DAIMLERCHRYSLER

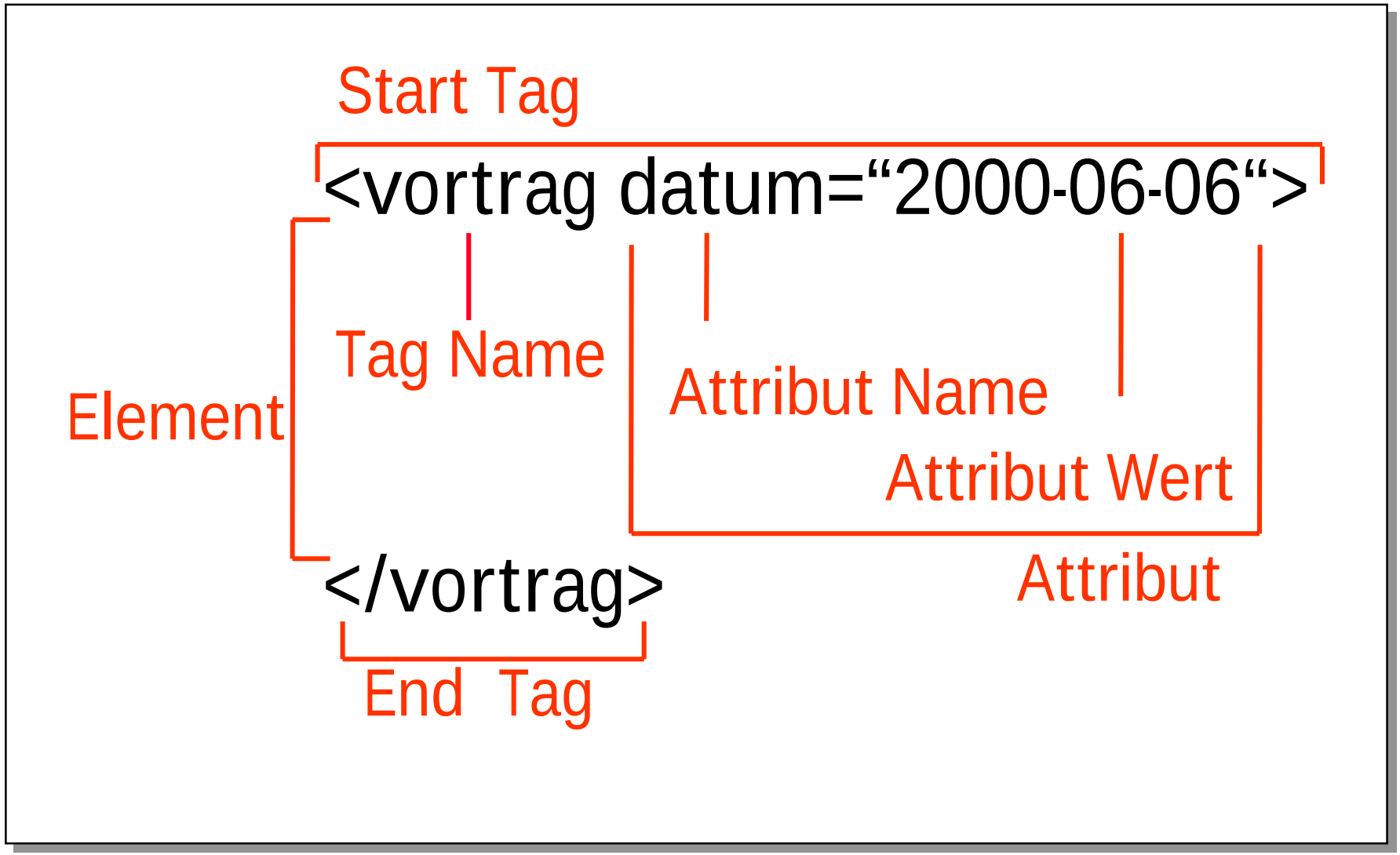
XML-Schema

der zukünftige Standard zur XML-Sprachdefinition

Mario Jeckle
mario.jeckle@daimlerchrysler.com
mario@jeckle.de
www.jeckle.de

DaimlerChrylser Research
dept. product development process chain (FT3/EK)

XML Terminologie



Zum Begriff *Schema*

[lat.-griech. *Haltung, Stellung; Gestalt, Figur, Form*]
(allgemein): Muster, Entwurf, Aufriss; Konzept.
(Philosophie und Logik):

1) Wissenschafts- und verfahrensmethodologisch werden Schemata bewusst und planmäßig in Form einer zielgerichteten Ordnung von Begriffen und Denkabläufen, als hypothetisches Konstrukt zur Erklärung von Zusammenhängen (z.B. *soziales Schema*) sowie Verfahrensplan (v.a. bei wiederkehrenden Situationen) angewendet.
2) Erkenntnistheoretisch hat nach I.Kant jeder reine Begriff (Kategorie) sein Schema, d.h. einen inneren Vorgang, zu durchlaufen, mit dem er sich durch eine stellvertretende Vorstellung anschaulich realisiert. Die Verstandesbegriffe haben je ein transzendentales Schema, wodurch ihnen erste eine Beziehung zum Sinnlichen gegeben wird. Der *Schematismus* ist somit das Verfahren des Verstandes, einem Begriff objektive Realität durch die ihm entsprechende Anschauung zu verschaffen.

[aus: Brockhaus -- Die Enzyklopädie, Mannheim, 1999, Bd. 19, S. 274]

Zum Begriff *Schema* in der Informatik

(allgemein) **Ein *Schema* beschreibt eine Datenstruktur**.
Schemata entstehend i.A. als Ergebnis eines (Informations- / Daten-) Modellierungsprozesses.
Typischerweise spricht man im Datenbank-Umfeld vom Schema, im Sinne der Struktur- und Inhaltsbeschreibung, der in der Datenbank abgelegten Daten.
In diesem Kontext werden auch verschiedene *Schematypen* (wie *konzeptuelles, logisches, internes* und *externes S*.) unterschieden.

Zum Begriff *Schema* im Kontext *XML*

Das Schema einer XML-Datei wird normgemäß durch eine *Document Type Definition* (DTD), gebildet.
Oftmals werden die Begriffe *Schema* und *Grammatik* im Umfeld XML synonym gebraucht, dies trifft jedoch nicht zu.
Während die *Grammatik* -- als Syntaxbeschreibung -- eins SGML-, und damit auch XML-, Dokuments absolut benötigt wird, um die Dokumentstruktur zu verstehen, definiert ein *Schema* Regeln und Einschränkungen an die logische Dokumentstruktur.
Als Konsequenz der fixierten Dokumentsyntax von XML (in SGML kann diese variieren) wird durch eine XML-DTD lediglich der semantische Anteil der Dokumentbeschreibung realisiert.
Es sind jedoch beliebige Beschreibungssprachen konkreter XML-Dokument-Ausprägungen denkbar.
XML-Schema ist eine XML-Sprache zur Beschreibung beliebiger XML-Dokumentinhalte.

Well-formedness

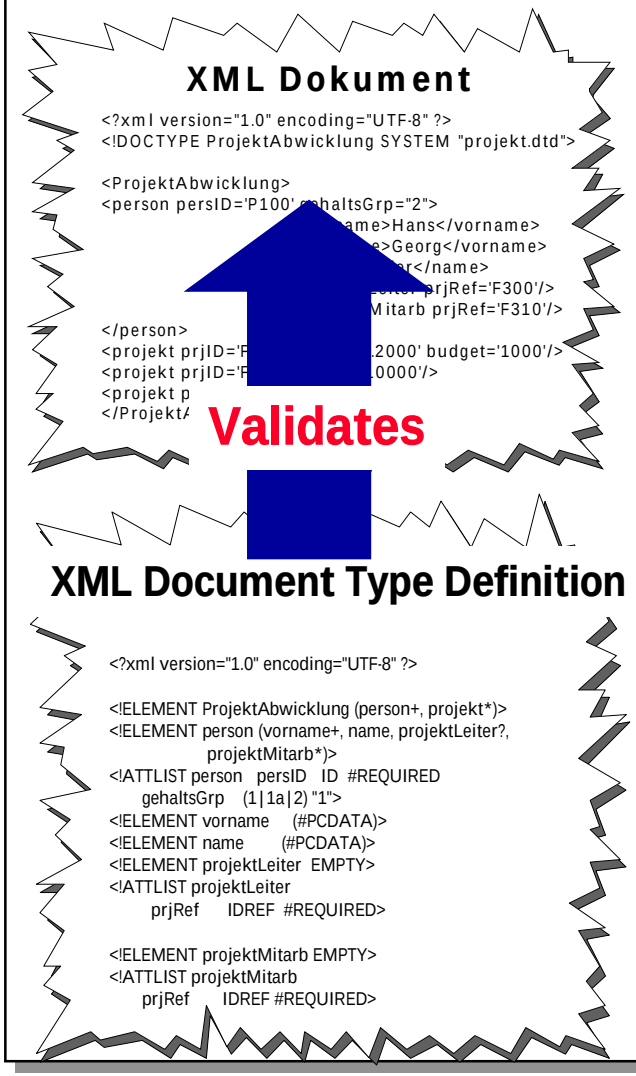
XML Dokument

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE ProjektAbwicklung SYSTEM "projekt.dtd">

<ProjektAbwicklung>
<person persID='P100' gehaltsGrp="2">
  <vorname>Hans</vorname>
  <vorname>Georg</vorname>
  <name>Meier</name>
  <projektLeiter prjRef='F300'/>
  <projektMitarb prjRef='F310'/>
</person>
<projekt prjID='F300' start='1.1.2000' budget='1000'/>
<projekt prjID='F310' budget='10000'/>
<projekt prjID='F320' budget='25000'/>
</ProjektAbwicklung>
```

Dokument...

- Enthält mindestens ein Element
- Baumartige Organisation (Wurzelelement existiert)
- Korrekte Elementschachtelung
- Übereinstimmende öffnende und schließende Tag-Namen
- Attributnamen sind innerhalb Elementen eindeutig
- Attributwerte sind in doppelten Anführungszeichen
- Attributwerte verweisen nicht auf externe Entities
- Keine Markupsymbole in Attributwerten
- Entity-Deklaration vor Referenzierung
- Keine Entity-Referenz verweist auf unparsed Entity

Validness



Ein XML-Dokument ist gültig, wenn

- eine DTD existiert
  (und referenziert oder eingebunden ist)
- die durch die DTD formulierten syntaktischen Regeln eingehalten sind

Validness

Allein die Existenz und Referenzierung oder Inkludierung einer DTD rechtfertigt es nicht, von einem gültigen XML-Dokument zu sprechen.

Die Klasse der nicht validierenden Parser läßt per definitionem jegliche DTD beim Prüfungsvorgang außer Acht. Auch kann ein Parser dieses Typs beliebige Teile der DTD (beispielsweise *extenal subsets*) ignorieren.

Well-formedness und validness

Die Begriffe *well-formed* und *valid* stammen von Tim Bray.
Trotz des Einwands durch Sperberg Mc-Queen, beide Begriffe wären bereits in anderen Kontexten vorbelegt, überlebte der Vorschlag -- nicht zuletzt in Ermangelung eines Besseren -- bis heute.

Der Begriff *well-formed* steht in keinem Bezug zur gleichen Bezeichnung in der formalen mathematischen Logik.

In SGML sind für *well-formed* bzw. *valid* die Begriffe *tag-valid* bzw. *type-valid* gebräuchlich.

Ein Dokument, für das nicht der Anspruch erhoben wird es sei *valid* hat:

- keine DTD (weder *internal subset* noch DOCTYPE-Referenz)
- keine vorliegende DTD (z.B. zugänglich über Netz, aber nicht lokal repliziert)
- eine vorliegende DTD, die jedoch nicht zur Validierung herangezogen wird

DTD-Mechanismus von XML v1.0

- streng hierarchisch
- *ELEMENTs* als innere Knoten
- *ATTLISTs* zur Attributierung der Knoten
- Keine Datentypen (abgesehen von CHAR-Data)
- Rudimentärer Referenzierungsmechanismus (ID, IDREF(S))
- Selektionstyp (enum)
- Vorgabewerte
- DTD ist nicht XML

=> Notwendige Konstrukte zum Ausdruck mächtigerer Semantik müssen aufwendig und proprietär realisiert werden

Motivation *Schema-Mechanismen für XML*

Gemeinsamer Wortschatz

- ermöglicht Kommunikation und Interaktion auf Basis einheitlich definierter Begriffe

Formale Beschreibung

- Grundvoraussetzung maschineller Verarbeitung
- Idealerweise (vergleichsweise) einfache Verarbeitung

(schlanke, eindeutige Definitionen, möglichst kontextfreie oder reguläre Sprache)

Austauschbasis

- Explizite Strukturdefinition ist Grundvoraussetzung des Informationsaustauschs.

Anforderungen an einen XML-Schema Mechanismus

Structural schemas

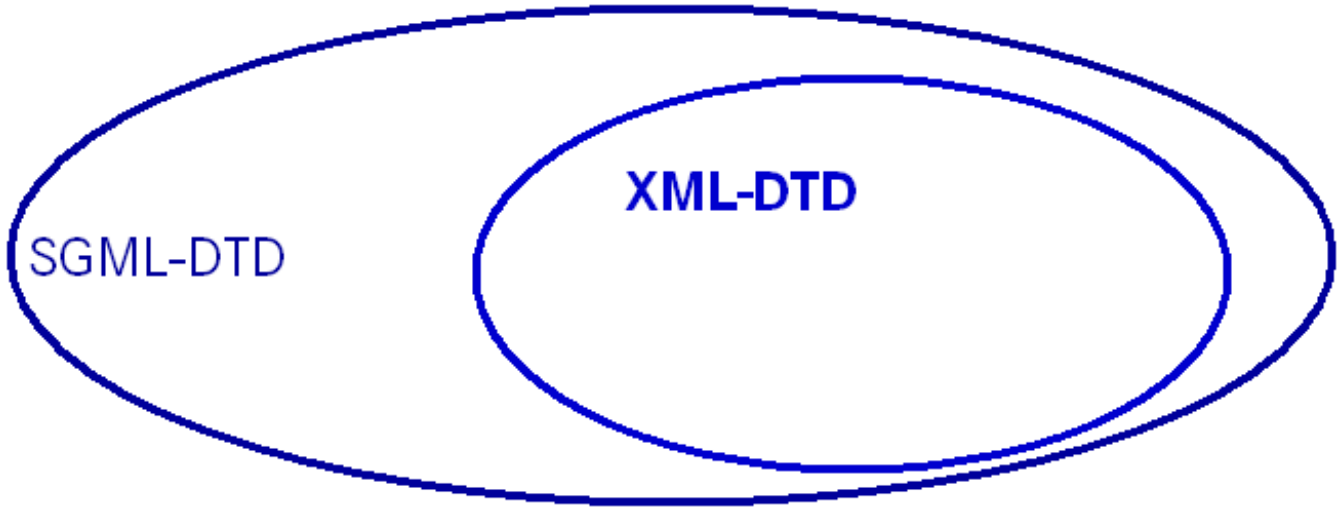
- Mächtigkeit analog des bestehenden DTD-Mechanismus um Dokumentstruktur (Reihenfolge, Auftrittsvielfachheit von Elementen und Attribute) zu beschreiben
  Insbesondere sollen folgende Erweiterungen verwirklicht werden:
  - Namespace Integration
  - Definition von Einschränkungen für Elementinhalte
  - Integration Strukturschema und primitive Datentypen
  - Vererbung: DTD unterstützt nur *kind-of*-Beziehungen
  - Erweiterter Referenzierungsmechanismus (URI)

Anforderungen an einen XML-Schema Mechanismus

Primitive Datentypen

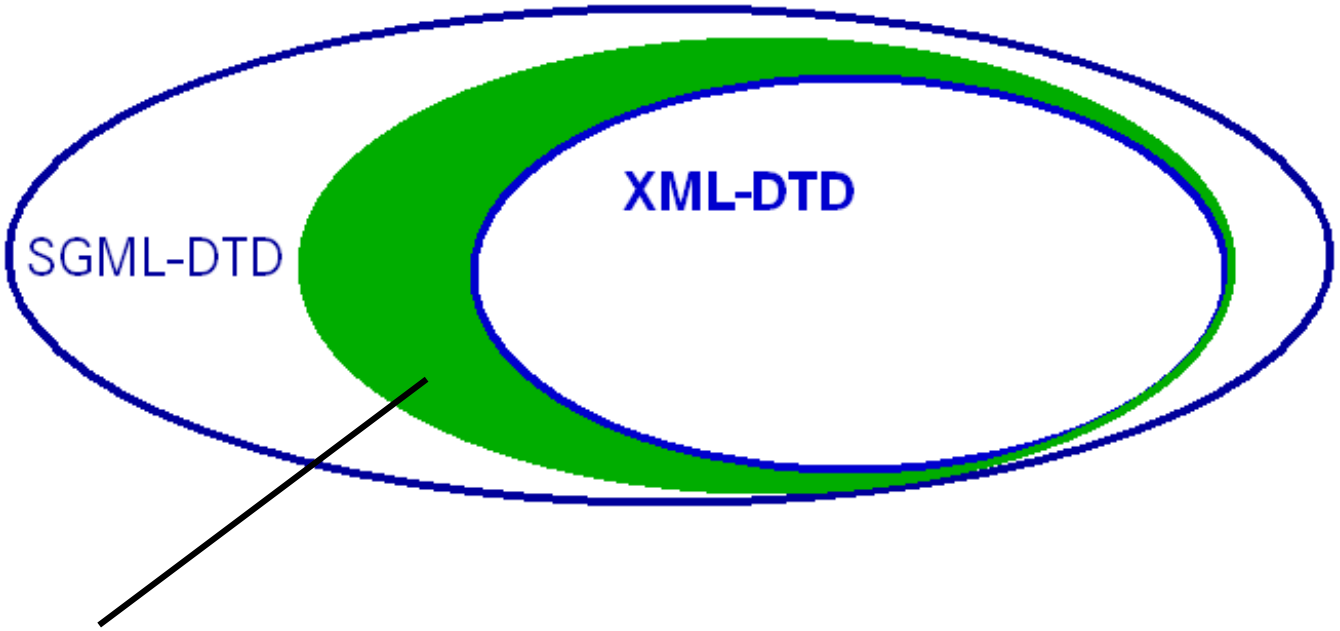
- „klassische“ atomare Datentypen, ergänzt um SQL-artige, wie *integer*, *date.*
- Programmiersprachen-übliche (typischerweise Java-artige) *build-in types*
- uninterpretierte Binärstrukturen
- (durch Anwender) erweiterbares Typsystem
- lexikalische Definitionen
- Einschränkungen an Typen

Weiterentwicklungsoptionen für DTDs



Ausdrucksmächtigkeit der XML-DTD bildet eine (echte) Untermenge des SGML-DTD Mechanismus.

Weiterentwicklungsoptionen für DTDs



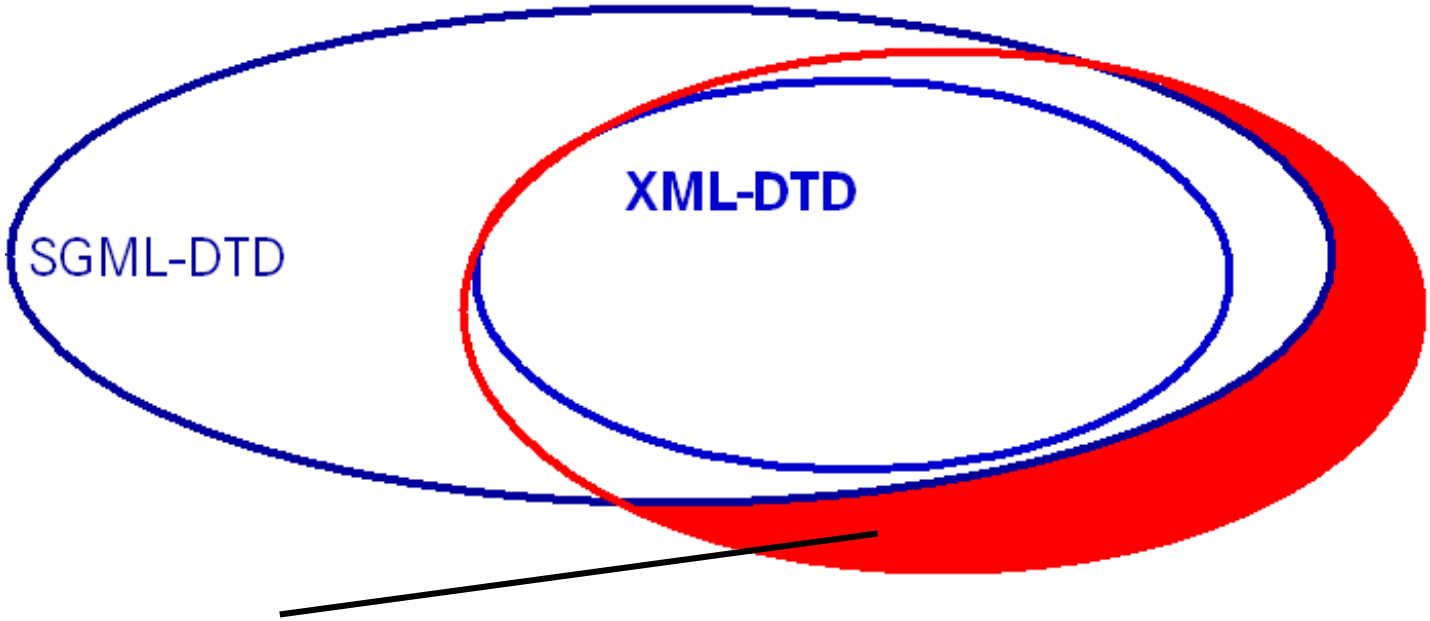
Erweiterung des XML-DTD Mechanismus um weitere Elemente der SGML-DTD.

(+) Ausdrucksmächtigkeit nähert sich (wieder) der von SGML an

(-) ...die Komplexität auch

(-) Ausdrucksmächtigkeit kann die von SGML niemals übertreffen

Weiterentwicklungsoptionen für DTDs



Erweiterung des XML-DTD Mechanismus um Elemente, die *nicht* mit SGML-Mitteln ausdrückbar sind.

(+) Freiheitsgrad hinsichtlich beliebiger Erweiterungen

(-) XML-Grundforderung nach Untermengenbeziehung zu SGML entfällt

(-) immernoch zwei verschiedene Sprachen für Inhalt und Schema

Existierende Schema-Sprachen für XML

Erweiterungen des bestehenden (SGML-/XML-)DTD-Mechanismus

- Data Types for DTD (DT4DTD)

Wissensbeschreibung

- Document Content Description for XML (DCD)

  (*RDF basierte Weiterentwicklung von XML-Data*)

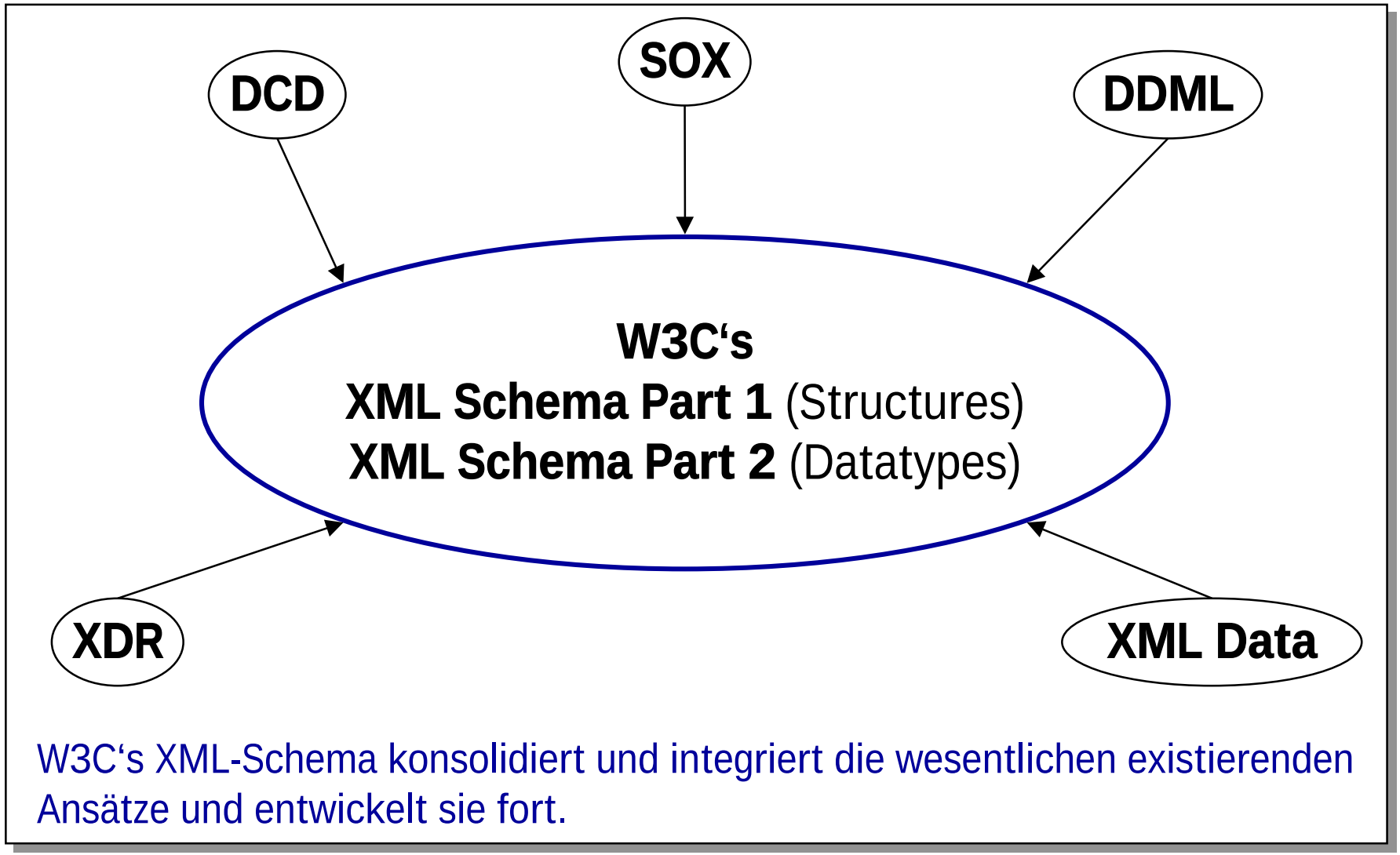
Inspiriert durch XML-API-Entwicklung

- Schema for Object oriented XML (SOX)

XML-Sprachen zur Inhaltsbeschreibung

- Document Definition Markup Language/XSchema (DDML)
- Schematron (XSL-basierte Auswertung der Dokumentstruktur)
- XML-Data/XML-Data Reduced (XDR) *(erster Ansatz noch vor Verabschiedung XML 1.0)*
- Document Structure Description (DSD)
- **W3C's XML-Schema**

W3C's *XML Schema*



W3C's XML-Schema konsolidiert und integriert die wesentlichen existierenden Ansätze und entwickelt sie fort.

W3C's *XML Schema*

W3C's XML beschreiben XML-Dokumentklassen
unter Verwendung der XML-Dokumentkonstrukte um

- Bedeutung
- Aussage
- Beziehung

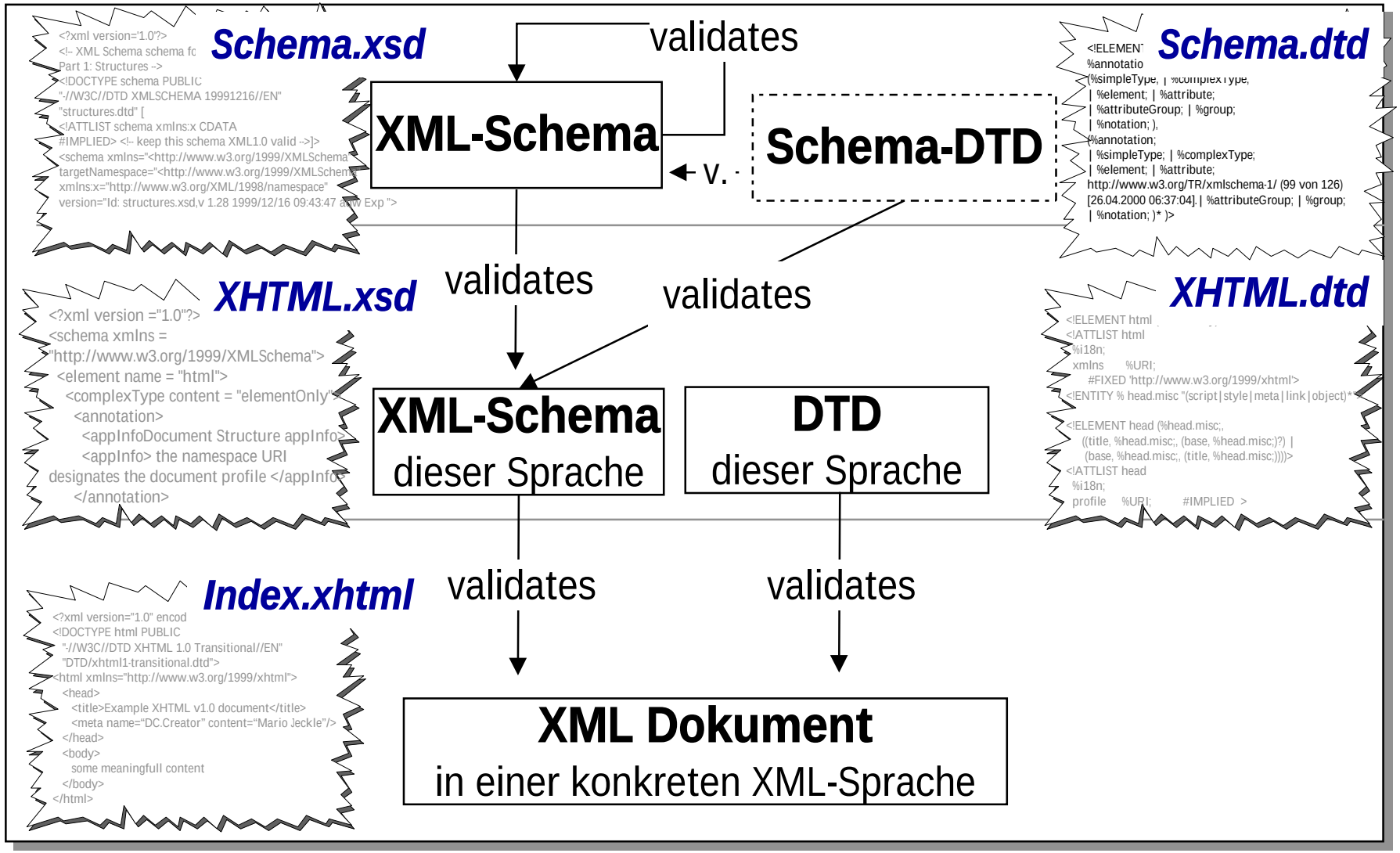
der Dokumentbestandteile einzuschränken und zu dokumentieren.

Bestandteile eines XML-Dokuments:

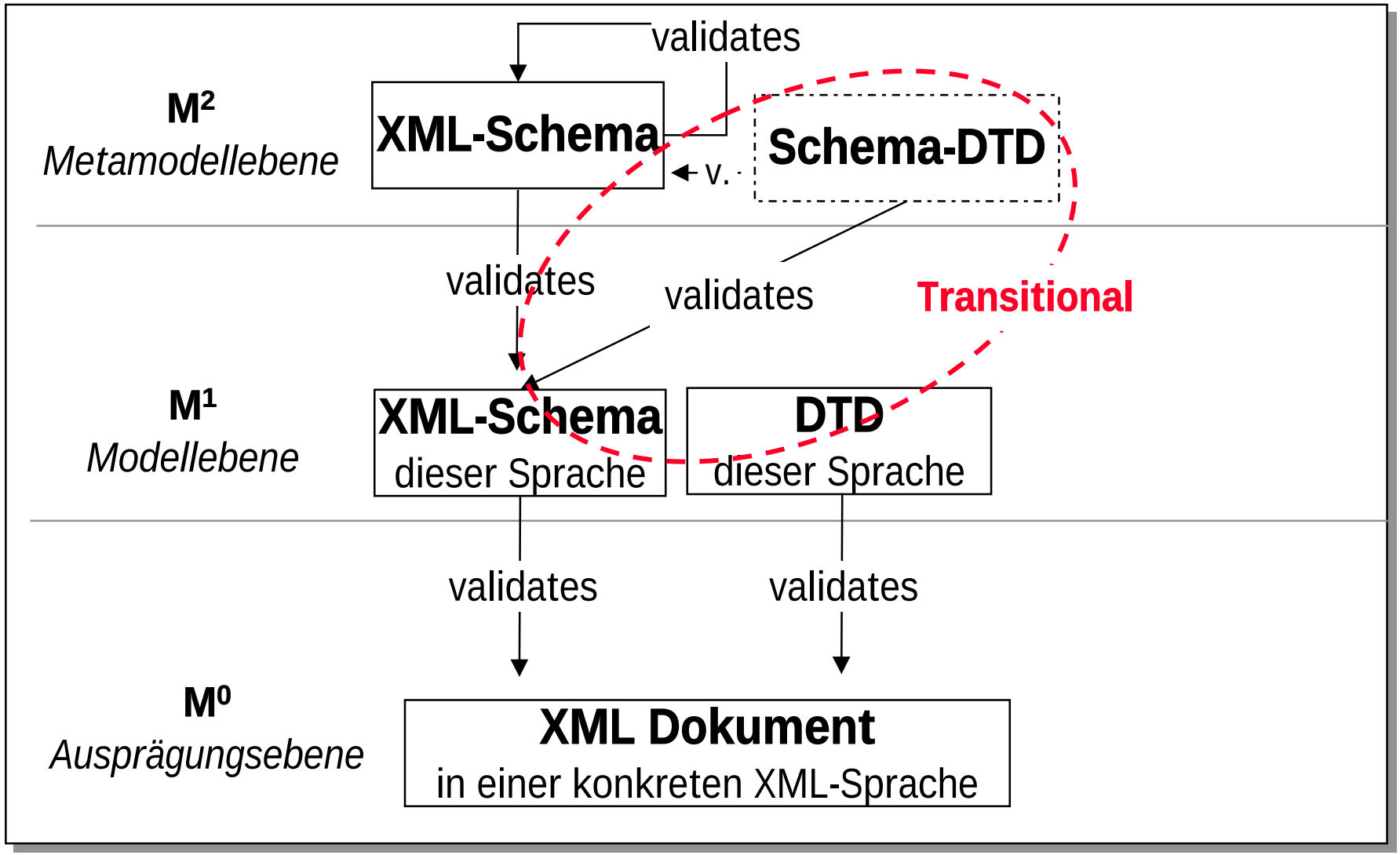
- Datentypen
- Elemente und ihr Inhalt
- Attribute und ihre Werte
- Entities und ihr Inhalt
- Notations

XML Schema sind selbstbeschreibend

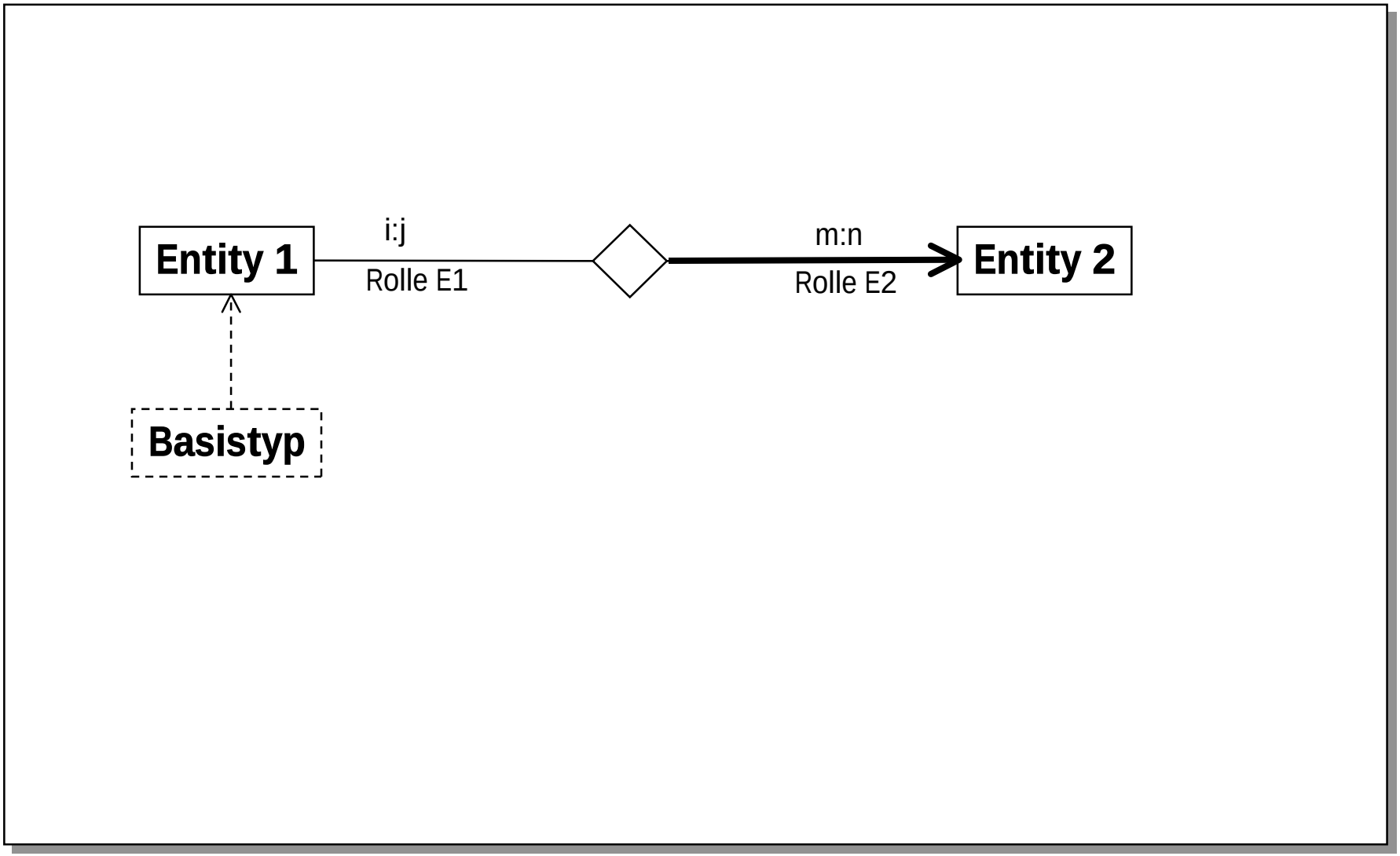
W3C's *XML Schema* -- Technologie (Metamodellierung)



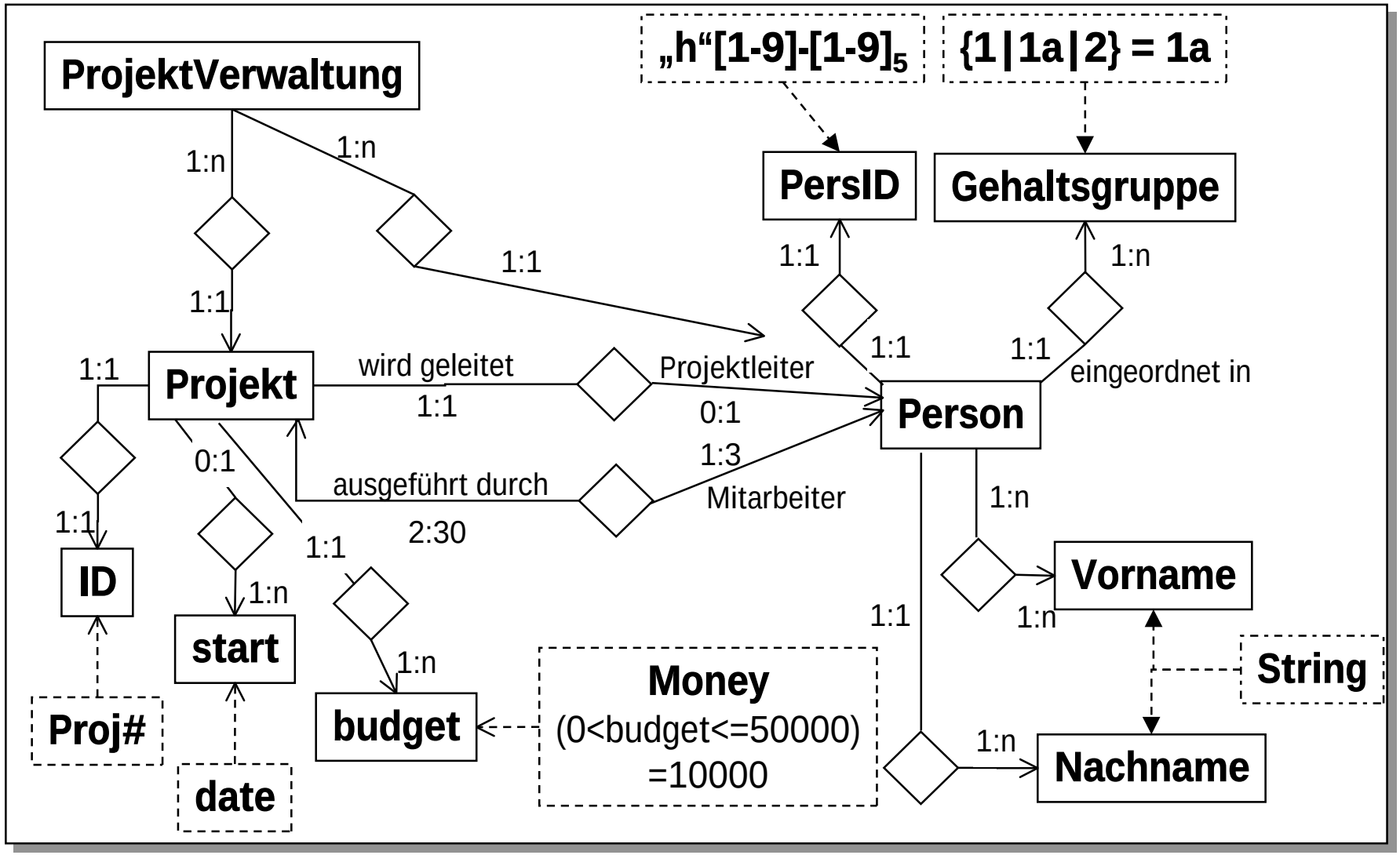
W3C's *XML Schema* -- Technologie (Metamodellierung)



W3C's *XML Schema* -- Beschreibungssprache des Beispiels

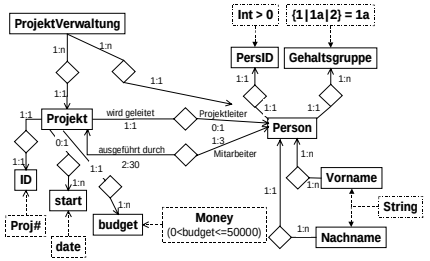


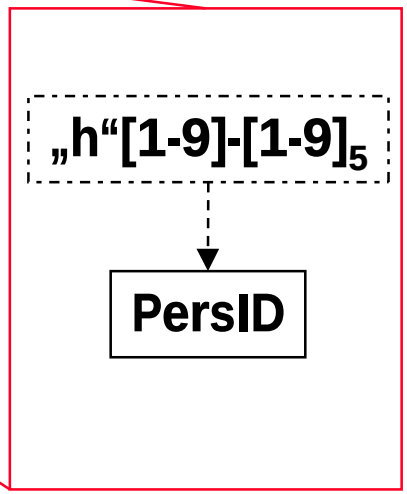
W3C's *XML Schema* -- Beispiel



W3C's *XML Schema* -- Beispiel (als DTD)

```
<!ELEMENT ProjektVerwaltung  (Person+ , Projekt+ )>

<!ELEMENT Person (Vorname+, Nachname+)>

<!ATTLIST Person
   PersID      ID   #REQUIRED
   Gehaltsgruppe   (1|1a|2) '1a'
   mitarbeitInProjekt   IDREFS   #REQUIRED>

<!ELEMENT Vorname (#PCDATA)>
<!ELEMENT Nachname (#PCDATA)>

<!ELEMENT Projekt EMPTY>

<!ATTLIST Projekt
   ID       ID    #REQUIRED
   date      CDATA   #IMPLIED
   budget      CDATA   #REQUIRED
   Projektleiter   IDREF   #REQUIRED
   Mitarbeiter   IDREFS   #REQUIRED>
```

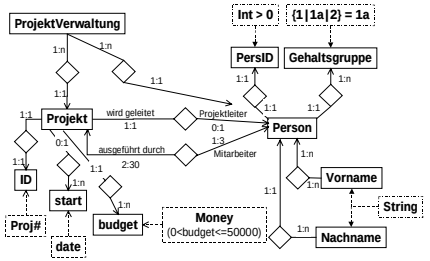


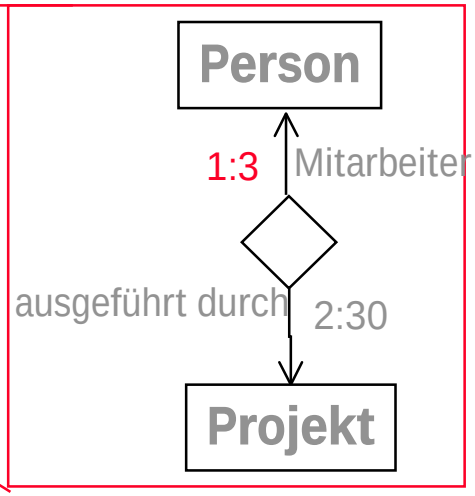


(lexikalische Festlegung)
Mit DTD nicht darstellbar!

W3C's *XML Schema* -- Beispiel (als DTD)

```
<!ELEMENT ProjektVerwaltung  (Person+ , Projekt+ )>

<!ELEMENT Person (Vorname+, Nachname+)>

<!ATTLIST Person
   PersID      ID   #REQUIRED
   Gehaltsgruppe    (1|1a|2) '1a'
   mitarbeitInProjekt   IDREFS   #REQUIRED>

<!ELEMENT Vorname (#PCDATA)>
<!ELEMENT Nachname (#PCDATA)>

<!ELEMENT Projekt EMPTY>

<!ATTLIST Projekt
   ID       ID    #REQUIRED
   date      CDATA   #IMPLIED
   budget      CDATA   #REQUIRED
   Projektleiter   IDREF   #REQUIRED
   Mitarbeiter   IDREFS   #REQUIRED>
```

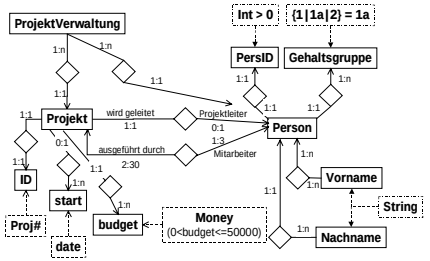


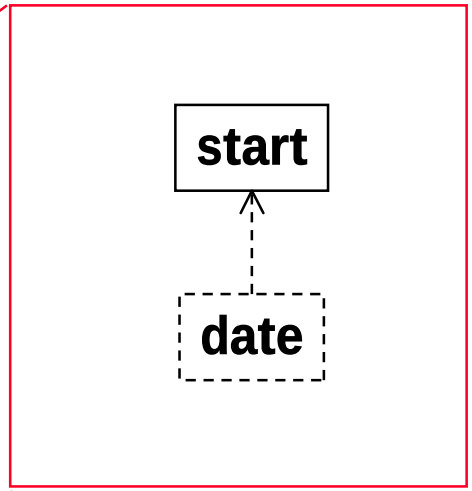


(Kardinalität)
Mit DTD nicht darstellbar!

W3C's *XML Schema* -- Beispiel (als DTD)

```
<!ELEMENT ProjektVerwaltung  (Person+ , Projekt+ )>

<!ELEMENT Person (Vorname+, Nachname+)>

<!ATTLIST Person
   PersID      ID   #REQUIRED
   Gehaltsgruppe    (1|1a|2) '1a'
   mitarbeitInProjekt   IDREFS   #REQUIRED>

<!ELEMENT Vorname (#PCDATA)>
<!ELEMENT Nachname (#PCDATA)>

<!ELEMENT Projekt EMPTY>

<!ATTLIST Projekt
   ID       ID    #REQUIRED
   start      CDATA   #IMPLIED
   budget      CDATA   #REQUIRED
   Projektleiter   IDREF   #REQUIRED
   Mitarbeiter   IDREFS   #REQUIRED>
```

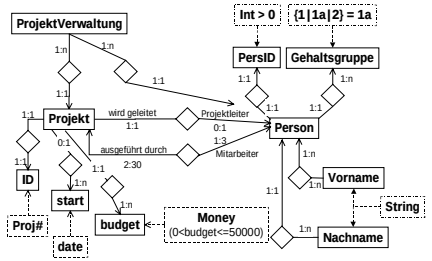


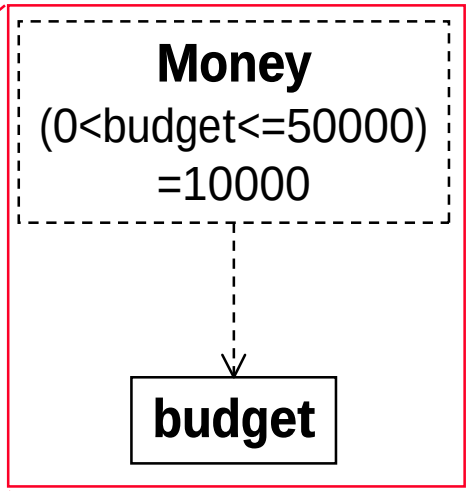


(Datentyp)
Mit DTD nicht darstellbar!

W3C's *XML Schema* -- Beispiel (als DTD)

```
<!ELEMENT ProjektVerwaltung  (Person+ , Projekt+ )>

<!ELEMENT Person (Vorname+, Nachname+)>

<!ATTLIST Person
   PersID      ID   #REQUIRED
   Gehaltsgruppe    (1|1a|2) '1a'
   mitarbeitInProjekt   IDREFS   #REQUIRED>

<!ELEMENT Vorname (#PCDATA)>
<!ELEMENT Nachname (#PCDATA)>

<!ELEMENT Projekt EMPTY>

<!ATTLIST Projekt
   ID       ID    #REQUIRED
   start      CDATA   #IMPLIED
   budget      CDATA   #REQUIRED
   Projektleiter   IDREF   #REQUIRED
   Mitarbeiter   IDREFS   #REQUIRED>
```





(Domänenrestriktion und **Vorgabewert)**
Mit DTD nicht darstellbar!

W3C's *XML Schema* -- Anatomie eines XSD-Dokuments

```
<xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <xsd:annotation>
    <xsd:documentation>
    Uniterpreted Commend
    </xsd:documentation>
  </xsd:annotation>
  <xsd:complexType ... >
    <xsd:element ...>
    <xsd:attribute ...>
  </xsd:complexType>
  <xsd:simpleType ...>
    <xsd:pattern ...>
  </xsd:simpleType>
</xsd:schema>
```

Schema-Start mit Namespacedeklaration

W3C's *XML Schema* -- Anatomie eines XSD-Dokuments

```
<xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <xsd:annotation>
    <xsd:documentation>
    Uniterpreted Commend
    </xsd:documentation>
  </xsd:annotation>
  <xsd:complexType ... >
    <xsd:element ...>
    <xsd:attribute ...>
  </xsd:complexType>
  <xsd:simpleType ...>
    <xsd:pattern ...>
  </xsd:simpleType>
</xsd:schema>
```

Dokumentierender Kommentar

W3C's *XML Schema* -- Anatomie eines XSD-Dokuments

```
<xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <xsd:annotation>
     <xsd:documentation>
     Uniterpreted Commend
     </xsd:documentation>
  </xsd:annotation>
  <xsd:complexType ... >
     <xsd:element ...>
     <xsd:attribute ...>
  </xsd:complexType>
  <xsd:simpleType ...>
     <xsd:pattern ...>
  </xsd:simpleType>
</xsd:schema>
```

Anwenderdefinierte (komplexe) Typen

W3C's *XML Schema* -- Anatomie eines XSD-Dokuments

```
<xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <xsd:annotation>
     <xsd:documentation>
     Uniterpreted Commend
     </xsd:documentation>
   </xsd:annotation>
   <xsd:complexType ... >
     <xsd:element ...>
     <xsd:attribute ...>
   </xsd:complexType>
   <xsd:simpleType ...>
     <xsd:pattern ...>
   </xsd:simpleType>
</xsd:schema>
```

Elemente und Attribute

W3C's *XML Schema* -- Anatomie eines XSD-Dokuments

```
<xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema">
 <xsd:annotation>
    <xsd:documentation>
    Uniterpreted Commend
    </xsd:documentation>
 </xsd:annotation>
 <xsd:complexType ... >
    <xsd:element ...>
    <xsd:attribute ...>
 </xsd:complexType>
 <xsd:simpleType ...>
    <xsd:pattern ...>
 </xsd:simpleType>
</xsd:schema>
```

Anwenderdefinierte (einfache) Datentypen

W3C's *XML Schema* -- Part 1: Structures

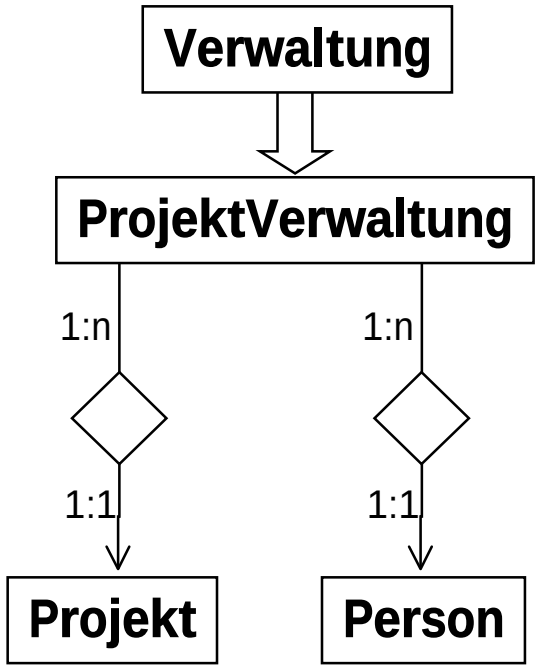
Typdefinition -- *complex type*

<complexType
abstract = boolean : false
base = QName
block = *#all or (possibly empty) subset of {extension, restriction}*
content = elementOnly | empty | mixed | textOnly
derivedBy = extension | restriction
final = *#all or (possibly empty) subset of {extension, restriction}*
id = ID
name = NCName>
Content: (annotation? , (((minExclusive | minInclusive | maxExclusive | maxInclusive
| precision | scale | length | minLength | maxLength | encoding | period
| duration | enumeration | pattern)* | (element | group | all | choice
| sequence | any)*) , ((attribute | attributeGroup)* , anyAttribute?)))
</complexType>

W3C's *XML Schema* -- Part 1: Structures

Typdefinition -- *complex type*

- Abstrakte Typdefinition möglich; Typ darf nicht in XML-Dokumentinstanzen auftreten (*abstract*)
- Restriktion verschiedener Substitutionsmimiken (*block*)
- Vererbungs-Restriktion (*final*)
- Vererbung durch Typerweiterung oder -Einschränung (*derivedBy*)




```
<xsd:complexType name="ProjektVerwaltungType" content="elementOnly"
  abstract="false" block="restriction" final="restriction" derivedBy="Extension"
  base="Verwaltung">
   <xsd:element name="Person" minOccurs="1" maxOccurs="unbound"/>
   <xsd:element name="Projekt" minOccurs="1" maxOccurs="unbound"/>
</xsd:complexType>
```

W3C's *XML Schema* -- Part 1: Structures

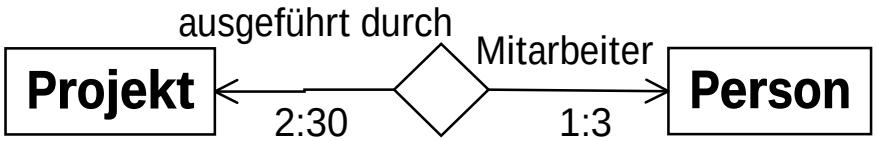
Typdefinition -- *element*

<element
abstract = boolean : false
block = *#all or (possibly empty) subset of {equivClass, extension, restriction}* : ''
default = string
equivClass = QName
final = *#all or (possibly empty) subset of {extension, restriction}* : ''
fixed = string
form = qualified | unqualified
id = ID
maxOccurs = string
minOccurs = nonNegativeInteger : 1
name = NCName
nullable = boolean : false
ref = QName
type = QName>
Content: (annotation? , (simpleType | complexType)? , (unique | key | keyref)*)
</element>

W3C's *XML Schema* -- Part 1: Structures

Typdefinition -- *element*

- (Polymorphie) Gleich benannte Elemente mit verschiedenen Inhaltsmodellen sind zulässig.



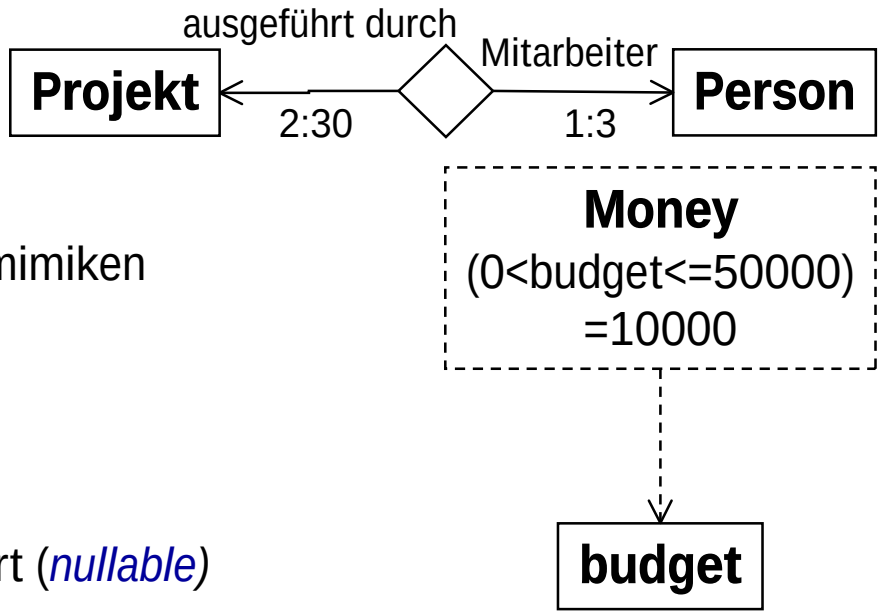
- Elementäquivalenz zu bestehenden Element (*equivClass*)
- Explizite Modalität und Kardinalität (*minOccurs*, *maxOccurs*)

```
<xsd:element name="Projekt" type="ProjektType"/>
<xsd:element name="Projekt" type="xsd:string"/>
<xsd:complexType name="Projekt">
  <xsd:element ref="Mitarbeiter" minOccurs="2" maxOccurs="30"/>
</xsd:complexType>
<xsd:element name="myComment" equivClass="comment"/>
```

W3C's *XML Schema* -- Part 1: Structures

Typdefinition -- *element*

- Abstrakte Elemente (*abstract*)
- Vorgabewerte (*default*)
- Restriktion verschiedener Substitutionsmimiken (*block*)
- Konstante Belegung (*fixed*)
- Vererbungs-Restriktion (*final*)
- Lukasiewicz- / Tri-State-Logik / NULL-Wert (*nullable*)
- Referenzierung auf bestehende Elemente (*ref*)




```
<xsd:element name="Mensch" abstract="true" block="restriction" />
<xsd:element name="budget" default="10000"/>
<xsd:element ref="Person"/>
```

W3C's *XML Schema* -- Part 1: Structures

Nullwerte

XML-Schema unterstützt kein explizites Symbol für fehlende Werte (*NULL).* Um die Unterstützung fehlender Werte zuzulassen muß das entsprechende Element im Schema mit *nullable="true"* attributiert werden.

```
<xsd:element name="nEl" nullable="true"/>
```

Im XML-Dokument wird das in XML-Schema vordefinierte Attribut *null* auf *true* gesetzt.
Der explizite *NULL*-Wert ist Teil des XML-Schema-Namensraums für XML-Schema Instanzen (*instance documents)* (http://www.w3.org/1999/XMLSchema-instance). Das Namespacepräfix *xsi* ist zwingend.

```
<nEl xsi:null="true"/>
```

W3C's *XML Schema* -- Part 1: Structures

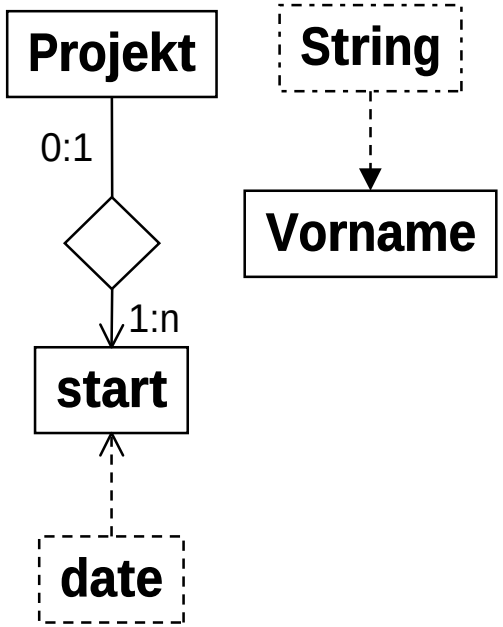
Typdefinition -- *attribute*

<attribute
form = qualified | unqualified
id = ID
name = NCName
ref = QName
type = QName
use = default | fixed | optional | prohibited | required : optional
value = string>
Content: (annotation? , simpleType?)
</attribute>

W3C's *XML Schema* -- Part 1: Structures

Typdefinition -- *attribute*

- Typisierung gemäß *XML Schema Part 2* (*type*)
- Vorgabewerte (use=*default*)
- Konstante Belegung (use=*fixed*)
- Optionale Vergabe (use=*optional*)
- Zwingende Vergabe / *mandatory* (use=*required*)
- Untersagung der Verwendung (use=*prohibited*)
- Referenzierung auf bestehendes Attribut (*ref*)
- Mengenwertige Attribute (*minOccurs*, *maxOccurs*)



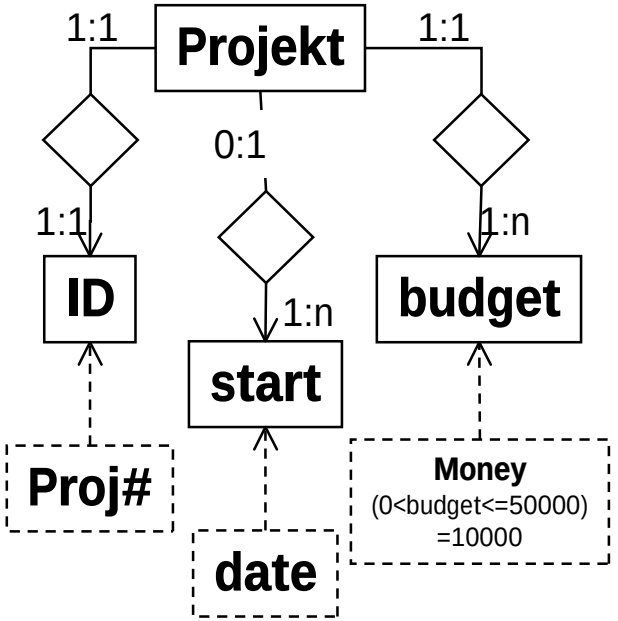

```
<xsd:attribute name="Vorname" type="dt:string" />
<xsd:attribute name="start" type="dt:date" use="optional" />
```

W3C's *XML Schema* -- Part 1: Structures

Typdefinition -- *attribute group*

- Zusammenfassung von Attributen zu einer benannten Gruppe
- Verbessert Les- und Wartbarkeit des Schemas durch Zentralisierung der Information

(ähnlich den *parameter entities* in XML 1.0)




```
<xsd:complexType name="Projekt">
  ...
  <attributeGroup ref="Project'sAttributes"/>
</xsd:complexType>
<xsd:attributeGroup name="Project'sAttributes">
  <xsd:attribute name="ID" type="ProjectNo"/>
  <xsd:attribute name="start" type="xsd:date"/>
  <xsd:attribute ref="BudgetAtt"/>
</xsd:attributeGroup>
```

W3C's *XML Schema* -- Part 1: Structures

Typdefinition -- *annotation*

- Explizite Dokumentationsmöglichkeit
- Informale Beschreibung für (meschliche) Schemanutzer (*documentation*)
- Formale Beschreibung für schemaverarbeitende Applikationen (*appInfo*)
- Werden im Validierungsprozess nicht berücksichtigt

```
<xsd:annotation>
   <xsd:appinfo source="http://www...."/>
    <xsd:documentation xml:lang="en-en">
       Schema created 2000-06-06 by ...
     </xsd:documentation
</xsd:annotation>
<xsd:annotation>
  <xsd:appinfo>
     <dcx:creator>...</dcx:creator><dcx:version>1.0</dcx.version>
</xsd:annotation>
```

W3C's *XML Schema* -- Part 2: Datatypes

- Zunehmend *datenorientierte* Betrachtung erfordert ein leistungsfähiges Typsystem zur Darstellung von "alltäglichen" Standardsituationen
- Erhöhung des Qualitätsgrades der entstehenden XML-Dokumente durch semantische Anreicherung
- Erweiterung generischer Parser um zusätzliche inhaltliche Prüfung
- Verminderung des *impedance mismatch* zwischen Anwendungssystem (Programmiersprache, Datenbanksystem, ...) und Streaming-Format
- Typsystem auf Basis bekannter Standards wie ISO 11404, SQL, Java
- Charakterisierung der Typen in
  - atomare vs. Listentypen
    (atomic vs. list types)
  - primitive vs. abgeleitete Typen
    (primitive vs. derived types)
  - Standardtypen vs. Andwenderdefinierte
    (built-in vs. user-derived datatypes)

W3C's *XML Schema* -- Part 2: Datatypes

Datentypcharakterisierung

Atomare Datentypen: Unteilbare Werte *(z.B. string)*

Listentypen: Endliche Menge atomarer Typausprägungen

(z.B. list of integer)

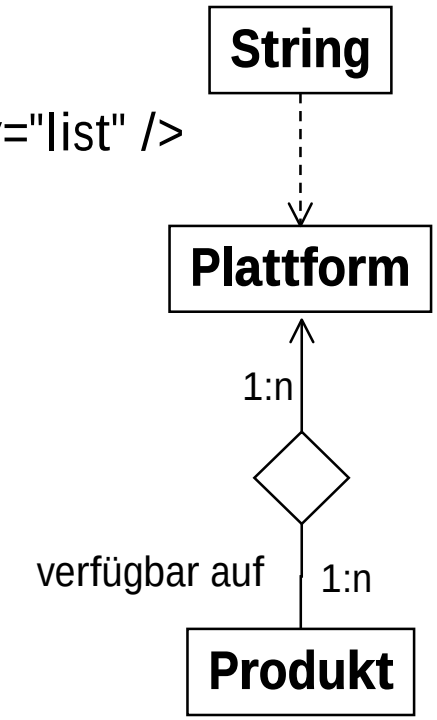
Definition:

```
<simpleType name="Plattformen" base="string" derivedBy="list" />
```

Verwendung (in XML-Dokumentinstanz)**:**

```
<verfügbarePlattformen xsi:type="Plattformen">
  Windows  HP-UX  AIX  Linux
</verfügbarePlattformen>
```

XML *whitespaces* (0x9, 0x20, 0xA, 0xD) gelten als Element-separatoren (=> Vorsicht bei Strings mit Leerzeichen)



W3C's *XML Schema* -- Part 2: Datatypes

Datentypcharakterisierung

Standardtypen: in der XML Schema 2-Spezifikation definierte

Anwenderdefinierte Typen: Abgeleitete Einschränkung bestehenderTypen

- Bezug auf Basistypen durch das *base*-Attribut
- Basistypen können selbst wieder primitiv oder abgeleitet sein (-> Typhierarchie)

```
<xsd:simpleType name="urSimpleType" base="urSimpleType" />
<xsd:simpleType name="decimal" base="urSimpleType" />
<xsd:simpleType name="integer" base="decimal" />
<xsd:simpleType name="nonPositiveInteger" base="integer" />
```

W3C's *XML Schema* -- Part 2: Datatypes

Datentypcharakterisierung

build-in Datentypen: Unabhängig von anderen Typen definiert

abgeleitete Datentypen: (abhängige) Definition unter Verwendung bereits definierter Datentypen

W3C's *XML Schema* -- Part 2: Datatypes

Build-in datatypes

- ID
- IDREF
- IDREFS
- ENTITY
- ENTITIES
- NMTOKEN
- NMTOKENS
- Name
- QName
- NCName
- NOTATION
- integer
- nonPositiveInteger
- negativeInteger
- long
- int
- short
- byte
- nonNegativeInteger
- unsignedLong
- unsingnedInt
- unsignedShort
- unsignedByte
- positiveInteger
- boolean
- float
- double
- decimal
- string
- date
- time
- timeInstant
- timePeriod
- month
- year
- century
- recurringDate
- recurringDay
- timeDuration
- recurringDuration
- binary
- uriReference
- language

W3C's *XML Schema* -- Part 2: Datatypes

• integer	-1, 0, 7683, +555
• nonPositiveInteger	{..., -2, -1, 0}
• negativeInteger	{..., -3, -2, -1}
• long	-2^{63} <= long <= $2^{63}-1$
• int	-2^{31}<= int <= $2^{31}-1$
• short	-2^{15} <= short <= short $2^{15}-1$
• byte	-2^{7} <= byte <= $2^{7}-1$
• nonNegativeInteger	{0,1,2,...}
• positiveInteger	{1,2,3,...}
• unsignedLong	0 <= unsignedLong <= $2^{64}-1$
• unsingnedInt	0 <= unsignedInt <= $2^{32}-1$
• unsignedShort	0 <= unsignedShort <= $2^{16}-1$
• unsignedByte	0 <= unsignedByte <= $2^{8}-1$
• boolean	{true, 1, false, 0}
• float	32-Bit Fließkommazahl gemäß IEEE 754-1985 -1E4, 12.64E8, 12e-2, INF
• double	64-Bit Fließkommazahl gemäß IEEE 754-1985
• decimal	-12, 8, 3.14151592, +1.0

W3C's *XML Schema* -- Part 2: Datatypes

• string	ISO 10646 und Unicode	"hello world"
• date	ISO 8601	2000-06-06
• time	ISO 8601	09:00:00+2:00
• timeInstant	ISO 8601	2000-06-06T09:00:00+2:00
• timePeriod	ISO 8601	P7M
• month	ISO 8601	P1M
• year	ISO 8601	P1Y
• century	ISO 8601	19
• recurringDate	ISO 8601	PT24H
• recurringDay	ISO 8601	P1M2D
• timeDuration	ISO 8601	P0Y0M0DT1H30M
• recurringDuration	*abstrakter Supertyp von* duration *und* period*; nicht direkt verwendbar!*	
• binary	*abstrakter Typ; nur verwandbar durch Ableitung und* encoding-*Spezifikation*	
• uriReference	IETF RFC2396	http://www.jeckle.de
• language	IETF RFC1766	de-de, en-uk, x-klingon

W3C's *XML Schema* -- Part 2: Datatypes

Typdefinition für...

...Elemente

```
<xsd:element name="elementName" type="TypeName" />
```

- Elementtyp kann als *complexType* gemäß Schema 1 strukturiert sein, oder gemäß Schema 2 *build-in* bzw. selbstdefiniert sein

...Attribute

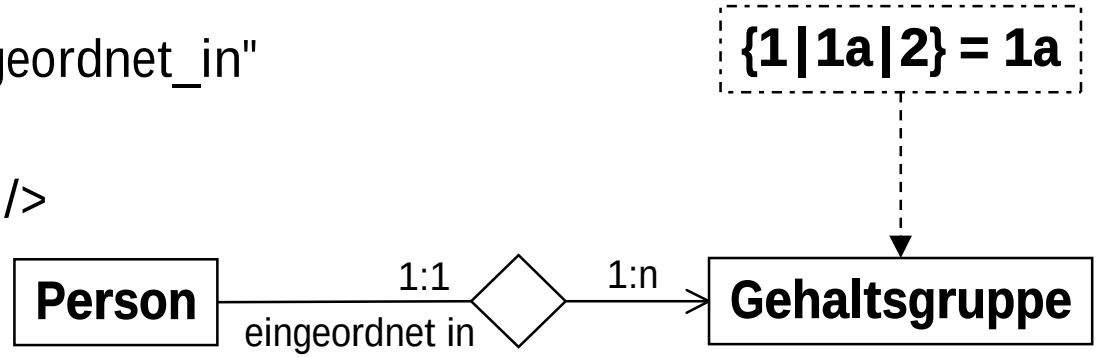
```
<xsd:attribute name="AttributeName" type="TypeName" />
```

- Attributtyp kann gemäß Schema 2 *build-in* bzw. selbstdefiniert sein
  => Wie in XML v1.0 üblich, keine (explizite) Strukturierung durch Markup innerhalb Attributen zugelassen

W3C's *XML Schema* -- Part 2: Datatypes

Typdefinition -- *enumeration*

```
...
  <xsd:attribute name="eingeordnet_in"
   type="Gehaltsgruppe"
    value="1a" use="default" />
...
```

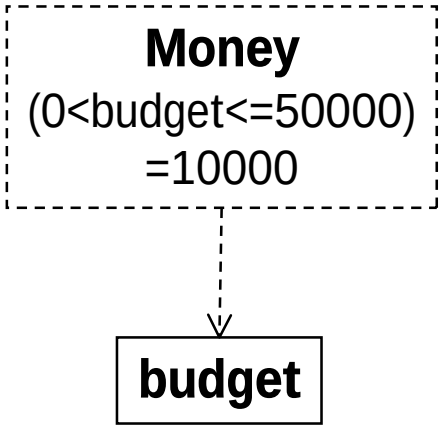



```
<xsd:simpleType name="Gehaltsgruppe" base="xsd:string">
  <xsd:annotation>
      <xsd:documentation>XY AG Gehaltsgruppen per 1999-12-31</xsd:documentation>
  </xsd:annotation>
  <xsd:enumeration value="1" />
  <xsd:enumeration value="1a" />
  <xsd:enumeration value="2" />
</xsd:simpleType>
```

W3C's *XML Schema* -- Part 2: Datatypes

Typdefinition -- Typableitung

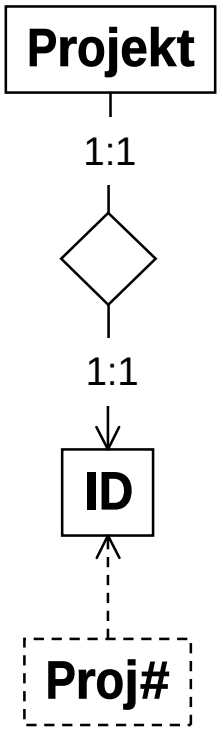
- Erweitung des bestehenden Typsystems durch anwenderdefinierte Typen.




```
<xsd:attribute name="budgetAtt" use="default" value="10000">
  <xsd:simpleType base="dt:float">
    <xsd:minExclusive value="0"/>
    <xsd:maxInclusive value="50000"/>
    <xsd:scale value="2"/>
  </xsd:simpleType>
</xsd:attribute>
```

W3C's *XML Schema* -- Part 2: Datatypes

Typdefinition -- Lexikalische Definition



- Spezifikation des Wertebereichs eines Datentyps durch reguläre Ausdrücke

```
<xsd:simpleType name="ProjektNo">
  <pattern value="P-\(19|20)\Nd{2,}-\Lu+\Nd{3,5}"/>
</xsd:simpleType>
```

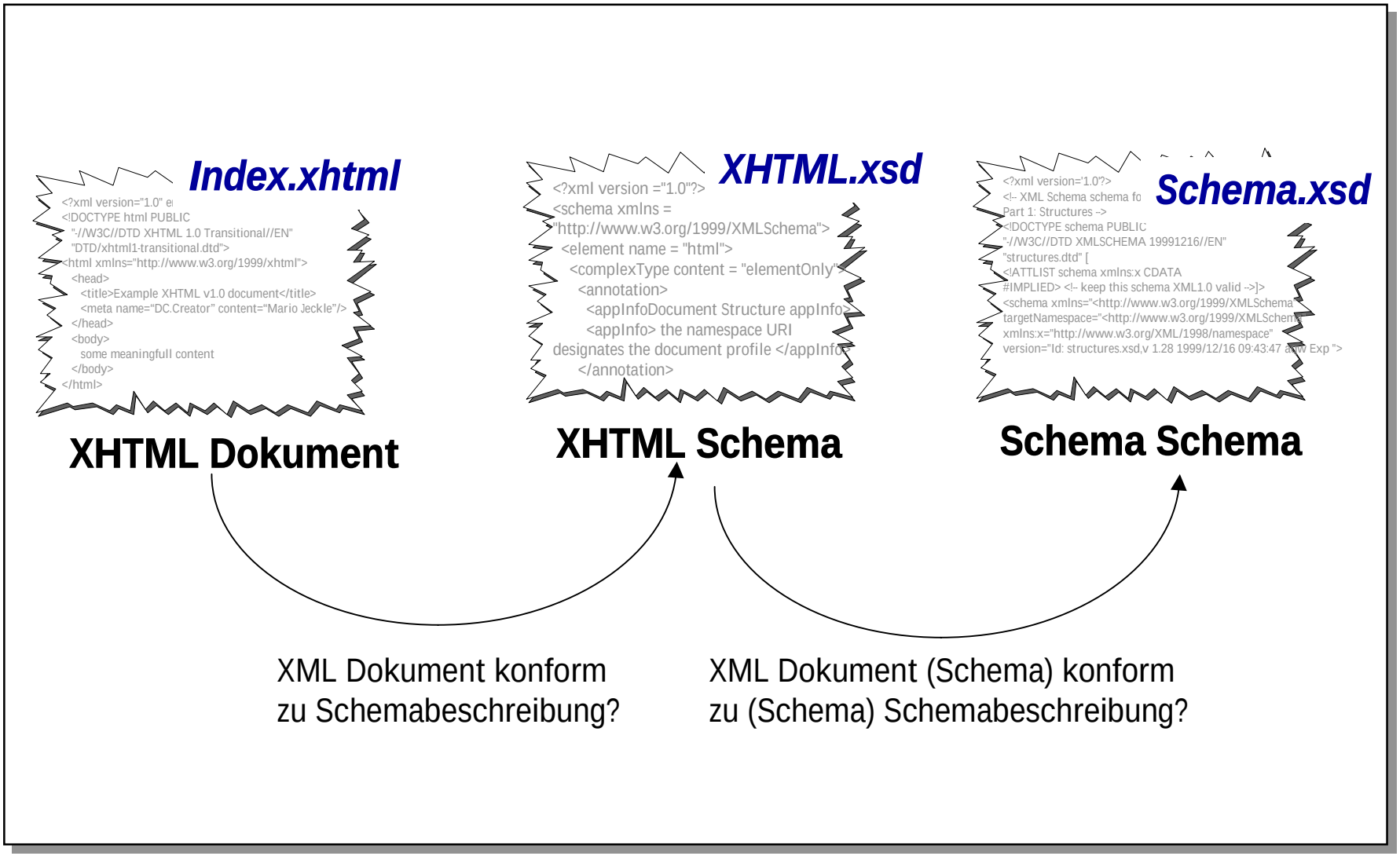
Projektnummern beginnen stets mit **P**, darauf folgt das Jahr, abgetrennt durch einen Bindestrich eine mindestens aus einem Großbuchstaben (*Lu*) bestehende Identifikation, an die sich mindestens drei, aber höchstens fünf numerische Ziffern anschließen.

Schema validness

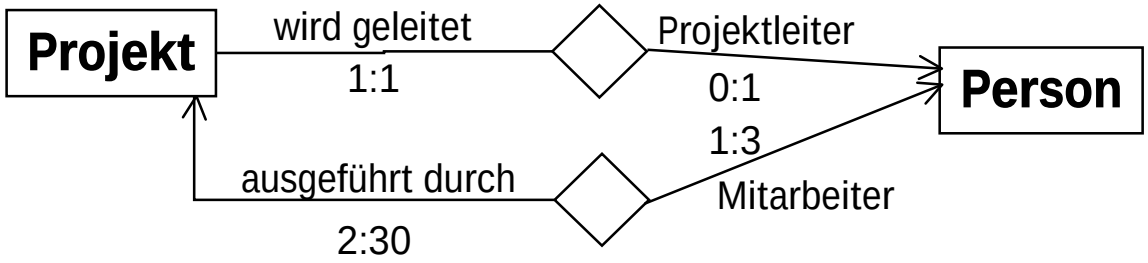
Aufbauend auf der generellen Konformität (*valdiness)* zu einer existierenden DTD wird die *schema validness* definiert:

- XML-Instanz referenziert "ihr" Schema (namespace-Deklaration im Root-Element)
- *XML-Schema-instance* Namespace (falls verwendet)
- Gültigkeitseinschränkungen an Schema Elemente
  (z.B. minOccurs <= maxOccurs)
- Konform zu *Schema for Schemas*

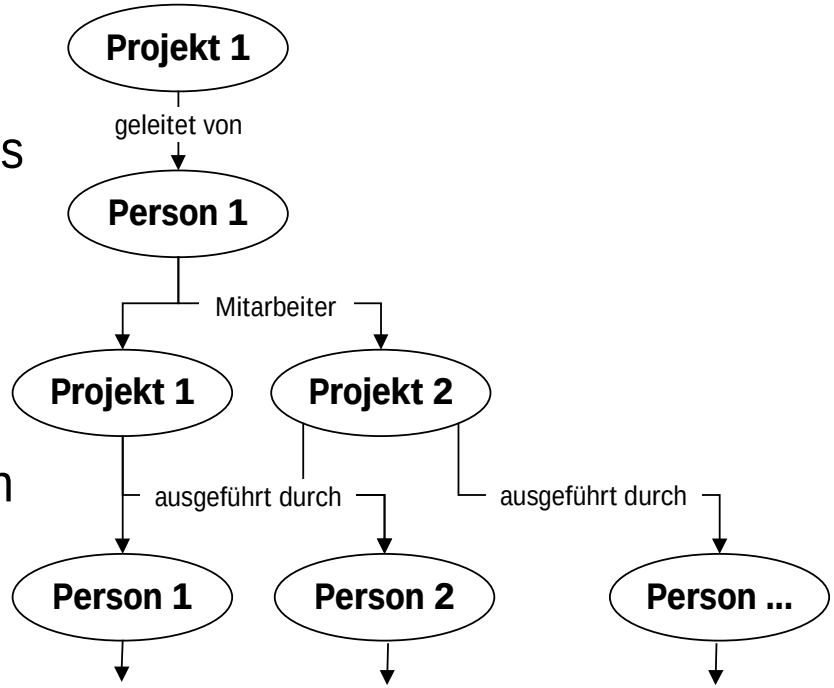
Schema validness



Grenzen der XML Schema Description Language

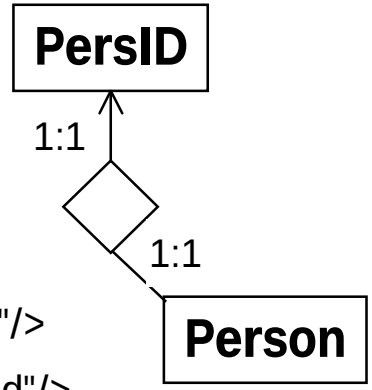


- Definition und Ausdruck nicht-hierarchischer Strukturen (Redundanzproblem durch mehrfaches Auftreten benötigter Information)
- Zirkuläre oder rekursive Strukturen
- => Strenge Hierarchisierung schränkt darstellbare Datenstrukturen a priori ein

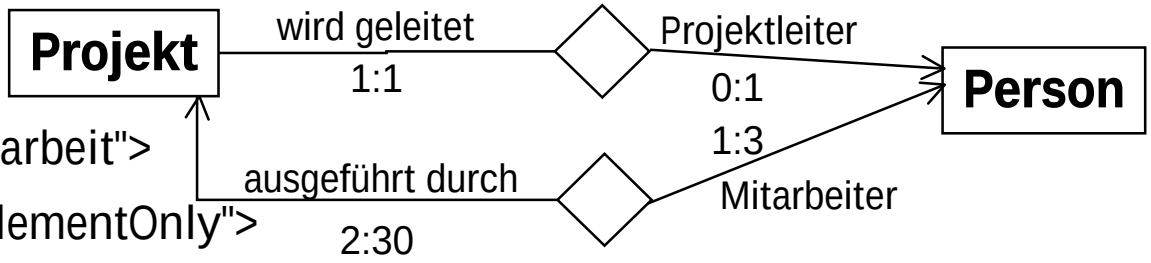


Schlüsselbeziehungen

```
<element name = "PersonType">
  <complexType content = "elementOnly">
    <sequence>
      <element ref = "Vorname" minOccurs = "1" maxOccurs = "unbounded"/>
      <element ref = "Nachname" minOccurs = "1" maxOccurs = "unbounded"/>
    </sequence>
    <attribute name = "eingeordnet_in" use = "fixed" value = "1a" type = "GehaltsgruppeType"/>
    <attribute name = "PersonalID" type = "PersIDType"/>
    <key name="PersonKey">
      <selector>Person</selector>
      <field>@PersonalID</field>
    </key>
  </complexType>
</element>
```



Schlüsselbeziehungen




```
<element name = "Projektmitarbeit">
  <complexType content = "elementOnly">
    <sequence>
      <element ref = "ausgefuehrtes_Projekt" minOccurs = "1" maxOccurs = "3">
        <keyref refer="ProjektKey"/>
      </element>
      <element ref = "Projektmitarbeiter" minOccurs = "1" maxOccurs = "30">
        <keyref refer="PersonKey"/>
      </element>
    </sequence>
  </complexType>
</element>
```

Zusammenfassung

- *XML Schema* bilden hinsichtlich Mächtigkeit eine Obermenge der XML DTD-Ausdrucksfähigkeit
- *XML Schema* stellen einen fundierten und zukunftsfähigen Ansatz zur Beschreibung von XML-Sprachen in Bezug auf Struktur und Datentypen dar
- Mittelfristig werden sie parallel zum XML v1.0 DTD-Mechanismus existieren und ergänzend eingesetzt werden
- Langfristig ist mit der breiten Marktakzeptanz, und als Konsequenz der Ablösung des bestehenden DTD-Mechanismus, der Schema-Sprachen zu rechnen
- *XML Schema* ist eine Sprache zur Beschreibung hierarchischer XML-Daten- und -Dokumentstrukturen, jedoch zur Datenmodellierung nur bedingt geeignet; *XML Schema* ist keine (*general purpose)* Datenmodellierungssprache
- Erste Toolimplementierungen liegen vor
- Mit der Verabschiedung (W3C Recommendateion) ist Q1 / 2000 zu rechnen

Einordnung in die XML-Standardfamilie / Ausblick

- *XML Schema* ist als XML-Sprache eine Anwendung der XML
- Jedes Schema ist ein XML-Dokument
- *Schema for Schema* ist selbst ein *schema-valid* XML-Dokument
- Nicht-hierarchische Strukturen sinnvollst mit XLL (XPointer, XLink, XPath) ausdrückbar
- Schemata für Modellierungssprachen und Datenmodelle (z.B. Austauschformat für UML-Diagramme) sinnvollst mit OMG's *XML Metadata Interchange* (XMI) ausdrückbar

Empfehlungen zum praktischen Einsatz

- Information die in verschiedenen Rollen auftritt sollte (generell) als *complexType* definiert werden, um die Wiederverwendbarkeit zu erhöhen
- Wenn möglich...
  - spezialisierte Datentypen nutzen
  - eigene Datentypen ableiten
  - reguläre Ausdrücke
  - Aufzählungstypen
- Bidirektional navigierbare Beziehungen mit Kardinalitätsanteil größer Eins in beiden Richtungen (sog. *n:m-Beziehungen*) in separate Elemente aufbrechen

XML-Schema des Beispiels

```
<?xml version ="1.0"?>
<schema xmlns = "http://www.w3.org/1999/XMLSchema">
<element name = "Projektverwaltung">
    <complexType content = "elementOnly">
      <sequence>
        <element ref = "Projekt" minOccurs = "1" maxOccurs = "unbounded"/>
        <element ref = "Person" minOccurs = "1" maxOccurs = "unbounded"/>
        <element ref = "Projektmitarbeit"/>
      </sequence>
    </complexType>
  </element>
  <element name = "Projekt">
    <complexType content = "elementOnly">
      <sequence>
        <element ref = "Projektleiter"/>
      </sequence>
      <attribute name = "ID" use = "required" type = "ProjektNo"/>
      <attribute name = "start" type = "date"/>
      <attribute name = "budget" use = "fixed" value = "10000" type = "budgetMoney"/>
      <key name=“ProjektKey”><selector>Projekt><field>@ID</field><key>
    </complexType>
  </element>
```

XML-Schema des Beispiels *(cont'd)*

```
<simpleType name = "ProjektNo" base = "string">
  <pattern value = "P-\(19|20)\Nd{2,}-\Lu+\Nd{3,5}"/>
</simpleType>

<simpleType name = "budgetMoney" base = "float">
  <minExclusive value = "0"/>
  <maxInclusive value = "50000"/>
  <scale value = "2"/>
</simpleType>

<simpleType name = "GehaltsgruppeType" base = "string">
  <enumeration value = "1"/>
  <enumeration value = "1a"/>
  <enumeration value = "2"/>
</simpleType>

<simpleType name = "PersIDType" base = "string">
  <pattern value = "h-\Nd{2,}-\Nd{5,}"/>
</simpleType>
```

XML-Schema des Beispiels *(cont'd)*

```
<element name = "Person">
  <complexType content = "elementOnly">
    <sequence>
      <element ref = "PersonType"/>
    </sequence>
  </complexType>
</element>

<element name = "PersonType">
  <complexType content = "elementOnly">
    <sequence>
      <element ref = "Vorname" minOccurs = "1" maxOccurs = "unbounded"/>
      <element ref = "Nachname" minOccurs = "1" maxOccurs = "unbounded"/>
    </sequence>
    <attribute name = "eingeordnet_in" use = "fixed" value = "1a" type = "GehaltsgruppeType"/>
    <attribute name = "PersonalID" type = "PersIDType"/>
    <key name="PersonKey">
      <selector>Person</selector>
      <field>@PersonalID</field>
    </key>
  </complexType>
</element>
```

XML-Schema des Beispiels *(cont'd)*

```
<element name = "Vorname" type = "string"/>
<element name = "Nachname" type = "string"/>

<element name = "Projektmitarbeit">
<complexType content = "elementOnly">
    <sequence>
      <element ref = "ausgefuehrtes_Projekt" minOccurs = "1" maxOccurs = "3">
        <keyref refer="ProjektKey"/>
      </element>
      <element ref = "Projektmitarbeiter" minOccurs = "1" maxOccurs = "30">
        <keyref refer="PersonKey"/>
      </element>
    </sequence>
  </complexType>
</element>

<element name = "Projektleiter">
  <complexType content = "elementOnly">
    <sequence>
      <element ref = "PersonType"/>
    </sequence>
  </complexType>
</element>
```

XML-Schema des Beispiels *(cont'd)*

```
  <element name = "Projektmitarbeiter">
    <complexType content = "elementOnly">
      <sequence>
        <element ref = "PersonType"/>
      </sequence>
    </complexType>
  </element>
</schema>
```

References

W3C's XML-Schema:

www.w3.org/TR/NOTE-xml-schema-req (*XML schema requirements*)

www.w3.org/TR/xml-schema-0
www.w3.org/TR/xml-schema-1
www.w3.org/TR/xml-schema-2

Alternativvorschläge:

www.w3.org/TR/dt4dtd
www.w3.org/TR/NOTE-dcd
www.w3.org/TR/NOTE-ddml
www.brics.dk/DSD/
www.w3.org/TR/NOTE-SOX/
www.ascc.net/xml/resource/schematron/schematron.html
www.w3.org/TR/1998/NOTE-XML-data-0105

References

Sekundärliteratur:

www.w3.org/TR/schema-arch

www.lindamann.com/xml/XML%20Schemas%20NG%20Guide%20HTML.htm
xml.com/pub/2000/02/23/xmldeviant/index.html?wwwrrr_20000223.txt
www.w3.org/TR/NOTE-xml-schema-req
www.iso.ch/cate/d19346.html *(ISO 11404)*

Werkzeuge:

www.alphaworks.ibm.com/formula/xml

xml.apache.org (*Xerces*)

www.extensibility.com (*XML Authority*)

Dieser Vortrag und weiterführende Information:

www.jeckle.de

References

Tangierte und weiterführende XML-Literatur:

www.w3.org / TR / REC-xml *(XML v1.0 Recommendation)*
www.w3.org / TR / REC-xml-names
www.w3.org / TR / xml-infoset

Dieser Vortrag und weiterführende Information:

www.jeckle.de