Gültige Dokumente = korrekte Dokumente?

Qualitätssicherung von XML-Dateien mit Schematron

5/20/2011

Manuel Montero Pineda

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

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

Hauptautor von „Professionelle XML-Verarbeitung mit Word“ und „XSL-FO in der Praxis“ beides erschienen beim dpunkt-Verlag.

Geschäftsführer der Firma data2type GmbH.

Validierung von XML-Dokumenten

Bei Publishing-Prozessen ist die Validierung von Dokumenten ein Freigabe-kriterium 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.

Beispielinstanz

```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>
```

Grammatikbasierte Sprachen im Vergleich

DTD

```
<?xml encoding="UTF-8"?>

<!ELEMENT ns1:arche (ns1:ladung,ns1:maxReproduktionsalter,ns1:nutzlast)>
<!ATTLIST ns1:arche
  xmlns:ns1 CDATA #FIXED 'http://www.schematron.de/arche'
  xmlns:xsi CDATA #FIXED 'http://www.w3.org/2001/XMLSchema-instance'
  xsi:schemaLocation CDATA #REQUIRED>

<!ELEMENT ns1:ladung (ns1:zimmer)+>
<!ATTLIST ns1:ladung
  xmlns:ns1 CDATA #FIXED 'http://www.schematron.de/arche'>

<!ELEMENT ns1:maxReproduktionsalter (ns1:tier_art)+>
<!ATTLIST ns1:maxReproduktionsalter
  xmlns:ns1 CDATA #FIXED 'http://www.schematron.de/arche'>

<!ELEMENT ns1:nutzlast (#PCDATA)>
<!ATTLIST ns1:nutzlast
  xmlns:ns1 CDATA #FIXED 'http://www.schematron.de/arche'
  einheit NMTOKEN #REQUIRED>

<!ELEMENT ns1:zimmer (ns1:tier)+>
```

XML-Schema

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <!--W3C Schema generated by XMLSpy v2005 rel. 3 U (http://www.altova.com)-->
3  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
4   targetNamespace="http://www.schematron.de/arche" xmlns="http://www.schematron.d
5   elementFormDefault="qualified">
6   <xs:element name="alter">
7    <xs:simpleType>
8     <xs:restriction base="xs:positiveInteger">
9      <xs:maxInclusive value="200"/>
10    </xs:restriction>
11   </xs:simpleType>
12  </xs:element>
13  <xs:element name="arche">
14   <xs:complexType>
15    <xs:sequence>
16     <xs:element ref="ladung"/>
17     <xs:element ref="maxReproduktionsalter"/>
18     <xs:element ref="nutzlast"/>
19    </xs:sequence>
20   </xs:complexType>
21  </xs:element>
22  <xs:element name="art" type="xs:string"/>
23  <xs:element name="gewicht">
```

Relax NG (XML-Schreibweise)

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <grammar ns="http://www.schematron.de/arche" xmlns:xsi="http://www.w3.org/2001/XMLS
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     <attribute name="xsi:schemaLocation"/>
8     <element name="ladung">
9      <oneOrMore>
10      <element name="zimmer">
11       <oneOrMore>
12        <element name="tier">
13         <attribute name="fleischfresser">
14          <data type="NCName"/>
15         </attribute>
16         <attribute name="geschlecht">
17          <data type="NCName"/>
18         </attribute>
19         <element name="art">
20          <data type="NCName"/>
```

Relax NG (Kompakt-Schreibweise)

```
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 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.

Weitere Vorteile von XML-Schema gegenüber DTDs

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

Nachteile 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.

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.

Zusammenfassung

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

Grammatikbasierte Sprachen bieten keine Möglichkeit Abhängigkeiten zu definieren. (Ausnahme ist XML-Schema 1.1)

Ablauf einer XML-Prüfung

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

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

Kohärenzprüfung

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>
```

Bedingung ist abbildbar durch XPATH!

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

XPATH in Schematron einbetten!

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.

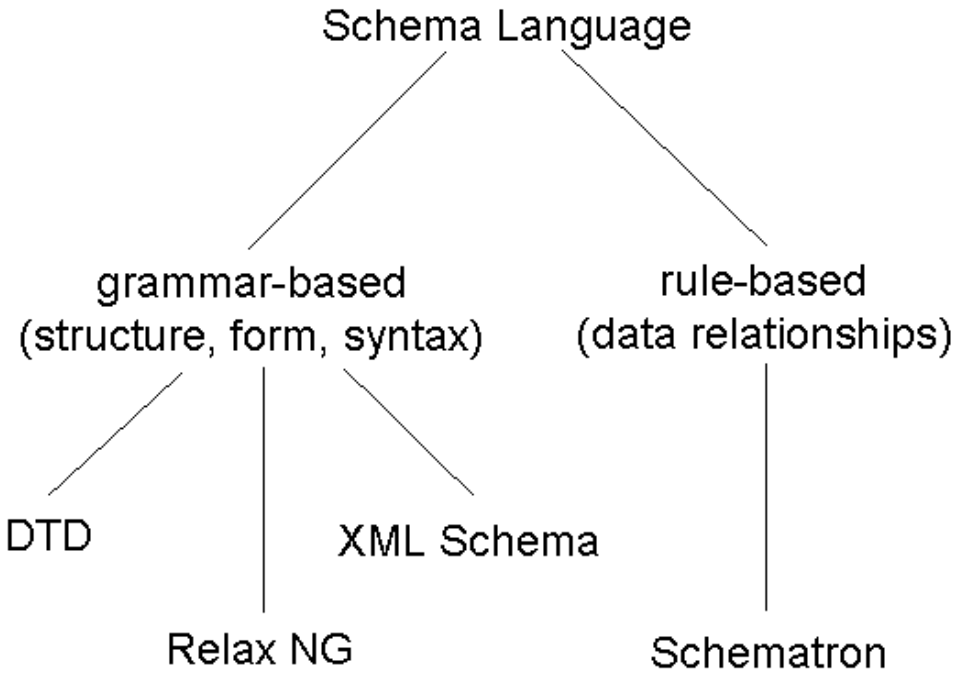
Lösungsansatz ist Schematron

- 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>
```

Schematron: Ein Ersatz für XML Schema?



-> Schematron als Ergänzung zu grammatikbasierten Sprachen

Grafik von Roger Costello
http://www.xfront.com/schematron/Two-types-of-XML-Schema-Language.html

Einsatzszenario

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.

Beispiel für eine Anforderung die beim Sichten aufgefallen ist.

Silbentrennungen

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

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?

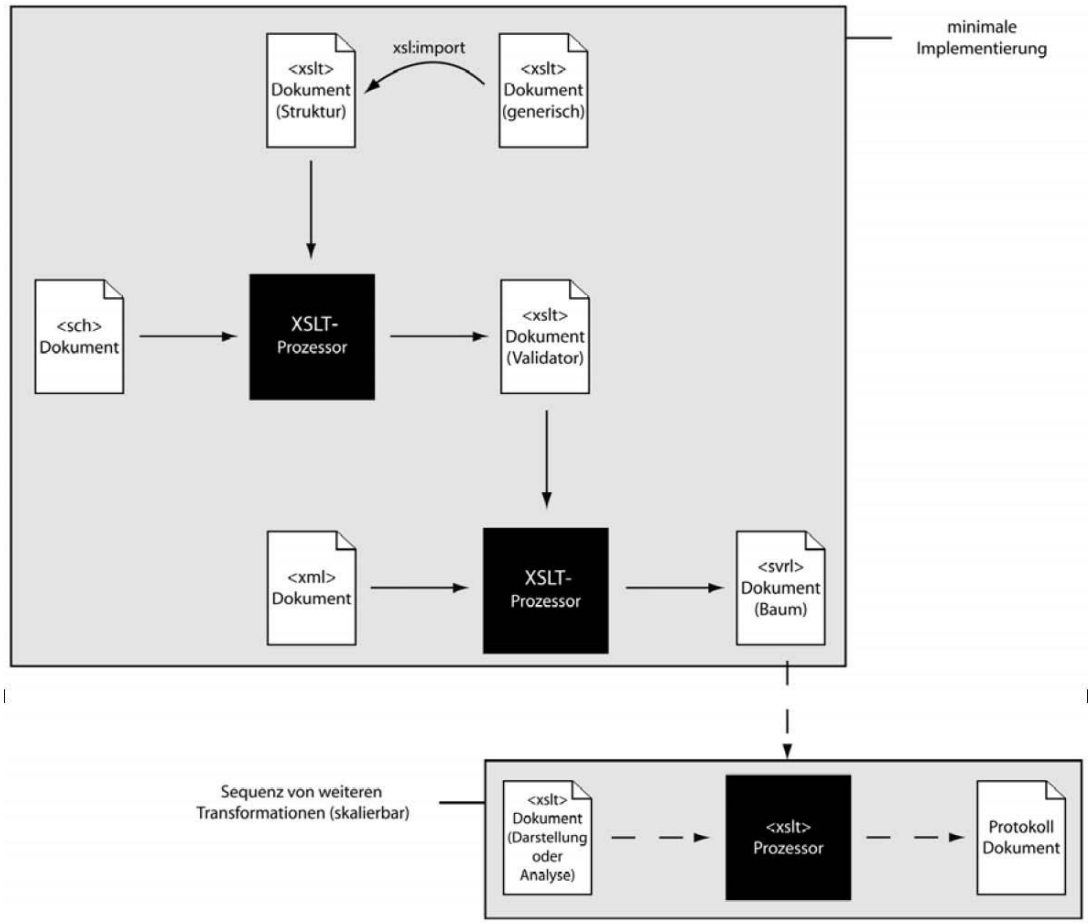
Szenario 1: Im Editor

Editoren die Schematron unterstützen:
Oxygen, XMLMind, ….

Szenario 2: Black-Box (Referenzimplementierung mit XSLT)

Szenario 3: Black-Box eingebettet in XProc

Schematron Einsatzszenario



Beispiele aus dem Publishing!

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>
```

SCH

```
<rule context="footnote">
   <assert test="not(.//footnote)">
        …
   </assert>
<rule>
```

PI´s

SCH

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

Verschachtelte Listen

SCH

```
<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>
```

Verwendung von Zeichen

Beispiel:

```
<para>Bitte tragen Sie hier ihren
        Namen ein: _________</para>


<pattern id="Zeichen" role="warning">
        <rule context="*[text()]">
  <report test="contains(text(),'____')">
                Mehr als drei Unterstriche hintereinander.
   </report>
</rule>
</pattern>
```

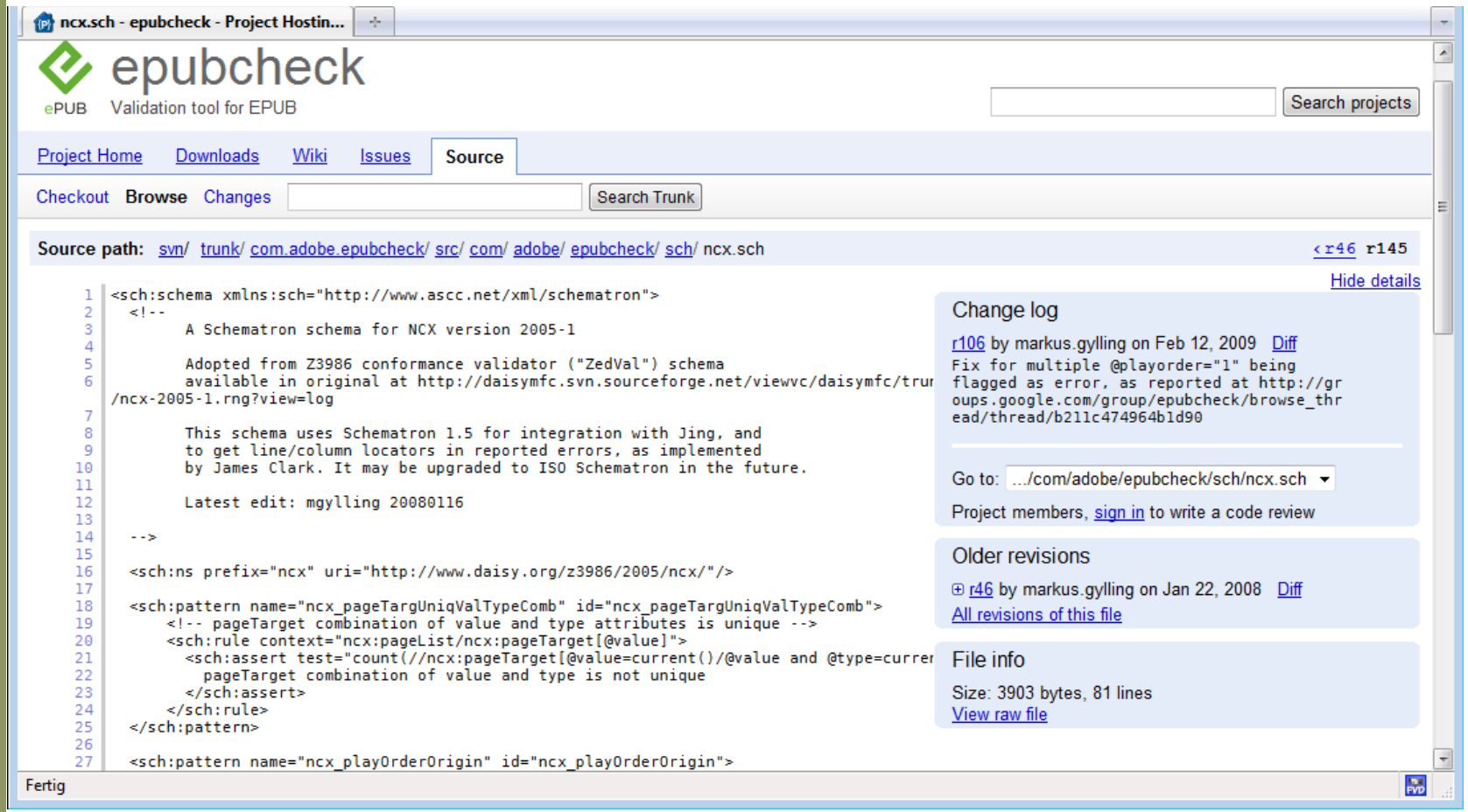
Schematron bei großen Grammatiken:

Wer benutzt Schematron?

- **DocBook**
- **DITA**
- **TEI**
- **EPUB**
- **…?**

Schematron in DocBook

Schematron in Epub-check



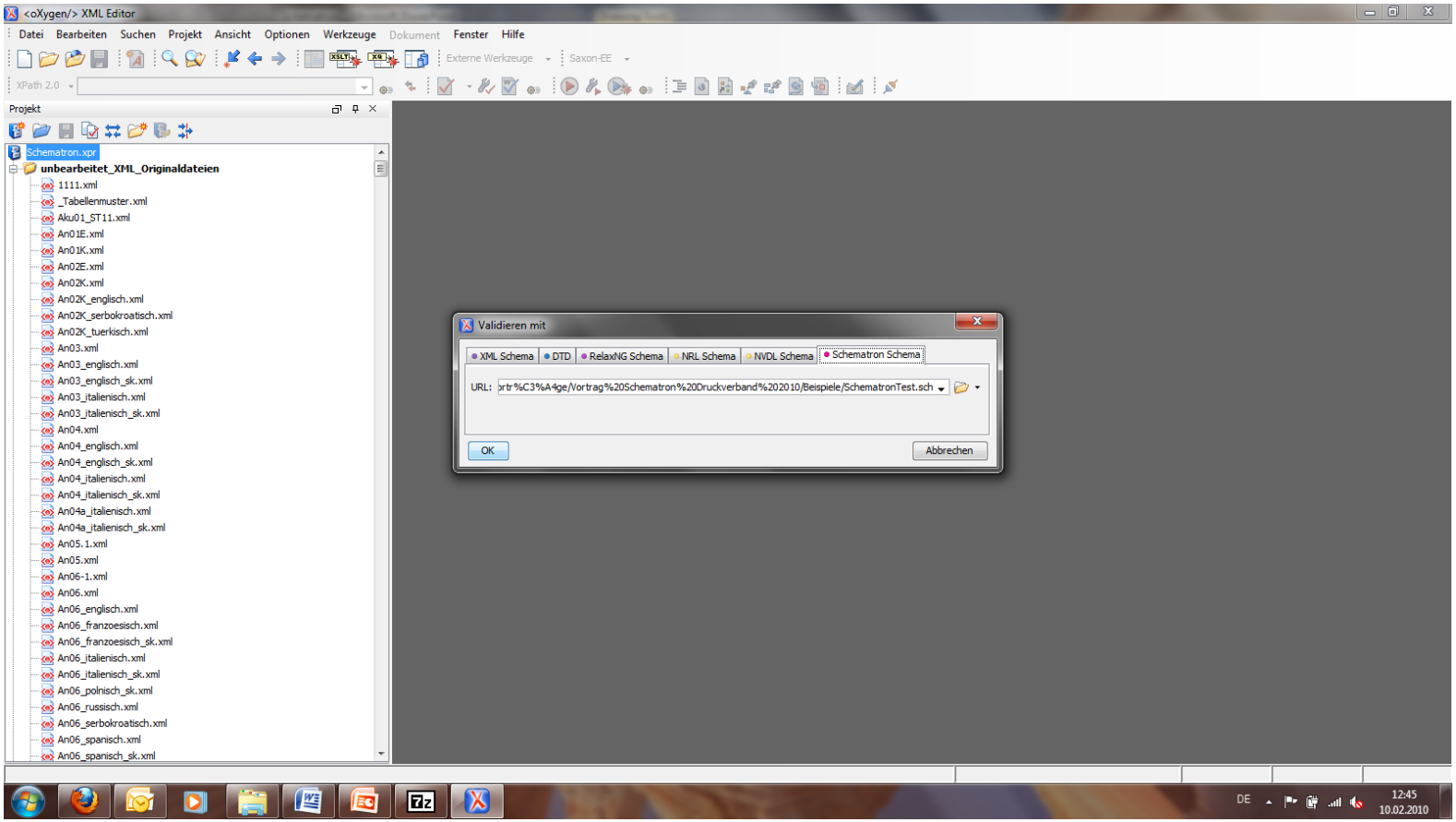
Erweiterungen mit Java (nur mit Saxon)

```
<pattern>
  <rule context="bild">
      <let name="path" value="concat($myDir,'\',@src)"/>
      <let name="file" value="f:new($path)"/>
      <assert test="f:exists($file)">Bild
          <value-of select="tokenize($path,'\\')[last()]"/>
          existiert nicht
      </assert>
    </rule>
</pattern>
```

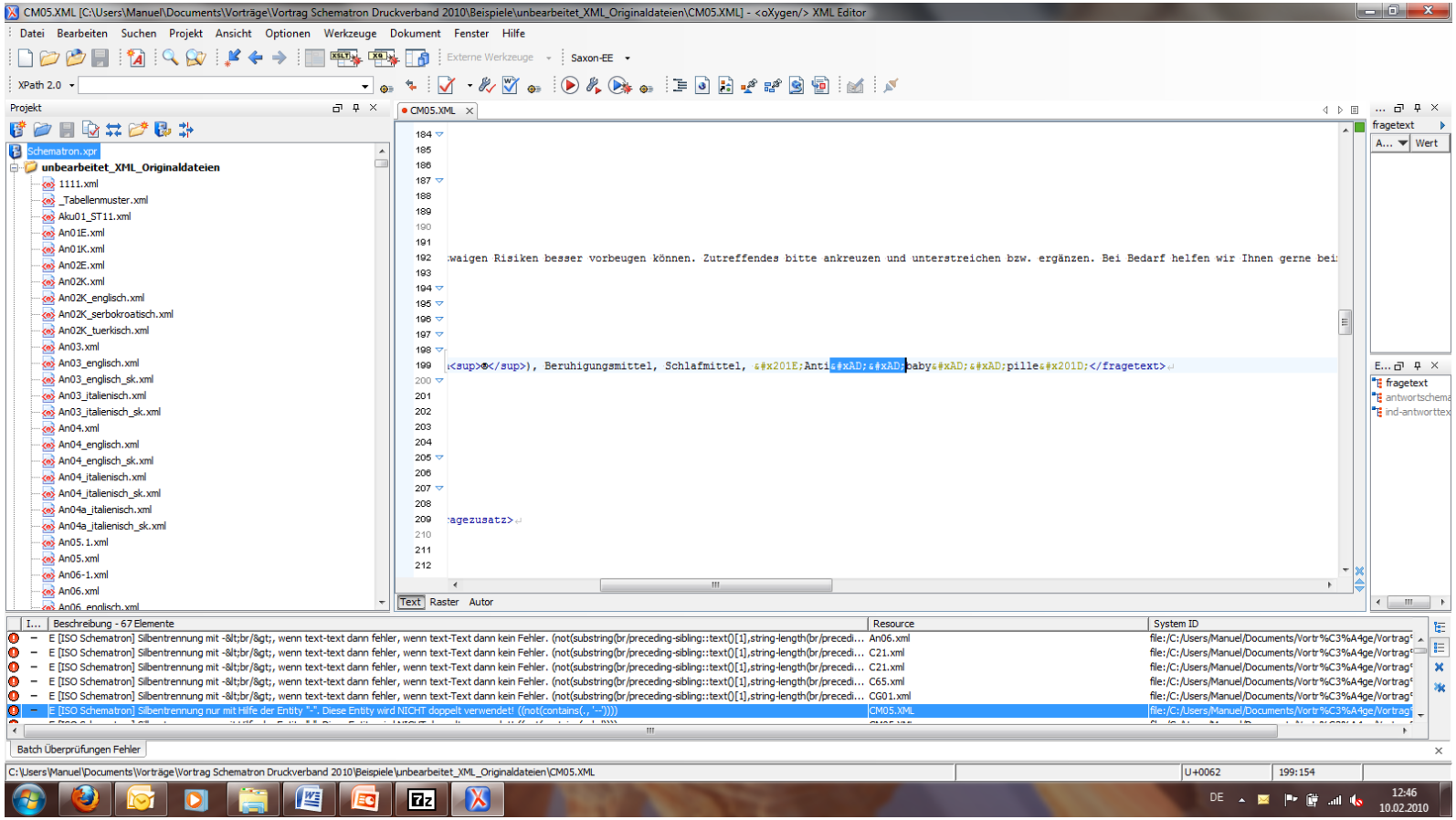
```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <?oxygen SCHSchema="schematron/bild.sch"?>
3
4  <collection>
5      <bild src="pics\bild.jpg"/>
6      <bild src="pics\bild2.jpg"/>
7      <bild src="pics\bild
8  </collection>
9
```

```
E [ISO Schematron (XSLT 2.0)]
Bild bild.jpg existiert nicht
(f:exists($file)) [assert]
```

Schematron validieren im Editor



Schematron validieren im Editor



Literaturhinweis in eigener Sache:

Unser Schematronbuch (Autoren sind Prof. Marko Hedler Hochschule der Medien Stuttgart und ich) erscheint im Sommer 2010 beim dpunkt Verlag.

Artikel zu Schematron:

Montero Pineda, Manuel, Hedler, Marko, Dünckel, Björn: Jedes Spiel hat seine Regeln, in: entwickler magazin, Heft 4 (2009).

http://entwickler-magazin.de/zonen/magazine/psecom,id,8,online,2459.html

Und natürlich die Seite zum Thema:

http://www.schematron.com/

Fragen?

data2type

Dammstr. 18
67059 Ludwigshafen
T: ++49-(0)621-68124792
F: ++49-(0)621-68124794
E: montero@data2type.de