XML Qualitätssicherung

Thomas Reinwart

XML befindet sich heutzutage in jeder modernen Applikation oder Schnittstelle. Die darin strukturierten Daten haben oft eine gewisse Komplexität und deren Felder eine Abhängigkeit untereinander, es ist also eine Art Businesslogik im XML untergebracht. Für die Validierung eines XMLs verwendet man üblicherweise eine XML Schema Definition, ein XSD File. Wer ist nicht schon mal vor dem Problem gestanden, trotz eines gut durchdachten XSD Files seine XML Daten nicht vollständig prüfen zu können?

Was kann XSD prüfen?

Stellen wir uns eine einfache Bedingung zweier Datenfelder vor. Die Datenstruktur eines Personenstamms enthält die Felder Geschlecht und Schwangerschaft. Durch eine fehlerhafte Verarbeitung kam im Datensatz die Kombination männlich und schwanger zustande, was für Heiterkeit sorgt und jedem Anwender natürlich gleich auffallen wird. Das ist zwar ein simples Beispiel, weit komplexer wird es jedoch, wenn etwa zahlreiche Bereiche von Messwerten im XML vorliegen. Solche Werte können untereinander eine entscheidende Abhängigkeit oder auch Ausschließungsgründe haben und in der weiteren Verarbeitung weitreichende Konsequenzen nach sich ziehen. Jeder der Leser hat bestimmt schon mal einen Laborbefund seines Blutes in Händen gehabt, ein anschauliches Beispiel wo Grenzwerte Abhängigkeiten untereinander haben. Hier hat jeder Wert seinen Bereich, aber gewisse Wertebereiche sind in Relation nicht zulässig, lösen somit beim Report am Ende des Berichts gewissermaßen einen Alarmzustand aus.

XSD beschreibt die Struktur des XMLs und die Datentypen der Elemente und Attribute. Aber das Schema unterstützt keine schematischen Prüfungen. Nur mit einer geeigneten Prüfsprache lassen sich die Business Rules einer XML Struktur schnell und automatisch prüfen.

Es fehlt also die Möglichkeit, eine geeignete Qualitätsprüfung der vorliegenden XML Daten durchzuführen. Sie muss die Prüfung aller Business Rules, Regeln für Namensbereiche oder auch ein Constraint Check einschließen.

Qualitätsicherung mit Schematron

Anders als DTD oder XML Schema dient Schematron [1] nicht zur Definition sondern zur Validierung von Inhalten der XML Dokumente. Schematron ist eine Sprache die nicht auf Grammatiken basiert. Dadurch wird es möglich, das XML anhand von Mustern und Regeln zu prüfen. Die Regeln selber sind auch in XML definiert. Damit lassen sich viele Komplexe Strukturen unterbringen, die sich mit rein grammatikbasierenden Schemasprachen schwer bis gar nicht beschreiben lassen.

Versionsüberblick

Schematron ist inzwischen ein ISO Standard geworden, der etwa im Finanzsektor, bei Versicherung oder in der beim Austausch medizinischer Daten eingesetzt wird. Es fängt da zu testen an, wo das W3C Schema aufhört. Schematon ist kein W3C Schema. Die Sprache selber besteht aus 5 Elementen, ist aber trotzdem mächtig genug viele Arten von Beschränkungen auszudrücken, die in anderen Sprachen unmöglich sind. Zu den einzelnen Schematron Elementen kommen wir später. Dabei wird besonders viel Wert darauf gelegt, das Regelwerk möglichst in menschlich verständlicher Sprache zu hinterlegen, was der Benutzerfreundlichkeit zu Gute kommt. Es lässt also Business Rules zu, deren Einschränkungen in anderen Sprachen schlecht ausgedrückt wer-

Version	Erscheinung	Neuerung
Schematron 1.0	1999	Erste Version von Rick Jelliffe innerhalb eines privaten Software Projekts.
Schematron 1.3, 1.5, 1.6	Bis 2002	Weiterentwicklungen der Spezifikation und Umsetzung in zahlreichen Implementierungen für unterschiedliche Plattformen
ISO Schematron	2006/05	Aufbauend auf Version 1.6 ISO/IEC-Standard Standard (ISO/IEC 19757-3:2006). Enthält das SVRL Ausgabeformat.

Einblick in die Welt der XMLs

XML (Extensible Markup Language)

Eine von W3C spezifizierte Sprache, um Daten strukturiert in Form von Textdateien plattformunabhängig austauschen zu können.

XSD (XML Schema)

Ist eine Empfehlung des W3C zum Definieren von Strukturen in XML, es handelt sich um eine Schemasprache. Dabei werden viele Datentypen unterstützt, einfache und komplexe. Datentypen können Wertebereiche und Enumeratoren besitzen.

XSL (Extensible Stylesheet Language)

Damit lässt sich das Layout definieren, mit der ein XML File transformiert werden kann um es optisch besser darstellen zu können.

XSLT (XSL Transformation)

Die Subsprache XSLT wird für die Transformation eines XML Formats in ein anderes XML Format verwendet.

XML Path Language (XPath)

Mit dieser von W3C entwickelten Abfragesprache lassen sich XML Dokumente auswerden können. Da die Regeln außerhalb des Codes Teiles in XML Form definiert sind, können diese im Laufe des Produktzyklus leicht an die neuen Business Rules angepasst werden. Schematon eignet sich auch für Ad-hoc Abfragen und Analysen.

SVRL *(Schematron Validation Report Language)*

SVRL ist eine einfache Report Sprache, die ebenfalls in ISO Schematron definiert ist. Es gibt Varianten für XSLT 1 und XSLT 2.

Der Validierungsprozess durchläuft die XML Dokumentenstruktur, prüft dabei die Anwesen- und Abwesenheit von Elementen, die Reihenfolge von Elementen und die Anwesen- und Abwesenheit von Attributen. Es validiert den Dokumenten- und Attributinhalt. Und vor allem die *co-occurrence constraints*, also die Regeln der Felder untereinander. Die Constraints können eine Struktur, Kardinalität und Konformität umfassen, dienen zur Identifikation, können Value Sets definieren, Wertebereiche beschreiben. Weiteres können Einheiten und Genauigkeit von Messwerten damit angegeben werden. Also steht alles in allem damit eine weitreichende Möglichkeit der Prüfung bereit.

Funktionsweise von Schematron

Auf der Schematron Homepage **[1]** erhält man die notwendigen iso-schematronfiles, als XSLT1 oder XSLT2 Variante. In der XSLT Zip Datei „`schematron-skeleton-api.htm`" befindet sich eine kompakte Dokumentation.

Als Transform Tool verwende ich in diesem Beispiel Saxon9 **[7]**, das nach der Installation das Command Line Tool „`Transform.exe`" bereitstellt. **(Abb.1)**

Bei diesen XML Daten soll nun eine Qualitätsprüfung mit Schematron durchgeführt werden: **(Abb.2)**

Abbildung2: Inhalt der Daten im File "„MyData.xml"

```
<?xml version="1.0"?>
- <arcticle>
  - <product id="1">
      <title>Orange</title>
      <state>fresh</state>
      <weight>2.5</weight>
      <country>Austria</country>
    </product>
  - <product id="2">
      <weight>3.5</weight>
      <state>fresh</state>
      <country>Austria</country>
    </product>
  - <product id="3">
      <title>Apricot</title>
      <state>fresh</state>
      <weight>0</weight>
      <country>Austria</country>
    </product>
  - <product id="4">
      <title>Apple</title>
      <state>outdated</state>
      <weight>4</weight>
      <country>Austria</country>
    </product>
  - <product id="5">
      <title>Pear</title>
      <state>fresh</state>
      <weight>4.5</weight>
      <country>Austria</country>
    </product>
  </arcticle>
```

Extensible Markup Language

XSLT 1

Seit 1999 auf dem Markt, wird in .net unterstützt.

XSLT 2

Verfügbar seit 2007, eine überarbeitete Version von XSLT 1.0.

.net unterstützt dies nativ nicht, es müssen zusätzliche Libraries eingebunden werden.

Die Features von XSLT2 gegenüber XSLT ist die Möglichkeit eigene Schema Typen zu definieren, es gibt eine strenge Typisierung und alle XSD Typen stehen zur Verfügung. Mit xsl:function können Funktionen definiert und erstellt werden. Es werden Regular Expression unterstützt und XPath 2.0 hat zahlreiche neue Operatoren.

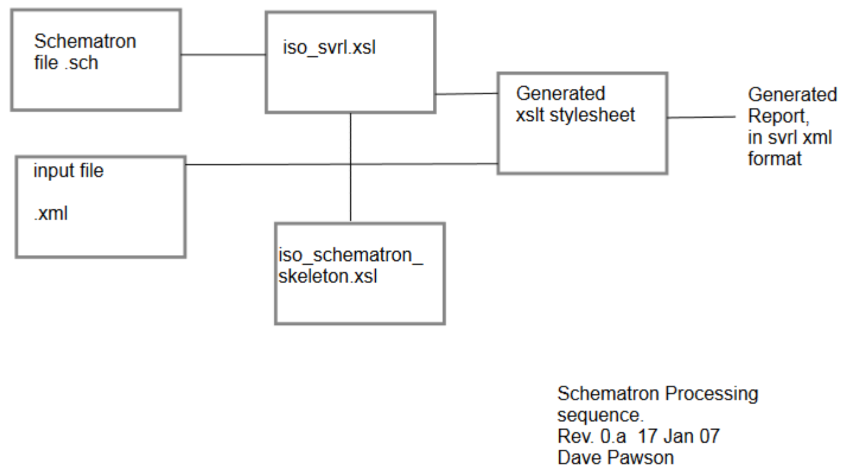


Abbildung 1: Die schematische Funktionsweise des Schematron Prozesses

```
<?xml version="1.0" encoding="utf-8"?>
<iso:schema xmlns="http://purl.oclc.org/dsdl/schematron"
xmlns:iso="http://purl.oclc.org/dsdl/schematron"
queryBinding='xslt2'
schemaVersion='ISO19757-3'>
    <iso:title>Obstkorb Check</iso:title>
    <iso:pattern>
        <iso:rule context="product">
            <iso:assert test="title">
                Das Produkt muss einen Titel haben.
            </iso:assert>
            <iso:assert test="state='fresh'">
                Das Obst muss frisch sein.
            </iso:assert>
            <iso:assert test="weight>'0'">
                Gewichtskontrolle durchführen, Artikel ist leer.
            </iso:assert>
            <iso:assert test="title!='Orange' and country='Austria'">
                In Österreich wachsen keine Orangen.
            </iso:assert>
            <iso:assert test="15 > sum(//weight)">
                Das Paket darf nicht schwerer als 15 KG wiegen.
            </iso:assert>
        </iso:rule>
    </iso:pattern>
</iso:schema>
```

Abbildung3: Inhalt des Schematron Files "„MyCheck.sch"

Die Businessregeln sollen wie folgt lauten:

Die Artikel eines Obstkorbs müssen eine Bezeichnung haben, die Zutaten frisch, keine leeren Tüten, nur Produkte die tatsächlich aus Österreich kommen und das Paket darf am Ende nicht schwerer als 15 KG haben.

Für die Bestimmung der Abfragen wird als Navigation innerhalb der Daten XPath verwenden.

Elemente von Schematron

Die Transformationsaufrufe selber habe ich in einer Batch Datei zusammengefasst:

Das Schematron File „`MyCheck.sch`", in dem alle Businessregeln für die Datenstruktur definiert sind, wird im ersten Aufruf benötigt. Das Resultat anschließend im zweiten Schritt mit dem nächsten Schematron File verarbeitet. Dieses Ergebnis wird mit dem passenden XSLT1 oder

Element	Beschreibung
schema	Das Dokument Element
title	Eine menschlich lesbare Beschreibung über den Zweck der Prüfung
pattern+	Eine Liste von verbunden Regeln, bietet den Context an
rule+	Eine Liste von Assertion zum Context
Assert or report	Deklariert Bedingungen die getestet werden müssen stellt Nachrichten bereit die zurückgegeben

XSLT2 weiterverarbeitet. Die zu prüfenden Daten liegen in der XML Datei MyData.xml vor, die nun an der Reihe ist. Als Ergebnis erhält man die Datei „`result.xml`“.

Sofern sich an den Businessregeln nichts mehr ändert, ist bei der Qualitätsprüfung der Datenfiles nur mehr der letzten Schritt für alle weiteren Files durchzuführen.

Tools

Eine Schematron Prüfung schaffen auch verschieden XML Editoren, dazu zählen etwa oXygen [2] – es ist nativ dazu in der Lage, oder auch XML Spy [3] mit dem XML ValidatorBuddy [4] PlugIn. Von beiden Tools wird eine 30 Tage Testversionen angeboten.

Libraries für .net

nMatrix [5] ist eine .net Implementation unter der Sourceforge Common Public Licence 1.0.

.net unterstützt XSLT 2.0 leider noch immer nicht nativ, Saxon9 hilft dabei. Es stellt eine Command Line als auch eine Library für .net und Java bereit.

Fazit

Oft steht man vor der Frage, wie man komplexe XML Daten vollständig auf ihre Richtigkeit prüft. Implementiert man sein Regelwerk im Code selber oder versucht man die bekannten Regeln in einer Rule Engine abzulegen, wird es nicht weniger komplex. Man hat außerdem auf eine schwer zu wartende, eventuell proprietäre Lösung gesetzt, die zudem keinem Standard entspricht. Ändert sich die Regeln zu häufig, bleibt viel Zeit bei der Wartung liegen. Schematron bietet durch seine ISO Norm einen Standard, auf den bereits viele Bereiche der Industrie setzen. Es ist Plattform unabhängig und es stehen Open Source sowie lizenzpflichtige Tools und Libraries bereit. Man sollte Schematron als Ergänzung aber nicht als Konkurrenz zu anderen Schemasprachen einsetzen, um potentiell die Möglichkeit zu haben, komplexe Datenvalidierungen vorzunehmen.

Links & Quellen

[1] http://www.schematron.com/

[2] http://www.oxygenxml.com/

[3] http://www.altova.com

[4] http://www.xml-tools.com/ValidatorBuddy.htm

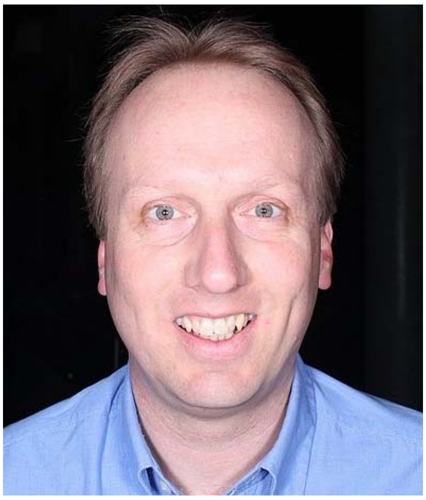
[5] http://sourceforge.net/projects/dotnetopensrc/

[6] http://relaxng.org/

[7] http://saxon.sourceforge.net/

Autorenbox

Thomas Reinwart verfügt über umfangreiche Berufserfahrung auf dem IT Sektor. In den letzten 20 Jahren war er in den Bereichen Softwareentwicklung, Softwaredesign, Architekt und als Consultant tätig. Technischer Fokus ist derzeit Microsoft .net und SQL Server, wo er alle aktuellen Microsoft Zertifizierungen hat.



```
"c:\Program Files\Saxonica\SaxonHE9.5N\bin\Transform.exe" -s:MyCheck.sch -xsl:iso-schematronfiles\iso_dsdl_include.xsl -o:stage1.sch

"c:\Program Files\Saxonica\SaxonHE9.5N\bin\Transform.exe" -s:stage1.sch -xsl:iso-schematronfiles\iso_abstract_expand.xsl -o:stage2.sch

"c:\Program Files\Saxonica\SaxonHE9.5N\bin\Transform.exe" -s:stage2.sch -xsl:iso-schematronfiles\iso_svrl_for_xslt2.xsl -o:stage3.xslt

"c:\Program Files\Saxonica\SaxonHE9.5N\bin\Transform.exe" -s:MyData.xml -xsl:stage3.xslt -o:result.xml
```


```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
- <svrl:schematron-output title="Obstkorb Check" schemaVersion="ISO19757-3" xmlns:xhtml="http://www.w3.org/1999/xhtml"
  xmlns:iso="http://purl.oclc.org/dsdl/schematron" xmlns:schold="http://www.ascc.net/xml/schematron" xmlns:saxon="http://saxon.sf.net/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:svrl="http://purl.oclc.org/dsdl/svrl">
      <!--   -->
    <svrl:active-pattern document="MyData.xml"/>
    <svrl:fired-rule context="product"/>
  - <svrl:failed-assert location="/arcticle[1]/product[1]" test="title!='Orange' and country='Austria'">
      <svrl:text> In Österreich wachsen keine Orangen. </svrl:text>
    </svrl:failed-assert>
    <svrl:fired-rule context="product"/>
  - <svrl:failed-assert location="/arcticle[1]/product[2]" test="title">
      <svrl:text> Das Produkt muss einen Titel haben. </svrl:text>
    </svrl:failed-assert>
  - <svrl:failed-assert location="/arcticle[1]/product[2]" test="title!='Orange' and country='Austria'">
      <svrl:text> In Österreich wachsen keine Orangen. </svrl:text>
    </svrl:failed-assert>
    <svrl:fired-rule context="product"/>
  - <svrl:failed-assert location="/arcticle[1]/product[3]" test="weight>'0'">
      <svrl:text> Gewichtskontrolle durchführen, Artikel ist leer. </svrl:text>
    </svrl:failed-assert>
    <svrl:fired-rule context="product"/>
  - <svrl:failed-assert location="/arcticle[1]/product[4]" test="state='fresh'">
      <svrl:text> Das Obst muss frisch sein. </svrl:text>
    </svrl:failed-assert>
    <svrl:fired-rule context="product"/>
</svrl:schematron-output>
```


Abbildung4: Ergebnis der Schematron Prüfung „result.xml“