



UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

■ **Edirom Summer School 2010**

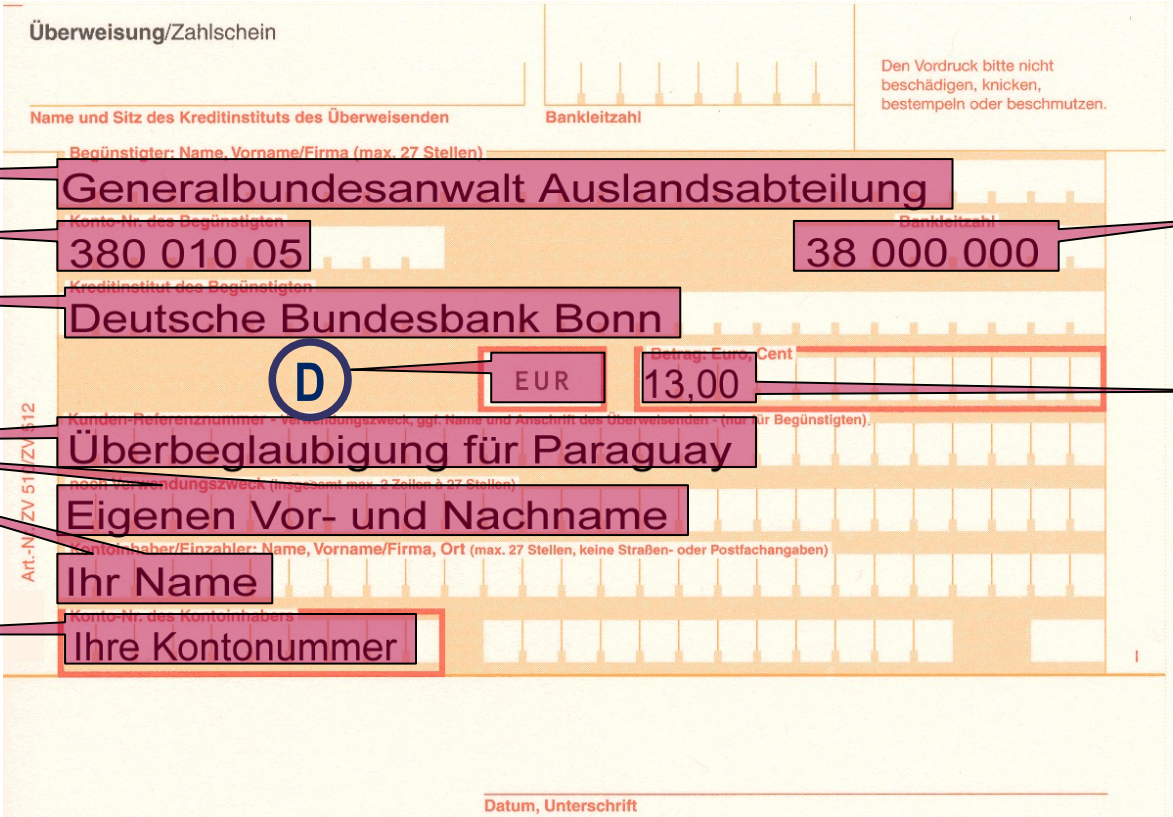
Workshop: XML und XML-Technologien
21.-24. November 2010

■ **M.Sc. Julian Dabbert**

Fakultät für Kulturwissenschaften der Universität Paderborn

Block 3: Schemata

Das Prinzip Schema



Überweisung/Zahlschein

Den Vordruck bitte nicht beschädigen, knicken, bestempeln oder beschmutzen.

Name und Sitz des Kreditinstituts des Überweisenden

Bankleitzahl

Art-Nr. ZV 51 / ZV 112

Begünstigter: Name, Vorname/Firma (max. 27 Stellen)

Konto-Nr. des Begünstigten

Kreditinstitut des Begünstigten

Betrag: Euro, Cent

Kunden-Referenznummer - Verwendungszweck, ggf. Name und Anschrift des Überweisenden (max. 27 Stellen für Begünstigten)

nach Verwendungszweck (insgesamt max. 2 Zeilen à 27 Stellen)

Kontoinhaber/Einzahler: Name, Vorname/Firma, Ort (max. 27 Stellen, keine Straßen- oder Postfachangaben)

Datum, Unterschrift

A: Zeichenkette
B: Ziffernfolge
C: Zeichenkette impliziert aus B)
D: Währungskürzel aus Großbuchstaben
E: Fließkommazahl

A:Zeichenkette B:Ziffernfolge C:Zeichenkette impliziert aus B)

D:Währungskürzel aus Großbuchstaben E: Fließkommazahl

Block 3: Schemata

Schema für XML

Ein Schema ist eine Vorschrift, wie ein XML Dokument aufgebaut wird

- Ein XML-Dokument verweist i.d.R. im Kopf auf ein Schema

```
<?xml version="1.0" standalone="no"?>  
<!DOCTYPE person SYSTEM  
"http://www.cafeconleche.org/dtds/person.dtd">
```

- Mit einem Validator (Programm) lässt sich ermitteln, ob das Dokument seinem angegebenen Schema entspricht (valide ist)
- Ein valides Dokument kann von einem Programm verstanden werden, das Eingaben gemäss dem Schema erwartet

■ Block 3: Schemata

■ Leben ohne Schema

Es geht auch ohne Schema („standalone“), aber dann...

```
<?xml version="1.0" standalone="yes"?>
```

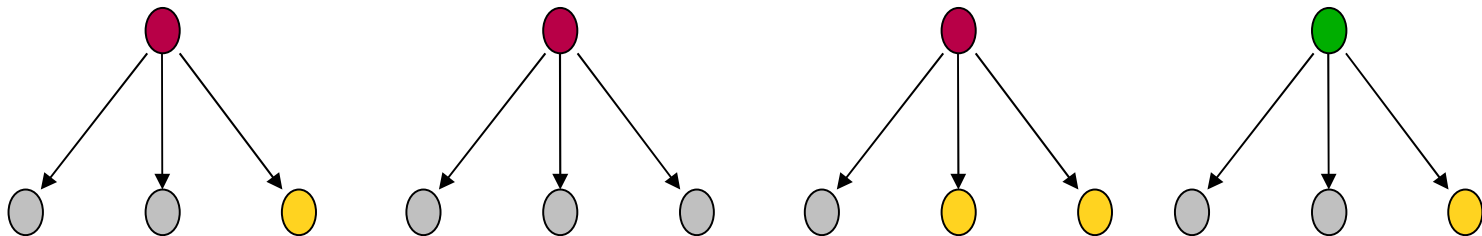
- Ein Parser kann zwar den Baum des Dokuments aufbauen
- Die Datentypen der Werte kann er aber nur vermuten
(id="200" → Ist das eine Zeichenkette oder eine Zahl?)
- Dass kein Mensch zwei Sozialversicherungsnummern besitzt, hat ihm niemand gesagt und daher fällt ihm der Fehler nicht auf
- Kein Programm erwartet seine Dateneingabe in dieser strukturlosen Form, daher sind die Daten nicht interpretierbar

■ Block 3: Schemata

■ Übung: Einfache Schemaklausel

Beispiel: XML Dokumente mit Knoten „block_3“ und „summerSchool“

- Jede summerSchool kann einen block_3 unter sich haben
- Es kann pro summerSchool max. einen block_3 geben
- block_3 darf auch unter anderen Kursen vorkommen
- Mathematisch: „summerSchool“ hat 0-1 Kinder „block_3“
- Welcher dieser Bäume ist also zulässig?



Block 3: Schemata

Schemasprache

Struktur und Inhalt von XML Dokumenten werden über Sprachen ausgedrückt, die sogenannten Schemasprachen

- Je nach Sprache können unterschiedliche Mengen an zulässigen Konstrukten ausgedrückt werden
- Schemasprachen i.d.R selbst in XML ausgedrückt
- Es gibt viele Schemasprachen, hier werden behandelt:
„XML Schema“ (weitverbreitet, eher mächtig),
„DTD“ (alt, eher schwach) sowie
„RelaxNG“ (neu, sehr mächtig)

■ Block 3: Schemata

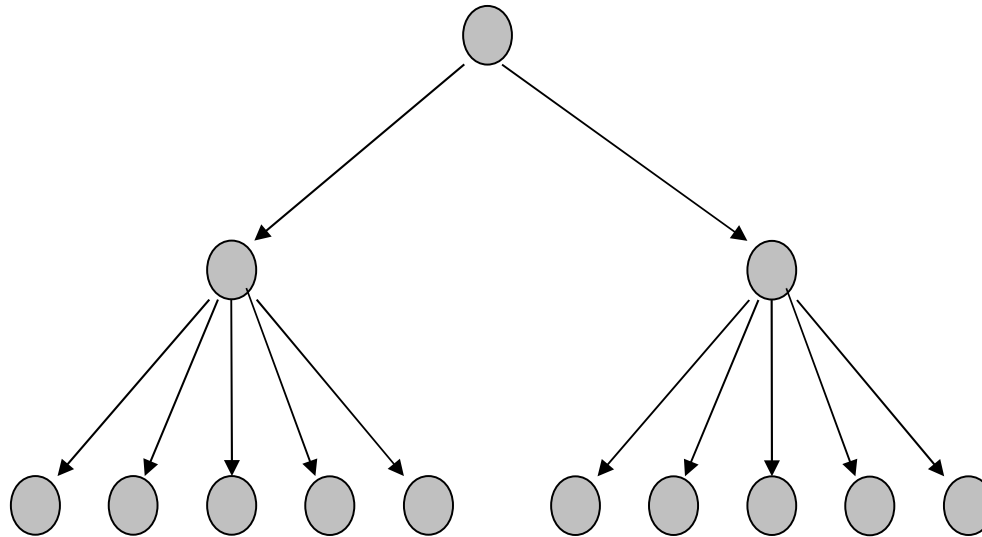
■ Übung: XML Baum mit informellem Schema schreiben

Das Schema ist nicht mehr beliebig, sondern es soll gelten:

- Die Wurzel bildet der Personalausweis
- Darunter zwei Knoten: Vorder- und Rückseite
- Unter der jeweiligen Seite als Kinder einfügen:
 - Name
 - Vorname
 - Geburtstag und -ort
 - Staatsangehörigkeit
 - Ablaufdatum
 - Anschrift
 - Grösse
 - Augenfarbe
 - Behörde
 - Datum

■ Block 3: Schemata

■ Lösung: XML Baum mit informellem Schema schreiben



■ Block 3: Schemata

■ Übung: XML Dokument aus Baumschema ableiten

Aus dem Schemabaum wird nun ein konkretes XML-Dokument abgeleitet:

- Wurzel heisst „personalausweis“
- Darunter zwei Elemente „vorderseite“ und „rueckseite“
- Unter der jeweiligen Seite als Elemente einfügen:
 - name
 - vorname
 - geburtstag_und_ort
 - staatsangehoerigkeit
 - ablaufdatum
 - anschrift
 - groesse
 - augenfarbe
 - behoerde
 - datum

■ Block 3: Schemata

■ Konstruktionsabsicht eines Schemas

Für denselben Datenbestand lassen sich unterschiedliche Schemata konstruieren.

- Die Struktur kann sich je nach Zweck anders gestalten
(Warum kann es wichtig sein, dass die Grösse im PA unter den Knoten für die Rückseite einsortiert wird?)
- Welche Daten des verfügbaren Bestandes in das Schema aufgenommen werden, wird durch Modellierung entschieden
(Ist es wichtig, dass die PA-Nummer nicht in unserem Schema ist?)
- Ein Schema sollte immer der Anwendung zuträglich sein, für die es entworfen wird!

■ Block 3: Schemata

■ Wirkung eines Schemas

Ein Schema legt fest:

- **Position eines Knotens**
 - **Eltern des Knotens**
 - **Zulässige Kinder und Attribute**
- **Anzahl / Optionalität eines Knotens**
 - **Maximal zulässiges Vorkommen**
 - **Minimal erforderliches Vorkommen**
- **Datentyp seiner Werte**

■ Block 3: Schemata

■ Datentypen

Die Festlegung des Datentyps ist wichtig für die Interpretation des Werts

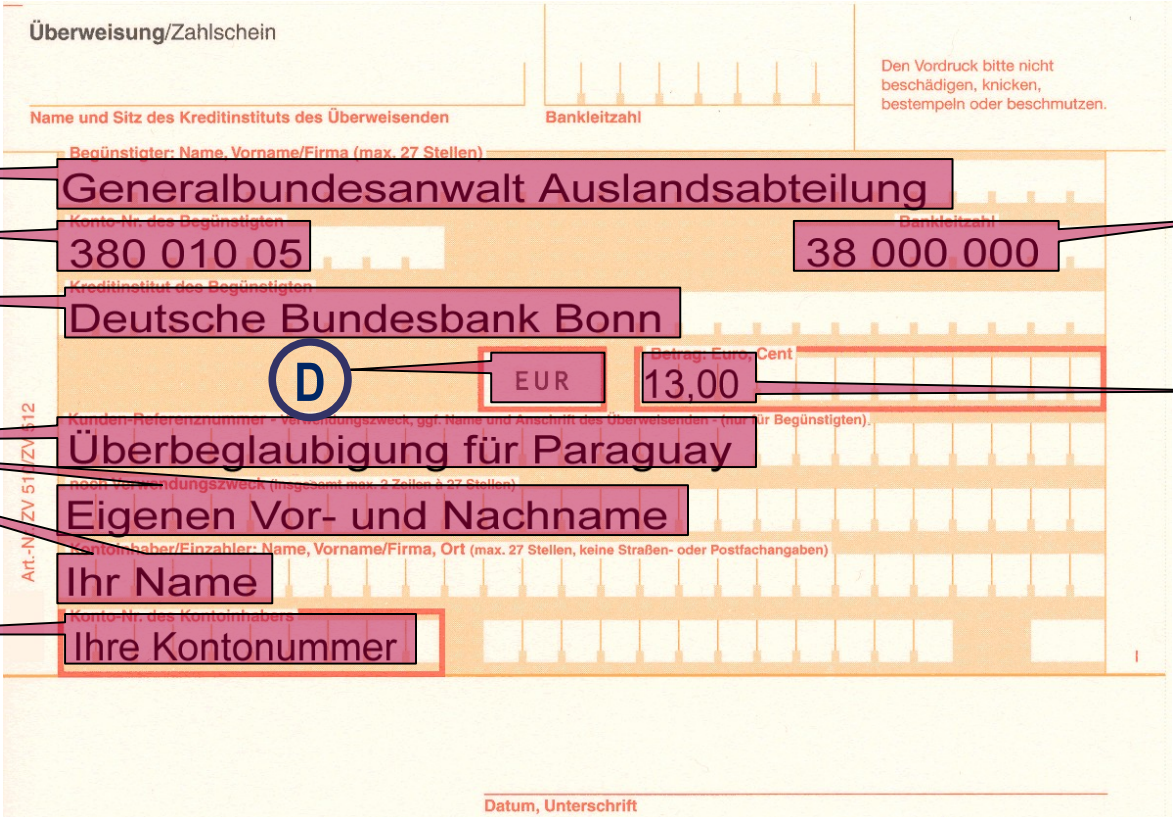
- **Grösse (in cm oder m?)**
- **Augenfarbe (ist „pflaume“ zulässig?)**
- **Geburtstag (amerikanisch: 09-22-2010 oder metrisch: 22.09.2010?)**

Daher kann der zulässige Typ eines Wertes im Schema definiert werden

- **Wichtigste Datentypen:**
Zeichenkette, Ganzzahl, Fließkommazahl, Wahrheitswert
- **Manche Schemasprachen lassen komplexere Strukturen zu**
(Geburtstag und -Ort: Datum gefolgt von Zeichenkette)

Block 3: Schemata

Bedingungen



The diagram shows a German bank transfer form (Überweisung/Zahlschein) with various fields. Annotations A through E point to specific fields:

- A** points to the field for the beneficiary's name (Begünstigter: Name, Vorname/Firma (max. 27 Stellen)).
- B** points to the field for the beneficiary's account number (Konto-Nr. des Begünstigten).
- C** points to the field for the beneficiary's bank (Kreditinstitut des Begünstigten).
- D** points to the field for the currency (Währungskürzel).
- E** points to the field for the amount (Betrag: Euro, Cent).

The form includes the following fields and labels:

- Überweisung/Zahlschein
- Name und Sitz des Kreditinstituts des Überweisenden
- Bankleitzahl
- Begünstigter: Name, Vorname/Firma (max. 27 Stellen)
- Konto-Nr. des Begünstigten
- Kreditinstitut des Begünstigten
- Betrag: Euro, Cent
- Kunden-Referenznummer - Verwendungszweck, ggf. Name und Anschrift des Überweisenden (max. 27 Stellen)
- Eigenen Vor- und Nachname
- Kontoinhaber/Einzahler: Name, Vorname/Firma, Ort (max. 27 Stellen, keine Straßen- oder Postfachangaben)
- Ihr Name
- Konto-Nr. des Kontoinhabers
- Ihre Kontonummer
- Datum, Unterschrift

A:Länge max. 27 B:Referenz aus Ganzzahl C:Zeichenkette impliziert aus B)

D:Währungskürzel aus Liste zul. Werte E: Fließkommazahl mit =2 Nks

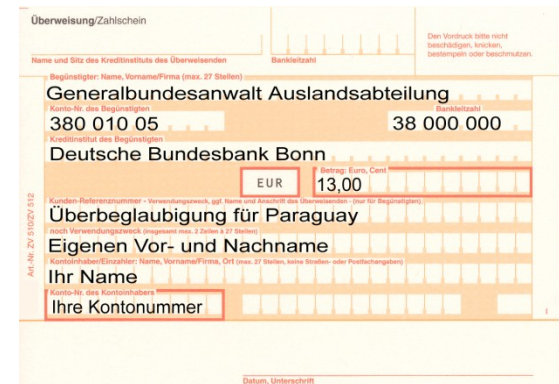
Block 3: Schemata

Grenzen der Schemaprüfung

Ein Schema kann solche Bedingungen gut prüfen, die sich auf die Form beziehen.

Wird jedoch Steuerungslogik verlangt, so ist ein Schema unpassend.

Analogie Überweisungsträger: Ein Bankangestellter kann durch kurzen Blick prüfen, ob der ausgefüllte Träger „gut aussieht“, also dem „Schema Überweisungsträger“ entspricht. Ob aber die Kontonummer zur Bankleitzahl und dem Kontoinhaber passt, kann er nicht erkennen.



■ Block 3: Schemata

■ Schemasprache „XML Schema“ (XSD)

Die Sprache XML Schema wurde im Mai 2001 vom
World Wide Web Consortium (W3C) als Empfehlung veröffentlicht

- XML Schema Definition → „XSD“
- Gegenwärtig weite Akzeptanz in der Industrie
- Unterstützt zusammengesetzte („komplexe“) Datentypen
- Dateiendung „.xsd“

■ Block 3: Schemata

■ XSD Konstrukte

XSD Klausel hat die Form eines XML Elements mit Attributliste

Element: `<xsd:element name="id" type="xsd:String"/>`

Attribut: `<xsd:attribute name="born" type="xsd:integer"/>`

Optionales Element / Attribut:

`<xsd:attribute name="born" type="xsd:integer" minOccurs="0"/>`

Folge von Elementen / Attributen:

`<xsd:element name="siblingBorn" type="xsd:integer" maxOccurs="5"/>`

Block 3: Schemata

Beispiel XSD Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="überweisungsträger">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name_empfänger" type="xs:string" />
        <xs:element name="kto_empfänger" type="xs:integer" />
        <xs:element name="blz_empfänger" type="xs:integer" />
        <xs:element name="bank_empfänger" type="xs:string" />
        <xs:element name="währung">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="EUR" />
              <xs:enumeration value="USD" />
              <xs:enumeration value="ATS" />
              <xs:enumeration value="DKK" />
              <xs:enumeration value="NLG" />
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="betrag" type="xs:float" />
        <xs:element name="verwendungszweck1" type="xs:string"/>
        <xs:element name="verwendungszweck2" type="xs:string" minOccurs="0" />
        <xs:element name="kto_sender" type="xs:integer" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Block 3: Schemata

Beispiel Instanz zum XSD

```
<?xml version="1.0" encoding="utf-8"?>
<überweisungsträger xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="überweisungsträger_schema.xsd">
  <name_empfänger>
    Hans-Peter Siegloch
  </name_empfänger>
  <kto_empfänger>
    12345678
  </kto_empfänger>
  <blz_empfänger>
    87654321
  </blz_empfänger>
  <bank_empfänger>
    Sparkasse Garmelshausen
  </bank_empfänger>
  <währung>EUR</währung>
  <betrag>1.23</betrag>
  <verwendungszweck1>
    Kursgebühr
  </verwendungszweck1>
  <kto_sender>
    01901234
  </kto_sender>
</überweisungsträger>
```

Referenz auf die XSD Datei

Fließkommazahl mit Punkt

Block 3: Schemata

■ Beispiel Validierung Instanz mit XSD

Im Internet gibt es viele Webseiten, die XML Instanzen gegen ein Schema validieren und eventuelle Fehler erklären.

Hier gezeigt: <http://xmlvalidator.new-studio.org/>

XML Validator Online BETA

XML Validation

☐ XPath ☒ Schema ☐ DTD


Charset:

Choose XML file to validate:

XML Schema:

XPath Explorer

Validation Result

 Validation successful!

■ Block 3: Schemata

■ Übung: XML Schema vervollständigen

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="personalausweis">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="vorderseite">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="name" type="xs:string"
            />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Instanz „personalausweis.xml“ aus Block 2 ergänzen um:

```
<personalausweis xmlns:xsi="
http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="personalausweis_schema.xsd">
```

Validieren mit der Webseite: <http://xmlvalidator.new-studio.org/>

■ Block 3: Schemata

■ Vorteile von XSD als de-facto-Standard

XSD ist in der Industrie weit verbreitet

- **Es gibt Generatoren, die aus einer XML-Instanz ein XSD generieren**
(natürlich eines ohne Prüfbedingungen,
aber als Rumpf eine sehr gute Ausgangslage)
- **Es gibt Generatoren, die aus einem XSD eine XML-Instanz generieren**
(Natürlich ohne sinnvolle Datenwerte,
aber als Rumpf eine sehr gute Ausgangslage)

■ Block 3: Schemata

■ Schemasprache „Document Type Definition“ (DTD)

Eine Dokumenttypdefinition legt die Struktur eines Dokuments fest, ist aber strenggenommen keine eigene Schemasprache

- Relativ altes Konzept, kommt aus der Mode
- Struktur einer Grammatik: Wurzel wird abgeleitet zu Blättern
- Kennt nur wenige Datentypen

(Wichtig: ID, PCDATA sowie CDATA: schwach spezifizierte Typen)

CDATA: Character Data = (ungeprüfte) Zeichenkette

PCDATA: Parsable CDATA = Inhalt wird vom Parser ausgewertet

- Dateiendung „.dtd“

■ Block 3: Schemata

■ DTD Konstrukte

DTD Klausel hat die Form einer XML Annotation mit zwei Parametern.

Eine Annotation wird nicht mit „/>“, sondern mit „>“ geschlossen

Element: `<!ELEMENT bin_knoten (bin_knoten, bin_knoten) >`

(Terminal-)Element: `<!ELEMENT bin_knoten (#PCDATA) >`

Attribut: `<!ATTLIST knoten_wert type CDATA „default_value“>`

Leeres Element: `<!ELEMENT null_knoten „EMPTY“ >`

Optionales Element: `<!ELEMENT ketten_knoten (ketten_knoten?)>`

Folge von Elementen: `<!ELEMENT gener_knoten (gener_knoten*)>`

Block 3: Schemata

■ Beispiel DTD Schema

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE überweisungsträger[
  <!ELEMENT überweisungsträger (name_empfänger, kto_empfänger, blz_empfänger, bank_empfänger,
währung, betrag, verwendungszweck1, verwendungszweck2?, kto_sender) >
  <!ELEMENT name_empfänger (#PCDATA) >
  <!ELEMENT kto_empfänger (#PCDATA) >
  <!ELEMENT blz_empfänger (#PCDATA) >
  <!ELEMENT bank_empfänger (#PCDATA) >
  <!ELEMENT währung (#PCDATA) >
  <!ELEMENT betrag (#PCDATA) >
  <!ELEMENT verwendungszweck1 (#PCDATA) >
  <!ELEMENT verwendungszweck2 (#PCDATA) >
  <!ELEMENT kto_sender (#PCDATA) >
]>
```


Block 3: Schemata

Beispiel Instanz zum DTD

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE person SYSTEM „überweisungsträger.dtd">
  <name_empfänger>
    Hans-Peter Siegloch
  </name_empfänger>
  <kto_empfänger>
    12345678
  </kto_empfänger>
  <blz_empfänger>
    87654321
  </blz_empfänger>
  <bank_empfänger>
    Sparkasse Garmelshausen
  </bank_empfänger>
  <währung>EUR</währung>
  <betrag>1.23</betrag>
  <verwendungszweck1>
    Kursgebühr
  </verwendungszweck1>
  <kto_sender>
    01901234
  </kto_sender>
</überweisungsträger>
```

Referenz auf die DTD Datei

Unformatierte Zeichenkette

■ Block 3: Schemata

■ Übung: DTD vervollständigen

```
<?xml version="1.0" encoding="utf-8"?>  
<!DOCTYPE ausweis [  
                                [...]
```

Inhalt der DTD in die Instanz einfügen

Instanz „personalausweis.xml“ aus Block 2 ergänzen um:

```
<!DOCTYPE ausweis [  
<überweisungsträger>    [...]
```

Validieren mit der Webseite: <http://xmlvalidator.new-studio.org/>

■ Block 3: Schemata

■ Schemasprache „Relax NG“ (RNG)

- Die Sprache Relax NG wurde 2003 ein ISO Standard. Die Syntax ist relativ simpel und orientiert sich an XSD
 - Regular Language for XML Next Generation → „RNG“
 - Findet Verbreitung, da durch populäre Auszeichnungssprachen angewendet: DocBook, TEI, OpenDocument
 - Zwei Syntaxvarianten: Einmal XML standardkonform, einmal kompakt und nicht-XML standardkonform
 - Dateiendung: xml-regulär „.rng“, kompakt „.rnc“

■ Block 3: Schemata

■ RNG Konstrukte

RNG Klausel hat die Form eines XML Elements mit Attributliste

Element: `<element name="id" /><data type="NCName"/> </element>`

Attribut: `<attribute name="born" /><data type="integer"/> </attribute>`

Optionales Element / Attribut:

`<zeroOrMore>`

`<element name="id" type="String"/>`

`</zeroOrMore>`

Folge von Elementen / Attributen:

`<oneOrMore>`

`<element name="id" type="String"/>`

`</oneOrMore>`

Block 3: Schemata

Beispiel RNG Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0" datatypeLibrary="
http://www.w3.org/2001/XMLSchema-datatypes">
  <start>
    <element name="überweisungsträger">
      <element name="name_empfänger">
        <data type="NCName"/>
      </element>
      <element name="kto_empfänger">
        <data type="integer"/>
      </element>
      <element name="blz_empfänger">
        <data type="integer"/>
      </element>
      <element name="bank_empfänger">
        <data type="NCName"/>
      </element>
      <element name="währung">
        <choice>
          <value>EUR</value>
          <value>USD</value>
          <value>ATS</value>
          <value>DKK</value>
          <value>NLG</value>
        </choice>
      </element>
```

```
      <element name="betrag">
        <data type="decimal"/>
      </element>
      <element name="verwendungszweck1">
        <data type="NCName"/>
      </element>
      <zeroOrMore>
        <element name="verwendungszweck2">
          <data type="NCName"/>
        </element>
      </zeroOrMore>
      <element name="kto_sender">
        <data type="integer"/>
      </element>
    </element>
  </start>
</grammar>
```

Block 3: Schemata

Beispiel Instanz zum RNG

```
<?xml version="1.0" encoding="utf-8"?>
<?oxygen RNGSchema="überweisungsträger_schema.rng" type="xml"?>
<überweisungsträger>
  <name_empfänger>
    Hans-Peter Siegloch
  </name_empfänger>
  <kto_empfänger>
    12345678
  </kto_empfänger>
  <blz_empfänger>
    87654321
  </blz_empfänger>
  <bank_empfänger>
    Sparkasse Garmelshausen
  </bank_empfänger>
  <währung>EUR</währung>
  <betrag>1.23</betrag>
  <verwendungszweck1>
    Kursgebühr
  </verwendungszweck1>
  <kto_sender>
    01901234
  </kto_sender>
</überweisungsträger>

<!-- validated by: oxygen -->
```

Referenz auf die RNG Datei

Fließkommazahl mit Punkt

■ Block 3: Schemata

■ Übung: RNG vervollständigen (ohne Validierung)

```
<?xml version="1.0" encoding="utf-8"?>
<grammar elementFormDefault="qualified"
  xmlns="http://relaxng.org/">
  <start>
    <element name="personalausweis">
      [...]
```

■ Block 3: Schemata

■ Konzept Namensräume

Namensräume dienen der Unterscheidung zwischen Elementen und Attributen von verschiedener Herkunft aber gleichen Namen

- **Beispiel: Kundenkartei exklusives Bekleidungsgeschäft**
Basisdaten werden vom Personalausweis erhoben („Grösse“)
Schuhgrösse wird von Payback-Karte ausgelesen („Grösse“)
→ Namenskonflikt: Kombination der Daten zweier Quellen kann zu Konflikten bei der Bezeichnung führen: Teekesselchen
- **Lösung: Je ein Namensraum für jede Datenquelle**
- **Schemasprachen gehen unterschiedlich mit Namensräumen um**

Block 3: Schemata

Problem Namenskonflikte



```
<?xml version="1.0" encoding="UTF-8"?>
<kundenkartei>
  <stammdaten>
    <personalausweis>
      <!-- vom personalausweis -->
      <vorname>Hans-Peter</vorname>
      <name>Siegloch</name>
      <groesse> 190 </groesse>
    </personalausweis>
    <payback>
      <!-- von payback -->
      <werberesistenz> 0.12 </werberesistenz>
      <schuhgeschmack> mittel </schuhgeschmack>
      <groesse> 48 </groesse>
    </payback>
  </stammdaten>
  <transaktionen/>
</kundenkartei>
```

**Verschiedene Quellen,
verschiedene Maßeinheiten**

Block 3: Schemata

Lösung Namensräume

Ein Namensraum definiert einen Kontext, innerhalb dessen identische Namen dieselbe Bindung haben

- Siehe Familiennamen zur Unterscheidung im Klassenraum:
„Kevin“ → ~3 Jungs fühlen sich angesprochen
„Kevin Klein“ → Namensraum Klein, Name Kevin ist eindeutiger
- Analog `xmlns="http://www.klein.de/"` :
Darin definierte Namen sind eindeutig innerhalb dieses Bereichs

■ Block 3: Schemata

■ Beispiel Namensräume



```
<?xml version="1.0" encoding="UTF-8"?>
<kundenkartei>
  <stammdaten>
    <personalausweis xmlns="http://www.personalausweis.de/">
      <!-- vom personalausweis -->
      <vorname>Hans-Peter</vorname>
      <name>Siegloch</name>
      <groesse> 190 </groesse>
    </personalausweis>
```



```
<payback xmlns="http://www.paybackski.ru/victims/data/">
  <!-- von payback -->
  <werberesistenz> 0.12 </werberesistenz>
  <schuhgeschmack> mittel </schuhgeschmack>
  <groesse> 48 </groesse>
</payback>
```

```
</stammdaten>
<transaktionen/>
</kundenkartei>
```



UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft



Ende des Blocks 3