XML

Prof. Dr.-Ing. Martin Gaedke

Technische Universität Chemnitz

Fakultät für Informatik

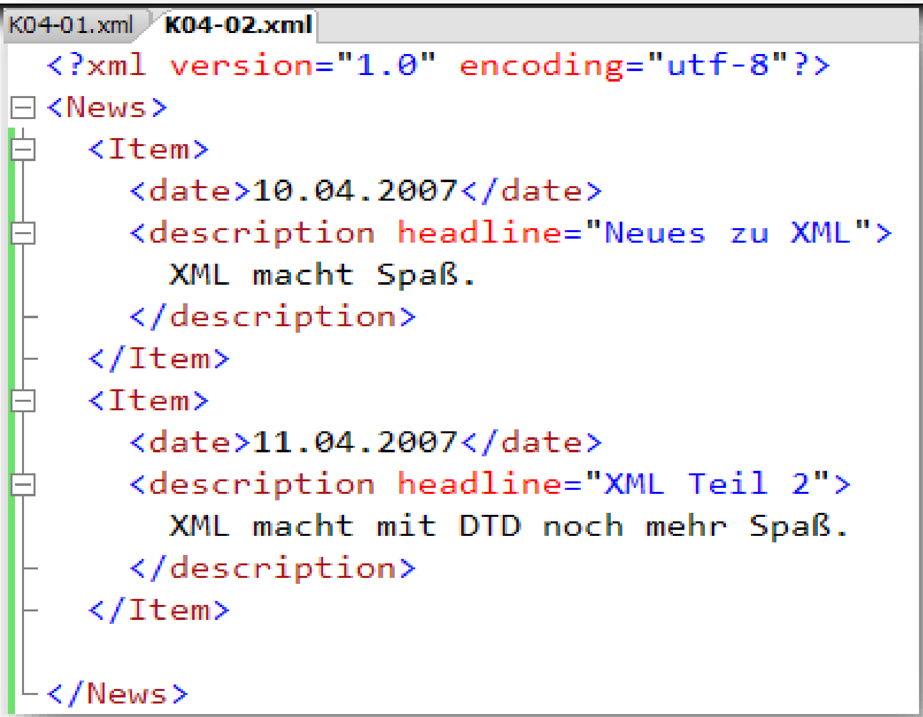
Professur Verteilte und Selbstorganisierende Rechnersysteme

http://vsr.informatik.tu-chemnitz.de

Chapter 4

DESCRIPTION OF XML DOCUMENTS

Messages… Again


K04-01.xml K04-02.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<News>
  <Item>
    <date>10.04.2007</date>
    <description headline="Neues zu XML">
      XML macht Spaß.
    </description>
  </Item>
  <Item>
    <date>11.04.2007</date>
    <description headline="XML Teil 2">
      XML macht mit DTD noch mehr Spaß.
    </description>
  </Item>

</News>
```

- Messages should be distributed to different newspapers in an XML format…
  - How should an XML message be written?
  - How can one check whether a message is valid?
  - *Solution*: Grammar use: DTD.

Document Description

- **W3C Definitions**
  - **"Document Type Declaration** – Contains or points to markup declarations that provides a grammar for a class of documents. This grammar is known as a document type definition, or DTD."
    - Parser directive for DTD use
  - **"Document Type Definition** – Set of markup declarations included in or referenced by an XML document."
    - Grammar describing the structure of XML data

XML Document Structure

XML Document

Prologue: Document declaration <?xml … ?>, Comments

Document-Type-Definition (DTD)
Element, Attribute, Entity
As well as external DTD

Document contents: root element
Elements, Attributes, PCDATA, Comments etc.

On Grammar…

<!DOCTYPE …>

- Specifies a DTD for the document
  - Grammar specification via URL
  - Grammar is specified as a part of the document
- Example: URL to an external DTD

<!DOCTYPE *News* SYSTEM "*http://example.org/news.dtd*">

Document Type Definition

- **Document Type Definition Syntax**
  - Doctypedecl ::=
    '<!DOCTYPE' S Name (S ExternalID)?
      S? ('[' (Markupdecl | DeclSep)* ']' S?)? '>'
  - DeclSep ::= PEReference | S
  - Markupdecl ::= elementdecl | AttlistDecl | EntityDecl |
    NotationDecl | PI | Comment

- 6 types of markup declaration:
  - **Element Type Declaration**
  - **Attribute-List Declaration**
  - Entity Declaration
  - Notation Declaration
  - Processing Instruction
  - Comment

→ *Details for all declarations see XML literature and links*

Element Type Declaration

- Definition of an element and its content model
  - **<!ELEMENT S Name S Content-Specification>**
  - Content-Specification
    - **ANY** – Arbitrary contents
    - **EMPTY** – Empty element
    - *Mixed* – Text and further subelements
      - **'(' S? '#PCDATA' (S? '|' S? Name)* S? ')*‘ | '(' S? '#PCDATA' S? ')'**
    - *Children* – sequence or set of subelements
      - **Children ::= (choice | seq) ('?' | '*' | '+')?**
      - **cp ::= (Name | choice | seq) ('?' | '*' | '+')?**
      - **choice ::= '(' S? cp (S? '|' S? cp)+ S? ')‘**
      - **seq ::= '(' S? cp (S? ',' S? cp)* S? ')'**

Element Type Declaration Example

- <!ELEMENT elem (item*,date,name?)>
  - Sequence of
    [0-n] <item>, <date> und [0-1] <name>
- <!ELEMENT recursion (item | (recursion, thing))>
  - Example:

```
<recursion>
        <recursion>
                <recursion>
                        <item/>
                </recursion>
                <thing/>
        </recursion>
        <thing/>
</recursion>
```

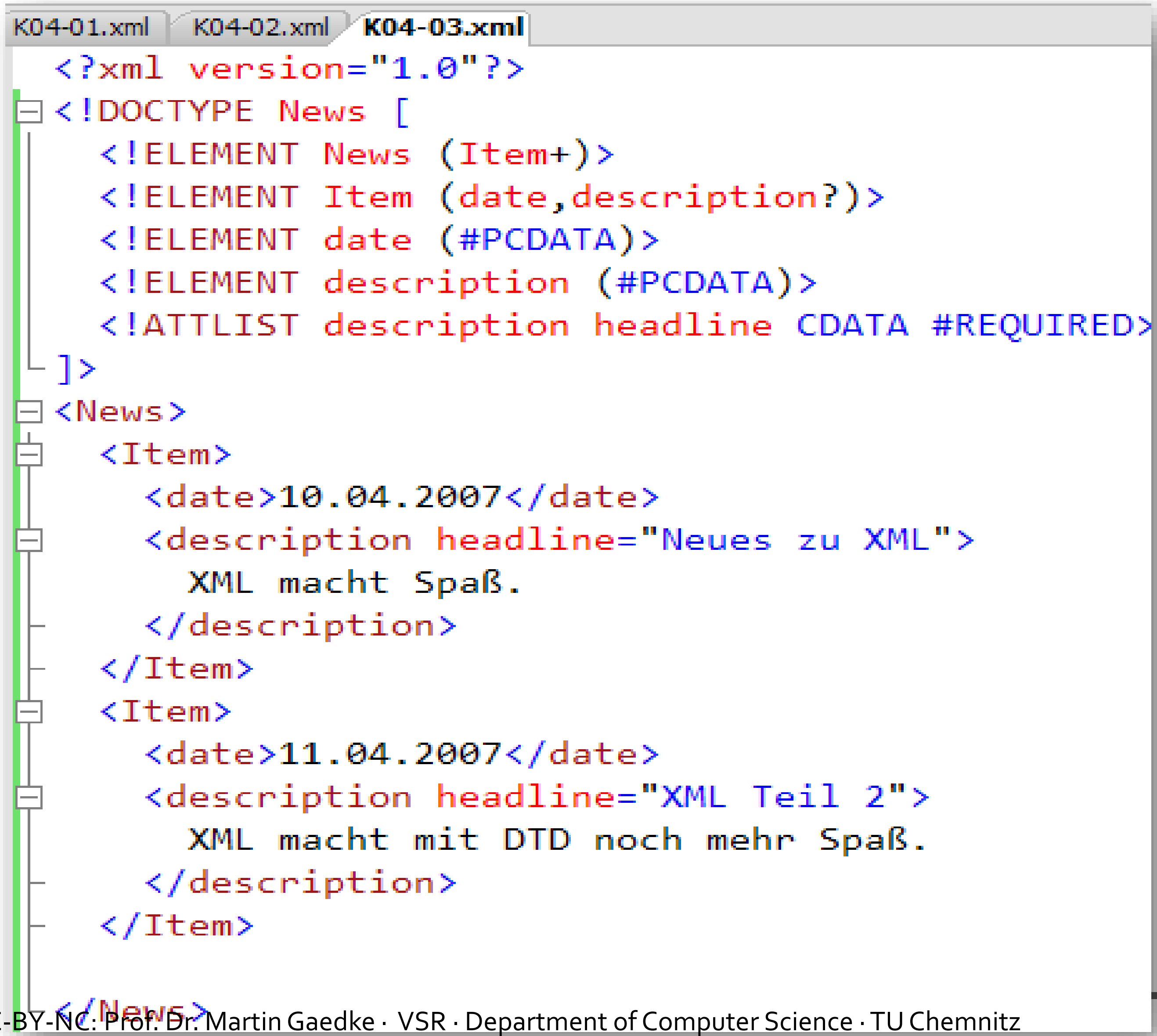
Attribute Declaration

- **Attribute-List Declaration:**
  '<!ATTLIST' S Name AttDef* S? '>'
  - *Name* is an element, which attribute (list!) is bound to
  - *AttDef*: Defines name and type as well as value characteristics
    - Types: for example, CDATA (String), ID, IDREF, IDREFS, ENTITY, ENTITIES, NMTOKEN, NMTOKENS
    - Possible value characteristics: #REQUIRED, #IMPLIED, #FIXED value

Attribute Declaration Example

- <!ATTLIST elemname
      myenumtype (true|false|dontknow) 'true'>
  - <elemname myenumtype=“dontknow”/>
- <!ATTLIST elem1 att2 CDATA #REQUIRED>
  - <elem1 att2=“Value must be set here”/>
- <!ATTLIST elem2 key ID #IMPLIED>
  - Attribute with name key must be unique (in the whole document)
  - <elem2 key=“a”/><elem2 key=“a”/> leads to error

Messages with DTD



K04-01.xml K04-02.xml K04-03.xml

```
<?xml version="1.0"?>
<!DOCTYPE News [
  <!ELEMENT News (Item+)>
  <!ELEMENT Item (date,description?)>
  <!ELEMENT date (#PCDATA)>
  <!ELEMENT description (#PCDATA)>
  <!ATTLIST description headline CDATA #REQUIRED>
]>
<News>
  <Item>
    <date>10.04.2007</date>
    <description headline="Neues zu XML">
      XML macht Spaß.
    </description>
  </Item>
  <Item>
    <date>11.04.2007</date>
    <description headline="XML Teil 2">
      XML macht mit DTD noch mehr Spaß.
    </description>
  </Item>

</News>
```

To the Tool

DTD Advantages / Disadvantages

- Advantages:
  - Simple in writing (and understanding)
  - Compact notation
  - Tool support, such as W3C validator
    - Xerces: Open Source vgl. Apache - http://www.apache.org/xerces-j/
    - Various Microsoft tools, such as XML-Notepad, MSXML Components, Visual Studio
    - XML Spy (GUI)
- Disadvantages:
  - Not an XML notation (double the learning curve)
  - Poor expressivity (small number of data types, no namespaces)
  - Little structuring possibilities
    - Example: Formulate set elements with {A,B}
    - Random order leads to permutations in grammar:
      <!ELEMENT Set ((A, B) | (B, A))>
    - Complexity O(n!)

Chapter 5

DESCRIPTION WITH XML-SCHEMA

XML Schema Definition Language

- **XML Schema Definition Language (XSD)**
  - Since May 2001 - a W3C recommendation
  - Consists of Part 1 and Part 2 as well as an overview Part 0 (*Homework*!!)
  - http://www.w3.org/XML/Schema
  - Sidenotes:
    - W3C recommends "Schemas" as a plural of Schema (Not Schemata!)
    - XDR (XML-Data Reduced) Predecessor of Microsoft now only XSD
- **Motivation:**
  - "While XML 1.0 supplies a mechanism, the Document Type Definition (DTD) for declaring constraints on the use of markup, automated processing of XML documents requires more rigorous and comprehensive facilities in this area. Requirements are for constraints on how the component parts of an application fit together, the document structure, attributes, data-typing, and so on."

XSD Possibilities

- More possibilities come to light:
  - Separation of tags and types
  - Integration of concepts from object-orientation
  - Inheritance, complex structures and reuse
  - Many data types
  - Use of namespaces for use of more grammars
  - Schema definition mechanisms with full typing
  - Documentation options

XML Schema Application of XML

- XML Schema entails all advantages of XML
  - Root element „schema“
  - Element for description of elements are defined in the W3C namespace „XMLSchema“ (Schema of all XML Schemas)

- Schema definition in XML

```
<?xml version="1.0" encoding="utf-8" ?>
<xsd:schema
   targetNamespace="http://example.org/Names"
   elementFormDefault="qualified"
   xmlns=" http://example.org/Names "
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
</xsd:schema>
```

XML Schema and DTD

- DTD can be converted to an XML Schema
  - Can be partially realized in XML tools, such as XMLSpy, Visual Studio
  - Other direction is not possible!
- DTD is short – XML Schema is extensive
  - XSD description of elements is very long
  - Data amount difference between DTD and XSD is enormous
  - XSD compresses very well
  - XSD can be read by a usual XML parser and modified the same simple way as a normal XML document

Example News – DTD vs. XSD

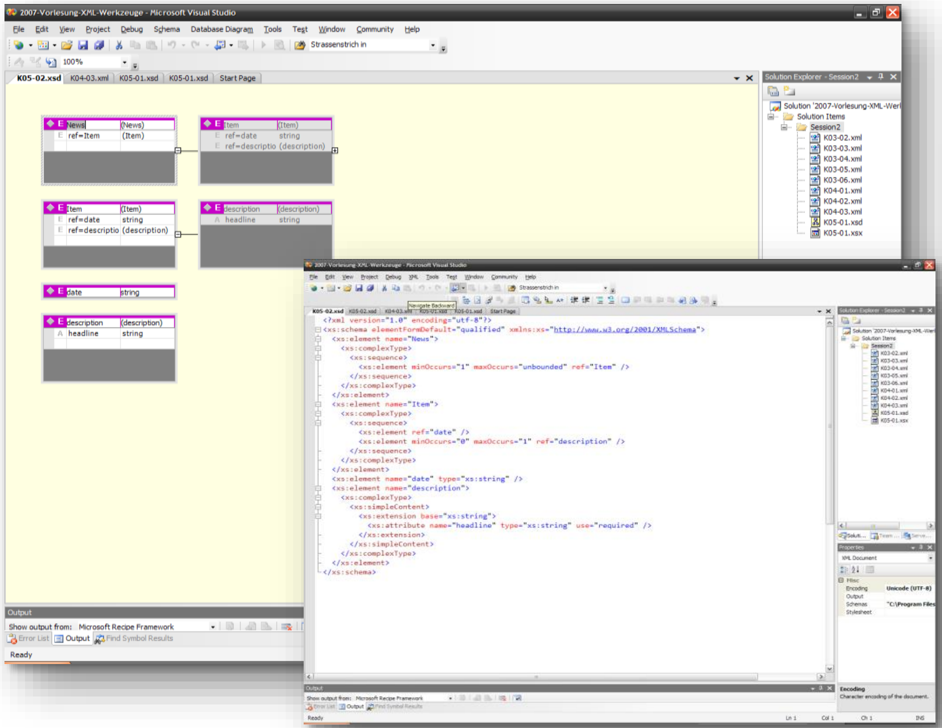
DTD

```
<!DOCTYPE News [
  <!ELEMENT News (Item+)>
  <!ELEMENT Item
     (date,description?)>
  <!ELEMENT date (#PCDATA)>
  <!ELEMENT description
     (#PCDATA)>
  <!ATTLIST description headline
     CDATA #REQUIRED>
]>
```

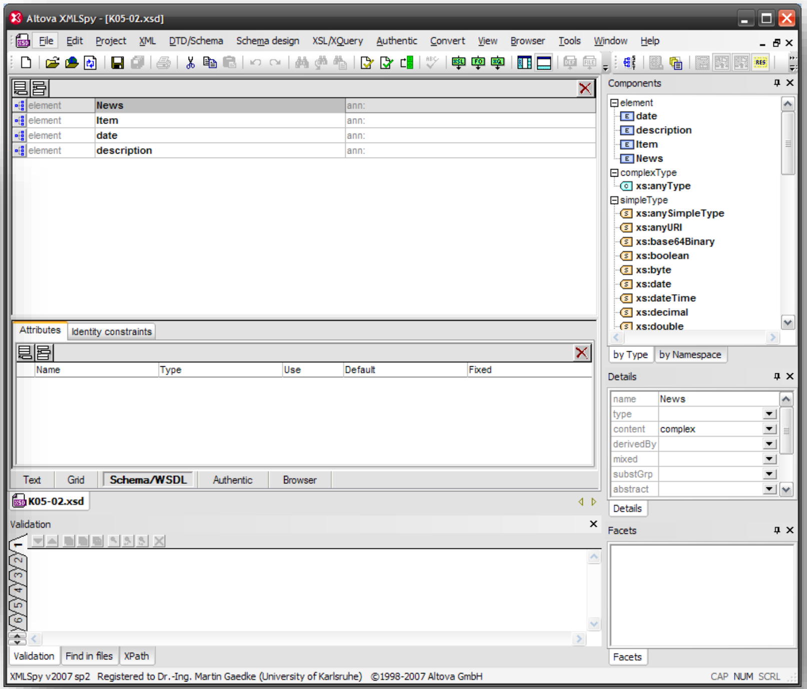
XSD

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified"
        xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="News">
  <xs:complexType>
   <xs:sequence>
    <xs:element minOccurs="1" maxOccurs="unbounded" ref="Item" />
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="Item">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="date" />
    <xs:element minOccurs="0" maxOccurs="1" ref="description" />
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="date" type="xs:string" />
 <xs:element name="description">
  <xs:complexType>
   <xs:simpleContent>
    <xs:extension base="xs:string">
     <xs:attribute name="headline" type="xs:string" use="required" />
    </xs:extension>
   </xs:simpleContent>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Tool Support



```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="News">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="1" maxOccurs="unbounded" ref="Item" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Item">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="date" />
        <xs:element minOccurs="0" maxOccurs="1" ref="description" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="date" type="xs:string" />
  <xs:element name="description">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="headline" type="xs:string" use="required" />
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
</xs:schema>
```



Visual Studio and XMLSpy

XML Target Namespace

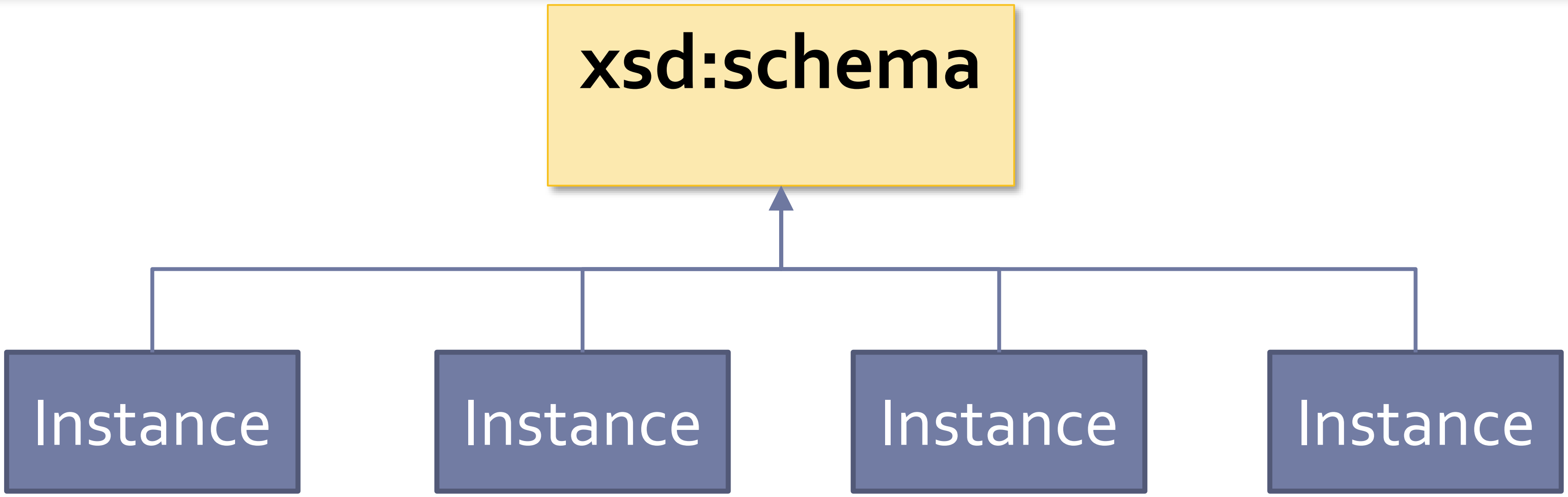
- What is it for?

```
<?xml version="1.0" encoding="utf-8" ?>
<xsd:schema
        targetNamespace="http://example.org/Names"
        elementFormDefault="qualified"
        xmlns="http://example.org/Names"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
</xsd:schema>
```

XML Target Namespace

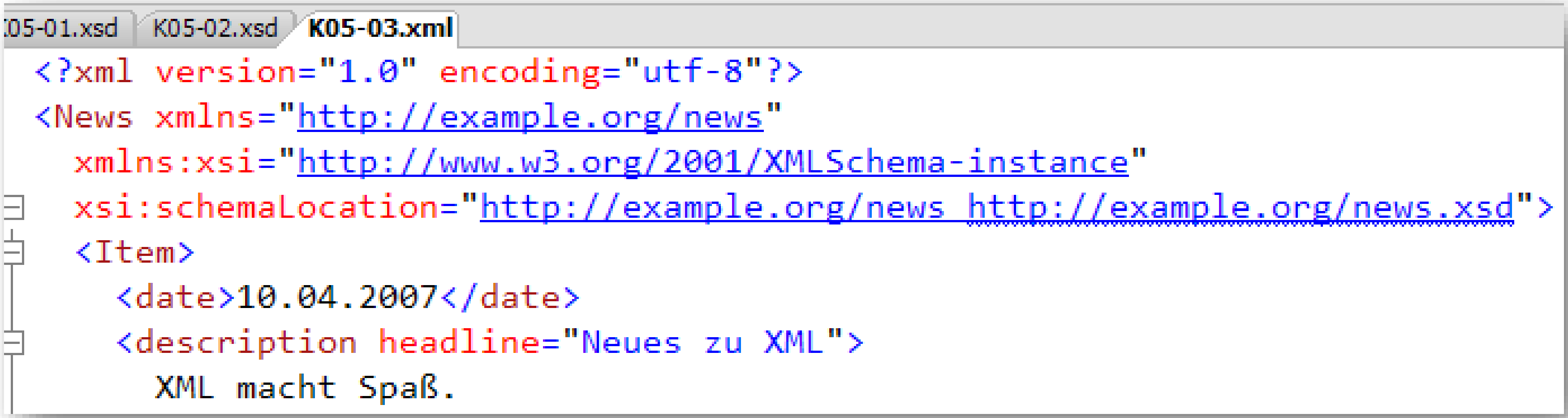
- XML Schema defines elements, attributes, etc. so, vocabulary
  - Problem: How can a yet not defined vocabulary be referenced?
- Vocabulary is assigned a namespace: ***targetNamespace***
- Further difference between DTD and Schema
  - DTD does not define a namespace
  - Schema does

XML-Schema in Action



- **Schema Instance** – Creation of an XML document using a schema
  - XML document, to which the targetNamespace of the XML schemas is assigned
  - **Instance** follows the schema rules

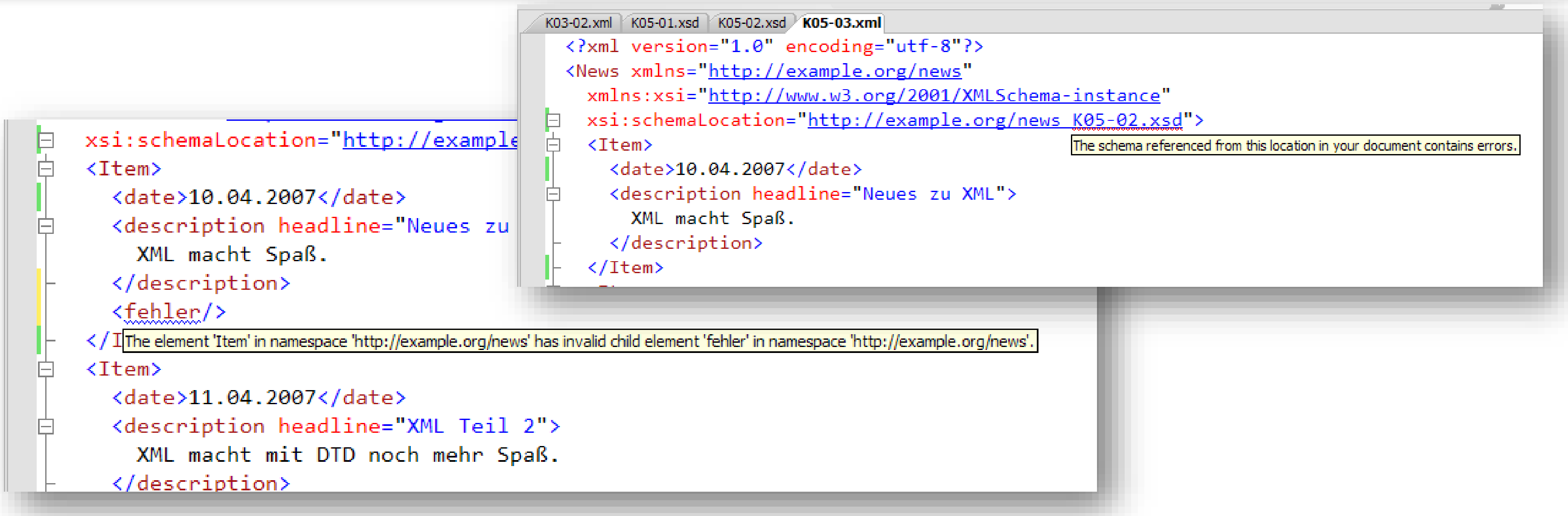
XML Instance Example



```
<?xml version="1.0" encoding="utf-8"?>
<News xmlns="http://example.org/news"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://example.org/news http://example.org/news.xsd">
  <Item>
    <date>10.04.2007</date>
    <description headline="Neues zu XML">
      XML macht Spaß.
```

- News (Root-Element) is provided with a namespace
  - Namespace corresponds to the targetNamespace of the schema, which is bound via schemaLocation
  - schemaLocation forwards the XML processor where the schema to be used (URI:…news) can be found (URL:…news.xsd)

XML Instance Validation



```
<?xml version="1.0" encoding="utf-8"?>
<News xmlns="http://example.org/news"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://example.org/news K05-02.xsd">
  <Item>
    <date>10.04.2007</date>
    <description headline="Neues zu XML">
      XML macht Spaß.
    </description>
  </Item>
```

The schema referenced from this location in your document contains errors.

```
  xsi:schemaLocation="http://example
  <Item>
    <date>10.04.2007</date>
    <description headline="Neues zu
      XML macht Spaß.
    </description>
    <fehler/>
  </I
  <Item>
    <date>11.04.2007</date>
    <description headline="XML Teil 2">
      XML macht mit DTD noch mehr Spaß.
    </description>
```

The element 'Item' in namespace 'http://example.org/news' has invalid child element 'fehler' in namespace 'http://example.org/news'.

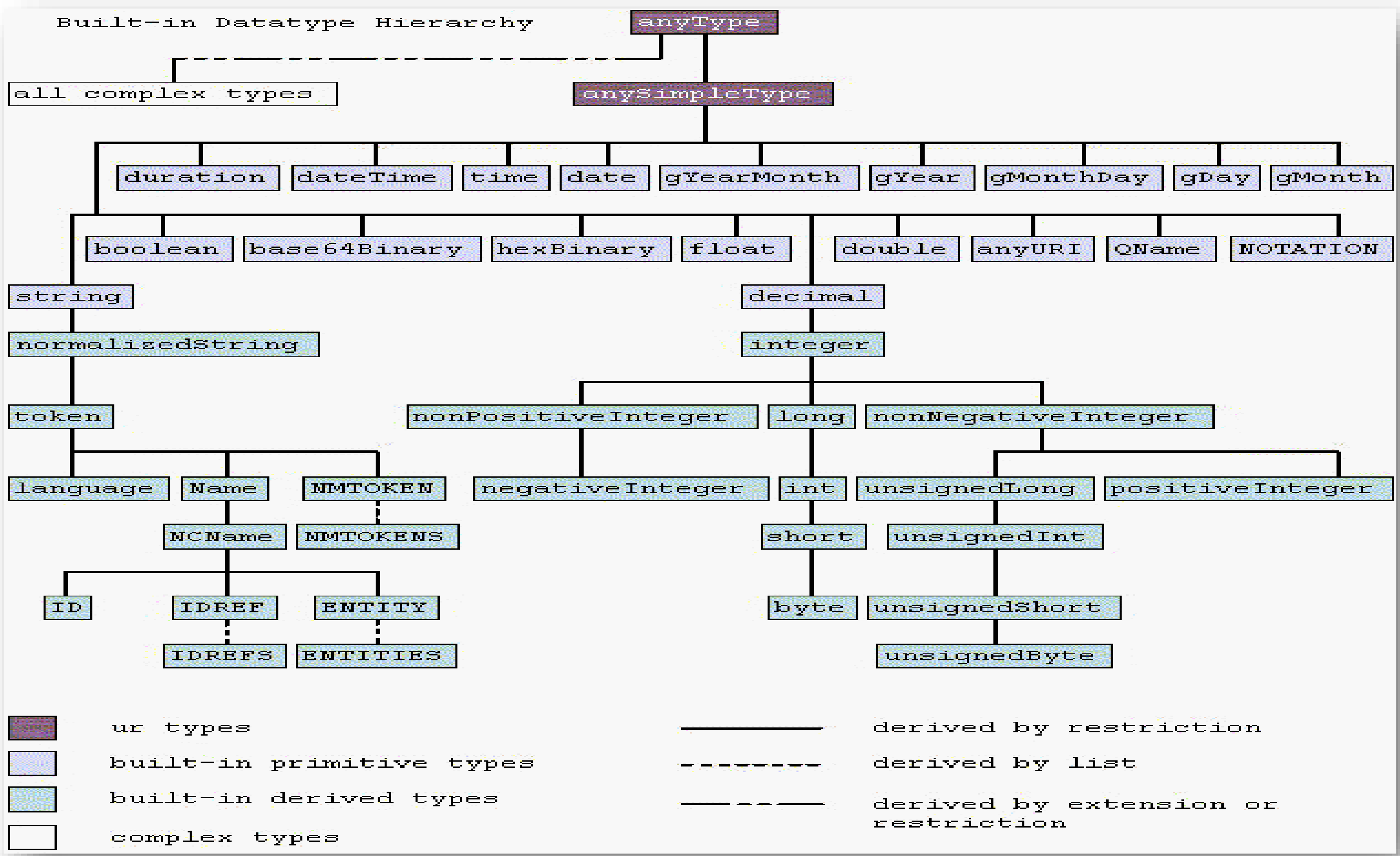
- The instance (XML) is checked against the rules defined in the schema
- The schema (XML) is checked against the rules defined in the W3C-XML schema.

elementFormDefault

- If elements of an instance have to belong to a namespace
  - elementFormDefault is the validator directive (default is "unqualified", element is not checked for namespace assignment)
  - Typically set to "qualified“

```
<?xml version="1.0" encoding="utf-8" ?>
<xsd:schema
        targetNamespace="http://example.org/Names"
        elementFormDefault="qualified"
        xmlns="http://example.org/Names"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
</xsd:schema>
```

DTD Data Types and Much More



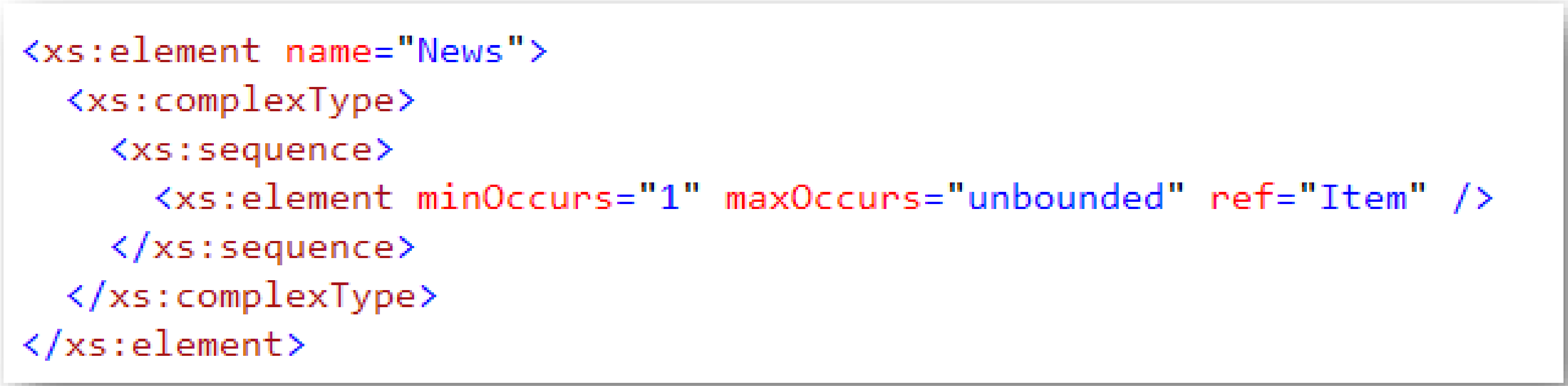
Quelle: W3C

Again News…

```
<!DOCTYPE News [
 <!ELEMENT News (Item+)>
 <!ELEMENT Item (date,description?)>
 <!ELEMENT date (#PCDATA)>
 <!ELEMENT description (#PCDATA)>
 <!ATTLIST description headline CDATA
   #REQUIRED>
]>
```

Again News: News

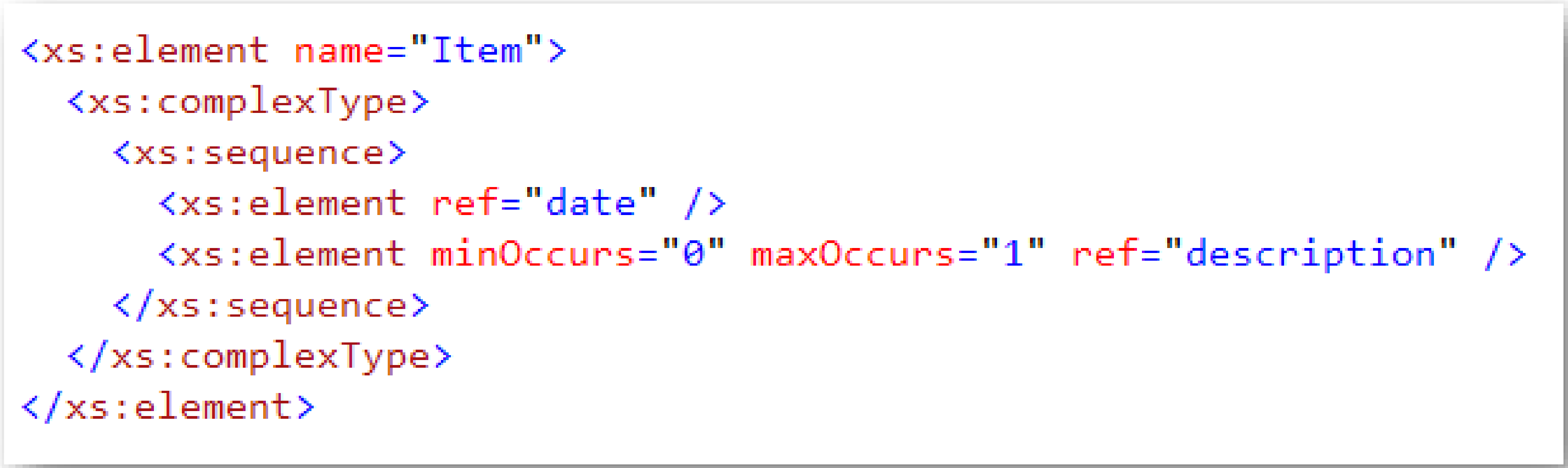
- <!ELEMENT News (Item+)>
- News is a complex element (not a simple data type):



```
<xs:element name="News">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="unbounded" ref="Item" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Again News: Item

- <!ELEMENT Item (date,description?)>
- Item is a complex element (not a simple data type):



```
<xs:element name="Item">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="date" />
      <xs:element minOccurs="0" maxOccurs="1" ref="description" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Again News: Item (2)

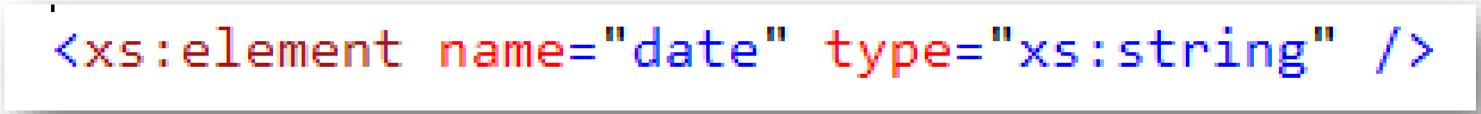
- <!ELEMENT Item (date,description?)>
- *date,description?* could also be a separate type.
  - *Reuse* of elements is possible
  - DateDescr is a so-called *Named Type*

```
<xs:complexType name="DateDescr">
  <xs:sequence>
    <xs:element ref="date" />
    <xs:element minOccurs="0" maxOccurs="1" ref="description" />
  </xs:sequence>
</xs:complexType>

<xs:element name="Item" type="DateDescr"/>
```

Again News: date

- <!ELEMENT date (#PCDATA)>
- Element is a simple type (simple Type):



```
<xs:element name="date" type="xs:string" />
```

Again News: date (2)

- <!ELEMENT date (#PCDATA)>
- Element is a simple type (simple Type) without elements or attributes:

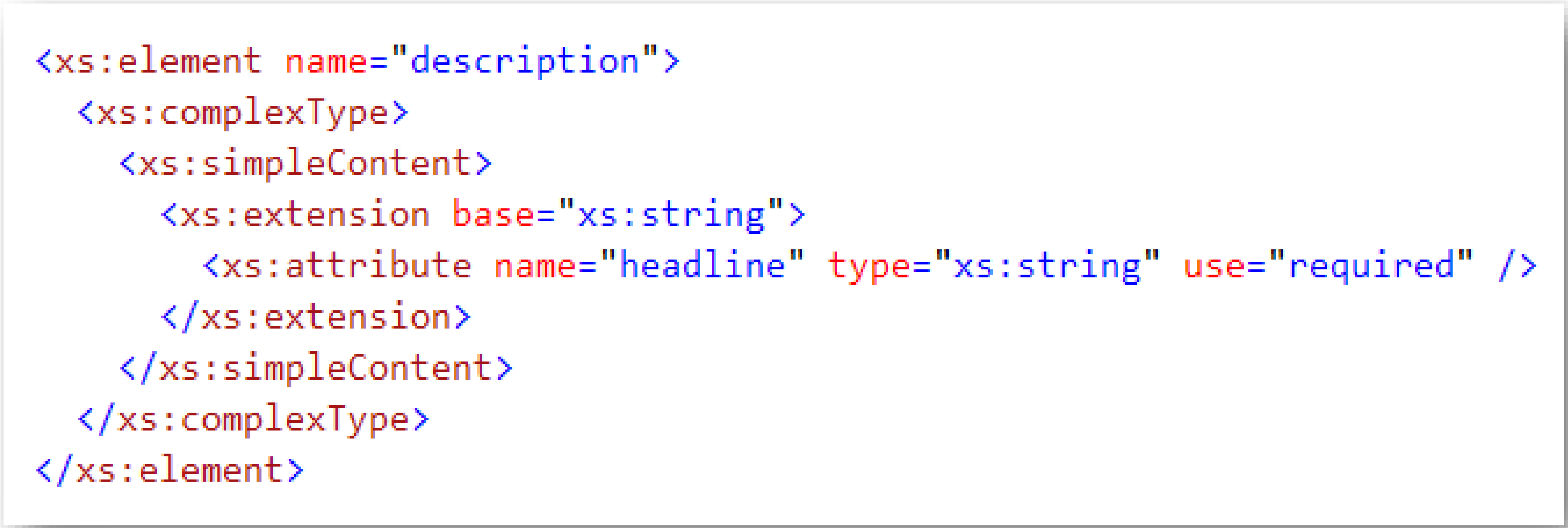
```
<xs:element name="date" type="xs:string" />
```

- What‘s not possible in DTD is simply:

```
<xs:element name="date" type="xs:date" />
```

Again News: description

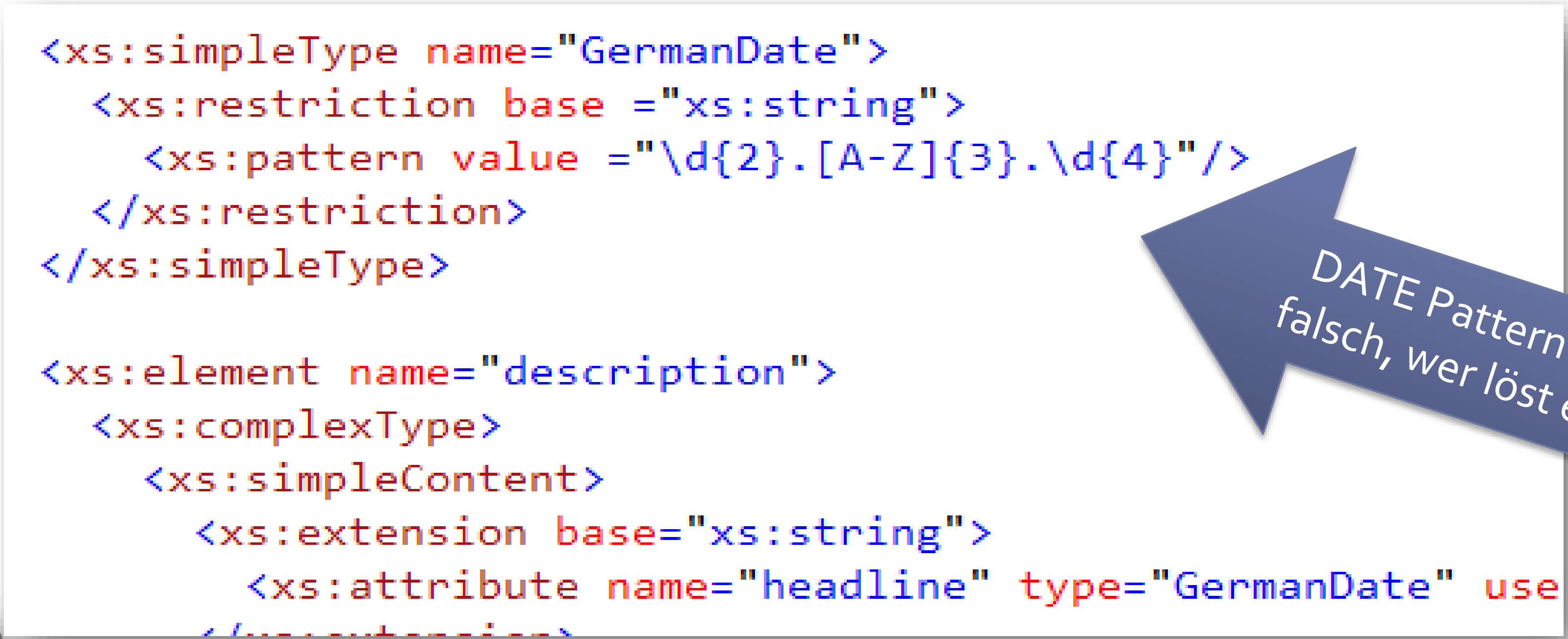
- <!ELEMENT description (#PCDATA)>
  - <!ATTLIST description headline CDATA #REQUIRED>
  - Description is complex, contains an attribute



```
<xs:element name="description">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="headline" type="xs:string" use="required" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

Again News & Attributes

- Inheritance of simple data types
  - Example date: 19.APR.2007
  - Apart from restrictions other „Facets“ are possible, such as Range, Enumeration, List, Union



```
<xs:simpleType name="GermanDate">
  <xs:restriction base ="xs:string">
    <xs:pattern value ="\d{2}.[A-Z]{3}.\d{4}"/>
  </xs:restriction>
</xs:simpleType>

<xs:element name="description">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="headline" type="GermanDate" use
```

DATE Pattern ist falsch, wer löst es?