Semistrukturierte Daten

Sommersemester 2012

Teil 3: Document Type Definitions (DTDs)

3.1. Dokumenttyp-Deklaration
3.2. Element-Deklaration
3.3. Attribut-Deklaration
3.4. Entitäten
3.5. weitere Bestandteile einer DTD

3.1. Dokumenttyp-Deklaration

- XML-Schemasprachen
- wohlgeformt vs. gültig
- interne/externe DTD-Teilmenge
- Validierung

XML Schemasprachen

- DTD
  - Teil der XML-Recommendation
  - keine XML Syntax
  - eingeschränkte Möglichkeiten ("XML als Dokument")
  - Tipp: **http://xml.coverpages.org/xmlApplications.html**
- XML Schema
  - W3C Recommendation
  - XML Syntax
  - erweiterte Möglichkeiten ("XML als Daten")
- Weitere Schemasprachen, z.B.:
  - RelaxNG (ISO): "Kompromiss" zwischen DTDs und XML Schema.
  - Schematron (ISO): http://www.schematron.com
  - DCD (Document Content Description): Microsoft/IBM

Wohlgeformtes vs. gültiges XML

- Parser sind nicht verpflichtet, ein XML-Dokument gegen ein Schema zu überprüfen.
  - "nicht-validierender" Parser: überprüft nur die Einhaltung der XML Syntaxregeln

    => Überprüft die Wohlgeformtheit
  - "validierender" Parser: überprüft ein XML-Dokument gegen konkrete DTD, XML-Schemadefinition, etc.

    => Überprüft die Gültigkeit
- Bemerkung: Das kann zu unterschiedlichen XML-Dokumenten führen (z.B.: Default-Werte, Entitäten)
- Verhalten im Fehlerfall:
  - Syntax-Fehler: fatal
  - "Validity Error": Parser kann trotzdem weitermachen

Dokument Typ Deklaration

- Einbettung in XML mittels DOCTYPE-Deklaration:
  `<!DOCTYPE ...>`
- direkt nach der XML Deklaration
- interner Teil vs. externer Teil
  - intern: zwischen eckigen Klammern in der Deklaration
  - extern: separat, wird im XML-Dokument referenziert
- Mischung extern/intern möglich:
  - interne Definition hat Vorrang gegenüber einer externen
  - Elemente: Mehrfachdefinitionen verboten
- "Standalone Dokument" bei ausschließlich internem Teil:
  - optional: `<?xml version="1.0" standalone="yes"?>`

interne/externe DTD-Teilmenge

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- DBAI -->
<!DOCTYPE lehre SYSTEM "lehre.dtd"
   [
   <!ATTLIST veranstaltung jahr CDATA #REQUIRED>
   ]
>

<lehre>
  <veranstaltung jahr = "2011">
    <titel>Semistrukturierte Daten</titel>
  ......
</lehre>
```

Externer Teil: System vs. Public

- System Identifier
  - für alle Nichtstandarddokumentarten: lokale Datei oder URL, z.B.

```
<!DOCTYPE seminar SYSTEM "http://www.seminar.at/se.dtd">
```

- Public Identifier
  - für bekannte Dokumentarten; ist eher ungebräuchlich
  - es muss dem XML Prozessor bekannt sein, was damit zu tun ist
  - kann zweiten Wert aufweisen, der SYSTEM entspricht (falls der erste Wert nicht aufgelöst werden kann), z.B.:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "xhtml11.dtd">
```

Syntax: Erstes Zeichen: + wenn ISO, sonst –
Zweiter Teil: DTD Eigentümer
Dritter Teil: DTD Beschreibung
Vierter Teil: Sprache

Validierung

XMLLINT: (http://xmlsoft.org/)

- portable C Bibliothek für Linux, Unix, MacOS, Windows, …
- Kommandozeilen-Aufruf:

```
xmllint --valid <xml-dateiname>
```

DOM/SAX-Parser in Java:

- Optionen: validierend oder nicht-validierend

dbai

3.2. Element-Deklaration

- Inhaltsmodelle
- Syntax

Inhaltsmodelle

- Mögliche Inhaltsmodelle bei DTDs
  - Text-Inhalt: keine Typisierungen, nur Strings
  - Element-Inhalt: nur Sub-Elemente
  - Gemischter Inhalt: Text + Sub-Elemente
  - leerer Inhalt
  - beliebiger Inhalt

Elementdeklaration: Syntax

- Schlüsselwörter
  - #PCDATA: Text-Inhalt (parsed character data)
  - EMPTY: leeres Element
  - ANY: beliebiger Inhalt (die vorkommenden Elemente müssen aber definiert sein in DTD)
- Auftretensindikatoren
  - reguläre Ausdrücke +,*,?
  - ohne: genau einmal
  - +: mindestens einmal und beliebig oft
  - *: 0-mal oder öfter
  - ?: 0-mal oder einmal
- Gruppierung
  - mit Klammern (...)

Beispiele

```
<!ELEMENT lehre ANY>
Instanz:          <lehre>irgendetwas<sonst>und noch
                  etwas</sonst></lehre>

<!ELEMENT lehre EMPTY>
Instanz:          <lehre/>

<!ELEMENT veranstaltung (name, jahr)>
Instanz:          <veranstaltung>
                     <name>Seminar</name><jahr>2011</jahr>
                  </veranstaltung>
keine Instanz:    <veranstaltung>
                     <jahr>2011</jahr><name>Seminar</name>
                  </veranstaltung>

<!ELEMENT lehre (#PCDATA)>
Instanz:          <lehre>Seminar</lehre>
keine Instanz:    <lehre> <jahr>2011</jahr> … </lehre>
```

Elementdeklaration: Syntax

- Konnektoren
  - Sequenz: ","
  - Auswahl: "|"
- "," angeführte Elemente in genau dieser Reihenfolge!
- "|" nur eines darf vorkommen (exklusives oder)
- "|" iteriert mit * oder + erlaubt daher beliebige Reihenfolge (und öfteres Auftreten)
- verschachtelte Klammerung, z.B.
  ```
  (lname, (fname | title))
  ```
- auch Rekursion möglich, z.B.
  ```
  <!ELEMENT ARTIKEL (NUMMER,BESTEHT)>)
  <!ELEMENT BESTEHT (ARTIKEL)*>
  ```

Elementdeklaration: Gemischter Inhalt

- Gemischter Inhalt vs. Element-Inhalt
  - Element-Inhalt: nur Elemente enthalten
    Beispiel: **`<!ELEMENT vorbesprechung (datum, zeit, ort)>`**
  - Gemischter Inhalt: mit | trennen in DTD und * am Ende
    z.B. nicht erlaubt: **`<!ELEMENT name (#PCDATA , fname , lname)*>`**
    erlaubt: **`<!ELEMENT name (#PCDATA | fname | lname)*>`**
  - #PCDATA immer an erster Stelle spezifizieren
  - #PCDATA darf nicht gemeinsam mit "," stehen
    z.B. nicht erlaubt: **`<!ELEMENT lehre (#PCDATA, veranstaltung)>`**
- Beispiel:

```
<inhalt>Das ist <i>ein <b>gemischter</b></i> Inhalt</inhalt>
```

Elementdeklaration: Weitere Beispiele

`<!ELEMENT lehre (veranstaltung+)>`

Lehre ist eine Liste von Veranstaltungen (mindestens eine in diesem Beispiel).

`<!ELEMENT buchtitel (deutsch | englisch* | italienisch)>`

Ein Buchtitel besteht entweder aus einer deutschen oder aus 0 bis mehreren englischen oder aus einer italienischen Bezeichnung.

`<!ELEMENT name (#PCDATA | fname | lname)*>`

Ein Name besteht aus Vorname, Nachname, oder einer Mischung daraus mit gewöhnlichem Text.

`<!ELEMENT veranstaltung ((name, jahr?)+)>`

Liste von Namen/Jahren, wobei Jahrangabe optional ist.

3.3. Attribut-Deklaration

- Syntax
- Attribut-Typen
- Vorgabedeklarationen
- Namespaces

Attributdeklaration

- Bestandteile der Attribut-Deklaration:
  Name, Typ, Vorgabedeklaration
- Definition über Attributlisten
- entweder alle Attribute in einer Deklaration oder verstreut über mehrere Deklarationen
- Angabe von zugehörigem Element
- Die Reihenfolge ist egal

Attributdeklaration: Beispiele

```
<!ELEMENT zeit (#PCDATA)>
<!ATTLIST zeit sine-tempore (yes|no) "no">
```

Zeit enthält ein Attribut. Das Attribut sine-tempore kann "yes" oder "no" annehmen, wobei "no" der Defaultwert ist.

```
<!ATTLIST test href CDATA #REQUIRED>
```

Das href Attribut muß in test immer angegeben werden, und kann einen beliebigen Stringwert annehmen.

```
<!ATTLIST test xml:lang NMTOKEN #IMPLIED>
```

Wenn dieses spezielle Attribut xml:lang im Dokument vorkommt, dann muss es "normal" in der DTD definiert werden, z.B.: hier als optionales Attribut vom Typ NMTOKEN

Attributdeklaration: Syntax

```
<!ELEMENT zeit (#PCDATA)>
<!ATTLIST zeit sine-tempore (yes|no) "no">
```

- Elementname: "zeit"
- Attributname: "sine-tempore"
- Attributtyp (im Prinzip nur Stringtypen), z.B.: Aufzählungstyp
- Vorgabedeklaration: Defaultwert "no"

Bemerkung: Attributwerte können stärker beschränkt werden als Elementwerte.

Attributtypen

- CDATA (Character Data, String):
  - Zeichenkette in " " oder ' '
- Token-Typen:
  - Werte können auf verschiedene Arten eingeschränkt werden (s.u.)
- Aufzählungstyp:
  - Mögliche Werte aufzählen: müssen XML-Namenstoken sein (d.h. kein Whitespace, keine Sonderzeichen außer . : _ -)
  - Syntax: Liste `(mo|di|mi|do|fr|sa|so)` ohne " "
  - bzw. mit Schlüsselwort NOTATION, falls die Elemente der Liste in der DTD definierte Notationen sind, z.B.:

```
NOTATION (GIF|JPEG|PNG)
```

Token-Typen

- ID:
  - Attribut muss in jedem Element einen (im gesamten XML-Dokument) eindeutigen Wert haben.
  - Dieser Wert muss ein erlaubter XML Elementname sein.
- IDREF bzw. IDREFS:
  - Referenz bzw. Liste von Referenzen auf IDs
  - Listentrennung durch Zwischenräume
- ENTITY bzw. ENTITIES:
  - externe, nicht geparste Entität (s.u.), die in der DTD deklariert sein muss, bzw. Liste von solchen Entitäten
- NMTOKEN bzw. NMTOKENS:
  - einzelner Name (im Prinzip wie XML Elementname, darf aber auch mit Zahl beginnen) bzw. Liste von Namen

Beispiel: ID, IDREF

- ID/IDREF-Attribute in DTD deklarieren:

```
<!ELEMENT PRODUKT (#PCDATA)>
<!ATTLIST PRODUKT WarenCode ID #REQUIRED
    GehoertZu IDREF #IMPLIED>
```

- ID/IDREF-Attribute verwenden:

```
<PRODUKT WarenCode="S034">Kinderfahrrad</PRODUKT>
<PRODUKT WarenCode="S039" GehoertZu="S034">
    Gepäckträger
</PRODUKT>
```

Beispiel 4

Vorgabedeklarationen

#REQUIRED: Dieses Attribut muss im Dokument vorkommen.

#IMPLIED: Dieses Attribut kann im Dokument vorkommen.

"*wert*": Dieser Wert wird als Defaultwert angenommen, wenn kein Wert angegeben ist.

#FIXED "*wert*": Attribut muss diesen Wert haben, falls angegeben.

Wie kann man fixen Wert fordern? Workaround mittels Aufzählungstyp!

***(wert)* #REQUIRED**

Namespaces in DTDs

- in DTDs nur implizit: wie normale Attribute
- Namespace in Element- und Attributdeklaration angeben
- Beispiel:

```
<!ELEMENT stud:bewertung (#PCDATA)>
<!ATTLIST lehre xmlns:stud CDATA #REQUIRED>
```

- Namespace als fixes Attribut angebbar, um es nicht in jeder Dokumentinstanz wiederholen zu müssen

```
<!ATTLIST lehre xmlns CDATA #FIXED
 "http://tuwis.tuwien.ac.at/lehre/2.0">
```

Beispiel 5

3.4. Entitäten

- Überblick
- die verschiedenen Entitäten im Einzelnen

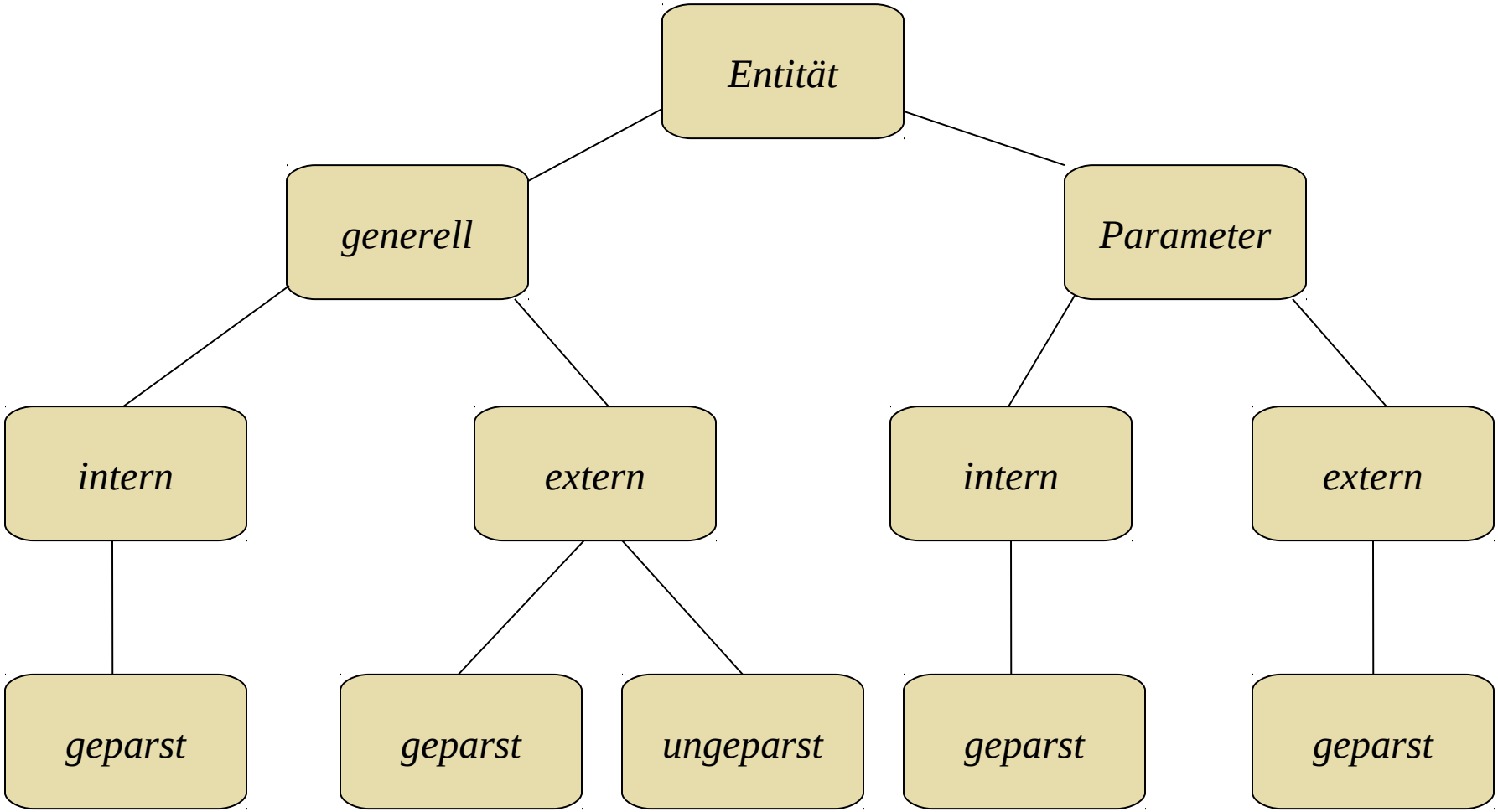
Entitäten in DTDs

- Entitäten:
  - Zeichenkette, die in der DTD als interne Entität definiert ist
  - externe Datei, die in der DTD als externe Entität definiert ist
  - Vordefinierte Entitäten
  - Character Entities
- Zweck:
  - Text-Makros: insbes. interne Entitäten
  - Modularisierung: insbes. externe Entitäten
- DTD Auswertung:
  - zunächst Entitätenexpansion
  - das veränderte Dokument und die DTD werden dann auf Wohlgeformtheit bzw. Gültigkeit geprüft

Arten von Entitäten in DTDs

- Generelle vs. Parameter Entität
  - Generell: innerhalb der Dokument-Instanz verwenden
  - Parameter Entität: innerhalb der DTD selbst verwenden
    in interner DTD: darf nicht in Elementen, Attributen, ... verwendet werden sondern **nur auf oberster Ebene**
- Interne vs. Externe Entität
  - Intern: Zeichenkette in Anführungszeichen
  - Extern: in einer separaten Datei
- Geparste vs. Ungeparste Entität
  - Geparst: XML-Text (Zeichendaten, Markup, ...). XML-Parser interpretiert den Inhalt
  - Ungeparst: beliebiger Datentyp (i.a. XML-fremde Daten)

Erlaubte Entitätstypen in DTDs



Generelle, interne, geparste Entität

- In DTD deklarieren

```
<!DOCTYPE ARTIKEL
[<!ELEMENT ARTIKEL (TITELSEITE, EINLEITUNG, ABSCHNITT*)>
 <!ELEMENT TITELSEITE (#PCDATA|UNTERTITEL)*>
    ...
 <!ENTITY titel "Die Geschichte von XML
    <UNTERTITEL>Die Sprache des Web</UNTERTITEL>">
```

- Im Dokument referenzieren

```
<ARTIKEL>
   <TITELSEITE> Titel: &titel; </TITELSEITE>
   ...
</ARTIKEL>
```

Generelle, externe, geparste Entität

- In DTD deklarieren:

```
<!DOCTYPE ARTIKEL ...
 <!ENTITY titel SYSTEM "Titel.xml">
```

- Inhalt der Datei Titel.xml

```
Die Geschichte von XML
<UNTERTITEL>Die Sprache des Web</UNTERTITEL>
```

- Im Dokument referenzieren:

```
<ARTIKEL>
   <TITELSEITE>Titel: &titel; </TITELSEITE>
   ...
</ARTIKEL>
```

Generelle, externe, ungeparste Entität

- In DTD deklarieren:

```
<!DOCTYPE BUCH
[<!ELEMENT BUCH (TITEL, AUTOR, COVERBILD)>
...
<!ATTLIST COVERBILD Quelle ENTITY #REQUIRED>
<!NOTATION GIF SYSTEM "bild/gif">
<!ENTITY christo SYSTEM "Christo.gif" NDATA GIF>
```

- Im Dokument referenzieren:

```
<BUCH>
   <TITEL>Der Graf von Monte Christo</TITEL>
   <AUTOR>Alexandre Dumas</AUTOR>
   <COVERBILD Quelle= "christo" />
</BUCH>
```

interne, geparste Parameterentität

- In DTD deklarieren und referenzieren:

```
<!ELEMENT lehre (veranstaltung+)>
<!ELEMENT veranstaltung (titel, schlagwort*, ...)>
...
<!ENTITY % boolean "(yes|no) 'no'">
<!ATTLIST zeit sine_tempore %boolean;>
```

- Einschränkung bei der Verwendung:
  - Verwendung der Parameterentität in Element-, Attribut-, ... Definition: nur in externer DTD oder in externer, geparster Parameterentität erlaubt!

externe, geparste Parameterentität

- In DTD deklarieren und referenzieren:

```
<!DOCTYPE  BESTAND
[<!ELEMENT BESTAND (BUCH | CD)*>
 <!ENTITY % buch_def SYSTEM "Buch.dtd">
 <!ENTITY % cd_def   SYSTEM "CD.dtd">
 %buch_def;
 %cd_def;
]
```

- Inhalt der Datei Buch.dtd:

```
<!ELEMENT BUCH (TITEL, AUTOR, COVERBILD)>
<!ELEMENT TITEL (#PCDATA | UNTERTITEL)*>
...
```

Weitere Entitäten

- Vordefinierte Entitäten
  - **<**
  - **>**
  - **&**
  - **"**
  - **'**
- Charakter-Entitäten (Unicode Wert)
  - **Ó** (dezimal)
  - **ó** (hex)

3.5. weitere Bestandteile einer DTD

- Notation
- Konditionale Abschnitte
- Zusammenfassung

Notationen

- Notation beschreibt ein bestimmtes Datenformat
  - URI eines Programms für Bearbeitung dieses Formats
  - URI eines Online-Dokuments, das dieses Format beschreibt
  - Einfache Beschreibung des Formats
- Verwendung (eher ungebräuchlich)
  - Format einer generellen, externen, ungeparsten Entität beschreiben
  - um Attribut vom Aufzählungstyp NOTATION zu definieren
- Beispiele von Deklarationen in der DTD

```
<!NOTATION DOC SYSTEM "WinWord.exe">
<!NOTATION BMP SYSTEM
"http://www.dbai.tuwien.at/hilfe/bmp.html">
 <!NOTATION GIF SYSTEM "Graphic Interchange Format">
```

Konditionale Abschnitte

IGNORE / INCLUDE-Blöcke:

- Nur in externer DTD-Teilmenge bzw. in externer Parameter-Entität erlaubt!
- IGNORE: "Auskommentieren" von Teilen einer DTD

```
<![IGNORE[ <!ELEMENT name (fname, lname)> ]]>
```

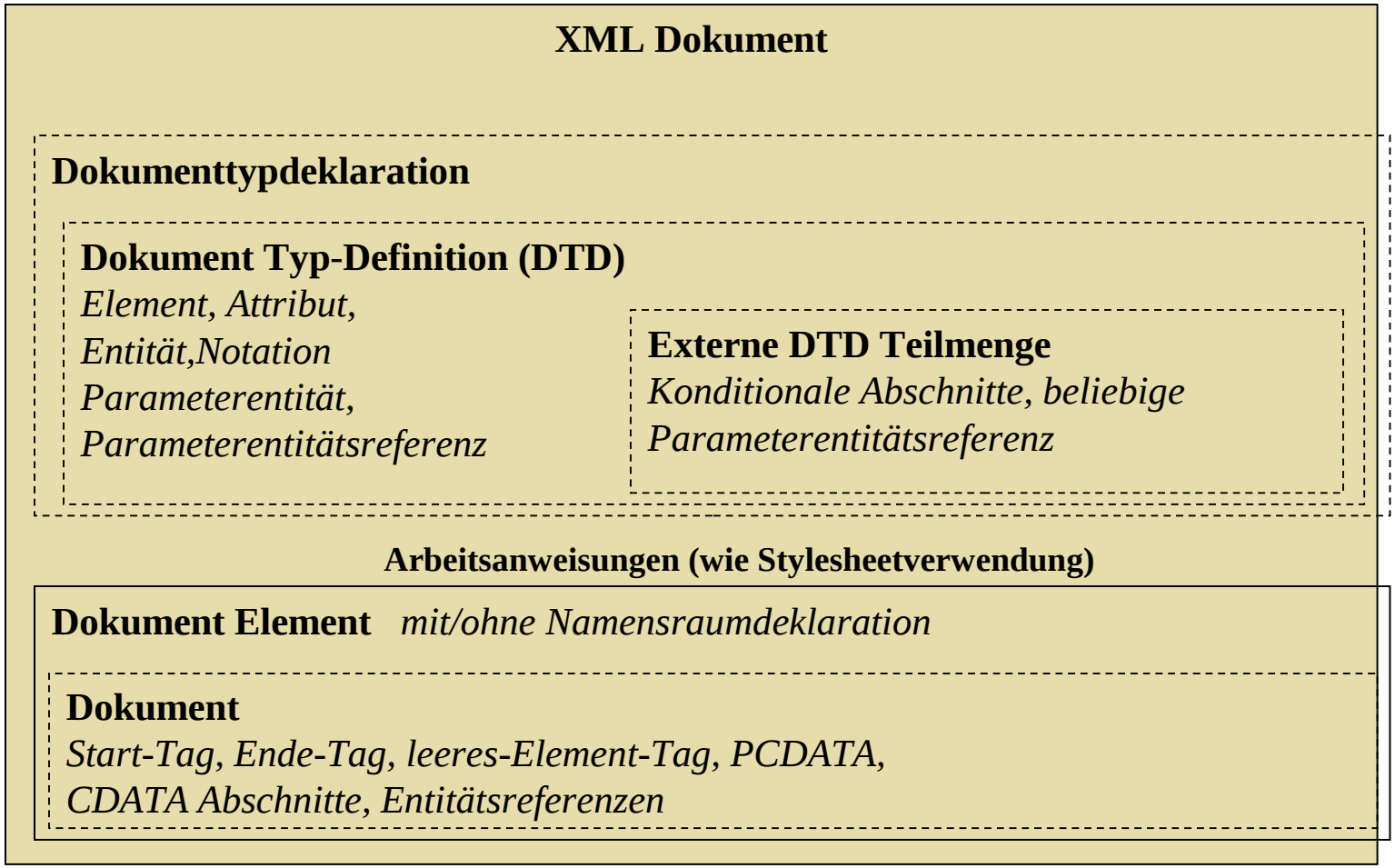
- INCLUDE: "Auskommentieren" beseitigen

```
<![INCLUDE[ <!ELEMENT name (fname, lname)> ]]>
```

- Parameter-Referenz ist manchmal praktisch:

```
<!ENTITY % flexibel 'IGNORE'>
<![%flexibel;[ <!ELEMENT name (fname, lname)> ]]>
```

Zusammenfassung (XML-Dokument + DTD)



Einschränkungen von DTDs (vs. XML-Schema)

- nicht in XML Syntax
- Rudimentäre Typisierung; beschränkte Auswahl an Attributtypen
- Keine Unterstützung von Namespaces
- Codierung beliebiger Kardinalitäten ist aufwendig
- keine objektorientierten Konzepte wie Vererbung; keine Wiederverwendbarkeit
- Rudimentäre Referenzierungen
- Erweiterbarkeit/Skalierbarkeit problematisch