
 - wodurch dies ausgelöst wird,
 - welche Aktionen dabei im Normalfall ablaufen und
 - welche Ausnahmen vom Normalfall auftreten können.
- Den größten Teil nimmt der *Ablauf im Normalfall* ein. Aktionen werden in der vorgesehenen Reihenfolge detailliert beschrieben. Dabei können auch Steuerelemente wie "Solange" oder "Wenn/Dann/Sonst" verwendet werden.



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Aufbau von Use-Case-Dokumenten

- *Name und ID*: Die Bezeichnung des UC
- *Status*: Z.B. In Arbeit, abgenommen, in Überarbeitung, ...
- *Version*: Mit Autoren und Datum
- *Description*: Eine Beschreibung, worum es bei dem UC geht, welchen Zweck er erfüllt, was der Nutzen für den Anwender ist.
- *Actors*: Welche Akteure nutzen das System? Das können Personen oder andere Systeme sein.
- *Basic Flow*: Ablauf im Normalfall
- *Alternate Flows*: Ausnahmen vom Normalfall
- *Trigger*: Gründe, daß der UC initiiert wird.
- *Preconditions*: Bedingungen, unter denen der UC initiiert wird.
- *Postconditions*: Bedingungen, die nach Abschluß des UC erfüllt sind.
- *Related Use-Cases*: Welche anderen Anwendungsfälle sind mit diesem per <<include>> oder <<extend>> verbunden?

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Beispiel für eine Ablaufbeschreibung in einem Use-Case-Dokument

UC8 Flow of Events for the *Buy House* Use Case

8.1 Preconditions:

1. It is the player's turn.
2. The player has not rolled the dice.
3. The player has monopoly on one or more color groups.

8.2 Main Flow:

When a player has all the tradable cells in a color group, this player is said to have monopoly on the color group. A player may build house(s) in the property cells in the color groups the player has monopoly on by pressing the Buy House button before he or she rolls the dice [S1] [E1 – E2]. The price of the house is determined by the cell. After buying the house(s), the status of the player is updated and displayed on the game board [UC13].

8.3 Subflows:

- [S1] When the Buy House button is clicked, the Buy House dialog shows up. The player selects the monopoly color group and the number of houses from that dialog. After clicking on OK in the dialog box, the player pays the fee, and the houses are created. All the property cells in the selected color group have the same number of houses.

8.4 Alternative Flows:

- [E1] Nothing happens if the player does not have enough money.
[E2] The player can build at most five houses in a cell.

Quelle: <http://agile.csc.ncsu.edu/SEMMaterials/UseCaseRequirements.pdf>

- 
- 1. Was ist Modellierung?**
 - 2. OOAD mit der UML**
 - 3. Entity-Relationship-Modelle**
 - 4. Geschäftsprozeß-modellierung**

Übungsaufgabe

- Lesen Sie das Use-Case-Dokument für den Use Case *Geld abheben* am Geldautomaten unter
http://www.utm.mx/~caff/doc/OpenUPWeb/openup/guidances/examples/resources/use_case_spec_withdraw_cash.doc
- Erstellen Sie nach dieser Vorlage ein Use-Case-Dokument für den Use Case *Freundschaftsanfrage annehmen* in einem Sozialen Netzwerk.



2.4.2 Use-Case-Diagramme



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Zusammenfassung einiger Prinzipien

- *encapsulate what varies*: Man soll alles kapseln, was sich verändern kann.
- *understand requirements*: Es ist die Verantwortung des Entwicklers, das System zu verstehen.
- *code to interfaces rather than to an implementation*: Der Code einer Klasse soll so unabhängig wie möglich vom Code der anderen Klassen sein.





1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Domänenanalyse

- Um die Aufgabe zu verstehen, muß man sie in die Sprache des Anwenders beschreiben.
- Dazu gehört, Informationen über das Umfeld des zu entwickelnden Systems zu sammeln und zu strukturieren.
- Informationsquellen sind vor allem der Kunde und die zukünftigen Anwender (meist die Mitarbeiter unseres Kunden, manchmal aber z.B. auch die Kunden unseres Kunden).
- Weitere Informationsquellen können z.B. sein:
 - existierende Systeme, mit denen unser System zusammenarbeiten soll oder die es ablösen soll
 - Theorien und Techniken der Domäne (Z.B. sollte man über grundlegende medizinische Kenntnisse verfügen, um eine Diagnosesoftware zu entwickeln.)





1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

OOA für große Projekte

- Unsere Beispiele führten bislang zu recht überschaubaren Use-Case-Dokumenten und Klassendiagrammen.
- Projekte in der Praxis sind meist viel umfangreicher.
- Die gleichen Methoden und Prinzipien lassen sich auch auf viel größere Projekte anwenden.
- Man muß dazu die Gesamtaufgabe sinnvoll in Teilaufgaben gliedern.

- 1. Was ist
Modellierung?**
- 2. OOAD mit der
UML**
- 3. Entity-
Relationship-
Modelle**
- 4. Geschäftsprozeß-
modellierung**

Die Aufgabe

Gary's Games
Vision Statement

Gary's Games provides frameworks that game designers can use to create turn-based strategy games. Unlike arcade-style shoot-'em-up games and games that rely on audio and video features to engage the player, our games will focus on the technical details of strategy and tactics. Our framework provides the bookkeeping details to make building a particular game easy, while removing the burden of coding repetitive tasks from the game design.

The game system framework (GSF) will provide the core of all of Gary's Games. It will be delivered as a library of classes with a well-defined API that should be usable by all board game development project teams within the company. The framework will provide standard capabilities for:

- ♦ Defining and representing a board configuration
- ♦ Defining troops and configuring armies or other fighting units
- ♦ Moving units on the board
- ♦ Determining legal moves
- ♦ Conducting battles
- ♦ Providing unit information

The GSF will simplify the task of developing a turn-based strategic board game so that the users of the GSF can devote their time to implementing the actual games.





1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Features

- Aus den Kundengesprächen ergibt sich grob, was das System können soll.
- Daraus müssen die Anforderungen abgeleitet werden.
 - . Feature: Verschiedene Gelände sind als Spielfeld verfügbar.
 - Requirement: Der Anwender kann eigene Gelände definieren.
 - Requirement: Elemente wie Flüsse und Berge stehen dafür zur Verfügung.
 - Requirement: Das Gelände bestimmt, wie die Truppen sich bewegen können.



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeßmodellierung

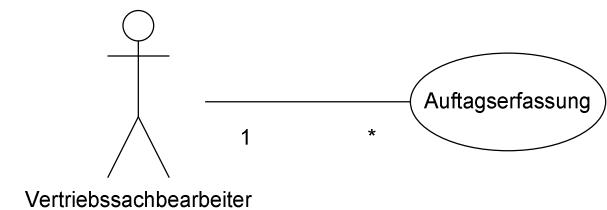
Use-Case-Diagramme

- Bevor man Use-Case-Dokumente für ein größeres System schreibt, braucht man einen Überblick, welche Use-Cases es überhaupt gibt.
- Use-Case-Diagramme geben auf einer sehr hohen Ebene Auskunft über die Funktionen eines Systems.
 - Sie abstrahieren von Details.
 - Sie geben nur Auskunft darüber, *was* das System tut, nicht *wie*.
 - Sie definieren auch keine Reihenfolge der Use Cases.

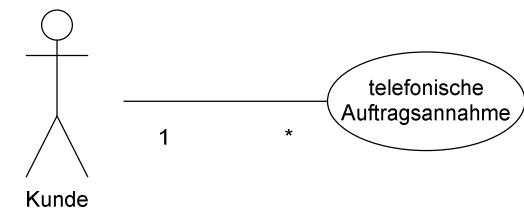
- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Use-Case-Diagramme

- Use cases beschreiben die Interaktion zwischen einem Akteur und einem System.
 - Das kann ein IT-System sein.



- Es kann aber auch ein Geschäftssystem sein.



- Use-Case-Diagramme beschreiben damit die Funktionalität eines Systems.
- Sie enthalten ein großes Rechteck, das die Systemgrenze darstellt. Damit ist klar, worauf sich das Diagramm bezieht.



Use-Case-Diagramme

1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

- Ein Use-Case-Diagramm besteht u.a. aus folgenden Elementen:
 - Akteur (Actor): Rolle, die ein externer Benutzer oder ein externes System einnimmt, wenn er mit dem System interagiert. Ein Akteur steht immer außerhalb des betrachteten Systems. Ein Akteur ist eine generische Einheit, heißt also z.B. "Kunde" und nicht "Herr Maier".





1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Use-Case-Diagramme

- Ein Use-Case-Diagramm besteht u.a. aus folgenden Elementen:
 - Anwendungsfall (Use Case): Funktionalität des betrachteten Systems, die ein Akteur nutzt. Ein Use Case besteht aus einer Menge von Aktionen. Er bringt dem Akteur in irgendeiner Form einen Nutzen.



Zur Granularität von Anwendungsfällen:

- Für jedes ovale Symbol im Use-Case-Diagramm wird ein Use-Case-Dokument erstellt.
- Man muß also abwägen zwischen
 - der Übersichtlichkeit des Diagramms
 - der Überschaubarkeit der Use-Case-Dokumente
 - der Überschneidungsfreiheit der Use-Case-Dokumente
- Alistair Cockburn's Coffee Break Test: "After I get done with this, I can take a coffee break."
 - D.h. Ein Use-Case ist zu umfangreich, wenn ein Akteur während des Use-Cases eine Kaffeepause machen würde.



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Use-Case-Diagramme

- Ein Use-Case-Diagramm besteht u.a. aus folgenden Elementen:
 - Assoziation: Beziehung zwischen einem Akteur und einem Anwendungsfall, den er nutzt.
Eine gerichtete Assoziation (mit Pfeilspitze) zeigt an, ob der Akteur den Use Case initiieren kann. Eine ungerichtete Assoziation lässt das offen.
Multiplizitäten können wie zwischen Klassen angegeben werden. Sie geben an, wie viele Objekte bei einem Aufruf des Use Cases von wie vielen Akteuren bearbeitet werden können.



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Use-Case-Diagramme

- Ein Use-Case-Diagramm besteht u.a. aus folgenden Elementen:
 - Beziehungen zwischen Anwendungsfällen: Ein Anwendungsfall wird immer (<<include>>) oder unter bestimmten Bedingungen (<<extend>>) in einen anderen eingebunden.
Sie sind sinnvoll, wenn ein Use Case einen anderen erweitert oder wenn gemeinsame Funktionalität in mehreren Use Cases verwendet wird.
Sie dürfen natürlich keinen Zyklus bilden.

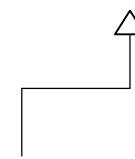
----->



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Use-Case-Diagramme

- Ein Use-Case-Diagramm besteht u.a. aus folgenden Elementen:
 - Generalisierungen zwischen Akteuren können sinnvoll sein und werden mit Generalisierungspfeilen wie bei Klassen gezeichnet.
 - Generalisierungen zwischen Use Cases sind ebenfalls möglich, aber eher unüblich.





1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

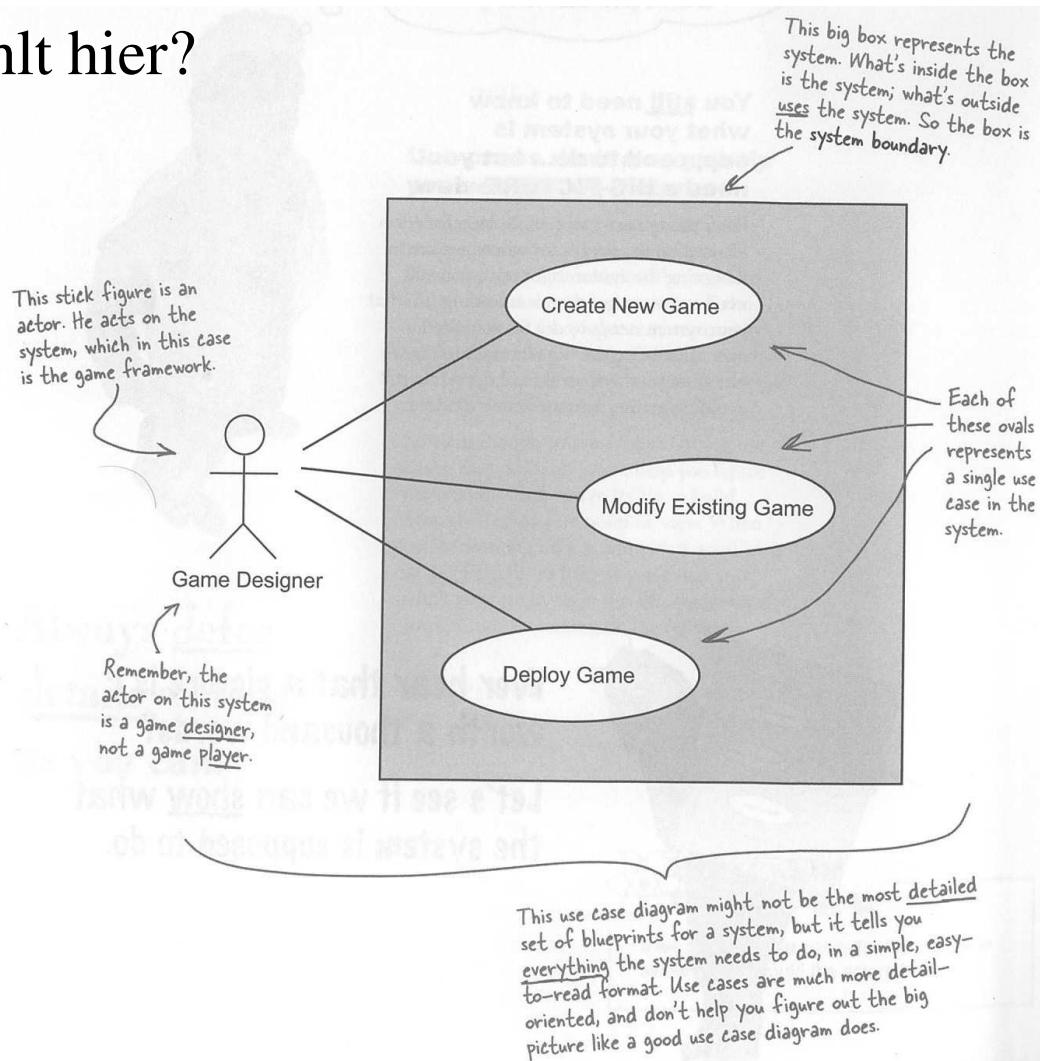
Mögliche Fragen zum Erstellen von Use-Case-Diagrammen

- Akteure: Welche Personen und externen Systeme sind die Akteure?
 - Akteure stehen außerhalb des zu modellierenden Systems
- Anwendungsfälle: Welche Anwendungsfälle gibt es?
 - Anwendungsfälle haben ein Ergebnis für den Akteur. Sie werden von ihm initiiert.
- Assoziationen: Welchen Akteuren stehen welche Anwendungsfälle zur Verfügung?
- include/extend: In welcher Beziehung stehen die Anwendungsfälle zueinander?

1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Use-Case-Diagramme

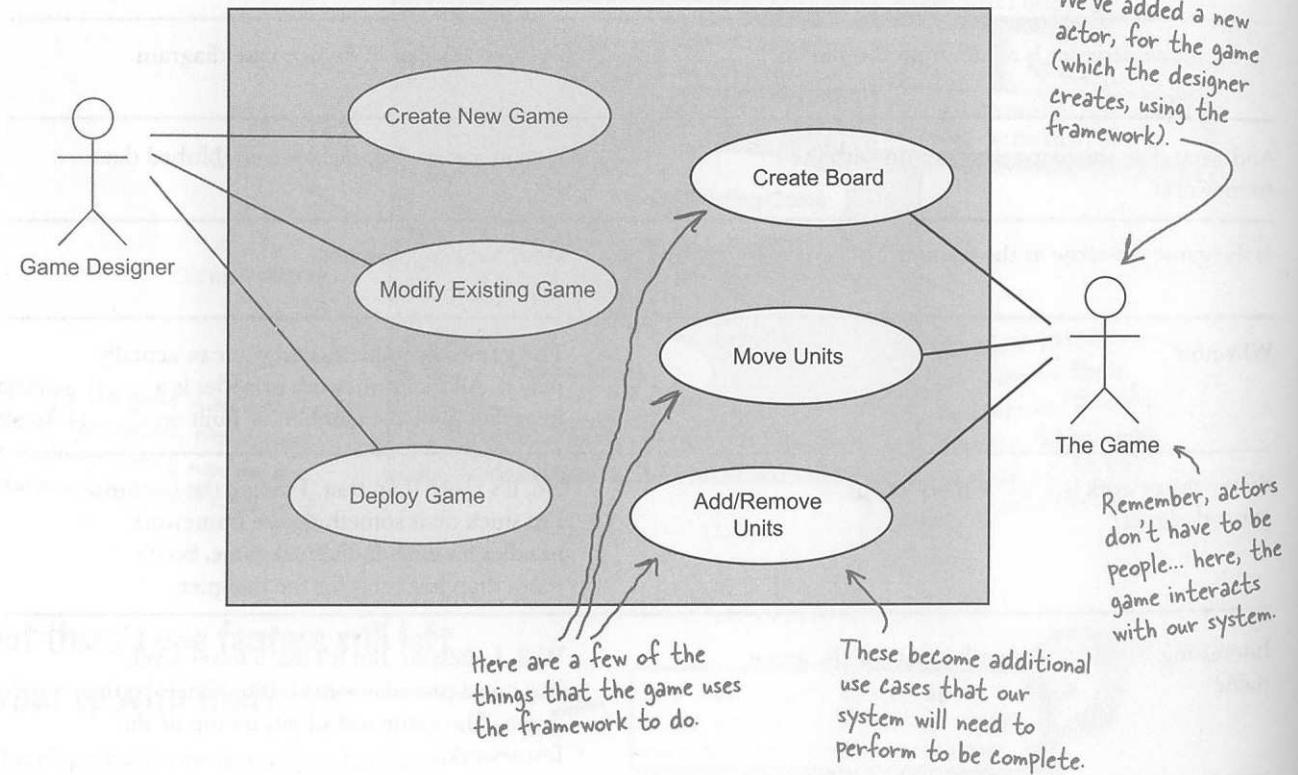
Was fehlt hier?



- 1. Was ist Modellierung?**
- 2. OOAD mit der UML**
- 3. Entity-Relationship-Modelle**
- 4. Geschäftsprozeß-modellierung**

Akteure sind auch nur Menschen (aber nicht immer)

It turns out that in addition to the game designer, the game itself is an actor on the framework you're building. Let's see how we can add a new actor to our use case diagram:

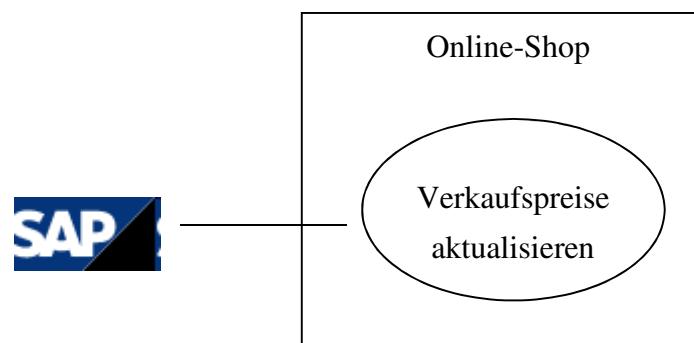




1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Use-Case-Diagramme

- Noch schöner ist es, nicht-menschliche Akteure als Rechteck statt als Strichmännchen darzustellen. Die UML erlaubt aber in beiden Fällen beides und außerdem eigene Symbole, wenn diese aussagekräftiger sind, z.B. hier:





1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Use-Case-Diagramme

- Vor allem die <<include>> und <<extends>>-Beziehungen verleiten dazu, das Use-Case-Diagramm als Ablaufplan zu missbrauchen. Dafür gibt es in der UML aber die Aktivitätsdiagramme. Use-Case-Diagramme beschreiben nur das *Was*, nicht das *Wie*.
- Die UML verlangt auch bei Use-Case-Diagrammen keine Vollständigkeit, sondern Zweckmäßigkeit. Ob eine Beziehung ins Diagramm gehört oder nicht, hängt von Einsatzzweck und Zielgruppe des Diagramms ab.
- Anhand der ersten "Feature List" kann man das Use-Case-Diagramm auf Vollständigkeit prüfen.



Aufteilung der Gesamtaufgabe

- Die Use-Cases sind ein Anhaltspunkt, wie man die Gesamtaufgabe sinnvoll in kleinere Pakete mit unterschiedlichen Zuständigkeiten unterteilen kann.
- Dies ist aber nicht zwingend. Oft sind andere Kriterien wichtiger. Welche fallen Ihnen ein?
- Einen sehr wichtigen Aspekt können wir hier nur am Rande erwähnen: Entwurfsmuster (Design Patterns) sind Beispiele erfolgreicher Systementwürfe. Das wohl bekannteste kennen Sie vielleicht schon: Es heißt Model-View-Controller (MVC)



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Aufgabe (lassen Sie 10 und 15 einfach leer)

 **OO&D Cross**

It's time for another left-brain workout. Below is a puzzle with lots of blank squares; to the right are some clues. You know what to do, so go for it!

Across

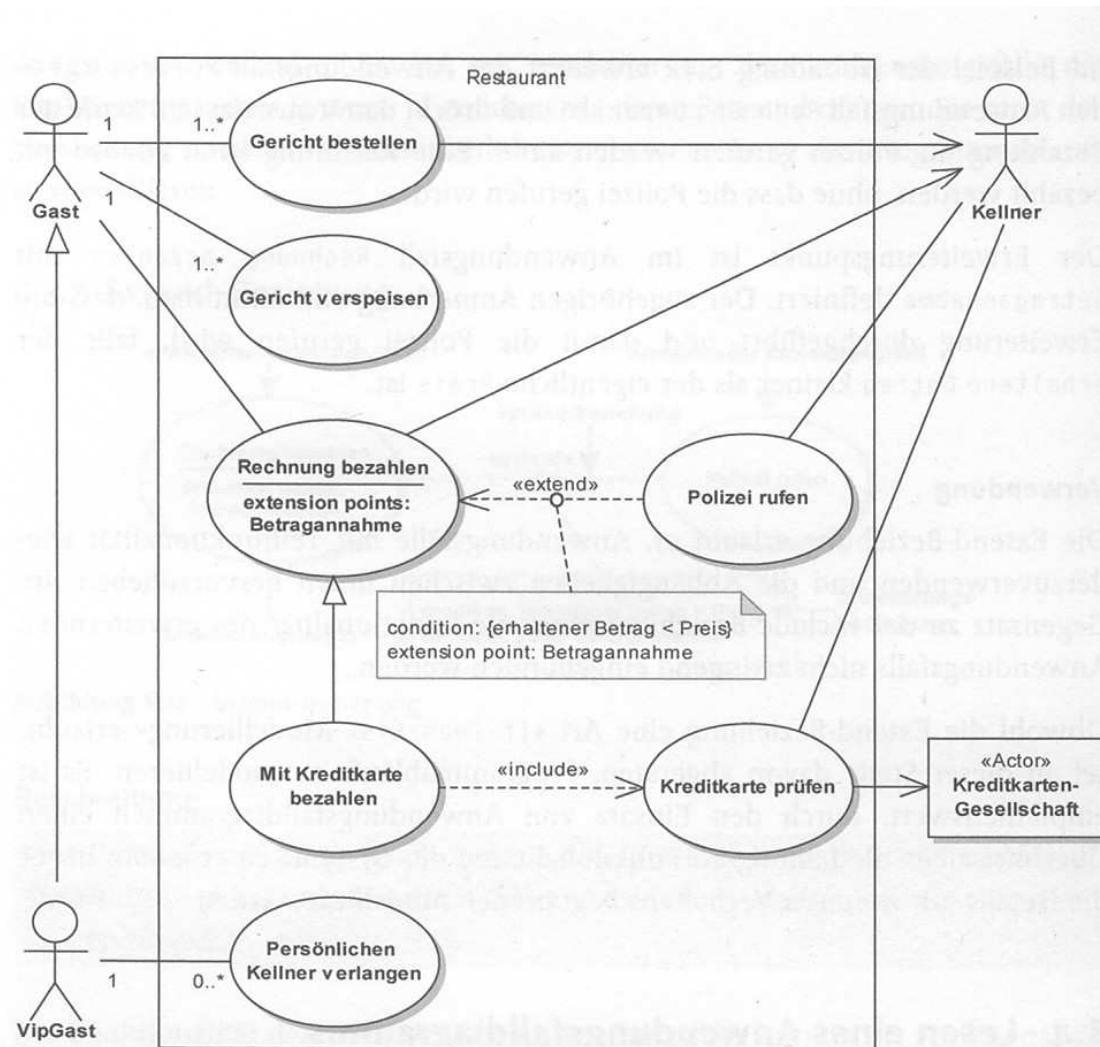
2. This helps you speak to the customer in their language (2 words)
3. You can use your feature list to make sure your use case diagram is this.
5. Use cases don't always help you see this (2 words).
7. These aren't always people.
8. A feature is a _____ description of something a system needs to do (2 words)
10. Art Vandelay's "real" last name
12. A use case diagram acts as this for your system.
14. You can figure out these based on your features.
16. This is the measure of how things are similar.

Down

1. An oval in a use case diagram represents one of these.
2. We applied ones of these to your Gary's framework.
4. The measure of how things are different.
6. You should solve a big problem by doing this to it (3 words)
9. You can figure out a system's features by _____ to the customer.
11. You solve big problems the _____ you solve small problems (2 words).
13. Defer these as long as possible.
15. He wasn't the customer.

1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeßmodellierung

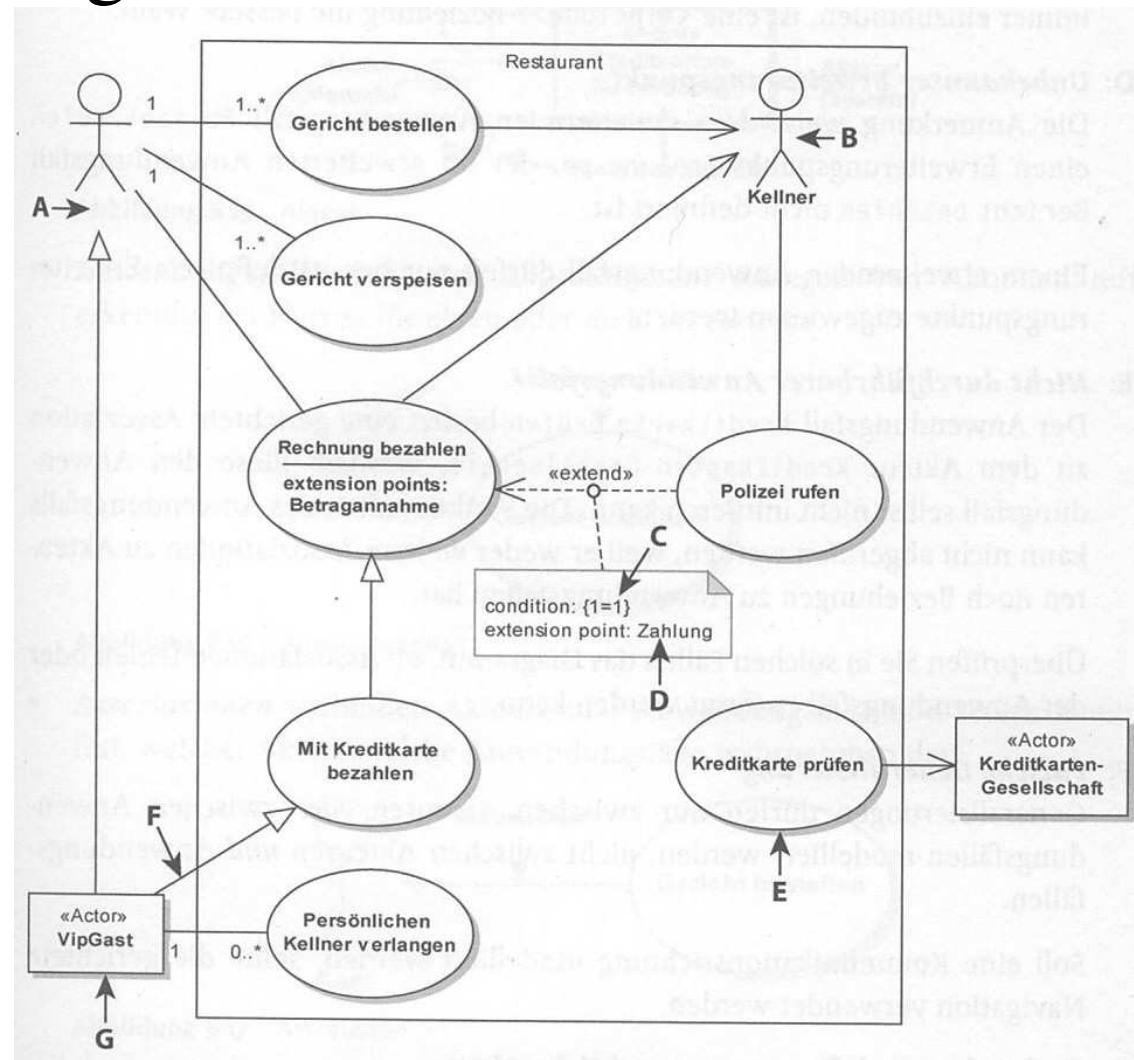
Beschreiben Sie verbal dieses Diagramm



Aus: Kecher, C., UML 2.0, Bonn 2006, S. 208

1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeßmodellierung

Bennennen Sie die Fehler, ohne im Original nachzusehen



Aus: Kecher, C., UML 2.0, Bonn 2006, S. 209

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Übungsaufgabe

- 1) Erstellen Sie ein Use-Case-Diagramm für das Gästebuch einer Unternehmenshomepage.
- 2) Erstellen Sie ein Use-Case-Diagramm für den Geldautomaten Ihrer Bank.



2.5 OOD: Klassendiagramme im Detail



- 1. Was ist Modellierung?**
- 2. OOAD mit der UML**
- 3. Entity-Relationship-Modelle**
- 4. Geschäftsprozeß-modellierung**

Klassendiagramme

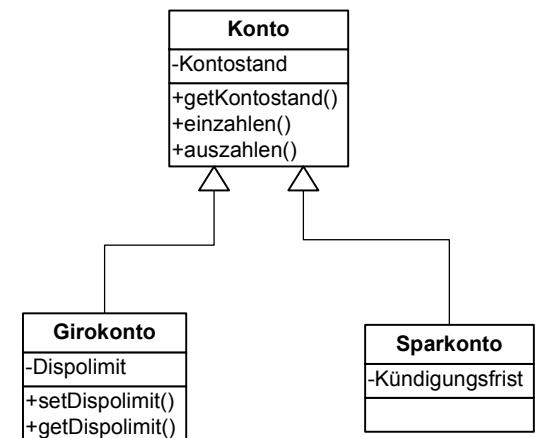
- Klassendiagramme beschreiben die Struktur eines objektorientierten Systems.
- Klassendiagramme können der Diskussion mit dem Anwender dienen. Für diesen Zweck sollte man sich auf die für den Anwender wesentlichen Klassen beschränken.
- Klassendiagramme beschreiben aber vor allem auch die interne Struktur eines Systems als Vorlage für die Entwicklung. Sie müssen deshalb alle Details enthalten.



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Klassendiagramme

- Ein Klassendiagramm besteht u.a. aus folgenden Elementen:
 - Klassen entstehen, indem ähnliche Objekte zusammengefaßt werden. Sie enthalten Attribute und Methoden.
 - Generalisierungspfeile zeigen von der speziellen auf die allgemeine Klasse.



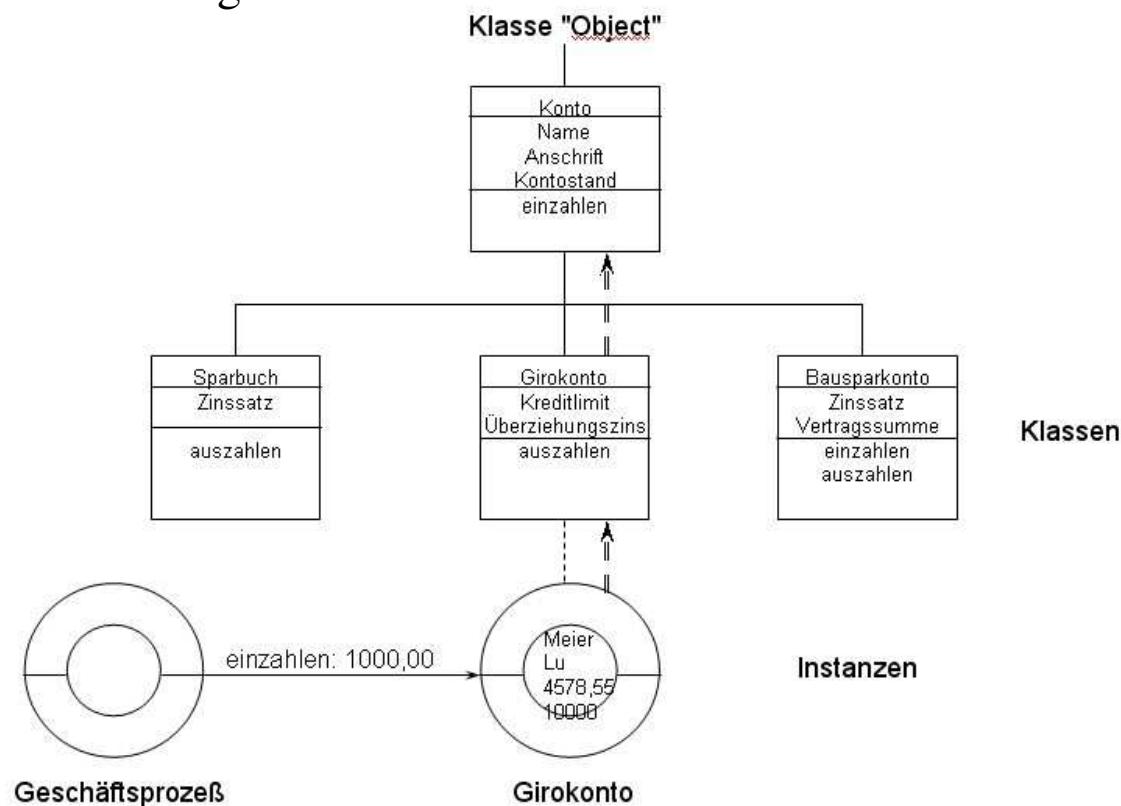


1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Geerbte Attribute und Methoden können überschrieben werden

Welche Methode wird für ein Objekt aufgerufen?

In der Klassenhierarchie wird von unten nach oben nach einer passenden Methode gesucht.



- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Übungsaufgabe

Flugzeuge werden mit Gepäckstücken und Frachtstücken beladen. Frachtstücke unterscheiden sich von Gepäckstücken dadurch, daß sie eine Gefährlichkeitsklasse haben und abhängig davon spezielle Vorkehrungen bei der Verladung zu treffen sind. Modellieren Sie einen Ausschnitt aus einem Klassendiagramm, der Gepäck- und Frachtstücke beschreibt.

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Klassendiagramme

- Ein Klassendiagramm besteht u.a. aus folgenden Elementen:
 - Assoziationen stellen eine Beziehung zwischen zwei Klassen dar.
 - Die Kardinalität gibt an, wie viele Objekte an einer Assoziation beteiligt sein können.

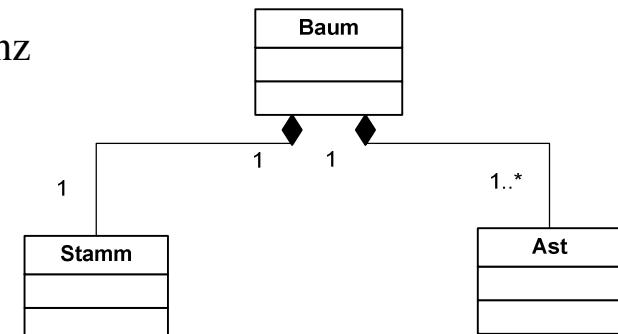
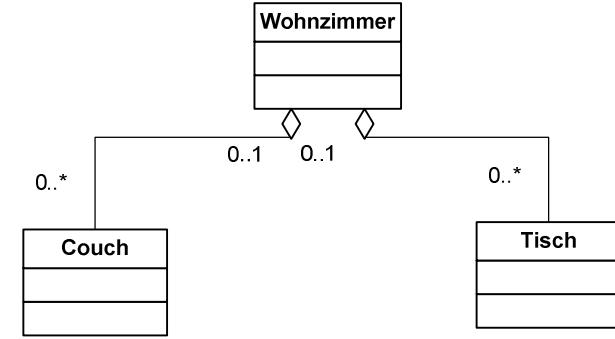


- 1. Was ist Modellierung?**
- 2. OOAD mit der UML**
- 3. Entity-Relationship-Modelle**
- 4. Geschäftsprozeß-modellierung**

Klassendiagramme

Ein Klassendiagramm besteht u.a. aus folgenden Elementen:

- Aggregation ist eine Form der Assoziation, bei der sich ein Objekt der einen Klasse u.a. aus einem Objekt der anderen Klasse zusammensetzt.
- Komposition ist eine Aggregation, bei der ein Objekt untrennbar mit einem Ganzen verbunden ist.
Wenn das ganze aufhört zu existieren, endet auch die Existenz der Teile.





1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Übungsaufgaben

- Überlegen Sie sich Beispiele für die folgenden Arten von Beziehungen zwischen Klassen: a) Komposition, b) Aggregation in allen Ausprägungen, c) Generalisierung, d) Assoziationen, die keine Kompositionen, Aggregationen oder Generalisierungen sind.
- Bestimmen Sie die Multiplizitäten der Beziehungen zwischen: Kunde, Ticket, Coupon, Flug, Flugnummer (Ein Coupon gilt für einen Streckenabschnitt, ein Flug ist durch Flugnummer und Datum gekennzeichnet.)



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeßmodellierung

Mögliche Fragen zum Erstellen von Klassendiagrammen

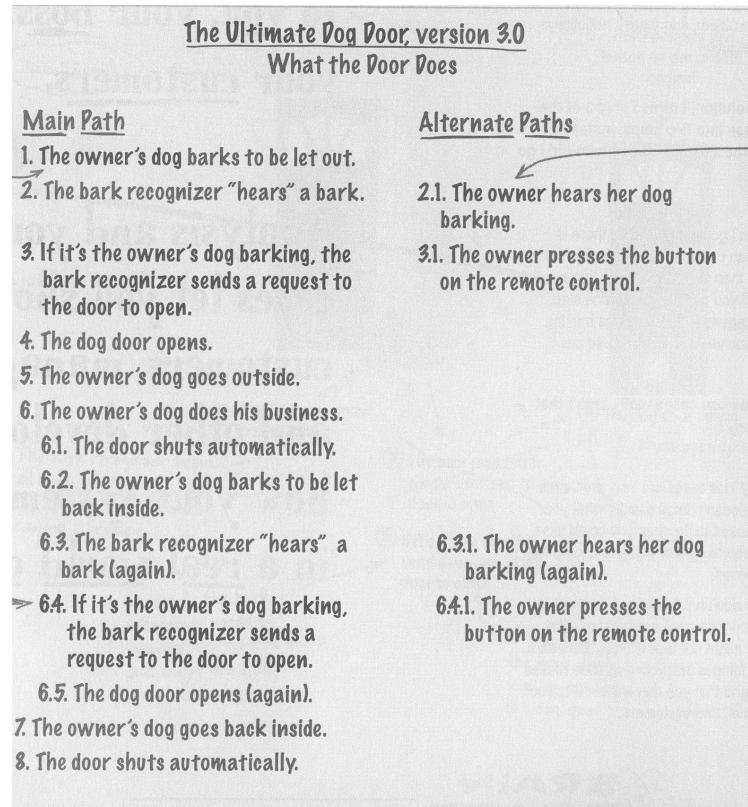
- Welche Informationen muß das System bereitstellen können?
- Klassen: Welche Objekte und Akteure gibt es?
- Assoziationen: Welche statischen Regeln gelten für die Beziehungen zwischen Objekten?
- Assoziationen: Welche Beziehungen zwischen Klassen bestehen direkt (also nicht über eine dritte Klasse)?
- Attribute: Wodurch werden Objekte beschrieben?



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Welche Klassen werden benötigt?

- Die Klassen stehen meistens als Substantive im Use-Case-Dokument.
- Aber nicht jedes Substantiv repräsentiert eine Klasse. (Weil einige Dinge außerhalb unseres Systems stehen)
- Welche Klassen sind das in unserem Beispiel?

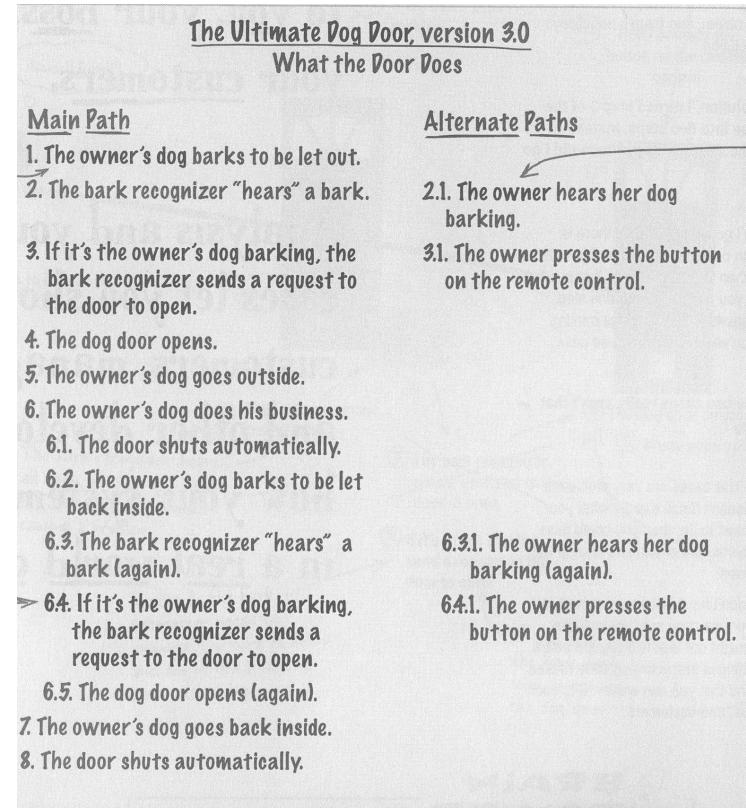




1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Welche Methoden werden benötigt?

- Die Methoden stehen meistens als Verben im Use-Case-Dokument.
- Aber auch hier gilt: Nicht jedes Verb repräsentiert eine Methode.
- Welche Methoden sind das in unserem Beispiel?

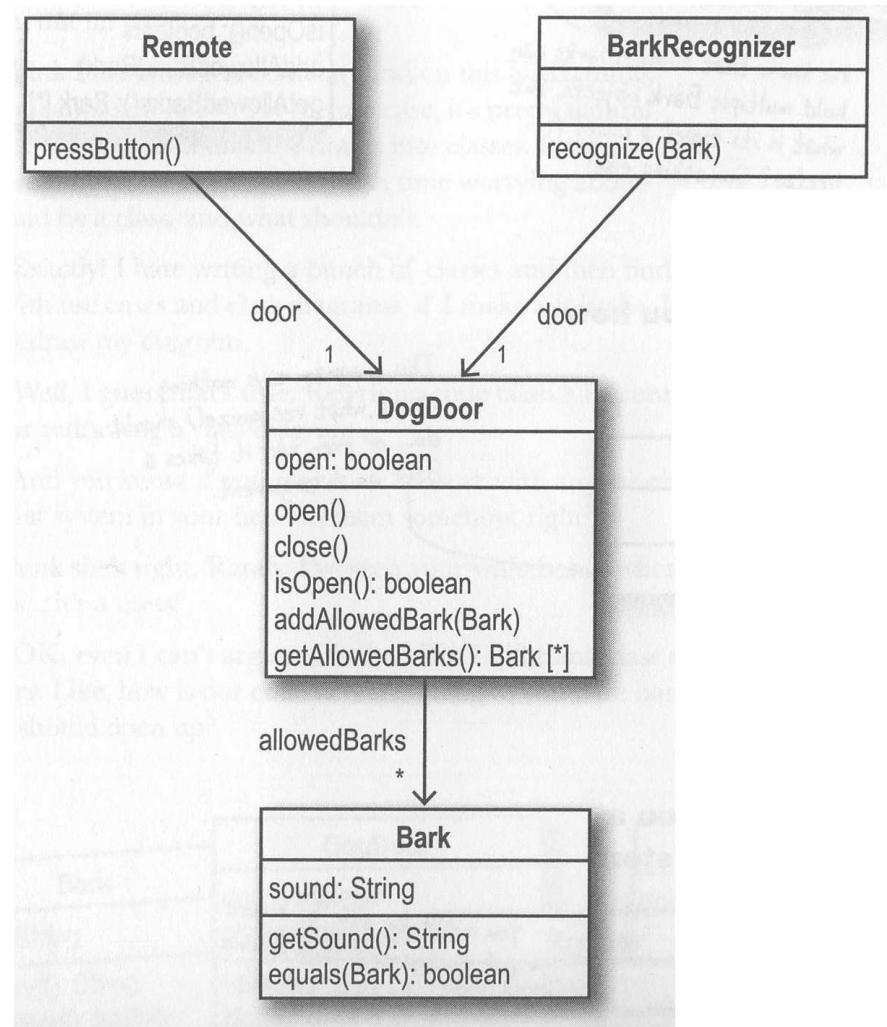


- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Ein Klassendiagramm

Was bedeuten die Pfeile?

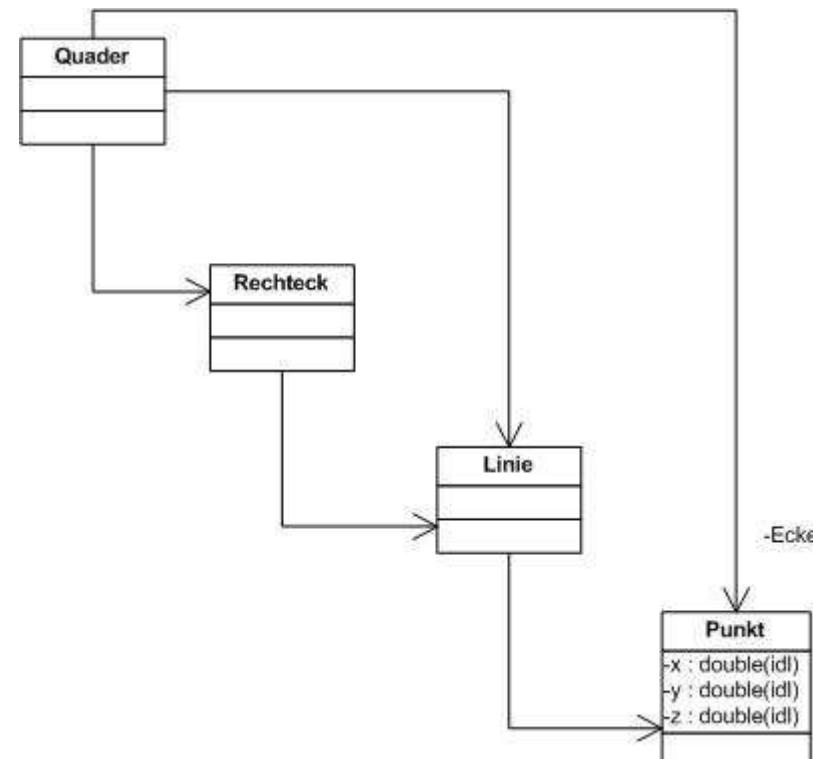
Was unterscheidet sie von Vererbungspfeilen?



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Übungsaufgabe

Ergänzen Sie das Klassendiagramm um Multiplizitäten.

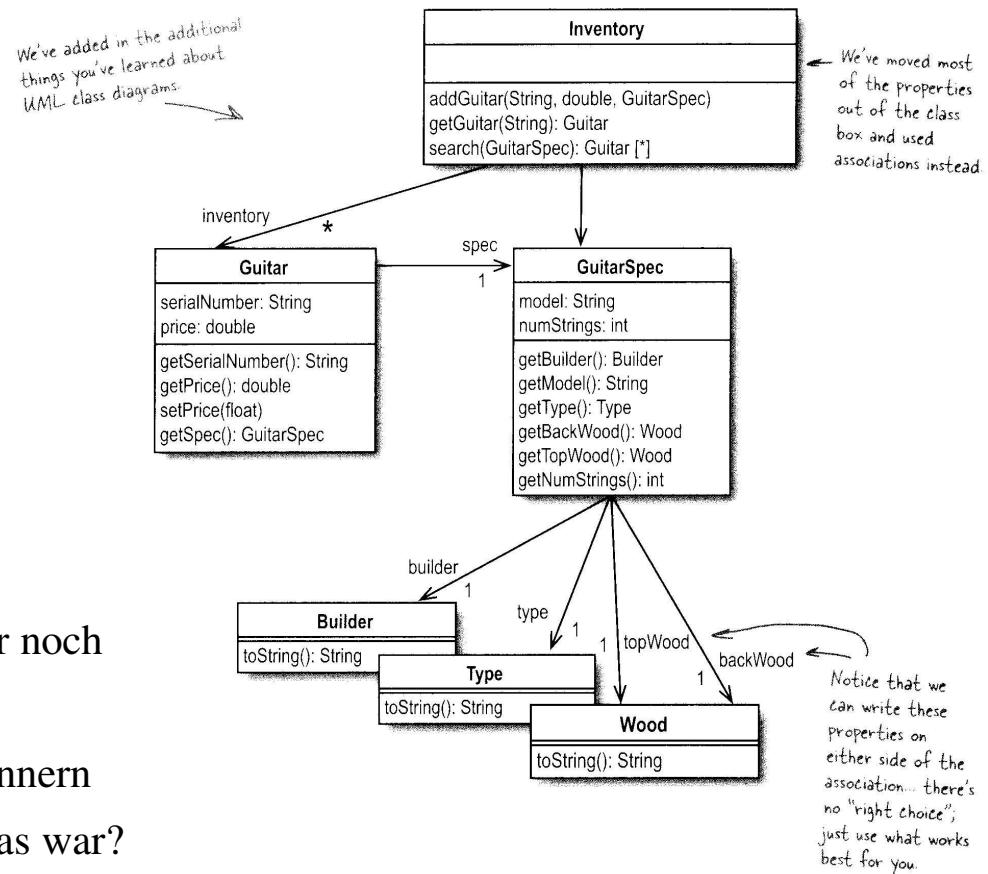




1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Zurück zum Beispiel Rick's Guitars

- Mit Assoziationen, Multiplizitäten und Rollen ergibt sich folgendes Klassendiagramm:



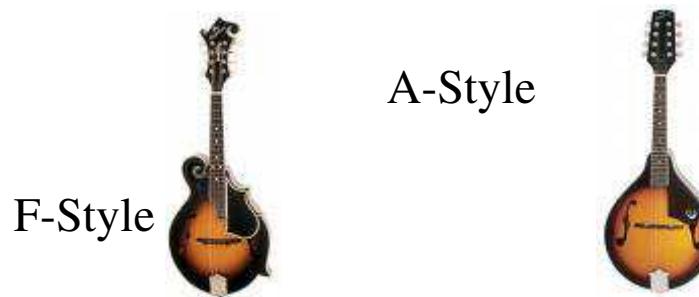
Übrigens fehlt hier noch unsere letzte Verbesserung. Erinnern Sie sich, welche das war?



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Zurück zum Beispiel Rick's Guitars

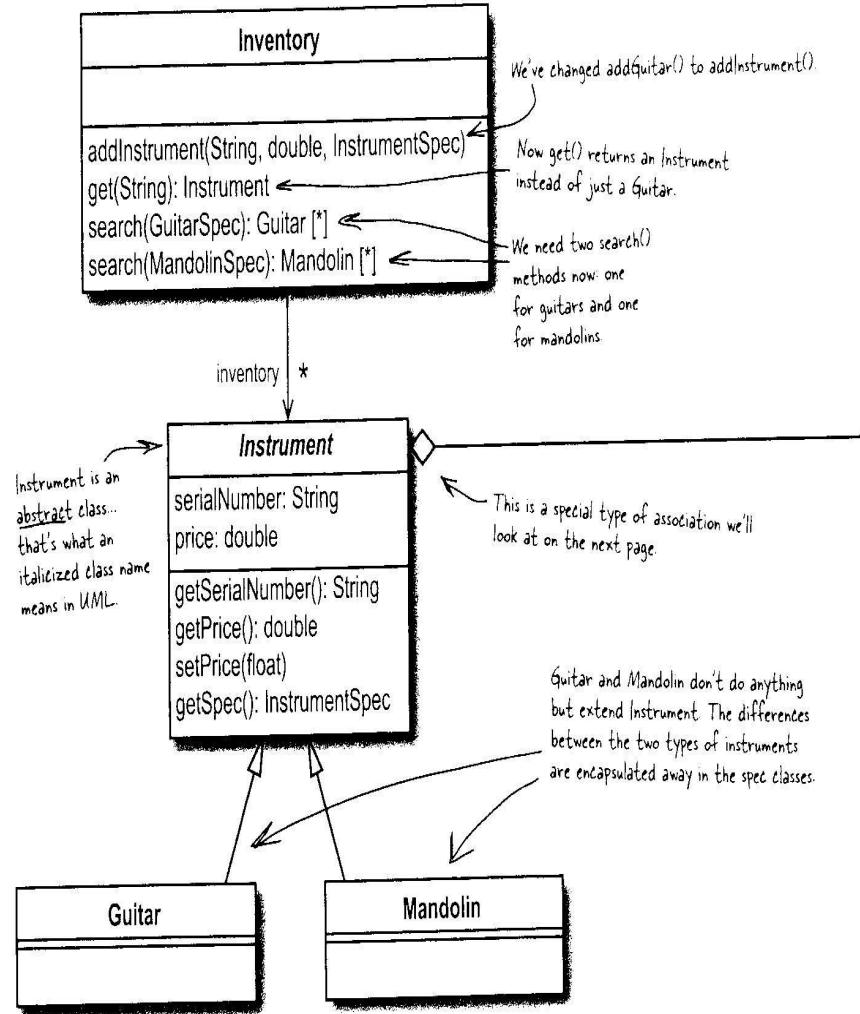
- Rick möchte zusätzlich Mandolinen verkaufen.
- Alle Mandolinen, die er verkauft, haben 8 Saiten.
- Es gibt unterschiedliche Stile von Mandolinen:



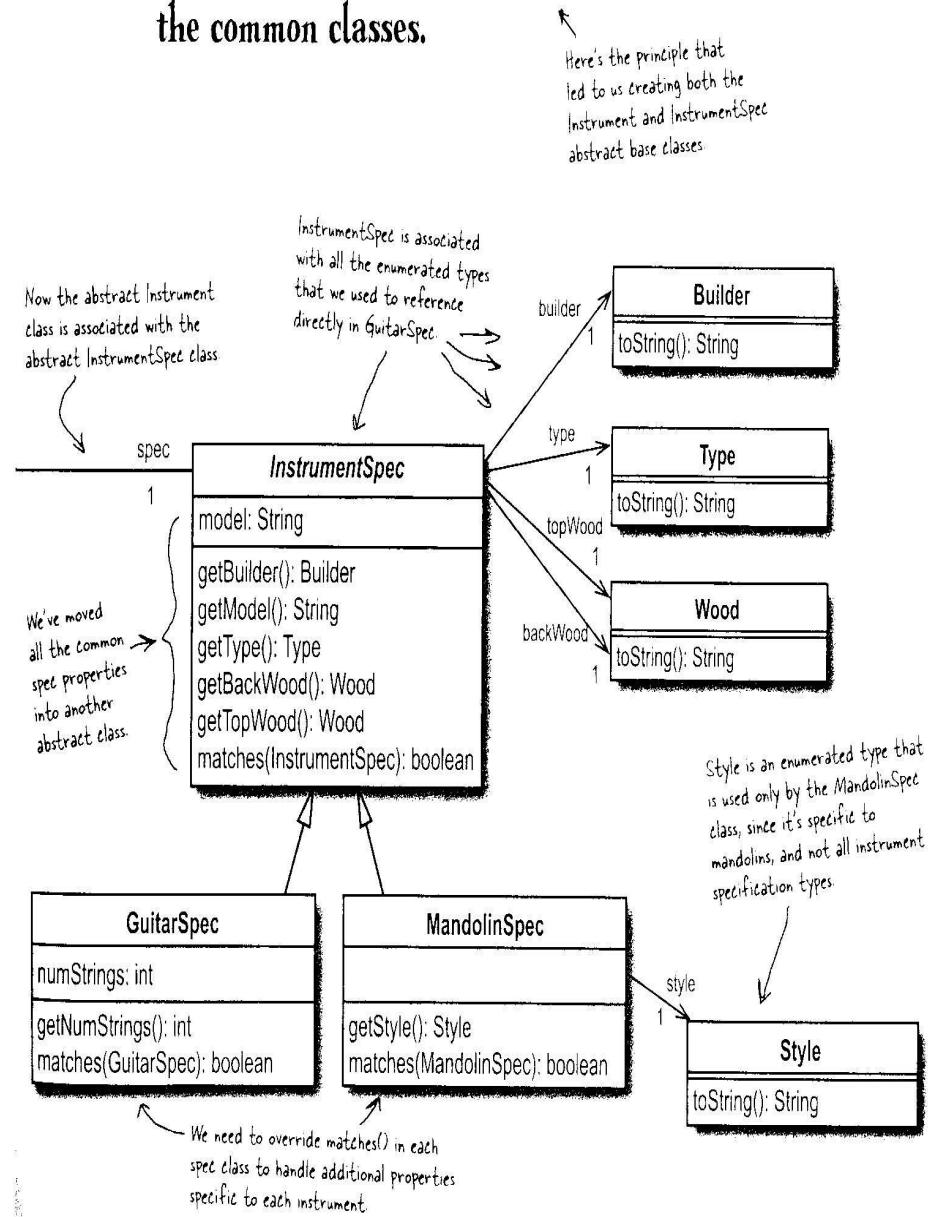
- Was ändert sich im Klassendiagramm?

Behold: Rick's new application

It looks like all that work on design back in Chapter 1 has paid off: it took us less than 10 pages to add support for mandolins to Rick's search tool. Here's the completed class diagram:



Whenever you find common behavior in two or more places, look to abstract that behavior into a class, and then reuse that behavior in the common classes.



- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

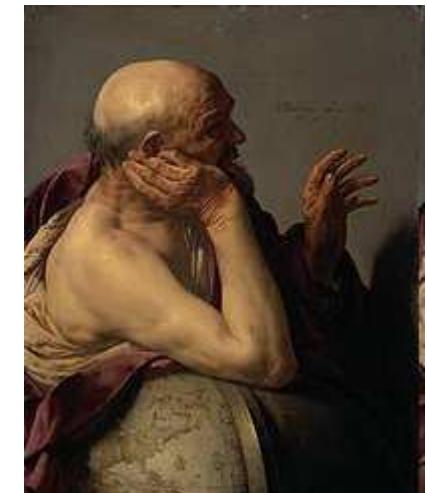
Beziehungen zwischen Klassen

UML Cheat Sheet		
<u>What we call it in Java</u>	<u>What we call it in UML</u>	<u>How we show it in UML</u>
Abstract Class	Abstract Class	<i>Italicized Class Name</i>
Relationship	Association	→
Inheritance	Generalization	→ ▷
Aggregation	Aggregation	→ ◇



- 1. Was ist Modellierung?**
- 2. OOAD mit der UML**
- 3. Entity-Relationship-Modelle**
- 4. Geschäftsprozeß-modellierung**

"The only constant is change"



Heraklitus von Ephesus

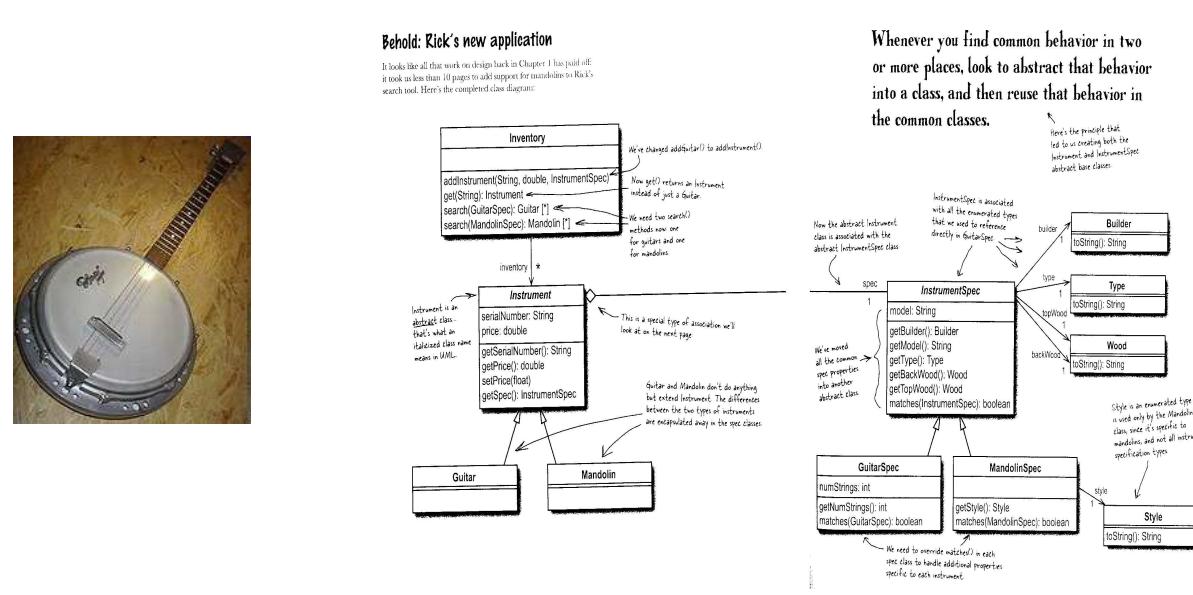
- Gutes Klassendesign zeichnet sich dadurch aus, daß...
 - ...es für jede Klasse so wenig wie möglich Gründe gibt, sich zu ändern.
(Im besten Fall nur einen einzigen, aber das ist nicht immer möglich)
 - ... von einer potentiellen Änderung so wenig wie möglich Klassen betroffen wären.
(Im besten Fall nur eine einzige, aber das ist nicht immer möglich)



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

"The only constant is change"

- Wie erweiterbar ist unsere Applikation jetzt?
- Was für Ergänzungen am Klassendiagramm wären z.B. nötig, um noch eine weitere Instrumentengattung (z.B. Banjos) hinzuzufügen?





1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Gutes Klassendesign ist auf Änderungen vorbereitet

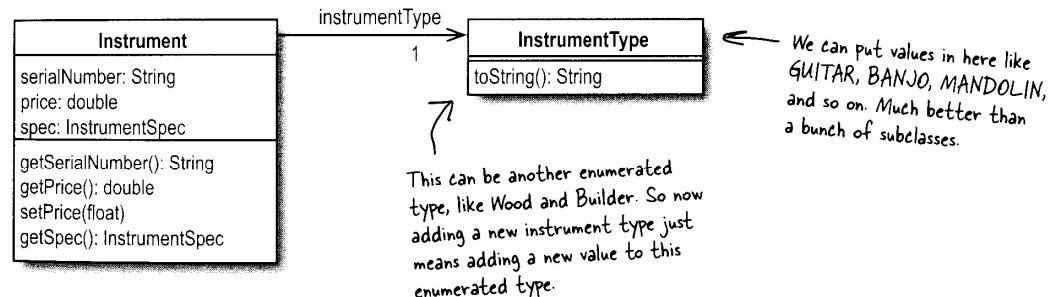
- Hauptproblem: Für jede neue Instrumentengattung müssen mindestens 2 neue Klassen und eine neue Suchmethode in der Klasse *Inventory* angelegt werden.
- Idee:
 - *GuitarSpec* und *MandolinSpec* haben zwar eigene Attribute, aber kein eigenes (d.h. unterklassenspezifisches) Verhalten. Dafür braucht man nicht unbedingt eine eigene Klasse.
 - Machen wir aus der abstrakten Klasse *InstrumentSpec* eine instanziierbare Klasse.
 - Damit können wir die Suchmethode in *Inventory* ganz allgemein halten: Sie bekommt ein Objekt vom Typ *InstrumentSpec* übergeben. Je nachdem, ob eine Gitarre oder eine Mandoline gesucht wird, ruft sie automatisch die richtige *matches*-Methode aus der entsprechenden Unterklasse auf. (Polymorphismus)



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Gutes Klassendesign ist auf Änderungen vorbereitet

- Es gibt nur noch eine *search*-Methode. Damit haben wir ein Problem gelöst, aber nicht alle.
- Immer noch müßten für jede neue Instrumentengattung zwei neue Klassen angelegt werden.
- Machen wir also aus der abstrakten Klasse *Instrument* jetzt auch noch eine instanzierbare Klasse. Die beiden Unterklassen *Guitar* und *Mandolin* bestanden ja ohnehin nur aus einem Konstruktor.
- Natürlich brauchen wir dann ein Attribut in der Klasse *Instrument*, das die Instrumentengattung enthält - am besten als Verweis auf eine Klasse *InstrumentType*, die Einträge wie Gitarre, Mandoline und Banjo enthalten kann.





1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Gutes Klassendesign ist auf Änderungen vorbereitet

- Da aber die Instrumentengattung (Gitarre, Mandoline etc.) das Instrument spezifiziert, gehört sie eigentlich nicht in *Instrument*, sondern in *InstrumentSpec*.
- Warum hatten wir nämlich die Klasse *InstrumentSprec* ursprünglich geschaffen?



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

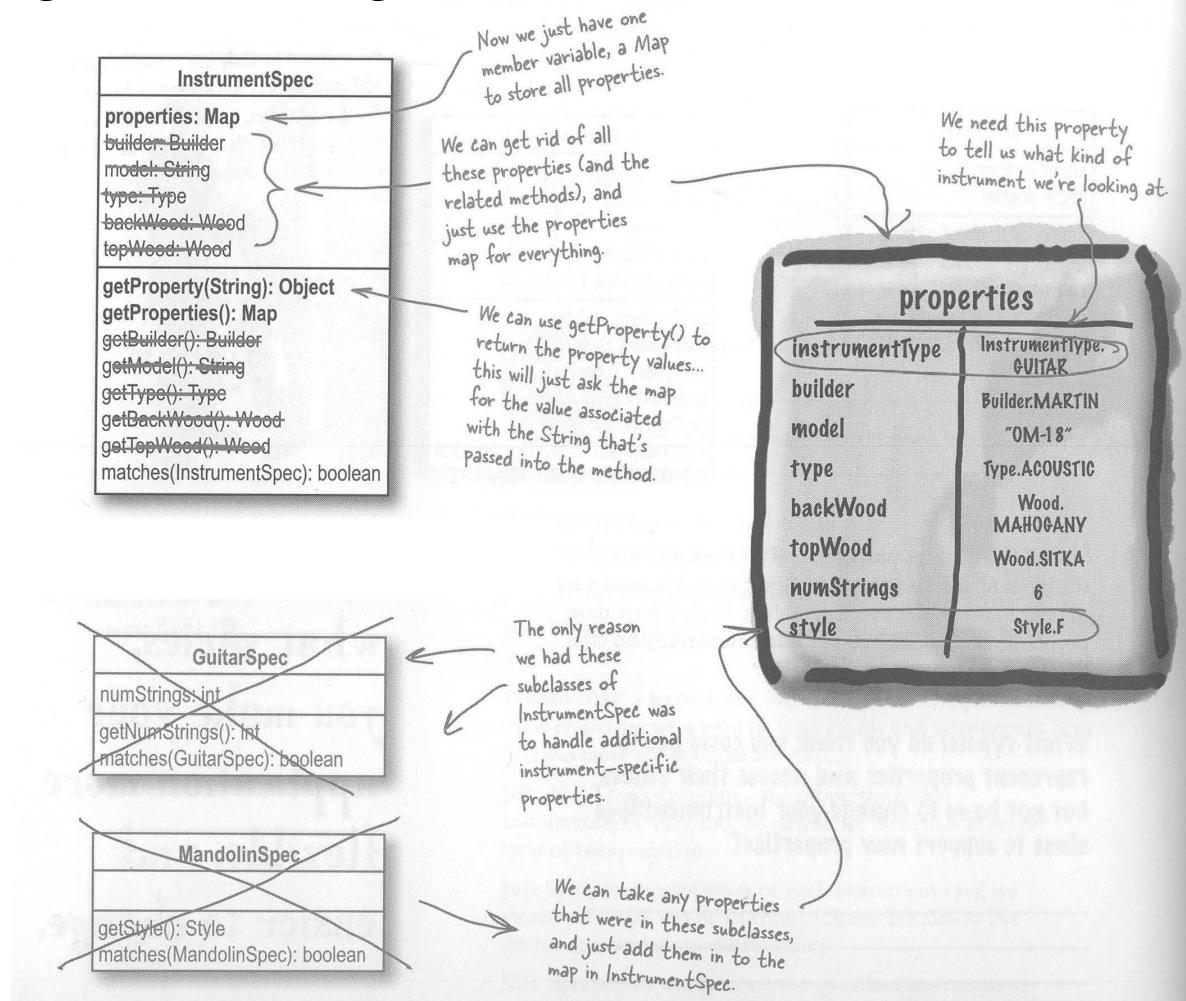
Gutes Klassendesign ist auf Änderungen vorbereitet

- *InstrumentSpec* kapselt die Eigenschaften des Instrumentes.
- Weil je nach Instrumentengattung diese Eigenschaften variieren, hatten wir Unterklassen für Gitarren und Mandolinen geschaffen.
- Das haben wir dann wieder verworfen, weil es zu uns zu unflexibel war.
- Wir brauchen also sozusagen noch eine zusätzliche Kapselung auf einer höheren Ebene.
- Wie könnten wir die Eigenschaften von Instrumenten in *InstrumentSpec* ohne Verwendung von Unterklassen verwalten, aber gleichzeitig berücksichtigen, daß von einem Attribut (nämlich der Instrumentengattung) die Anzahl und Bezeichnung der anderen Attribute abhängt?

1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

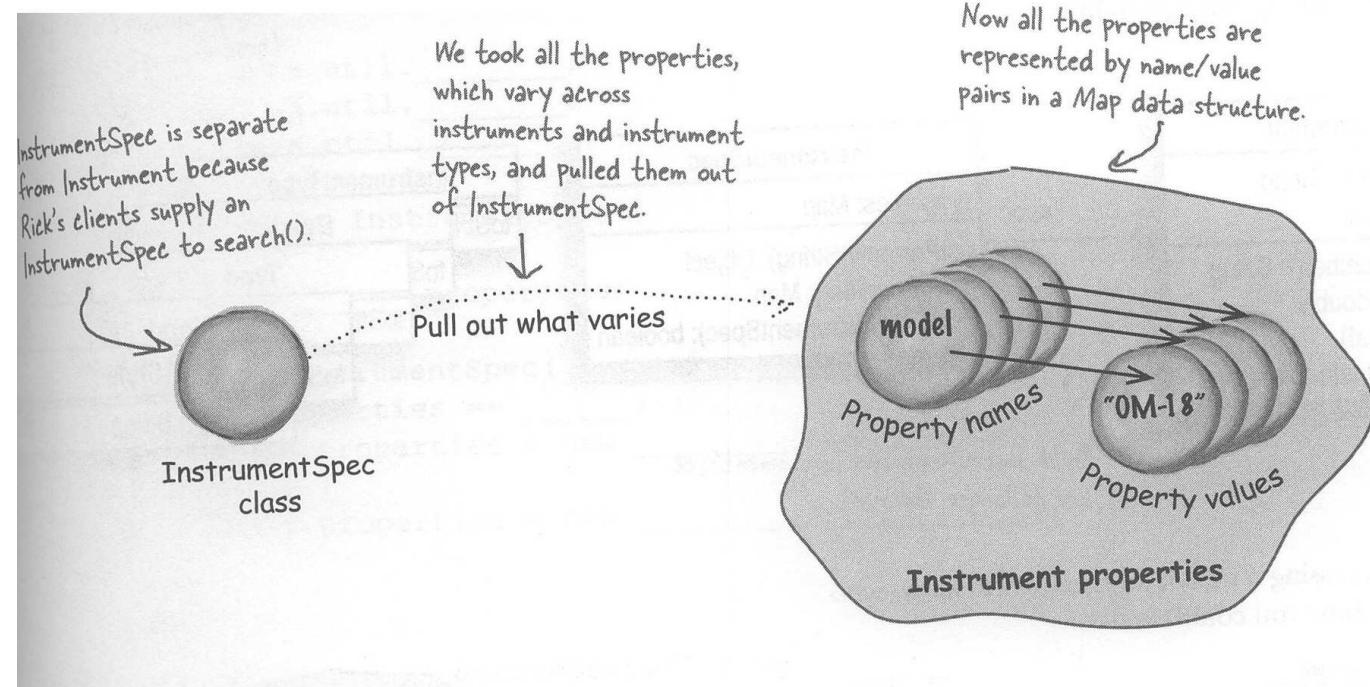
Gutes Klassendesign ist auf Änderungen vorbereitet

Eine mögliche Lösung:



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Kapselung macht flexibel





1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Fazit

- Was sich ändern kann, wird möglichst in nur eine einzige Klasse gekapselt.
- In diesem Fall wurde das Design flexibler, indem wir die Anzahl der Klassen reduziert haben. Es ist aber auch möglich, daß man Klassen hinzufügen muß, um flexibel zu werden.
- Ein guter Entwickler kann beides:
 - ein gutes, flexibles Klassendesign finden, das den Prinzipien der Objektorientierung folgt,
 - dieses Design mit korrekter UML-Syntax modellieren.

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Übungsaufgabe

- Betrachten Sie das Klassendiagramm auf Folie 102 (Seite 234 im Buch).
- Zeichnen Sie ein Klassendiagramm, das alle unsere Änderungen enthält.



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Zusatzfragen

- Wieviele Klassen müßten wir jetzt hinzufügen, falls Rick Lauten verkaufen will?
- Wieviele Klassen müßten wir jetzt ändern, falls Rick Lauten verkaufen will?
- Wieviele Klassen müßten wir jetzt ändern, falls Rick das Baujahr seiner Instrumente speichern will?
- Wieviele Klassen müßten wir jetzt hinzufügen, falls Rick das Material der Mechanikgriffe zum Stimmen des Instrumentes speichern will? (Z.B. Messing, Ebenholz, Kunststoff)



1995





1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Detaillierungsgrad von Klassendiagrammen

- Wir haben Klassendiagramme aus Use-Case-Dokumenten abgeleitet. Dabei haben wir ihre wichtigsten Notationselemente schon kennengelernt.
- Die UML kennt noch weitere Elemente. Wie immer gilt: Der Detaillierungsgrad hängt vom Zweck des Diagramms ab.
- Entwerfen Sie zunächst ein gutes (d.h. vor allem erweiterbares) Klassendesign, bevor Sie Details ausarbeiten.
- Klassendiagramme, die als Vorlage für Entwickler dienen, sollten alle relevanten Klassen, Attribute und Methoden enthalten. Das bedeutet aber nicht, daß sie auch alle Notationselemente enthalten müssen.

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Attribute in Klassendiagrammen

- Alles außer dem Namen des Attributes ist optional. Ob man die Symbole verwendet, hängt also wieder vom Einsatzzweck ab.
- Klassenattribute werden unterstrichen.
- Der Datentyp wird nach einem : angegeben. Das können außer den elementaren Datentypen (int, long etc.) natürlich auch andere Klassen des Diagramms sein.
- Defaultwerte werden nach einem = angegeben.

Schueler
<u>-Durchschnittsnote</u> : float(idl)
-Note : float(idl) = 5
-Matrikel : long(idl)



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Attribute in Klassendiagrammen

- Attribute, die nicht gespeichert, sondern berechnet werden (z.B. das Alter aus dem Geburtsdatum), werden mit / gekennzeichnet.
- Die Sichtbarkeit eines Attributes wird mit folgenden Symbolen festgelegt:
 - + public, für alle Klassen sichtbar
 - # protected, für eigene Erben sichtbar
 - private, für andere Klassen unsichtbar
 - ~ package, innerhalb des Paketes sichtbar
- Ist die Sichtbarkeit nicht angegeben, so ist sie undefiniert.



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Attribute in Klassendiagrammen

- Die Multiplizität kann bei einem Attribut z.B. in der Form [4], [0..6], [1..*] angegeben werden.
 - Attribute, die eine Assoziation implementieren, werden im Klassendiagramm aber meist weggelassen.
 - Die Multiplizität steht dann statt dessen an der Assoziation.
 - Attribute, denen keine Assoziation entspricht (mit elementaren Datentypen wie long oder char) sind wiederum selten mengenwertig.
 - Deshalb findet dieses Notationselement eher selten Verwendung.

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Attribute in Klassendiagrammen

- Am Ende der Attributzeile im Klassendiagramm kann dem Attribut in geschweiften Klammern eine Eigenschaft zugewiesen werden, z.B.
 - {readOnly} Der Attributwert kann nicht verändert werden.
 - {unique} Jeder Attributwert darf bei maximal einem Objekt vorkommen.



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Attribute in Klassendiagrammen

- Die Reihenfolge aller Notationselemente für Attribute lautet:

Sichtbarkeit / Name :Typ Multiplizität =Defaultwert {Eigenschaft}



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Übungsaufgaben

- 1) Erstellen Sie einen Auszug aus einem Klassendiagramm für die Attribute der Klasse Kunde.
Verwenden Sie dabei möglichst viele Notationselemente, indem Sie sinnvolle Attribute dafür finden. Methoden brauchen Sie nicht zu modellieren.
(Übrigens wäre das in der Praxis natürlich eine schlechte Aufgabenstellung. Man verwendet nicht möglichst viele, sondern möglichst sinnvolle Elemente)
- 2) Wie implementiert man die folgenden Notationselemente der UML in Java:
`+, #, -, ~, /,`
`attribut,`
`=default,`
`{readOnly},`
`Klassename`



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Methoden in Klassendiagrammen

- Alles außer dem Namen der Methode, gefolgt von () ist optional. Welche Symbole man verwendet, hängt also auch hier wieder vom Einsatzzweck ab. Als Vorlage für die Entwickler sollte das Diagramm möglichst wenig Interpretationen zulassen.
- Statische Methoden (Klassenmethoden) werden unterstrichen.
- Die Sichtbarkeit wird wie bei Attributen angegeben.

Schueler
<u>-Durchschnittsnote</u> : float(idl)
-Note : float(idl) = 5
-Klassenbuchnummer : long(idl)
+getDurchschnittsnote() : float(idl)
+getNote() : float(idl)
+setNote(in Note : float(idl))



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Methoden in Klassendiagrammen

- Die Reihenfolge aller Notationselemente für Methoden lautet:

Sichtbarkeit Name (Parameterliste) :Rückgabetyp {Eigenschaft}



Parameter müssen mindestens Name und Typ enthalten und werden mit Kommata voneinander getrennt. Ein Parameter wird angegeben mit:

Übergabeart Name :Typ Multiplizität =Defaultwert {Eigenschaft}



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Methoden in Klassendiagrammen

- Mögliche Übergabearten sind *in*, *out* und *inout*.
- Da Java die explizite Unterscheidung zwischen call-by-value und call-by-reference nicht kennt, wird man i.d.R. wie folgt modellieren, wenn Java Zielsprache ist:
 - Alle Parameter mit elementaren Datentypen (char, int etc.) werden als in-Parameter modelliert.
 - Parameter mit Klassen als Typ bekommen Objekte übergeben. Sie werden als inout-Parameter modelliert.
 - Out-Parameter werden nicht modelliert. Die Methode gibt einen Wert zurück, der mit dem Rückgabetyp spezifiziert wird.



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Methoden in Klassendiagrammen

- Die Multiplizität in der Parameterliste definiert, aus wie vielen Teilen ein Parameter bestehen darf. Sie wird wie bei Attributen angegeben.
- Wird für einen Parameter ein Defaultwert angegeben, so ist er beim Aufruf optional. Fehlt er, wird der Defaultwert verwendet.
- Abstrakte Klassen (kursiver Klassename) können abstrakte Methoden (kursiver Methodename) enthalten. Jede nicht-abstrakte erbende Klasse *muß* alle geerbten abstrakten Methoden selbst implementieren.



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Methoden in Klassendiagrammen

- Als Eigenschaften der Parameter und des Rückgabetyps sind z.B. möglich:
 - {sequence}
Tupel: Die Werte sind geordnet.
 - {bag}
Menge: Die Werte sind ungeordnet. Eine Reihenfolge ist nicht definiert.
- Überladene Methoden werden mehrfach im Klassendiagramm aufgeführt. Die Einträge haben dann denselben Namen, aber unterschiedliche Parameterlisten.



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Übungsaufgabe

- 1) Erstellen Sie einen Auszug aus einem Klassendiagramm für die Methoden der Klasse Kunde.
Verwenden Sie dabei möglichst viele Notationselemente, indem Sie sinnvolle Methoden dafür finden.
- 2) Wie implementiert man die folgenden Methoden der Klasse Schueler aus der UML in Java:
+getDurchschnittsnote():float
+setNote(in note:float)
+getNebensitzer():Schueler [0..2]



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Assoziationen in Klassendiagrammen

- Eine Assoziation kann mit einem Namen beschriftet werden. Ein Dreieck kann die Leserichtung angeben.
- Die Multiplizität ist immer zu einem Zeitpunkt zu verstehen. (Ein Mitarbeiter kann im Lauf der Unternehmenszugehörigkeit durchaus zu mehreren Abteilungen gehören, aber nicht gleichzeitig)

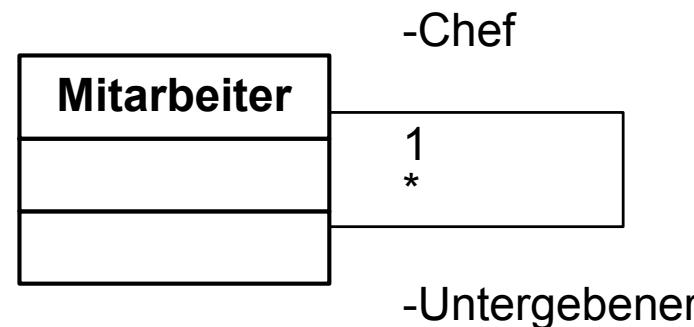




1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Assoziationen in Klassendiagrammen

- Rollen sind bei reflexiven Assoziationen besonders wichtig. Für sie kann die Sichtbarkeit definiert werden wie für Attribute.

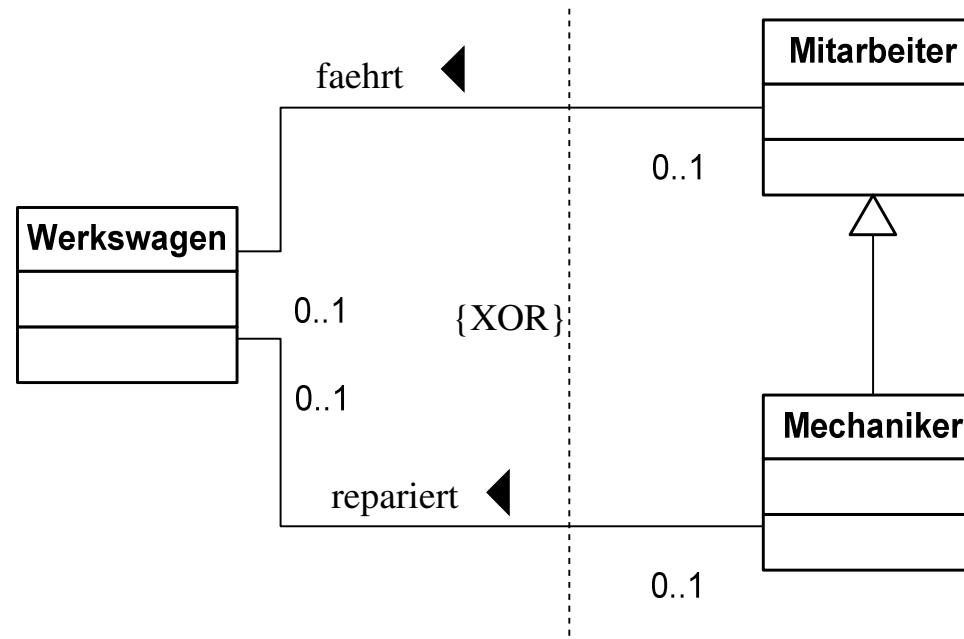


- Übungsaufgabe: Implementieren Sie die Attribute und den Konstruktor der Klasse **Mitarbeiter** in Java.

1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Assoziationen in Klassendiagrammen

- Eine XOR-Einschränkung besagt, daß Objekte der beteiligten Klassen nur an einer der Assoziationen beteiligt sein dürfen.

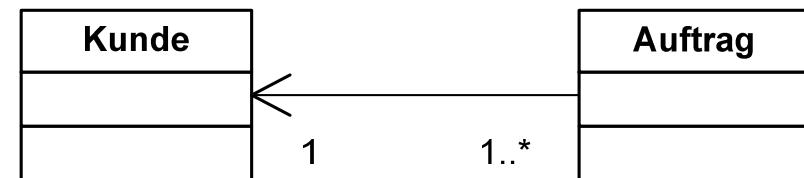




1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Assoziationen in Klassendiagrammen

- Assoziationen können implementiert werden, indem Objekte einer Klasse ein Attribut bekommen, das auf ein Objekt der anderen Klasse verweist. Die Navigierbarkeit einer Assoziation gibt an, in welcher Richtung die Klassen auf diese Art Kenntnis voneinander haben.
- Assoziationen mit Pfeil schreiben die Navigierbarkeit vor: Jedes Auftragsobjekt hat Kenntnis des Kunden, der den Auftrag abgegeben hat.



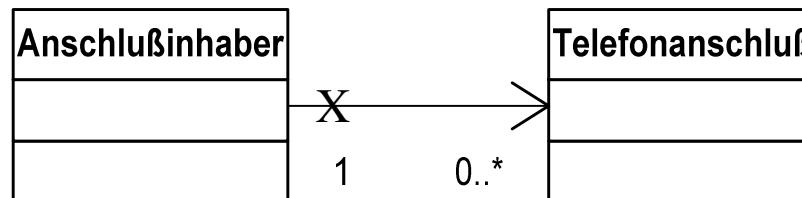
- Mit welchem Datentyp könnte man dies implementieren, wenn die Assoziation an beiden Enden Pfeilspitzen hätte?



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Assoziationen in Klassendiagrammen

- Assoziationen mit einem x verbieten die Navigierbarkeit. Das folgende Diagramm schließt die Rückwärtssuche in einem Telefonverzeichnis aus.



- Die UML erlaubt sogar ein beiderseitiges Navigationsverbot. Streng genommen gibt es dann gar keine Assoziation mehr, zumindest ist sie nicht implementierbar.

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Assoziationen in Klassendiagrammen

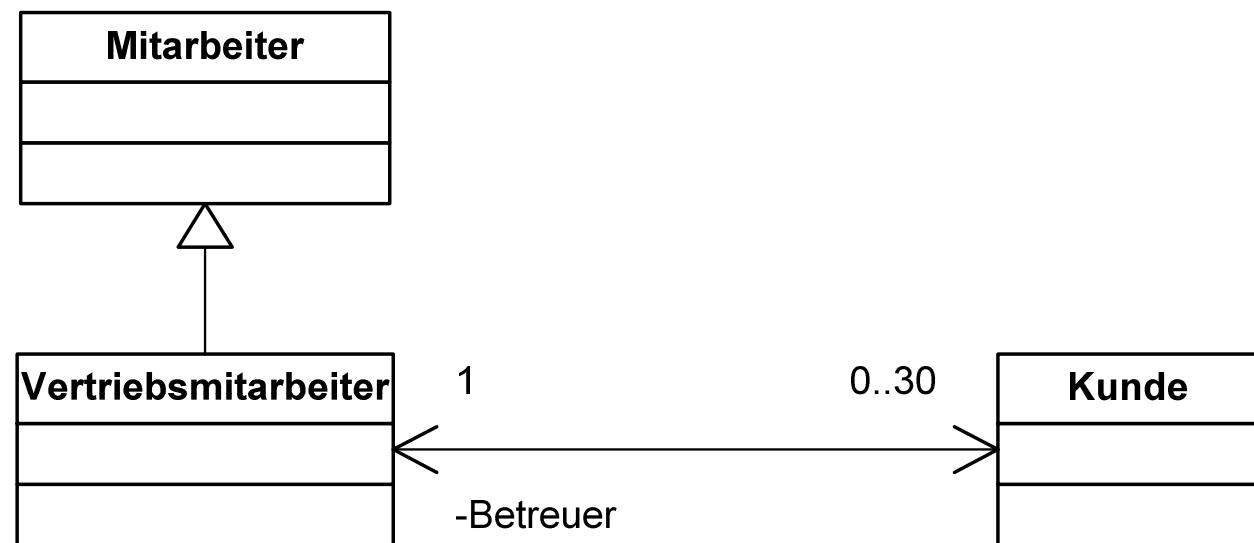
- Assoziationsenden ohne Pfeil und ohne x spezifizieren die Navigierbarkeit gar nicht. Sie ist dann weder verboten noch vorgeschrieben; die Entscheidung obliegt dem Implementierer.
- An beliebigen Stellen im Diagramm können Anmerkungen mit folgendem Symbol eingefügt werden:

Dies ist meine Lieblingsklasse.
Euer Modellierer.

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeßmodellierung

Übungsaufgabe

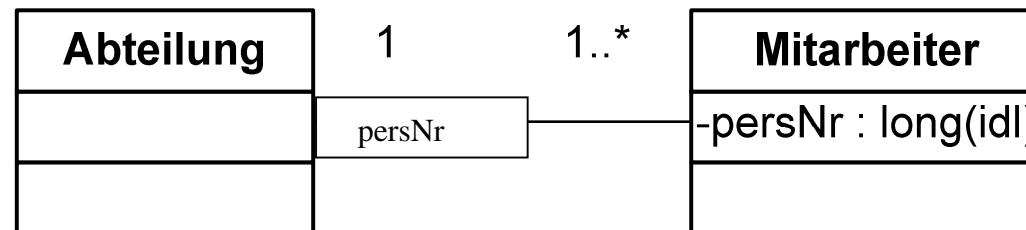
- 1) Wie implementiert man die folgende Assoziation aus der UML in Java? (Attribute und Konstruktoren der beiden beteiligten Klassen)



- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Assoziationen in Klassendiagrammen

- Qualifizierer geben an, anhand welcher Attribute einem Objekt der Klasse ein Objekt der assoziierten Klasse zugeordnet wird.

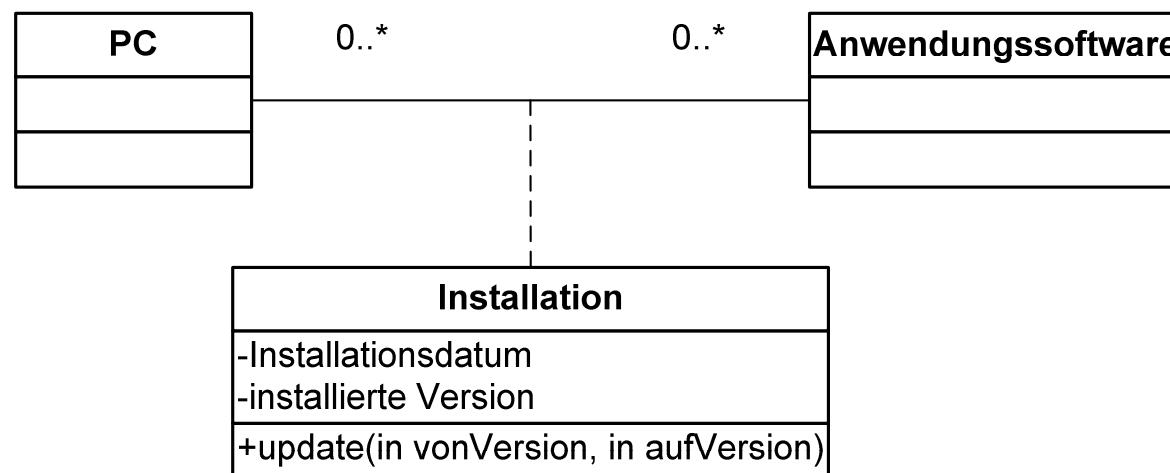




1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Assoziationen in Klassendiagrammen

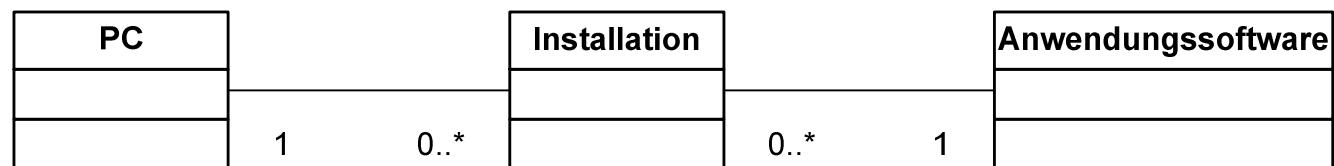
- Assoziationsklassen erlauben die Modellierung von Eigenschaften einer Assoziation.
- Bei m:n-Assoziationen (d.h. auf beiden Seiten ist die Obergrenze der Multiplizität >1 oder *) gibt es oft Attribute und Methoden, die zu keiner der Klassen, sondern zur Assoziation gehören. Sie werden dann der Assoziationsklasse zugeordnet.



- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Assoziationen in Klassendiagrammen

- Da gängige Programmiersprachen Assoziationsklassen nicht direkt implementieren können, ist eine Umwandlung in normale Klassen nötig.
- Häufig wird dies bereits bei der Modellierung berücksichtigt und auf die Verwendung von Assoziationsklassen verzichtet.



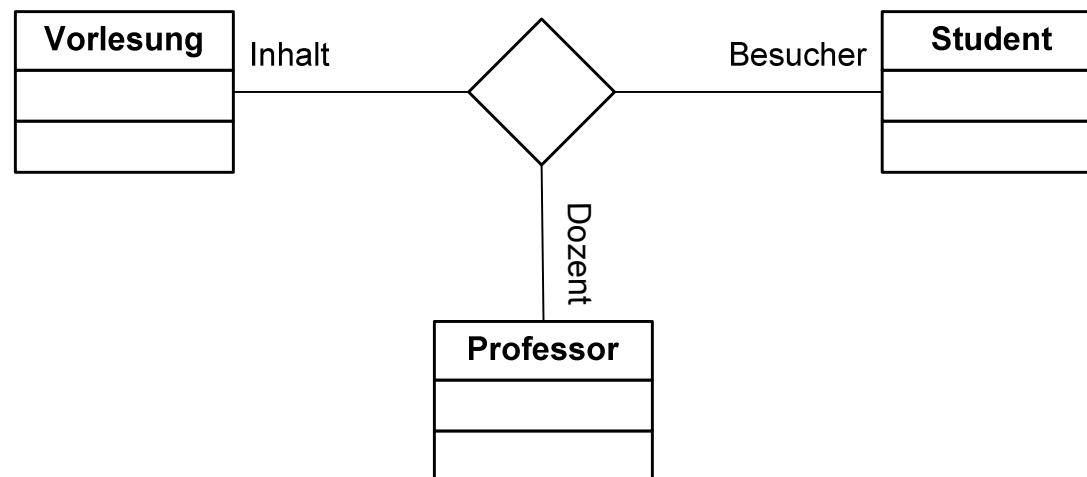
- Wie würden sich die Multiplizitäten in beiden Modellen ändern, wenn die Klasse Anwendungssoftware nicht ein Softwareprodukt (z.B. "MS Office"), sondern dessen Instanz (z.B. MS Office Product Key xxxxxxxxxxxx) enthielte?



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

n-äre Assoziationen in Klassendiagrammen

- Assoziationen können auch Beziehungen zwischen mehr als zwei Klassen beschreiben.
- Sie werden dann als Raute gezeichnet und können fast alle Notationselemente binärer Assoziationen enthalten.
 - Ausnahmen: Die Navigierbarkeit wird nicht angegeben. Qualifizierer werden nicht verwendet.

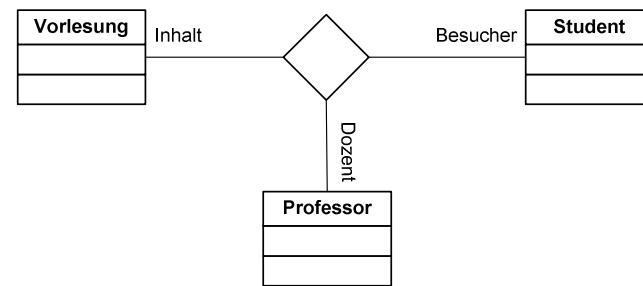




1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

n-äre Assoziationen in Klassendiagrammen

- Multiplizitäten bei einer Klasse beziehen sich auf die Kombinationen von Objekten aller anderen Klassen.
- Multiplizitäten mit einer anderen Untergrenze als 0 sind deshalb meist nicht sinnvoll.
- Eine Multiplizität von [1..30] bei Student würde z.B. bedeuten, daß es für jede Kombination aus Professor und Vorlesung mindestens einen Studenten geben muß. (also auch für die Kombination aus Informatikprofessor und Philosophievorlesung)

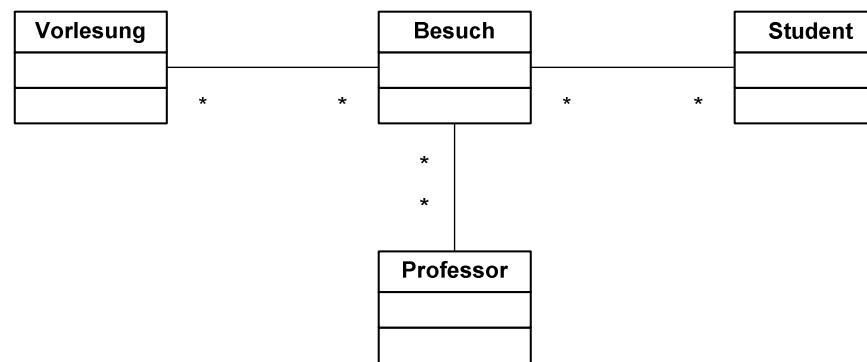




1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

n-äre Assoziationen in Klassendiagrammen

- n-äre Assoziationen werden eher selten verwendet, obwohl sie in der Realität durchaus vorkommen, weil...
 - sie die Komplexität der Realität nicht so sehr reduzieren wie binäre Assoziationen. (...und das schließlich der eigentliche Sinn von Modellen ist.)
 - sie in gängigen Programmiersprachen nicht direkt umsetzbar sind.
 - sie in mehrere binäre Assoziationen aufgelöst werden können.

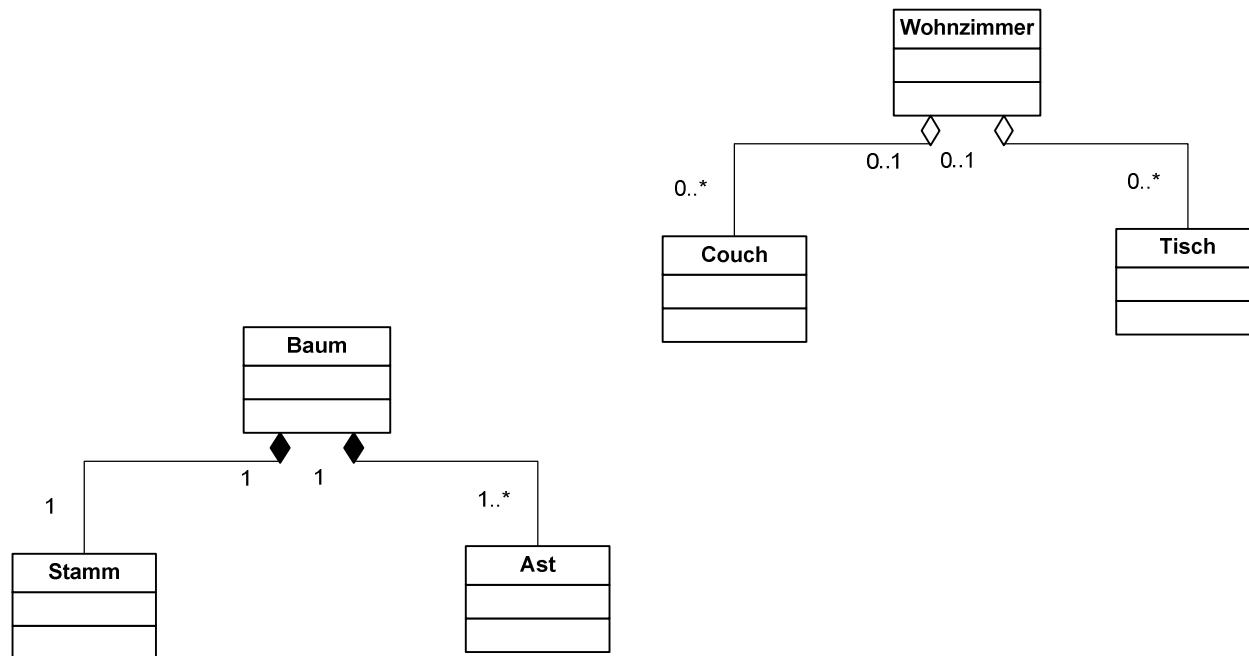




1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Aggregation und Komposition in Klassendiagrammen

- Wiederholung: Was ist der wesentliche Unterschied zwischen Aggregation und Komposition?





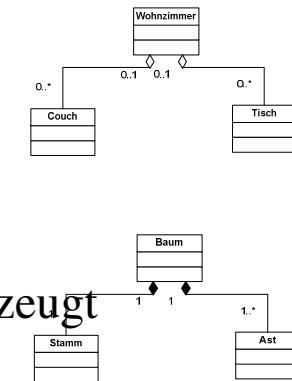
1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Aggregation und Komposition in Klassendiagrammen

- Multiplizitäten geben bei Aggregation und Komposition die Anzahl der Teile in einem Ganzen an. Bei der Komposition ist die Multiplizität auf Seite des Ganzen immer [1] und braucht deshalb nicht angegeben werden.
- Implementierung:
 - Aggregation: Das Ganze und seine Teile sind voneinander unabhängig. Die Klasse des Ganzen erzeugt ihre Teile nicht selbst. Sie können auch entfernt werden.

```
public void addCouch (Couch c) { . . . }  
public Couch removeCouch () { . . . }
```
 - Komposition: Das Ganze und seine Teile sind voneinander abhängig. Die Klasse des Ganzen erzeugt ihre Teile selbst.

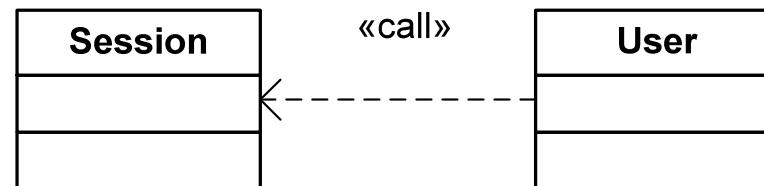
```
public void addAst ()  
{Ast a = new Ast(...); . . . }
```



- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Abhängigkeiten in Klassendiagrammen

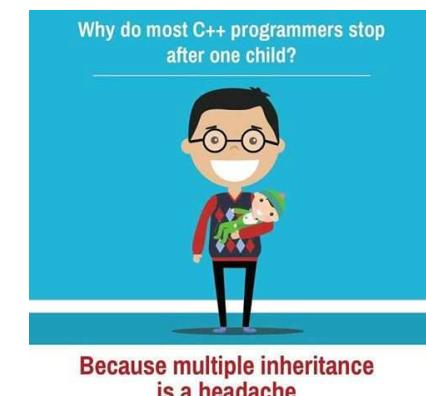
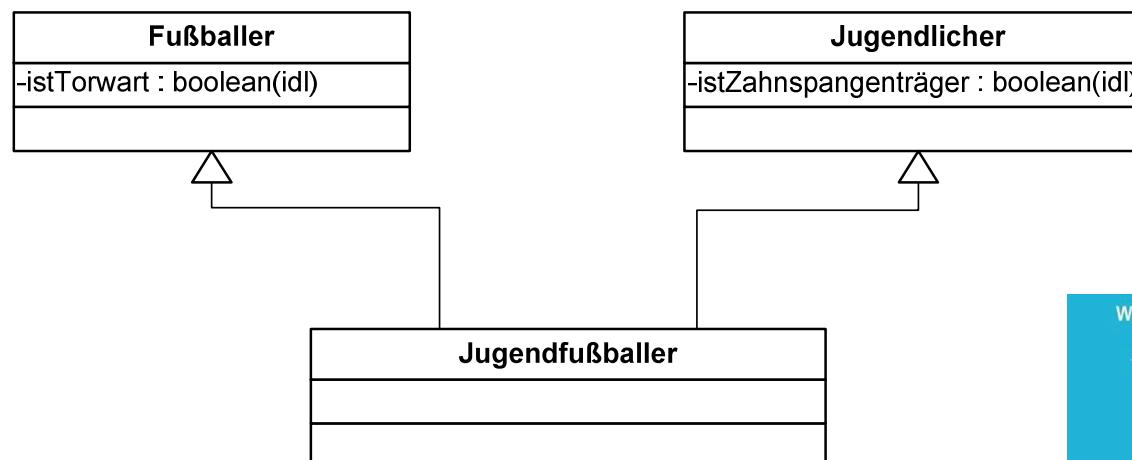
- Es kann viele Gründe geben, warum eine Klasse nicht ohne eine andere Klasse implementiert werden kann.
- Diese Abhängigkeit wird mit einem gestrichelten Pfeil beschrieben.
- Er wird mit einem Stereotypen beschriftet, der den Grund der Anhängigkeit benennt. Die wichtigsten sind:
 - <<call>>: Aufruf einer Methode
 - <<create>>: Erzeugung von Instanzen
 - Bsp.: Der Benutzer einer Webapplikation ruft Methoden eines Session-Objektes auf.



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Generalisierung in Klassendiagrammen

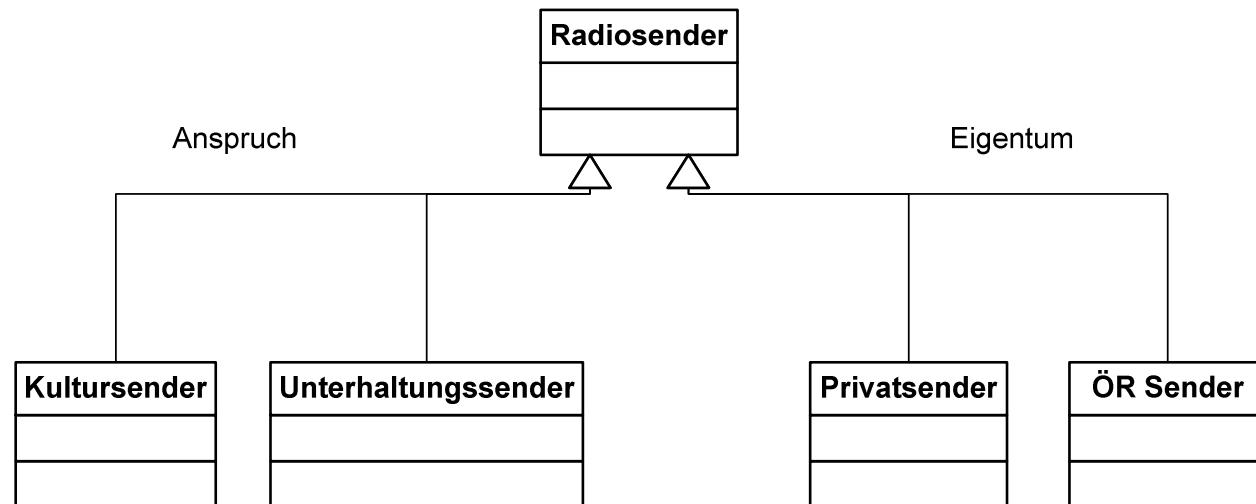
- Die UML erlaubt Mehrfachvererbung zwischen Klassen.
 - Bsp.: Jugendfußballer erbt von Fußballer und von Jugendlicher
 - Ist die Zielsprache Java, so kann dieses Konzept nicht unmittelbar implementiert werden.



- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Generalisierung in Klassendiagrammen

- Vererbungsbeziehungen können zu Generalisierungsgruppen zusammengefaßt werden.

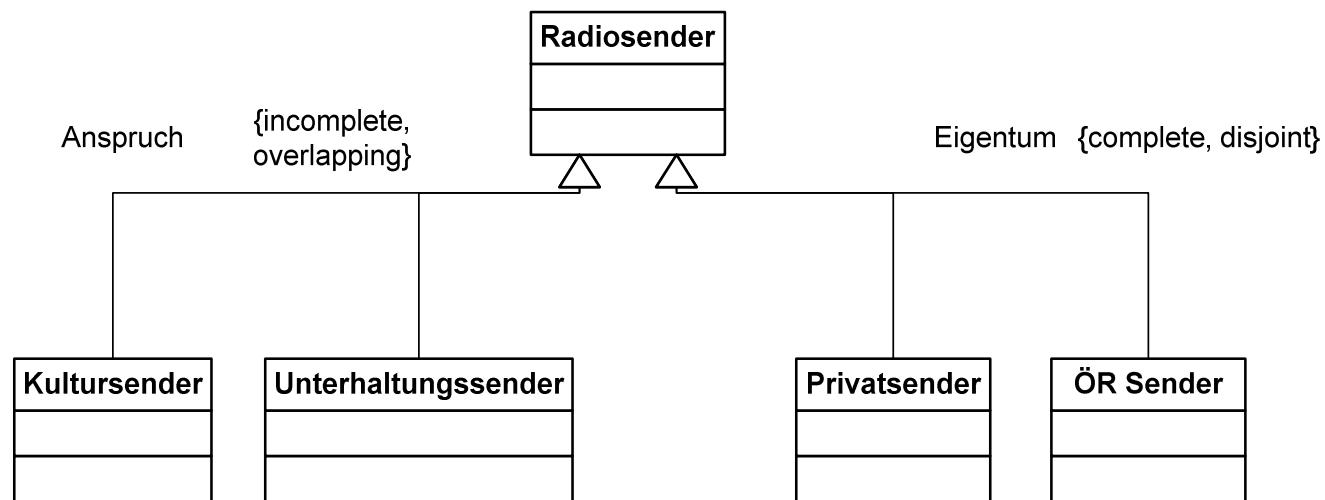




1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Generalisierung in Klassendiagrammen

- Für jede Generalisierungsgruppe können Eigenschaften festlegen,
 - ob die Unterklassen zusammen alle Objekte der Oberklasse umfassen (complete) oder nicht (incomplete)
 - und ob die Unterklassen sich überlappen (overlapping) oder nicht (disjoint).





1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Generalisierung in Klassendiagrammen

- Wie allen UML-Notationselemente kann man auch Klassen mit Stereotypen beschreiben. Stereotypen modellieren nicht wie andere Notationselemente die Realität, sondern beschreiben die anderen Notationselemente.
- Am häufigsten wird wohl <<enumeration>> für Aufzählungstypen verwendet. (Wie im Beispiel der Deckenhölzer für Rick's Gitarren)
- <<utility>>-Klassen stellen statische Attribute und Methoden für andere Klassen bereit. Sie sind üblicherweise abstrakt.

«utility»
MWSt
+vollerMehrwertsteuersatz : float(idl) = 0.19
+ermaessigterMehrwertsteuersatz : float(idl) = 0.7

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Templates in Klassendiagrammen

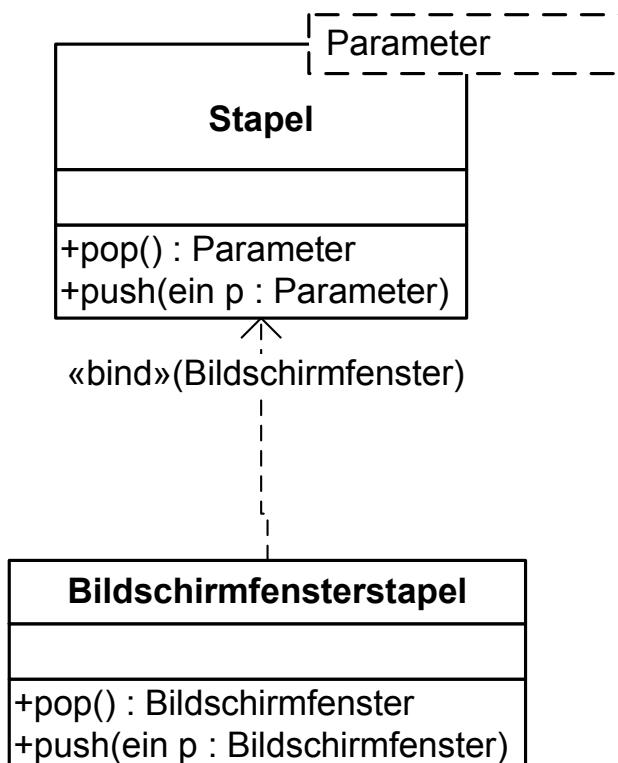
- Ein Template ist eine Vorlage für eine Klasse.
- In Abhängigkeit von einem Parameter wird die Klasse realisiert.
- Der Typ des Parameters ist beliebig. Wird kein Typ angegeben, so kann eine beliebige Klasse als Parameter übergeben werden.
- Templates werden verwendet, um Datenstrukturen für beliebige Klassen zu definieren, z.B. Stapel, Listen, Mengen.



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Templates in Klassendiagrammen

- Erst beim Binden wird mit der Übergabe des Parameters eine Klasse aus dem Template.
- Dies modelliert man mittels gestricheltem Pfeil.





1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Schnittstellen in Klassendiagrammen

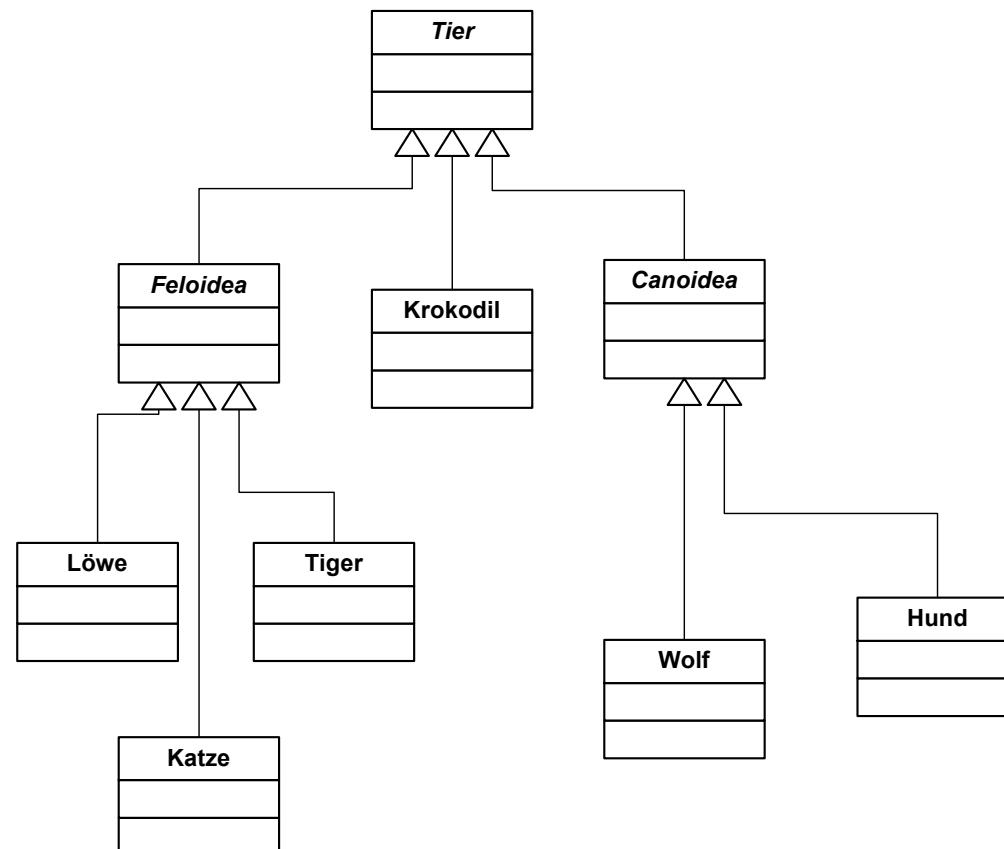
- Eine Schnittstelle ist eine Vorschrift, welche Attribute und Methoden eine Klasse zu implementieren hat.
- Sie wird mit einem Klassensymbol modelliert, das als <<interface>> gekennzeichnet ist.
- Schnittstellen ähneln abstrakten Klassen, implementieren aber keinerlei Methoden.
- Sie schreiben lediglich vor, welche Methoden von jeder Klasse zu implementieren sind, die von sich behauptet, diese Schnittstelle zu implementieren.
- Jede andere Klasse kann sich dann darauf verlassen, diese aufrufen zu können.

«Schnittstelle»
Haustier
+streicheln()

1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Schnittstellen in Klassendiagrammen

- Gegeben die folgende Vererbungshierarchie eines Zoos. Wo sollen die Methoden für Haustiere (Hund, Katze) implementiert werden?

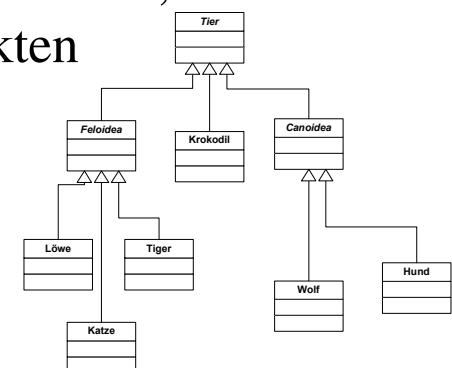




1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Schnittstellen in Klassendiagrammen

- Option 1: In der Klasse Tier
Nachteil: Löwen und Wölfe bekommen die Methoden, die Haustieren vorbehalten sein sollten, z.B.
`public void verkaufenAnPrivatmann(Kunde kaeufer)`
- Option 1a: In der Klasse Tier als abstrakte Klassen
Nachteil: Für Löwen und Wölfe müssen unnötigerweise eigene Haustiermethoden implementiert werden. (Vermutlich wird der Entwickler dann nur Dummymethoden schreiben, um pro forma die Anforderung der abstrakten Methoden zu erfüllen.)

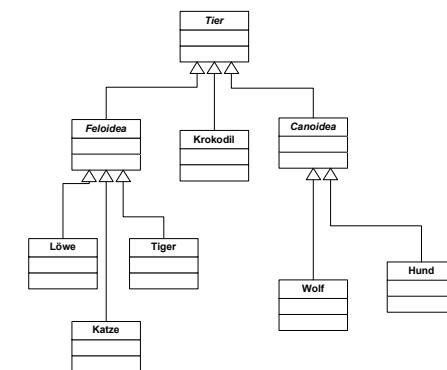




1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Schnittstellen in Klassendiagrammen

- Option 2: In den Klassen Hund und Katze
Nachteil: Die Entwickler müssen außerhalb des Systems festlegen, wie die Methoden in den Haustierklassen benannt werden sowie welche Parameter sie bekommen und zurückgeben, damit das einheitlich gehandhabt wird. Außerdem kann man die Methoden nicht polymorph für die Oberklasse Tier anwenden.





1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Schnittstellen in Klassendiagrammen

- Da Mehrfachvererbung andere Nachteile hat, bleiben nur Schnittstellen als Lösung.
- Die Klassen Hund und Katze implementieren die Schnittstelle Haustier.
- Andere Klassen, die die Methoden von Haustier aufrufen wollen, können sich darauf verlassen: Hund und Katze müssen das implementiert haben, sonst würden sie der Schnittstelle nicht genügen.

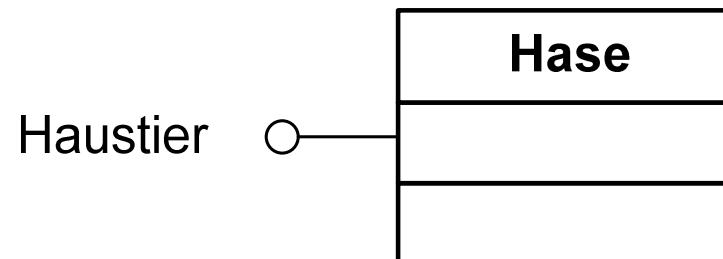
«Schnittstelle»
Haustier
+streichen()



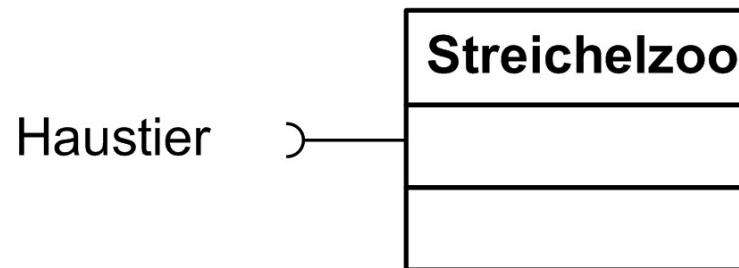
1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Schnittstellen in Klassendiagrammen

- Eine realisierte Schnittstelle wird als Ball-End modelliert.



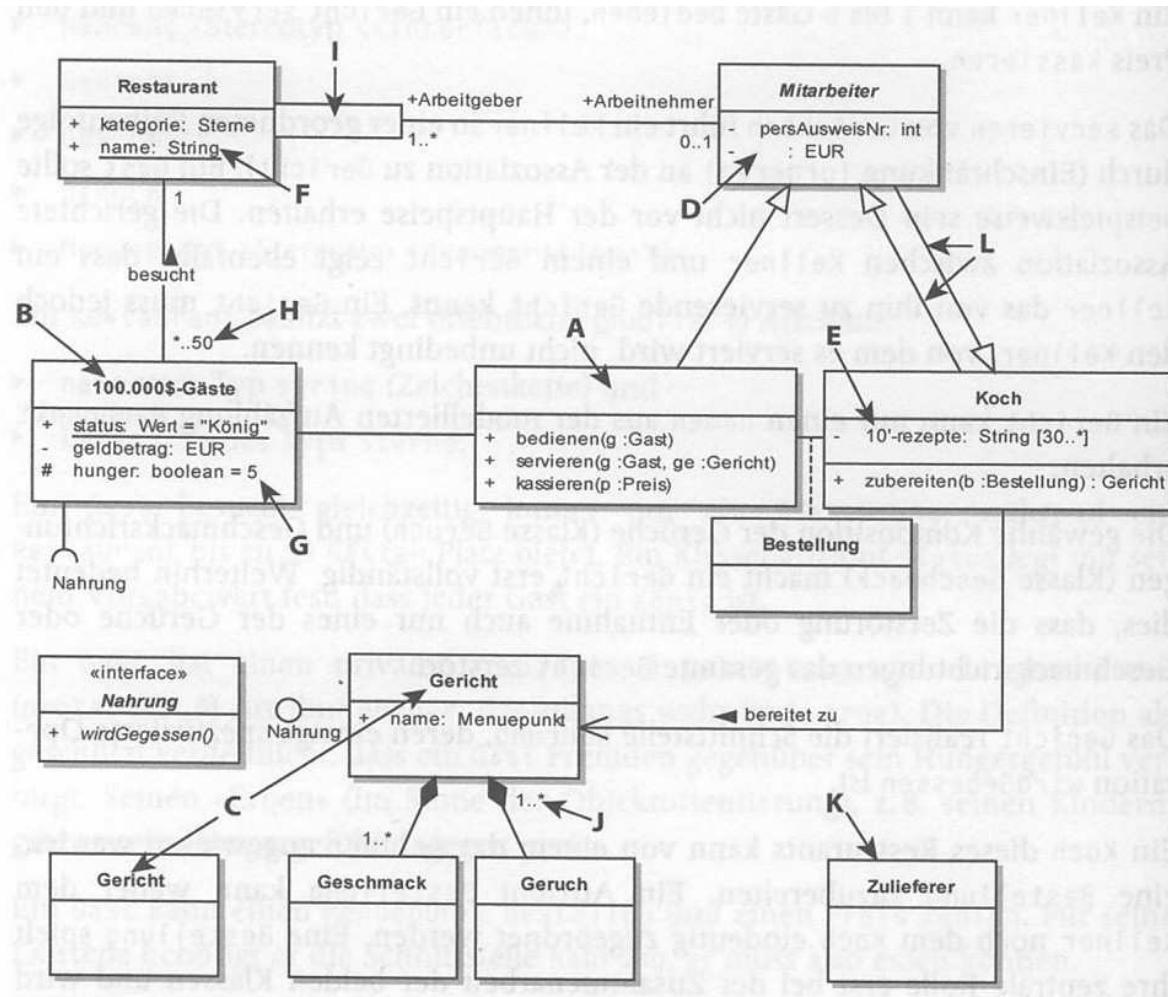
- Eine benötigte Schnittstelle wird als Socket-Symbol modelliert.





1. Was ist
Modellierung?
 2. OOAD mit der
UML
 3. Entity-
Relationship-
Modelle
 4. Geschäftsprozeß-
modellierung

Bennennen Sie die Fehler, ohne im Original nachzusehen



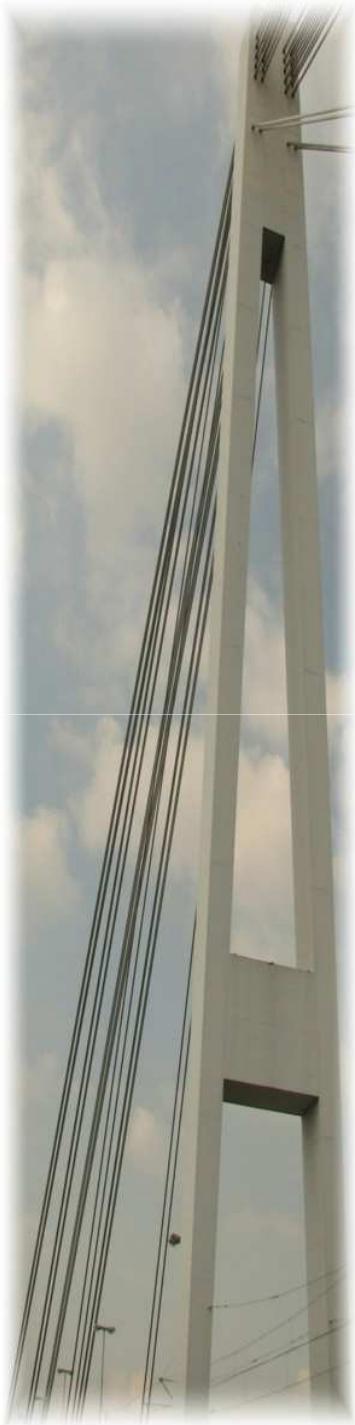
Aus: Kecher, C., UML 2.0, Bonn 2006, S. 106



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Übungsaufgabe

- 1) Erstellen Sie ein Klassendiagramm für die Bestandsführung einer Stadtbibliothek. Berücksichtigen Sie u.a. verschiedene Medienarten (z.B. DVD, Buch, Zeitschrift), Benutzergruppen (z.B. Schüler, Erwachsene), Entleihbeziehungen, Tarife. Verwenden Sie sinnvolle Attribute und Methoden. Definieren Sie sinnvolle Multiplizitäten und Navigierbarkeiten. Finden Sie dabei sinnvolle Anwendungen für die Notationselemente, die Sie kennen. (Bsp.: Verbundmedien aus einem Buch und beiliegender CD, Sprachen der DVD-Untertitel, maximale Anzahl entleihbarer Medien...)



2.6 Sequenz- und Aktivitätsdiagramme



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Aktivitätsdiagramme

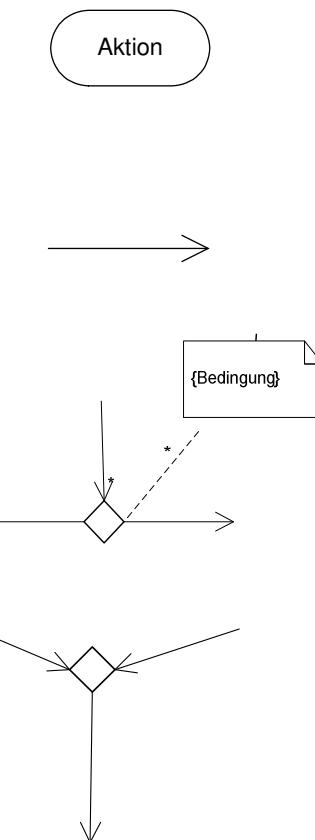
- Aktivitätsdiagramme beschreiben das Verhalten von Systemen.
- Sie können z.B. zur Detaillierung von Use Cases dienen.
 - Use Case: Was wird gemacht?
 - Aktivitätsdiagramm: Wie wird es gemacht?
- Sie können zur Modellierung von Geschäftsprozessen dienen.
- Sie können auch Abläufe in einem System beschreiben.
- Mit immer weiterem Detaillierungsgrad können sie sogar zur Beschreibung von Algorithmen dienen (und damit z.B. Nassi-Shneiderman-Diagramme ersetzen)



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Aktivitätsdiagramme

- Ein Aktivitätsdiagramm besteht u.a. aus folgenden Elementen:
 - Aktion: Kleinste modellierte Einheit ausführbarer Funktionalität
 - Kontrollfluß: Definiert die Reihenfolge der Aktionen
 - Entscheidungsknoten: Verzweigung des Kontrollflusses in Alternativen
 - Verbindungsknoten: Zusammenführung alternativer Kontrollflüsse

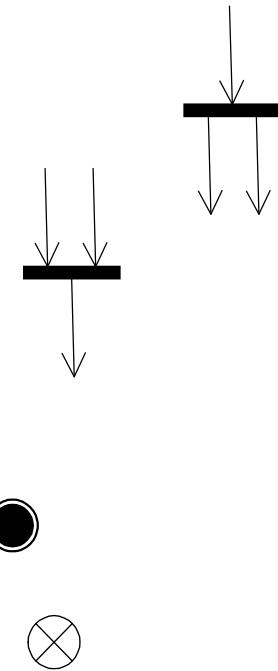




1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeßmodellierung

Aktivitätsdiagramme

- Ein Aktivitätsdiagramm besteht u.a. aus folgenden Elementen:
 - Gabelung : Aufspaltung in parallele Kontrollflüsse
 - Vereinigung: Zusammenführung paralleler Kontrollflüsse
 - Startzustand
 - Endzustand: Aktivitätsende, beendet alle Kontrollflüsse
 - Endknoten: Flußende, beendet nur den in ihn laufenden Kontrollfluß





- 1. Was ist Modellierung?**
- 2. OOAD mit der UML**
- 3. Entity-Relationship-Modelle**
- 4. Geschäftsprozeß-modellierung**

Aktivitätsdiagramme

- Ein Aktivitätsdiagramm besteht u.a. aus folgenden Elementen:
 - Aktivitätsbereich: Gruppiert Aktivitäten nach Zuständigkeit/Verantwortung zu Organisationseinheiten
 - Objekt: Wird bei Aktionen erzeugt und übermittelt.



- 1. Was ist Modellierung?**
- 2. OOAD mit der UML**
- 3. Entity-Relationship-Modelle**
- 4. Geschäftsprozeß-modellierung**

Mögliche Fragen zum Erstellen von Aktivitätsdiagrammen

- Aktivitätsbereiche und Aktionen: Welche Arbeitsschritte sind erforderlich und wer ist dafür verantwortlich?
- Kontrollfluß: Welche Reihenfolge der Aktionen gilt unter welcher Bedingung? Welche Aktionen können parallel zueinander ausgeführt werden?
- Objekte: Welche Objekte werden bei den Aktionen erzeugt?

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Übungsaufgabe

Erstellen Sie Aktivitätsdiagramme aus externer Sicht für

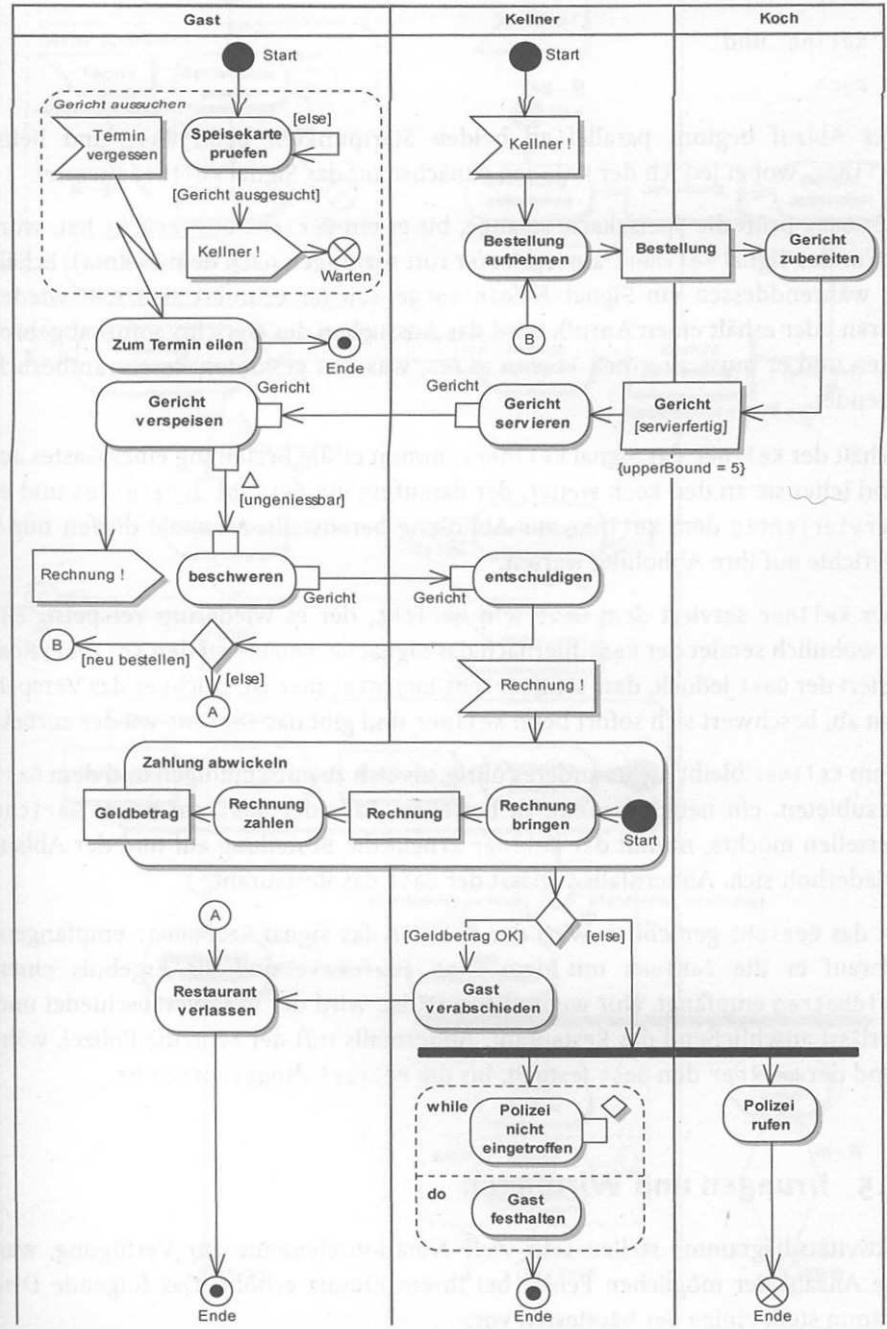
- a) Beschaffung eines neuen Personalausweises
- b) Mittagessen kaufen in der Mensa
- c) Geld abheben am Geldautomaten

1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeßmodellierung

Beschreiben Sie verbal dieses Diagramm.

Überlegen Sie, welche Bedeutung die Ihnen bislang unbekannten Notationselemente haben.

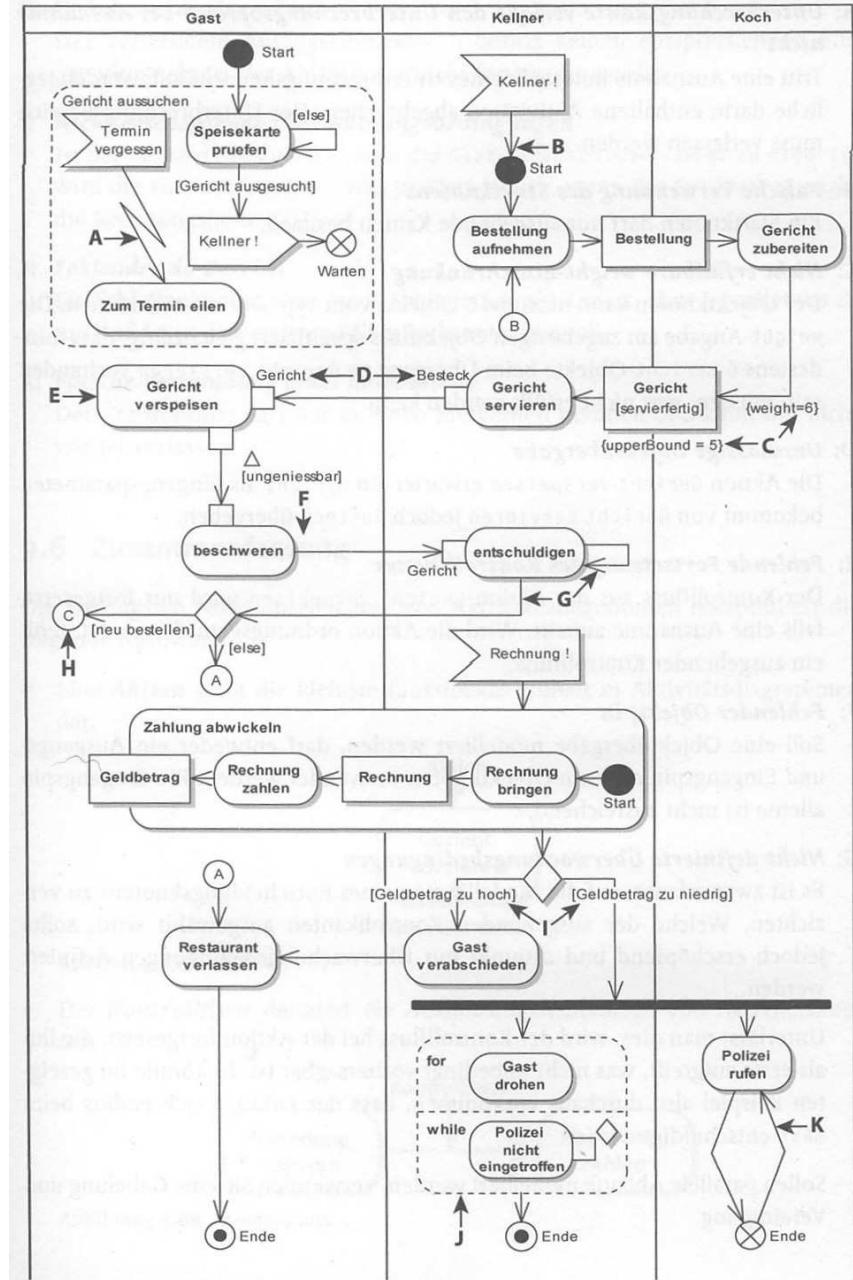
Aus: Kecher, C., UML 2.0, Bonn 2006, S. 283



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeßmodellierung

Bennennen Sie die Fehler, ohne im Original nachzusehen

Aus: Kecher, C., UML 2.0, Bonn 2006, S. 285





1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Sequenzdiagramme

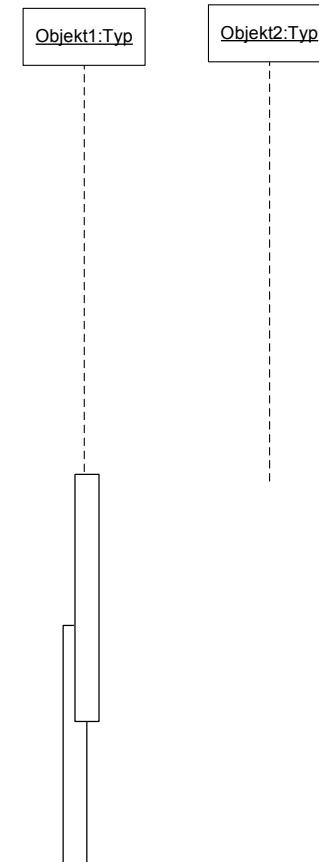
- Sequenzdiagramme haben die gleichen Anwendungsfelder wie Aktivitätsdiagramme. Der Schwerpunkt liegt aber auf der Chronologie der ausgetauschten Nachrichten zwischen Akteuren und System.
- Dabei kann das System wiederum ein IT-System sein, aber auch eine Organisation.
- Außerdem kommen sie zur Anwendung, um
 - den Nachrichtenaustausch zwischen Systemmodulen zu beschreiben,
 - die gegenseitigen Methodenaufrufe zwischen den Klassen eines Systems darzustellen,
 - die Interaktion zwischen Benutzer und Software, also die GUI-Funktionalität zu beschreiben.



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Sequenzdiagramme

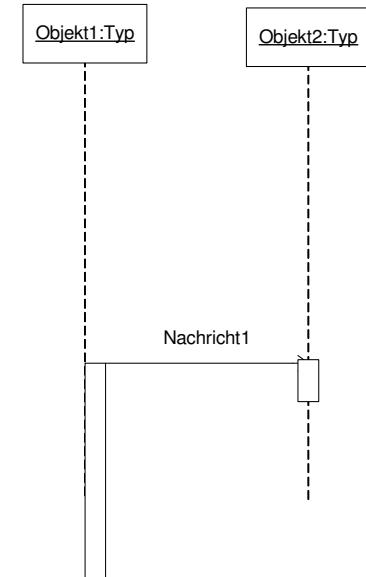
- Ein Sequenzdiagramm besteht u.a. aus folgenden Elementen:
 - Objekte mit Lebenslinien: Zeitstrahl von oben nach unten für ein Objekt. Wie bei Use cases werden Menschen als Strichmännchen, Systeme als Rechtecke gezeichnet.
 - Die Lebenslinie wird zum Ausführungsbalken, sobald ein Objekt aktiv wird, also z.B. Nachrichten sendet oder empfängt. Übereinander liegende Balken stehen für parallele Aufgaben eines Objektes.



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Sequenzdiagramme

- Ein Sequenzdiagramm besteht u.a. aus folgenden Elementen:
 - Nachrichten sind entweder der Aufruf einer Operation oder das Senden eines Signals. Sie werden als Pfeil notiert. Antwortnachrichten werden gestrichelt gezeichnet.





1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

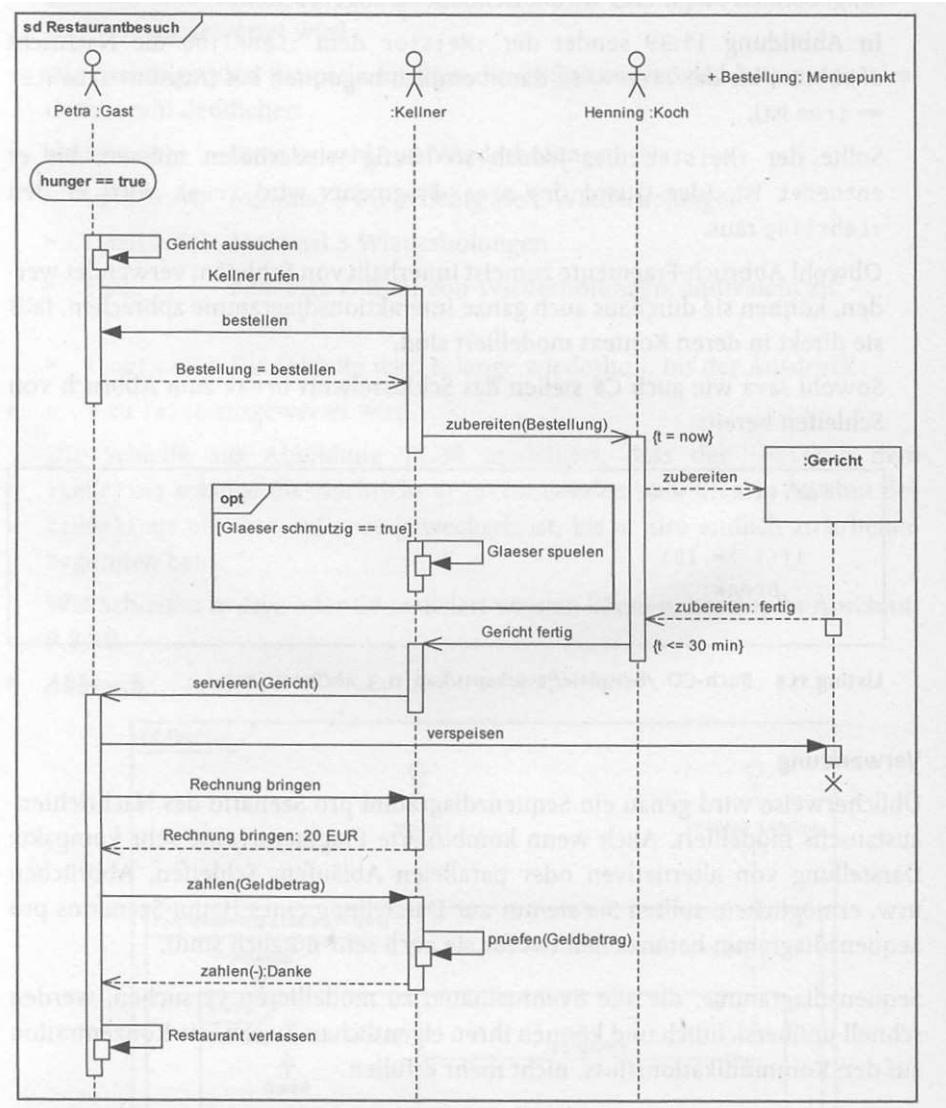
Mögliche Fragen zum Erstellen von Sequenzdiagrammen

- Objekte und Lebenslinien: Welche Menschen und welche Systeme treten als Objekte auf, die Nachrichten versenden oder empfangen können?
- Nachrichten: In welcher Reihenfolge wird zwischen welchen Objekten was kommuniziert?

1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Beschreiben Sie verbal dieses Diagramm.

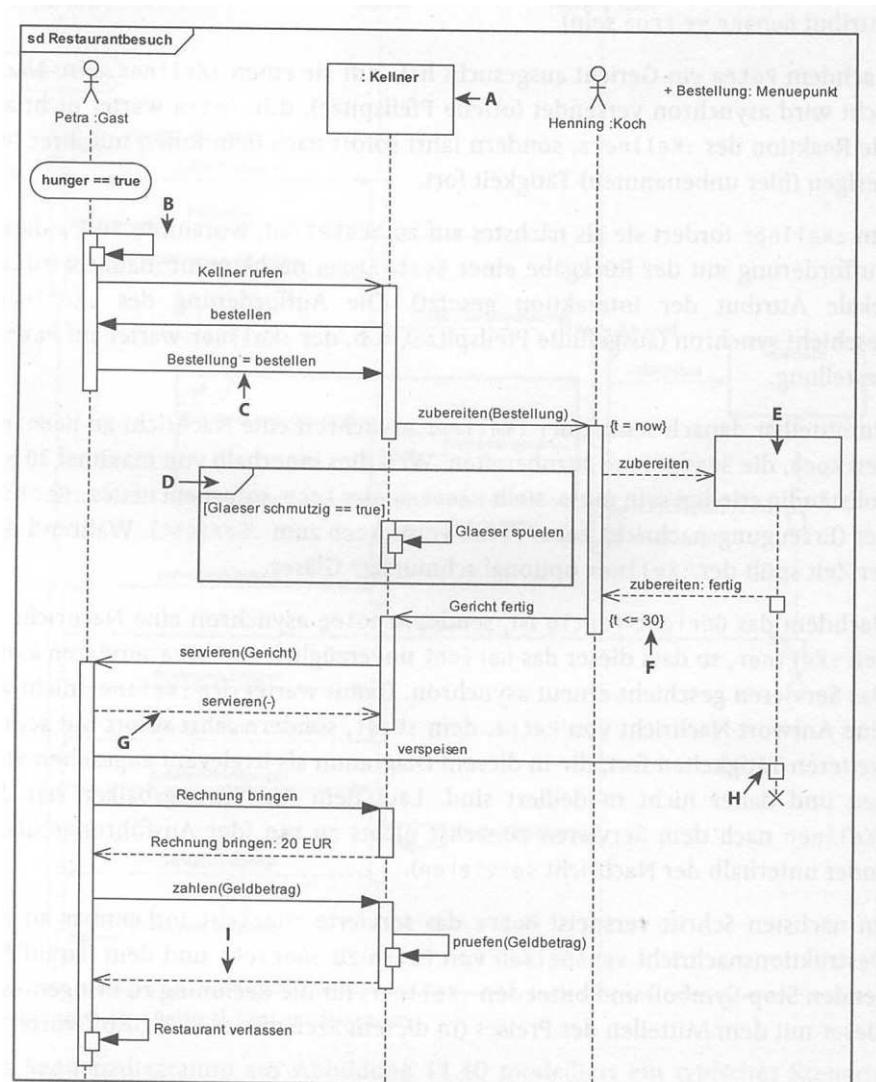
Überlegen Sie, welche Bedeutung die Ihnen bislang unbekannten Notationselemente haben.



Aus: Kecher, C., UML 2.0, Bonn 2006, S. 372

1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Bennennen Sie die Fehler, ohne im Original nachzusehen



Aus: Kecher, C., UML 2.0, Bonn 2006, S. 374



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Übungsaufgabe

1. Erstellen Sie Sequenzdiagramme aus externer Sicht für
 - a) Beschaffung eines neuen Personalausweises
 - b) Mittagessen kaufen in der Mensa
 - c) Geld abheben am Geldautomaten
2. Vergleichen Sie Ihre Modelle mit den entsprechenden Aktivitätsdiagrammen. Für welche Zwecke ist ein Sequenz-, für welche ein Aktivitätsdiagramm geeigneter?



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeßmodellierung

Übungsaufgabe

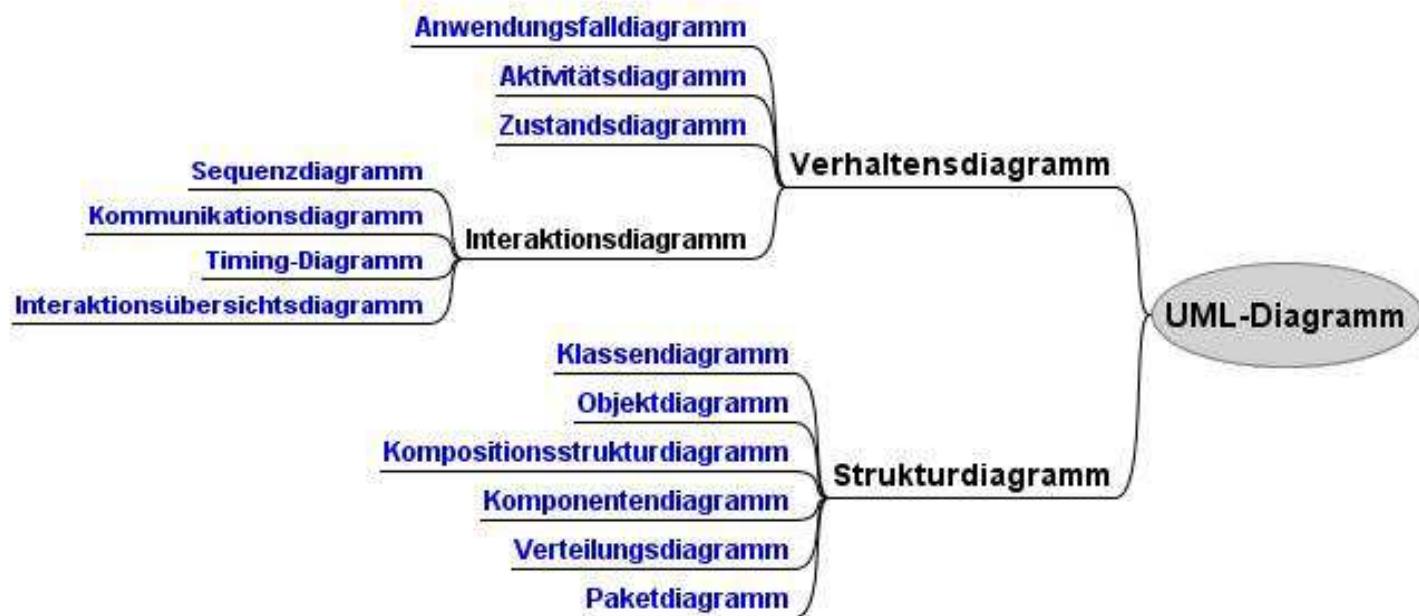
- Erstellen Sie ein Aktivitätsdiagramm für den Anwendungsfall "Bestellung tätigen" in einem Online-Shop Ihrer Wahl. Ergänzen Sie es danach um die Sicht interner Aktivitäten, z.B. die Zuständigkeiten von Online-Shop, Beschaffung, Lager und Debitorenbuchhaltung.



2.7 Weitere UML-Diagramme im Überblick

1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Übersicht



Außerdem gibt es noch Profildiagramme. Sie dienen ausschließlich der Metamodellierung und sind deshalb für den Endanwender der UML ohne weitere Bedeutung.



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Kompositionsstrukturdiagramme, Paketdiagramme, Komponentendiagramme

- Sie alle beschreiben den Aufbau eines Systems aus Komponenten.
- Sie dienen der Übersicht über den Aufbau komplexer Systeme und der Verteilung von Verantwortlichkeiten für Teilsysteme.
- Sie können hilfreich sein, wenn man Komponentengrenzen z.B. nach dem Kriterium der Wiederverwendbarkeit oder der Qualifikation von Teams definieren will.
- Komponentendiagramme bieten eine Übersicht über Schnittstellen zwischen den Komponenten eines Systems.
- Kompositionsstrukturdiagramme sind detailliert bezüglich des internen Aufbaus der Komponenten.
- Paketdiagramme beschreiben z.B. Import-Beziehungen zwischen Paketen.



Deploymentdiagramme (Verteilungsdiagramme)

1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

- Deploymentdiagramme beschreiben den Aufbau von Hard- und Software zu einem lauffähigen System.
- Sie enthalten z.B. Informationen darüber, welche Softwarepakete auf welchem Server installiert werden und auf welche anderen Pakete auf welchen anderen Servern zugreifen.



- 1. Was ist Modellierung?**
- 2. OOAD mit der UML**
- 3. Entity-Relationship-Modelle**
- 4. Geschäftsprozeß-modellierung**

Objektdiagramme

- Objektdiagramme haben große Ähnlichkeit mit Klassendiagrammen.
- Sie beschreiben aber nicht allgemeine Beziehungen zwischen Klassen, sondern eine Momentaufnahme mit Objekten, die erst zur Laufzeit erzeugt werden.
- Sie werden vor allem ergänzend zu Klassendiagrammen eingesetzt, um diese anhand von Beispielen zu veranschaulichen.

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Zustandsdiagramme

- Zustandsdiagramme ergänzen vor allem Aktivitätsdiagramme.
- Sie beschreiben, unter welchen Bedingungen ein Objekt seinen Zustand wechselt
- Damit eignen sie sich auch für die Definition von Kommunikationsprotokollen, wie sie beim Austausch von Nachrichten zwischen verschiedenen Systemen notwendig werden.

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Kommunikationsdiagramme

- Kommunikationsdiagramme beschreiben die Kommunikation zwischen Objekten auf einem höheren Abstraktionsniveau als Sequenzdiagramme.
- Sie beschreiben nicht die Nachrichten selbst, sondern nur, welche Objekte überhaupt in irgendeiner Form miteinander kommunizieren.



- 1. Was ist Modellierung?**
- 2. OOAD mit der UML**
- 3. Entity-Relationship-Modelle**
- 4. Geschäftsprozeß-modellierung**

Timingdiagramme

- Timingdiagramme beschreiben die Kommunikation zwischen Objekten.
- Sie betonen den zeitlichen Ablauf der Kommunikation.
- Zur Unterscheidung:
 - Kommunikationsdiagramm: Wer kommuniziert?
 - Sequenzdiagramm: Wie und in welcher Reihenfolge wird kommuniziert?
 - Timingdiagramm: In welchem zeitlichen Abstand wird kommuniziert?



- 1. Was ist Modellierung?**
- 2. OOAD mit der UML**
- 3. Entity-Relationship-Modelle**
- 4. Geschäftsprozeß-modellierung**

Interaktionsübersichtsdiagramme

- Ein Interaktionsübersichtsdiagramm ist eine Art Aktivitätsdiagramm, dessen Aktionen wieder ganze Sequenz-, Kommunikations- und Timingdiagramme sind.
- Sie haben den Vorteil, daß man gleichzeitig die Übersicht und den Blick auf die Details hat.



3. Modellierung relationaler Datenbanken mit Entity-Relationship- Modellen



1. Was ist
Modellierung?
 2. OOAD mit der
UML
 3. Entity-
Relationship-
Modelle
 4. Geschäftsprozeß-
modellierung

Entity-Relationship-Modelle

- Entity-Relationship-Modelle dienen der Einrichtung von relationalen Datenbanken.
 - Sie gehen zurück auf einen Aufsatz von Peter Chen, der 1976 eine "einheitliche Sicht auf Daten" vorgeschlagen hat.

The Entity-Relationship Model—Toward a Unified View of Data

PETER PIN-SHAN CHEN
Massachusetts Institute of Technology

A data model, called the entity-relationship model, is proposed. This model incorporates some of the important semantic information about the real world. A special diagrammatic technique is introduced as a tool for database design. An example of database design and description using the model and the diagrammatic technique is given. Some implications for data integrity, information retrieval, and data manipulation are discussed.

The entity-relationship model can be used as a basis for unification of different views of data: the network model, the relational model, and the entity set model. Semantic ambiguities in these models are analyzed. Possible ways to derive their views of data from the entity-relationship model are presented.

Key Words and Phrases: database design, logical view of data, semantics of data, data models, entity-relationship model, relational model, Data Base Task Group, network model, entity set model, data definition and manipulation, data integrity and consistency

CR Categories: 3.50, 3.70, 4.33, 4.34

1. INTRODUCTION

The logical view of data has been an important area of research. Several data models have been proposed: the network model [8], and the entity set model [25]. These models have their own strengths and weaknesses. The network model provides a more natural way of representing entities and relationships (to a certain extent independence has been challenged [8]). The relational theory and can achieve a high degree of data consistency. The entity set model, which is based on set theory, also has its advantages, but its viewing of values such as sets and sequences has been questioned by some people [25].

This paper presents the entity-relationship advantages of the above three models. The more natural view that the real world cons



Copyright © 1976, Association for Computing Machinery, Inc. General permission to republish, but not for profit, all or part of this material is granted provided that ACM's copyright notice is given and that reference is made to the publication, to its date of issue, and to the fact that reprinting privileges were granted by permission of the Association for Computing Machinery. A version of this paper was presented at the International Conference on Very Large Data Bases, Framingham, Mass., Sept. 22-26, 1975.

Author's address: Center for Information System Research, Alfred P. Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA 02139.

ACM Transactions on Database Systems, Vol. 1, No. 1, March 1976, Pages 9-36

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Entity-Relationship-Modelle

- Entity-Relationship-Modelle sind nicht objektorientiert.
- Da in vielen Projekten relationale DBMS mit objektorientierten Programmiersprachen kombiniert werden, werden ERM nach wie vor häufig verwendet.
- Bei der Entwicklung von Anwendungen dagegen werden objektorientierte Modellierungsmethoden verwendet.
- Dies führt zu vielen Problemen, weil Klassen und Relationen sehr unterschiedlich sind. Beim Speichern und Wiederherstellen von Objekten muß immer zwischen beiden Konzepten "übersetzt" werden. (Impedance Mismatch)



- 1. Was ist Modellierung?**
- 2. OOAD mit der UML**
- 3. Entity-Relationship-Modelle**
- 4. Geschäftsprozeß-modellierung**

Entity-Relationship-Modelle

- Es führt aber auch schon für die Modellierung zu einer unschönen Situation:
 - Die Datenstrukturen werden als Klassendiagramm modelliert, das als Vorlage für die Entwicklung dient.
 - Persistent sind die Objekte jedoch erst in der (meistens nicht objektorientierten, sondern relationalen) Datenbank.
 - Das Modell zur Einrichtung der Datenbank ist das ERM.
 - Man muß also zwei Diagramme erstellen, die sehr eng miteinander verwandt sind, aber doch deutliche Unterschiede haben und zu allem Überfluß auch noch konsistent gehalten werden müssen.

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeßmodellierung

Entity-Relationship-Modelle

- Entwurfsebenen für Relationale Datenbanken
 - Konzeptuelle Ebene: Graphische Darstellung der Datenstrukturen mit Modellierungsverfahren
 - Gebräuchlich: Entity-Relationship-Diagramme
 - Implementierungsebene: Realisierung des konzeptuellen Modells im DBMS
 - Bei RDBMS nicht Klassen, sondern Relationen (Tabellen)
 - Physische Ebene: Abbildung auf das Speichermedium
 - Dient vor allem der Performance
 - Anordnung von Daten, die häufig gemeinsam genutzt werden, auf zusammenliegende Bereiche der Platte
 - Anlegen von Indizes
 - Je nach verwendetem DBMS und Betriebssystem sehr unterschiedlich



- 1. Was ist Modellierung?**
- 2. OOAD mit der UML**
- 3. Entity-Relationship-Modelle**
- 4. Geschäftsprozeß-modellierung**

Entity-Relationship-Modelle

- Konzeptueller Entwurf mit dem Entity-Relationship-Modell (ERM)
 - Bevor die Datenbank eingerichtet wird, macht man sich zunächst ein graphisches Modell des abzubildenden Ausschnittes aus der Realwelt.
 - Konzepte des ERM sind
 - Entitäten (Seiendes)
 - Relationen (Beziehungen)
 - Attribute (Eigenschaften)
 - Rollen

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Entity-Relationship-Modelle

- Entitäten...
 - sind Gegenstände aus der Realwelt, die die Datenbank abbilden soll.
 - werden zu Entitätstypen zusammengefaßt.
 - werden als Rechteck dargestellt, in dem der Name des Entitätstyps (i.d.R. ein Substantiv) steht.

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Entity-Relationship-Modelle

- Beziehungen...
 - bestehen zwischen den Entitäten.
 - werden zu Beziehungstypen zwischen Entitätstypen zusammengefaßt.
 - werden als Raute dargestellt, in der der Name des Beziehungstyps (i.d.R. ein Verb) steht.
 - werden durch Linien mit den beteiligten Entitätstypen verbunden.
 - können auch rekursiv sein (zwischen zwei Entitäten des selben Typs)



- 1. Was ist Modellierung?**
- 2. OOAD mit der UML**
- 3. Entity-Relationship-Modelle**
- 4. Geschäftsprozeß-modellierung**

Entity-Relationship-Modelle

- Rollen...
 - beschreiben, auf welche Art eine Entität eine Beziehung eingeht.
 - werden der Verbindung zwischen einem Beziehungstyp und einem Entitätstyp zugeordnet.
 - werden dargestellt als Beschriftung der Linien zwischen einem Entitätstyp und einem Beziehungstyp.

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Entity-Relationship-Modelle

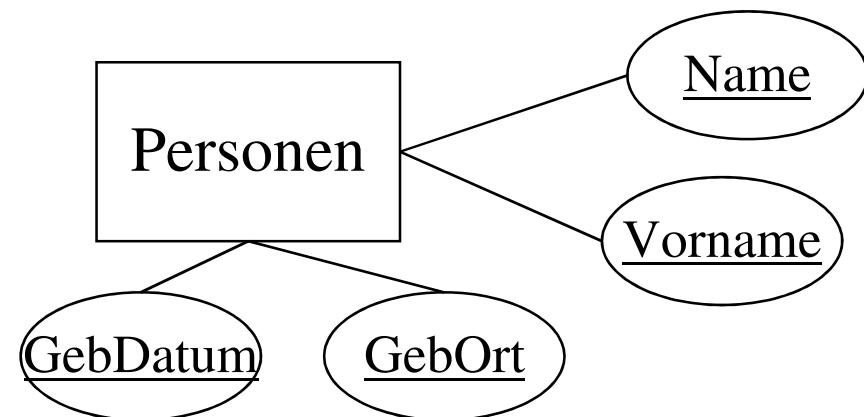
- Attribute...
 - beschreiben Entitätstypen oder Beziehungstypen.
 - werden als Ovale dargestellt, in denen der Name des Attributes steht.
 - werden durch Linien mit den beschriebenen Beziehungs- oder Entitätstypen verbunden.



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Entity-Relationship-Modelle

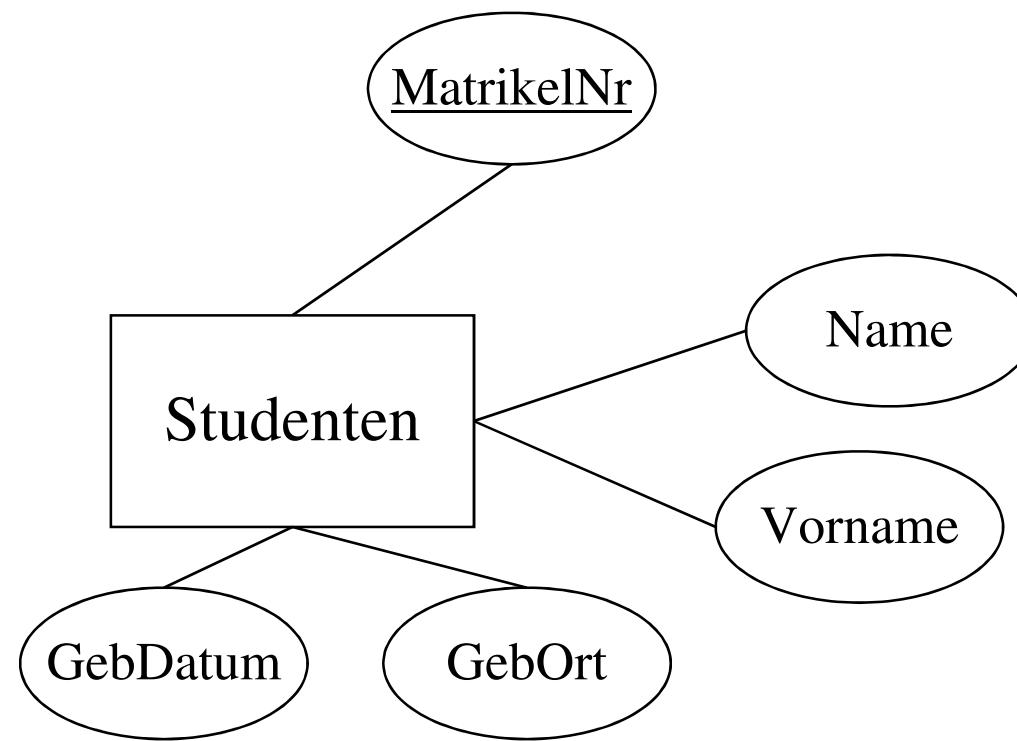
- Schlüsselattribute...
 - sind eine minimale Menge von Attributen, deren Werte eine Entität eindeutig von allen anderen Entitäten des selben Typs unterscheiden.
 - werden durch Unterstreichung gekennzeichnet
 - Beispiel: Behörden gehen üblicherweise davon aus, daß die Menge von Attributen {Name, Vorname, Geburtstag, Geburtsort} eine Person eindeutig identifiziert.



- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Entity-Relationship-Modelle

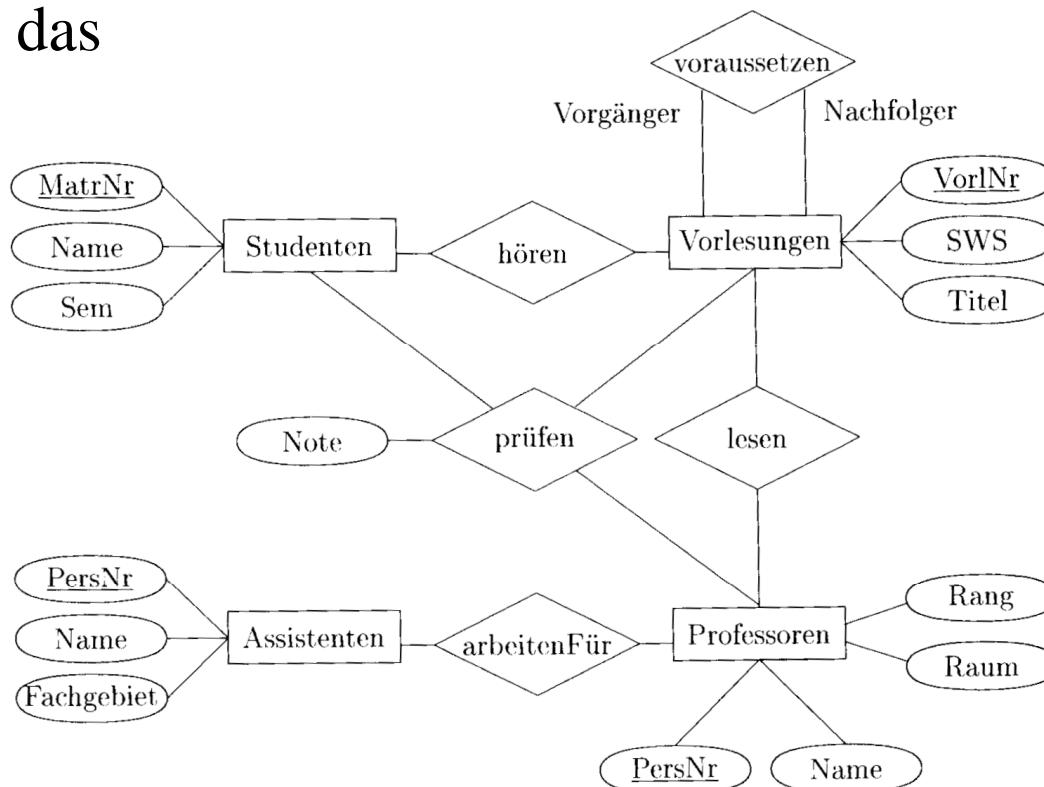
- In IT-Systemen wird häufig ein einzelnes Attribut als künstlicher Schlüssel eingeführt.



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeßmodellierung

Übungsaufgabe

Beschreiben
Sie verbal das
ERM:



Quelle: [KE], S.36



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Entity-Relationship-Modelle

- Entitätstypen können als Mengen E_k ihrer Entitäten $\{e_{k1}, e_{k2}, e_{k3} \dots\}$ betrachtet werden.
 - Beispiel: In einer Datenbank enthält
 - die Menge "Politiker" E_1 die Entitäten $e_{11}=\text{Merkel}$, $e_{12}=\text{Köhler}$, $e_{13}=\text{Schröder}$.
 $E_1=\{\text{Merkel, Köhler, Schröder}\}$
 - die Menge der Ämter E_2 die Entitäten $e_{21}=\text{Bundespräsident}$, $e_{22}=\text{Bundeskanzler}$.
 $E_2=\{\text{Bundespräsident, Bundeskanzler}\}$



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Entity-Relationship-Modelle

- Die Beziehungstypen können als Relation (Teilmenge des kartesischen Produkts) betrachtet werden
$$R \subseteq E_1 \times E_2 \times \dots \times E_n$$
 - Beispiel: In obiger Datenbank enthält die Relation
$$R \subseteq E_1 \times E_2$$
 "hat inne" die Werte
$$R = \{(e_{11}, e_{22}), (e_{12}, e_{21})\}$$
- An den meisten Beziehungstypen sind genau zwei Entitätstypen beteiligt. Sie heißen *binär*.
- Es kann durchaus Entitäten e_{ki} geben, die an gar keiner Relation beteiligt sind.

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Übungsaufgabe

E_1 sei die Menge der Bundesligavereine = {Bayern, VfB, Dortmund...}

E_2 sei die Menge der Titel = {Meister, Pokalsieger, Vizemeister, UEFA-Cup-Teilnehmer}

- a) Wie lautet in Mengenschreibweise
die Relation "hat Titel inne"
die Relation "hat in der letzten Saison daheim besiegt"
- b) Zeichnen Sie ein ERM

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Entity-Relationship-Modelle

- Funktionalitäten von binären Beziehungen sind Integritätsbedingungen.
- 1:1-Beziehungen
Jedem $e_{1i} \in E_1$ darf höchstens ein $e_{2j} \in E_2$ zugeordnet sein. Umgekehrt darf jedem $e_{2j} \in E_2$ höchstens ein $e_{1i} \in E_1$ zugeordnet sein.
Beispiel: Relation "ist verheiratet mit"
- 1:n-Beziehungen
Jedem $e_{1i} \in E_1$ darf höchstens ein $e_{2j} \in E_2$ zugeordnet sein. Umgekehrt dürfen jedem $e_{2j} \in E_2$ beliebig viele $e_{1i} \in E_1$ zugeordnet sein.
Beispiel: Relation "ist Vater von"

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Entity-Relationship-Modelle

- n:m-Beziehungen
Es gelten keine Restriktionen
Beispiel: Relation "sind Geschwister"
- Funktionalitäten sollten nur eingesetzt werden, wenn sie echte Regeln der Realwelt repräsentieren.
- Wenn die Inhalte einer Datenbank zufällig die Bedingung einer Funktionalität erfüllen, dies aber nicht müssen, sollte keine Funktionalität definiert werden.
Beispiel: In der Datenbank ist zufällig nur ein Geschwisterpaar enthalten. Dennoch ist die Relation "sind Geschwister" keine 1:1-Beziehung.
- Nicht alle Integritätsbedingungen lassen sich im ERM darstellen. Viele werden erst später auf Ebene der Programme realisiert, die auf die Datenbank zugreifen.



Übungsaufgabe

- 1. Was ist Modellierung?**
- 2. OOAD mit der UML**
- 3. Entity-Relationship-Modelle**
- 4. Geschäftsprozeß-modellierung**

Erstellen Sie ein ERM mit Funktionalitäten für die Verwaltung Ihrer CD-Sammlung!



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

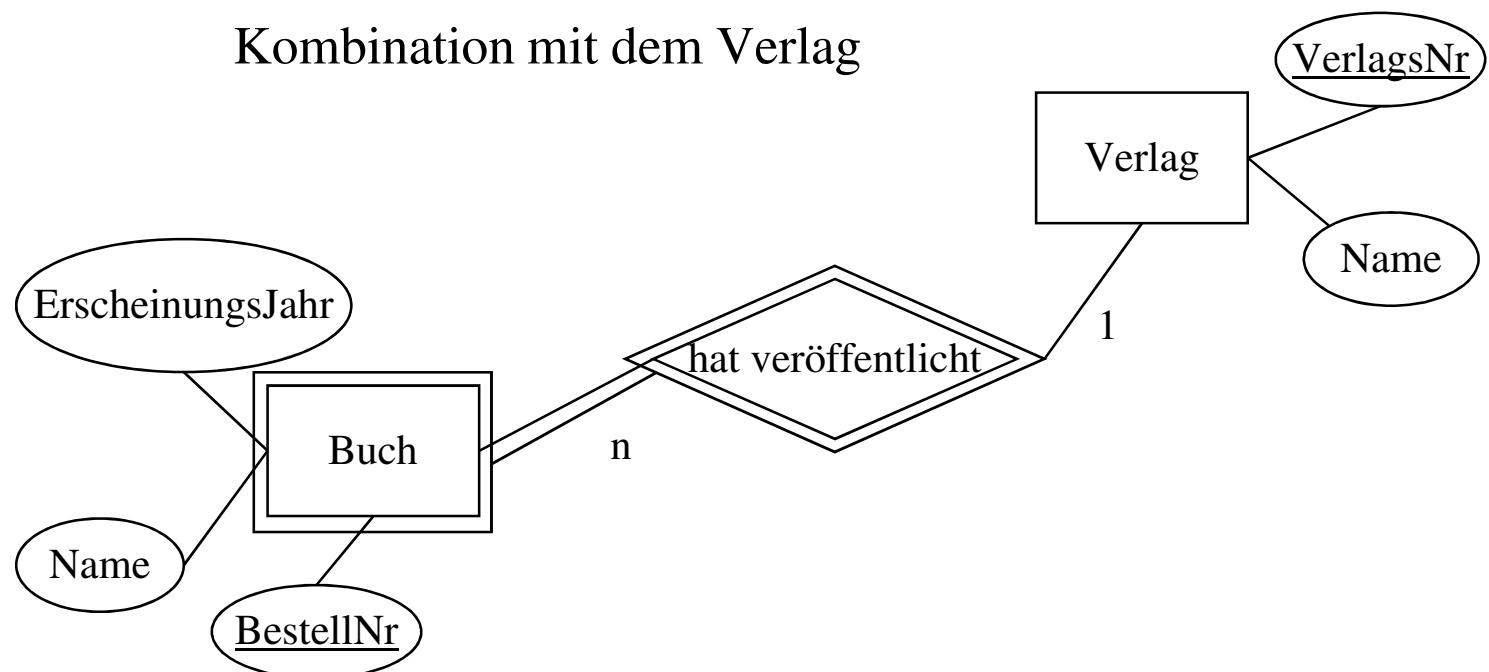
Entity-Relationship-Modelle

- Schwache (existenzabhängige) Entitätstypen
 - Entitäten schwacher Entitätstypen sind nicht durch ihre eigenen Attribute eindeutig identifizierbar.
 - Zusätzlich benötigt man zur eindeutigen Identifizierung die Relation zu einem übergeordneten Entitätstyp.
 - Diese Relation kann keine n:m-Beziehung sein, da sonst eine Identifizierung nicht eindeutig wäre.
 - Das Rechteck und die Linie der "identitätsstiftenden" Beziehung werden doppelt gezeichnet.

1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeßmodellierung

Entity-Relationship-Modelle

- Beispiel für einen schwachen Entitätstypen
 - Die Bestellnummer identifiziert ein Buch nur in Kombination mit dem Verlag



- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Entity-Relationship-Modelle

- Generalisierung
 - Ähnliche Entitätstypen werden zu Obertypen zusammengefaßt, die die gemeinsamen Attribute und Beziehungen enthalten.
 - Spezielle Attribute und Beziehungen werden auf Ebene der Untertypen modelliert.
 - Dies entspricht der Vererbung in objektorientierten Programmiersprachen. Einziger Unterschied: Methoden werden in ERM nicht modelliert, weil sie in RDBMS ohnehin nicht unterstützt werden.
 - Modelliert wird die Generalisierung durch ein Sechseck. Pfeile verlaufen von Untertypen zur Generalisierung und von der Generalisierung zum Obertypen.

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Entity-Relationship-Modelle

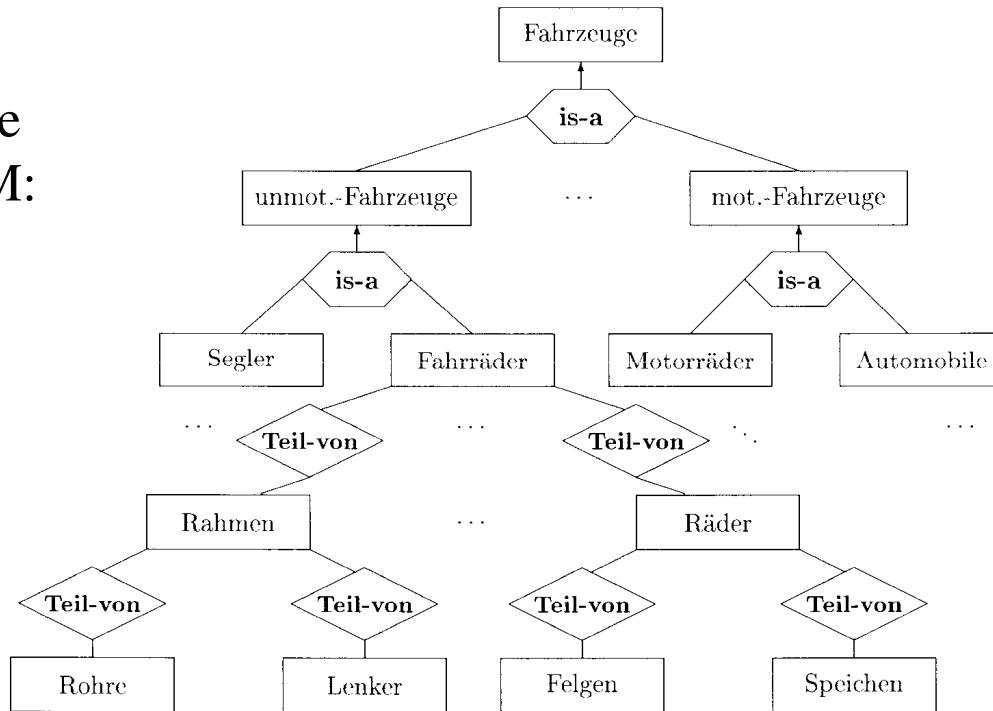
- Aggregation
 - Entitäten verschiedener Typen sind Teil einer Entität eines übergeordneten Typen.
 - Dies ist eine Spezialform der Beziehungstypen.
 - Aggregation wird daher als gewöhnlicher Beziehungstyp modelliert.
 - Man kann Rollen zuweisen, die bezeichnen, welcher Typ in welchen anderen eingeht.

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Entity-Relationship-Modelle

Aufgabe:

Beschreiben Sie verbal das ERM:



Quelle: [KE], S.51

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Übungsaufgabe

Erstellen Sie ein ERM für ein Buchungssystem einer Fluggesellschaft. Folgende Annahmen sind zu beachten.

- Ein Flug hat eine eindeutige Flugnummer und eine Abflugszeit.
- Ein Flugsteig hat eine eindeutige Nummer und eine Ortsbezeichnung.
- Jeder Passagier hat einen Namen und besitzt Tickets mit einer eindeutigen Nummer.
- Jeder Sitz im Flugzeug hat eine Nummer. (Nur sinnvoll in Verbindung mit der Flugnummer)
- Zu jedem Sitz ist hinterlegt, ob er für Raucher oder für Nichtraucher gedacht ist.
- Ein Flug wird an genau einem Flugsteig abgefertigt.
- An einem Flugsteig können beliebig viele Flüge abgefertigt werden.
- Eine Ticketnummer ist genau einem Flug zugeordnet.
- Eine Ticketnummer ist genau einem Sitzplatz zugeordnet.
- Ein Sitzplatz kann leer oder von einer Person besetzt sein.
- Ein Sitzplatz ist genau einem Flug zugeordnet



4. Geschäftsprozeßmodellierung mit Ereignisgesteuerten Prozeßketten (EPK)



- 1. Was ist Modellierung?**
- 2. OOAD mit der UML**
- 3. Entity-Relationship-Modelle**
- 4. Geschäftsprozeßmodellierung**

Geschäftsprozesse

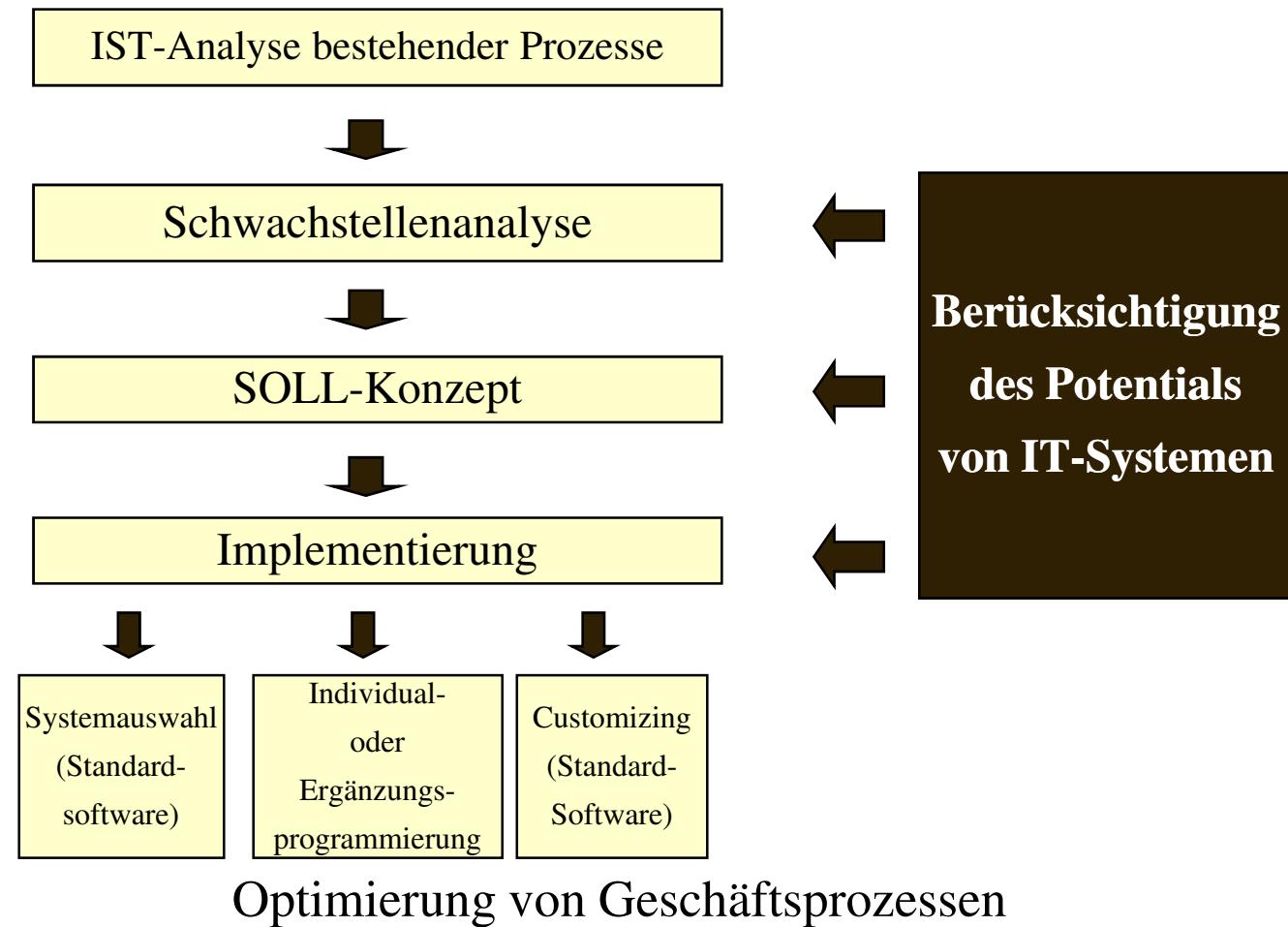
Definition Geschäftsprozeß

"zusammenhängende abgeschlossene Folge von Tätigkeiten, die zur Erfüllung einer betrieblichen Aufgabe notwendig sind" (Staud, J., a.a.O., S. 9)

- Ein Geschäftsprozeß zieht sich i.d.R. durch mehrere Organisationseinheiten, die jeweils für einzelne Funktionen (d.h. Verrichtungen an einem betriebswirtschaftlichen Objekt) zuständig sind.
- Noch selten: Optimierte Geschäftsprozesse zwischen Unternehmen

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeßmodellierung

Geschäftsprozesse





- 1. Was ist Modellierung?**
- 2. OOAD mit der UML**
- 3. Entity-Relationship-Modelle**
- 4. Geschäftsprozeßmodellierung**

Geschäftsprozesse

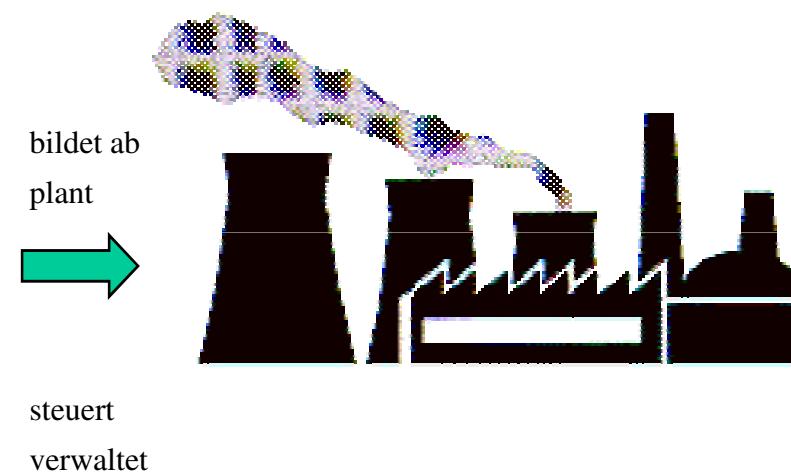
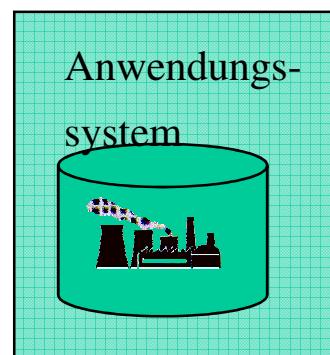
- Geschäftsprozeßmodelle in Anwendungssystemen
 - Explizite Abbildung: Ein Modell des Geschäftsprozesses liegt grafisch vor (IST-Analyse und SOLL-Konzept)
 - Implizite Abbildung: Der Geschäftsprozeß ist im Anwendungssystem eingerichtet (Implementierung)
"Software von SAP beispielsweise ist ... gefrorenes Organisationswissen." *

* A.-W. Scheer
(in seiner Eigenschaft als Bitkom-Präsident)
in einem Interview
mit der FAZ am 04.09.2007)



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeßmodellierung

Geschäftsprozesse in Anwendungssystemen



- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeßmodellierung

Geschäftsprozesse

Softwareeinsatz bei der Geschäftsprozeßoptimierung

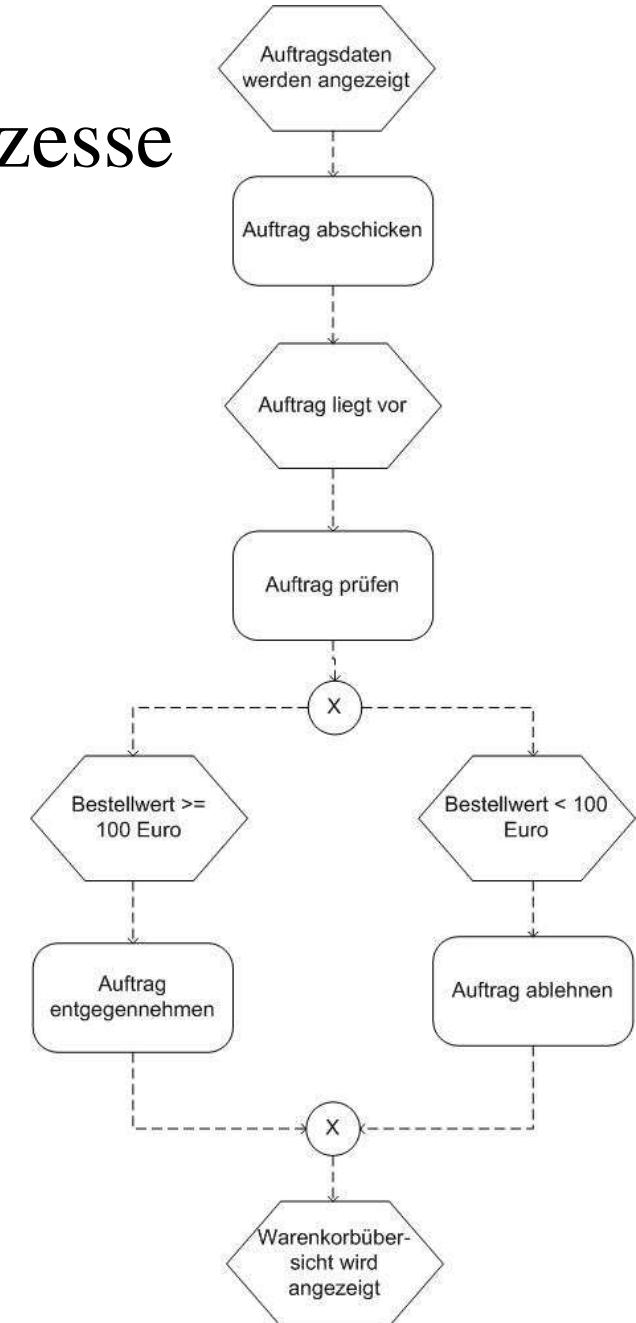
Prozeßmodellierung	Prozeßunterstützung
Einsatz von Tools bei <ul style="list-style-type: none">• IST-Analyse• SOLL-Konzept	Integrierte Systeme für <ul style="list-style-type: none">• ERP• CRM• EAI• E-Business• ...
Bsp.: Ereignisgesteuerte Prozeßketten mit ARIS	Bsp.: Mysap

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeßmodellierung

Geschäftsprozesse

Beispiel: Prozeßmodell

1. Kunde sieht im Online-Shop seine Auftragsdaten
2. Mindestbestellwert wird geprüft
3. Auftrag wird entgegengenommen oder abgelehnt
4. Kunde sieht die Warenkorbübersicht



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Geschäftsprozesse

Umsetzung durch den Administrator im Online-Shop-System

The screenshot shows a web-based administration interface for a customer profile. The title bar reads "Heidelberg - Maintain SSU data - Microsoft Internet Explorer". The address bar shows the URL "http://weapp41011:9040/ols/admin/profile/ssu.jsp". The main content area is titled "Maintain SSU data". On the left, there is a sidebar with links: Welcome, SSU data (selected), Catalog, Customers, New Customer, Batch Upload, Change Password, Reports, Scenarios, and Help Administration. The main form contains the following fields:

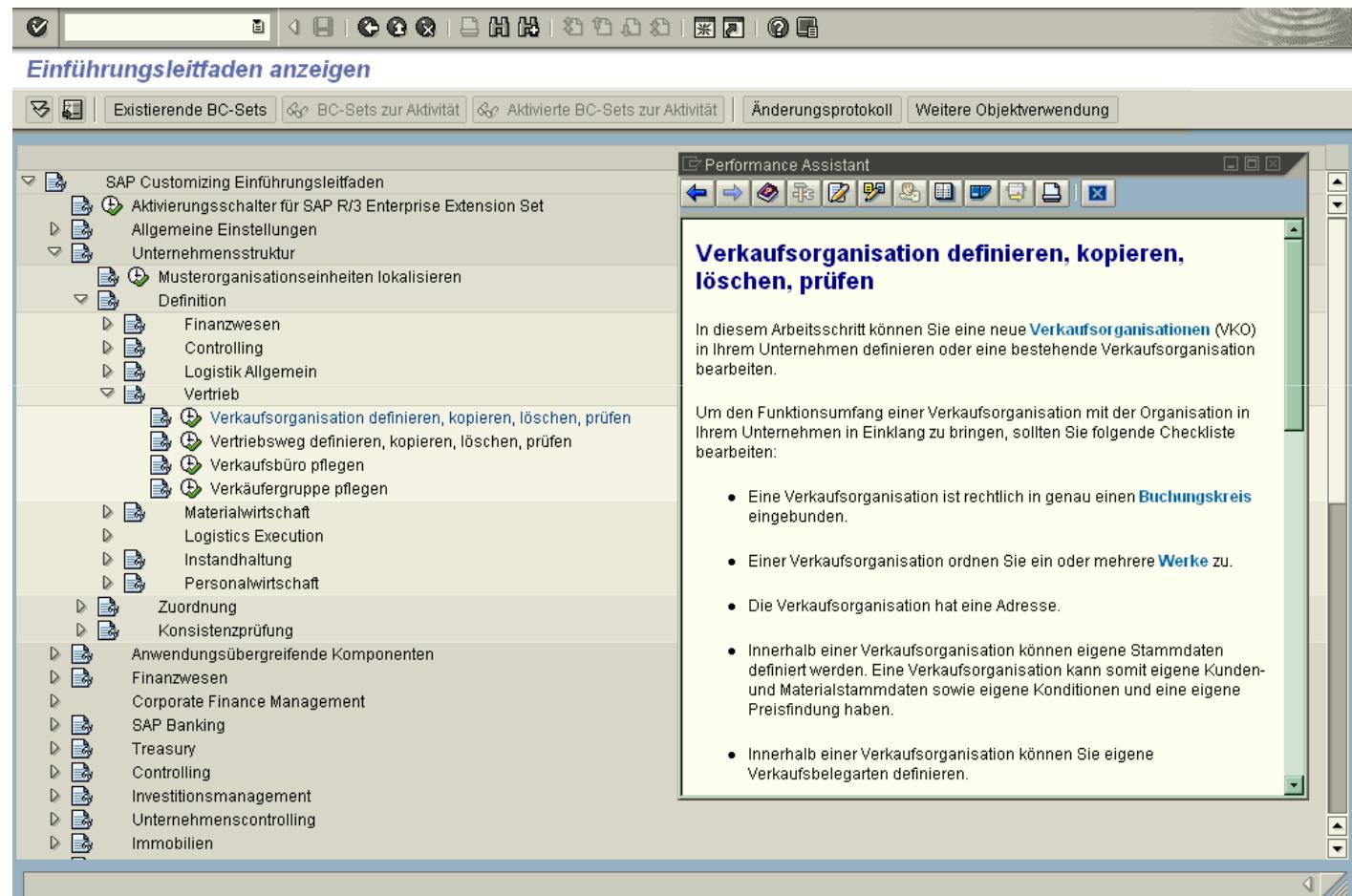
SSU ID	23
SSU Short Name	HAT
SSU Name	Heidelberger Druckmaschinen Österreich
Admin e-mail	admin.ols@heidelberg.com
Sales Group	salesgroup
Sales Office	salesoffice2
Purchase Order Type	OLS
Dummy Backend Customer ID	A10898
Warning: Do not change this customer because this is a very special customer with no rebates, also no product-related rebates (special programming has been done for this customer in most backend systems)!	
Default Shipping Condition	01
Call Center	017001400
Minimum Order Limit	100.0
Advertising Link 1	http://www.heidelberg.com
Advertising Link 2	http://www.heidelberg.de

A red circle highlights the "Minimum Order Limit" field, which is set to "100.0".

1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeßmodellierung

Geschäftsprozesse

Beispiel: Customizing R/3



- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

Fallbeispiel

Ein Systemhaus bietet Computersysteme und Netzwerklösungen an. Sie haben die Aufgabe, die Vertriebsprozesse zu optimieren. Ein Interview mit einem Vertriebsmitarbeiter ergibt folgende Situation:

- Der Mitarbeiter erhält morgens per Post Kundenanfragen. Er besorgt sich die tagesaktuelle Preisliste aus einer vom Geschäftsführer gepflegten und werktäglich per Rundmail verschickten Exceldatei, die Kundenrabatte aus einem DOS-Programm zur Auftragsabwicklung und errechnet daraus einen Angebotspreis. Das Angebot erstellt er mit Word und sendet es dem Kunden.
- In der Post sind auch Aufträge enthalten. Der Mitarbeiter gibt sie in das DOS-Programm ein, schreibt eine Auftragsbestätigung mit Word, sendet sie an den Kunden und gibt eine Kopie an den Techniker, der den Computer zusammenbaut und ausliefert.
- Ein Onlineshop auf Basis von LAMP wird abends ausgewertet, indem der Mitarbeiter sich in die MySQL-Datenbank einloggt und eine Auswertung der neuen Aufträge ausdruckt. Er gibt sie in die DOS-Auftragsabwicklung ein. Anschließend pflegt er die aktuellen Preise aus der Exceldatei in die MySQL-Datenbank ein.



- 1. Was ist Modellierung?**
- 2. OOAD mit der UML**
- 3. Entity-Relationship-Modelle**
- 4. Geschäftsprozeß-modellierung**

IT als Wegbereiter für Geschäftsprozesse

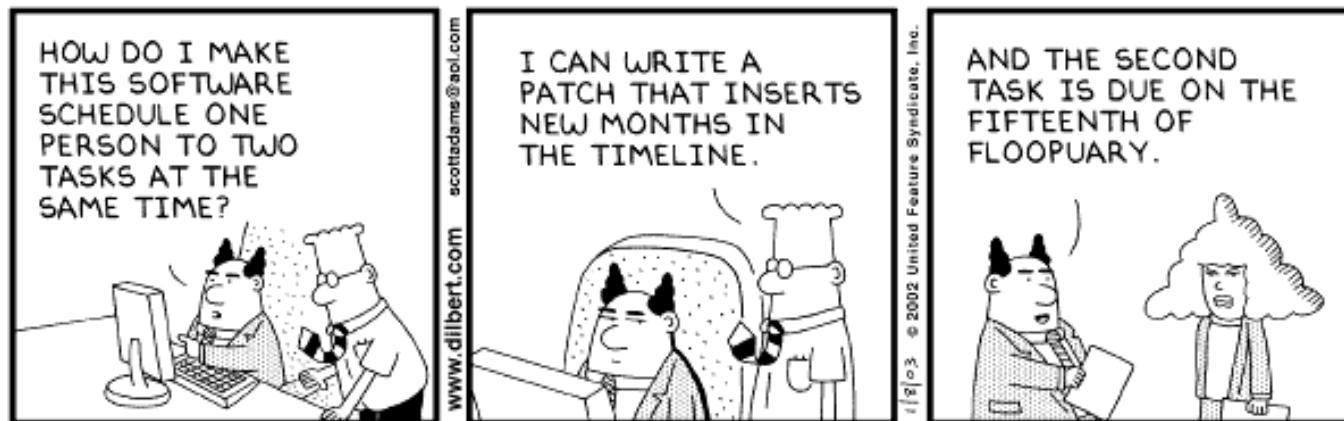
- Geographisch und zeitlich verteilte Zusammenarbeit wird erst möglich durch zentrale Systeme, auf die dezentral zugriffen werden kann.
- Referenzprozesse werden oft mit den Anwendungssystemen als Modell ausgeliefert.
- Auswertung der Prozeßdurchführung zwecks kontinuierlicher Verbesserung ist technisch möglich.
(Unterliegt in Deutschland rechtlichen Einschränkungen)
- Eliminierung manueller Schnittstellen mittels Integrationsebene (EAI)
- ...



IT als Werkzeug zur Realisierung von Geschäftsprozessen

1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeßmodellierung

- Projektmanagementsysteme
- CASE
- Modellierungswerkzeuge



Copyright © 2003 United Feature Syndicate, Inc.



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

IT als Hindernis für die Optimierung von Geschäftsprozessen

- Falsch verstandener Investitionsschutz
- IT als Selbstzweck
- Projektexogene Systemauswahl
 - IT-Standards sind grundsätzlich sehr sinnvoll!
 - Wenn sie für einzelne Projekte einen Nachteil bedeuten, muß der Projektleiter dies kommunizieren und dokumentieren.



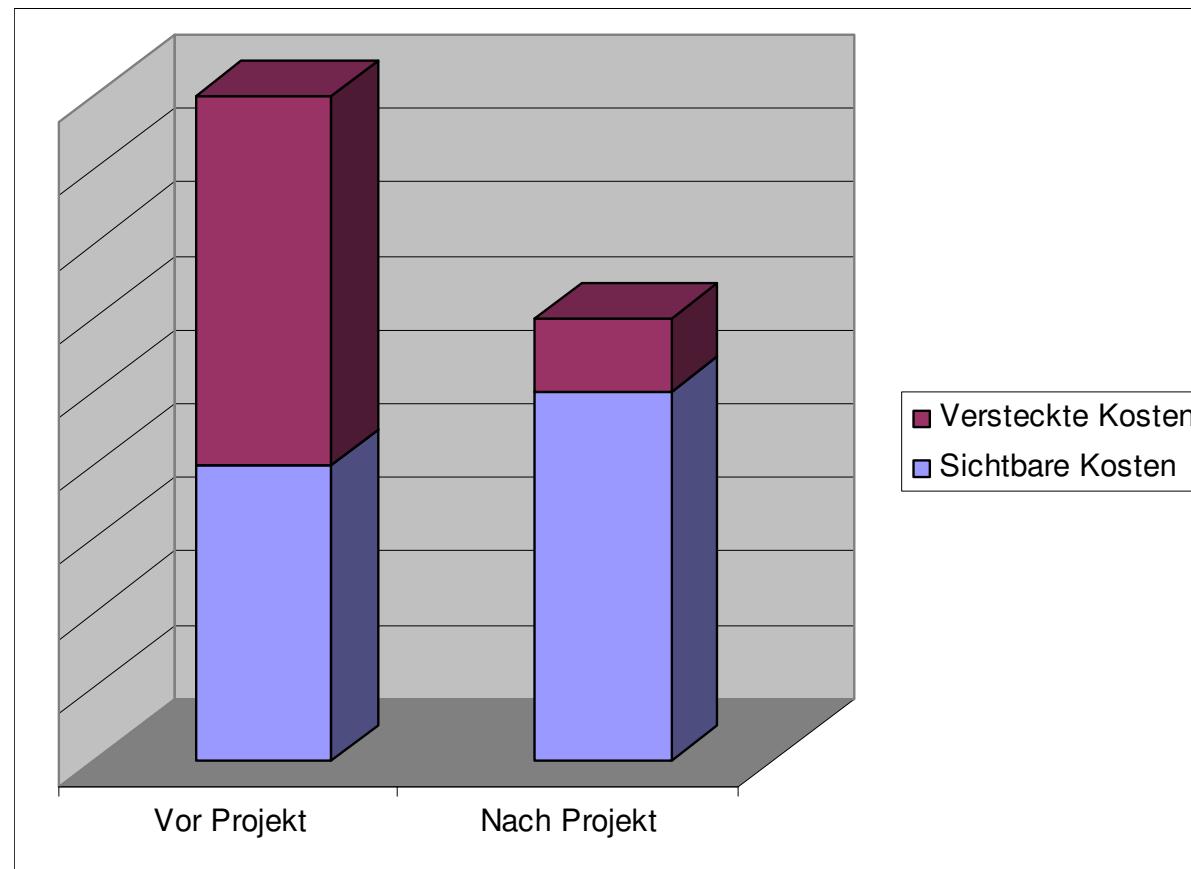
© Scott Adams, Inc./Dist. by UFS, Inc.



- 1. Was ist Modellierung?**
- 2. OOAD mit der UML**
- 3. Entity-Relationship-Modelle**
- 4. Geschäftsprozeßmodellierung**

Vorsicht Falle bei der Geschäftsprozeßoptimierung

Scheinbare Kostensteigerung

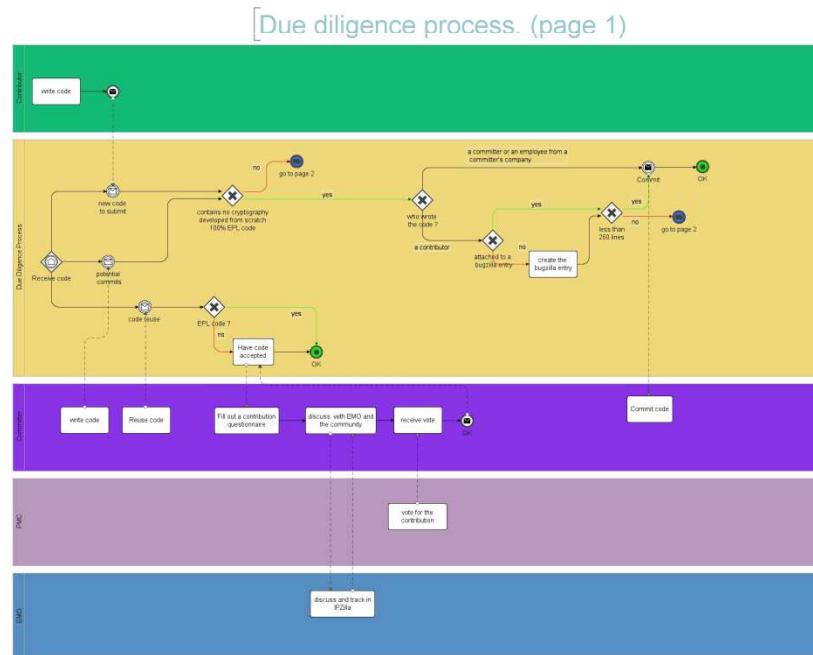




1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeßmodellierung

Geschäftsprozeßmodellierung

- Modellierungsmethode für Geschäftsprozesse:
Business Process Modelling Notation (BPMN)
- OMG-Standard
- Ähnlichkeit zu UML-Aktivitätendiagrammen



Quelle: eclipse.org



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeßmodellierung

Geschäftsprozeßmodellierung

- Ausführungssprache für Geschäftsprozesse:
Business Process Execution Language (BPEL)
- OASIS-Standard
- XML-basiert
- beschreibt, wie ein Prozeß sich aus Web-Services zusammensetzt
- Ausführbar mit Middleware-Plattformen wie SAP XI

```
<!-- Prepare the input for the Book Rating -->
<assign>
  <copy>
    <from variable="BookPurchase" part="book"/>
    <to variable="BookRatingRequest" part="book"/>
  </copy>
</assign>
<!-- Synchronously invoke the Book Rating Web Service -->
<invoke partnerLink="BookRating"
       portType="bkr:BookRatingPT" operation="BookRating"
       inputVariable="BookRatingRequest"
       outputVariable="BookRatingResponse" />
```

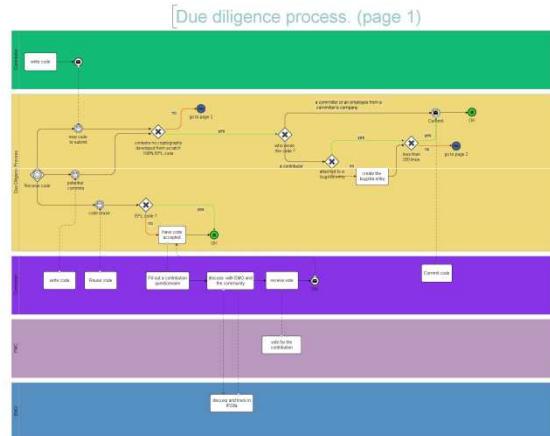
Quelle: oracle.com



- 1. Was ist Modellierung?**
- 2. OOAD mit der UML**
- 3. Entity-Relationship-Modelle**
- 4. Geschäftsprozeßmodellierung**

Geschäftsprozeßmodellierung

- Ziel: Roundtrip-Engineering mit automatischer Generierung von BPMN aus BPEL und umgekehrt. Noch sehr rudimentär.



```
<!-- Prepare the input for the Book Rating -->
<assign>
<copy>
<from variable="BookPurchase" part="book"/>
<to variable="BookRatingRequest" part="book"/>
</copy>
</assign>
<!-- Synchronously invoke the Book Rating Web Service --> <invoke
partnerLink="BookRating"
portType="bkr:BookRatingPT" operation="BookRating"
inputVariable="BookRatingRequest"
outputVariable="BookRatingResponse" />
```



- 1. Was ist Modellierung?**
- 2. OOAD mit der UML**
- 3. Entity-Relationship-Modelle**
- 4. Geschäftsprozeßmodellierung**

Geschäftsprozeßmodellierung

- Modellierungsmethode für Geschäftsprozesse: Ereignisgesteuerte Prozeßketten (EPK)
 - Entwickelt von A.-W. Scheer
 - Verbreitet u.a. wegen der Verwendung durch SAP
 - Bestandteile: Funktionen, Ereignisse, Organisationseinheiten, Informationsobjekte, Operatoren
 - Herkunft: BWL, Wirtschaftsinformatik (Im Gegensatz zu den UML-Aktivitätendiagrammen aus der Informatik)
 - Auch hier gilt wieder: Die Güte eines Modells mißt sich nicht nur an seiner formalen Korrektheit, sondern vor allem an seiner Zweckmäßigkeit. Für den selben Prozeß sind viele verschiedene Modelle denkbar. Für jeden konkreten Einsatzzweck sind sie unterschiedlich zweckmäßig.



- 1. Was ist Modellierung?**
- 2. OOAD mit der UML**
- 3. Entity-Relationship-Modelle**
- 4. Geschäftsprozeß-modellierung**

EPK

-

Funktionen

- Tätigkeiten, Verrichtungen, Vorgänge
- Der Umfang der Funktion wird vom Modellierer festgelegt (nach Zweckmäßigkeit)
- Beispiel: Auftragsdaten prüfen

Funktion



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeßmodellierung

EPK

- Ereignisse

- Sind Bedingungen für Funktionen oder Ergebnis von Funktionen
- Beispiel: Auftragsdaten sind eingetroffen
- Startereignis: Beginn eines Geschäftsprozesses
- Endereignis: Abschluß eines Geschäftsprozesses



- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

EPK

- Organisationseinheiten
 - Stellen oder Abteilungen, die für eine Funktion zuständig sind, werden ihr zugeordnet
 - Beispiel: Stelle "Auftragsbearbeitung Export Schweiz"

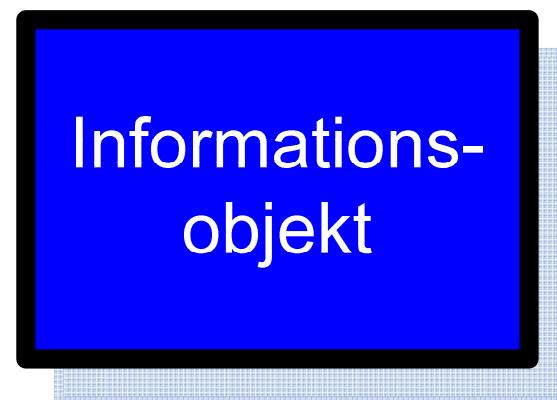




1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

EPK

- Informationsobjekte
 - Funktionen bearbeiten Daten. Je nachdem, ob sie die Daten lesen oder schreiben, geht ein Pfeil vom Informationsobjekt zur Funktion oder umgekehrt.
 - Beispiel: Kundenstammdaten





1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeßmodellierung

EPK

- Operatoren
 - Verbinden mehrere (mind. 2) Ereignisse oder mehrere (mind. 2) Funktionen miteinander
 - Zulässige Operatoren:
 - AND (alle verknüpften Ereignisse müssen eintreten bzw. alle verknüpften Funktionen müssen ausgeführt werden)
 - OR (mindestens eines der Ereignisse muß eintreten bzw. mindestens eine der Funktionen muß ausgeführt werden)
 - XOR (genau eines der Ereignisse muß eintreten bzw. genau eine der Funktionen muß ausgeführt werden)





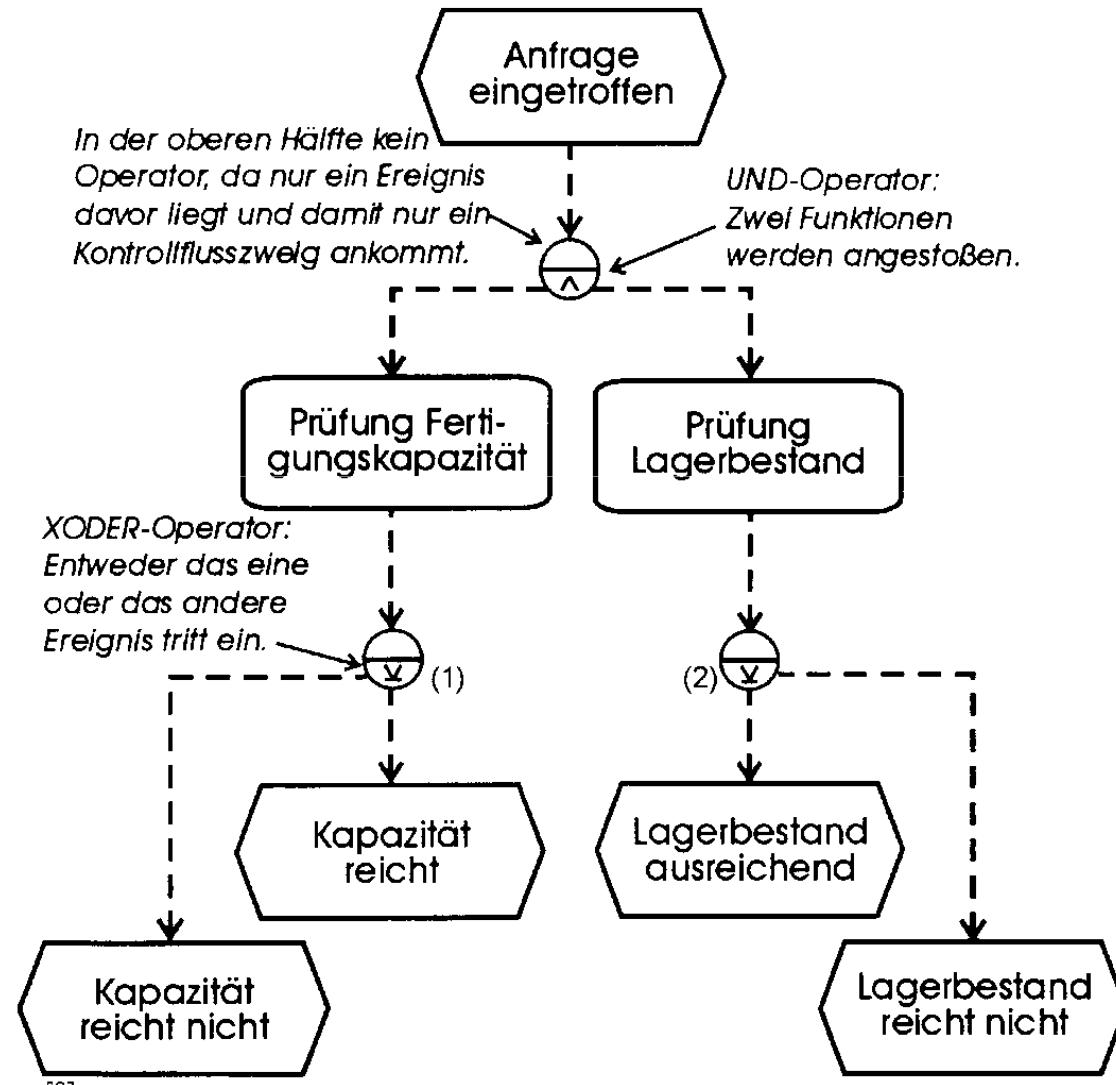
- 1. Was ist Modellierung?**
- 2. OOAD mit der UML**
- 3. Entity-Relationship-Modelle**
- 4. Geschäftsprozeß-modellierung**

EPK

- Operatoren
 - Werden mehrere Funktionen mit mehreren Ereignissen verbunden, so sind zwei Operatoren notwendig: Einer verknüpft die eingehenden Zweige, einer teilt die ausgehenden Zweige.

1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeßmodellierung

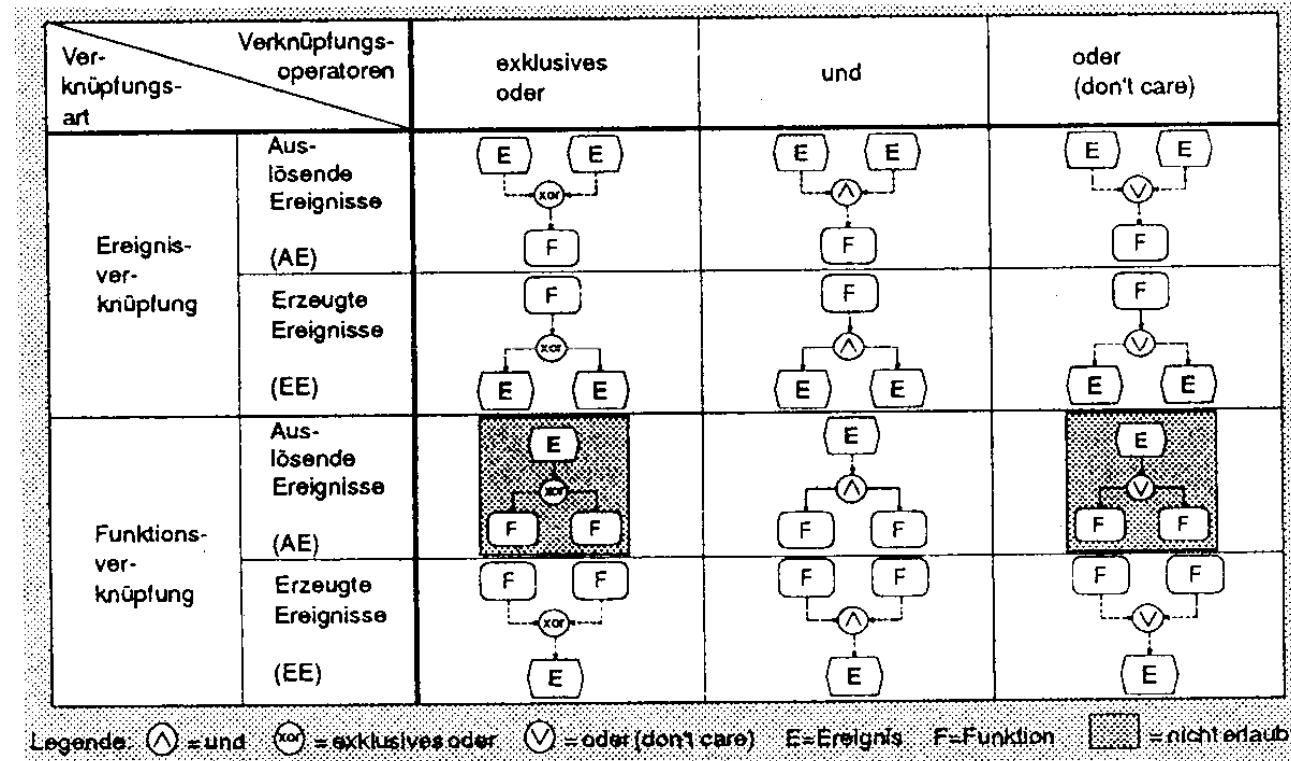
EPK



Quelle: [S], S. 71

EPK

1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeßmodellierung



Quelle: [S], S. 82

- 
- 1. Was ist Modellierung?**
 - 2. OOAD mit der UML**
 - 3. Entity-Relationship-Modelle**
 - 4. Geschäftsprozeß-modellierung**

		XODER	UND	ODER
Ereignis-verknüpfung: Ereignisse werden verknüpft.	Auslösende Ereignisse: Nachfolgende Funktion(en).			
	Die Ereignisse sind Bedingungen für die nachfolgend auszuführende(n) Funktion(en).	Genau eines der alternativen Ereignisse muss eintreten.	Alle Ereignisse müssen eintreten.	Mindestens eines der Ereignisse muss eintreten.
Funktions-verknüpfung: Funktionen werden verknüpft.	Erzeugte Ereignisse: Vorangehende Funktion(en).			
	Die Ereignisse signalisieren die Erledigung von Aufgaben („Ergebnisereignisse“)	Die Ereignisse geben die möglichen alternativen Ergebnisse an.	Die Ereignisse geben Teilaufgaben an.	Mindestens ein Ergebnis (Ergebnis) tritt ein
Funktions-verknüpfung: Funktionen werden verknüpft.	Auslösende Ereignisse: Nachfolgende Funktion(en).	Das Ereignis stößt genau eine Funktion an. VERBOTENE STRUKTUR.		Das Ereignis stößt mindestens eine Funktion an. VERBOTENE STRUKTUR.
	Das Ereignis stößt mehrere Funktionen an.	Das Ereignis stößt mehrere Funktionen an.		
	Erzeugte Ereignisse: Vorangehende Funktion(en).			
	Das Ereignis signalisiert die Beendigung mehrerer Funktionen.	Das Ereignis tritt ein, wenn genau eine der Funktionen durchgeführt wurde.	Das Ereignis tritt ein, wenn alle Funktionen durchgeführt wurden.	Das Ereignis tritt ein, wenn mindestens eine der Funktionen durchgeführt wurde.

Quelle: [S], S. 106

- 
1. Was ist Modellierung?
 2. OOAD mit der UML
 3. Entity-Relationship-Modelle
 4. Geschäftsprozeß-modellierung

EPK

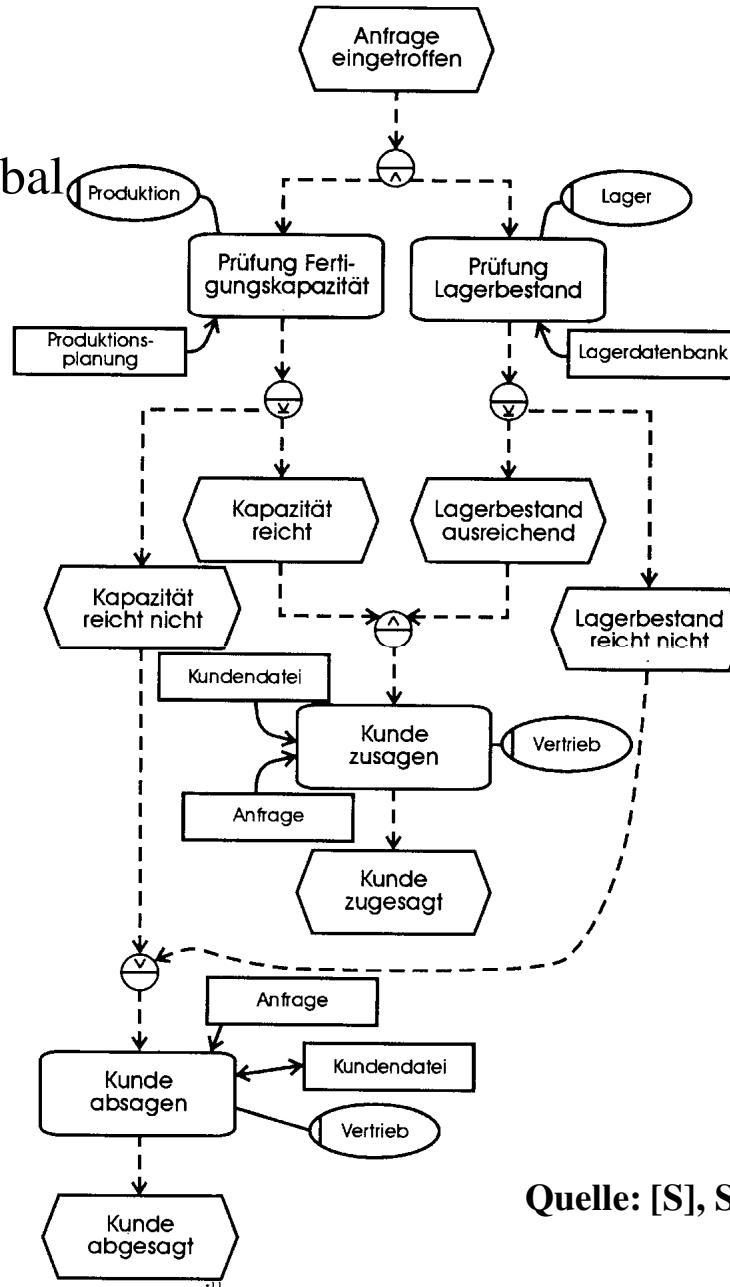
- Kontrollfluß
 - Gestrichelte Pfeile von oben nach unten
 - Ereignisse und Funktionen wechseln sich ab.
 - Der Geschäftsprozeß beginnt und schließt mit einem Ereignis.

1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeß-modellierung

Beispiel:

Beschreiben Sie verbal

diesen Prozeß.



Quelle: [S], S. 79



- 1. Was ist Modellierung?**
- 2. OOAD mit der UML**
- 3. Entity-Relationship-Modelle**
- 4. Geschäftsprozeßmodellierung**

EPK

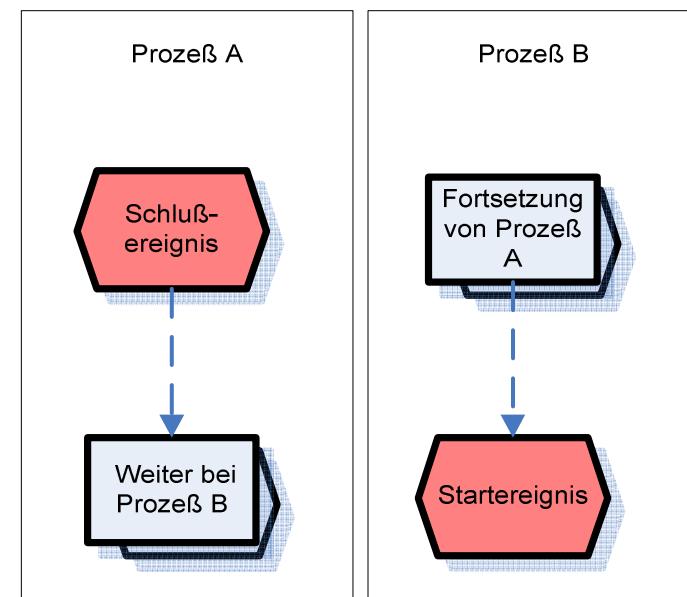
- Prozeßpfade
 - Große Geschäftsprozesse lassen sich nicht übersichtlich auf einem Bildschirm oder auf einem Blatt Papier darstellen.
Für wichtige Prozesse, die man im Projekt diskutiert, empfiehlt sich ein A1- oder A0-Plotter. Die Kombination von Informationsgehalt im Detail und Übersichtlichkeit in der Gesamtheit lässt sich durch nichts ersetzen.
Zweitbeste Lösung ist die Verwendung von Prozeßpfaden zur Aufteilung eines Schaubildes.
 - Häufig referenzierte "Prozeßbausteine" kann man wiederverwenden, indem man Prozeßpfade einsetzt.



1. Was ist Modellierung?
2. OOAD mit der UML
3. Entity-Relationship-Modelle
4. Geschäftsprozeßmodellierung

EPK

- Prozeßpfade
 - Der Prozeßpfad steht üblicherweise am Ende des aufrufenden und am Beginn des aufgerufenen Prozesses.





- 1. Was ist Modellierung?**
- 2. OOAD mit der UML**
- 3. Entity-Relationship-Modelle**
- 4. Geschäftsprozeß-modellierung**

Übungsaufgabe

Modellieren Sie einen Prozeß der Reklamationsbearbeitung eines Maschinenbauunternehmens als EPK. Interviews mit den beteiligten Mitarbeitern haben folgendes ergeben:

- Eingehende Reklamationen werden von der Poststelle an den Assistenten A des Vertriebsleiters V geschickt.
- A sortiert die Reklamationen in kaufmännische (z.B. betreffend die Lieferzeit, Beschwerden über Mitarbeiter) und technische (produktbezogene) Reklamationen.
- Kaufmännische Reklamationen gibt er an seinen Vorgesetzten V weiter. Dieser entscheidet, ob und in welcher Form er dem Kunden antwortet.
- Technische Reklamationen prüft er darauf, ob ein Garantiefall vorliegt. Falls ja, sendet er das Original an Sachbearbeiter S und eine Kopie an den jeweils zuständigen Produktmanager. Falls nein, geht das Original an den zuständigen Produktmanager. Dieser entscheidet, ob ein Kulanzfall vorliegt und sendet dem Kunden eine Antwort.