SCHEMATRON IN DER QUALITÄTSSICHERUNG

Workshop »Digitale Editionen – Methoden und Technologien für Fortgeschrittene«

10. Oktober 2012

Stefan Krause
Editura GmbH & Co. KG, Berlin
http://www.editura.de

Gliederung

- Allgemeines
- Schematron für Anfänger
- Schematron für Fortgeschrittene
- Praxisbeispiele
- Schematron für Profis
- Literatur
- Fragen? Projekte: Übungen, Lösungen

ALLGEMEINES

typischer Projektablauf

- ggfs. Scannen
- Vorlagen sichten, Erfassungsanweisungen erstellen
- Datenerfassung (bei Editura typischerweise in Rumänien, Indien, China u. a.)
- Projektbetreuung, Verfeinerung der Erfassungsanweisungen
- Kontrolle, Korrektur, Redaktion
- Auslieferung

typischer Projektablauf

- ggfs. Scannen
- Vorlagen sichten, Erfassungsanweisungen erstellen
- Datenerfassung (bei Editura typischerweise in Rumänien, Indien, China u. a.)
- Projektbetreuung, Verfeinerung der Erfassungsanweisungen
- Kontrolle, Korrektur, Redaktion
- Auslieferung

Kontrolle, Korrektur, Redaktion

- Vollständigkeit
- Fehlerfreiheit der Texterfassung und des Tagging
- Einhaltung der Erfassungsanweisung
- Korrekturen, Ergänzungen, Datenanreicherung, Verknüpfungen

inhaltliche vs. technische Prüfung

- inhaltliche Probleme können nur durch intellektuelle Beurteilung gefunden werden
- technische Probleme können mit technischen Hilfsmitteln geprüft werden, z.B.
  - XML-Schema, DTD, RNG
  - Schematron
  - Prüfskripte

inhaltliche vs. technische Prüfung

- inhaltliche Probleme können nur durch intellektuelle Beurteilung gefunden werden
- technische Probleme können mit technischen Hilfsmitteln geprüft werden, z.B.
  - XML-Schema, DTD, RNG
  - Schematron
  - Prüfskripte

Fehler vs. Warnung

- Fehler verhindern die weitere Verarbeitung der Daten (Show-Stopper). Sie sind oft technisch bedingt.
- Warnungen können auf fehlende Informationen, unerwünschte Eigenschaften, hässliche Darstellung u.v.m. hinweisen, sind aber (irgendwie) tolerabel.
- Dieser Unterschied sollte immer im Auge behalten werden. Alle Projektbeteiligten müssen diese Unterscheidung treffen können, was v. a. ein kommunikatives Problem ist.

Wie entstehen Prüfwerkzeuge?

- systematische Analyse der Erfassungsanweisung und Festlegung, mit welchem Werkzeug technische Prüfungen durchgeführt werden
  - Schema (XSD, RNG)
  - Schematron
  - spezielle Skripte und Prüfsoftware
- laufende Ergänzung während der Projektlaufzeit

SCHEMATRON FÜR ANFÄNGER

Warum Schematron I?

- Manche Prüfungen lassen sich mit den anderen Schemaarten nicht oder nur schwer durchführen, z. B.
  - Kind-Elemente in Abhängigkeit von Attributen
  - fehlerhafte Textinhalte: »z.B.« vs. »z. B.«
  - Berechnungen

Warum Schematron II?

- Es können aussagekräftige Fehlermeldungen formuliert werden.
- Es sind punktuelle Prüfungen möglich, d. h. es wird kein vollständiges Schema benötigt.
- Schematron ist sehr gut in OxygenXML integriert.
- (Trennung von Fehlern und Warnungen durch Aufteilung auf verschiedene Schemata.)

Was ist Schematron?

- eine Schema-Sprache für XML-Dokumente
- ISO-Standard
- basiert auf XPath, mit dessen ganzer Mächtigkeit
- u. a. in XSLT implementiert und deshalb plattformunabhängig

Hello World!

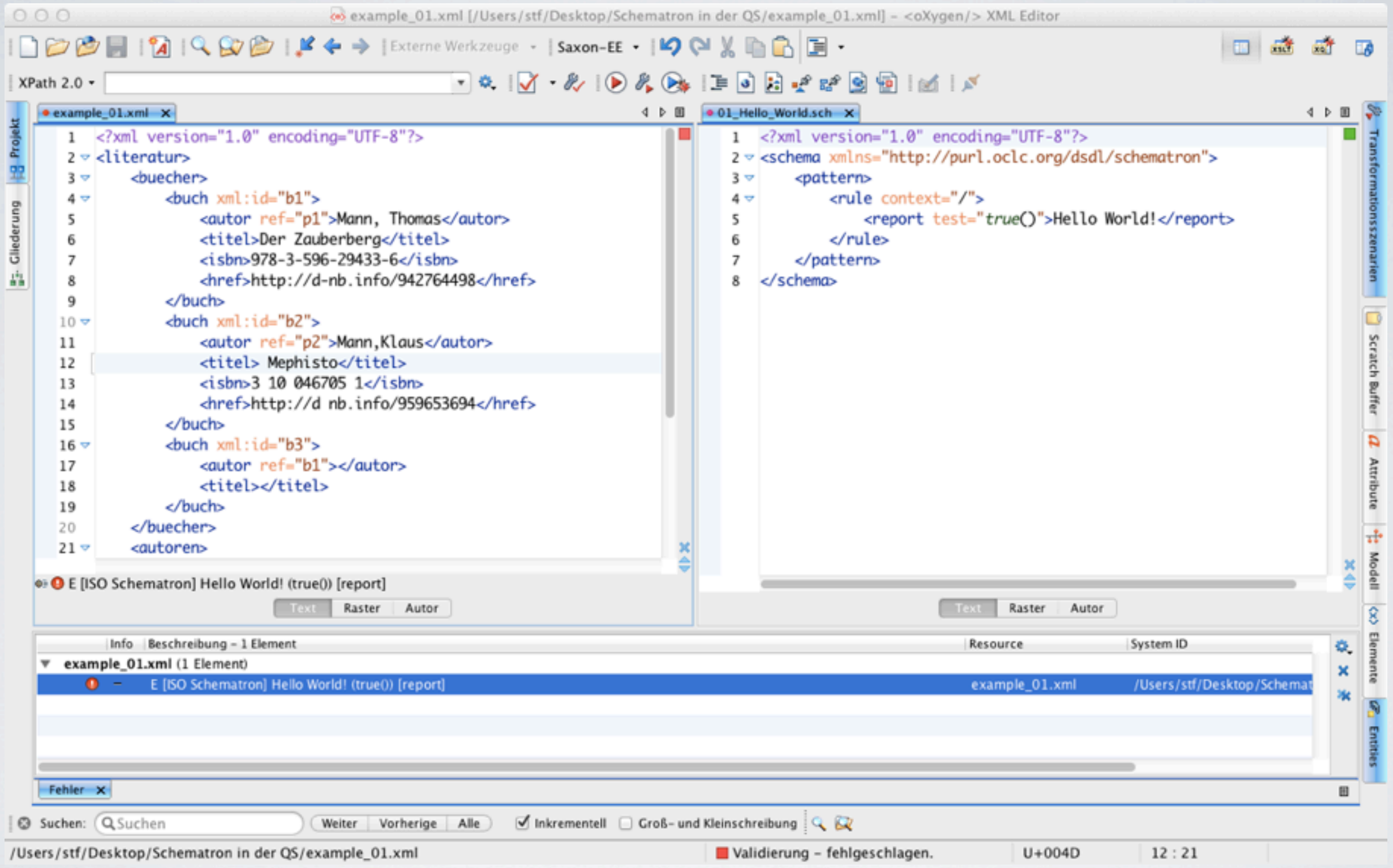
01_Hello_World.sch

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
    <pattern>
        <rule context="/">
            <report test="true()">Hello World!</report>
        </rule>
    </pattern>
</schema>
```

example_01.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<literatur>
    <buecher>
        <buch xml:id="b1">
            <autor ref="p1">Mann, Thomas</autor>
            <titel>Der Zauberberg</titel>
            <isbn>978-3-596-29433-6</isbn>
            <href>http://d-nb.info/942764498</href>
        </buch>
        <buch xml:id="b2">
            <autor ref="p2">Mann,Klaus</autor>
            <titel>Mephisto</titel>
            <isbn>3 10 046705 1</isbn>
            <href>http://d nb.info/959653694</href>
        </buch>
        <buch xml:id="b3">
            <autor ref="b1"></autor>
            <titel></titel>
        </buch>
    </buecher>
    <autoren>
        <person xml:id="p1">
            <vorname>Thomas</vorname>
            <nachname>Mann</nachname>
        </person>
        <person xml:id="p2">
```

Hello World!


example_01.xml [/Users/stf/Desktop/Schematron in der QS/example_01.xml] - <oXygen/> XML Editor
Externe Werkzeuge
Saxon-EE
XPath 2.0
example_01.xml
01_Hello_World.sch
<?xml version="1.0" encoding="UTF-8"?>
<literatur>
<buecher>
<buch xml:id="b1">
<autor ref="p1">Mann, Thomas</autor>
<titel>Der Zauberberg</titel>
<isbn>978-3-596-29433-6</isbn>
<href>http://d-nb.info/942764498</href>
</buch>
<buch xml:id="b2">
<autor ref="p2">Mann,Klaus</autor>
<titel> Mephisto</titel>
<isbn>3 10 046705 1</isbn>
<href>http://d nb.info/959653694</href>
</buch>
<buch xml:id="b3">
<autor ref="b1"></autor>
<titel></titel>
</buch>
</buecher>
<autoren>
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
<pattern>
<rule context="/">
<report test="true()">Hello World!</report>
</rule>
</pattern>
</schema>
E [ISO Schematron] Hello World! (true()) [report]
Text
Raster
Autor
Info
Beschreibung - 1 Element
Resource
System ID
example_01.xml (1 Element)
E [ISO Schematron] Hello World! (true()) [report]
example_01.xml
/Users/stf/Desktop/Schemat
Fehler
Suchen:
Suchen
Weiter
Vorherige
Alle
Inkrementell
Groß- und Kleinschreibung
/Users/stf/Desktop/Schematron in der QS/example_01.xml
Validierung - fehlgeschlagen.
U+004D
12 : 21
Projekt
Gliederung
Transformationsszenarien
Scratch Buffer
Attribute
Modell
Elemente
Entities

Prüfung: `report`

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
    <pattern>
        <rule context="/">
            <report test="true()">Hello World!</report>
        </rule>
    </pattern>
</schema>
```

- gibt eine Meldung aus,
- wenn die Bedingung in `@test` erfüllt ist

- Inhalt von `@test` ist ein XPath-Ausdruck.
- Die Meldung kann frei formuliert werden.

Kontext: `rule`

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
    <pattern>
        <rule context="/">
            <report test="true()">Hello World!</report>
        </rule>
    </pattern>
</schema>
```

- bestimmt in `@context` den Kontext für die enthaltenen Elemente
- `@context` ist üblicherweise ein Element, oder Attribut, z.B. `context="buch"`
- vergleichbar zu `xsl:template match="..."`

Gruppieren: `pattern`

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
   <pattern>
      <rule context="/">
         <report test="true()">Hello World!</report>
      </rule>
   </pattern>
</schema>
```

- gruppiert mehrere `rule`-Elemente

Hello World!

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
   <pattern>
      <rule context="/">
         <report test="true()">Hello World!</report>
      </rule>
   </pattern>
</schema>
```

erste »echte« Prüfung

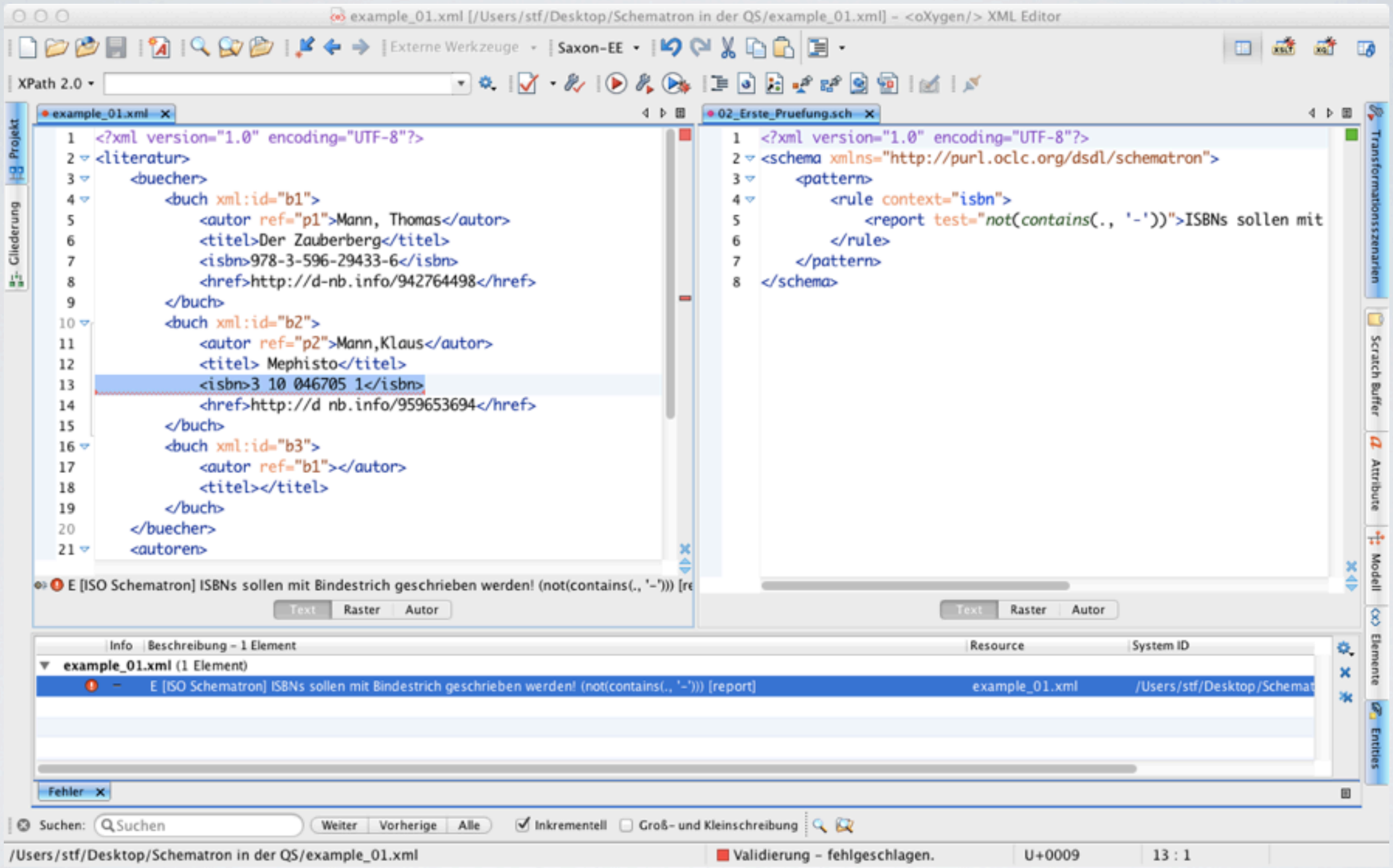
02_Erste_Prüfung.sch

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
    <pattern>
        <rule context="isbn">
            <report test="not(contains(., '-'))">ISBNs
sollen mit Bindestrich geschrieben werden!</report>
        </rule>
    </pattern>
</schema>
```

example_01.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<literatur>
    <buecher>
        <buch xml:id="b1">
            <autor ref="p1">Mann, Thomas</autor>
            <titel>Der Zauberberg</titel>
            <isbn>978-3-596-29433-7</isbn>
            <href>http://d-nb.info/942764498</href>
        </buch>
        <buch xml:id="b2">
            <autor ref="p2">Mann,Klaus</autor>
            <titel>Mephisto</titel>
            <isbn>3 10 046705 1</isbn>
            <href>http://d nb.info/959653694</href>
        </buch>
        <buch xml:id="b3">
            <autor ref="b1"></autor>
            <titel></titel>
        </buch>
    </buecher>
    <autoren>
        <person xml:id="p1">
            <vorname>Thomas</vorname>
            <nachname>Mann</nachname>
        </person>
        <person xml:id="p2">
```

erste »echte« Prüfung



auch Prüfung: assert

02_Erste_Prüfung.sch

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
    <pattern>
        <rule context="isbn">
            <report test="not(contains(., '-'))">ISBNs
sollen mit Bindestrich geschrieben werden!</report>
        </rule>
    </pattern>
</schema>
```

02_Erste_Prüfung_mit_assert.sch

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
    <pattern>
        <rule context="isbn">
            <assert test="contains(., '-')">ISBNs sollen mit
Bindestrich geschrieben werden!</assert>
        </rule>
    </pattern>
</schema>
```

auch Prüfung: `assert`

02_Erste_Prüfung.sch

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
    <pattern>
        <rule context="isbn">
            <report test="not(contains(., '-'))">ISBNs
sollen mit Bindestrich geschrieben werden!</report>
        </rule>
    </pattern>
</schema>
```

02_Erste_Prüfung_mit_assert.sch

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
    <pattern>
        <rule context="isbn">
            <assert test="contains(., '-')">ISBNs sollen mit
Bindestrich geschrieben werden!</assert>
        </rule>
    </pattern>
</schema>
```

- `not(...)` ist umständlich
- deshalb kann an Stelle von `report` auch `assert` verwendet werden
- gibt eine Meldung aus, wenn die Bedingung in `@test` **nicht** erfüllt ist

mehrere Tests

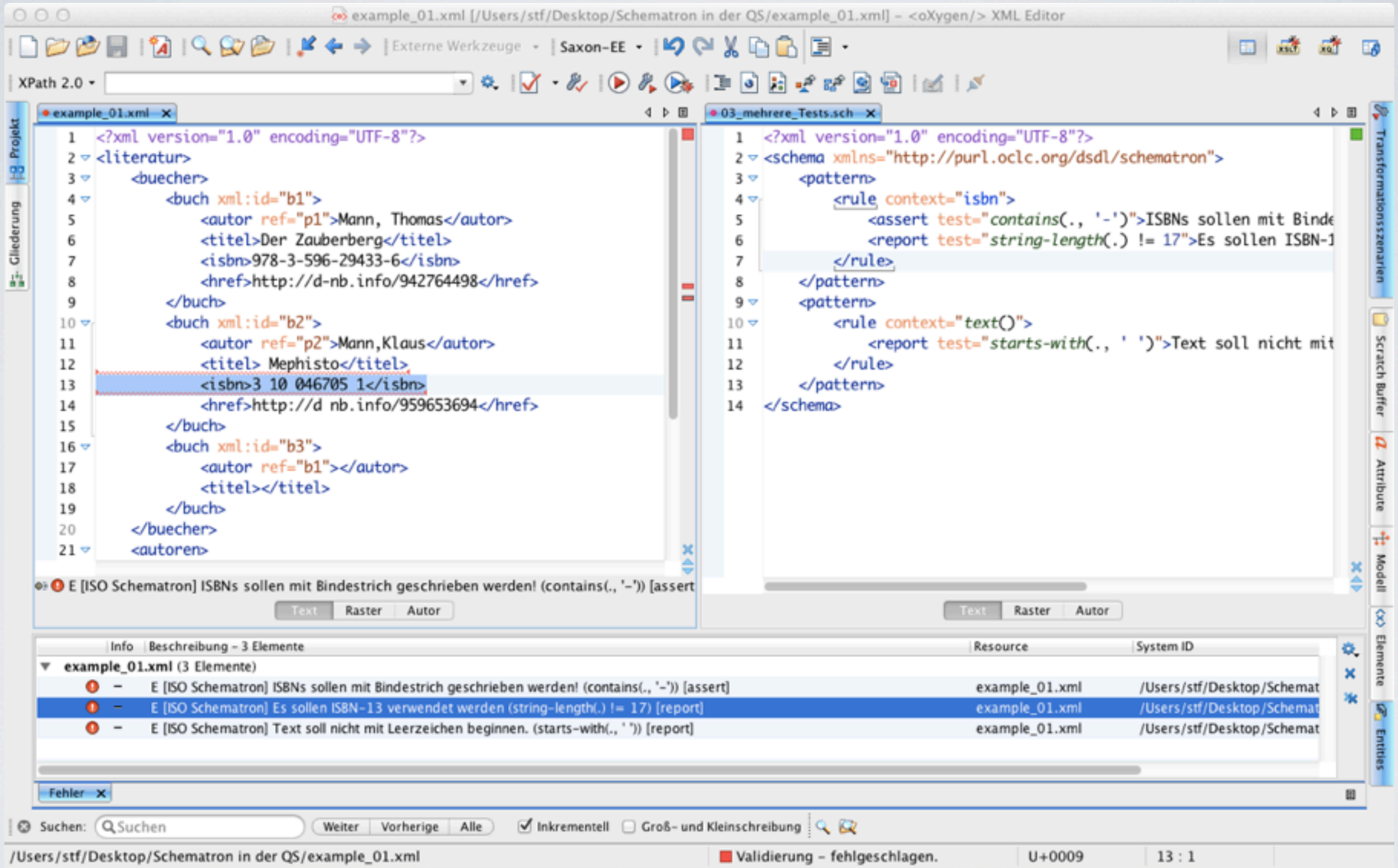
02_mehrere_Tests.sch

```
<?xml version="1.0" encoding=
<schema xmlns="http://purl.oclc.org/dsdl/schematron">

    <pattern>
        <rule context="isbn">
            <assert test="contains(., '-')">
                ISBNs sollen mit Bindestrich
                geschrieben werden!</assert>
            <report test="string-length(.) != 17">
                Es sollen ISBN-13 verwendet
                werden</report>
        </rule>
    </pattern>

    <pattern>
        <rule context="text()">
            <report test="starts-with(., ' ')">
                Text soll nicht mit Leerzeichen
                beginnen.</report>
        </rule>
    </pattern>

</schema>
```

example_01.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<literatur>
    <buecher>
        <buch xml:id="b1">
            <autor ref="p1">Mann, Thomas</autor>
            <titel>Der Zauberberg</titel>
            <isbn>978-3-596-29433-7</isbn>
            <href>http://d-nb.info/942764498</href>
        </buch>
        <buch xml:id="b2">
            <autor ref="p2">Mann,Klaus</autor>
            <titel>Mephisto</titel>
            <isbn>3 10 046705 1</isbn>
            <href>http://d nb.info/959653694</href>
        </buch>
        <buch xml:id="b3">
            <autor ref="b1"></autor>
            <titel></titel>
        </buch>
    </buecher>
    <autoren>
        <person xml:id="p1">
            <vorname>Thomas</vorname>
            <nachname>Mann</nachname>
        </person>
        <person xml:id="p2">
```

mehrere Tests



Namespaces: ns

04_namespace.sch

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
    <ns prefix="tei" uri="http://www.tei-c.org/ns/1.0"/>
    <pattern>
        <rule context="
/tei:TEI/tei:teiHeader/tei:fileDesc/tei:titleStmt">
            <assert test="tei:title[@type = 'main']">
                Es muss ein title-Element mit
                @type="main" geben</assert>
        </rule>
    </pattern>
    <pattern>
        <rule context="
/tei:TEI/tei:teiHeader/tei:fileDesc/tei:titleStmt">
            <assert test="tei:title[@type = 'sub']">
                Es muss ein title-Element mit
                @type="sub" geben</assert>
        </rule>
    </pattern>
</schema>
```

example_02_Dingler_Bd-333.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<tei:TEI xmlns:tei="http://www.tei-c.org/ns/1.0"
xml:lang="de-DE">
    <tei:teiHeader>
        <tei:fileDesc>
            <tei:titleStmt>
                <tei:title type="_main">Dingler-Online I
Das digitalisierte Polytechnische Journal</tei:title>
                <tei:title type="_sub">Band 333,
Jahrgang 1918</tei:title>
            </tei:titleStmt>
            <tei:publicationStmt>
                <tei:publisher>Humboldt-Universität
                </tei:publisher>
                <tei:pubPlace>Berlin</tei:pubPlace>
                <tei:date when="2012"/>
            </tei:publicationStmt>
            <tei:seriesStmt>
                <tei:title level="j">Dingler-Online I Das
digitalisierte Polytechnische Journal</tei:title>
                <tei:respStmt>
                    <tei:resp>Projektträger:</tei:resp>
                    <tei:name xml:id="HU">Humboldt-
```

Namespaces: ns

04_namespace.sch

example_02_Dingler_Bd-333.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
    <ns prefix="tei" uri="http://www.tei-c.org/ns/1.0"/>
    <pattern>
        <rule context="
    /tei:TEI/tei:teiHeader/tei:fileDesc/tei:titleStmt">
            <assert test="tei:title[@type = 'main']">
                Es muss ein title-Element mit
                @type="main" geben</assert>
        </rule>
    </pattern>
    <pattern>
        <rule context="
    /tei:TEI/tei:teiHeader/tei:fileDesc/tei:titleStmt">
            <assert test="tei:title[@type = 'sub']">
                Es muss ein title-Element mit
                @type="sub" geben</assert>
        </rule>
    </pattern>
</schema>
```

Namespaces: ns

04_namespace.sch

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://purl.oclc.org/dsdl/schematron">

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

    <pattern>
        <rule context="
/tei:TEI/tei:teiHeader/tei:fileDesc/tei:titleStmt">
            <assert test="tei:title[@type = 'main']">
                Es muss ein title-Element mit
                @type="main" geben</assert>
        </rule>
    </pattern>
    <pattern>
        <rule context="
/tei:TEI/tei:teiHeader/tei:fileDesc/tei:titleStmt">
            <assert test="tei:title[@type = 'sub']">
                Es muss ein title-Element mit
                @type="sub" geben</assert>
        </rule>
    </pattern>
</schema>
```

example_02_Dingler_Bd-333.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<TEI xmlns="http://www.tei-c.org/ns/1.0" xml:lang="de-
DE">
    <teiHeader>
        <fileDesc>
            <titleStmt>
                <title type="_main">Dingler-Online I Das
digitalisierte Polytechnische Journal</title>
                <title type="_sub">Band 333, Jahrgang
1918</title>
            </titleStmt>
            <publicationStmt>
                <publisher>Humboldt-Universität
                </publisher>
                <pubPlace>Berlin</pubPlace>
                <date when="2012"/>
            </publicationStmt>
            <seriesStmt>
                <title level="j">Dingler-Online I Das
digitalisierte Polytechnische Journal</title>
                <respStmt>
                    <resp>Projektträger:</resp>
                    <name xml:id="HU">Humboldt-
```

Zwischenstand

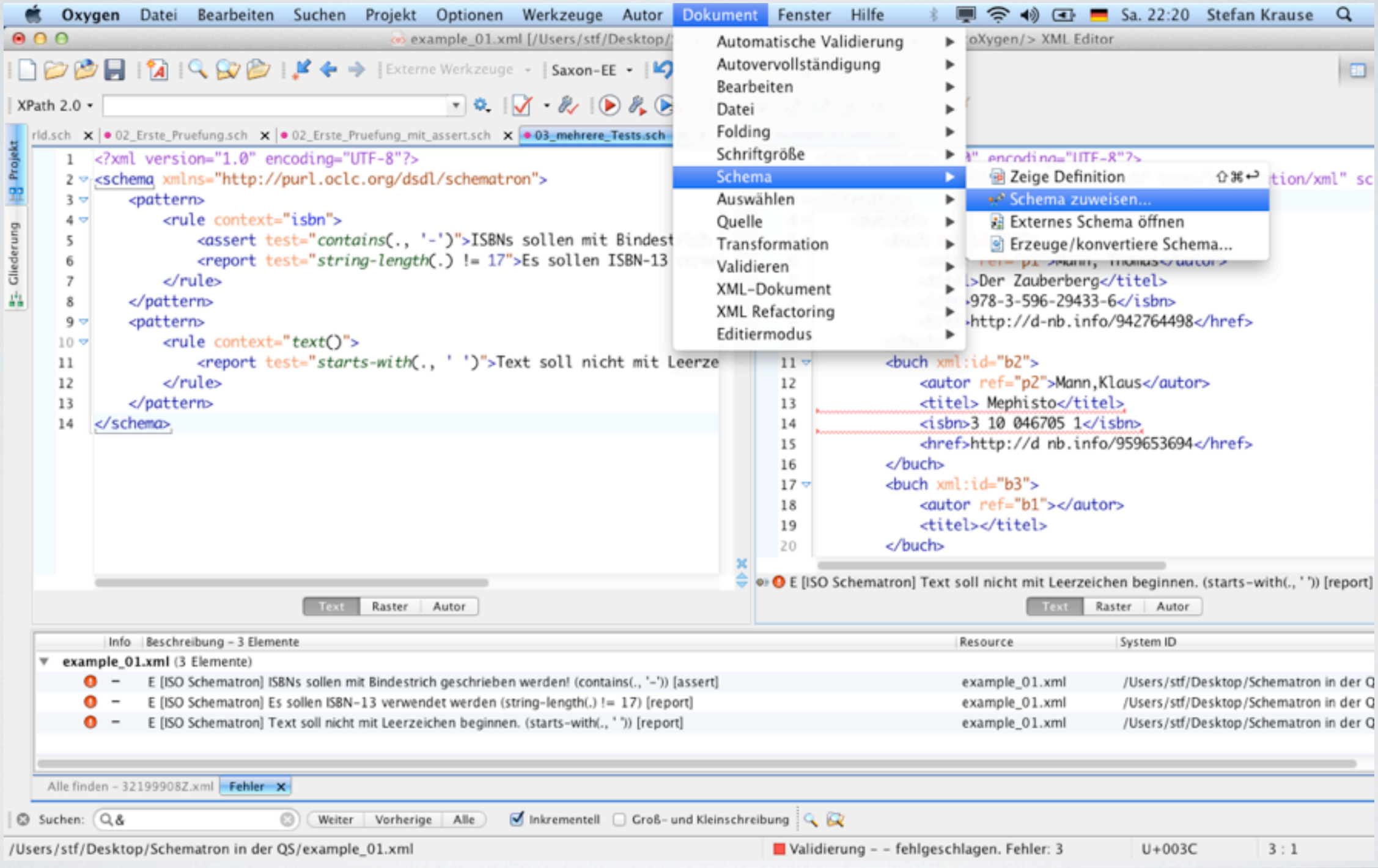
- Sechs Elemente – `schema`, `pattern`, `rule`, `assert`/`report`, `ns` – genügen, um Schematron-Prüfungen zu schreiben.
- Die Mächtigkeit entsteht durch XPath.
- So schwer ist das doch gar nicht!

Tipp: Schema einbinden

- zum einfachen Validieren in OxygenXML kann das Schema in die XML-Datei eingebunden werden:
  - Dokument ⇒ Schema ⇒ Schema zuweisen…
  - Schema auswählen & bestätigen

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="03_mehrere_Tests.sch" type="application/xml"
schematypens="http://purl.oclc.org/dsdl/schematron"?>
<literatur>
    <buecher>
        <buch xml:id="b1">
            <autor ref="p1">Mann, Thomas</autor>
```

Tipp: Schema einbinden



Tipp: ein `pattern`, ein `rule`

- Wenn man nicht ganz genau weiß, was man macht (Und wer weiß das schon?), gilt:
  - Je `pattern`-Element eine `rule`-Element!
  - Mehrere `assert`/`report` in einer `rule` sind OK.

Tipp: Fehler-IDs vergeben

- Es ist gute Praxis, Meldungen mit einer ID zu versehen.
  - Prüfungs-Listen lassen sich leichter überblicken.
  - Meldungen sind eindeutig.
  - Die Quelle einer Meldung im Schematron ist eindeutig.
  - Externe Fehlerbeschreibungen sind eindeutig.

Tipp: Fehler-IDs vergeben

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://purl.oclc.org/dsdl/schematron">

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

        <pattern>
            <rule context="/tei:TEI/tei:teiHeader/tei:fileDesc/tei:titleStmt">
                <assert test="tei:title[@type = 'main']">[E001] Es muss ein title-Element
mit @type="main" geben</assert>
            </rule>
        </pattern>
        <pattern>
            <rule context="/tei:TEI/tei:teiHeader/tei:fileDesc/tei:titleStmt">
                <assert test="tei:title[@type = 'sub']">[E002] Es muss ein title-Element
mit @type="sub" geben</assert>
            </rule>
        </pattern>
</schema>
```

Tipp: Fehler vs. Warnung

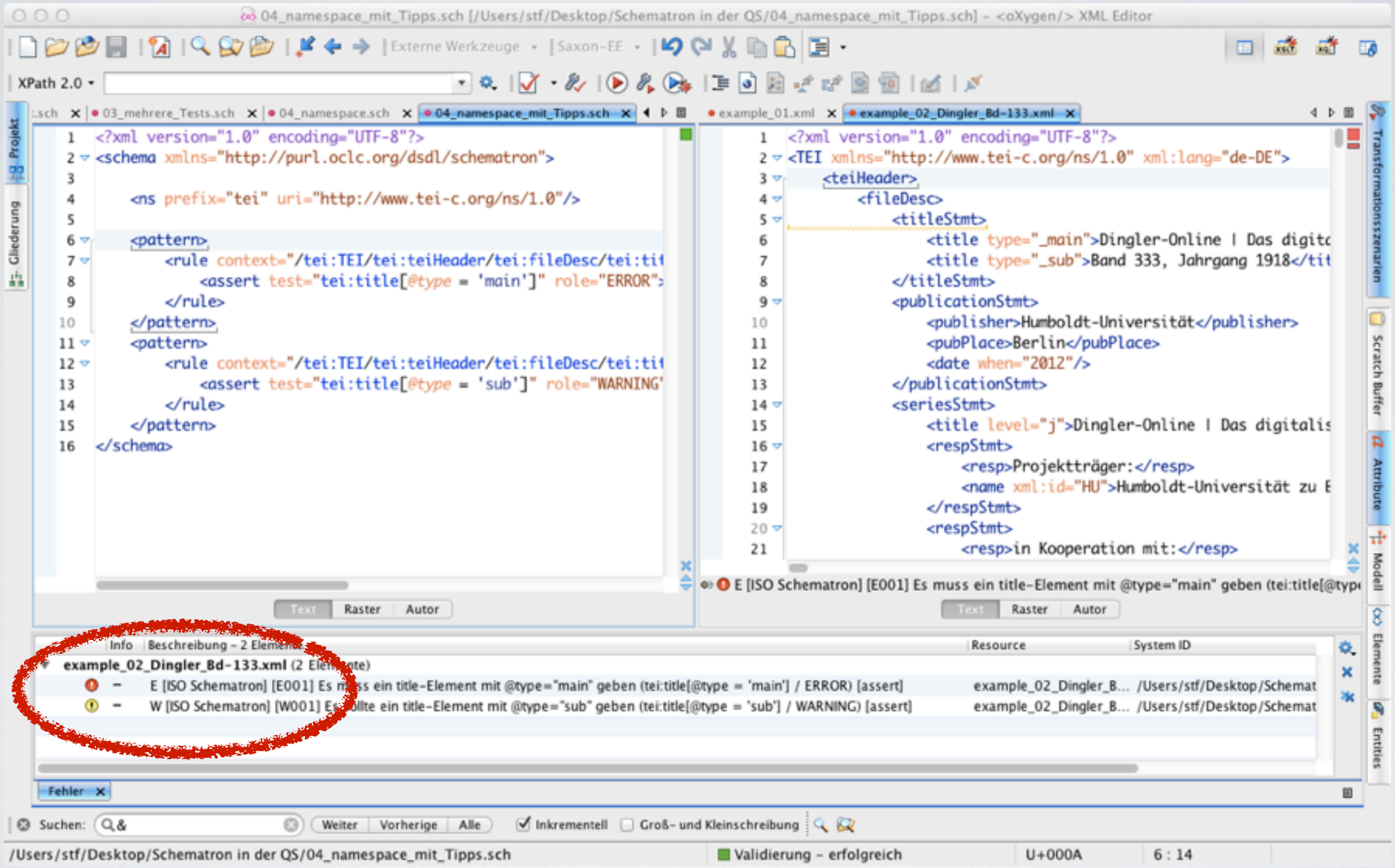
04_namespace_mit_Tipps.sch

example_02_Dingler_Bd-333.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
        <ns prefix="tei" uri="http://www.tei-c.org/ns/1.0"/>


        <pattern>
            <rule context="/tei:TEI/tei:teiHeader/tei:fileDesc/tei:titleStmt">
                <assert test="tei:title[@type = 'main']" role="ERROR">
                                [E001] Es muss ein title-Element mit @type="main" geben</assert>
            </rule>
        </pattern>


        <pattern>
            <rule context="/tei:TEI/tei:teiHeader/tei:fileDesc/tei:titleStmt">
                <assert test="tei:title[@type = 'sub']" role="WARNING">
                                [W001] Es sollte ein title-Element mit @type="sub" geben</assert>
            </rule>
        </pattern>
</schema>
```

- Mögliche Werte für @role in OxygenXML: `fatal, error, warn/warning, info/information`

Tipp: Fehler vs. Warnung



SCHEMATRON FÜR FORTGESCHRITTENE

XPath 2.0

05_XPath_2_usw.sch

- XPath 2.0 macht das Leben vieeeel einfacher.

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron" queryBinding="xslt2">
```

- dann funktionieren auch wichtige Funktionen wie `matches()`, `tokenize()` und Operatoren wie `every … in … satisfies …`

Vergleich mit externen Referenzen I

- ist ein XPath-Thema, nicht Schematron-spezifisch

```
<pattern>
        <rule context="tei:sourceDesc/tei:biblStruct/tei:monogr/tei:idno[@type eq 'local']">

            <assert test="doc(resolve-uri('referenz.xml', base-uri(.) ) )//band[idno/text()
                                                                    eq current()/text()]">
                [E002] Band-ID nicht in externer Referenz gefunden
            </assert>
        </rule>
</pattern>
```

05_XPath_2_usw.sch

example_02_Dingler_Bd-333.xml

- externe xml-Datei mit `doc()`
- `resolve-uri()`, um den Pfad zu finden (Alternative: absolute URL)
- `current()`, weil der Kontext während der Evaluierung des XPath mit `doc()` vom aktuellen Knoten wegbewegt wurde

Vergleich mit externen Referenzen II

- `doc()` funktioniert auch mit Web-Services
- http://xisbn.worldcat.org/webservices/xid/isbn/978-3-596-29433-7?method=fixChecksum&format=xml liefert

```
<rsp xmlns="http://worldcat.org/xid/isbn/" stat="ok">
    <isbn>978-3-596-29433-6</isbn>
</rsp>
```

- Schematron:

06_WebService_nutzen.sch

example_01.xml

```
<pattern>
    <rule context="isbn[string-length(.) eq 17]">
        <assert test=". eq doc(
            concat('http://xisbn.worldcat.org/webservices/xid/isbn/', .,
                '?method=fixChecksum&amp;format=xml') )/*/*:isbn">
            [E025] ungültige ISBN</assert>
    </rule>
</pattern>
```

let + value-of

```
<pattern>
    <rule context="isbn[string-length(.) eq 17]">
        <let name="result"
            value="doc(concat('http://xisbn.worldcat.org/webservices/xid/isbn/', .,
            '?method=fixChecksum&amp;format=xml') )/*/*:isbn"/>
        <assert test=". eq $result" role="ERROR">
            [E025] ungültige ISBN, erwartet »<value-of select="$result"/>«
        </assert>
    </rule>
</pattern>
```

07_let_value-of.sch

example_01.xml

- `let` definiert Variable
  - bessere Übersicht
  - Wiederverwend-barkeit
- `value-of` gibt den Wert eines XPath-Ausdruckes aus
- Variablenname analog zu XSLT

PRAXISBEISPIELE

Attribute ausgefüllt?

example_02_Dingler_Bd-333.xml

```
<tei:text type="issue" n="15" xml:id="is333015">[…]
```

08_Praxisbeispiele.sch

```
<pattern>
        <rule context="tei:text">
                <assert test="normalize-space(@xml:id)" role="ERROR">
                                                [E001] @xml:id muss ausgefüllt werden</assert>
                <assert test="normalize-space(@type)" role="ERROR">
                                                [E002] @type muss ausgefüllt werden</assert>
        </rule>
</pattern>
```

Kindelement vorhanden?

example_02_Dingler_Bd-333.xml

```
<tei:text type="volume" xml:lang="german" n="333" xml:id="pj333">
<tei:milestone type="additional-counting" subtype="age-group"
                                ed="Sechsundsiebzigster Jahrgang" unit="volume" n="333"/>
[…]
```

08_Praxisbeispiele.sch

```
<pattern>
        <rule context="tei:text[@type eq 'volume' ]">
            <assert test="tei:milestone[
                                @type eq 'additional-counting' and
                                @subtype eq 'age-group' and
                                @unit eq 'volume' ]" role="WARNING">
            [W001] tei:milestone[@type eq 'additional-counting' and
            @subtype eq 'age-group' and @unit eq 'volume' ] fehlt</assert>
        </rule>
</pattern>
```

Tipp: gefährlicher =-Operator

```
<tei:div type="misc_undef" xml:id="mi333is14_1">[…]
<tei:div type="patent">[…]
```

example_02_Dingler_Bd-333.xml

```
<pattern>
    <rule context="tei:div[@type = ('misc_undef', 'patent')]">
        <assert test="normalize-space(@xml:id)" role="ERROR">
                                    [E011] @xml:id muss ausgefüllt werden</assert>
    </rule>
</pattern>
```

08_Praxisbeispiele.sch

- Wenn man nicht ganz genau weiß, was man macht (Und wer weiß das schon?), gilt: Niemals den =-Operator verwenden, immer `eq`!
- siehe auch letzte Folie

Element genau einmal vorhanden?

08_Praxisbeispiele.sch | example_02_Dingler_Bd-333.xml

```
<pattern>
    <rule context="tei:TEI">
        <let name="volumes" value="//tei:text[@type eq 'volume' ]"/>
        <assert test="$volumes" role="ERROR">
                                        [E021] tei:text[@type eq 'volume' ] fehlt</assert>
        <report test="$volumes[2]" role="ERROR">
                                    [E022] mehr als ein tei:text[@type eq 'volume']</report>


        <let name="titlePages" value="$volumes//tei:titlePage[@type eq 'volume' ]"/>
        <assert test="$titlePages" role="ERROR">
                    [E023] tei:text[@type eq 'volume' ]//tei:titlePage[@type eq 'volume' ]
                    fehlt</assert>
        <report test="$titlePages[2]" role="ERROR">
                    [E024] mehr als ein
                    tei:text[@type eq 'volume' ]//tei:titlePage[@type eq 'volume' ]</report>
    </rule>
</pattern>
```

interne Referenzen vorhanden?

example_02_Dingler_Bd-333.xml

```
<tei:rendition xml:id="large" scheme="css">font-size: large;</rendition>
<tei:rendition xml:id="bold" scheme="css">font-weight: bold</rendition>
[…]
<tei:p rendition="#bold #large">[…]
```

08_Praxisbeispiele.sch

```
<pattern>
    <rule context="@*[starts-with(., '#')]">
        <let name="idrefs" value="for $i in tokenize(., '[ ]')[normalize-space(.)]
            return translate($i, '#', '')"/>
        <assert test="every $idref in $idrefs satisfies //@xml:id[. eq $idref]">
            [E031] fehlende(s) //@xml:id für »<value-of select="."/>«</assert>
    </rule>
</pattern>
```

externe Referenzen vorhanden?

```
<tei:pb n="interleaf" facs="32199908Z/00000004" xml:id="pj333_pbinterleaf_003"/>
```

example_02_Dingler_Bd-333.xml

08_Praxisbeispiele.sch

```
<pattern>
    <rule context="@facs[normalize-space(.)]">
        <let name="filename"
            value="concat(substring-after(., '/'), '.png')"/>
        <let name="listed_files"
            value="doc(resolve-uri('files.xml', base-uri(.) ) )//Datei/Name[. eq $filename]"/>
        <assert test="$listed_files" role="ERROR">
                    [E041] referenzierte Datei »<value-of select="$filename"/>«
                    nicht in Dateiliste verzeichnet</assert>
    </rule>
</pattern>
```

- mit durch externes Tool generiertem `files.xml`:

```
<root>
    <Verzeichnis>
        <Name>32199908Z</Name>
        <Datei>
            <Name>00000001.png</Name>
```

files.xml

SCHEMATRON FÜR PROFIS

xsl:key und key()

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      queryBinding="xslt2">

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

      <xsl:key name="xml-IDs" match="@xml:id" use="."/>

      <pattern>
            <rule context="@*[starts-with(., '#')]">
                  <let name="idrefs" value="
                        for $i in tokenize(., '[ ]')[normalize-space(.)]
                              return translate($i, '#', '')"/>
                  <assert
                        test="every $idref in $idrefs
                                    satisfies key('xml-IDs', $idref)"
                        role="ERROR">
                              [E031] fehlende(s) //@xml:id(s) für
                              »<value-of select="."/>«</assert>
            </rule>
      </pattern>
</schema>
```

09_keys.sch

example_02_Dingler_Bd-333.xml

- Bei sehr vielen Referenzen können Keys einen Geschwindigkeits-vorteil bringen.
- Schematron »borgt« sich Keys von XSLT.
- Dazu muss der XSLT-Namespace in Schematron bekannt gemacht werden.

xsl:key und key()

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      queryBinding="xslt2">

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

      <xsl:key name="xml-IDs" match="@xml:id" use="."/>

      <pattern>
            <rule context="@*[starts-with(., '#')]">
                  <let name="idrefs" value="
                        for $i in tokenize(., '[ ]')[normalize-space(.)]
                              return translate($i, '#', '')"/>
                  <assert
                        test="every $idref in $idrefs
                              satisfies key('xml-IDs', $idref)"
                        role="ERROR">
                        [E031] fehlende(s) //@xml:id(s) für
                        »<value-of select="."/>«</assert>
            </rule>
      </pattern>
</schema>
```

09_keys.sch

example_02_Dingler_Bd-333.xml

- Bei sehr vielen Referenzen können Keys einen Geschwindigkeits-vorteil bringen.
- Schematron »borgt« sich Keys von XSLT.
- Dazu muss der XSLT-Namespace in Schematron bekannt gemacht werden.

xsl:key und key()

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      queryBinding="xslt2">

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

      <xsl:key name="xml-IDs" match="@xml:id" use="."/>

      <pattern>
            <rule context="@*[starts-with(., '#')]">
                  <let name="idrefs" value="
                        for $i in tokenize(., '[ ]')[normalize-space(.)]
                              return translate($i, '#', '')"/>
                  <assert
                        test="every $idref in $idrefs
                                    satisfies key('xml-IDs', $idref)"
                        role="ERROR">
                              [E031] fehlende(s) //@xml:id(s) für
                              »<value-of select="."/>«</assert>
            </rule>
      </pattern>
</schema>
```

09_keys.sch

example_02_Dingler_Bd-333.xml

- Bei sehr vielen Referenzen können Keys einen Geschwindigkeits-vorteil bringen.
- Schematron »borgt« sich Keys von XSLT.
- Dazu muss der XSLT-Namespace in Schematron bekannt gemacht werden.

xsl:key und key()

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    queryBinding="xslt2">

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

    <xsl:key name="xml-IDs" match="@xml:id" use="."/>

    <pattern>
        <rule context="@*[starts-with(., '#')]">
            <let name="idrefs" value="
                for $i in tokenize(., '[ ]')[normalize-space(.)]
                    return translate($i, '#', '')"/>
            <assert
                test="every $idref in $idrefs
                        satisfies key('xml-IDs', $idref)"
                role="ERROR">
                    [E031] fehlende(s) //@xml:id(s) für
                    »<value-of select="."/>«</assert>
        </rule>
    </pattern>
</schema>
```

09_keys.sch

example_02_Dingler_Bd-333.xml

- Bei sehr vielen Referenzen können Keys einen Geschwindigkeits-vorteil bringen.
- Schematron »borgt« sich Keys von XSLT.
- Dazu muss der XSLT-Namespace in Schematron bekannt gemacht werden.

Funktions-Definitionen I

- Benutzerdefinierte XPath-Funktionen ermöglichen:
  - die Wiederverwendung von Code,
  - die Aufteilung komplizierter Berechnungen,
  - damit mehr Übersicht und
  - bessere Wartbarkeit.

Funktions-Definitionen II

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      queryBinding="xslt2">

      <ns prefix="my" uri="irgendeine_gueltige_URI"/>

      <xsl:function name="my:ids" as="xs:string*">
            <xsl:param name="idref-string" as="xs:string?"/>
            <xsl:sequence select="for $i in tokenize($idref-string, '[ ]')[normalize-space(.)]
                                                                  return translate($i, '#', '')"/>
      </xsl:function>

      <pattern>
            <rule context="@*[starts-with(., '#')]">
                  <let name="idrefs" value="my:ids(.)"/>
                  <assert test="every $idref in $idrefs satisfies //@xml:id[. eq $idref]">
                        [E031] fehlende(s) //@xml:id für »<value-of select="."/>«</assert>
            </rule>
      </pattern>
</schema>
```

10_Funktionen_intern.sch

example_02_Dingler_Bd-333.xml

Funktions-Definitionen II

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      queryBinding="xslt2">

      <ns prefix="my" uri="irgendeine_gueltige_URI"/>

      <xsl:function name="my:ids" as="xs:string*">
            <xsl:param name="idref-string" as="xs:string?"/>
            <xsl:sequence select="for $i in tokenize($idref-string, '[ ]')[normalize-space(.)]
                                                                  return translate($i, '#', '')"/>
      </xsl:function>

      <pattern>
            <rule context="@*[starts-with(., '#')]">
                  <let name="idrefs" value="my:ids(.)"/>
                  <assert test="every $idref in $idrefs satisfies //@xml:id[. eq $idref]">
                        [E031] fehlende(s) //@xml:id für »<value-of select="."/>«</assert>
            </rule>
      </pattern>
</schema>
```

10_Funktionen_intern.sch

example_02_Dingler_Bd-333.xml

Funktions-Definitionen II

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      queryBinding="xslt2">

      <ns prefix="my" uri="irgendeine_gueltige_URI"/>

      <xsl:function name="my:ids" as="xs:string*">
            <xsl:param name="idref-string" as="xs:string?"/>
            <xsl:sequence select="for $i in tokenize($idref-string, '[ ]')[normalize-space(.)]
                                                                  return translate($i, '#', '')"/>
      </xsl:function>

      <pattern>
            <rule context="@*[starts-with(., '#')]">
                  <let name="idrefs" value="my:ids(.)"/>
                  <assert test="every $idref in $idrefs satisfies //@xml:id[. eq $idref]">
                        [E031] fehlende(s) //@xml:id für »<value-of select="."/>«</assert>
            </rule>
      </pattern>
</schema>
```

10_Funktionen_intern.sch

example_02_Dingler_Bd-333.xml

Funktions-Definitionen II

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    queryBinding="xslt2">

    <ns prefix="my" uri="irgendeine_gueltige_URI"/>

    <xsl:function name="my:ids" as="xs:string*">
        <xsl:param name="idref-string" as="xs:string?"/>
        <xsl:sequence select="for $i in tokenize($idref-string, '[ ]')[normalize-space(.)]
                                                    return translate($i, '#', '')"/>
    </xsl:function>

    <pattern>
        <rule context="@*[starts-with(., '#')]">
            <let name="idrefs" value="my:ids(.)"/>
            <assert test="every $idref in $idrefs satisfies //@xml:id[. eq $idref]">
                [E031] fehlende(s) //@xml:id für »<value-of select="."/>«</assert>
        </rule>
    </pattern>
</schema>
```

10_Funktionen_intern.sch

example_02_Dingler_Bd-333.xml

Funktions-Definitionen II

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      queryBinding="xslt2">

      <ns prefix="my" uri="irgendeine_gueltige_URI"/>

      <xsl:function name="my:ids" as="xs:string*">
            <xsl:param name="idref-string" as="xs:string?"/>
            <xsl:sequence select="for $i in tokenize($idref-string, '[ ]')[normalize-space(.)]
                                                                  return translate($i, '#', '')"/>
      </xsl:function>

      <pattern>
            <rule context="@*[starts-with(., '#')]">
                  <let name="idrefs" value="my:ids(.)"/>
                  <assert test="every $idref in $idrefs satisfies //@xml:id[. eq $idref]">
                        [E031] fehlende(s) //@xml:id für »<value-of select="."/>«</assert>
            </rule>
      </pattern>
</schema>
```

10_Funktionen_intern.sch

example_02_Dingler_Bd-333.xml

Funktions-Definitionen II

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      queryBinding="xslt2">

   <ns prefix="my" uri="irgendeine_gueltige_URI"/>

      <xsl:function name="my:ids" as="xs:string*">
            <xsl:param name="idref-string" as="xs:string?"/>
            <xsl:sequence select="for $i in tokenize($idref-string, '[ ]')[normalize-space(.)]
                                                                  return translate($i, '#', '')"/>
      </xsl:function>

      <pattern>
            <rule context="@*[starts-with(., '#')]">
                  <let name="idrefs" value="my:ids(.)"/>
                  <assert test="every $idref in $idrefs satisfies //@xml:id[. eq $idref]">
                        [E031] fehlende(s) //@xml:id für »<value-of select="."/>«</assert>
            </rule>
      </pattern>
</schema>
```

10_Funktionen_intern.sch

example_02_Dingler_Bd-333.xml

Funktions-Definitionen II

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      queryBinding="xslt2">

      <ns prefix="my" uri="irgendeine_gueltige_URI"/>

      <xsl:function name="my:ids" as="xs:string*">
            <xsl:param name="idref-string" as="xs:string?"/>
            <xsl:sequence select="for $i in tokenize($idref-string, '[ ]')[normalize-space(.)]
                                                                  return translate($i, '#', '')"/>
      </xsl:function>

      <pattern>
            <rule context="@*[starts-with(., '#')]">
                  <let name="idrefs" value="my:ids(.)"/>
                  <assert test="every $idref in $idrefs satisfies //@xml:id[. eq $idref]">
                        [E031] fehlende(s) //@xml:id für »<value-of select="."/>«</assert>
            </rule>
      </pattern>
</schema>
```

10_Funktionen_intern.sch

example_02_Dingler_Bd-333.xml

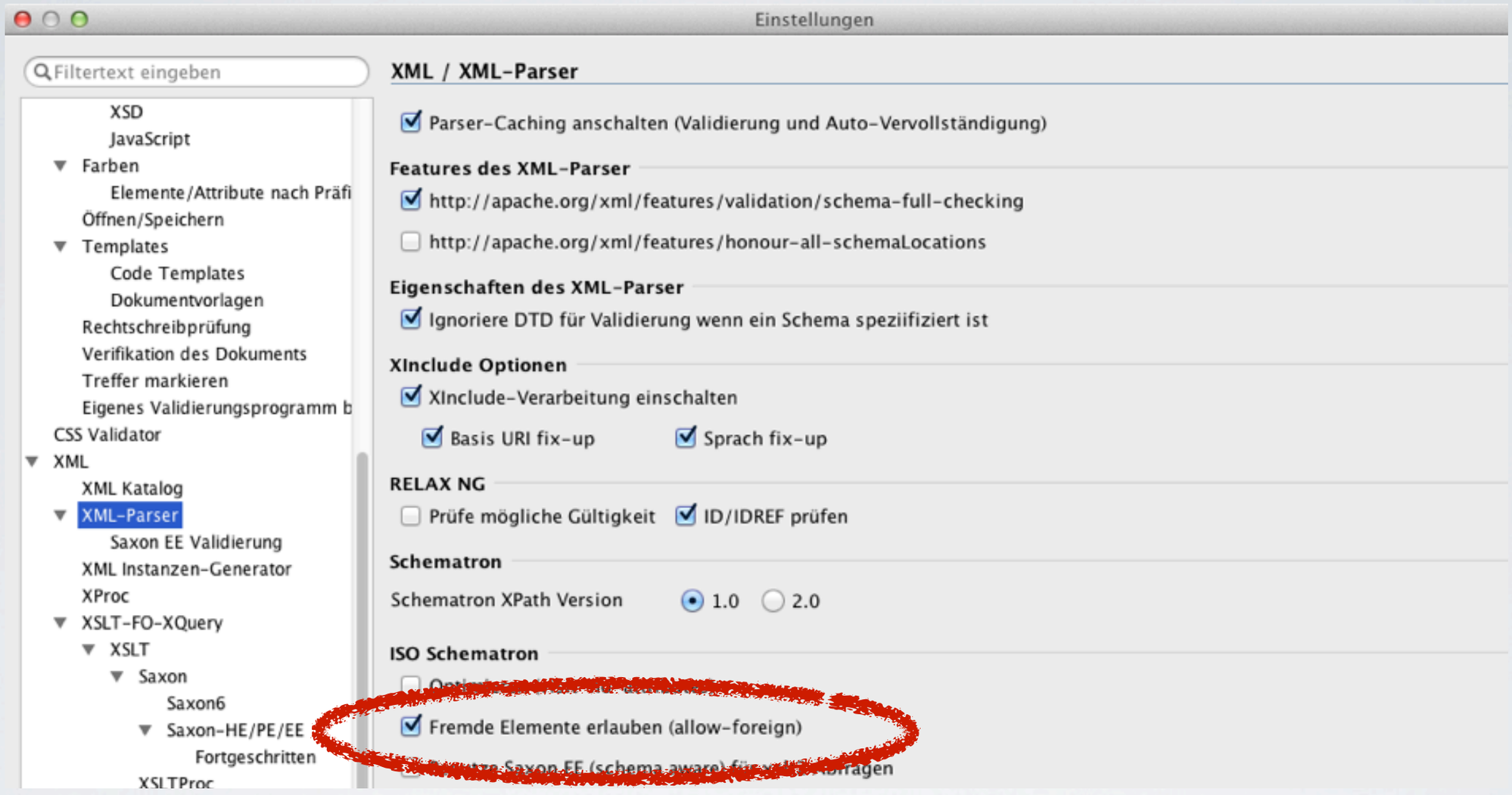
Funktions-Definitionen II

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      queryBinding="xslt2">

      <ns prefix="my" uri="irgendeine_gueltige_URI"/>

      <xsl:function name="my:ids" as="xs:string*">
            <xsl:param name="idref-string" as="xs:string?"/>
            <xsl:sequence select="for $i in tokenize($idref-string, '[ ]')[normalize-space(.)]
                                                                  return translate($i, '#', '')"/>
      </xsl:function>

      <pattern>
            <rule context="@*[starts-with(., '#')]">
                  <let name="idrefs" value="my:ids(.)"/>
                  <assert test="every $idref in $idrefs satisfies //@xml:id[. eq $idref]">
                        [E031] fehlende(s) //@xml:id für »<value-of select="."/>«</assert>
            </rule>
      </pattern>
</schema>
```

10_Funktionen_intern.sch

example_02_Dingler_Bd-333.xml

Funktions-Definitionen III



Voraussetzung: XSLT-Verarbeitung in Schematron muss freigeschaltet werden.
Z. B. in OxygenXML: Optionen ⇒ Einstellungen ⇒ XML ⇒ XML-Parser ⇒

Fremde Elemente erlauben (allow-foreign)

externe Funktionen I

- projektübergreifende Standardfunktionen
  - wiederverwendbarer Code, v. a. im Hinblick auf XSLT, mit dem die Daten verarbeitet werden sollen
  - fremde Funktionsbibliotheken
  - Auf externe XSLT-Dateien können XSLT-Prüfwerkzeuge angewendet werden (z. B. XSpec).

externe Funktionen II

11_Funktionen_extern.sch

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

      <xsl:include href="my-functions.xsl"/>

      <ns prefix="my" uri="irgendeine_gueltige_URI"/>

      <pattern>
            <rule context="@*[starts-with(., '#')]">
                  <let name="idrefs" value="my:ids(.)"/>
                  <assert
                        test="every $idref in $idrefs satisfies
                        /@xml:id[. eq $idref]"
role="ERROR">
                        [E031] fehlende(s) //@xml:id für
                              »<value-of select="."/>«</
assert>
            </rule>
      </pattern>
</schema>
```

my-functions.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:my="irgendeine_gueltige_URI"
      exclude-result-prefixes="#all"
      version="2.0">

      <xsl:function name="my:ids" as="xs:string*">
            <xsl:param name="idref-string" as="xs:string?"/>
            <xsl:sequence
                  select="for $i in tokenize($idref-string, '[ ]')
                                    [normalize-space(.)]
                              return translate($i, '#', '')"/>
      </xsl:function>

</xsl:stylesheet>
```

externe Funktionen II

11_Funktionen_extern.sch

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

        <xsl:include href="my-functions.xsl"/>

        <ns prefix="my" uri="irgendeine_gueltige_URI"/>

        <pattern>
            <rule context="@*[starts-with(., '#')]">
                <let name="idrefs" value="my:ids(.)"/>
                <assert
                    test="every $idref in $idrefs satisfies
                    /@xml:id[. eq $idref]"
role="ERROR">
                    [E031] fehlende(s) //@xml:id für
                        »<value-of select="."/>«</
assert>
            </rule>
        </pattern>
</schema>
```

my-functions.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:my="irgendeine_gueltige_URI"
        exclude-result-prefixes="#all"
        version="2.0">

    <xsl:function name="my:ids" as="xs:string*">
        <xsl:param name="idref-string" as="xs:string?"/>
        <xsl:sequence
            select="for $i in tokenize($idref-string, '[ ]')
                                [normalize-space(.)]
                    return translate($i, '#', '')"/>
    </xsl:function>

</xsl:stylesheet>
```

externe Funktionen II

11_Funktionen_extern.sch

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

      <xsl:include href="my-functions.xsl"/>

      <ns prefix="my" uri="irgendeine_gueltige_URI"/>

      <pattern>
            <rule context="@*[starts-with(., '#')]">
                  <let name="idrefs" value="my:ids(.)"/>
                  <assert
                        test="every $idref in $idrefs satisfies
                        /@xml:id[. eq $idref]"
role="ERROR">
                        [E031] fehlende(s) //@xml:id für
                              »<value-of select="."/>«</
assert>
            </rule>
      </pattern>
</schema>
```

my-functions.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:my="irgendeine_gueltige_URI"
      exclude-result-prefixes="#all"
      version="2.0">

      <xsl:function name="my:ids" as="xs:string*">
            <xsl:param name="idref-string" as="xs:string?"/>
            <xsl:sequence
                  select="for $i in tokenize($idref-string, '[ ]')
                                    [normalize-space(.)]
                              return translate($i, '#', '')"/>
      </xsl:function>

</xsl:stylesheet>
```

externe Funktionen II

11_Funktionen_extern.sch

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

      <xsl:include href="my-functions.xsl"/>

      <ns prefix="my" uri="irgendeine_gueltige_URI"/>
      <pattern>
            <rule context="@*[starts-with(., '#')]">
                  <let name="idrefs" value="my:ids(.)"/>
                  <assert
                        test="every $idref in $idrefs satisfies
                        /@xml:id[. eq $idref]"
role="ERROR">
                        [E031] fehlende(s) //@xml:id für
                              »<value-of select="."/>«</
assert>
            </rule>
      </pattern>
</schema>
```

my-functions.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:my="irgendeine_gueltige_URI"
      exclude-result-prefixes="#all"
      version="2.0">

      <xsl:function name="my:ids" as="xs:string*">
            <xsl:param name="idref-string" as="xs:string?"/>
            <xsl:sequence
                  select="for $i in tokenize($idref-string, '[ ]')
                                    [normalize-space(.)]
                              return translate($i, '#', '')"/>
      </xsl:function>

</xsl:stylesheet>
```

Tipp: solved-Attribute I

- Ein zusätzliches, optionales Attribut am Kontext-Element unterdrückt Schematron-Meldungen.
- Das Attribut wird mit einem je `assert/report` definierten Token befüllt, wenn die Fehlermeldung unterdrückt werden soll, z.B. `<TEI solved="MissingTitlePage">[…]`.
- Damit tauchen gelöste Fälle bei späteren Prüfungen nicht mehr auf.

Tipp: solved-Attribute II

12_solved-attribut.sch

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        queryBinding="xslt2">
        <ns prefix="my" uri="irgendeine_gueltige_URI"/>
        <ns prefix="tei" uri="http://www.tei-c.org/ns/1.0"/>
        <xsl:function name="my:solved" as="xs:boolean">
                <xsl:param name="solved-string" as="xs:string?"/>
                <xsl:param name="searched-token" as="xs:string"/>
                <xsl:choose>
                        <xsl:when test="some $i in tokenize($solved-string, '\s') satisfies ($i eq $searched-token)">
                                <xsl:sequence select="true()"/>
                        </xsl:when>
                        <xsl:otherwise>
                                <xsl:sequence select="false()"/>
                        </xsl:otherwise>
                </xsl:choose>
        </xsl:function>
        <pattern>
                <rule context="tei:TEI">
                        <let name="volumes" value="//tei:text[@type eq 'volume' ]"/>
                        <let name="titlePages" value="$volumes//tei:titlePage[@type eq 'volume' ]"/>
                        <assert test="$titlePages or my:solved(@solved, 'MissingTitlePage')" role="WARNING">[W023]
tei:text[@type eq 'volume' ]//tei:titlePage[@type eq 'volume' ] fehlt (Solved-Token: »MissingTitlePage«)</assert>
                        <report test="$titlePages[2] and not(my:solved(@solved, 'MoreThanOneTitlePage'))"
role="WARNING">[W024] mehr als ein tei:text[@type eq 'volume' ]//tei:titlePage[@type eq 'volume' ] (Solved-Token:
»MoreThanOneTitlePage«)</report>
                </rule>
        </pattern>
</schema>
```

Tipp: solved-Attribute II

12_solved-attribut.sch

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        queryBinding="xslt2">
        <ns prefix="my" uri="irgendeine_gueltige_URI"/>
        <ns prefix="tei" uri="http://www.tei-c.org/ns/1.0"/>
    <xsl:function name="my:solved" as="xs:boolean">
            <xsl:param name="solved-string" as="xs:string?"/>
            <xsl:param name="searched-token" as="xs:string"/>
            <xsl:choose>
                    <xsl:when test="some $i in tokenize($solved-string, '\s') satisfies ($i eq $searched-token)">
                            <xsl:sequence select="true()"/>
                    </xsl:when>
                    <xsl:otherwise>
                            <xsl:sequence select="false()"/>
                    </xsl:otherwise>
            </xsl:choose>
    </xsl:function>
        <pattern>
                <rule context="tei:TEI">
                        <let name="volumes" value="//tei:text[@type eq 'volume' ]"/>
                        <let name="titlePages" value="$volumes//tei:titlePage[@type eq 'volume' ]"/>
                        <assert test="$titlePages or my:solved(@solved, 'MissingTitlePage')" role="WARNING">[W023]
tei:text[@type eq 'volume' ]//tei:titlePage[@type eq 'volume' ] fehlt (Solved-Token: »MissingTitlePage«)</assert>
                        <report test="$titlePages[2] and not(my:solved(@solved, 'MoreThanOneTitlePage'))"
role="WARNING">[W024] mehr als ein tei:text[@type eq 'volume' ]//tei:titlePage[@type eq 'volume' ] (Solved-Token:
»MoreThanOneTitlePage«)</report>
                </rule>
        </pattern>
</schema>
```

Tipp: solved-Attribute II

12_solved-attribut.sch

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        queryBinding="xslt2">
        <ns prefix="my" uri="irgendeine_gueltige_URI"/>
        <ns prefix="tei" uri="http://www.tei-c.org/ns/1.0"/>
        <xsl:function name="my:solved" as="xs:boolean">
                <xsl:param name="solved-string" as="xs:string?"/>
                <xsl:param name="searched-token" as="xs:string"/>
                <xsl:choose>
                        <xsl:when test="some $i in tokenize($solved-string, '\s') satisfies ($i eq $searched-token)">
                                <xsl:sequence select="true()"/>
                        </xsl:when>
                        <xsl:otherwise>
                                <xsl:sequence select="false()"/>
                        </xsl:otherwise>
                </xsl:choose>
        </xsl:function>
        <pattern>
                <rule context="tei:TEI">
                        <let name="volumes" value="//tei:text[@type eq 'volume' ]"/>
                        <let name="titlePages" value="$volumes//tei:titlePage[@type eq 'volume' ]"/>
                        <assert test="$titlePages or my:solved(@solved, 'MissingTitlePage')" role="WARNING">[W023]
tei:text[@type eq 'volume' ]//tei:titlePage[@type eq 'volume' ] fehlt (Solved-Token: »MissingTitlePage«)</assert>
                        <report test="$titlePages[2] and not(my:solved(@solved, 'MoreThanOneTitlePage'))"
role="WARNING">[W024] mehr als ein tei:text[@type eq 'volume' ]//tei:titlePage[@type eq 'volume' ] (Solved-Token:
»MoreThanOneTitlePage«)</report>
                </rule>
        </pattern>
</schema>
```

Tipp: solved-Attribute II

12_solved-attribut.sch

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        queryBinding="xslt2">
        <ns prefix="my" uri="irgendeine_gueltige_URI"/>
        <ns prefix="tei" uri="http://www.tei-c.org/ns/1.0"/>
        <xsl:function name="my:solved" as="xs:boolean">
                <xsl:param name="solved-string" as="xs:string?"/>
                <xsl:param name="searched-token" as="xs:string"/>
                <xsl:choose>
                        <xsl:when test="some $i in tokenize($solved-string, '\s') satisfies ($i eq $searched-token)">
                                <xsl:sequence select="true()"/>
                        </xsl:when>
                        <xsl:otherwise>
                                <xsl:sequence select="false()"/>
                        </xsl:otherwise>
                </xsl:choose>
        </xsl:function>
        <pattern>
                <rule context="tei:TEI">
                        <let name="volumes" value="//tei:text[@type eq 'volume' ]"/>
                        <let name="titlePages" value="$volumes//tei:titlePage[@type eq 'volume' ]"/>
                        <assert test="$titlePages or my:solved(@solved, 'MissingTitlePage')" role="WARNING">[W023]
tei:text[@type eq 'volume' ]//tei:titlePage[@type eq 'volume' ] (Solved-Token: »MissingTitlePage«)</assert>
                        <report test="$titlePages[2] and not(my:solved(@solved, 'MoreThanOneTitlePage'))"
role="WARNING">[W024] Mehr als ein tei:text[@type eq 'volume' ]//tei:titlePage[@type eq 'volume' ] (Solved-Token:
»MoreThanOneTitlePage«)</report>
                </rule>
        </pattern>
</schema>
```

Tipp: solved-Attribute II

12_solved-attribut.sch

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        queryBinding="xslt2">
        <ns prefix="my" uri="irgendeine_gueltige_URI"/>
        <ns prefix="tei" uri="http://www.tei-c.org/ns/1.0"/>
        <xsl:function name="my:solved" as="xs:boolean">
                <xsl:param name="solved-string" as="xs:string?"/>
                <xsl:param name="searched-token" as="xs:string"/>
                <xsl:choose>
                        <xsl:when test="some $i in tokenize($solved-string, '\s') satisfies ($i eq $searched-token)">
                                <xsl:sequence select="true()"/>
                        </xsl:when>
                        <xsl:otherwise>
                                <xsl:sequence select="false()"/>
                        </xsl:otherwise>
                </xsl:choose>
        </xsl:function>
        <pattern>
                <rule context="tei:TEI">
                        <let name="volumes" value="//tei:text[@type eq 'volume' ]"/>
                        <let name="titlePages" value="$volumes//tei:titlePage[@type eq 'volume' ]"/>
                        <assert test="$titlePages or my:solved(@solved, 'MissingTitlePage')" role="WARNING">[W023]
tei:text[@type eq 'volume' ]//tei:titlePage[@type eq 'volume' ] fehlt (Solved-Token: »MissingTitlePage«)</assert>
                        <report test="$titlePages[2] and not(my:solved(@solved, 'MoreThanOneTitlePage'))"
role="WARNING">[W024] mehr als ein tei:text[@type eq 'volume' ]//tei:titlePage[@type eq 'volume' ] (Solved-Token:
»MoreThanOneTitlePage«)</report>
                </rule>
        </pattern>
</schema>
```

Schematron-Code-Prüfung I

- Überprüfung, ob Programmier-Regeln eingehalten werden
  - Regeln müssen aufgestellt werden, z. B.
    - jedes `assert` oder `report` muss ein `@role` haben
    - `@role` muss einen der Werte `fatal`, `error`, `warn/warning`, `info/information` haben

Schematron-Code-Prüfung II

13_Schematron_Test.sch

```
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron"
      queryBinding="xslt2">

      <sch:ns prefix="sch" uri="http://purl.oclc.org/dsdl/schematron"/>

      <sch:pattern>
            <sch:rule context="sch:assertIsch:report">
                  <sch:assert test="normalize-space(@role)" role="WARNING">
                        [W001] Jedes assert und report soll ein @role haben!</sch:assert>
            </sch:rule>
      </sch:pattern>

      <sch:pattern>
            <sch:rule context="sch:assert/@roleIsch:report/@role">
                  <sch:assert test="
                        some $i in ('fatal', 'error', 'warn', 'info') satisfies ($i eq lower-case(.) )">
    [W002] @role muss einer der Werte »fatal«, »error«, »warn«, »info« sein!</sch:assert>
            </sch:rule>
      </sch:pattern>
</sch:schema>
```

Heute nicht besprochen wurden

- `phase` zum logischen Aufteilen von Prüfungen,
- `include` zum Modularisieren,
- diverse Möglichkeiten zur Formatierung,
- abstrakte rules und pattern
- u. a.,
- aber diese Möglichkeiten von Schematron wird man vielleicht nicht so schnell brauchen.

LITERATUR

Zum Weiterlesen

- Marko Hedler/Manuel Montero Pineda/Nico Kutscherauer: **Schematron. Effiziente Business Rules für XML-Dokumente**. Heidelberg: dpunkt.verlag, 2011. ISBN 978-3-89864-721-2
- Dave Pawson/Roger Costello/Florent Georges: **ISO Schematron tutorial. An introductroy guide.** 2007. http://www.dpawson.co.uk/schematron/
- diverse Blogbeiträge unter http://blog.expedimentum.com/kategorien/xml-validierung/schematron/

SCHEMATRON IN DER QUALITÄTSSICHERUNG

Herzlichen Dank für Ihre Aufmerksamkeit!

Stefan Krause
Editura GmbH & Co. KG, Berlin
http://www.editura.de

FRAGEN?

PROJEKTE: ÜBUNGEN, LÖSUNGEN

Anmerkung: Die folgenden Folien beziehen sich auf Fragen, die während des Präsentation gestellt wurden, und waren deshalb im originalen Vortrag nicht enthalten.

Worte, die mit »u« beginnen

Nicht_mit_u.sch

example_02_Dingler_Bd-333.xml

```
<schema
        xmlns="http://purl.oclc.org/dsdl/schematron"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        queryBinding="xslt2">

        <ns prefix="my" uri="irgendeine_gueltige_URI"/>

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

        <xsl:include href="my-functions.xsl"/>

        <pattern>
                <rule context="tei:text//text()">
                        <let name="gesuchter_wortanfang" value="'u'"/>
                        <assert test="every $i in tokenize(., ' ') satisfies not(starts-with($i, $gesuchter_wortanfang))">
                                Wörter dürfen nicht mit »<value-of select="$gesuchter_wortanfang"/>« beginnen. Kontext:
                                »<value-of select="my:render(., $gesuchter_wortanfang)"/>«</assert>
                </rule>
        </pattern>
</schema>
```

- Das Darstellen einer Fundstelle als übersichtliche Wortgruppe ist relativ aufwendig und wurde daher in die externen XSLT-Funktionen ausgelagert.

Warum sind = und != böse?

```
<palette>
        <braun>
                <farbe>rot</farbe>
                <farbe>grün</farbe>
        </braun>
        <orange>
                <farbe>rot</farbe>
                <farbe>gelb</farbe>
        </orange>
</palette>
```

- `/palette/braun//text() = /palette/orange//text()` evaluiert zu `true()`
- `/palette/braun//text() != /palette/orange//text()` evaluiert zu `true()`
- alternativ in Sequenz-Schreibweise der atomaren Werte:
  - `('rot', 'grün') = ('rot', 'gelb')` evaluiert zu `true()`
  - `('rot', 'grün') != ('rot', 'gelb')` evaluiert zu `true()`
  - Bonus `('rot', 'rot') != ('rot')` evaluiert zu `false()`
- Bei Verwendung von `eq` oder `ne` wird dagegen ein Fehler erzeugt.
- Bei Mischung von ungetypten und (implizit) getypten Werten sowie in Verbindung mit Leersequenzen gibt es weitere Merkwürdigkeiten.
- zum Hintergrund: http://www.w3.org/TR/xpath20/#id-comparisons