UNIVERSITÄT BERN

E-Business und digitale Prozesse

Übung

Grundkurs XML II:

XML-Schema verstehen – Definition von Elementen und Typen

Prof. Dr. Thomas Myrach
Universität Bern
Institut für Wirtschaftsinformatik
Abteilung Informationsmanagement

Stand: 15. Oktober 2022

Lernziele

- Sie können die Grundstruktur eines XML-Schemas formulieren.
- Sie können XML-Elemente mit ihren jeweiligen Typen in XML-Schema definieren.
- Sie kennen das Typkonzept von XML-Schema.
- Sie können die vordefinierten Simple Types einsetzen.
- Sie können komplexe Typen definieren.
- Sie kennen die Bedeutung von komplexen Typen für den Aufbau von Elementhierarchien.
- Sie kennen die verschiedenen Varianten von XML-Schema zum Aufbau von Elementhierarchien.
- Sie können das mehrfache Auftreten von Elementen über Kardinalitäten festlegen.

Agenda: Übersicht

XML-Schemas und Elemente

Elemente und vordefinierte Simple Types

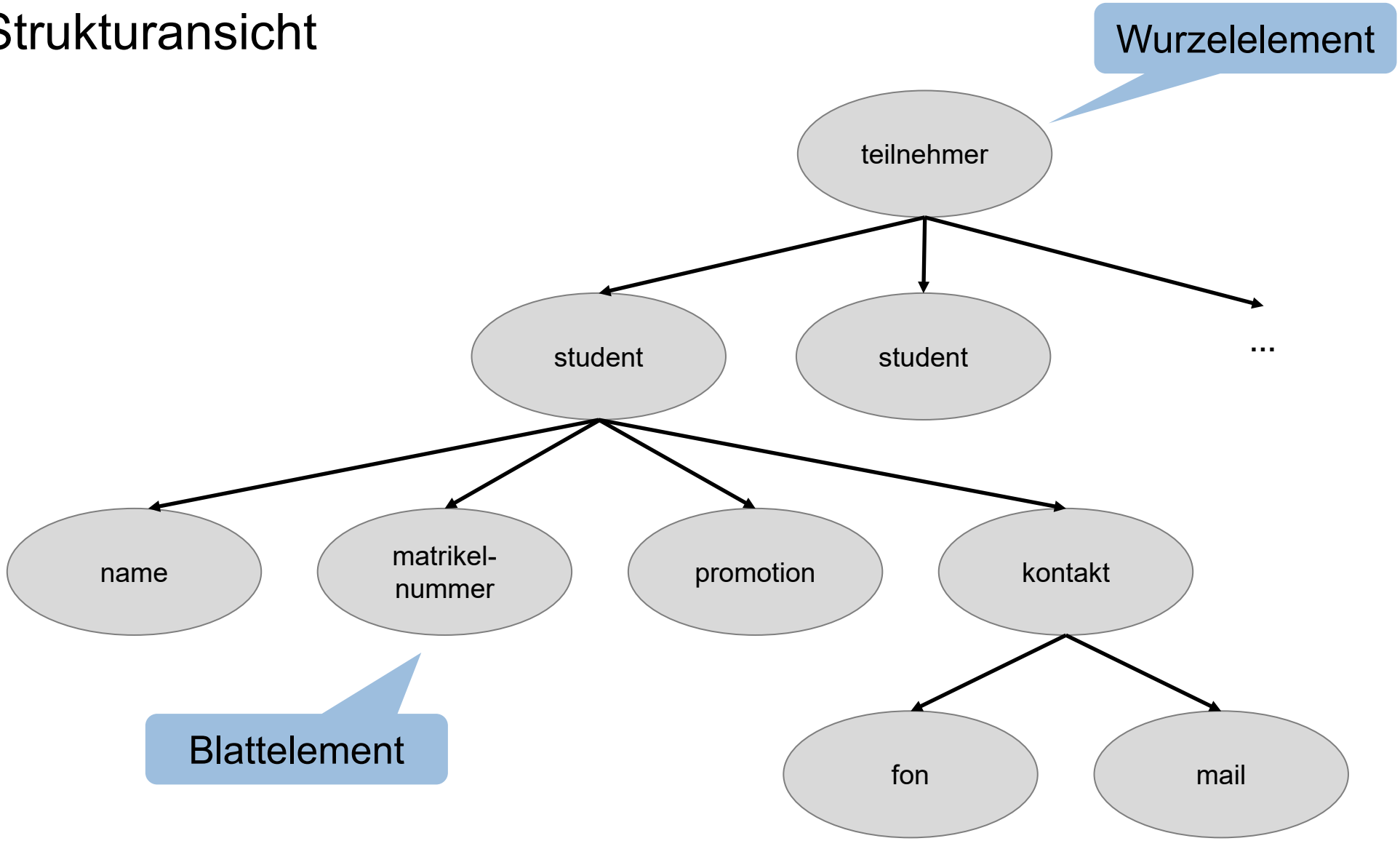
Elemente und Complex Types

XML-Schema

- Mit einem XML-Schema wird festgelegt, welche Struktur eine XML Datei aufweisen muss.
- XML-Schema ist die Grundlage für das Validieren von XML-Dokumenten.
- Seit Mai 2001 hat das XML-Schema den Status einer W3C-Empfehlung (http://www.w3.org/XML/Schema).
- Ein XML-Schema ist selber in XML geschrieben und die Spezifikation erfolgt über XML-Auszeichnungen.
- Für die Definition eines XML-Schemas sind spezifische XML-Auszeichnungen vorgesehen.
- In der Regel wird das XML-Schema in Form einer Datei vom Typ «.xsd» abgelegt.

Beispiel einer XML-Datei

Strukturansicht



Beispiel einer XSD-Datei (1/2)

UNIVERSITÄT BERN

Textansicht

Namespace

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <xsd:element name="teilnehmer">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element="student" maxOccurs="unbounded">
                        <xsd:complexType>
                            <xsd:sequence>
                                <xsd:element name="name" type="xsd:string"/>
                                ...
                            </xsd:sequence>
                        </xsd:complexType>
                    </xsd:element>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
</xsd:schema>
```

Elementdefinition

"anonymer" complex Type

vordefinierter simple Type

XML-Schema

Wurzelement <schema>

- Das Wurzelelement eines XML-Schema-Dokuments heisst <schema>.
- Das XML-Schema wird von anderen Elementen eines XML-Dokuments über einen Namespace abgegrenzt.
- Gemäss Konvention ist das Präfix für den Namespace eines XML-Schemas «xsd» oder «xs».
- Grundgerüst eines XML-Schemas:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
                    …
</xsd:schema>
```

Verweis auf die Definition von XML-Schemas

XML-Namespace mit dem Präfix xsd

XML-Schema (*.xsd)

Exkurs: Namespace

- XML-Namespaces werden dazu eingesetzt, um mehrere XML-Vokabulare zu mischen oder um ein bestimmtes XML-Vokabular besonders zu kennzeichnen.
- Die Deklaration eines Namespace erfolgt über das Attribut "xmlns" zu einem Element.
- Der Namespace gilt für das betreffende Element sowie seine Unterelemente.
- Definiert werden Namensräume über die Referenzierung einer URI (meistens URL).
- Die Referenzierung wird nicht durch einen Parser ausgewertet, sondern dient nur der Namensfestlegung.
- Zusammen mit dem Namespace kann ein Präfix definiert werden, der als Kürzel für den Namespace dient.
- Dieser Präfix wird mit dem Attribut spezifiziert: "xmlns:präfix".

Definition eines Elements

- Im Rahmen eines XML-Schemas können die Elemente definiert werden, die in einem XML-Dokument gebraucht werden sollen.
- Das gewünschte Element wird mit der XML-Elementdeklaration eingeführt, mit dem der Name des Elements festgelegt wird: *<element name="tbd"…/>*
- Beispiel:
  - *XSD:* *<xsd:element name="teilnehmer" … />*
  - *XML:* *<teilnehmer> … </teilnehmer>*
- Der Inhalt von Elementen wird in XML-Schema über das Typkonzept definiert.
- Die wichtigste Besonderheit von XML-Schema ist das Typkonzept:
  - Einfacher Typ: Inhalte von Blattelementen
  - Komplexer Typ: Inhalte von Elternelementen mit Kinderelementen.

Agenda: Übersicht

XML-Schemas

Elemente und vordefinierte Simple Types

Elemente und Complex Types

Simple Types

Vordefinierte Datentypen

- In XML-Schema steht eine Anzahl von vordefinierten simple Types zur Verfügung (built-in datatypes).
- Diese entsprechen den üblichen Datentypen, wie sie in Programmiersprachen oder Datenbanksystemen unterstützt werden.
- Vordefinierte simple Types müssen nicht extra definiert werden.
- Vordefiniere simple Types können einem Element direkt über ihren Namen zugeordnet werden.
- Dies geschieht über das Attribut "*type*".
- Da die vordefinierten simple Types zum XML-Schema gehören, müssen sie mit dem entsprechenden Namespace (*xsd:*) als Präfix definiert werden.

Simple Types

Zeichenketten

- Ein zentraler Typ im XML-Schema ist die Spezifikation von Zeichenketten über den Datentyp „string“.
- In diesem Element für Zeichenketten kann praktisch beliebiger und beliebig langer Inhalt eingefügt werden.
- Dies umfasst auch spezifische Zeichenketten wie Zahlen oder Datumsangaben.

- Beispiel:
  - *XSD:* *<xsd:element name="titel" type="xsd:string"/>*
  - *XML:* *<titel>Vorteile von XML-Schema gegenüber DTD</titel>*

Simple Types

Zahlen

- Im XML-Schema kann festgelegt werden, dass ein Inhalt nur aus Zahlen bestehen darf.
- Zur Spezifikation von Zahlen stehen verschiedene Datentypen zur Verfügung:
  - decimal (dezimale Zahl)
  - integer (ganze Zahl)
  - positiveInteger (positive Ganzzahl)
  - usw.
- Beispiel:
  - *XSD:* `<xsd:element name="preis" type="xsd:decimal"/>`
  - *XML:* `<preis>49.50</preis>`

Simple Types

Datum und Zeit

- Im XML-Schema können Datentypen so hinterlegt werden, dass sich damit Datums- und Zeitangaben im XML-Dokument validieren lassen.
- Für die Spezifikation von Datum und Zeit im XML-Schema kann aus verschiedenen Datentypen gewählt werden:
  - date (Kalenderdatum)
  - time (Zeitangabe)
  - dateTime (Zeitpunkt)
  - usw.
- Beispiel:
  - *XSD:* `<xsd:element name="datum" type="xsd:date"/>`
  - *XML:* `<datum>2012-09-02</datum>`

Agenda: Übersicht

XML-Schemas

Elemente und vordefinierte Simple Types

Elemente und Complex Types

Complex Types

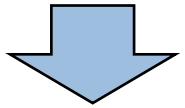
Einsatzzweck

- Ein komplexer Datentyp besteht aus einem Inhaltsmodell, das nicht nur aus einem einzigen Datenwert besteht.
- Das Inhaltsmodell kann mehrere Elemente und auch Attribute umfassen, die wiederum bestimmte Inhalte haben.
- Die im Rahmen komplexer Typen definierten (Unter-)Elemente können einfache oder wiederum komplexe Typen aufweisen.
- Komplexe Typen können dazu benutzt werden, hierarchische Strukturen von Elementen zu modellieren.
- Komplexe Typen sind nicht vorgegeben, sondern müssen explizit definiert werden.
- Für die Definition von komplexen Typen bestehen alternative Formulierungsmöglichkeiten.

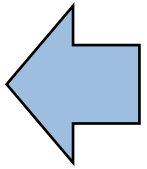
Complex Types

Simple Type versus complex Type

- Definition eines Namens mit einem simple Type:
- *<xsd:element name="name" type="xsd:string"/>*



- *XML simpleType:*
  <name>Max Muster</name>
- *XML complexType:*
  <name>
  <vorname>Max</vorname>
  <nachname>Muster</nachname>
  </name>



- Definition eines Namens mit einem complex Type:
- *<xsd:element name="name" type="name_typ"/>*
- *<xsd:complexType name="name_typ">*
  <xsd:sequence>
  <xsd:element name="vorname" type="xsd:string"/>
  <xsd:element name="nachname" type="xsd:string"/>
  </xsd:sequence>
  </xsd:complexType>

Complex Types

Anordnung von Unterelementen

- Ein komplexer Datentyp kann (Unter-)Elemente enthalten.
- Zwingend anzugeben ist ein eigenes Schema-Element, über das die Anordnung der Unterelemente festgelegt wird:
  - <sequence>: Die gegebene Reihenfolge ist einzuhalten.
  - <choice>: Aus der hinterlegten Liste kann ausgewählt werden.
  - <all>: Alle Elemente können vorkommen, jedoch jeweils nur einmal.
- Beispiel:
  - *<xsd:complexType>*
    <xsd:sequence>
    <xsd:element name="student" type="…"/>
    </xsd:sequence>
    </xsd:complexType>

Complex Types

Anonym oder Eigenständig

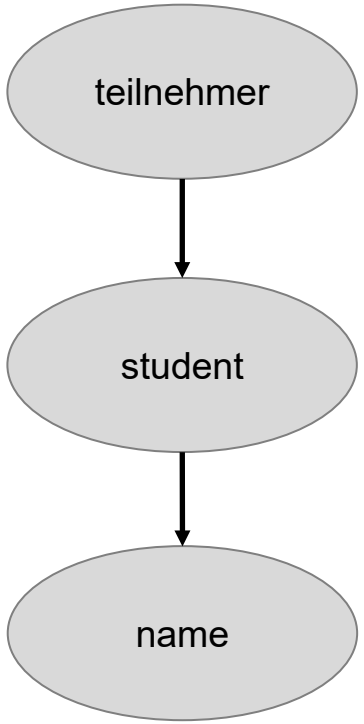
- Komplexe Typen können "anonym" direkt dem betreffenden Element zugeordnet werden:
  - *<xsd:element name="...">*
    <xsd:complexType>...</xsd:complexType>
    </xsd:element>

- Komplexe Typen können als eigenständiger Datentyp mit Namen definiert werden:
  - *<xsd:element name="..." type="eigenerTyp"/>*
  - *<xsd:complexType name="eigenerTyp">*
    ...
    </xsd:complexType>

Elementhierarchien

Variante 1: Schachtelung mit anonymen complex Types

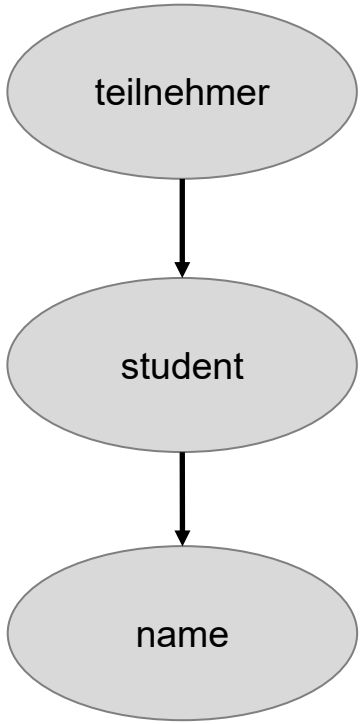
```
<xsd:element name="teilnehmer">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="student">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="name" type="xsd:string"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```



Elementhierarchien

Variante 2: Referenzierungen mit anonymen complex Types

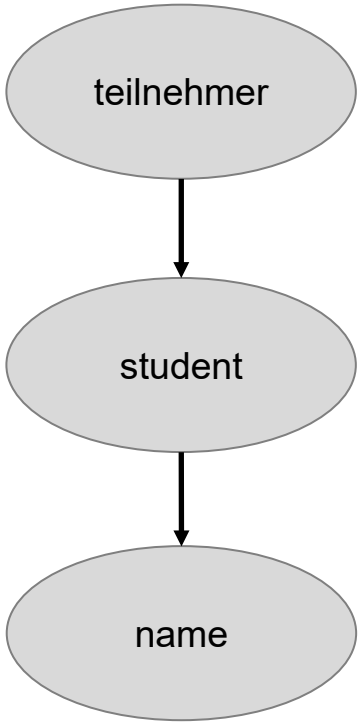
- <xsd:**element name="teilnehmer"** >
  <xsd:complexType>
  <xsd:sequence>
  <xsd:element ref="student"/>
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>
- <xsd:**element name="student"**>
  <xsd:complexType>
  <xsd:sequence>
  <xsd:element ref="name" />
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>
- <xsd:**element name="name"** type="xsd:string"/>



Elementhierarchien

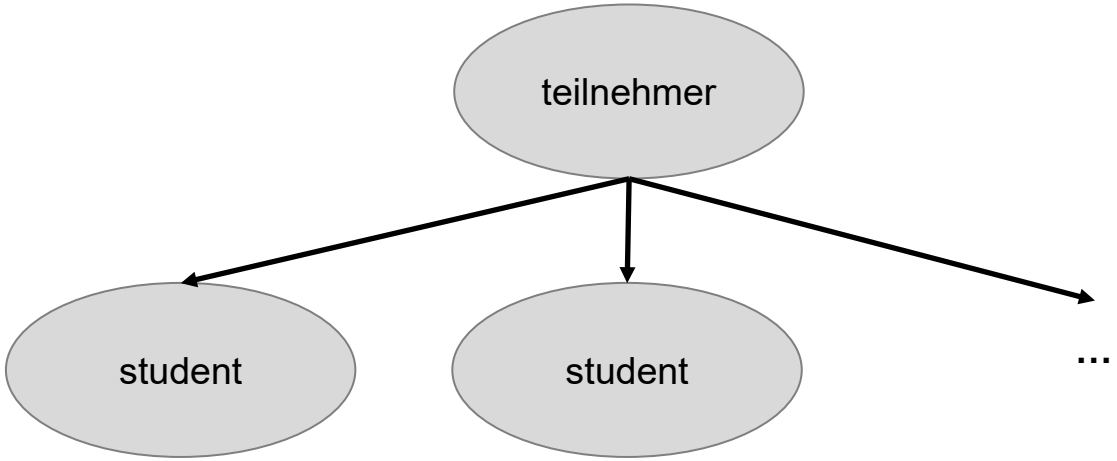
Variante 3: Eigene complex Types

- *<xsd:**element name="teilnehmer"** type="teilnehmer_typ"/>*
- *<xsd:complexType name="teilnehmer_typ">*
  <xsd:sequence>
  *<xsd:**element name="student"** type= "student_typ"/>*
  </xsd:sequence>
  </xsd:complexType>
- *<xsd:complexType name="student_typ">*
  <xsd:sequence>
  *<xsd:**element name="name"** type="xsd:string"/>*
  </xsd:sequence>
  </xsd:complexType>



Elementhierarchien

Kardinalitäten (1/2)



- In einer XML-Hierarchie können einem übergeordneten Element mehrere Instanzen eines untergeordnete Elemente zugeordnet sein.
- In der Elementdeklaration kann die Anzahl möglicher untergeordneter Elemente über Attribute festgelegt werden.
- Diese Werte heissen Kardinalitäten.

Elementhierarchien

Kardinalitäten (2/2)

- Kardinalitäten werden in Elementdeklarationen mit den Attributen minOccurs und maxOccurs ausgedrückt.
- Dabei gilt ohne weitere Angaben: minOccurs="1" und maxOccurs="1"

Kardinalität	XML-Schema	Kommentar
1:1	minOccurs="1" maxOccurs="1"	beide nicht notwendig
0:1	minOccurs="0" maxOccurs="1"	minOccurs reicht
1:n	minOccurs="1" maxOccurs="unbounded"	maxOccurs reicht
0:n	minOccurs="0" maxOccurs="unbounded"	beide notwendig

Elementhierarchien

Beispiel Kardinalitätsangaben

- In der Teilnehmerliste müssen mehrere Studierende aufgeführt werden können.

```
<xsd:element name="teilnehmer">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="student" type="…"
                                    minOccurs="1" maxOccurs="unbounded" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

- Da der Standardwert von minOccurs 1 ist, kann die Angabe entfallen:

```
…
<xsd:element name="student" type="…" maxOccurs="unbounded" />
…
```

Fazit

- XML-Schema ist ein Instrument, um die Datenstruktur von XML-Dateien festzulegen.
- XML-Schema basiert wiederum auf XML, wobei die Elemente von XML-Schema üblicherweise mit dem Präfix eines Namespaces gekennzeichnet werden.
- Mit XML-Schema können die zulässigen Elemente für XML-Dateien festgelegt werden.
- Jedes Element weist einen Datentyp auf.
- SimpleTypes erlauben einfache Inhalte, von denen etliche vorgegeben sind.
- Um Elementhierarchien zu bilden, werden complexType`s gebildet.
- Es existieren verschiedene Varianten, wie mit complexType`s ausgedrückt werden kann, dass ein Element bestimmte Unterelemente haben kann.