data<2>type®

Gültige Dokumente = korrekte Dokumente?

Qualitätssicherung von XML-Dateien: DTD - Schema - Schematron – RelaxNG

Donnerstag, 24. Mai 2012

Zur Person:

Manuel Montero Pineda

Dipl.-Wirtschaftsinformatiker (FH), M.A.

XML-Entwickler und Berater im Bereich XML-Schema, OOXML, XSLT, XSL-FO, uvm..

Veröffentlichungen u.a. „Professionelle XML-Verarbeitung mit Word“, „XSL-FO in der Praxis“ und „Schematron“ erschienen beim dpunkt-Verlag.

Geschäftsführer der Firma data2type GmbH.

Stufen der Validierung von XML-Dokumenten

Ablauf einer XML-Prüfung

1. Prüfung der Syntax (Wohlgeformheit).
2. Prüfung der Grammatik (parsen gegen eine DTD, XML-Schema, Relax NG, etc.).
3. Prüfung der Kohärenz (Prüfung auf sogenannte Business-Rules) NEU.

Danach Freigabe für weitere Verarbeitungsprozesse wie z.B. XSLT-Stylesheets für Crossmediale Ausgaben.

Wohlgeformtheit

Ein Auszug der wichtigsten Regeln:

1. Elemente müssen geschachtelt sein.
2. Namenskonventionen müssen eingehalten werden.
   - Namen müssen mit einem Buchstaben beginnen.
   - Namen können darüberhinaus auch Zahlen, Interpunktionszeichen enthalten.
   - Namen dürfen keine Leerzeichen und Doppelpunkte enthalten.
3. Attribute müssen einen Wert haben, der in Hochkommas oder Anführungsstrichen stehen muss. Dieser Wert kann auch leer sein.

...

Sie Fragen sich warum ich das sage?

Wohlgeformtheit wird in manchen Firmen v.a. Verlagen als Hightech angesehen!

Schemasprachen

Bei Publishing-Prozessen ist die Validierung von Dokumenten ein Freigabekriterium für die weitere Abläufe.

Hierzu verwendete Schemasprachen sind:

- DTD (gerade bei Verlagen anzutreffen, wird aber nicht weiterentwickelt)
- XML-Schema (vom W3C herausgegebene Schema-Sprache)
- RelaxNG (ISO-zertifizierte Sprache)

Diese sogenannten grammatikbasierten Schemasprachen regeln beispielsweise das Auftreten und die Anordnung von Elementen, die Häufigkeit jedes Elementes sowie die Datentypen von Elementen und Attributen.

Document Type Definitions (DTD)

Noch aus SGML Zeiten stammen die DTDs.

Sind ein ISO/W3C-Standard.

- W3C innerhalb der XML-Recommendation http://www.w3.org/TR/2000/REC-xml-20001006
- Document Schema Definition Languages (DSDL) Part 9 http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=41009

Werden bedingt weiterentwickelt.

Document Type Definitions (DTD)

Nachprüfbar mittels Parser, der in den XML-Editoren integriert ist.
Eine DTD beschreibt:

- welche Elemente es gibt
- welche Elemente optional sind und welche nicht
- die Reihenfolge der Elemente
- die Häufigkeit der Elemente
- die Attribute, welche die Elemente besitzen
- ...

Document Type Definitions (DTD)

Verbreitung der DTDs

- Oft anzutreffen bei Verlagen und z.T. auch in der Technischen Dokumentation.
- Im Bereich Softwareentwicklung und Datenbanken nahezu unbekannt.

Beispiel XML mit DTD-Aufruf

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE arch:arche SYSTEM "Arche.dtd">
<arch:arche xmlns:arch='http://www.schematron.de/arche'>
    <arch:ladung>
        <arch:zimmer>
            <arch:tier geschlecht="weiblich" fleischfresser="nein">
                <arch:art>Zebra</arch:art>
                <arch:gewicht einheit="kg">200</arch:gewicht>
                <arch:alter>40</arch:alter>
            </arch:tier>
            <arch:tier geschlecht="männlich" fleischfresser="nein">
                <arch:art>Zebra</arch:art>
                <arch:gewicht einheit="kg">250</arch:gewicht>
                <arch:alter>40</arch:alter>
            </arch:tier>
            <arch:tier geschlecht="männlich" fleischfresser="nein">
                <arch:art>Zebra</arch:art>
                <arch:gewicht einheit="kg">280</arch:gewicht>
                <arch:alter>40</arch:alter>
            </arch:tier>
            <arch:tier geschlecht="weiblich" fleischfresser="ja">
                <arch:art>Löwe</arch:art>
                <arch:gewicht einheit="kg">200</arch:gewicht>
                <arch:alter>16</arch:alter>
            </arch:tier>
        </arch:zimmer>
        <arch:zimmer>
            <arch:tier geschlecht="weiblich" fleischfresser="ja">
                <arch:art>Löwe</arch:art>
                <arch:gewicht einheit="kg">200</arch:gewicht>
                <arch:alter>30</arch:alter>
            </arch:tier>
```

DTD-Beispiel

```
<?xml encoding="UTF-8"?>
<!-- Beispiel einer Archen-DTD  -->
<!ELEMENT arch:arche (arch:ladung,arch:maxReproduktionsalter,arch:nutzlast)>
<!ATTLIST arch:arche
  xmlns:arch CDATA #FIXED 'http://www.schematron.de/arche'
>

<!ELEMENT arch:ladung (arch:zimmer)+>

<!ELEMENT arch:maxReproduktionsalter (arch:tier_art)+>

<!ELEMENT arch:nutzlast (#PCDATA)>
<!ATTLIST arch:nutzlast
  einheit (kg | t) #REQUIRED>

<!ELEMENT arch:zimmer (arch:tier)+>

<!ELEMENT arch:tier_art (arch:name,arch:männlich,arch:weiblich)>

<!ELEMENT arch:tier (arch:art,arch:gewicht,arch:alter)>
<!ATTLIST arch:tier
  fleischfresser (ja | nein) #REQUIRED
  geschlecht (männlich | weiblich) #REQUIRED>

<!ELEMENT arch:name (#PCDATA)>

<!ELEMENT arch:männlich (#PCDATA)>

<!ELEMENT arch:weiblich (#PCDATA)>

<!ELEMENT arch:art (#PCDATA)>

<!-- Hier bitte nur ganze Zahlen  -->
<!ELEMENT arch:gewicht (#PCDATA)>
<!ATTLIST arch:gewicht
  einheit CDATA #REQUIRED>
```

Probleme bei DTDs

- In der W3C Recommendation keine Namensraumunterstützung
- Kaum Datentypen.
- Unterschied zwischen einfachen Datentypen bei Attributen und Elementen. Zum Beispiel keine geschlossene Liste für Elementinhalte möglich.
- Keine XML-Syntax in den DTDs.
- Keine Zuordnung von Kommentaren möglich.
- Keine Weiterentwicklung des Standards.
- Neuere Software unterstützt oft keine DTDs.
- Rekursionen können nicht ausgeschlossen werden.
- ...

Vorteile von DTDs

- Sehr einfach zu erlernen.

XML Schema

Am 2.Mai 2002 wurde vom W3C nach einer zweijährigen Entwicklungszeit die Recommendation für XML-Schema verabschiedet.

Schemata sind mächtiger als DTDs und bieten mehr Möglichkeiten Datenkontrolle als DTDs.

Vor allem wurde Wert auf die Datentypen gelegt. Neben dutzenden von vorgefertigten Datentypen, wie Integer, date, etc. lassen sich auch Typen selbst definieren, bis hin zur Kontrolle über Reguläre Ausdrücke und Wertebereichen.

Jede DTD lässt sich informationsverlustfrei in ein Schema umwandeln.

Beispiel XML mit XML-Schema-Aufruf

```xml
<?xml version="1.0" encoding="UTF-8"?>
<arche xmlns="http://www.schematron.de/arche" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.schematron.de/arche arche.xsd">
    <ladung>
        <zimmer>
            <tier geschlecht="weiblich" fleischfresser="nein">
                <art>Zebra</art>
                <gewicht einheit="kg">200</gewicht>
                <alter>40</alter>
            </tier>
            <tier geschlecht="männlich" fleischfresser="nein">
                <art>Zebra</art>
                <gewicht einheit="kg">250</gewicht>
                <alter>40</alter>
            </tier>
            <tier geschlecht="männlich" fleischfresser="nein">
                <art>Zebra</art>
                <gewicht einheit="kg">280</gewicht>
                <alter>40</alter>
            </tier>
            <tier geschlecht="weiblich" fleischfresser="ja">
                <art>Löwe</art>
                <gewicht einheit="kg">200</gewicht>
                <alter>16</alter>
            </tier>
        </zimmer>
        <zimmer>
            <tier geschlecht="weiblich" fleischfresser="ja">
                <art>Löwe</art>
                <gewicht einheit="kg">200</gewicht>
                <alter>30</alter>
            </tier>
        </zimmer>
```

XML-Schema-Beispiel

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.schematron.de/arche" xmlns="http://www.schematron.de/arche"
    elementFormDefault="qualified">
    <xs:element name="alter">
        <xs:simpleType>
            <xs:restriction base="xs:positiveInteger">
                <xs:maxInclusive value="200"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:element>
    <xs:element name="arche">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="ladung"/>
                <xs:element ref="maxReproduktionsalter"/>
                <xs:element ref="nutzlast"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="art" type="xs:string"/>
    <xs:element name="gewicht">
        <xs:annotation>
            <xs:documentation>
                Das ist die Beschreibung von gewicht!
            </xs:documentation>
        </xs:annotation>
        <xs:complexType mixed="true">
            <xs:attribute name="einheit" type="gewichtsEinheit" use="required"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="ladung">
        <xs:complexType>
```

Einfache Datentypen bei XML-Schema

Beispielinstanz ohne vernünftige Datenprüfung:

```
<?xml version="1.0" encoding="UTF-8"?>
<Personendaten xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Beispiel1_Personendaten.xsd">
        <Vorname>Manuel</Vorname>
        <Nachname>Montero</Nachname>
        <PLZ>PLZ-69118</PLZ>
        <Ort>Heidelberg</Ort>
        <Geburtsdatum>31.02.1980</Geburtsdatum>
</Personendaten>
```

Einfache Datentypen bei XML-Schema

Namensraum für schemavalidierte XML-Dateien:
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

Pfad zum Schema:
xsi:noNamespaceSchemaLocation="Beispiel1_Personendaten.xsd"

Einfache Datentypen bei XML-Schema

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" >
<xs:element name="Personendaten">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Vorname" type="xs:string"/>
                <xs:element name="Nachname" type="xs:string"/>
                <xs:element name="PLZ" type="xs:integer"/>
                <xs:element name="Ort" type="xs:string"/>
                <xs:element name="Geburtsdatum" type="xs:date"/>
            </xs:sequence>
        </xs:complexType>
</xs:element>
</xs:schema>
```

Einfache Datentypen bei XML-Schema

Die valide XML-Instanz:

```
<?xml version="1.0" encoding="UTF-8"?>
<Personendaten xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Beispiel1_Personendaten.xsd">
        <Vorname>Manuel</Vorname>
        <Nachname>Montero</Nachname>
        <PLZ>69118</PLZ>
        <Ort>Heidelberg</Ort>
        <Geburtsdatum>1980-02-28</Geburtsdatum>
</Personendaten>
```

Einfache Datentypen bei XML-Schema

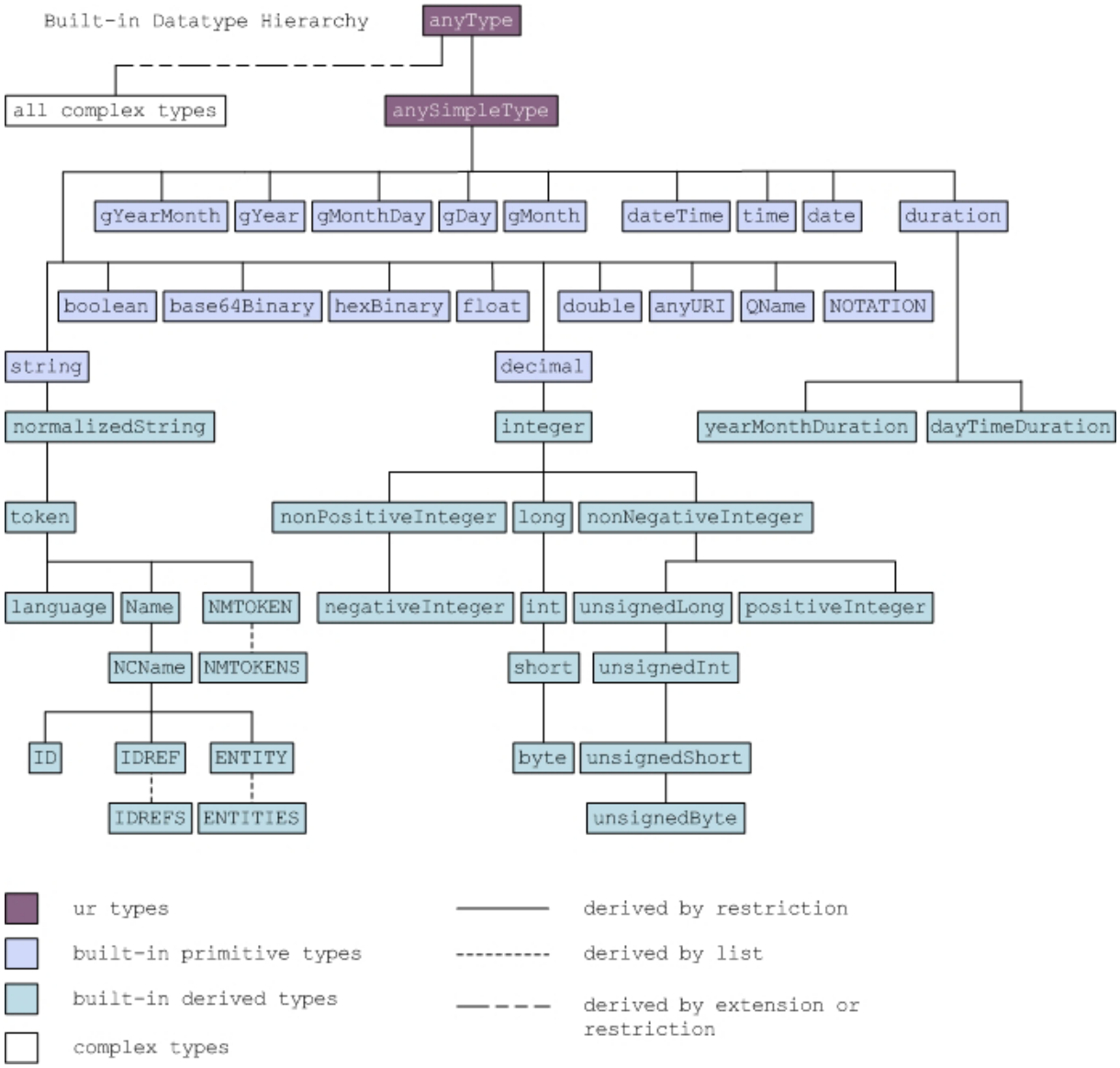


Bild aus: http://www.w3.org/TR/xmlschema-2/

Wertebereiche

Einschränkungen von Werten

Mit dem Element restriction lassen sich Wertebereiche, Listen und ähnliche Einschränkungen des Grundtypen definieren:

Wertebereiche:

```
<xs:element name="PLZ">
    <xs:simpleType>
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="01001"/>
            <xs:maxInclusive value="99999"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

Reguläre Ausdrücke

Wertebereiche von Zeichenketten:

```
<xs:element name="NamenIni" type="Initialen"/>
    <xs:simpleType name="Initialen">
        <xs:restriction base="xs:string">
            <xs:pattern value="[A-Z][A-Z]"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

Pattern erlaubt die Verwendung von regulären Ausdrücken (regular expression)

Dokumentation

Alle XML-Entwicklungsumgebungen und auch AntillesXML (http://www.data2type.de/software/antillesxml) bieten die Möglichkeit vollautomatisch aus XML-Schemata heraus Dokumentationen zu generieren.

```
<xs:element name="h1">
  <xs:annotation>
    <xs:documentation>Das Element &lt;h1&gt; ist eine Überschrift der ersten Ebene, das an allen Positionen gestattet ist,
        wo das Element &lt;block&gt; erlaubt ist. Da nicht immer auf eine Kapitelstruktur verwendet werden muss,
        können so Überschriften manuell eine Ebene zugewiesen werden. Hierzu gibt es die Elemente &lt;h1&gt;,
        &lt;h2&gt;, &lt;h3&gt;, &lt;h4&gt; und &lt;h5&gt;, die jeweils für die entsprechende Ebene
        gelten.</xs:documentation>
    <xs:documentation>Als Inhalt hat das Element Text, der mit inzeiligen Elementen ausgezeichnet sein darf (Siehe
        &lt;inzeilig&gt;) und maximal ein &lt;term&gt;-Element, mit dem der Überschrift ein Label zugeordnet werden
        kann.</xs:documentation>
  </xs:annotation>
  <xs:complexType mixed="true">
    <xs:sequence>
...
```

Beispiel eine Dokumentation

Simple Type: css

Super-types:	xs:string < **css** (by restriction)
Sub-types:	None

Name	css
Used by (from the same schema document)	Element h1 , Element h2 , Element h3 , Element h4 , Element h5 , Element para , Element variablelist , Element varlistentry , Element itemizedlist , Element orderedlist , Element listitem , Element mediaobject , Element imageobject , Element table , Element thead , Element tfoot , Element tbody , Element colgroup , Element col , Element tr , Element td
Content	• Base XSD Type: string
Documentation	

Eigenschaft	Beschreibung
CSS-Maßangaben	Alle Maßangaben können relativ oder absolut gemacht werden. Relative Maßangaben sind beispielsweise in Pixel (px), in Prozent (%), in 'ex' oder in 'em'. 'ex' und 'em' beziehen sich dabei auf die in dem Bereich gültige Schriftgröße. 1ex ist dabei die höhe eines x (Kleinbuchstabe) und 1em dabei die Höhe eines M (Großbuchstabe). Absolute Angaben können in Punkt (pt), Pica (pc), Inch (in), Millimeter (mm) oder Zentimeter (cm) gemacht werden. Als "Komma" wird immer nach der amerikanischen Art der Punkt verwendet.
color	Legt die Schriftfarbe für den textuellen Inhalt des Elementes fest. Für die Farbe gibt es die Werte black, maroon, green, olive, navy, purple, teal, silver, gray, red, lime, yellow, blue, fuchsia, aqua und white (HTML-Farben)sowie eine RGB-Angabe die durch ein '#' vorstehend gekennzeichnet ist und eine zweistellige Hexadezimalangabe für den Rot-, Grün- und Blau-Wert aneinander gereiht gemacht wird. Beispiel: #00FFA0 hat die Werte Rot=0(00), Grün=255(FF) und Blau=100(A0).
background-color	Legt die Hintergrundfarbe für den ausgewählten Bereich (z.B. Absatz, Tabellenzelle, Tabellenspalte, etc.) fest. Für die Farbe gibt es die Werte black, maroon, green, olive, navy, purple, teal, silver, gray, red, lime, yellow, blue, fuchsia, aqua und white (HTML-Farben)sowie eine RGB-Angabe die durch ein '#' vorstehend gekennzeichnet ist und eine zweistellige Hexadezimalangabe für den Rot-, Grün- und Blau-Wert aneinander gereiht gemacht wird. Beispiel: #00FFA0 hat die Werte Rot=0(00), Grün=255(FF) und Blau=100(A0).
border-bottom-style	Legt die style-Eigenschaft für die untere Rahmenlinie fest. Die style-Eigenschaft ist die "Strichart" der Linie. Werte: `none` keine Rahmenlinie `hidden` versteckte Rahmenlinie `dotted` gepunktete Linie `dashed` gestrichelte Linie `solid` durchgezogene Linie (Default-Wert) `double` doppelt durchgezogene Linie
border-left-style	Legt die style-Eigenschaft für die linke Rahmenlinie fest. Die style-Eigenschaft ist die "Strichart" der Linie. Werte: `none` keine Rahmenlinie `hidden` versteckte Rahmenlinie

Vorteile von XML-Schema gegenüber DTDs

- Es ist einfacher den Elementinhalt und Dokumentaufbau zu beschreiben.
- Bessere Überprüfung der Dateninhalte.
- Einfacheres Zusammenspiel mit Datenbanken, da dort ähnliche Datentypen definiert werden können.
- Sehr genaue Definition von erlaubten individuellen Zeichenfolgen möglich.
- Konvertierung (casten) zwischen verschiedenen Datentypen ist möglich. Z.B. mache aus einem String einen Integer.
- Schemata unterstützen Namensräume. Dadurch ist eine Wiederverwendung in anderen Projekten einfacher.
- Schemata sind selbst in der XML-Syntax geschrieben.
- Dadurch können sie von XML-Parsern auf Validität überprüft werden.
- Schemata können prinzipiell mit XSLT verändert oder ausgewertet werden.

Vorteile von XML-Schema gegenüber DTDs

Weitere Vorteile von XML-Schema gegenüber DTDs

- Schemata können mit jedem XML-Editor erstellt werden.
- Integriertes Dokumentationsmodell.
- Sehr gute Modularisierungsmöglichkeiten.

Nachteile von XML-Schema gegenüber DTDs

- Syntax von DTD ist zwar nicht XML-basiert aber dennoch relativ einfach.
- Große Schemata lassen sich ohne Entwicklungsumgebungen bzw. speziellen Editoren kaum pflegen bzw. entwickeln.

RELAX NG

Regular **La**nguage Description for **X**ML **N**ew **G**eneration (RELAX NG) ist eine Schemasprache für XML.

Sie basiert auf Makoto Muratas RELAX und James Clarks TREX. RELAX NG ist beschrieben in einem Dokument der OASIS RELAX NG Technical Committee (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=relax-ng) und darüber hinaus als internationaler Standard ISO/IEC 19757-2 innerhalb der Document Schema Definition Languages (DSDL).

Beispiel XML mit RELAX NG-Schema-Aufruf

```xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="Arche.rng" type="application/xml" schematypens="http://relaxng.org/ns/structure/1.0"?>
<arche xmlns="http://www.schematron.de/arche" >
    <ladung>
        <zimmer>
            <tier geschlecht="weiblich" fleischfresser="nein">
                <art>Zebra</art>
                <gewicht einheit="kg">200</gewicht>
                <alter>40</alter>
            </tier>
            <tier geschlecht="männlich" fleischfresser="nein">
                <art>Zebra</art>
                <gewicht einheit="kg">250</gewicht>
                <alter>40</alter>
            </tier>
            <tier geschlecht="männlich" fleischfresser="nein">
                <art>Zebra</art>
                <gewicht einheit="kg">280</gewicht>
                <alter>40</alter>
            </tier>
            <tier geschlecht="weiblich" fleischfresser="ja">
                <art>Löwe</art>
                <gewicht einheit="kg">200</gewicht>
                <alter>16</alter>
            </tier>
        </zimmer>
        <zimmer>
            <tier geschlecht="weiblich" fleischfresser="ja">
                <art>Löwe</art>
                <gewicht einheit="kg">200</gewicht>
                <alter>30</alter>
```

Relax NG-Schema-Beispiel XML-Syntax

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <grammar ns="http://www.schematron.de/arche"
3      xmlns="http://relaxng.org/ns/structure/1.0"
4      datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
5      <start>
6          <element name="arche">
7              <element name="ladung">
8                  <oneOrMore>
9                      <element name="zimmer">
10                         <oneOrMore>
11                             <element name="tier">
12                                 <attribute name="fleischfresser">
13                                     <data type="NCName"/>
14                                 </attribute>
15                                 <attribute name="geschlecht">
16                                     <data type="NCName"/>
17                                 </attribute>
18                                 <element name="art">
19                                     <data type="NCName"/>
20                                 </element>
21                                 <element name="gewicht">
22                                     <attribute name="einheit">
23                                         <data type="NCName"/>
24                                     </attribute>
25                                     <data type="integer"/>
26                                 </element>
27                                 <element name="alter">
28                                     <data type="integer"/>
29                                 </element>
30                             </element>
```

Relax NG-Schema Kompakt-Syntax

```
default namespace = "http://www.schematron.de/arche"
namespace xsi = "http://www.w3.org/2001/XMLSchema-instance"

start =
  element arche {
    attribute xsi:schemaLocation { text },
    element ladung {
      element zimmer {
        element tier {
          attribute fleischfresser { xsd:NCName },
          attribute geschlecht { xsd:NCName },
          element art { xsd:NCName },
          element gewicht {
            attribute einheit { xsd:NCName },
            xsd:integer
          },
          element alter { xsd:integer }
        }+
      }+
    },
    element maxReproduktionsalter {
      element tier_art {
        element name { xsd:NCName },
        element männlich { xsd:integer },
        element weiblich { xsd:integer }
      }+
    },
    element nutzlast {
      attribute einheit { xsd:NCName },
      xsd:integer
    }
  }

```

Vorteile von Relax NG gegenüber XML-Schema

- Bietet mehr Prüfmöglichkeiten
- Bessere Syntax.
- Intuitive Kompaktschreibweise

Nachteile von Relax NG gegenüber XML-Schema

- Wird von wenigen Editoren unterstützt.
- Scheint sich nicht durchzusetzen.

Kurzmatrix für Schemasprachen

	DTD	Schema	Relax NG
Verbreitung	++	+++	-
Mächtigkeit	-	++	+++
Zukunftsfähig	-	+++	+
Softwareunter-stützung	+	++	-
Datentypen	-	++	++

Schematron

XML

```
<arche>
    <zimmer>
        <tier
fleischfresser="nein">
            <art>Zebra</art>
        </tier>
        <tier fleischfresser="ja">
            <art>Gepard</art>
        </tier>
    </zimmer>
</arche>
```

XSD 1.0

```
<xs:element name="zimmer">
          <xs:complexType>
              <xs:sequence>
                  <xs:element minOccurs="2"
                      maxOccurs="unbounded" ref="tier"/>
              </xs:sequence>
          </xs:complexType>
</xs:element>
```

Kohärenzprüfung

Arche-Noah-Business-Rule:

Gibt es Fleischfresser und Vegetarier in einem Zimmer?

```
tier[@fleischfresser='ja'] and tier[@fleischfresser='nein']
```

Möglichkeiten von XPATH:

- Zugriff auf alle Knoten des XML-Dokuments
- Verschiedene Knotenoperationen
- Einsatz der XPATH-Funktionsbibliothek

Kohärenzprüfung

Was ist Schematron?

- Teil des DSDL (ISO 19757)
- Hosting Sprache in XML formuliert

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://purl.oclc.org/dsdl/schematron" queryBinding="[query language]">
        <pattern>
                <rule context="[pattern]">
                        <assert test="[boolean-expression]"/>
                </rule>
        </pattern>
</schema>
```

Kohärenzprüfung

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://purl.oclc.org/dsdl/schematron" queryBinding="xslt2">
  <pattern>
    <rule context="zimmer">
        <assert test="not(tier[@fleischfresser='ja'] and tier[@fleischfresser='nein'])
                      "/>
        </pattern>
</schema>
```

Was ist möglich bei der Kohärenz-Prüfung?

- Überprüfung von Kindstrukturen in Abhängigkeit von Attributwerten
- Überprüfung der Häufigkeit von Elementen in Abhängigkeit von Geschwisterelementen.
- Überprüfung komplexer Inhaltsregeln, wie z.B. die fehlerhafte Verwendung von Überspannungen bei CALS-Tabellen.
- Einschränkung bei der Verwendung von rekursiven Strukturen.
- Vermeidung von speziellen textuellen Inhalten, wie Folgen von Unterstrichen oder Punkten, die im dokumentenzentrierten Umfeld oft als Gestaltungselemente „missbraucht“ werden.
- Überprüfung von Summen, z.B. innerhalb von Rechnungsformularen.
- Überprüfung des Einsatzes von Processing-Instructions.

Was ist möglich bei der Kohärenzprüfung?

Lösungsansatz ist Schematron

- Schematron ist ein ISO-Standard seit Mai 2006.
- Die benötigten Programme sind kostenlos verfügbar.
- Unterstützung durch diverse XML-Editoren wie oXygen, XMLmind, etc.
- Einfache Programmierung mittels XPath.
- Individualisierte Fehlermeldungen.

Beispiel für ein Schematronschema

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
<pattern id="Unterstriche">
   <rule context="a">
      <assert test="not(contains(., '__'))">
         Statt Unterstrichen sollte das Element leerzeile verwendet werden!</assert>
   </rule>
</pattern>
</schema>
```

Einsatzszenario für Schematron

Ein Verlag lässt im Ausland Daten erfassen.

Nach einer Sichtung der Daten fallen einige Fehler auf, die nicht zu Validierungsproblemen führen, aber dennoch im Rahmen einer Qualitätssicherung behoben werden müssen.

Außerdem soll dokumentiert werden an welchen Stellen und wie häufig ein Eingriff notwendig war.

Einsatzszenario für Schematron

Beispiel für eine Anforderung die beim Sichten aufgefallen ist.

Falsche Silbentrennungen kommen in unterschiedlichster Form vor:

- Feste Trennung: Schema-tron
- Feste Trennung mit Break: Schema-
tron
- Bedingte Trennung mit Leerschritt: Schema tron
- Mehrfache bedingte Trennung: Schematron
- Korrekt ist: Schematron

Einsatzszenario für Schematron

In Schematron:

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
  <pattern id="_6Silbentrennung">
    <rule context="*[normalize-space(./text())]">
      <assert test="not(contains(., '&#xAD; '))">
      Silbentrennung nur mit Hilfe der Entity   "&#xAD;" Leerstellen danach sind nicht
        zulässig!</assert>
      <assert test="(not(contains(.,'&#xAD;&#xAD;')))">
      Silbentrennung nur mit Hilfe einer Entity "&#xAD;".  Entity darf nicht doppelt
        verwendet werden!</assert>
      ...
    </rule>
  </pattern>
</schema>
```

Wie nutzt man Schematron

Szenario 1: Im Editor

Editoren die Schematron unterstützen: Oxygen, XMLMind, ...

Szenario 2: Black-Box (Referenzimplementierung mit XSLT)

Szenario 3: Black-Box eingebettet in XProc

Wie nutzt man Schematron

Szenario 1: Im Editor

Editoren die Schematron unterstützen: Oxygen, XMLMind, ...

Szenario 2: Black-Box (Referenzimplementierung mit XSLT)

Szenario 3: Black-Box eingebettet in XProc

Publishingbeispiele Fußnoten

```
<para>In order to view this text in a browser, you have to start the browser.
        <footnote>
                <para>browser: Tool to view (X)HTML.
                  <footnote><para>(X)HTML: XML Hypertext Markup Language</para>
                  </footnote>
                </para>
        </footnote>
</para>
In Schematron
<rule context="footnote">
  <assert test="not(.//footnote)">
        ...
  </assert>
<rule>
```

Publishingbeispiele PIs

```
<pattern id="PIs">
    <rule context="processing-instruction()">
        <assert test="starts-with(name() 'd2t')">Diese PI stammt nicht aus von
        data2type. Bitte dokumentieren Sie Ihre Funktion. Name der PI: <value-of
        select="name()"/>
        </assert>
    </rule>
</pattern>
```

Publishingbeispiele verschachtelte Listen

```
<pattern id="Listen">
    <rule context="listitem">
     report test="count(ancestor::listitem) > 2">
          Eine Liste darf nicht mehr als drei Ebenen haben.
          Wir sind in der <value-of select="count(ancestor::listitem)+1"/>. Ebene.
     </report>
    </rule>
</pattern>
```

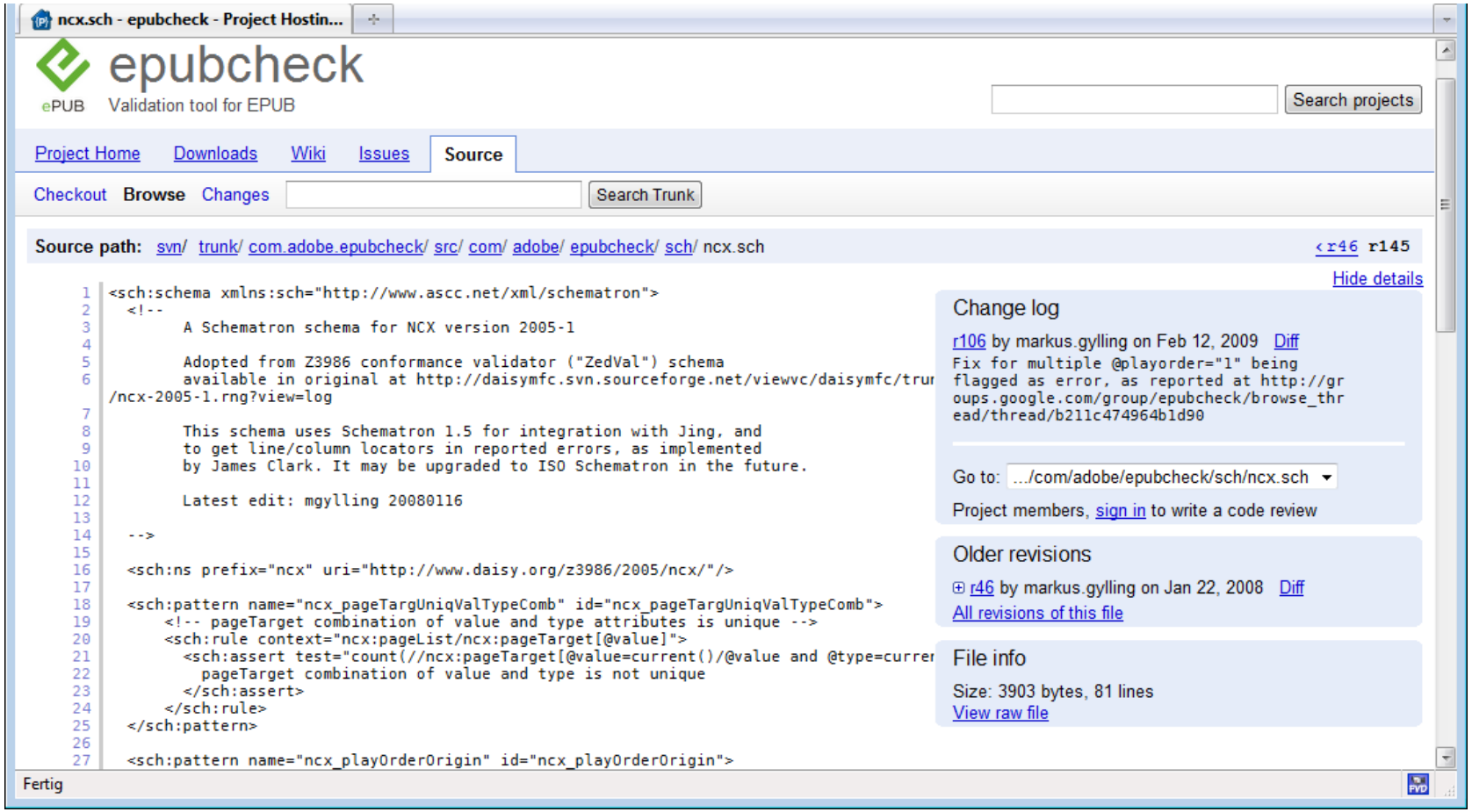
Schematron in großen Grammatiken

Wer benutzt Schematron?

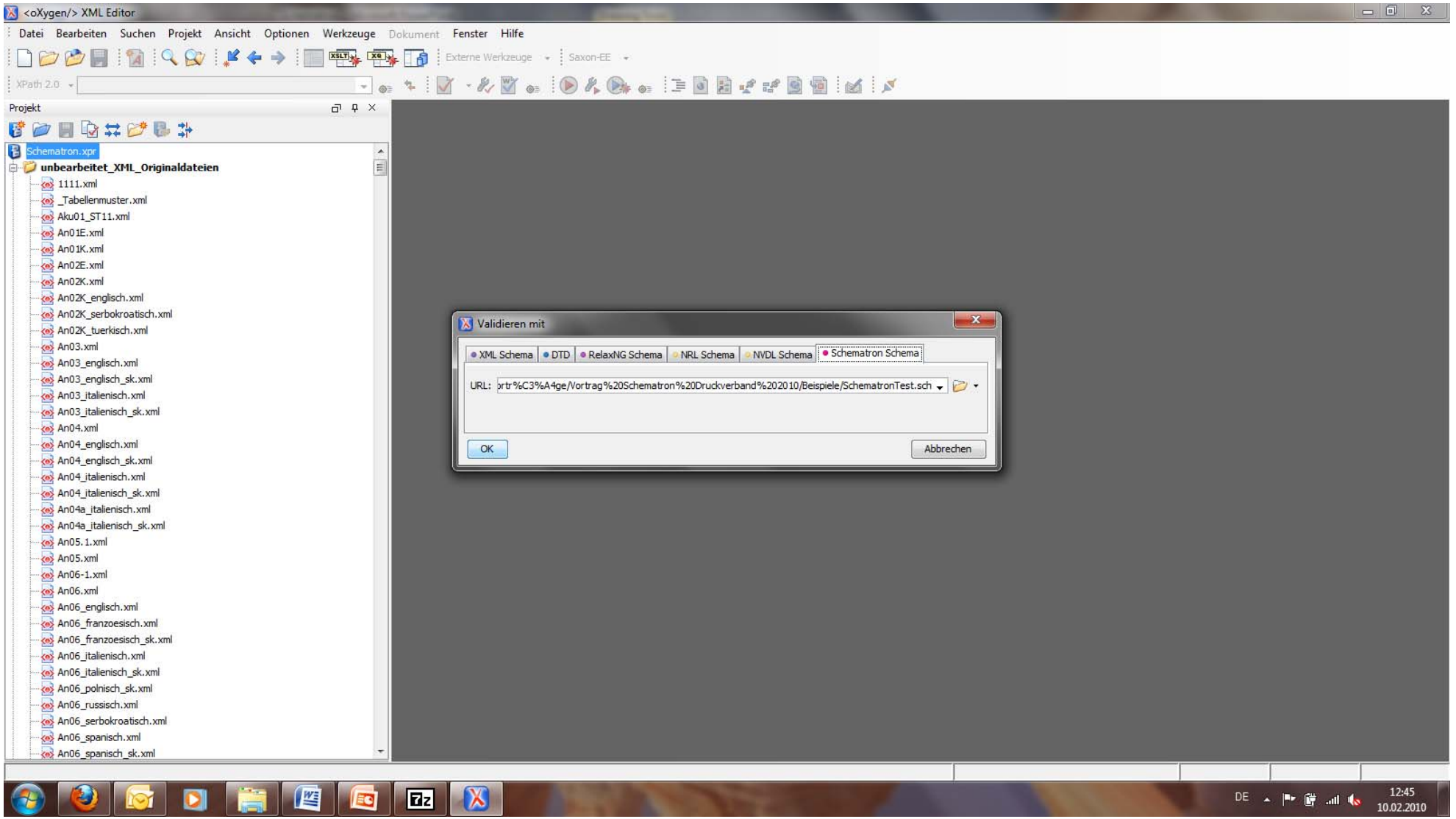
- DocBook
- DITA
- TEI
- EPUB
- ...?

Schematron in großen Grammatiken

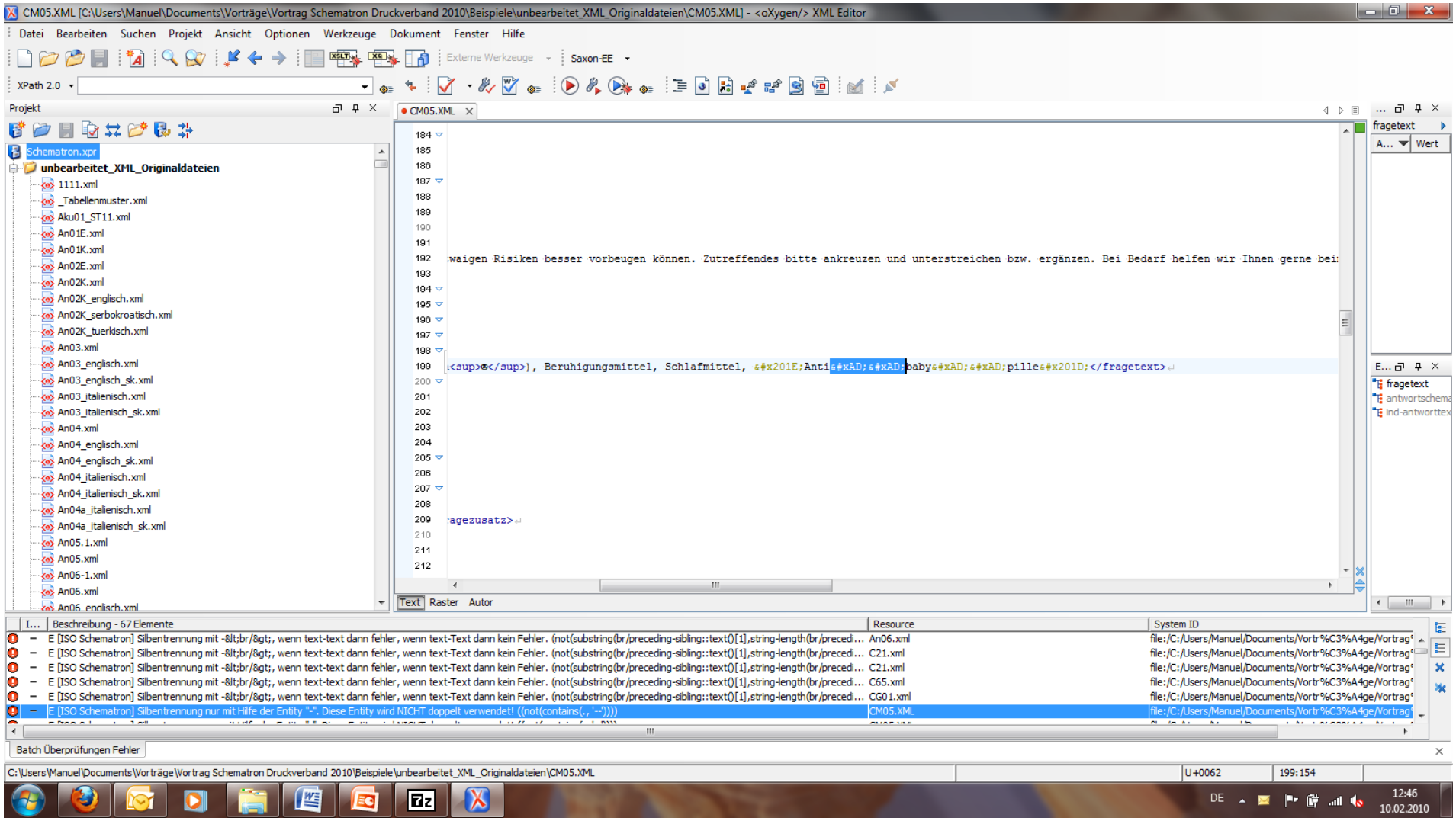
Schematron in großen Grammatiken



Schematron im Editor



Schematron im Editor



Technische Matrix

Eigenschaft	DTD	XML-Schema	RELAX NG	Schematron
XML Syntax	N	J	J	J
Namensräume	N	J	J	J
einfache und komplexe Datentypen	N	J	J	J[1]
Attribute	J	J	J	N
Attribute gruppieren[2]	N	N	J	J[1]
Optionale oder Pflichtattribute	J	J	J	J[1]
Werte bei Attributen	P	J	J	J
Bedingungen zwischen Elementeninhalten und den dazugehörigen Attributen	N	N	N	J
minimale und maximale Anzahl von vorkommenden Elementen	P	J	P	J

[1] Kann in Schematron mit Hilfe von XPath simuliert werden.

[2] Bestimmte Attribute dürfen nur zusammen oder gar nicht auftreten.

Technische Matrix

Eigenschaft	DTD	XML-Schema	RELAX NG	Schematron
Elemente, die nur unter bestimmten Bedingungen erlaubt sind	N	N	P	J
Vererbung, wie bei objektorientierten Sprachen	N	N	N	N
Einschränkungen von Dantentypen. Also eingeschränkte Varianten	N	J	P	N
Attributwerte, die Unique sein müssen	J	J	J	J
Elementinhalte, die Unique sein müssen	N	J	J	J
Definierte Konstrukte zur Dokumentierung des Quellcodes	N	J	J	J

[1] Kann in Schematron mit Hilfe von XPath simuliert werden.
[2] Bestimmte Attribute dürfen nur zusammen oder gar nicht auftreten.

Fragen?

Wieblinger Weg 92A
69123 Heidelberg
T: ++49-(0)6221-7391264
F: ++49-(0)6221-7391266
E: montero@data2type.de