

Modellierung

Vorlesung

Magdeburg Research and Competence Cluster
Arbeitsgruppe Wirtschaftsinformatik

Fakultät für Informatik
Institut für Technische und Betriebliche Informationssysteme

Otto-von-Guericke-Universität Magdeburg

Prof. Dr. Klaus Turowski

klaus.turowski@ovgu.de
<https://mrcc.ovgu.de>

Einführung

Was stimmt hier nicht?

„Programmieren ihr noch oder
spezifiziert ihr schon?“

Typische Aufgabe

- Benötigt wird ein Programm das Daten über die Mitarbeiter, die in einem Unternehmen arbeiten, verwaltet
- Das Programm soll z.B. folgende Fragen beantworten können:
 - Wie viele Mitarbeiter arbeiten bei uns?
 - Wer arbeitet in welcher Abteilung?
 - Wer ist in Teilzeit beschäftigt?
 - Wer fertigt welches Produkt?
 - Wie hoch sind unsere monatlichen Personalkosten?
 - ...
- Als Anforderungen wurden u.a. genannt:

Ein Mitarbeiter mit Nachname, Vorname, Adresse und Gehalt wird identifiziert durch seine Mitarbeiternummer. Ein Produkt (mit Produktnummer und Name) wird von vielen, aber mindestens einem Mitarbeiter produziert und ein Mitarbeiter produziert viele Produkte. In einem Projekt, identifiziert durch die Projektnummer und gekennzeichnet mit einer Beschreibung, arbeiten viele Mitarbeiter in unterschiedlichen Rollen mit einer eindeutigen Rollenummer und einer Beschreibung und ein Mitarbeiter arbeitet in vielen Projekten. Ein Mitarbeiter arbeitet genau in einer Abteilung und in einer Abteilung arbeiten mehrere Mitarbeiter. Eine Abteilung wird gekennzeichnet durch eine Abteilungsnummer (eindeutig) und durch einen Abteilungsnamen. Eine Abteilung gehört genau zu einem (numerisch gekennzeichneten) Unternehmen; ein Unternehmen besteht aus mehreren Abteilungen. ...

Typische Fragen dazu:

- Welche Daten soll das Programm verwalten?
- Wie hängen die Daten zusammen?
- Welche Daten müssen erfasst werden?
- Wie sollen die Daten programmintern verwaltet werden?
- Wie wird sicher gestellt, dass die Daten nicht verloren gehen, wenn das Programm angehalten wird?
- ...

Warum Modellierung?

- Erklärung komplexer oder komplizierter Phänomene
 - Modell als abstrahierende Rekonstruktion der Realität

Beispiel: Warum erzeugt eine Ampelsteuerung Staus an einer Kreuzung?

- Gestaltung
 - Aufzeigen von Optionen zur (Um-)Gestaltung der Realwelt
 - Modell hat konstruktiven Charakter

Beispiel: Arbeitsteilige Entwicklung eines Web-Shops

Hinweis

- Was erwarten wir bei Software-Projekten als Dokumentation?
 - Anwendungsfälle (z.B. Use Case Diagramm)
 - Ablaufbeschreibung (z.B. Aktivitätsdiagramm)
 - Konzeptuelles Datenmodell (z.B. Klassendiagramm)
 - Architekturbeschreibung (z.B. Komponentendiagramm)
 - Schnittstellendokumentation

Modelle und Modellierung

- Definitionen
 - Ein Modell ist ein abstraktes, immaterielles Abbild realer Strukturen bzw. des realen Verhaltens für Zwecke des Subjekts.
 - Ein Modell ist:
 - adäquates
 - vereinfachendes
 - idealisierendes
 - Abbild der Realität
- Modellierung ist der kreative Prozess des Modellentwurfs

Begriffswelt

- Modellbildung
 - Informationen über ein Original erschaffen
- Information
 - „Aussagen, die den Erkenntnis– bzw. Wissensstand eines Subjektes [...] über ein Objekt [...] in einer gegebenen Situation und Umwelt [...] zur Erfüllung einer Aufgabe verbessern.“
[Szyperski 1980]
 - Kurz: Handlungsrelevantes Wissen

Modelldarstellung

- Ist Denken ohne Sprache möglich?
- Was ist eine Sprache?
- Modellierungssprachen
 - Mathematik
 - Programmiersprachen
 - (Semi-formale) grafische Sprachen (auch: Modellierungsmethoden)

Modellierung mit Entity-Relationship-Diagrammen Teil 1



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

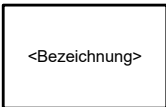
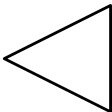

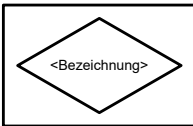



1. INTRODUCTION

The logical view of data has been an important issue in recent years. Three major data models have been proposed: the network model [2, 3, 7], the relational model [8], and the entity set model [25]. These models have their own strengths and weaknesses. The network model provides a more natural view of data by separating

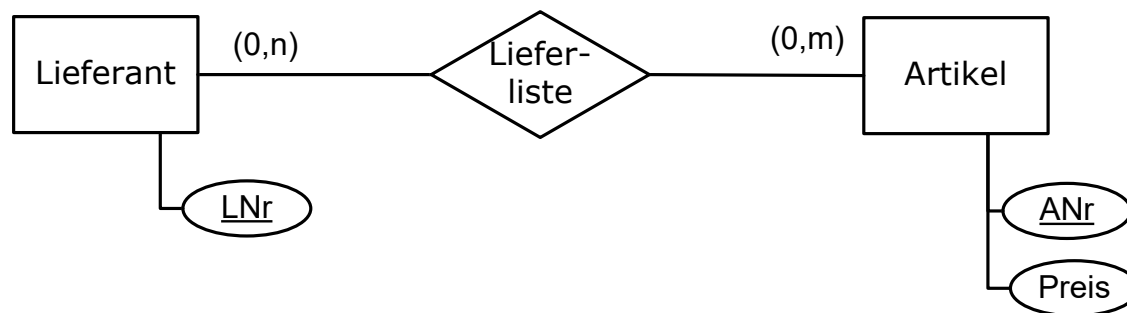
Eine einfache Modellierungssprache

- Das Entity–Relationship–Modell (ERM) [Chen 1976]
- Zugrundeliegende Annahme:
 - Die Realwelt besteht aus Objekten (entities/Entitäten) und Beziehungen (relationships) zwischen diesen

Grundelemente der Modellierungssprache

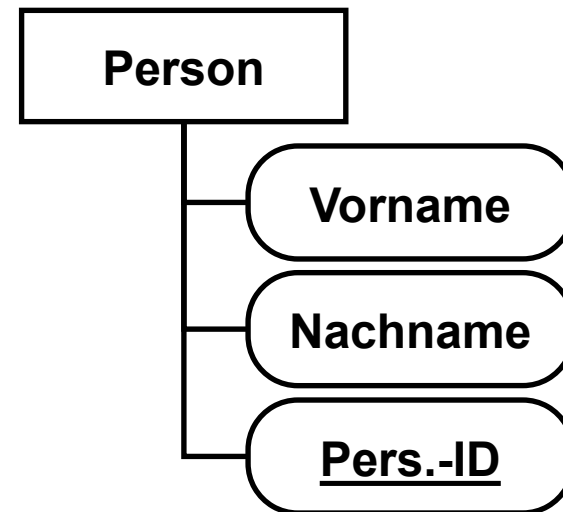
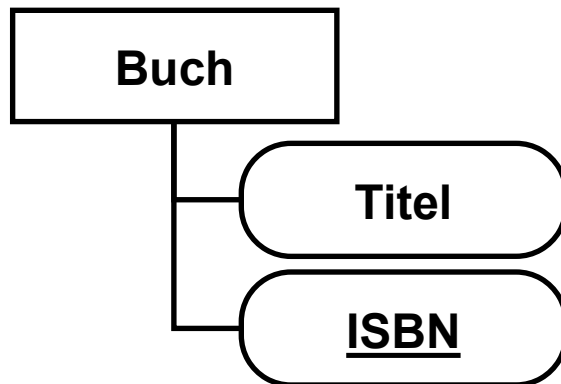
Bezeichnung	Symbol	Definition	Bezeichnung	Symbol	Definition
Entitytyp		Generalisierende Bezeichnung für eine Vielzahl von Betrachtungsobjekten, die gemeinsame Eigenschaften aufweisen	Generalisierung, Spezialisierung		Ähnliche Entity-Typen werden zu einem <i>übergreifenden</i> Entity-Typ (Supertyp) zusammengefasst
Relationshiptyp		Sinnvolle Verbindung oder Beziehung zwischen zwei oder mehreren Entity-Typen	Uminterpretierter Beziehungstyp		Die Uminterpretation von Beziehungstypen wird erforderlich, um Beziehungen zwischen Beziehungstypen darstellen zu können
Attribut		Relevante Eigenschaft eines Betrachtungsobjekts	Kante		Zeigt die Verbindung zwischen den Typen auf
Schlüsselattribut		Eigenschaft, die für jede Ausprägung des Entity-Typs einzigartig ist, d.h. eine Entity eindeutig identifiziert	Kardinalität in (min,max)-Notation	(0,1) (1,1) (0,n) (1,n)	Zeigt die minimale und maximale Beziehungsausprägung am Beziehungstyp

■ Beispiel:



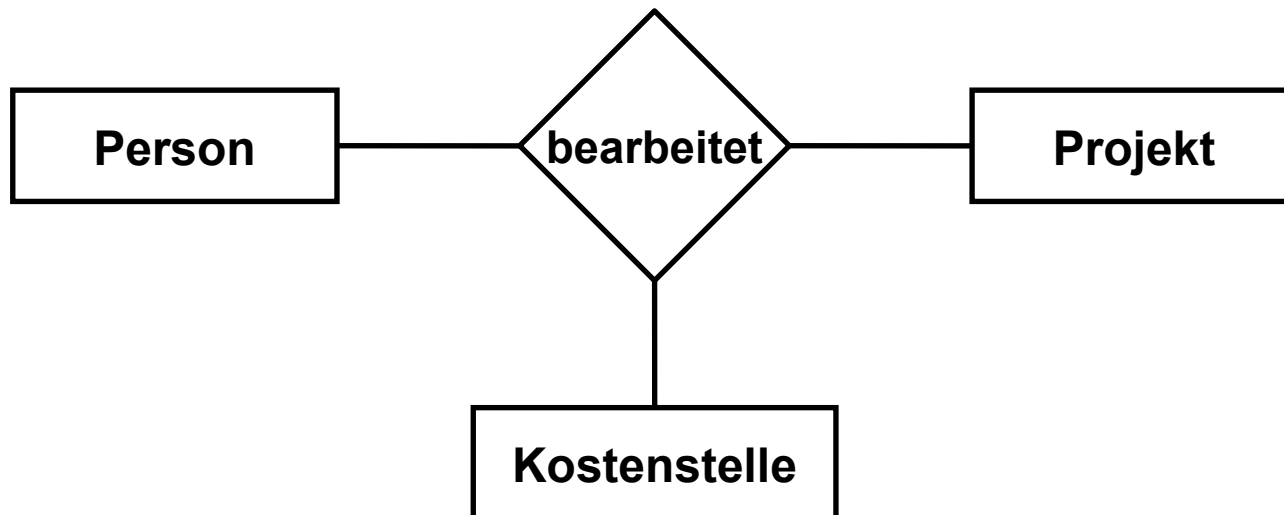
Erklärung: Schlüssel

- Schlüssel dienen zur eindeutigen Identifizierung einer Entität
- Beispiel: die ISBN eines Buches, die Personalausweisnummer einer Person
- Grafische Repräsentation:



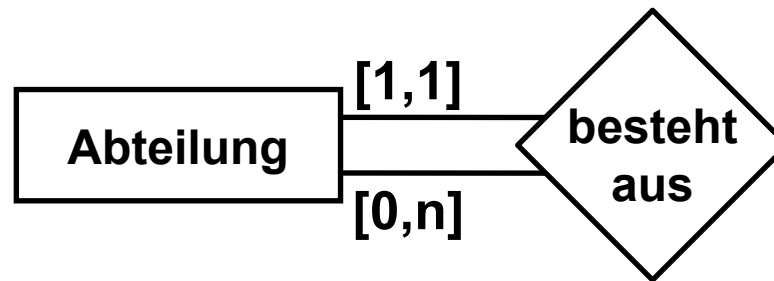
Komplexere Sachverhalte in der Modellierung I

- Eine Beziehung kann n Entitäten miteinander verknüpfen
- Beispiel: Ein Projekt wird von mehreren Mitarbeitern bearbeitet und durch mehrere Kostenstellen finanziert

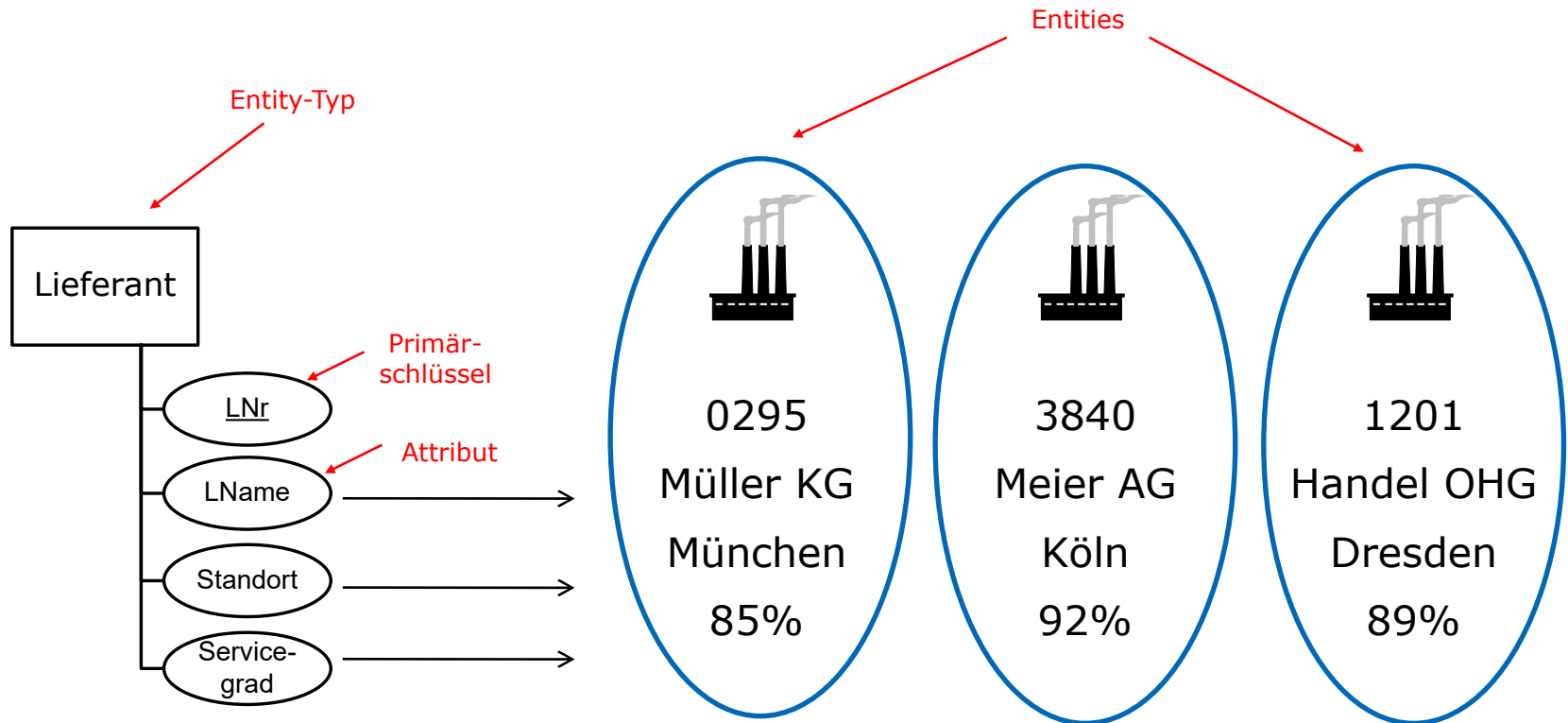


Komplexere Sachverhalte in der Modellierung II

- Abbildung von hierarchischen Strukturen, z. B. Organisationseinheiten
- Beispiel: Ein Unternehmen ist in mehreren Abteilungen strukturiert, die wiederum beliebig viele Unterabteilungen besitzen können.



Unterschied zwischen „Entity“ und „Entity-Typ“



„How to Find the Object“

- Konzeptuelle Modellierung

- Aussagen

- „Studenten besuchen eine Vorlesung“
 - „Vorlesungen werden von Dozenten gehalten “
 - „Vorlesungen finden in einem Raum statt“
 - „Ein Raum enthält Stühle, Tische und ein Whiteboard“

- Was sind hier die

- Entitätstypen,
 - Beziehungstypen und
 - Attribute?

Hinweis zu „konzeptuell“

- Von späteren Implementierungsentscheidungen abstrahierend
 - Ein konzeptuelles Modell kann zu verschiedenen Implementierungen führen

Konzeptionelle Modellierung und Umsetzung

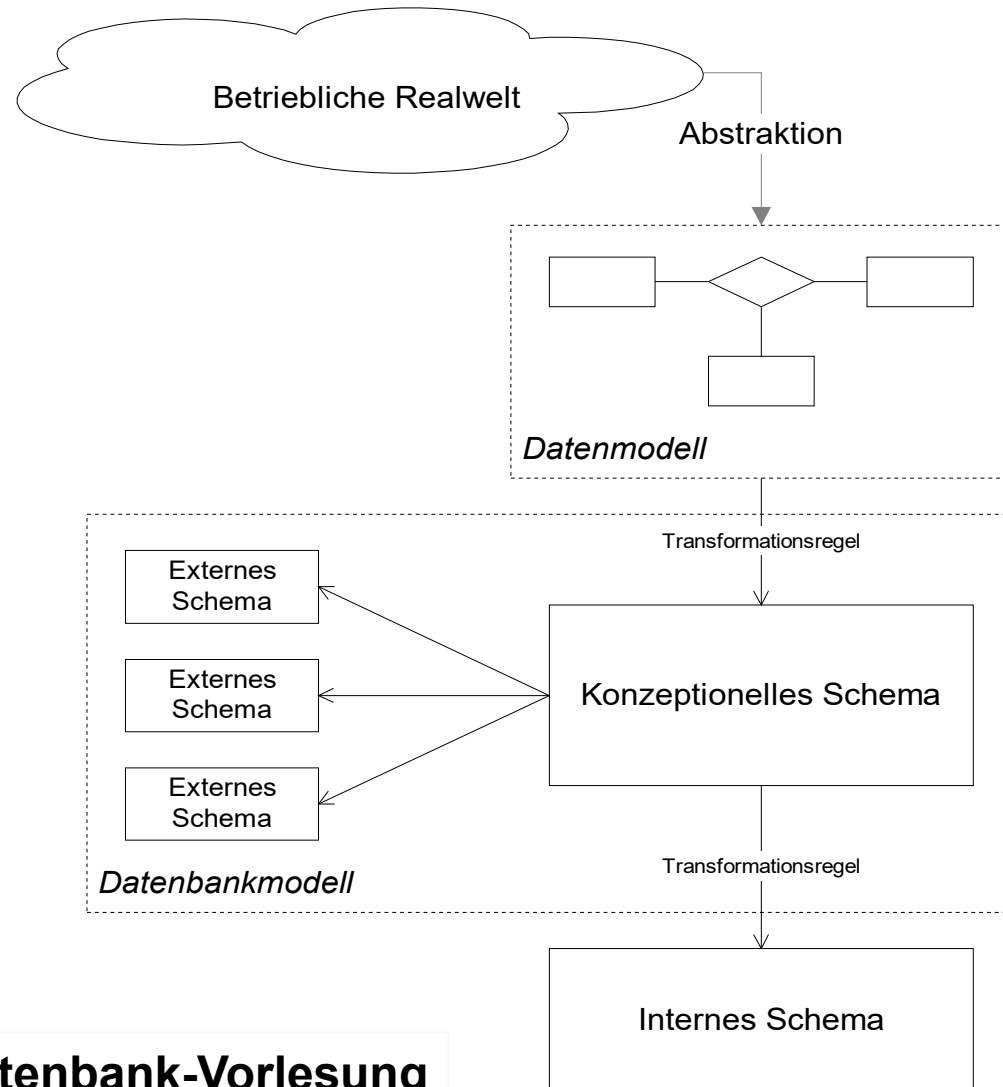
Implementierungsalternativen: Daten- und Datenbankmodelle

- Konzeptuelles/semantisches Datenmodell

- Beschreibung der fachlichen Anforderungen des in der Datenbank abzubildenden Realitätsausschnitts

- Datenbankmodell

- Konkrete, Datenbankmanagementsystem-bezogene Umsetzung des Datenmodells



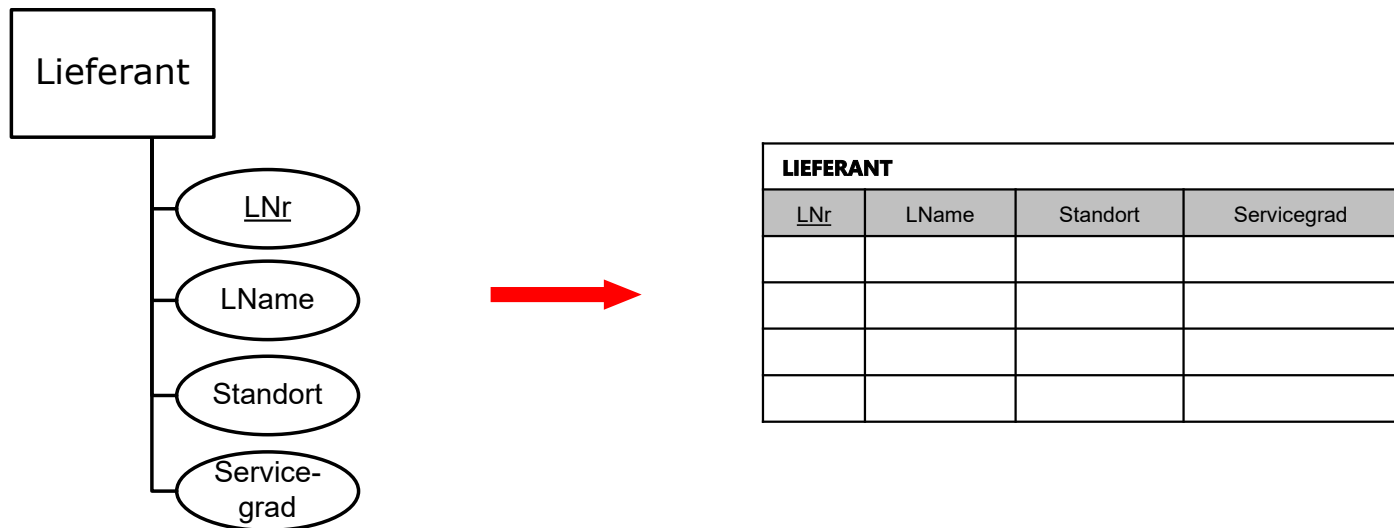
→ **Weiterführung: Datenbank-Vorlesung**

Implementierungsalternativen: Java Script

```
1  <!DOCTYPE HTML>
2  <html>
3    <head>
4      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
5    </head>
6    <body>
7      <script type="text/javascript">
8
9        var l1 = {Lnr: "1234", LName: "Müller KG", Standort: "Magdeburg", Servicegrad: "90%"};
10       var l2 = {Lnr: "5678", LName: "SASD GmbH", Standort: "Berlin", Servicegrad: "80%"};
11       var l3 = {Lnr: "1212", LName: "Tele 50000 AG", Standort: "Stendal", Servicegrad: "34%"};
12
13       var data = [l1, l2, l3];
14
15       console.log(data);
16
17     </script>
18   </body>
19 </html>
```

```
▼ Array[3] ⓘ
  ▼ 0: Object
    LName: "Müller KG"
    LNr: "1234"
    Servicegrad: "90%"
    Standort: "Magdeburg"
    ► __proto__: Object
  ▼ 1: Object
    LName: "SASD GmbH"
    LNr: "5678"
    Servicegrad: "80%"
    Standort: "Berlin"
    ► __proto__: Object
  ▼ 2: Object
    LName: "Tele 50000 AG"
    LNr: "1212"
    Servicegrad: "34%"
    Standort: "Stendal"
    ► __proto__: Object
    length: 3
    ► __proto__: Array[0]
```

Implementierungsalternativen: Transformation des Entity-Typs



→ **Weiterführung: Datenbank-Vorlesung**

ER-Diagramme und relationale Datenbanken

- Modellierung ist kein Selbstzweck
- ER-Diagramme beschreiben die konzeptuelle Sicht auf Daten
- Konzeptuelle Sicht muss auf technische Implementierung (Datenbanktabellen) übertragen werden

1:n Beziehungen

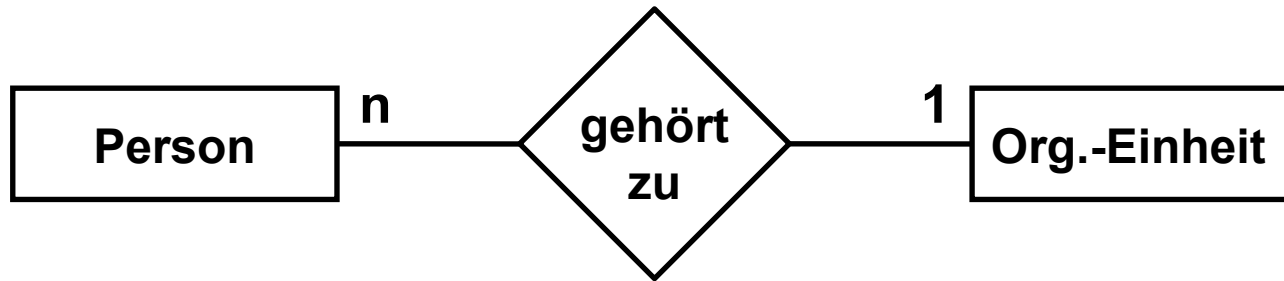


Tabelle: Person

<u>PkPersId</u>	Vorname	Nachname	FkOrgId
001	Claus	Rautenstrauch	12345
002	Jubran	Rajub	12345
007	James	Blond	42235
...

Tabelle: OrgEinheit

<u>PkOrgId</u>	Name	Standort
12345	AG WI	Magdeburg
42235	MI 5	London
...

n:m Beziehungen

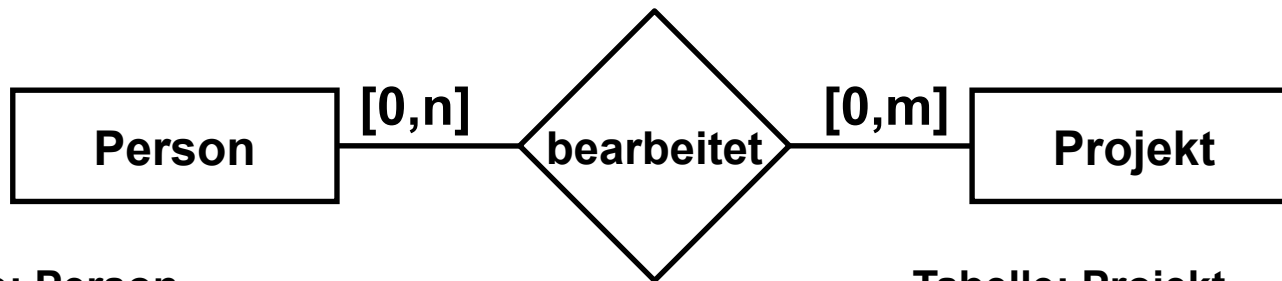


Tabelle: Person

<u>PkPersId</u>	Vorname	Nachname
001	Claus	Rautenstrauch
002	Jubran	Rajub
007	James	Blond
...

Tabelle: Projekt

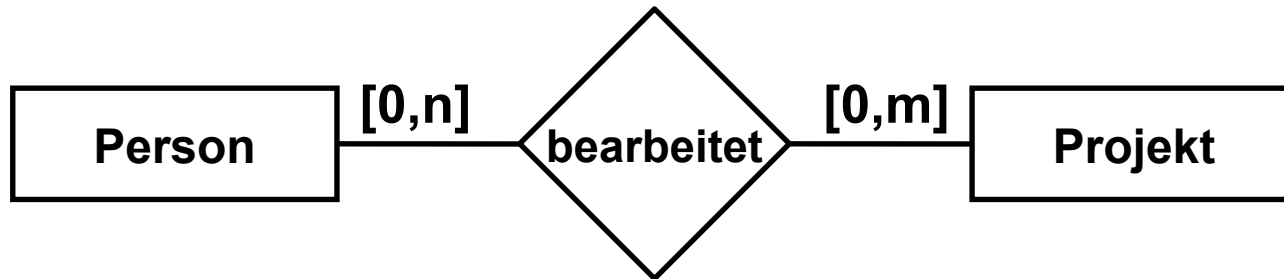
<u>PkProjektId</u>	Name
3423	EVL
1231	iSeal
5678	DoNothing

Tabelle: Projektmitarbeiter

<u>FkPersId</u>	<u>FkProjektId</u>
001	3423
001	1231
002	1231
...	...

Eine Person bearbeitet mehrere Projekte.

n:m Beziehungen



<u>PkPersId</u>	Vorname	Nachname
001	Claus	Rautenstrauch
002	Jubran	Rajub
007	James	Blond
...

<u>PkProjektId</u>	Name
3423	EVL
1231	iSeal
5678	DoNothing

Tabelle: Projektmitarbeiter

<u>FkPersId</u>	<u>FkProjektId</u>
001	3423
001	1231
002	1231
...	...

Mehrere Personen bearbeiten ein Projekt.

n:m Beziehungen

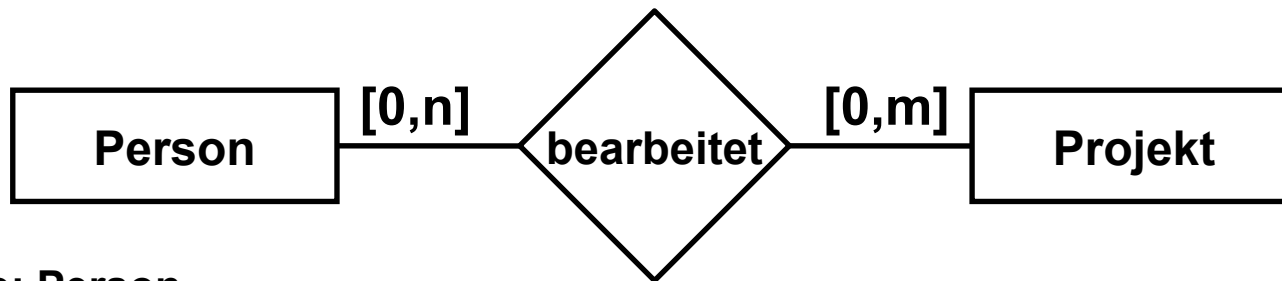


Tabelle: Person

<u>PkPersId</u>	Vorname	Nachname
001	Claus	Rautenstrauch
002	Jubran	Rajub
007	James	Blond
...

<u>PkProjektId</u>	Name
3423	EVL
1231	iSeal
5678	DoNothing

Tabelle: Projektmitarbeiter

<u>FkPersId</u>	<u>FkProjektId</u>
001	3423
001	1231
002	1231
...	...

Es gibt Personen,
die kein Projekt bearbeiten.

Es gibt Projekte, die von
niemandem bearbeitet werden.

Implementierungsalternativen: Einbetten von SQL in Java

```
import java.sql.*;

public class SQLDemo{
    public static void main(String args[]){
        //Treiber laden
        Class.forName("org.gjt.mm.mysql.Driver");

        //Verbindungsaufbau
        Connection conn = DriverManager.getConnection(
            jdbc:mysql://localhost/meineDatenbank", "benutzername", "passwort");

        //Formulierung der Anfrage
        String sql = "select * from Lieferant";

        //Anfrage erstellen
        Statement stm = conn.createStatement();

        //Anfrage ausführen
        ResultSet rs = stm.executeQuery(sql);

        //solange Ergebnistupel vorhanden sind hole nächstes Tupel
        while(rs.next()) {
            System.out.println(rs.getString("Nachname"));
        }
        //Anfrage schließen & Verbindung abbauen
        stm.close();
        conn.close();
    }
}
```

Implementierungsalternative XML und XML als Meta- Auszeichnungssprache

Implementierungsalternativen: XML

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <Lieferanten>
4    <Lieferant>
5      <LNr>1234</LNr>
6      <LName>Müller KG</LName>
7      <Standort>Magdeburg</Standort>
8      <Servicegrad>90%</Servicegrad>
9    </Lieferant>
10   <Lieferant>
11     <LNr>5678</LNr>
12     <LName>SASD GmbH</LName>
13     <Standort>Berlin</Standort>
14     <Servicegrad>80%</Servicegrad>
15   </Lieferant>
16   <Lieferant>
17     <LNr>1212</LNr>
18     <LName>Tele 50000 AG</LName>
19     <Standort>Stendal</Standort>
20     <Servicegrad>34%</Servicegrad>
21   </Lieferant>
22 </Lieferanten>
```

Vom ER-Diagramm zum Datenbankmodell

- Aus dem Fachkonzept wird das DV-Konzept abgeleitet
 - Fachkonzept: Semantische Datenmodelle (ER-Diagramm)
 - DV-Konzept: Relationales Modell, Hierarchisches Modell, Dokumentenmodell ...

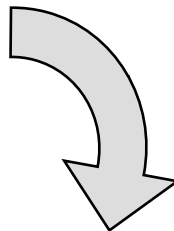
- Das Datenmodell wird in ein spezielles Datenbankmodell überführt



- Überführung/Repräsentation als Tabelle
 - relationales Datenbankmanagementsystem (s.o)
- Überführung/Repräsentation als Dokument
 - XML

Lieferant	
L1	Fa. X
L2	Fa. Y
L3	Fa. Z

Artikel	
A1	Holz
A2	Stahl
A3	Kalk

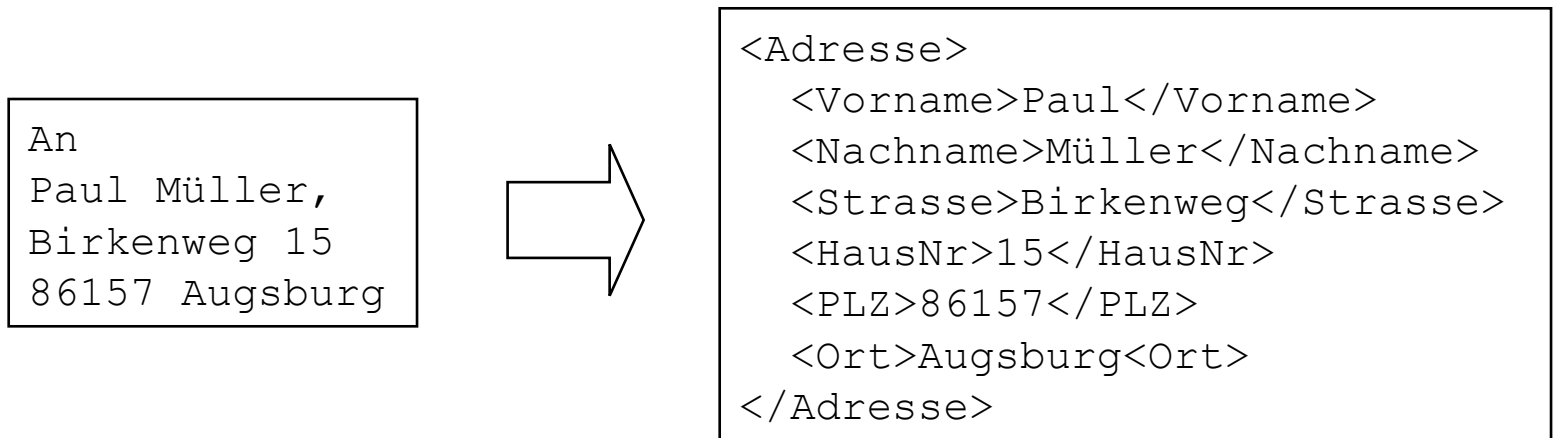


liefert	
L1	A2
L3	A2
L3	A3

Auszeichnungssprachen

■ Idee von Auszeichnungssprachen

- ▶ Auszeichnung von Daten durch zusätzlichen Text (Markup) in einem Dokument
- ▶ Meta-Informationen werden ebenfalls übertragen und ausgewertet



■ Markup enthält Informationen über das Dokument

- ▶ Dokumentstruktur
- ▶ Dokumentinhalt
- ▶ Dokumentverarbeitung

Struktur	<code><titel>Neuer Vorstand bei der Trinktechnik AG</titel></code> <code><absatz>Hans Müller ist neuer Vorstand bei der Trinktechnik AG</absatz></code>
Inhalt	Neuer Vorstand bei der <code><firma>Trinktechnik AG</firma></code>
Verarbeitung	<code><fett>Neuer Vorstand bei der Trinktechnik AG</fett></code>

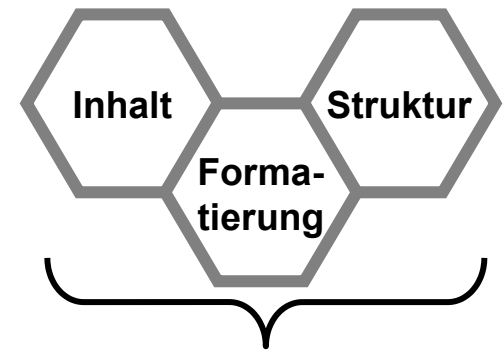
Extensible Markup Language (XML)

- XML ist eine Meta-Auszeichnungssprache, mit der verschiedenste Markup-Sprachen definiert werden können
- Es existieren Regeln zur Bildung von XML-Dokumenten
 - ▶ XML-Syntaxregeln führen zu wohlgeformten Dokumenten
 - ▶ XML-Grammatiken führen zu validen Dokumenten
- XML ist für den Einsatz im WWW angepasst
 - ▶ Plattformunabhängig
 - ▶ Maschinenlesbar
- XML ermöglicht
 - ▶ Beschreibung, Austausch, Darstellung und Manipulation von strukturierten Daten

XML – Technologien

■ Zusammenspiel verschiedener Technologien ermöglicht Trennung von Inhalt, Struktur und Formatierung

- ▶ Inhalt: XML-Tags
- ▶ Struktur: Document Type Definitions (DTD) , XML Stylesheets (XSD)
- ▶ Formatierung: Extensible Style Sheet Language (XSL), Cascading Style Sheets (CSS)



person-syntax.dtd

```
<!ELEMENT person (nachname, vorname)>
<!ELEMENT nachname (#PCDATA)>
<!ELEMENT vorname (#PCDATA)>
```

person-darstellung.css

```
nachname
{
  font-size: 20pt;
  color: #cccccc;
}
```

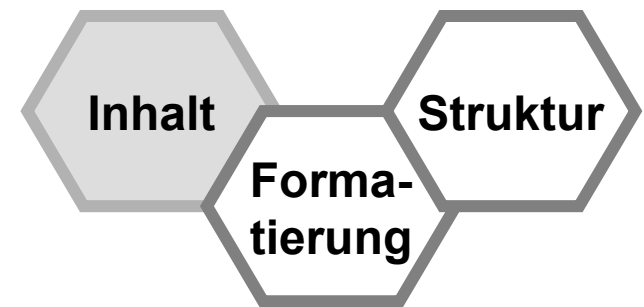
person.xml

```
<?xml version="1.0"?>
<person>
  <nachname>Meier</nachname>
  <vorname>Hans</vorname>
</person>
```

Präsentation

Meier Hans

XML – Inhalt

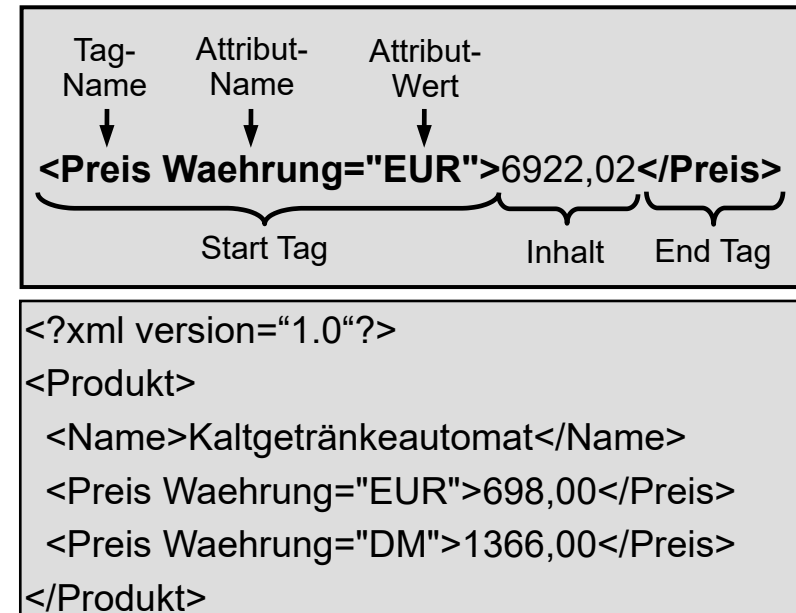


■ XML-Tags

- ▶ Können beliebig, individuellen Zielen entsprechend, benannt werden

■ XML-Syntaxregeln

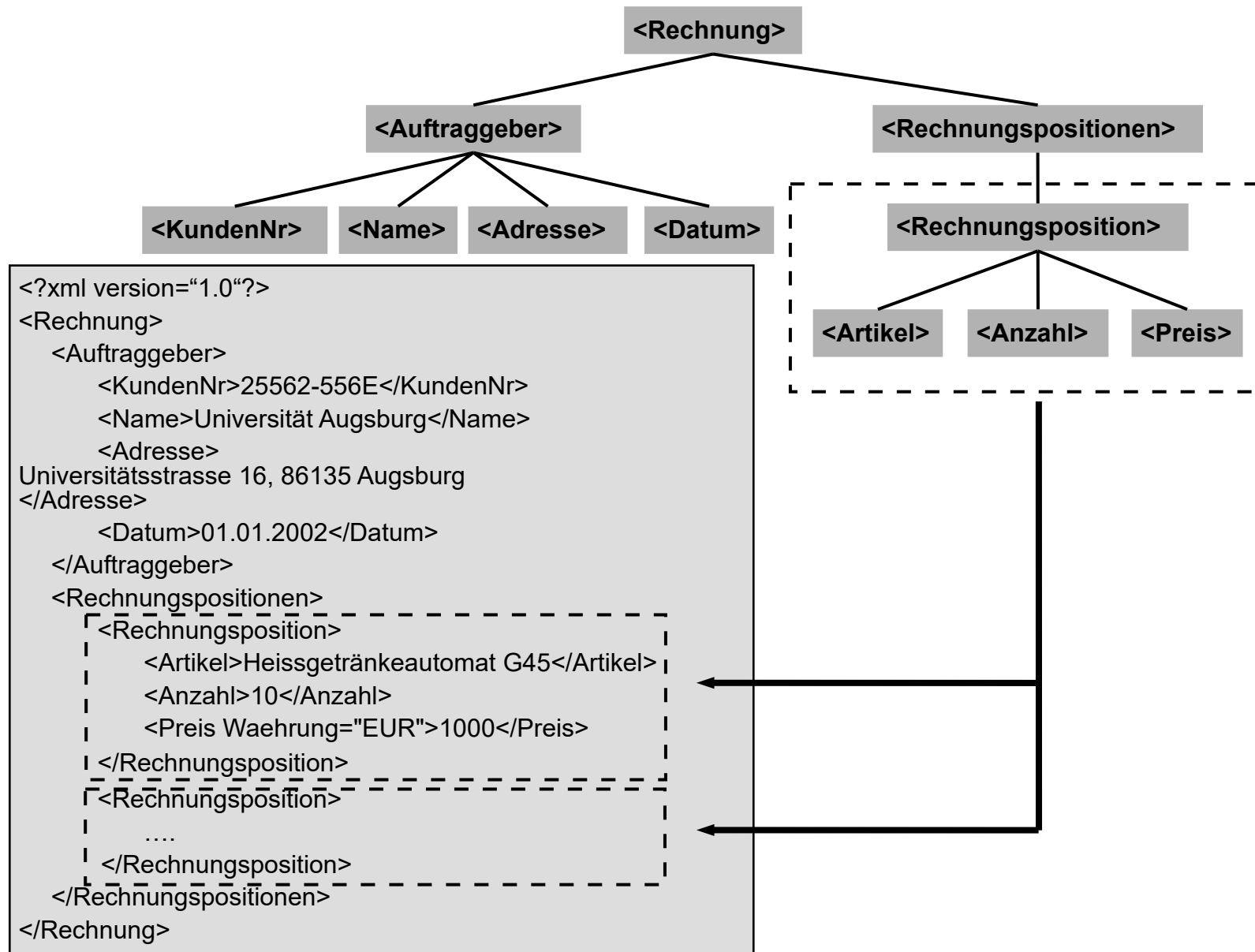
- ▶ XML-Deklaration am Anfang
 - ▶ XML-Dokument muss ein Wurzelement beinhalten, das alle anderen Elemente umfasst
 - ▶ Elemente müssen korrekt geschachtelt sein
 - ▶ Tags müssen immer geschlossen werden
 - ▶ Attributwerte immer in Anführungszeichen
 - ▶ Groß-/Kleinschreibung wird unterschieden
- ➔ Entspricht ein XML-Dokument den Regeln, wird es als wohlgeformt bezeichnet



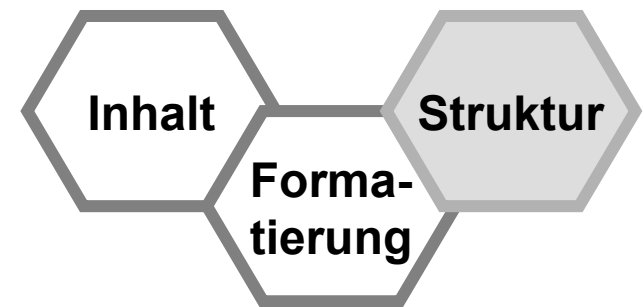
XML – Beispieldokument

```
<?xml version="1.0"?>
<Kunde>
  <Kund-Nr>3416484</Kund-Nr>
  <Name>Bayer</Name>
  <Adresse>
    <Str>Waldweg</Str>
    <Nr>12</Nr>
    <PLZ>39106</PLZ>
    <Ort>Magdeburg</Ort>
  </Adresse>
  <Status>3</Status>
</Kunde>
```

XML – Hierarchischer Aufbau



XML – Struktur



■ Individuelle Auszeichnungssprache

- ▶ Angabe einer formalen Grammatik, um nur syntaktisch korrekte Dokumente zuzulassen
- ▶ Zwei Arten der Definition in der Praxis
 - ▶▶ Document Type Definition
 - ▶▶ XML-Schema Description Language
- ▶ Prüfung der Konformität eines Dokuments durch XML-Parser
 - ▶▶ Ist das Dokument wohlgeformt?
 - ▶▶ Entspricht es den Regeln der Grammatik (ist es gültig/valide)?

■ Allgemeine Definition einer Grammatik mit der erweiterten Backus-Naur-Form (EBNF)

Rechnung	:= Auftraggeber Rechnungspositionen
Auftraggeber	:= KundenNr Name Adresse Datum
Rechnungspositionen	:= Position+ Gesamtpreis
Position	:= Artikel Anzahl Preis

AB	A gefolgt von B
A B	A oder B
[A]	nichts oder A
{ A } bzw. A *	nichts, A oder beliebige Wiederholung von A
A+	A oder beliebige Wiederholung von A
(A B) C	AC oder BC

XML – Document Type Definition (DTD)

■ Struktur

- ▶ `<!ELEMENT name model>`
- ▶ Name: Name/n des Elements/mehrerer Elemente
- ▶ Model: Erlaubter Inhalt für das Element

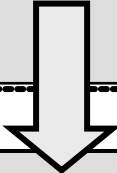
■ Syntax des Inhaltsmodells

a	a muss genau einmal auftreten	a, b	b folgt auf a
a?	a kann einmal oder keinmal auftreten	a b	entweder a oder b
a+	a muss einmal und kann mehrmals auftreten	(a, b) (b, a)	a folgt auf b oder umgekehrt
a*	a kann keinmal, einmal oder mehrmals auftreten	#PCDATA	Zeichenkette

```
<!ELEMENT Rechnung      (Auftraggeber,Rechnungspositionen)>
<!ELEMENT Auftraggeber   (KundenNr,Name,Adresse,Datum)>
<!ELEMENT Rechnungspositionen (Position+,Gesamtpreis)>
<!ELEMENT Position        (Artikel,Anzahl,Preis)>
<!ELEMENT KundenNr        (#PCDATA)>
<!ELEMENT Preis            (#PCDATA)>
<!ATTLIST Preis Waehrung CDATA>
```


DTD – Beispiel

```
<?xml version="1.0"?>
<Rechnung>
  <Auftraggeber>
    <KundenNr>25562-556E</KundenNr>
    <Name>Universität Augsburg</Name>
    <Adresse>Universitätsstrasse 16, 86135 Augsburg</Adresse>
    <Datum>01.01.2002</Datum>
  </Auftraggeber>
  <Rechnungspositionen>
    <Position>
      <Artikel>Heissgetränkeautomat G45</Artikel>
      <Anzahl>10</Anzahl>
      <Preis Waehrung="EUR">1000</Preis>
    </Position>
    <Gesamtpreis Waehrung="EUR">10000</Gesamtpreis>
  </Rechnungspositionen>
</Rechnung>
```




```
<!ELEMENT Rechnung      (Auftraggeber,Rechnungspositionen)>
<!ELEMENT Auftraggeber  (KundenNr,Name,Adresse,Datum)>
<!ELEMENT Rechnungspositionen  (Position+,Gesamtpreis)>
<!ELEMENT Position      (Artikel,Anzahl,Preis)>
<!ELEMENT KundenNr      (#PCDATA)>
<!ELEMENT Preis          (#PCDATA)>
<!ATTLIST Preis Waehrung CDATA>
<!ELEMENT Gesamtpreis   (#PCDATA)>
<!ATTLIST Gesamtpreis Waehrung CDATA>
```

XML – Schema Description Language (XSD)

■ Struktur

- ▶ Ist ein XML-Dokument
- ▶ Schema ist immer Wurzelement
- ▶ Verwendung von Namensräumen
 - ▶▶ Für das Schema selbst
 - ▶▶ Für das Dokument



```
<?xml version="1.0" encoding="ISO-8859-1" ?>  
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">  
<xs:element name="Artikel" type="Artikeltyp"/>  
...
```

■ Vorteile

- ▶ Schemata sind in XML geschrieben und können mit den selben Werkzeugen wie XML-Dokumente eingelesen werden
- ▶ Erweiterte Typdefinitionen für Elemente möglich (beliebige Strukturen)
- ▶ Angabe von Wertebereichen möglich

■ Nachteile

- ▶ Sehr umfangreich und komplex!

XML-Schema – Aufbau

■ Einfache Datentypen

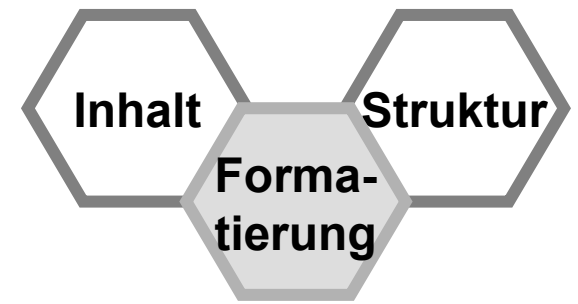
- ▶ `<xs:element name="Bezeichnung" type="xs:string"/>`
 - ▶▶ Weitere Datentypen: `xs:decimal`, `xs:integer`, `xs:boolean`, `xs:date`, `xs:time`
- ▶ `<xs:attribute name="id" type="xs:integer" use="required"/>`
 - ▶▶ `default=".."` legt einen Standardwert fest
 - ▶▶ `fixed=".."` überschreibt den Wert in der Datei

■ Komplexe Datentypen

- ▶ `<xs:sequence>...</xs:sequence>`
 - ▶▶ Mehrere Elemente in einer festen Reihenfolge

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Artikel" type="Artikeltyp"/>
  <xs:complexType name="Artikeltyp">
    <xs:sequence>
      <xs:element name="Bezeichnung" type="xs:string"/>
      <xs:element name="Preis" type="xs:decimal"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:integer" use="required"/>
  </xs:complexType>
</xs:schema>
```

XML – Formatierung



■ Extensible Style Sheet Language (XSL)

- ▶ Formatvorlagen für XML-Dokumente
- ▶ Speicherung der *Darstellungsinformationen*
- ▶ Verschiedene Präsentationsformen durch Austausch der Formatvorlage

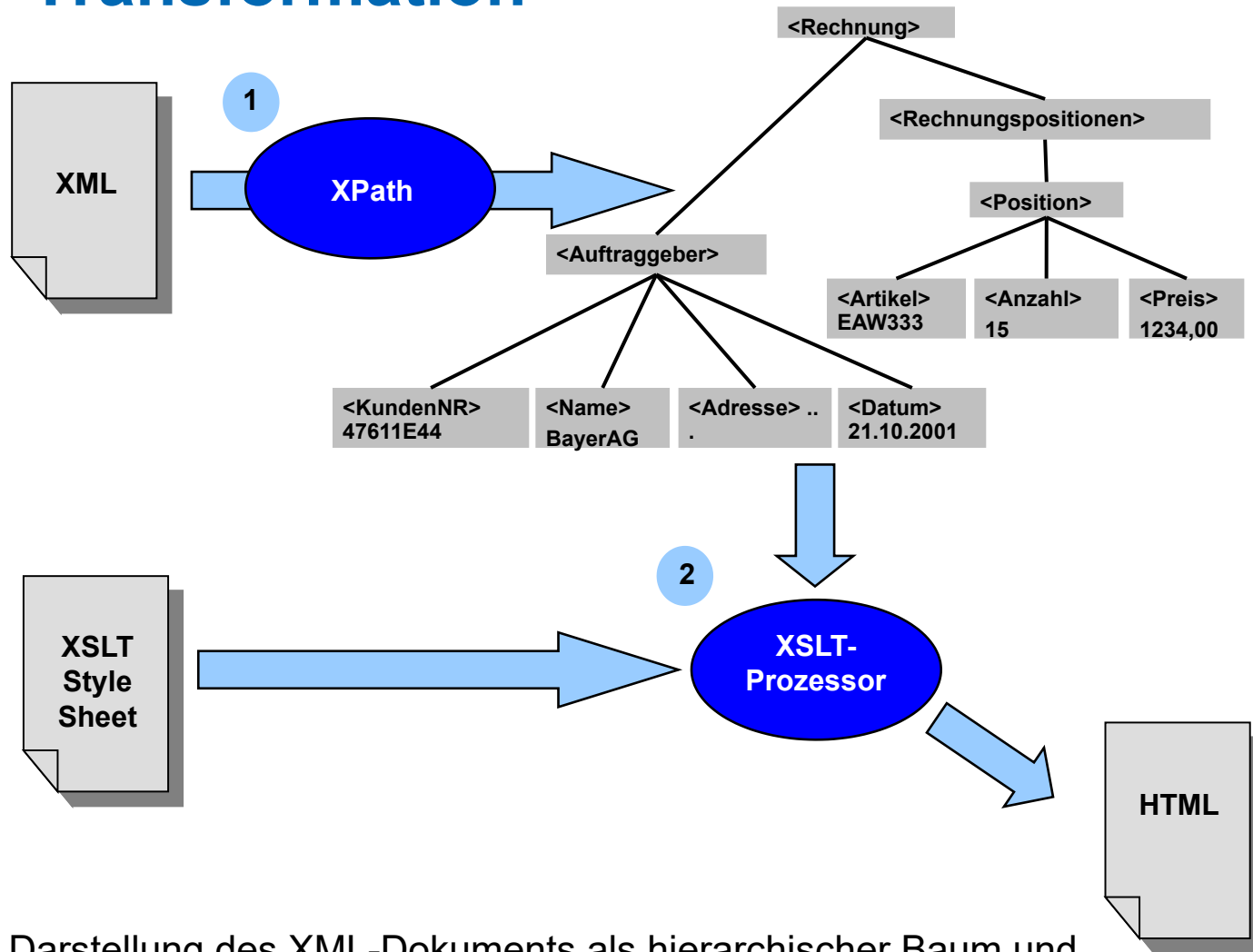
■ Formatierungsmöglichkeiten

- ▶ Hinzufügen von statischem Text (z.B. HTML-Tags)
- ▶ Umordnen, Duplizieren, Sortieren und bedingte Ausgabe von Inhalten
- ▶ Berechnung neuer Inhalte aus alten

■ XSL verwendet XPath (XML Path Language)

- ▶ Addressierung und Selektion von Inhaltsteilen durch Angabe des Pfades im XML-Baum
- ▶ Auch als Abfragesprache verwendbar
- ▶ XSL Transformation (XSLT) konvertiert XML-Dokument gemäß der XSL-Formatvorlage

XML – Transformation



- 1 Darstellung des XML-Dokuments als hierarchischer Baum und Auswahl der gewünschten Elemente (z.B.: /Position/Anzahl)
- 2 Transformation in HTML gemäß Formatvorlage

XSL – Transformation (XSLT)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="Rechnungen.xsl"?>
```

```
<Rechnung>
  <Rechnungskopf>
    <Name>Maier</Name>
    <Vorname>Wilhelm</Vorname>
  </Rechnungskopf>
  <Rechnungspositionen>
    <Position id="1">
      <Artikel>Gummibären</Artikel>
      <Anzahl>200</Anzahl>
      <Preis>1,50</Preis>
      <Zeilensumme>300,00</Zeilensumme>
    </Position>
    <Position id="2">
      <Artikel>Kaugummi</Artikel>
      <Anzahl>15</Anzahl>
      <Preis>0,90</Preis>
      <Zeilensumme>13,50</Zeilensumme>
    </Position>
    <Gesamtsumme>313,50</Gesamtsumme>
  </Rechnungspositionen>
</Rechnung>
```

XML-Datei

Transformation

```
<?xml version="1.0"?>
<xsl:transform version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:template match="/">
    <html>
      <body>
        <h2>Lieferschein</h2>
        <table border="1">
          <tr bgcolor="#9acd32">
            <th align="left">Anzahl</th>
            <th align="left">Artikel</th>
          </tr>
          <xsl:for-each
            select="/Rechnung/Rechnungspositionen/Position">
            <tr>
              <td><xsl:value-of select="Anzahl"/></td>
              <td><xsl:value-of select="Artikel"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:transform>
```

XPath

Lieferschein

Anzahl	Artikel
200	Gummibären
15	Kaugummi

XSL-Datei

XPath – Syntax

■ Auffinden von Elementen

- ▶ `/Rechnung/Rechnungspositionen/Position`
 - ▶▶ Wählt alle Elemente auf die dieser Pfad passt
- ▶ `//Datum`
 - ▶▶ Wählt alle "Datum"-Elemente

■ Auswählen unbekannter Elemente

- ▶ `//*[@Name]`
 - ▶▶ Wählt alle "Name"-Elemente die Vorfahren haben
- ▶ `/Rechnung/Rechnungspositionen/*`
 - ▶▶ Selektiert alle Elemente unterhalb der Rechnungspositionen

```
<?xml version="1.0"?>
<Rechnung>
  <Auftraggeber>
    <KundenNr>25562-556E</KundenNr>
    <Name>Universität Augsburg</Name>
    <Adresse>
      Universitätsstrasse 16, 86135 Augsburg
    </Adresse>
    <Datum>01.01.2002</Datum>
  </Auftraggeber>
  <Rechnungspositionen>
    <Position>
      <Artikel>Heissgetränkeautomat G45</Artikel>
      <Anzahl>10</Anzahl>
      <Preis Waehrung="EUR">1000</Preis>
    </Position>
    <Gesamtpreis Waehrung="EUR">1000</Gesamtpreis>
  </Rechnungspositionen>
</Rechnung>
```

XPath – Syntax

■ Auswahl bestimmter Elemente

- ▶ `/Rechnung/Rechnungspositionen/Position[Artikel='Kaugummi']`
 - ▶▶ Wählt alle Positionen die den Artikel „Kaugummi“ beinhalten
- ▶ `//Rechnungskopf[Name]`
 - ▶▶ Selektiert alle Rechnungsköpfe, die ein Element Name enthalten

■ Auffinden von Attributen

- ▶ `//*[@id]`
 - ▶▶ Sucht alle Elemente, die ein `id` Attribut haben
- ▶ `//Position[@*]`
 - ▶▶ Wählt alle "Position"-Elemente, irgendein Attribut haben



Modellierung mit Entity-Relationship-Diagrammen Teil 2

Kardinalitäten: Beziehungen zwischen Mengen

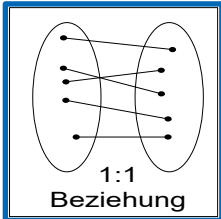


Abbildung auf Kardinalitäten
im ER Diagramm

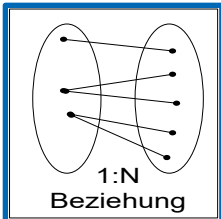


Abbildung auf Kardinalitäten
im ER Diagramm

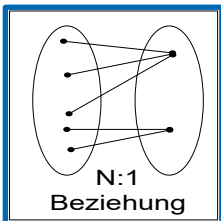
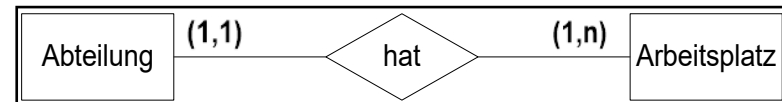


Abbildung auf Kardinalitäten
im ER Diagramm

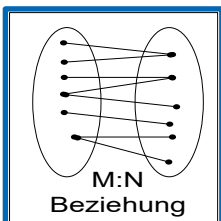
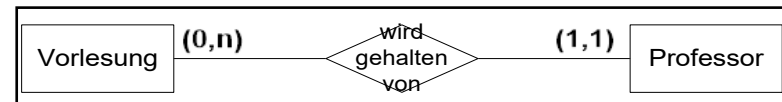
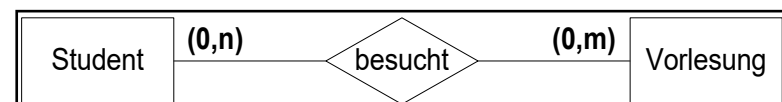
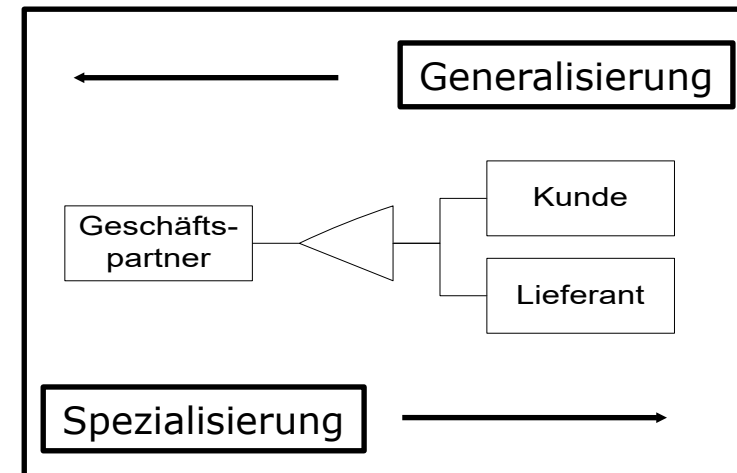


Abbildung auf Kardinalitäten
im ER Diagramm

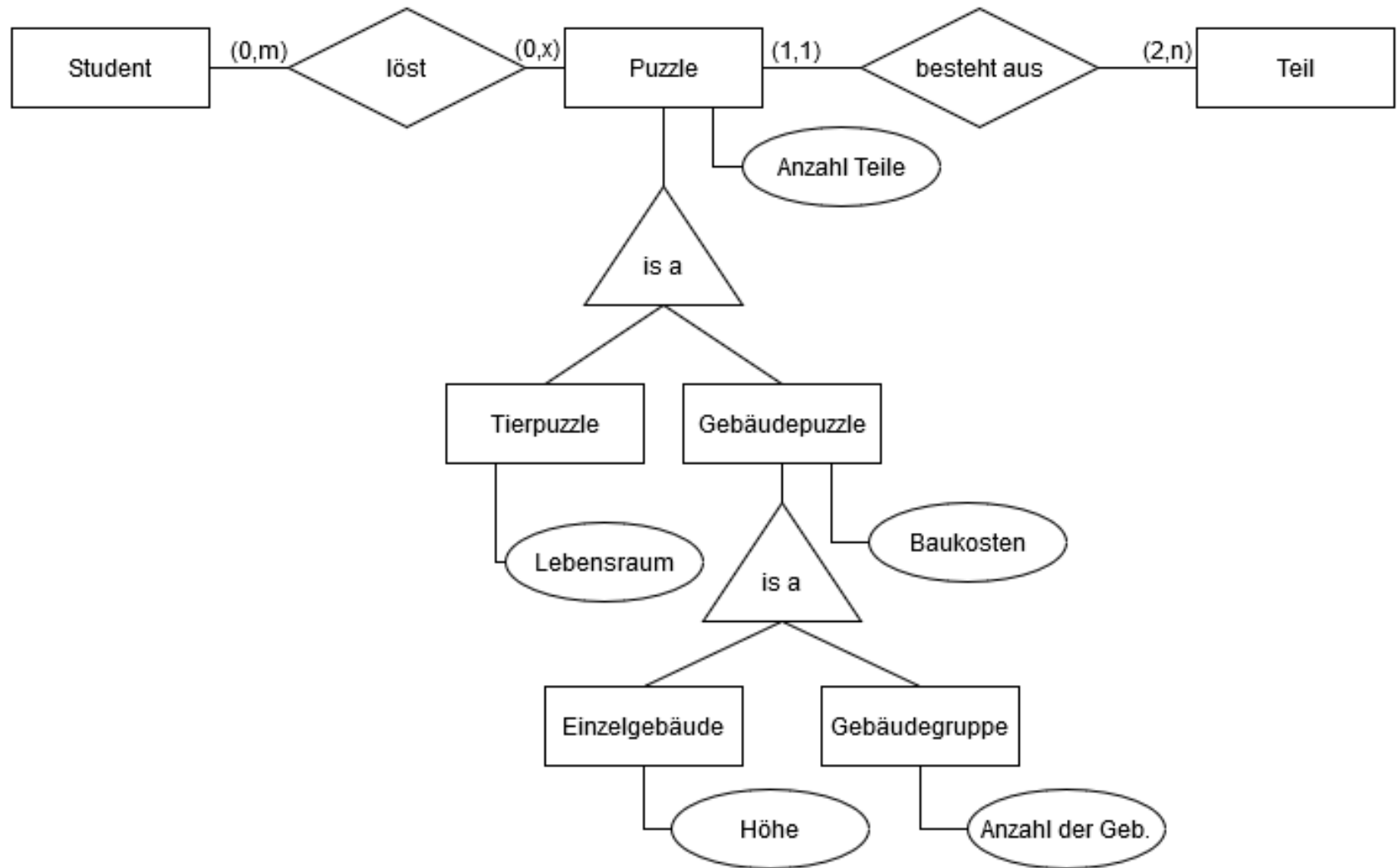


Generalisierung und Spezialisierung

- Entity-Typen werden zu einem übergreifenden Entity-Typ (Supertyp) zusammengefasst bzw. ein Entity-Typ wird in einzelne Typen zerlegt.
- Generalisierung
 - Betrachtung vom Speziellen (Subtyp) zum Allgemeinen (Supertyp)
- Spezialisierung
 - Betrachtung vom Allgemeinen (Supertyp) zum Speziellen (Subtyp)
- Attribute
 - Eigenschaften (Attribute) der Supertypen werden auf Subtypen vererbt
 - Gemeinsame Attribute werden dem Supertyp zugewiesen
 - Untergeordnete Entity-Typen (Subtypen) besitzen nur noch speziell sie ausweisende Eigenschaften



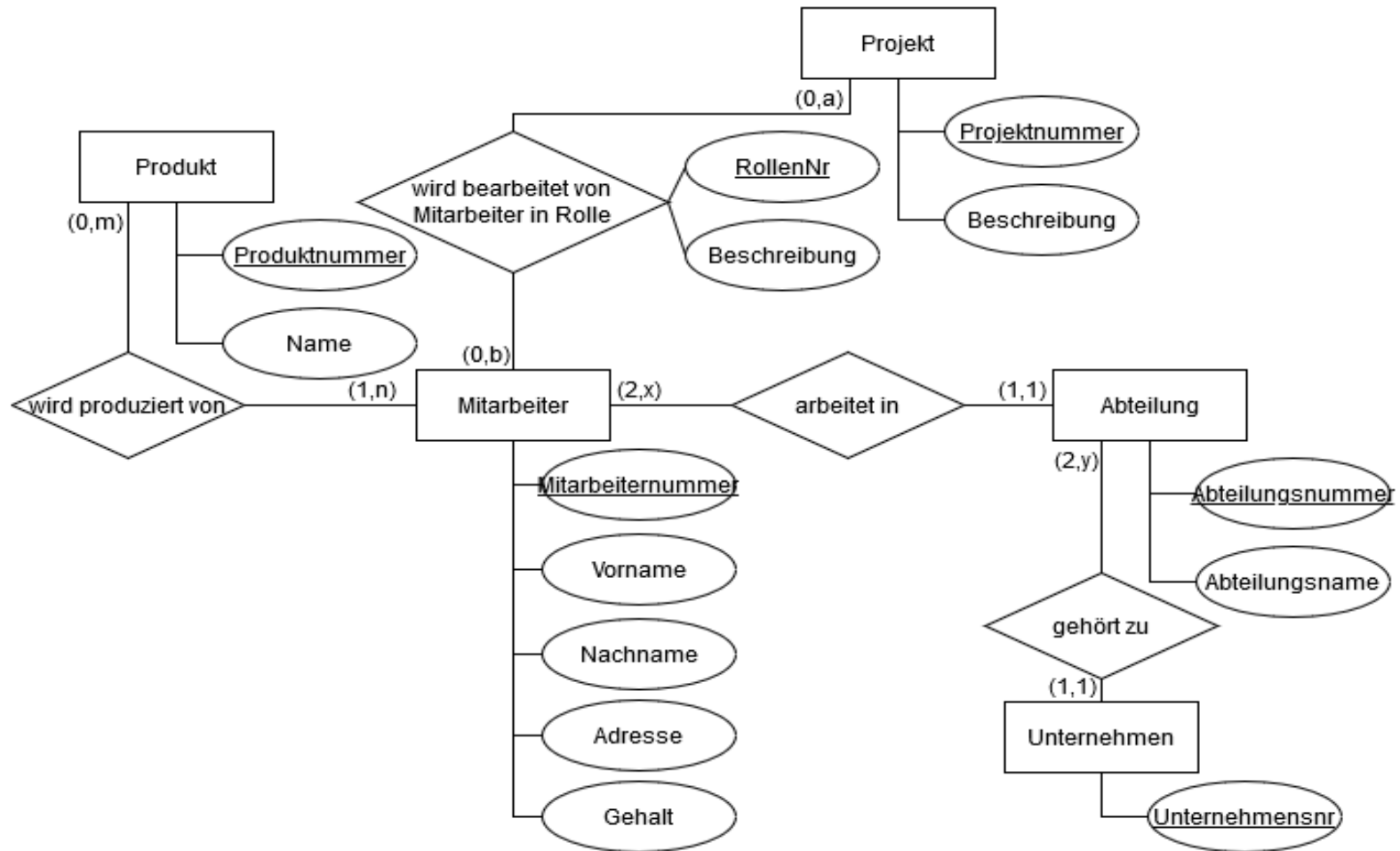
Beispiel: „Puzzle“



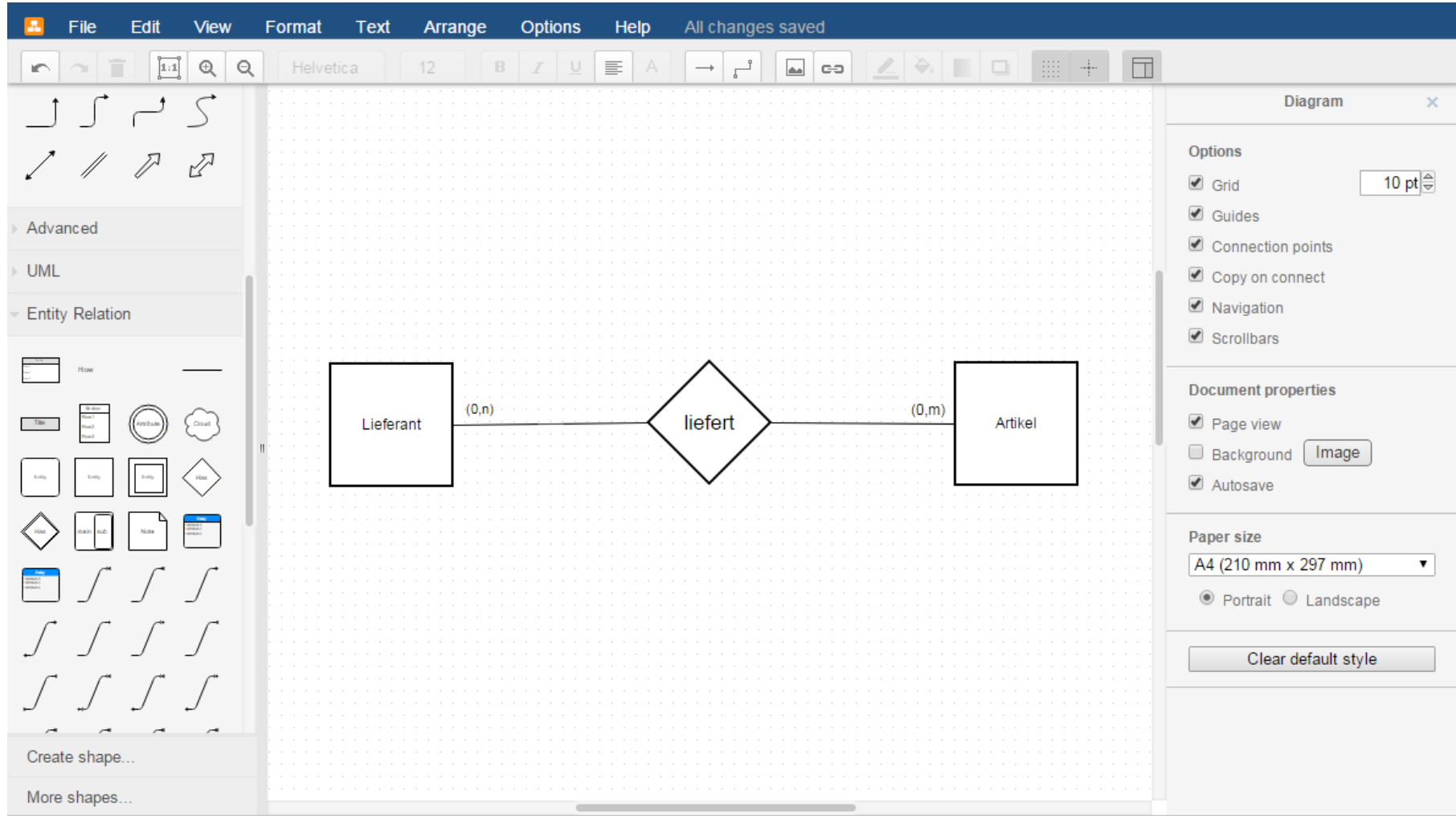
Beispiel: „Mitarbeiter“

1. Ein Mitarbeiter mit Nachname, Vorname, Adresse und Gehalt wird identifiziert durch seine Mitarbeiternummer.
2. Ein Produkt (mit Produktnummer und Name) wird von vielen, aber mindestens einem Mitarbeiter produziert und ein Mitarbeiter produziert viele Produkte.
3. In einem Projekt, identifiziert durch die Projektnummer und gekennzeichnet mit einer Beschreibung, arbeiten viele Mitarbeiter in unterschiedlichen Rollen mit einer eindeutigen RollenNr und einer Beschreibung und ein Mitarbeiter arbeitet in vielen Projekten.
4. Ein Mitarbeiter arbeitet genau in einer Abteilung und in einer Abteilung arbeiten mehrere Mitarbeiter
5. Eine Abteilung wird gekennzeichnet durch eine Abteilungsnummer (eindeutig) und durch einen Abteilungsnamen.
6. Eine Abteilung gehört genau zu einem (numerisch gekennzeichneten) Unternehmen; ein Unternehmen besteht aus mehreren Abteilungen.

Lösung „Mitarbeiter“



Tools: z.B. draw.io



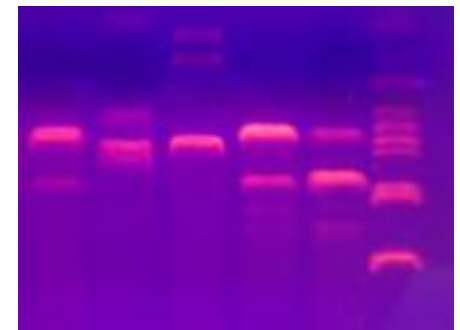
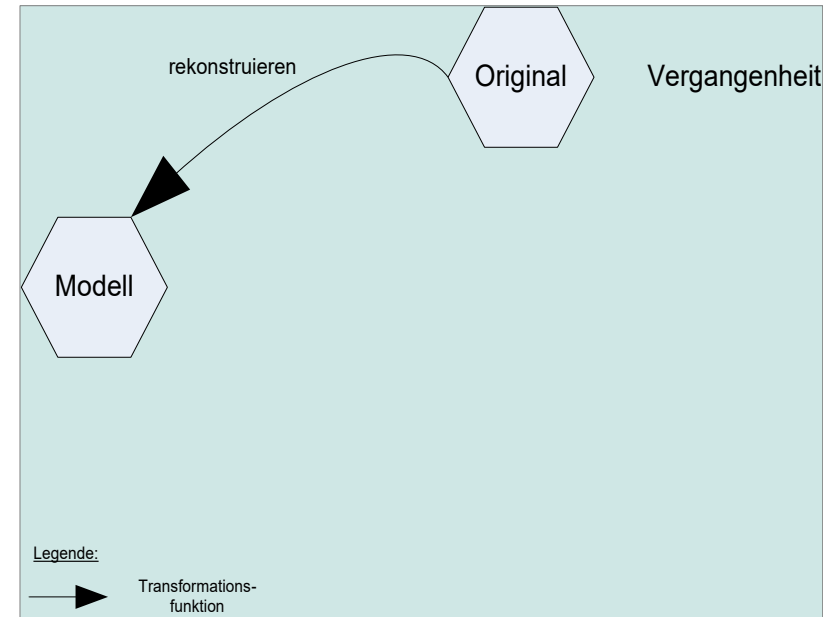
Modelle – wozu?

Modelle – wozu?

- Modellbildung ist Informationsarbeit.
- Modelle können kommuniziert werden.
 - Dienen Informationsaustausch zwischen Menschen und/oder Maschinen
- Drei Bedeutungen der Modell–Original–Abbildungen im Zeitverlauf:
 - Rekonstruieren
 - Konstruieren
 - Entwerfen

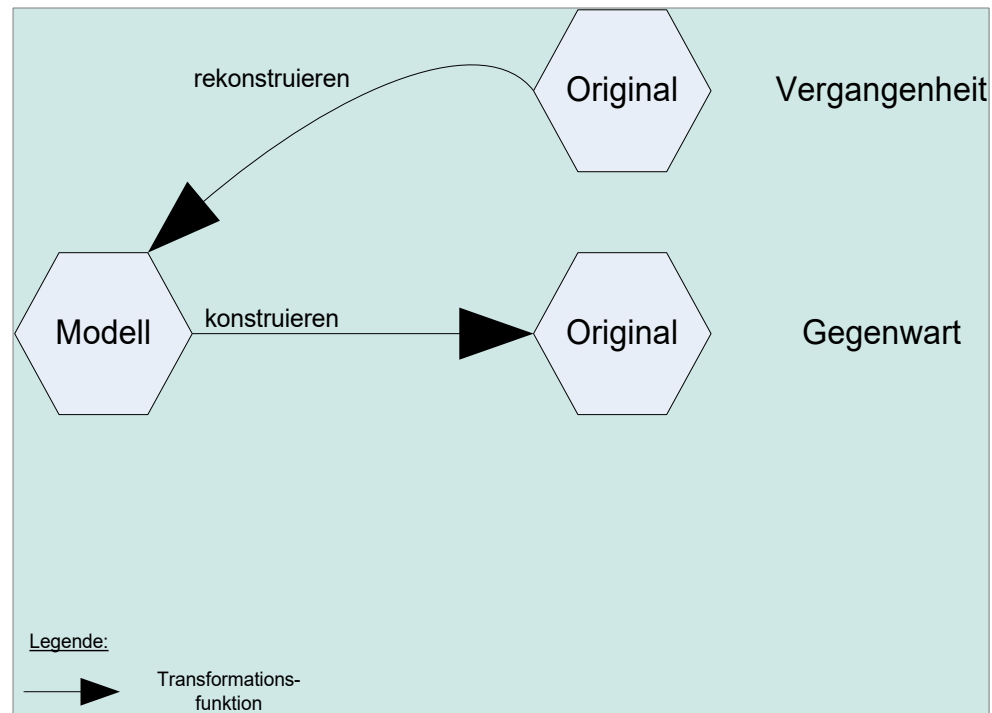
Modellierung – Rekonstruieren

- Etymologisch: „wiederherstellen“, „nachbilden“ eines ursprünglichen Zustands
- Modellierung:
 - Erstellung eines Modells, welches einen Originalzustand der Vergangenheit beschreibt
- Beispiele:
 - SW-Reengineering Modelle
 - Risikomodelle (Aktien)
 - Referenzmodelle?
 - Periodensystem
 - Aminosäure Codons
 - Fachsprachen

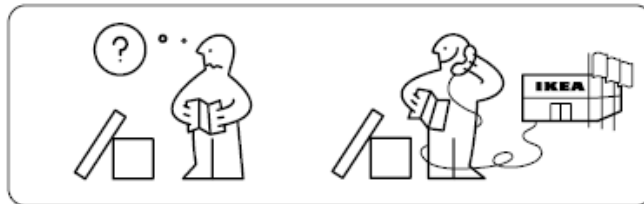
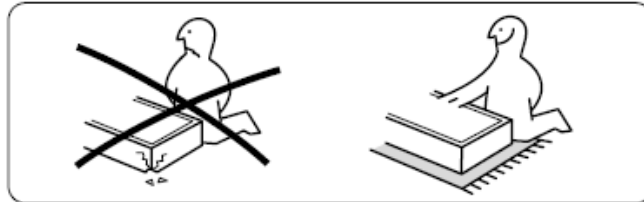
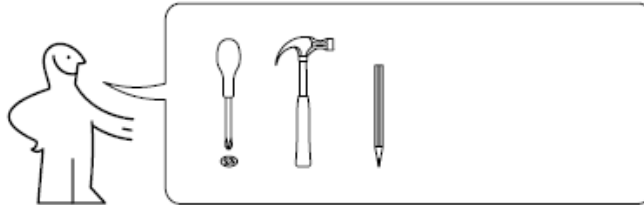


Modellierung – Konstruieren

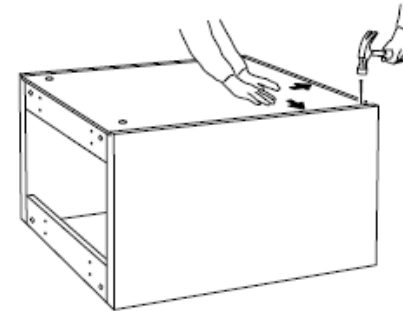
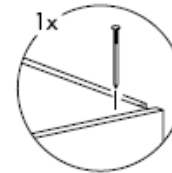
- Etymologisch: „errichten“, „erbauen“ oder „zusammenschichten“
- Modellierung:
 - Aus Modell wird das Original erstellt
 - Original kann ebenfalls Modell sein
- Beispiele:
 - Konstruktionszeichnungen
 - Source Code
 - CAD/CAM Modelle



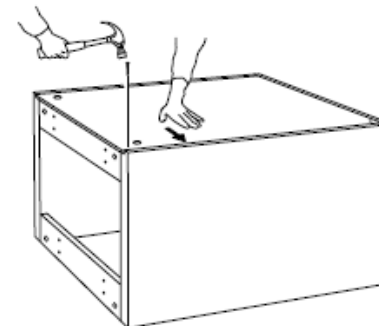
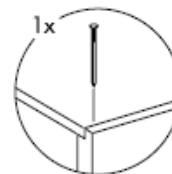
Modellierung – Konstruieren



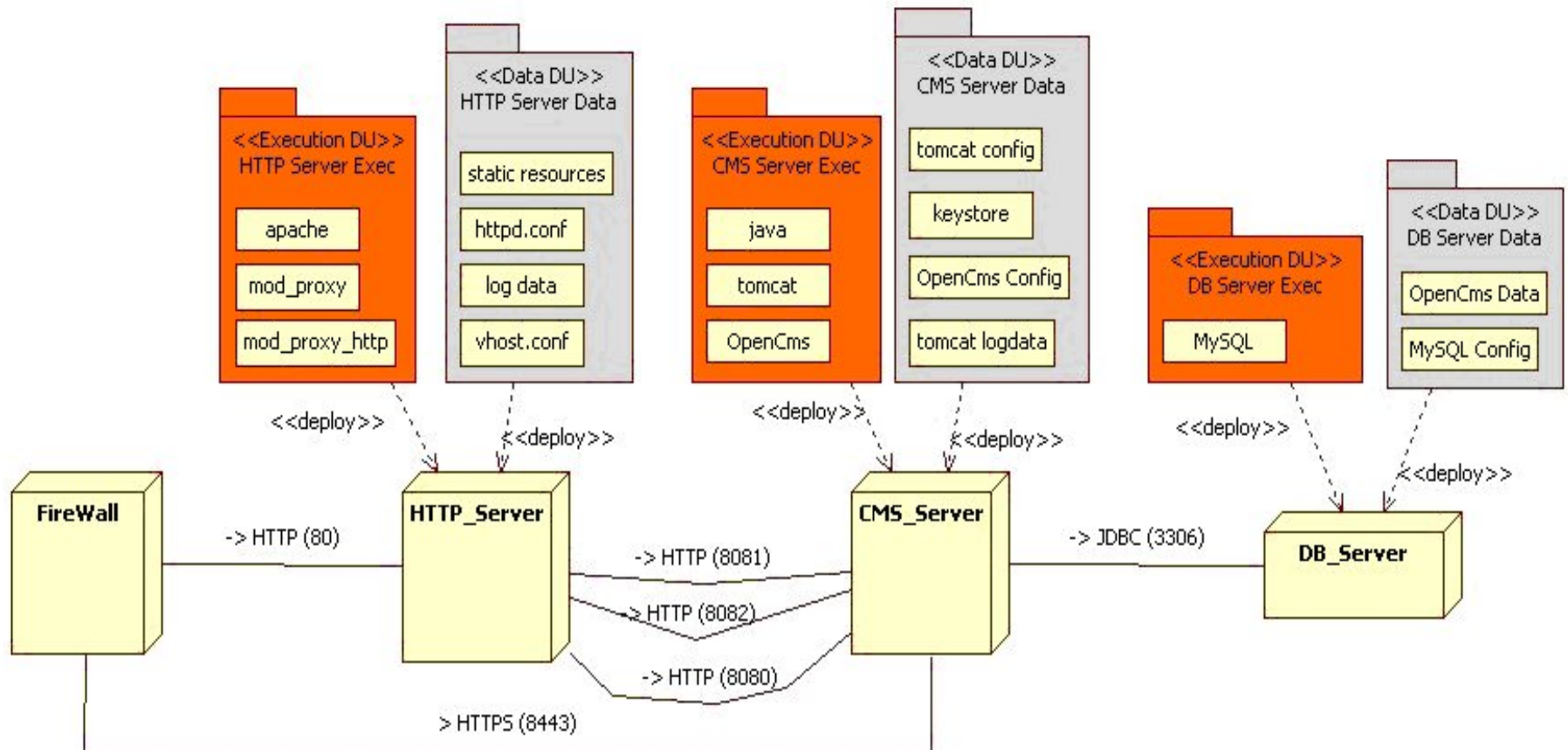
4



5

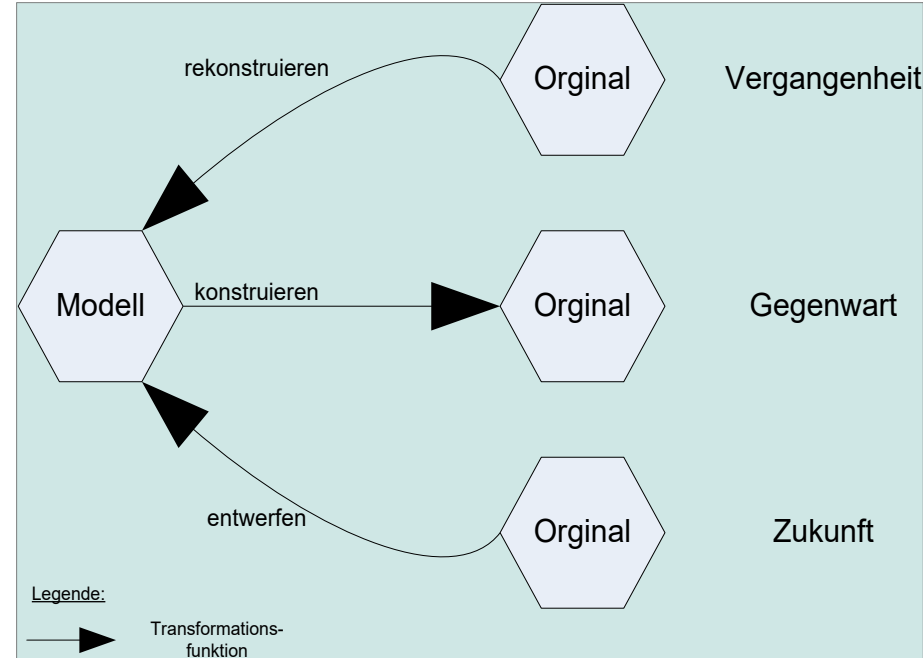


Modellierung – Konstruieren



Modellierung – Entwerfen

- Etymologisch: „ein Bild gestalten“, „literarisches und geistiges Gestalten“, vorausgedachtes Vorläufiges
 - Modellierung:
 - Zustand eines zukünftigen Originals
 - Annahme:
 - Transformation des gegenwärtigen Originals möglich
 - Konstruktionshypothese
- Beispiele:
 - Gebäudearchitekturen
 - Systemarchitekturen



Modellierung – Entwerfen

