XML

Historie: EDI

- EDI – Electronic Data Interchange
- Informationstechnischer Austausch von strukturierten Geschäftsdaten
- Alternative Begriffsverständnisse:
  - generelles Ziel / Konzept
  - spezielle Architekturen / Protokolle / Standards
- Beispiel: EDIFACT: EDI for Administration, Commerce and Transport
  - branchenübergreifendes Rahmenwerk, spezifische Subsets

Historie: EDI

Struktur von UN/EDIFACT

UNA: Service String Advice
UNB: Interchange Header
- UNG: Functional Group Header
  - UNH: Message Header
    - AAA:
    - ⋮
    - ZZZ:
  - UNT: Message Trailer
  - ⋮
  - UNH: Message Header
  - ⋮
  - UNT: Message Trailer
- UNE: Functional Group Trailer
- ⋮
- UNG: Functional Group Header
- ⋮
- UNE: Functional Group Trailer

UNZ: Interchange Trailer

Beispiel für eine EDIFACT-Rechnung (INVOIC)

```
UNA:+.?'
UNB+UNOA:1+126123+126321+930430:1409+REF800'
UNH+INV001+INVOIC:1++1'
BGM+380+1300+930315'
NAD+BY+126321:91++Hacker AG+MODEMSTR 2+
ZUERICH+8000'
NAD+SE+126123:92++PC SCHWEIZ AG+RAMSTR 40+
ZUERICH+8025'
UNS+D'
LIN++5050+10:21:PC+1500:CA:1+10+15000'
LIN++4750+5:21:PC+500:CA:1+5+2500'
UNS+S'
TMA+17500'
UNT+8+INV001'
UNZ+1+REF800'
```

Beispiel für eine konventionelle Rechnung

```
PC Schweiz AG
Ramstr. 40
8025 Zürich                          Hacker AG
                                     Modemstr. 2
                                     8000 Zürich

RECHNUNG Nr. 1300                     30. April 1993

Nr.    Bezeichnung   Menge   Preis     Betrag

5050   PC 80486        10    1500.--  15000.--
4750   Monitor 14       5     500.--   2500.--

Total                                  17500.--
                                       ========
```

- **XML**: Extensible Markup Language
  - Auszeichnungssprache („descriptive markup“)
  - Metasprache zur Definition von Auszeichnungssprachen
- Universeller Basismechanismus für die strukturierte Repräsentation und den Austausch von Informationen (→ Integration)
- Alternative Ursprünge / Sichtweisen / Anwendungsfelder
  - Austausch von „Daten“ zwischen Maschinen
  - Austausch von „Dokumenten“ zwischen Personen

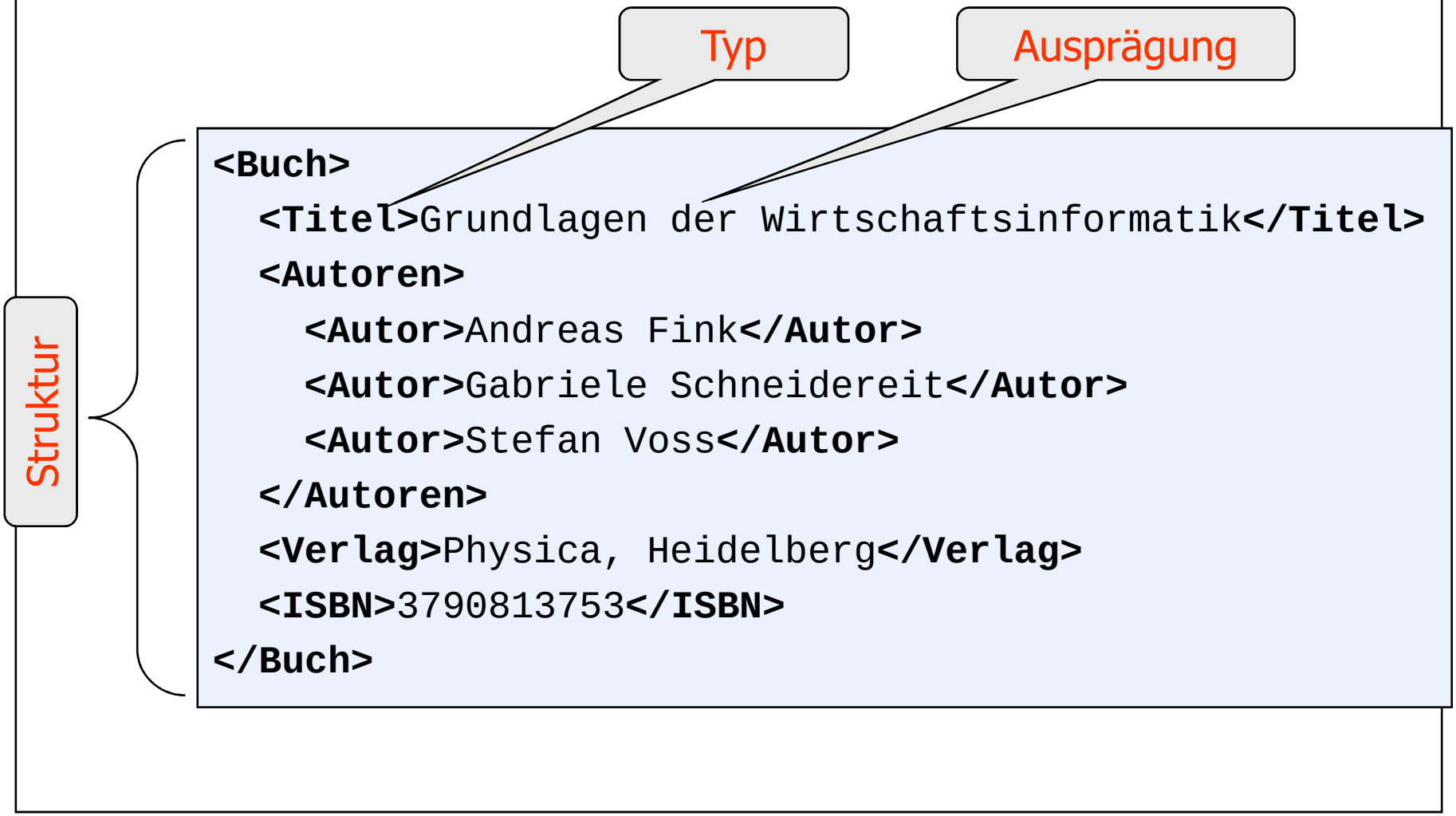
- Deklarative Auszeichnungssprache mit Fokus auf Struktur
- Sprachfamilie:
  - prinzipielle Dokumentsyntax: **XML**
  - Meta-Ebene:
    - **DTD** (Document Type Definition)
    - XML-Schema (**XSD**)
  - Layout/Präsentation: **XSL** (Extensible Style Sheet Language)
  - Transformation: **XSLT** (XSL Transformations), **XPath**
  - **XLink**, **XPointer** (Verknüpfungen)
  - **DOM**, **SAX**, **XQuery** ... (Zugriffsmodelle)
  - *... und einiges mehr ...*

- XSLT: Extensible Style Sheet Language Transformation
- XSLT dient dazu ein XML-Dokument in ein anderes Format zu wandeln
- XSLT wird beispielsweise eingesetzt, um die Daten eines XML-Dokuments in
  - HTML
  - WAP
  - PDF
  - oder ähnliches

  zu transformieren.

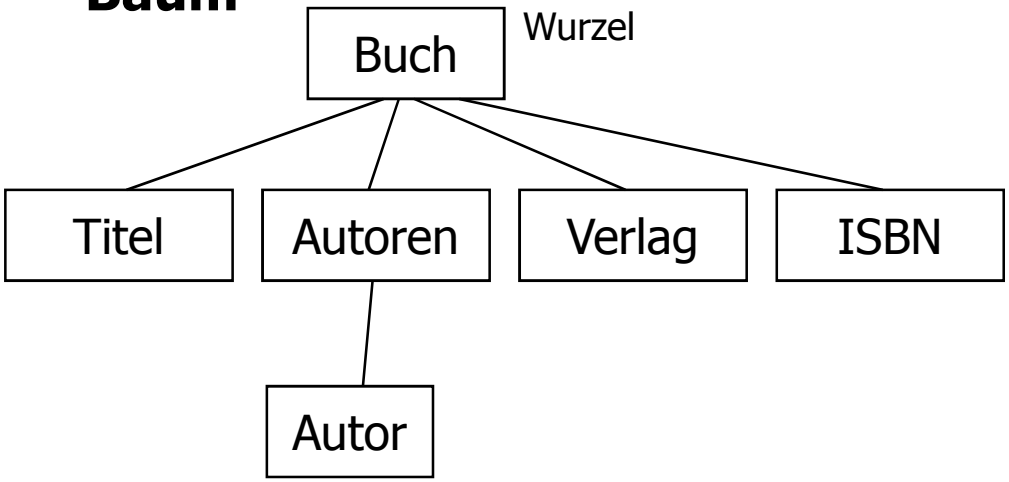
Hierarchische Dokumentstruktur, geschachtelte Tags

- Beispiel für ein XML-Dokument:

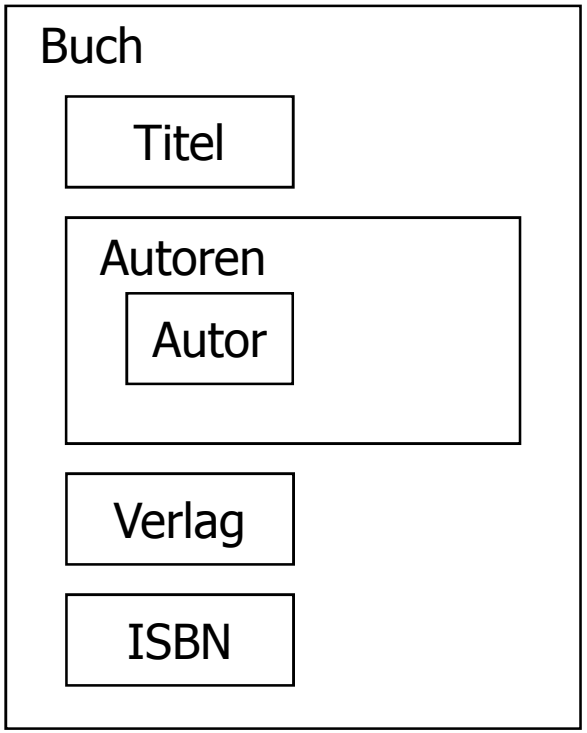


Alternative Veranschaulichungsformen

Baum



Blockstruktur



Textuell per Klammerung

(Buch (Titel) (Autoren (Autor)) (Verlag) (ISBN))

Sprachdefinition

- Standardisierung durch W3C (World Wide Web Consortium)
- Syntax (Grammatik):
  - hierarchische Schachtelung von Beginn- und End-Tags
  - je XML-Dokument genau ein Wurzel-Element (Root)
  - Groß-/Kleinschreibung relevant
  - Attributwerte müssen mit Anführungsstrichen eingeschlossen sein (zunächst universelle Interpretation als Zeichenketten)

- Elemente
  - **`<tag> ... </tag>`** ⇨ streng hierarchische Schachtelung !
  - **`<tag />`** (Abkürzungsform bei leerem Tag)
- Attribute
  - **`<tag attribute-name="name"> ... </tag>`**
  - Attribute sind je Tag eindeutig.
- Entitäten (Referenzen/Textmakros, textuelle Substitutionen)
  - z.B. **`<`** für <
- Kommentare
  - **`<!-- ... -->`**
- Prolog
  - z.B. **`<?xml="1.0" encoding="UTF-8"?>`**
- Verarbeitungsinstruktionen (Processing Instructions)
  - **`<?xml-stylesheet type="text/xsl“ href="mystyle.xsl" ?>`**
- CDATA: nicht durch XML-Parser interpretierte Zeichenketten
  - **`<![CDATA[...]]>`**

Beispiel

```
<?xml version="1.0" ?>
<Buch ISBN="3790813753">
  <Titel>Grundlagen der Wirtschaftsinformatik</Titel>
  <!-- 3. Auflage in Vorbereitung -->
  <Autoren>
    <Autor>Andreas Fink</Autor>
    <Autor>Gabriele Schneidereit</Autor>
    <Autor>Stefan Voss</Autor>
  </Autoren>
  <Verlag>Physica, Heidelberg</Verlag>
  <Cover source="http://www.physica.de/3790813753.png"/>
</Buch>
```

- Einhaltung der XML-Syntax

  ⇨ **Wohlgeformtes** XML-Dokument („well formed“)

- Wohlgeformt + konformes Exemplar eines Dokumenttyps

  ⇨ **Gültiges** XML-Dokument („valid“)

- Metasprachliche Festlegung von Dokumenttypen

  - **DTD**: Document Type Definition
  - **XSD**: XML-Schema Definition

DTDs (Document Type Definitions)

- Festlegung eines Dokumenttyps (Grammatik) über die Definition zulässiger Elemente, Attribute, Entitäten, ...
- Elementdeklaration:
  - `<!ELEMENT element-name inhalt>`
  - Terminalinhalt: #PCDATA
  - Wiederholungsfaktoren für Elemente:
    - `?` : optional
    - `+` : mindestens einmal
    - `*` : kein oder mehrfaches Auftreten
  - Elementauswahl: `( a | b )`
  - Elementsequenz: `( a, b )`

DTDs (Document Type Definitions)

- Attributdeklaration:
  - <!ATTLIST element-name
    attribute-name1 attribute-type1 default-value1
    attribute-name2 attribute-type1 default-value1 ... >
  - z.B. <!ATTLIST Lieferbedingung
    Zeit #PCDATA
    Kosten #PCDATA #REQUIRED>
- Definition / Verwendung von DTDs
  - intern in XML-Dokumenten
  - extern (separate Datei) und Referenz per `<!DOCTYPE ...>`
- Fokus auf Dokumentstrukturen

DTD, Beispiel

```
<!ELEMENT Kundenliste (Kunde*) >
<!ELEMENT Kunde (Name, Adresse?, Telefon*)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Adresse (#PCDATA)>
<!ELEMENT Telefon (#PCDATA)>
<!ATTLIST Telefon
   Typ (mobil | privat | geschäftlich) #REQUIRED>
```

Beispiel, XML nach DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Kundenliste SYSTEM 'kundenliste.dtd'>
<Kundenliste>
  <Kunde>
    <Name>Hans Muster</Name>
    <Telefon Typ="mobil">0171 987</Telefon>
    <Telefon Typ="privat">040 555</Telefon>
    <Telefon Typ="geschäftlich">040428</Telefon>
  </Kunde>
  <Kunde>
    <Name>Heidi Muster</Name>
    <Adresse>Weg 12, 20123 Hamburg</Adresse>
    <Telefon Typ="mobil">0171 123</Telefon>
  </Kunde>
</Kundenliste>
```

DTDs (Document Type Definitions)

- schwache Ausdrucksmöglichkeiten bezüglich Datentypen
- strukturelle Defizite (z.B. keine zweckmäßigen Namensräume)
- keine XML-Sprache
  - keine XML-Werkzeuge nutzbar
  - es existiert keine DTD für eine DTD

  → keine automatische Validierung

XML Schema Definition

- XSD: XML Schema Definition
- Im XSD-Dokument wird der Aufbau eines XML-Dokuments definiert.
- Es beschreibt ähnlich wie ein Datenbankschema die Struktur der XML-Dokumente, die die Daten enthalten
- Die XSD ist die erste Meta-Ebene. Sie umfasst die Definition von Datenstrukturen und die Typisierung
- Aufbau:
  - Definition von Elemente
  - Definition von Attributen
  - Definition der Elemente, die innerhalb anderer Elementen vorkommen und deren Beziehungen
  - Definition, ob ein Element Text enthält oder nicht
  - Definition der Datentypen für Elemente und Attribute
  - Definition von Standard- oder festgelegten Werten

Definition von Elementen

- Mit Hilfe des `<element>`-Tags aus dem Namensraum xsd="http://www.w3.org/2001/XMLSchema" werden Elemente für ein XML-Dokument definiert:

  ```
  <xsd:element name="..." type="..." />
  ```

- Jedes Element erhält einen Namen (z.B. Kundenname) und einen Typ (z.B. `string`, `long`, `double` etc.)
- Mit Hilfe von weiteren Attributen des `<element>`-Tags können zusätzliche Einstellungen gemacht werden:
  - `minOccurs`, `maxOccurs`: Häufigkeit des Auftretens (falls Element optional ist, gilt `minOccurs =0`)
  - `fixed`: Festlegung eines konstanten Werts
  - `default`: Festlegung eines Default -Werts

Definition von Attributen

- Attribute werden für bestimmte Elemente definiert

```
<xsd:attribute name="..." type="..." />
```

- Die Attribute `name` und `type` legen den Namen und den Datentypen fest.
- Daneben können auch Default-Werte (`default`) oder kostnate Werte (`fixed`) für das Attribut bestimmt werden.
- Ein Attribut kann höchstens einmal je Element vorkommen. Dabei kann mit `use` angegeben werden, ob ein Attribut optional (`optional`), erforderlich (`required`) oder verboten (`prohibited`) ist.

Complex Type

- Complex Type-Elemente können weitere Elemente und Attribute beinhalten.
- Wenn innerhalb eines Elements mehrere Einträge erforderlich sind, muss das entsprechend definiert werden:
  - Definition des Namen des Elements des Complex Type <xsd:complexType name="...">
  - Definition einer Folge von Elementen für den Complex Type <xsd:sequence>
  - Definition der einzelnen Elementeund deren Reihenfolge <xsd:element name="..." type="..."/>

Complex Type, Beispiel

```
<xsd:complexType name="Kundenadresse" >
 <xsd:sequence>
  <xsd:element name=“Name" type="xsd:string"/>
  <xsd:element name=“Strasse"type="xsd:string"/>
  <xsd:element name=“PLZ" type="xsd:decimal"/>
  <xsd:element name=“Ort" type="xsd:string"/>
 </xsd:sequence>
 <xsd:attribute name=“Land" type="xsd:string"
  fixed=“Deutschland"/>
</xsd:complexType>
```

XML Schema-Technologie

- XML Schema-Technologie ist leistungsfähiger als DTDs.
- Existenz von Datentypen
- XML Schema basiert auf XML und ist somit erweiterbar
- Definition von Namensräume, in denen verschiedene Datentypen zusammengefasst werden.

  Beispiel:

  Im Namensraum http://www.w3.org/2001/XMLSchema werden einfache Datentypen wie `string`, `long` oder `double` definiert

XML-Schema: Beispiel

```
<?xml version="1.0" encoding="Cp1252"?>
<xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema“
    targetNamespace="http://iwi.edu“
    xmlns:tns="http://iwi.edu“
    elementFormDefault="qualified">

<xsd:element name="Kundenliste“
             type="tns:myKundenliste"/>
  <xsd:complexType name="myKundenliste">
    <xsd:sequence>
      <xsd:element name="Kunde" type="tns:myKunde“
               minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
```

XML-Schema: Beispiel

```
<xsd:complexType name="myKunde">
  <xsd:sequence>
    <xsd:element name="Name" type="xsd:string"/>
    <xsd:element name="Adresse" type="xsd:string“
                 minOccurs="0" />
    <xsd:element name="Telefon“
                 minOccurs="0" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:simpleContent>
          <xsd:extension base="xsd:string">
           <xsd:attribute name="Typ“
                          type="tns:myTelefontypen"/>
          </xsd:extension>
        </xsd:simpleContent>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
```

XML-Schema: Beispiel

```
  <xsd:simpleType name="myTelefontypen">
   <xsd:restriction base="xsd:string">
     <xsd:enumeration value="mobil"/>
     <xsd:enumeration value="privat"/>
     <xsd:enumeration value="geschäftlich"/>
   </xsd:restriction>
 </xsd:simpleType>
</xsd:schema>
```

- An eine Maschine sollen die zu produzierenden Aufträge eines Tages übermittelt werden. Die Auftragsliste besteht aus Aufträgen, denen einzelne Produkte zugeordnet sind, die jeweils in einer bestimmten Menge produziert werden sollen. Die Mengeneinheit ist entweder l (Liter), kg oder t (Tonne). Für die Produktion werden Rohstoffe in bestimmten Mengen benötigt. Außerdem sind Einstellungen an den Maschinen vorzunehmen. Eine Einstellung besteht aus einer Bezeichnung und einem Zahlenwert.
  a) Beschreiben Sie für diese Aufgabe eine möglichst zweckmäßige allgemeine XML-Dokumentstruktur in der Form einer graphischen Baumdarstellung.
  b) Geben Sie ein gültiges XML-Dokument an.
  c) Skizzieren Sie eine entsprechende DTD und XSD
  d) Definieren Sie ein relationales Datemodell, welches den Sachverhalt beschreibt.