Einführung zu

Schematron

Thomas Klampfl

31.05.2025

Einführung:

Art. *Schematron* `https://de.wikipedia.org/wiki/Schematron` (last visited 30.05.2025).

Schematron ist eine Schemasprache zur Validierung von Inhalt und Struktur von XML-Dokumenten. Schematron wurde 1999 von Rick Jelliffe am *Academia Sinica Computing Centre* in Taipeh, Taiwan entwickelt. Seit Mai 2006 ist Schematron 1.6 als offizieller ISO/IEC-Standard unter der Nummer 19757-3:2006 registriert (genannt ISO Schematron). Zu den möglichen Einsatzgebieten zählen komplexe Regelwerke, die Abhängigkeiten zwischen verschiedenen Teilen des Dokumentbaums ausdrücken oder dynamische Berechnungen erfordern.

Ein einfaches Beispiel:

TEI Dokument:

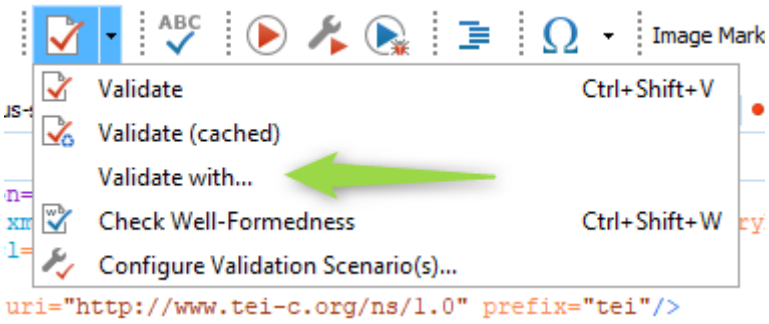
```
<text>
     <body>
         <p xml:lang="en">Some text here.</p>
         <p xml:lang="de">Ein weiterer Text.</p>
         <p xml:lang="eng">English Text.</p>
     </body>
 </text>
```

Schematron Datei zur Überprüfung des `xml:lang` Attributs (erlaubte Werte: `en`, `de`):

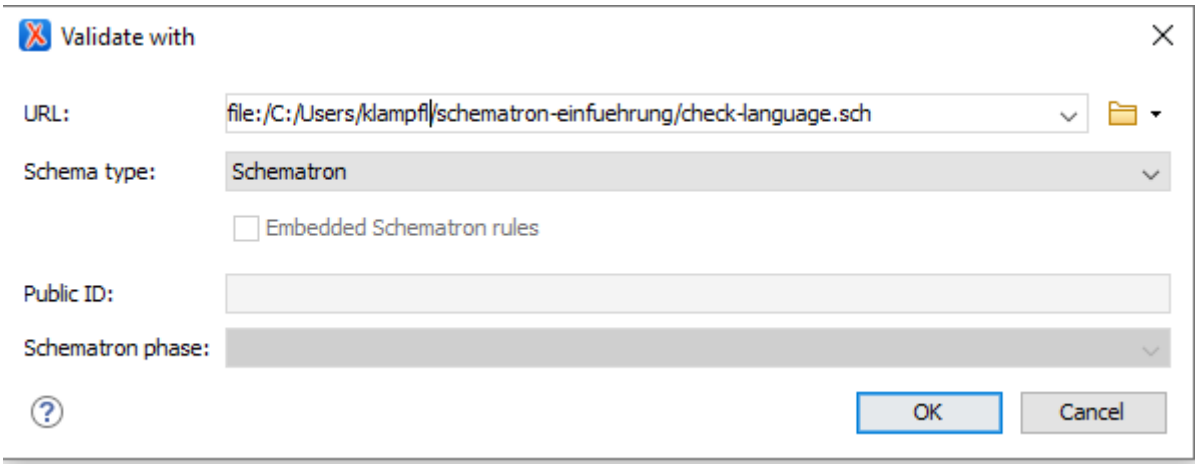
```
<?xml version="1.0" encoding="UTF-8"?>
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    queryBinding="xslt2">

    <sch:ns uri="http://www.tei-c.org/ns/1.0" prefix="tei"/>

    <sch:title>Check language tags</sch:title>
    <sch:pattern>
        <sch:rule context="tei:body/tei:p">
            <sch:assert test="@xml:lang = ('en','de')">
                Found wrong language tag:
                <xsl:value-of select="./@xml:lang"/>.
                Please use on of: 'de' | 'en'.
            </sch:assert>
        </sch:rule>
    </sch:pattern>

</sch:schema>
```

Anwenden der Validierung in *Oxygen*:

Validate – Validate with...



Schematron auswählen und Pfad angeben:



OK anklicken.

Für eine automatische Validierung kann die Schematron Datei auch in die XML Quelldatei eingebunden werden:

```
<?xml-model href="check-language.sch" type="application/xml"
      schematypens="http://purl.oclc.org/dsdl/schematron"?>
```

Erklärung der Validierungsdatei:

- Das Wurzelelement einer Schematron Datei ist immer `schema`.
- Das `queryBinding` Attribut teilt dem Schematronprozessor mit, welche Abfragesprache verwendet werden kann. Mit der Angabe `xslt3` kann XSLT 3.0 und Xpath 3.1 genutzt werden.
- Das `title` Element gibt der folgenden Regel bzw. den folgenden Regeln einen Titel.
- Mit dem `pattern` Element ist die eigentliche Validierungsregel gegeben.
- Das `rule` Element hat ein `context` Attribut, welches angibt, welche Teile des Dokuments geprüft werden sollen. Im Beispiel sind dies alle Paragraphen (`tei:p`) im Körper (`tei:body`) des TEI-Dokuments.
- Das `assert` Element überprüft nun etwas am Kontextknoten; was überprüft wird, ist im `test` Attribut angegeben. Im Beispiel wird überprüft, ob das `xml:lang` Attribut einen Wert aus der Menge: `de`, `en`, hat. Das heißt, der dritte Paragraph wirft einen Fehler. Wenn

der Ausdruck im `test` Attribut zu *false* auswertet, wird der Text innerhalb des `assert` Elements ausgegeben.

Sprachelemente:

`pattern – rule – assert`

Das `pattern` Element stellt den eigentlichen Kern von Schematron dar. Es enthält ein `rule` Element, oder mehrere `rule` Elemente. Bei mehreren `rule` Elementen gilt, dass wenn ein `rule` Element matched, die nachfolgenden `rule` Elemente nicht ausgewertet werden. Das `assert` Element überprüft, ob die Bedingung im `test` Attribut wahr oder falsch ist; wertet sie zu *false* aus, wird der Text innerhalb der `assert` Anweisung ausgegeben. Finden sich mehrere `assert` Elemente, werden alle durchgegangen. Im `test` Attribut muss sich ein XPath Ausdruck finden, der zu einem *boolschen* Ergebnis auswertet. Die Nachricht im `assert` Element, soll so geschrieben werden, dass der Benutzer sie leicht verstehen kann. Der Schematron Standard sagt: „The natural language assertion shall be a positive statement of a constraint. Das Kontextitem ist außerhalb eines `rule` Elements der *document node*, innerhalb eines `rule` Elements der Knoten, der im `context` Attribut selektiert wurde.

`report`

Das `report` Element funktioniert analog dem `assert` Element, mit dem Unterschied, dass der Content ausgegeben wird, wenn der Wert im `test` Attribut zu *true* auswertet.

`let`

Zur Definition einer Variablen wird das Element `let` verwendet.

`diagnostics – diagnostic`

Diese Elemente erlauben die Definition von wiederverwendbaren Nachrichten. Das Element `diagnostic` muss ein `id` Attribut tragen. Im `assert` findet sich keine Nachricht, vielmehr wird mit dem `diagnostics` Attribut eine Nachricht referenziert.

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron"
queryBinding="xslt2"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

    <sch:ns uri="http://www.tei-c.org/ns/1.0" prefix="tei"/>

    <sch:title>Check language tags</sch:title>
    <sch:pattern>
        <sch:rule context="tei:body/tei:p">
            <sch:assert test="@xml:lang = ('en','de')"
                diagnostics="wrong-language-tag supported-values"/>
        </sch:rule>
    </sch:pattern>

    <sch:diagnostics>
        <sch:diagnostic id="wrong-language-tag">
            Found wrong language tag:
```

```
            <xsl:value-of select="./@xml:lang"/>.
          </sch:diagnostic>
          <sch:diagnostic id="supported-values">
              Please use on of 'de' | 'en'.
          </sch:diagnostic>
      </sch:diagnostics>

</sch:schema>
```

Beispiel 2: Überprüfung von Referenzen

TEI Dokument:

```
...
<div n="4.4" type="psalmverse">
   <div xml:lang="grc" type="text">
      <div type="psalmtext">
         <quote xml:id="edition-ps-4-4a" type="psalmtext" n="4a">
            καὶ γνῶτε ὅτι ἐθαυμάστωσεν κύριος τὸν ὅσιον αὐτοῦ·</quote>
      </div>
      <div type="commentary">
         <p xml:id="edition-4-4-1" n="44">
            Ἀντὶ τοῦ γνῶτε τοιγαροῦν οἱ πεποιθότες ἐπὶ
            πλήθει, ὡς θαυμαστὸν ἀπέδειξε τὸν πεποιθότα ἐπ᾽ αὐτῷ: –</p>
         <div type="links">
            <p>
               <listRef>
                  <ref target="#vat-gr-754:vat-gr-754-fr-4-22"/>
                  <ref target="#mosq-syn-194:mosq-syn-194-fr-4-2"/>
                  <ref target="#par-gr-166:par-gr-166-fr-4-5"/>
                  <ref target="#ambr-m-47-sup:ambr-m-47-sup-fr-4-6"/>
                  <ref target="#oxon-s-trin-78:oxon-s-trin-78-fr-4-3"/>
               </listRef>
            </p>
         </div>
      </div>
      <div type="textcritic">
         <app type="fragment">
             ...
         </app>
         <app type="text">
             ...
         </app>
             ...
      </div>
   </div>
   <div xml:lang="de" type="translation">
      <quote corresp="#edition-ps-4-4a" type="psalmtext" n="4a">
         Und erkennt, dass der Herr seinen Frommen wunderbar
         behandelt hat.</quote>
      <p corresp="#edition-4-4-1" n="44">
         Anstelle von 'Wisset also, die ihr auf eine
         große Schar vertraut, dass er Wunderbares an dem gewirkt hat,
         der auf ihn vertraut'.</p>
      <quote corresp="#edition-ps-4-4b" type="psalmtext" n="4b">
         Der Herr wird mich anhören, während ich unaufhörlich zu ihm
         schreie.</quote>
```

```
      <p corresp="#edition-4-4-2-1" n="45a">
         Anstelle von 'er hörte mich an'.</p>
      <p corresp="#edition-4-4-2-2" n="45b">
         Er hat eine Zeitform anstelle einer anderen
         genommen. Denn anstelle von 'er hat angehört' sagte er das Verb
         'er wird anhören'.</p>
   </div>
</div>
...
```

Die Struktur dieses Abschnitts ist folgende:

```
div @type = psalmverse
     div @type = text
          div @type = commentary
               p @xml:id
     div @type = translation
          p @corresp
```

Validierungsdatei:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://purl.oclc.org/dsdl/schematron" queryBinding="xslt2"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

    <ns uri="http://www.tei-c.org/ns/1.0" prefix="tei"/>

    <title>Check the references to manuscript @ids</title>
    <pattern>
        <rule context="tei:ref">
            <assert test="starts-with(@target,'#')">
                Missing '#' in ref/@target.
                @target must start with a '#' sign.</assert>
            <assert test="starts-with(@target,'#vat-gr-754:')
                or starts-with(@target,'#vat-gr-1422:')
                or starts-with(@target,'#coislin-10:')
                or starts-with(@target,'#coislin-12:')
                or starts-with(@target,'#coislin-187:')
                or starts-with(@target,'#ambr-b-106-sup:')
                or starts-with(@target,'#ambr-m-47-sup:')
                or starts-with(@target,'#bodl-auct-d-4-1:')
                or starts-with(@target,'#oxon-s-trin-78:')
                or starts-with(@target,'#par-gr-139:')
                or starts-with(@target,'#par-gr-164')
                or starts-with(@target,'#par-gr-166:')
                or starts-with(@target,'#plut-6-3:')
                or starts-with(@target,'#plut-5-30')
                or starts-with(@target,'#franzon-3:')
                or starts-with(@target,'#mosq-syn-194:')">
                Wrong prefix in ref/@target.
                @target must start with one of: #vat-gr-754:',
                '#vat-gr-1422:','#coislin-10:','#coislin-12:',
                '#coislin-187:','#ambr-b-106-sup:',
                '#ambr-m-47-sup:','#bodl-auct-d-4-1:',
                '#oxon-s-trin-78:','#par-gr-139:','#par-gr-164',
                '#par-gr-166:','#mosq-syn-194:','#plut-6-3:',
                '#plut-5-30:','#franzon-3:'</assert>
```

```
        </rule>
    </pattern>

    <title>
       Check the value of the @corresp attribute on the
       translation</title>
    <pattern>
        <rule
           context="tei:div[(@type = 'translation')
                    and (@xml:lang = 'de')]/tei:p">
            <let name="reference" value="@corresp"/>
            <assert test="starts-with($reference,'#')">
               Missing '#' in paragraph of translation.
               @corresp must start with a '#' sign.</assert>
            <assert
               test="substring-after($reference,'#') =
                     parent::tei:div[@type = 'translation']/
                     parent::tei:div[@type = 'psalmverse']/
                     child::tei:div[@type = 'text']/
                     child::tei:div[@type = 'commentary']/
                     child::tei:p/@xml:id">
                Value of @corresp attribute does not match an @id.
                @corresp must have the value of the @id of the paragraph
                with greek text the text is a translation of
            </assert>
        </rule>
    </pattern>
</schema>
```

Es wird hier nur das letzte `assert` genauer besprochen: Im ersten Schritt wird die `@id` aus dem `corresp` Attribut extrahiert (*substring-after()*), und dann verglichen mit dem `@id` Attribut auf dem entsprechenden Paragraphen. Dazu wird die Hierarchie der `divs` hinauf- und wieder heruntergegangen. Alternativ wäre es möglich mit `root()` oder `'//'` zu arbeiten. Zu beachten ist, dass es sich um einen Vergleich mit einer Menge handelt: Es wird nicht auf Gleichheit überprüft, sondern ob ein String in einer Menge von Strings enthalten ist. Deshalb muss die Menge nach dem '=' stehen. Es wäre nicht möglich mit `eq` (Stringvergleich) zu arbeiten.

Weitere Sprachelemente:

`phase – active pattern`

Phasen in Schematron können genutzt werden, um Pattern eines Schemas zu aktivieren bzw. zu deaktivieren.

```
phase @id = check-for-hash
      active pattern = check-for-hash-references
      active pattern = check-for-hash-corresp

phase @id = check-for-id
      active pattern = check-for-id-corresp

pattern @id = check-for-hash-references
pattern @id = check-for-hash-corresp
pattern @id = check-for-id-corresp
```

Anwendungsszenarien sind etwa, dass manche Checks rechenintensiv sind, oder das komplette Dokument voraussetzen, und nicht nur ein work in progress Dokument.

Abstrakte `rule` Elemente:

Es ist möglich `rule` Elemente als abstrakt zu deklarieren, und dann in eine andere `rule` einzubinden:

```
<rule abstract="true" id="check-for-hashes">
     <assert test="starts-with(@corresp,'#')">...</assert>
</rule>

<pattern>
     <rule context="tei:p">
          <extends rule="check-for-hashes"/>
          <assert test="...">...</assert>
     </rule>
</pattern>
```

Abstrakte `pattern` Elemente:

```
<pattern abstract="true" id="check-for-hash-pattern">
   <rule context="$attribute-with-hash">
      <assert test="starts-with(.,'#')">
         The attribute
         <value-of select="local-name(.)"/>
         needs a hash sign ('#').</assert>
   </rule>
</pattern>

<pattern is-a="check-for-hash-pattern">
   <param name="attribute-with-hash" value="tei:p/@corresp"/>
</pattern>

<pattern is-a="check-for-hash-pattern">
   <param name="attribute-with-hash" value="tei:ref/@target"/>
</pattern>
```

Einbinden von Regeln mittels `include`.

Benachbarte Technologien:

`SVRL` – *Schematron Validation Reporting Language*. Ist ein XML basiertes Format, dass die Ergebnisse einer Schematron Validierung berichtet.

`SQF` – *Schematron Quick Fix*: Ermöglicht Lösungsvorschläge für Fehler in XML Daten anzubieten, aus denen der Benutzer auswählen kann.

Siegel, Eric, *Schematron. A Language for Validating XML*. Denver, CO: XML Press, 2022.