Report

Analyse der Metadatenqualität und Interoperabilität

Jürgen Braun

KIM AP 2.2 „Analyse der Metadatenqualität und Interoperabilität“
Stand: Februar 2011

Inhalt

1. Einleitung

Das schnelle und zielgerichtete Auffinden und Selektieren von Ressourcen im Internet erfordert strukturierte Metadaten zur Beschreibung der Ressourcen. Metadaten in diesem Sinne beschreiben die semantischen Eigenschaften von Ressourcen. Im Idealfall entstehen diese Metadaten schon beim Publikationsprozess eines Dokumentes durch den Autor, der die entsprechenden Daten erfasst. Oft werden die Metadaten aber erst nachträglich in Datenbanksystemen mit proprietären Formaten erfasst.

Zudem führen die verschiedenen Anforderungen der einzelnen Fachcommunities und die Notwendigkeit unterschiedliche Objekte zu beschreiben, zu heterogenen Metadatenformaten für spezifische Anwendungen. Diese Vielfalt der verwendeten Formate erschwert die Nachnutzung der Metadaten erheblich.

Bei vielen Informationsangeboten im Netz wie Fachportalen und großen Metadatensammlungen, aber auch für zukünftige Semantic Web Anwendungen und Mash-ups ist die Nachnutzbarkeit von Metadaten eine grundlegende Voraussetzung. Metadaten sind aber nur dann in unterschiedlichen Systemen nutzbar, wenn sie interoperabel sind. Interoperabilität kann durch den Einsatz eines standardisierten Metadatenprofils oder durch entsprechende Mappings auf ein Zielformat erreicht werden. In der Praxis zeigt sich aber, daß Metadaten, selbst wenn sie nach einem standardsierten Metadatenformat wie dem “Dublin Core Metadata Element Set” erstellt wurden, in Syntax und Semantik nicht übereinstimmen. Das liegt zum einen an Fehlern bei der Erfassung der Metadaten, aber auch an fehlenden Regeln zur Erfassung. Beim automatisierten Mapping von Metadatenformaten kann es dann zu semantisch ungenauen oder falschen Zuordnungen kommen.

Für die Nutzung der Metadaten in Suchsystemen ist die Qualität der Daten von entscheidender Bedeutung. Bei syntaktisch oder semantisch fehlerhaften Metadaten ist die Ressource nicht mehr auffindbar. Zur Qualitätssicherung von heterogenen Metadaten ist eine syntaktische und nach Möglichkeit auch semantische Validierung deshalb unabdingbar. Diese Validierung von Metadaten kann mit Hilfe von Software-Tools weitgehend automatisiert werden.

In diesem Bericht wird anhand von Use Cases ein Kriterienkatalog für Software-Tools zur Validierung von Metadaten erstellt. Auf der Grundlage dieses Kriterienkataloges

werden dann einige ausgewählte Tools betrachtet und im Hinblick auf ihre Eignung als Validierungstools für unterschiedliche Metadatenformate untersucht. Schließlich werden auf der konzeptionellen Ebene Funktionen beschrieben, die ein Tool erfüllen muss, um den im Kriterienkatalog genannten Kriterien zu entsprechen.

2. Definition von Use Cases

Im ersten Schritt wurden die verschiedenen Nutzungsmöglichkeiten für die Validierung von Metadaten in Hinblick auf Qualität und Interoperabilität betrachtet. Die Use Cases, die dabei diskutiert wurden (s. Anhang 1), lassen sich unter den folgenden Nutzungsszenarien zusammenfassen.

2.1 Aggregation von heterogenen Metadaten

Die Zusammenführung von heterogenen Metadaten zum Nachweis von Ressourcen aus unterschiedlichen Quellen wird zunehmend wichtiger. Fachportale und große Metadatensammlungen wie ZVDD[1] oder Europeana[2] sammeln Metadaten aus unterschiedlichsten Quellen um sie unter einer Suchoberfläche anzubieten. Diese Anwendungsfälle sind im Anhang 1 unter den Usecases 1,2, und 4 beschrieben.

Die Heterogenität der Metadaten führt zu einer Reihe von Qualitätsproblemen, auf die in diesem Bericht noch näher eingegangen wird. Im Idealfall entsprechen solche Daten einem definierten und weit verbreiteten Anwendungsprofil und lassen sich somit relativ leicht auf andere Zielformate mappen. In der Realität handelt es sich meist um Daten mit sehr unterschiedlicher Qualität und in proprietären Formaten. Um solche Daten in *einem System* anzubieten, ist eine vorhergehende Analyse der Qualität der Metadaten unumgänglich. Die Daten müssen zum einen auf die Übereinstimmung mit einem Anwendungsprofil (sofern vorhanden) überprüft werden, und zum anderen muss die Konsistenz der Daten kontrolliert werden (im Hinblick auf Eingabefehler, Datentypen, Syntax und semantische Plausibilität). Diese Prüfung sollte im Idealfall bereits durch den Datenlieferanten selbst erfolgen, in der Praxis wird sie aber oft erst vom Serviceprovider vorgenommen.

1 http://www.zvdd.de/

2 http://www.europeana.eu/

2.2 Metadaten werden von den Autoren selbst erstellt

Bei vielen Web-Publikationen erfolgt die Eingabe der Metadaten durch den Autor selbst über Autorenwerkzeuge oder Erfassungsformulare. Im Anhang 1 wird dieser Fall unter Use Case 3 aufgeführt. Idealerweise werden die Metadaten schon bei der Erfassung kontrolliert. Wenn diese Prüfroutinen bei der Eingabe nicht ausreichend sind, müssen die Metadaten vor der Übernahme in ein Dokumenten-Management-System (z. B. Repository) validiert werden.

2.3 Prüfung des Ergebnisses einer Datenkonversion (Use Case 5)

In vielen Fällen ist es notwendig, Metadaten aus anderen Systemen auf ein eigenes Metadatenprofil zu mappen. Dieser Fall wird als Use Case 5 im Anhang 1 beschrieben. Eine umfassende qualitative und quantitative Beurteilung des Ergebnisses einer Konversion lässt sich nur mit maschinellen Prüfroutinen erreichen. Bei einer stichprobenartigen Überprüfung allein würden viele Probleme möglicherweise gar nicht erkannt werden. Das Ergebnis der Konversion muß auf syntaktische und semantische Übereinstimmung mit dem Zielformat überprüft werden.

3. Erstellung eines Kriterienkatalogs für Validierungstools

Auf der Grundlage der oben genannten Use Cases wurde ein Kriterienkatalog erarbeitet. Diese Kriterien dienten dazu, bereits bestehende Software-Tools für die Validierung von Metadaten auf ihre Tauglichkeit in den von uns definierten Use Cases zu untersuchen. Ausgangspunkt für die Erstellung des Katalogs waren neben grundsätzlichen Anforderungen auch die Forderung nach offenen Standards und Schnittstellen, die die plattformunabhängige Nutzung sicherstellen.

3.1 Grundsätzliche Anforderungen an die Validierungsroutine

Ausgehend von den Use Cases ergeben sich die folgenden grundlegenden Anforderungen an ein Validierungs-Tool:

Verpflichtend

- Die syntaktische und semantische Validierung von Metadaten anhand eines XML Schemas oder Application Profiles muss möglich sein
- Das Validierungs-Tool muss für unterschiedliche Metadatenprofile einsetzbar sein.
- Unterstützung von unterschiedlichen Datenformaten (XML, Daten in Tabellenform, ISO 2709 Daten).
- Unterstützung und Validierung von wichtigen Zeichensätzen (UNICODE, UTF8, ISO Zeichensätze)
- Statistikfunktionen zur Auswertung der Validierungsergebnisse

Optional

- Es sollte möglich sein, unbekannte Datenformate für die Validierung einzulesen.

Bei der Validierung von Metadaten muss zwischen der syntaktischen und der semantischen Ebene unterschieden werden. Liegen die Daten in XML vor, kann die Validierung der Syntax mithilfe eines XML Schemas[3] oder einer DTD automatisiert werden. Mit diesen Werkzeugen kann überprüft werden, ob alle verwendeten Datenelemente gültig sind, wie oft die einzelnen Elemente vorkommen dürfen und ob alle verpflichtenden Elemente vorhanden sind. Mit einem XML Schema können darüber hinaus die Gültigkeit des Zeichensatzes und die zulässigen Datentypen überprüft werden.

Mit der Validierung der zulässigen Datentypen kann eine Kontrolle des Inhalts einzelner Elemente erfolgen. So kann z. B. mit den sogenannten Facets[4] in XML Schema genau festgelegt werden, welche Zeichen ein bestimmtes Element enthalten darf.

Die Semantik vieler Metadatenelemente wie z.B. des Titels einer Ressource kann mit den Möglichkeiten von XML Schema nicht automatisch validiert werden.

[3] http://www.w3.org/TR/xmlschema-0/
[4] http://www.w3.org/TR/xmlschema-2/#facets

Deshalb sollte ein Validierungs-Tool die Visualisierung der Inhalte von einzelnen Datenelementen anbieten und damit eine zusätzliche manuelle Überprüfung von einzelnen, nichtkontrollierten Elementen ermöglichen.

Bei Datenelementen die kontrolliertes Vokabular enthalten (wie Schlagwortnormdateien, Thesauri, Klassifikationen) kann ein maschineller Abgleich mit den zugelassenen Werten erfolgen.

Da eine Vielzahl von Metadatenformaten existieren, muss das Tool in der Lage sein, unterschiedlichste Metadatenformate einzulesen und zu validieren. Diese Forderung ist am schwierigsten zu realisieren, da es neben den XML basierten Formaten noch weitere Datenformate für Metadaten gibt. Hier ist vor allem das im bibliothekarischen Bereich noch weit verbreitete ISO 2709[5] Format und das bei Microsoft Excel und Datenbankschnittstellen verwendete csv Format zu nennen. Hinzu kommt die Problematik, unbekannte oder fehlerhafte Formate automatisch zu verarbeiten, wenn die Syntax nicht exakt eingehalten wurde. In der Praxis zeigt sich, dass syntaktische Formatfehler oft vorkommen. Nicht selten kann ein fehlerhaftes Format dann von der Software überhaupt nicht eingelesen werden. Ein Validierungstool muss insoweit fehlertolerant sein, als auch syntaktisch fehlerhafte Formate eingelesen werden können. Eine weitere wichtige Funktion eines Validierungs-Tools sind die statistische Auswertung und die graphische Visualisierung der Validierungsergebnisse. Vor allem bei umfangreichen Metadatensammlungen bietet die graphische Darstellung von Häufigkeiten einen schnellen Überblick. Sie erlaubt erste Rückschlüsse auf die Qualität der Metadaten und hilft, Fehlerquellen zu identifizieren. So kann z. B. überprüft werden, wie vollständig die Daten sind und ob ggf. vorgeschriebene Pflichtfelder in jedem Datensatz vorhanden sind.

3.2. Verwendung von offenen Standards und Schnittstellen

Durch die Verwendung von offenen Standards bei der Entwicklung des Validierungs-Tools ist die plattformunabhängige Nutzbarkeit des Tools sicherzustellen. Die Verwendung von XML erlaubt den Einsatz von existierenden Software-Werkzeugen und Schema Sprachen. Auf die vom W3C veröffentlichte XML Schema Sprache und davon abgeleitete Sprachen wie RELAX[6] und Schematron[7] wird im nächsten Kapitel

5 http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=41319

6 http://relaxng.org/

näher eingegangen. OAI[8] und SRU[9] sind wichtige Standards für den Import der Daten und sollten deshalb bei der Entwicklung des Tools berücksichtigt werden.

3.3 Weitere technische Anforderungen

Bei der Zusammenstellung der folgenden Kriterien wurde vor allem die Plattformunabhängigkeit und die Offenheit der Systeme berücksichtigt, um sicherzustellen, dass es möglichst keine herstellerspezifischen Einschränkungen in der Anwendung der Software gibt. Für die generische Nutzung muss es möglich sein, den Anwendungsbereich des Tools bei Bedarf zu erweitern, ohne den Quellcode zu verändern. Sehr wichtig erscheint auch die Möglichkeit, die Software als Modul in bestehende oder zukünftige Anwendungen integrieren zu können. Auch die Verwendung von offenen Standards und die Performance und Skalierbarkeit bei großen Datensammlungen sind wichtige Punkte. Das System sollte möglichst fehlertolerant sein. Aus der Sicht des Anwenders sind eine gute Dokumentation und ein möglichst geringer Einarbeitungsaufwand entscheidende Kriterien um die Kosten für die Nutzung niedrig zu halten.

Plattformunabhängigkeit

- Ist die Software plattformunabhängig?
- Läuft die Software lokal oder auf einem Server? Handelt es sich um eine serverseitige Anwendung oder ein Stand-Alone Programm ?
- Welche Betriebssysteme werden unterstützt, falls nicht plattformunabhängig?

Offenheit

- Handelt es sich um Open Source Software?

Integration in bestehende Systeme

- Kann das Programm serverseitig eingebunden werden?
- Ist ein Web-Interface vorhanden, bzw. eine Schnittstelle für die Anbindung eines Web-Interface?

---

7 http://www.schematron.com
8 http://www.openarchives.org/
9 http://www.d-nb.de/service/zd/sru.htm

Flexibilität

- Kann das Programm an neue Anforderungen angepasst werden?
- Ist die Integration neuer Module möglich?
- Ist es möglich Validierungsroutinen für neue Metadatenprofile einzubinden, ohne dass eine Änderung des Quellcodes notwendig ist ?

Performance

- Wie sind die Antwortzeiten der Anwendung bei größeren Dateien?
- Wie stabil läuft die Anwendung?

Skalierbarkeit

- Können auch große Dateien validiert werden?

Fehlertoleranz

- Werden bei Falscheingaben und Fehlbedienung verständliche Fehlermeldungen ausgegeben?
- Wie stabil läuft die Software bei Bedienungsfehlern?

Dokumentation

- Gibt es eine online verfügbare Dokumentation?
- Ist die Dokumentation vollständig und verständlich?

Aufwand bei der Einführung der Software

- Wie hoch ist der Einarbeitungs- bzw. Schulungsaufwand?
- Wie hoch ist der Aufwand für die Inbetriebnahme?

4. Untersuchung von Validierungs-Tools

Im Rahmen des Arbeitspakets wurden 5 verschiedene Software-Tools im Hinblick auf den oben erläuterten Kriterienkatalog untersucht. Es wurden bewusst Tools mit unterschiedlichen Ansätzen ausgesucht, um die Stärken und Schwächen der jeweiligen Konzepte vergleichen zu können.

4.1 vascoda Checker Tool

Das vascoda Checker-Tool wurde zur Validierung von Metadaten im vascoda Projekt an der SUB Göttingen entwickelt. Das Tool kann ausschliesslich Daten, die nach dem vascoda Application Profile[10] erfasst wurden, validieren. Ohne eine Modifikation des Quellcodes können andere Metadatenformate damit nicht validiert werden. Der Anwendungsbereich des Tools ist damit sehr eng auf den vascoda Kontext begrenzt. Selbst bei kleinen Änderungen des Application Profiles müsste der Quellcode entsprechend angepasst werden.

Der vascoda Checker ist an die vascoda Plattform gebunden und kann nur unter großem Aufwand in andere Systeme integriert werden. Entsprechend gering ist die Flexibilität des Tools. Selbst die Anpassung an Weiterentwicklungen des vascoda Application Profiles erfordert umfangreichen Programmieraufwand.

Die Performance des Tools ist bei Verarbeitung kleiner Datenmengen recht gut, handelt es sich um große Datenmengen, sind die Antwortzeiten allerdings lang.

Bedienungsfehler werden durch verständliche Fehlermeldungen abgefangen.

Die Dokumentation ist kurz und leicht verständlich, der Einarbeitungsaufwand ist gering.

Das Tool liefert Fehlermeldungen bei Syntax-Fehlern und Warnungen bei eventuellen semantischen Fehlern (z. B. wird bei Titeln mit weniger als fünf Zeichen eine Warnung ausgegeben). Warnungen müssen daraufhin intellektuell geprüft werden, ob es sich tatsächlich um Fehler handelt.

Zusätzlich wird eine graphische Übersicht über die verwendeten Metadatenelemente erzeugt. Es gibt keine Möglichkeit benutzerdefinierte Indizes zu erstellen.

Zusammenfassend lässt sich sagen, dass der vascoda Checker zwar einfach zu benutzen ist, aufgrund der geringen Flexibilität aber für die von uns benannten Use Cases nicht verwendbar ist.

Da das vascoda Portal mittlerweile offline[11] ist, ist auch das vascoda Checker Tool nicht mehr verfügbar.

10 http://www.dl-forum.pt-dlr.de/dateien/vascoda_AP_1.0_vorb.pdf
11 http://vascoda.wordpress.com/2011/01/14/vascoda-portal-offline/

4.2 Falcon

Falcon ist eine Analyse- und Konvertierungssoftware, die auf der Allegro-C Exportsprache basiert. Falcon wurde für Konvertierungs- und Validierungsaufgaben an der Verbundzentrale des GBV in Göttingen entwickelt.

Es handelt sich um ein Tool, das Daten, die in einer Feldstruktur vorliegen (ISO 2709, Textdateien), einlesen kann, nachdem eine entsprechende maschinenlesbare Formatbeschreibung erstellt wurde. XML Daten können zwar eingelesen werden, die XML Syntax wird aber nicht explizit unterstützt, da Falcon nur Daten in Feldstruktur unterstützt und deshalb nur 2 Hierarchieebenen darstellen kann.

In Falcon ist die Anzeige unterschiedlicher Zeichensätze möglich. Die Software erstellt automatisch eine statistische Auswertung der verwendeten Datenelemente. Die Analyse von unbekannten Formaten ist damit möglich.

Falcon ist kein Open Source Produkt, eigene Erweiterungen oder Anpassungen sind deshalb nicht möglich. Die Software ist nicht plattformunabhängig, sie kann aber in serverseitige Anwendungen integriert werden.

Benutzerdefinierte Indizes können leicht erstellt werden, auch für größere Dateien ist die Verarbeitungsgeschwindigkeit gut. Bei fehlerhafter Bedienung kann es zum Programmabsturz kommen.

Mit Falcon sind durch die Verwendung einer Parametrierungssprache komplexe Prüfroutinen möglich, die die Formulierung von Constraints erlauben. Die Einarbeitung ist aber aufgrund des generischen Ansatzes relativ aufwendig. Es gibt eine (allerdings unvollständige) Dokumentation.

Falcon eignet sich gut für die Validierung unterschiedlichster Formate, die in Feldern strukturiert sind. Für XML oder RDF Daten ist Falcon jedoch nur bedingt geeignet, da nicht mehr als 2 Hierarchieebenen abgebildet werden können.

4.3 Schematron

Schematron[12] ist eine Schema-Sprache zur Validierung von XML-Daten, die auf XPATH und XSLT Sprachelementen basiert. Im Unterschied zur grammatikbasierten Validierung mit Schema-Sprachen wie XML- Schema oder Relax erlaubt Schematron

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

die Formulierung von Regeln zur Validierung. Es wird keine formale Grammatik zur Validierung verwendet, sondern nach Mustern im XML Dokument gesucht. Dadurch ist es möglich, komplexe Regeln zur Validierung von Metadaten zu erstellen. Mit Hilfe von XPATH und XSLT können einzelne Elemente adressiert und überprüft werden.

Mit Schematron sind folgende Validierungen möglich:

1. *Cardinality*: Prüfung, ob bestimmte Elemente oder Werte vorhanden sind
2. *Co-constraints*: Abhängigkeiten zwischen verschiedenen Elementen können überprüft werden
3. *Algorithmic processing of data*: Formulierung von komplexen Algorithmen z.B. für die Berechnung von zulässigen Werten.

Schematron ermöglicht außerdem die freie Formulierung von Fehlermeldungen. Die Schematron Regeln können mit einer grammatikbasierten Schemasprache kombiniert werden (XML Schema oder RELAX), sie können aber auch separat verwendet werden.

Das Format des Schematron Reports kann durch entsprechende Stylesheets bestimmt werden, z.B. in HTML oder XML. Es gibt fertige Stylesheets für die Erstellung von Fehlerreports, die nach den eigenen Bedürfnissen angepasst werden können. Die Fehlermeldungen können auch mit anderen Anwendungen weiterverarbeitet werden.

Für die Validierung mit Schematron ist ein XML Prozessor nötig (z.B. Saxon), der die XML Metadaten gegen das Schematron Stylesheet validiert. Die Validierung kann somit über einen Batch oder innerhalb einer Anwendung aufgerufen werden. Es gibt aber auch XML Editoren, die Schematron unterstützen, z.B. Oxygen. Allerdings lassen sich damit nur kleinere Dateien validieren, da zur Validierung die gesamte Datei eingelesen werden muss. Mit ereignisorientierten XML Prozessoren wie Saxon können jedoch auch große Dateien verarbeitet werden.

Schematron ist ein ISO Standard und kann in verschiedenen Umgebungen implementiert werden (Java, C++, Perl, NET, Batch files, XProc, ANT). Schematron

unterstützt XSLT1 und XSLT2. Für die Schematron-Skeleton Implementierung gibt es eine API[13].

Die Performance und die Fehlertoleranz hängt stark von dem eingesetzten XML Prozessor und der Softwareumgebung ab. Deshalb kann darüber keine allgemeine Aussage gemacht werden. Schematron ist gut dokumentiert, der Einarbeitungsaufwand ist bei vorhandenen XML Kenntnissen relativ gering.

Schematron ist aufgrund seiner Flexibilität, der Plattformunabhängigkeit und der vielseitigen Einsatzmöglichkeiten gut für die definierten Use Cases geeignet. Durch die offene Konzeption und die Verwendung von XPATH und XSLT erscheint dieser Ansatz auch zukunftssicher.

Allerdings kann Schematron nur XML Daten validieren, andere Datenformate (ISO 2709, Tabellen) müssen für die Validierung zuerst in XML umgewandelt werden.

4.4 Talend Open-Profiler

Der Talend Open-Profiler[14] ist ein Code Generator zur Entwicklung von Validierungs- und Konversionsroutinen. Talend Open Data Solutions bietet Open Source Software zur Datenintegration an und wurde 2005 in Suresnes (Frankreich) gegründet. Talend unterhält Niederlassungen in Los Altos (USA), Lüttich (Belgien) und Nürnberg.

Es gibt vorgefertigte Routinen für die Validierung von XML Daten und Daten in Tabellenform. Der direkte Zugriff auf MySQL und andere Datenbanken ist möglich, die Daten können aber auch offline eingelesen werden.

Die Software ist vor allem für die Qualitätsprüfung von Geschäftsdaten in Tabellenform konzipiert. Für das Einlesen von XML Daten muss zuerst die XML Struktur erfasst werden. XML Schema Dateien werden nicht unterstützt, aber es ist möglich Regeln für die Validierung von XML Daten zu formulieren.

---

13 http://www.schematron.com/schematron-skeleton-api.htm

14 http://www.talend.com/products-data-quality/talend-open-profiler.php

Die Verarbeitung von großen Datenmengen ist problematisch, da die komplette Datei zuerst eingelesen werden muss. Größere Dateien müssen zuerst gesplittet und dann sukzessiv abgearbeitet werden, was relativ aufwendig ist.

UTF-8 Zeichensätze werden unterstützt. Nichtrelationale Daten müssen zuerst in eine Tabellenform gebracht werden, bevor sie validiert werden können.

Auch komplexe statistische Auswertungen zur Analyse der Qualität der Daten sind möglich. Mit dem Open-Profiler können Fehlerreports erstellt und mit Stylesheets angepasst werden. Die Ausgabe der Reports kann in verschiedenen Formaten erfolgen (PDF, HTML, XML). Mit dem Code Generator kann das Tool durch benutzerdefinierte Routinen erweitert werden. Der Code Generator erlaubt auch die Formulierung von komplexen Prüfroutinen. Das Tool ist nicht plattformunabhängig, es kann nur unter Windows und Linux eingesetzt werden. Durch Schnittstellen zu verschiedenen Datenbanksystemen (Oracle, Microsoft Access, Microsoft SQL, IBM DB2, MySQL, PostgreSQL) und die Möglichkeit der Einbindung von Web Services ist die Software in komplexere Anwendungen integrierbar.

Bedienungsfehler werden durch entsprechende Fehlermeldungen abgefangen.

Das Tool ist Teil des umfangreichen Softwarepaktes Talend Open Studio. Es gibt umfangreiche Dokumentationen und zahlreiche Online Tutorials, die die Einsatzmöglichkeiten in Schritt für Schritt Anleitungen erläutern.

Trotz der umfangreichen Möglichkeiten, die der Code Generator bietet, erscheint das Tool für die oben beschriebenen Use Cases weniger gut geeignet, da XML Schema nicht unterstützt wird und die Verarbeitung von größeren Dateien problematisch ist.

4.5 Spotfire

Spotfire[15] ist eine kommerzielle Software zur statistischen Auswertung von großen Datenmengen. Spotfire ist ein Produkt der kalifornischen Softwarefirma TIBCO, die vor allem Softwarelösungen für Unternehmen anbietet. Spotfire ist für die Auswertung und Visualisierung von Unternehmensdaten konzipiert. Das Tool bietet mit einer tabellarischen und grafischen Anzeige einen schnellen Überblick über das Vorkommen von einzelnen Metadatenelementen. Das Tool ist nicht generisch, es ist

15 http://spotfire.tibco.de/

nur die Auswertung von tabellarischen Daten möglich. XML Daten müssen deshalb zuerst in Tabellen umgewandelt werden.

Die Analyse von anderen Formaten und maschinelle Prüfroutinen sind nicht möglich. Die Stärke des Tools sind die statistischen Funktionen zur Auswertung von Metadaten. Fehlerreports können damit nicht erstellt werden. Da es sich um eine kommerzielle Software handelt, sind eigene Erweiterungen nicht möglich.

Die National Science Digital Library (NSDL)[16] setzt das Tool zur Validierung von geharvesteten Metadaten ein[17].

Die Software ist nur ein Hilfswerkzeug für die manuelle Validierung von Metadaten, sie ermöglicht keine automatisierte Validierung der Syntax von Metadaten.

Das Tool ist einfach zu bedienen und es gibt eine umfangreiche englischsprachige Dokumentation zu den Anwendungsmöglichkeiten.

Für die manuelle Validierung von großen Datensammlungen kann Spotfire ergänzend eingesetzt werden, die Software hat aber den Nachteil, dass damit nur Daten in Tabellenform verarbeitet werden können. Im Hinblick auf die oben formulierten Anforderungen erscheint das Tool für die definierten Anwendungsfälle deshalb nur von begrenztem Nutzen.

4.6 Tabelle Kriterienkatalog

Kriterien	vascoda Tool	Falcon	Schematron	Talend Open Profiler	Spotfire
Syntaktische Validierung möglich	Ja	Ja	Ja	Ja	Nur manuell
Generisches Tool	Nein	Ja	nur XML	Nein	Nein
Unterstützung unterschiedlicher Formate	Nein	Ja	Ja, XML	Ja, XML	Nein
Komplexe Prüfroutinen möglich	Ja	Ja	Ja	Ja	Nein

16 http://nsdl.org/
17 dcpapers.dublincore.org/ojs/pubs/article/download/744/740

Schema Unterstützung	Nein	Nein	Ja	Nein	Nein
Unterstützung von verschiedenen Zeichensätzen	Ja	Ja	Ja	Ja	-
Analyse von unbekannten Datenformaten möglich	Nein	Ja	Nein	Nur in Tabellenform	Nein
Statistische Auswertung der Datenelemente	Ja	Ja	Ja, mit Stylesheets	Ja	Ja
Verwendung von offenen Standards und Schnittstellen	Nein	Nein	Ja	Ja	Nein
Erstellung von Fehlerreports	Ja	Ja	Ja	Ja	Nein
Erstellung von benutzerdefinierten Indizes	Nein	Ja	Ja	Ja	-
Plattformunabhängig	Nein	Nein	Ja	Nein	Nein
Offenheit	Nein	Nein	Ja	Ja	Nein
Integrationsfähigkeit	Nein	Nein	Ja	Ja	Nein
Flexibilität	Nein	Ja	Ja	Ja	Nein
Performance	Gut	Gut	Gut	Weniger gut bei großen Dateien	Gut
Fehlertoleranz	Gut	Weniger gut	Weniger gut	Weniger gut	-
Skalierbarkeit	Gut	Gut	Gut	Weniger gut	-
Dokumentation	Nicht mehr vorhanden	Unvollständig, englisch	Tutorials und Spezifikationen in englisch	Umfangreiche Anleitungen in englisch	Umfangreiche Anleitungen in englisch
Einarbeitungs- bzw Schulungsaufwand	Gering	Mittel, für komplexe	Bei XSL Kenntnissen	Mittel	Gering

		Routinen höher	relativ gering, sonst höher		

4.8 Zusammenfassung der Ergebnisse

Wie der Tabelle zu entnehmen ist, erfüllen Schematron und der Talend Open Profiler die gestellten Anforderungen weitgehend. Das offene Konzept von Schematron ist erweiterbar und verwendet mit XPATH und XSLT Standards, die beständig weiterentwickelt werden und zukunftssicher sind. Für die Validierung von kleineren Datensammlungen kommt aber auch der Talend Open Profiler in Betracht. Der Talend Open Profiler kann XML Daten, Tabellen und Daten aus relationalen Datenbanken validieren. XML Schema wird vom Talend Open Profiler jedoch nicht unterstützt.

Aber auch bei Schematron gibt es Einschränkungen im Hinblick auf die generische Verwendung für heterogene Metadatenformate. Da Schematron nur XML Daten validieren kann, müssen Metadaten, die nicht in XML vorliegen, zuerst in XML umgewandelt werden. Vorhandene Schema Validierungen können in Schematron eingebunden werden.

Zur Analyse und Validierung von Metadatenformaten, die in strukturierten Feldern vorliegen, ist besonders Falcon gut geeignet.

5. Anwendungsfall ZVDD Daten

Um die Leistungsfähigkeit von XML Schema und Schematron zu testen, wurden im ZVDD[18] Kontext gelieferte Metadaten im MODS[19] Format validiert. Die Daten wurden über die OAI Schnittstelle geharvestet und liegen in XML vor. Es wurde eine Testmenge von 1197 Datensätzen überprüft.

Die Testdaten waren bereits manuell auf Fehler überprüft. Art und Umfang der Fehler waren somit weitgehend bekannt (vgl. 5.2). Ziel des Tests war es, festzustellen, wie

[18] http://www.zvdd.de
[19] http://www.loc.gov/standards/mods/

gründlich und effizient die Validierung mit dem von der Library of Congress erstellten MODS Schema[20] und mit einem entsprechenden Schematron Stylesheet ist.

5.1 Validierung mit dem MODS Schema der Library of Congress

Das MODS Schema wird in den Beispieldaten unter schemaLocation über die URL referenziert. Die Validierung wurde mit dem XML Editor Oxygen[21] durchgeführt.

Dabei hat der XML-Prozessor aber nur eine einzige Fehlerart erkannt:

Ungültige Attributwerte z.B. Attribut ‘own’ in ‘mods:title’.

Bei der vorangegangenen Validierung dieser Daten mit zusätzlicher manueller Überprüfung wurden jedoch weitaus mehr Fehler entdeckt, die unter 5.2 aufgelistet sind.

Die ausschließliche Validierung mit dem vorhandenen MODS Schema der LOC hat sich für die Testdaten als unzureichend erwiesen, da das ZVDD MODS Application Profile[22] sehr viel restriktiver ist als das Schema der LOC. Das Schema der LOC ist sehr offen gehalten und hat nur sehr wenige Restriktionen. Für die Validierung der ZVDD Daten müsste deshalb das MODS Schema modifiziert werden, oder ein eigenes Schema erstellt werden.

5.2 Nicht erkannte Fehlerarten

Die folgenden Fehlerarten wurden bei der manuellen Überprüfung der Testdaten gefunden, nicht aber bei der maschinellen Validierung mit dem MODS Schema. Die Fehlermeldungen sind kursiv gedruckt.

Elementinhalte sind semantisch nicht korrekt

- *Körperschaften werden teilw. in mods:name[@type='person'] aufgeführt*
- *mehrere Namen in einem Feld*
- *Name ist nicht invertiert*
- *mods:namePart[@type='family'] enthält Vorname und Nachname*
- *Vorname enthält die Wendung "u.a."*
- *mods:placeTerm enthält mehr als einen Ortsnamen*

[20] http://www.loc.gov/standards/mods/mods.xsd
[21] http://www.oxygenxml.com/
[22] http://www.zvdd.de/fileadmin/AGSDD-Redaktion/zvdd_MODS_Application_Profile_2008-11-13.pdf

- *mods:placeTerm enthält einen Namen, der kein Ortsname ist*
- *ungültiger Sprachcode*
- *Namen enthalten bisweilen Funktionsbezeichnung; teilw. im Widerspruch zum angegebenen Relatorterm*

Diese Fehler auf der semantischen Ebene können nur durch einen Vergleich mit entsprechenden Normdaten erkannt werden. Die Validierung der Syntax mit XML Schema ist hier nicht ausreichend. Schematron dagegen bietet die Möglichkeit eines Abgleichs mit entsprechenden Normdaten.

Der Datentyp von Elementen ist ungültig

- *Datenelement enthält eine öffnende Winkelklammer ohne schließende Winkelklammer*
- *ungültiger Integer Wert in Attribut order*
- *Das Datumsformat entspricht nicht 'w3cdtf', obwohl im Attribut encoding so angegeben*
- *Identifier enthalten Leerzeichen*

Solche Fehler können durch Überprüfung des Datentyps mit Facetten erkannt werden (XML Schema) oder auch alternativ durch einen Mustervergleich mit Schematron.

Pflichtelemente

- *`Element kommt nicht vor, obwohl Pflicht`*

Die Überprüfung auf verpflichtende Metadaten-Elemente anhand eines Application Profiles kann mit XML Schema automatisiert werden. Für die vollständige Validierung auf zulässige Elemente ist XML Schema am besten geeignet, da bei der Validierung für alle nicht definierten Elemente eine Fehlermeldung ausgegeben wird.

- *`mods:titleInfo Element ist leer`*

Die Prüfung auf leere Elemente kann mit XML Schema oder Schematron durchgeführt werden.

- *`mods:relatedItem[@type='host'] enthält nicht das vorgeschriebene Element`*
- *`mods:relatedItem[@type='series'] enthält kein mods:part Element`*
- *`mods:originInfo/mods:place fehlt in einigen Sätzen, obwohl es sich nicht um Bandsätze handelt`*

Elemente, deren Verpflichtungsgrad von anderen Elementen oder Attributwerten abhängt, können nur mit Schematron überprüft werden.

Identifier nicht eindeutig

- *Identifier in einigen Datensätzen nicht eindeutig, da sie mehrfach vorkommen.*

Diese Fehlerart kann durch die Verwendung der Funktion „Count“ mit Schematron kontrolliert werden.

Orthografische Fehler

- *Tippfehler im displayLabel Attribut: (More informations)*

Wenn es sich um nicht kontrolliertes Vokabular wie in diesem Fall handelt, sind orthografische Fehler nur durch manuelle Überprüfung erkennbar.

5.3 Regelbasierte Validierung mit Schematron

Für die Schematron Validierung wurde ein Schematron Stylesheet mit den Valdierungsregeln erstellt[23]. Diese Regeln wurden auf der Grundlage des ZVDD MODS Anwendungsprofils[24] erstellt. Das Schematron Stylesheet wurde nur zu Testzwecken erstellt und erhebt keinen Anspruch auf Vollständigkeit.

Mit der Schematron Validierung wurden folgende Regeln überprüft:

- Pflichtelemente müssen vorhanden sein
- Wenn bestimmte Eltern-Elemente vorhanden sind, müssen auch die zugehörigen Kind-Elemente vorhanden sein
- Zulässige Elemente in Abhängigkeit der Attributwerte
- Elemente dürfen nicht leer sein
- Encoding Schemes müssen zugelassen sein

[23] s. Anhang 2: mods-schema.sch

- Das Element mods:roleTerms darf nur gültige Werte enthalten
- Funktionsbezeichnungen in mods:name müssen mit dem Relatorterm übereinstimmen
- mods:namePart darf nicht mehrere Namen enthalten
- es dürfen keine HTML Sonderzeichen enthalten sein
- das Element mods:placeTerm darf nicht mehrere Ortsnamen enthalten
- das Element mods:languageTerm darf nur gültige Sprachcodes nach ISO 639-2b enthalten
- das element recordIdentifier darf keine Leerzeichen enthalten.

Die Validierung ergab, dass mit der Kombination von XML Schema und einer zusätzlichen Schematron Validierung fast alle vorkommenden Fehler erkannt werden konnten. Orthografische Fehler in Elementen mit nicht kontrolliertem Vokabular ließen sich jedoch nur durch manuelle Überprüfung finden. Für die Überprüfung auf die zulässigen Datenelemente eignet sich das XML Schema mit dem grammatikbasierten Ansatz besser als Schematron. Für die Überprüfung von Abhängigkeiten zwischen Elementen und Attributen ist der regelbasierte Ansatz von Schematron besser geeignet. Das MODS Schema der Library of Congress ist jedoch nicht ausreichend für eine vollständige Validierung, da es zu offen konzipiert ist.

Die Validierung mit Falcon hat gegenüber der Schema Validierung den Nachteil, daß die XML Daten nicht direkt eingelesen werden können, da XML nicht unterstützt wird. Bei komplexeren XML Strukturen kann der Aufwand, der für das Einlesen der Daten nötig ist, sehr hoch werden. Zudem können vorhandene XML Schemata nicht eingebunden werden. Beim Talend Open Profiler wird XML zwar unterstützt, aber XML Schemata können ebenfalls nicht direkt eingebunden werden. Aus Effizienzgründen erscheint deshalb der Ansatz mit XML Schema und Schematron am erfolgversprechendsten, da vorhandene Schemata nachgenutzt werden können.

[24] http://www.zvdd.de/fileadmin/AGSDD-Redaktion/zvdd_MODS_Application_Profile_2008-11-13.pdf

6. Konzept für eine Validierungssoftware auf XML-Basis

Wie bereits ausgeführt, entspricht die Kombination von XML Schema mit Schematron noch am ehesten den von uns erarbeiteten Kriterien. Doch auch dieses Tool erfüllt nicht alle genannten Kriterien. Insbesondere dadurch, dass die Daten in XML vorliegen müssen, sind die Möglichkeiten, das Tool generisch einzusetzen, begrenzt. Im Folgenden soll ein Konzept für die Gestaltung eines Validierungstools entworfen werden, das die von uns genannten Kriterien erfüllt.

Dieses Konzept verbindet die Möglichkeiten von Schema Sprachen wie XML Schema oder RELAX mit der regelbasierten Validierung von Schematron, wobei Daten, die nicht in XML vorliegen, zunächst in XML umgewandelt werden müssen. Ausgehend von den oben formulierten Anforderungen werden im Folgenden die dazu notwendigen Funktionen beschrieben.

6.1 Funktionen

6.1.1 Parsen der Metadaten (Prüfen auf Wohlgeformtheit)

In einem ersten Schritt sollte es möglich sein, beliebige Metadatenformate einzulesen und zu parsen. Die Realisierung dieser Anforderung ist für Metadaten in XML kein Problem, da entsprechende Parser als Open Source Software zur Verfügung stehen. Bei anderen Datenformaten (z.B. Daten im ISO 2709 Format oder Tabellen) ist das nicht unbedingt der Fall. Um auch Daten, die nicht in XML vorliegen, validieren zu können sind folgende Ansätze denkbar:

- Die Software ermöglicht die manuelle Eingabe einer maschinenlesbaren Formatbeschreibung für beliebige Formate. Diese Beschreibung muss nur einmal für ein bestimmtes Format erfasst werden und kann dann wiederverwendet werden. Dazu muss natürlich bekannt sein, wie die Daten strukturiert sind. Anhand der Formatbeschreibung könnten dann XML Daten für den weiteren Validierungsprozess generiert werden.
- Die Software erkennt die Dateistrukturen wie z.B. ISO 2709 oder csv Formate schon beim Einlesen und erzeugt daraus validierbare XML Daten.

Ideal erscheint eine Kombination beider Ansätze, um auch unbekannte oder fehlerhafte Formate mit einer manuell erstellten Formatbeschreibung einlesen zu können.

Das Erstellen einer Formatbeschreibung und das Parsen der zu validierenden Metadaten sollte über eine einfach zu bedienende grafische Benutzeroberfläche möglich sein.

Für das Parsen von XML Daten gibt es zur Zeit folgende De-facto-Standards:

- Event-based : SAX (Simple API for XML)[25]
- Tree-based : DOM (Document-Object-Model)[26]

Ein SAX Parser arbeitet ereignisorientiert und liest die Daten sequentiell ein. Dies erlaubt eine hohe Verarbeitungsgeschwindigkeit, deshalb eignet sich SAX vor allem für das Parsen von großen Datenmengen.

Die DOM API erlaubt die Navigation zwischen den einzelnen Knoten eines Dokuments und das Erzeugen, Verschieben und Löschen von Elementen und Textinhalten. Da dazu das komplette Dokument in den Arbeitsspeicher geladen werden muss, erfordert DOM allerdings viel Speicherkapazität und die Verarbeitung ist dementsprechend langsamer.

Mit XQuery[27] können einzelne Teile eines Dokuments adressiert werden, es sind (bislang) aber keine Änderungen an den Daten möglich.

Eine wichtige Anforderung an ein Validierungstool ist, dass auch große Dateien von umfangreichen Datensammlungen verarbeitet werden können. Aus diesem Grund ist die Verwendung eines SAX-Parsers oder XQuery zur Validierung besser geeignet als ein DOM Parser, der zur Validierung zuerst die komplette Datei einliest. Der SAX-Parser prüft die Daten beim Einlesen auf Wohlgeformtheit und erzeugt schon beim Einlesen ggf. entsprechende Fehlermeldungen.

25 http://www.saxproject.org
26 http://www.w3.org/DOM
27 http://www.w3.org/TR/xquery/

6.1.2 Prüfen auf Gültigkeit der XML Daten

Ein XML Parser prüft nur auf Wohlgeformtheit der XML Daten, für die weitere Validierung muss deshalb ein entsprechendes Schema eingelesen werden, das die XML Daten auf Gültigkeit gegenüber einem XML Schema prüft. Für die gängigen Metadatenformate liegen solche XML Schema Dateien bereits vor und können nachgenutzt werden. Wenn die XML Metadaten Angaben zu dem verwendeten Schema enthalten, können die entsprechenden XML Schema Dateien über den URL des Namespaces identifiziert, importiert und für die Validierung eingesetzt werden.

Falls für die zu validierenden Metadaten kein Schema existiert, muss es zuerst erstellt werden. Das Tool sollte dafür eine Eingabemaske mit entsprechenden Prüfroutinen bereitstellen. Die meisten XML Editoren bieten eine menügesteuerte Unterstützung zur Erstellung von Schemata an (z. B. Stylus Studio oder Altova Spy). Für nichtkommerzielle Zwecke gibt es auch entsprechende Freeware Produkte.

Mit der XML Schemasprache kann geprüft werden, ob die XML Metadaten gegenüber einem Anwendungsprofil gültig sind. Im einzelnen kann geprüft werden:

- Gültigkeit der verwendeten Metadatenelemente
- Vorhandensein der verpflichtenden Elemente
- Zulässige Anzahl von Elementen
- Überprüfung von gültigen Attributwerten
- Überprüfung der zulässigen Wertebereiche der Elemente und Attribute
- Überprüfung der Werte bei Elementen mit einem definierten Format wie z.B. Datumsangaben mit entsprechenden Encoding Schemes[28].

6.1.3 Regelbasierte Validierung

Neben der grammatischen Validierung mit einem Schema ermöglicht Schematron[29] eine weitergehende regelbasierte Überprüfung der Metadaten. Mit der Schematron Assertion Language können Validierungsregeln formuliert und überprüft werden z. B:

[28] http://dublincore.org/documents/2010/10/11/dcmi-terms/#H5
[29] http://www.schematron.com/

- Überprüfung von Parent-Child Strukturen in Abhängigkeit von Attributwerten
- Überprüfung von kontrolliertem Vokabular anhand von externen Normdateien (diese Daten können online eingelesen werden oder auch dynamisch aus unterschiedlichen Quellen zusammengestellt werden) .

Die Schematron Regeln können separat verarbeitet werden oder mit <appinfo> in ein XML Schema eingebunden werden. Die Kombination von Schematron mit XML-Schema oder RELAX ermöglicht neben der grammatikalischen Prüfung der Metadaten auch eine regelbasierte Validierung. Zudem ist es mit Schematron möglich, Fehlermeldungen selbst zu formulieren und damit auch konkrete Handlungsanweisungen zu geben.

Ein Schematron Schema gliedert sich in Assertions, Rules, Patterns und Phasen. Assertions sind Behauptungen, die auf Wahrheit überprüft werden. So kann z. B. kontrolliert werden, ob ein bestimmtes Attribut zu einem Element vorhanden ist.

Mit ‘assert test=XPATH Ausdruck’ erfolgt eine Reaktion wenn der Ausdruck nicht wahr ist.

Mit ‘report test=XPATH Ausdruck’ erfolgt eine Reaktion wenn der Ausdruck wahr ist.

Mehrere Assertions können zu einer Rule gruppiert werden, die sich auf einen bestimmten Kontext bezieht. Diese Rules können wiederum in ein Pattern zusammengefasst werden, das über den Namen bzw. eine ID referenziert werden kann. Patterns können wieder in Phasen zusammengefasst werden. Dieser modulare Aufbau ermöglicht es dem Anwender, die Validierung in unterschiedlichen Stufen vorzunehmen. Ein Schema kann so für unterschiedliche Anwendungen eingesetzt werden.

Die Schematron Regeln können auch in ein XML Schema eingebunden werden.

Die Formulierung von Schematron Regeln erfolgt mit den Sprachelementen von XPATH[30] und XSLT[31].

Als Eingabehilfe sollte das Validierungstool über eine Editier- und Kontrollfunktion zur Formulierung von neuen Regeln bzw. zur Modifizierung von bestehenden Regeln verfügen.

30 http://www.w3.org/TR/xpath
31 http://www.w3.org/TR/xslt

6.1.4 Ausgabe von Fehlermeldungen

Eine wichtige Anforderung ist die Ausgabe von verständlichen Fehlermeldungen, die auch Hinweise zur Behebung der Fehler liefern. Die Meldungen müssen eine Referenz auf den Metadatensatz, der den fehlerhaften Code enthält, liefern. Es muss möglich sein, fehlerhafte Metadatenelemente in der Originaldatei zu lokalisieren, um gegebenenfalls sofort Korrekturen vornehmen zu können. Idealerweise wird dazu ein Link verwendet, der auf das Metadatenelement referenziert.

Bei Schematron können die Fehlermeldungen bei der Erstellung der Regeln vom Anwender selbst formuliert werden. XML Schema bietet diese Möglichkeit nicht, die Fehlermeldungen sind im Quellcode des Parsers codiert und deshalb vom Anwender nicht zu ändern.

6.1.5 Erstellung einer Report-Datei

Die Fehlermeldungen müssen in einer Report-Datei dauerhaft gespeichert werden. Diese Datei kann entweder Links zu den betreffenden Metadatenelementen in der überprüften XML Datei oder die Zeilennummer enthalten. Aus den Schematron Fehlermeldungen in XML lassen sich mit Hilfe von entsprechenden Stylesheets Reports mit Links auf die Originaldaten in X(HTML) generieren. Durch die entsprechende Modifizierung der vorhandenen Schematron Stylesheets lassen sich aber auch andere Dateiformate erzeugen (z.B. PDF, RTF, einfache Textdatei).

6.1.6 Statistische Auswertung der Validierungsergebnisse

Um einen schnellen Überblick über alle Fehler zu erhalten, sollten identische Fehlermeldungen summiert und in Relation zu der Gesamtzahl der entsprechenden Datenelemente grafisch dargestellt werden. So lassen sich schnell häufig auftretende Probleme erkennen. Die statistische Auswertung der Fehlerhäufigkeiten ermöglicht außerdem eine erste Einschätzung über die Qualität der Metadaten.

Statistische Auswertungen der Fehlerhäufigkeiten lassen sich mit den entsprechenden Stylesheets aus dem Schematron Fehlerreport generieren.

Alle im Dokument vorkommenden Zeichen sollten mit den entsprechenden Dezimal- und Hex-werten aufgelistet werden. Damit lassen sich eventuelle Inkonsistenzen auf der Zeichenebene schnell erkennen.

6.1.7 Erstellung von benutzerdefinierten Indizes

Die semantische Korrektheit der Inhalte von Metadatenelementen, die kein kontrolliertes Vokabular enthalten, kann durch eine automatische Validierung allein nicht geprüft werden.

Deshalb muss es für einzelne Metadatenelemente (z.B. Titel einer Ressource) möglich sein, Indizes zu erstellen, die nur bestimmte Elemente zur manuellen Überprüfung auflisten. Das manuelle Prüfen der Daten wird dadurch vereinfacht und der Zeitaufwand dafür deutlich reduziert.

Bei alphabetisch sortierten Indexlisten fallen Inkonsistenzen der Daten schnell auf, am Anfang und Ende eines alphabetischen Indexes finden sich oft falsche Einträge (ungültige Zeichen, nicht erlaubte Abkürzungen, usw.)

Alphabetische Indizes ermöglichen außerdem den schnellen Zugriff auf bestimmte Elemente zur manuellen Überprüfung.

6.1.8 Änderungen an den Metadaten

In bestimmten Fällen (Use case 1 und 2) kann es nützlich sein, die durch die Validierung entdeckten fehlerhaften Metadaten sofort korrigieren zu können, ohne dazu ein anderes Programm zu öffnen. Dazu ist die DOM API am besten geeignet, da sie die direkte Adressierung und Änderung des Inhalts von einzelnen Metadatenelementen ermöglicht.

Das gesamte Dokument muß dazu in den Speicher geladen werden, was bei großen Dateien problematisch ist. Eine mögliche Lösung könnte sein, nur die betroffenen Teile der Datei zu laden und zu bearbeiten.

Eventuell lässt sich diese Funktion in Zukunft auch mit XQuery realisieren, zur Zeit gibt es bei XQuery aber noch keine Möglichkeit des direkten Editierens der Metadaten.

Für andere Szenarien, z.B. die Validierung von umfangreichen Metadatensammlungen ist dieses Vorgehen zu aufwendig, hier sind eher automatische Korrekturläufe auf der Grundlage der Fehlerreports zu empfehlen.

6.1.9 Verwaltung der Report-Dateien

Bei einer wiederkehrenden Bearbeitung gleicher oder analoger Datensammlungen kann die Entwicklung der Metadatenqualität über längere Zeiträume von Interesse sein. Für derartige Analysen müssen die Ergebnisse der einzelnen Prüfungen so aufbewahrt werden, dass sie für aggregierte Auswertungen zur Verfügung stehen.

Für die Verwaltung der Report-Dateien gibt es - je nach Ausgabeformat - unterschiedliche Möglichkeiten. Bei XML Dateien könnten die Verwaltung und der Zugriff auf die Reports mit XQuery erfolgen.

6.2 Benutzeroberfläche

Die Benutzeroberfläche sollte eine Auswahlmöglichkeit für die gängigen Metadatenformate und die dazugehörigen Schema und Schematron-Dateien bieten. Es muss möglich sein, verschiedene Stufen der Validierung auszuwählen (Wohlgeformtheit, Gültigkeit anhand eines Schemas, Überprüfung durch Schematron-Regeln). Für die Formulierung von eigenen Schema bzw. Schematron Regeln sollten ein Eingabeformular und eine Validierung vorhanden sein.

Schematron lässt sich relativ leicht in ein GUI implementieren. Am Markt vorhandene XML Editoren wie Oxygen bieten bereits eine maschinelle Validierung mit dem W3C-Schema oder RELAX und Schematron an.

Ein Beispiel für eine Schematron Implementierung ist Schematron-Report[32], eine Schematron Erweiterung, die den Validierungsreport in einer HTML Seite mit zwei Fenstern ausgibt. Das obere Fenster enthält die Meldungen, das untere Fenster zeigt die Original XML Daten. Die Fehlermeldungen referenzieren direkt auf die entsprechenden Elemente in der Original XML Datei. Für größere Datensammlungen ist die Ausgabe des Fehlerreports in einer HTML Datei jedoch nicht geeignet, da diese Art der Ausgabe sehr speicherintensiv ist. Bei anderen Implementierungen gibt es die Option, den Fehlerreport auch als einfache Textdatei zu speichern.

32 http://xml.ascc.net/resource/schematron/schematron-report.html

7. Schlußbetrachtung

Von den fünf verschiedenen Validierungs-Tools, die wir im Rahmen dieser Analyse untersucht haben, hat sich für unsere Nutzungsszenarien eine Kombination aus Schema und Schemtron als die erfolgversprechendste Lösung erwiesen.

Die Validierung von XML Metadaten lässt sich mit den Möglichkeiten von XML Schema und Schematron weitgehend automatisieren. Die für die Realisierung nötigen Software Module sind als Open Source verfügbar und können durch die vorhandenen Schnittstellen zu einem leistungsfähigen Software-Tool kombiniert werden. Auch die Einbindung der Validierungssoftware in komplexe serverseitige Anwendungen ist durch die Offenheit der Komponenten problemlos möglich. Ein XML-basiertes System ist für zukünftige Entwicklungen offen und damit zukunftssicher.

Eine automatische Überprüfung der Semantik von Metadaten ist ohne den Rückgriff auf kontrolliertes Vokabular jedoch nicht möglich. Eine manuelle Überprüfung sollte deshalb für den Anwender durch statistische und grafische Elemente, die einen Überblick über die Metadatenelemente und die Feldinhalte geben, möglichst einfach gestaltet werden.

Die dafür nötigen Daten können durch die Auswertung der Reportdateien gewonnen werden und mit modifizierbaren Stylesheets präsentiert werden.

Die Validierung von Metadaten in XML ließe sich so mit relativ wenig Programmieraufwand realisieren. Schwieriger ist die Überprüfung von Metadaten, die nicht in XML vorliegen (z. B. ISO2709). Diese Datenformate müssten zuerst in XML konvertiert werden, um die XML Tools nutzen zu können.

In Kapitel 6 haben wir die Funktionalitäten aufgezeigt, die ein auf Schema und Schematron basierendes Validierungstool erfüllen müsste, um die von uns genannten Kriterien zu erfüllen. Dieses Tool sollte

- die Funktionen von Schema und Schematron so kombinieren, dass die Validierung weitgehend automatisiert erfolgt,
- das Parsen unterschiedlicher Formate erlauben, auch dann, wenn diese syntaktisch nicht fehlerfrei sind,

- die Konversion von Formaten, die nicht in XML vorliegen, in ein XML-Format unterstützen,
- eine Benutzeroberfläche zur Verfügung stellen, die das Einbinden neuer Formate ermöglicht,
- das Einbinden kontrollierter Vokabulare in die Validierungsroutine ermöglichen,
- übersichtliche, menschenlesbare Fehlerberichte generieren,
- die manuelle Überprüfung der Metadateninhalte durch benutzerdefinierte Indizes unterstützen.