Development and Validation of an XML Schema for Automated Environmental Reporting on XML Basis

Jorge Marx Gomez, Christoph Hermann[1]
Ralf Isenmann[2]

Abstract

Today automated environmental reporting requires an application of modern sustainable technologies in order to fulfil the exalted claims of the target groups. An advantageous approach is introduced here for a processing of the growing information supply and the generated environmental reports. This approach is usable to semi-automatically create XML Schemas and to check environmental reports, generated in XML, for formal correctness. This paper discusses the advantages and disadvantages of the usage of XML Schemas versus the use of DTDs. Furthermore, it is explained how a XML Schema can be semi-automatically generated from an already available DTD. It will also be presented how XML Schemas can be used for validation of environmental report instances in XML.

Keywords

DTD, Environmental reports, standardization, validation, XML Schema

1 Introduction

The first environmental reports were published in Germany at the beginning of nineties. Since that time the interest on information supply and the number of different addressees has risen continuously. This requires fair environmental reporting for target groups and data preparation for different people. The first step to automated environmental reporting has already been done and it is possible to automatically manage the information flow and to generate the documents from it. This happened due to the use of XML[3] and the creation of a Document Type Definition[4] (DTD) as a basis for an application system for production and fair target group presentation of

[1] Otto-von-Guericke-Universität Magdeburg, Faculty of Computer Science, Institute for Technical and Business Information Systems, P.O. Box 4120, 39016 Magdeburg, Germany, email: gomez@iti.cs.uni-magdeburg.de, Internet: http://www-wi.cs.uni-magdeburg.de

[2] University of Kaiserslautern, Department of Business Information Systems and Operations Research (BiOR), Gottlieb-Daimler-Straße, P.O. Box 3049, 67653 Kaiserslautern, Germany, email: isenmann@bior.de; brosowski@bior.de; beisel@bior.de; Internet: http://www.bior.de

[3] see Wyke (2002)

[4] http://www.w3.org/TR/REC-xml

environmental reports. The provided DTD allows the generated XML documents to be tested for formal correctness[5].

However, to allow a better structuring of the data and an easier automated processing of standardized models, a structure of data and metadata must be standardized. Up to now it is not possible to do this by the use of a DTD. An approach suggested by the World Wide Web Consortium[6] (W3C) is the use of XML Schema[7]. It forces a specified document structure for XML documents and then, in the next step, validates generated documents with help of XML Schema.

XML Schema is a powerful alternative to a DTD, which enables to define a class for XML documents and to validate the instances of this document, i.e. documents can be checked on structural validity. XML Schema allows the formulation of complicated restrictions (constraints) by the use of elements and attributes, which are beyond DTD possibilities. The XML Schema is a formal specification of a valid XML document whereas this schema itself is also an XML document[8].

The exact advantages of XML Schema usage vs. DTD are explained in chapter 2 “Advantages of XML Schema versus DTD“. However, in chapter 1 a short introduction to XML, DTD and XML Schema will be given in order to enable a better understanding of the precise backgrounds.

A motivation for the application of XML and XML Schema are advantages, which arise during their usage in form of an automated generation, validation and sustainability of the created environmental reports and their copies.

2 Representation of XML Schema

From DTD to XML pattern – an overview:

For example, an easy XML document with reference to a suitable DTD is:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE artikel SYSTEM "artikel.dtd">
<artikel>
<titel>Titel des Dokumentes</titel>
<teaser>Ein kurzer Abriss über den Text</teaser>
<inhalt>Ein sehr langer und interessanter Inhalt eines Dokumen-
   tes</inhalt>
</artikel>
```

Listing 1: artikel.xml

[5] http://www.devmag.net/xml/xml_schema_einfuehrung.htm
[6] http://www.w3.org/
[7] http://www.w3.org/XML/Schema
[8] http://klever.multimedia.fh-augsburg.de/publications/talks/DECUS2001.pdf

```
<!ELEMENT artikel (titel, teaser, inhalt )>
<!ELEMENT titel (#PCDATA)>
<!ELEMENT teaser (#PCDATA)>
<!ELEMENT inhalt (#PCDATA)>
```

Listing 2: artikel.dtd

The corresponding XML Schema is, as already described, very similar to the XML document. Generally, an XML Schema has the file extension *.xsd.

```
<?xml version="1.0"encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="artikel">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="titel" type="xs:string"/>
        <xs:element name="teaser" type="xs:string"/>
        <xs:element name="inhalt" type="xs:string"/>
       </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Listing 3: artikel.xsd

The first line of the XML Schema defines the namespace for the elements of this schema. This means it will be referenced on the XML Schema definition of the W3C, so that the parser recognizes where the elements are defined[9]. This namespace definition enfolds the whole schema and is the root element of the document. The namespace to use is defined in `xmlns`. This reference is important to avoid conflicts by using same identifiers for different purposes.

Afterwards, the element Article is defined. Article is an element with several subelements. `Complextype` takes care for the definition of contained elements. Other elements are so-called "easy" elements; they contain no nesting levels. A data type, assigned to those elements, defines what type the contained data are.

An integration[10] of XML Schema in a XML document occurs in contrast to DTD as follows:

```
<artikel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="artikel.xsd">
```

Listing 4: Integration of XML Schema

The complete XML document with integrated XML Schema looks like this:

[9] see Binstock (2003)
[10] see Niedermair (2002)

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<artikel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="artikel.xsd">

<titel>Titel des Dokumentes</titel>

<teaser>Ein kurzer Abriss über den Text</teaser>

<inhalt>Ein sehr langer und interessanter Inhalt eines Dokumen-
   tes</inhalt>

</artikel>
```

Listing 5: Complete XML Document

As far as environmental reporting is concerned XML Schema has for example the advantage, that data types for emission of pollutant amounts can be defined. This advantage is discussed in the next chapter in more detail.

3 Advantages of XML Schema versus DTD

DTDs derive from the time of XML predecessor's language SGML[11] and pursue the document-oriented approach, which was usual in that time. Nevertheless, XML[12] and especially XML environmental reports with the use of XML Schema are data oriented because they use different data types. XML itself is not output oriented, because the final representation of documents is defined by special formatting models[13]. So, XML is data oriented. DTDs allow basic validation of XML documents regarding elements' nesting, frequency restrictions for elements, default values, permissible attributes and attribute types. However, DTDs do not provide deep control of format, elements' data types and attribute values.

DTDs have an insufficient data type support. Only five content models are possible: sub models, ANY, EMPTY, PCDATA and mixed content. If it is once defined that an element contains character data, no further control of length, type or format is possible. For simple documents, which do not depend on a data structure, this kind of control is absolutely sufficient. Caused by the use of XML in applications with a big data amount as for example environmental reporting, a further exact control of elements' and attributes' contents is always more important. The XML Schema supports this functionality.

The XML Schema Standard of the W3C encloses the following functionalities: simple and complex data types, derivation and inheritance of types, frequency restrictions for elements, documentation, namespace sensitive element- and attribute declarations.

The most important of those functionalities is the supplement of simple data types by parsed character data and attribute values. In contrast to DTDs, schemas define precise rules for a content of attributes and elements. One should not use only sim-

[11] http://www.w3.org/MarkUp/SGML/

[12] http://www.jeckle.de/files/UStuttgart2002.pdf

[13] http://www.linux-magazin.de/Artikel/ausgabe/2002/01/xml/xml.html

ple data types[14] (more than 40 exist); it is also possible to introduce new data types, to derive new types from available ones and to reuse those from other schemas. With XML Schema one also has a possibility to verify the content of elements according to so-called text pattern, i.e. the content of elements must correspond to the regular expressions defined in the pattern.

Beside simple data types, there are also enlarged functionalities for specification of number and order of child elements for certain places in the document. Indeed, there are certain content models among DTDs, but the content models of XML Schema offer more possibilities. In DTDs one can declare only limited cardinality: whether the element is optional or not (?, *) and whether it is repeatable or not (+, *). In XML it is possible to declare cardinalities accurately and, therefore, to define exactly a number of elements, which can occur (minOccur, maxOccur).

Different people and organizations everywhere in the world exchange their XML documents. Therefore, to avoid misunderstanding of names, the use of namespaces becomes more and more important. Namespaces are fully supported in XML Schema in contrast to DTDs[15]. DTDs do not support the declaration of namespaces because the XML Namespaces Recommendation was published after the recommendation for XML 1.0. Different from DTDs, where elements must have a namespace-prefix, in XML Schema the validation occurs according to the combination of name space-URI[16] and locally used names and not towards a prefix name.

In DTDs no standard storage location exists for documentation of single elements or for metadata e.g. producer and date. This can be realized through comments, however, in XML Schema it is solved by the element `<documentation>`. The main disadvantage of using XML comments to insert additional information, is that while automatically processing XML documents the parser is not obliged to keep and process the XML comments. It can happen that this information would get lost during the processing. It is also easier for an application to read additional information if these are structured like the rest of the document. If one marks additional information, like elements and attributes described by this information, with mark-up character, innumerable possibilities arise for automated documentation creation.

4 Development of XML Schema based on the existing DTD

The transformation of already existing DTD can be implemented in 3 separate steps. The first two will be considered here exactly. At the first step existing DTDs are automatically transferred with corresponding tools (here XMLSpy[17]) into XML Schemas. I the second step, XML Schemas will be changed manually in order to use the advantages of XML Schemas. Up to now, those manual changes had intuitive character. Nevertheless, it would be possible to develop a similar approach to Schraml[18] or to do changes by means of software engineering methods. The third

[14] http://www.w3.org/TR/xmlschema-0/#simpleTypesTable

[15] In DTDs all elements declarations are global, i.e. applicable in each context

[16] Uniform Ressource Identifier

[17] http://www.xmlspy.com/

[18] see Schraml (1997)

step consists not only of usage of the advantages mentioned in chapter 3, but also in a continuation of XML Schema development.

The first two steps will be described by means of two Hasseroeder Brewery[19] (environmental-report.dtd and environmental-report-part.dtd) DTDs, which are descended from Mario Krueger master thesis[20]. Because size of a whole DTD/schema would exceed the length of this document, the DTDs are made available on-line under http://guschtel.heim3.tu-clausthal.de/paper/. Also environmental-report.dtd[21] as well as environmental-report-part.dtd[22] can be found there.

Step 1: Automated transfer of DTD in XML Schema:

At the first step DTDs were automatically converted into XML Schema by means of XMLSpy. One reaches this in XMLSpy simply through the menu item DTD/Schema => Convert DTD/Schema.

The dialog window as shown in figure 1 opens where one selects that the given DTD should be transferred in XML Schema.

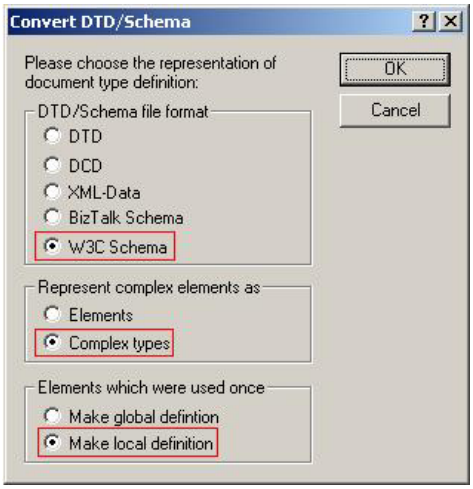

Fig. 1: Conversion of the DTD by Means of XMLSpy

As an output format a W3C XML Schema is chosen. Furthermore, it is selected that complex elements are represented as complex types, so that in the 2nd step of the DTD transformation it is possible to introduce corresponding restrictions for some data types. Elements, used only once, should be defined locally, so that there is no overlapping with other elements.

[19] http://www.hasseroeder.de/
[20] see Krueger (2001)
[21] http://guschtel.heim3.tu-clausthal.de/paper/environmental-report.dtd
[22] http://guschtel.heim3.tu-clausthal.de/paper/environmental-report-part.dtd

Hence, the automatically generated XML Schemas environmental-report.xsd[23] and environmental-report-part.xsd[24] can be found again in the Internet.

Step 2: Manual improvement of generated XML Schema:

Other changes in XML Schema will be shown on the following two abstracts of automatically generated environmental-report.xsd. They will help to understand for example changes, which were made manually.

```
<xs:complexType name="addressType">
        <xs:sequence>
                <xs:element ref="name" minOccurs="0"/>
                <xs:element name="organization"
   type="xs:string"/>
                <xs:element name="street" type="xs:string"/>
                <xs:element name="city" type="xs:string"/>
                <xs:element name="state" type="xs:string"/>
                <xs:element name="zip" type="xs:string"/>
                <xs:element name="country" type="xs:string"/>
                <xs:element name="email" type="xs:string"/>
                <xs:element name="telephone" type="xs:string"/>
                <xs:element name="fax" type="xs:string"/>
                <xs:element ref="URL" minOccurs="0"/>
        </xs:sequence>
</xs:complexType>
```

Listing 6: Bit 1 from environmental-report.xsd

and

```
<xs:element name="name" type="xs:string"/>
<xs:element name="day" type="xs:string"/>
<xs:element name="month" type="xs:string"/>
<xs:element name="year" type="xs:string"/>
<xs:element name="org-unit" type="xs:string"/>
```

Listing 7: Bit 2 from environmental-report.xsd

In the first bit, different restrictions were inserted like restrictions with regular expressions, an enumeration for the federal states and other limitations like length of the string and kind of data types. After manual modification of the first abstract (listing 6**),** XML Schema code looks like this:

[23] http://guschtel.heim3.tu-clausthal.de/paper/environmental-report.xsd
[24] http://guschtel.heim3.tu-clausthal.de/paper/environmental-report-part.xsd

```
<xs:complexType name="addressType">
        <xs:sequence>
                <xs:element name="name" minOccurs="0"
   type="xs:string"/>
                <xs:element name="organization"
   type="xs:string"/>
                <xs:element name="street" type="xs:string"/>
                <xs:element name="city" type="xs:string"/>
                <xs:element name="state">
                        <xs:simpleType>
                                <xs:restriction base="xs:string">
                                        <xs:enumeration va-
   lue="Baden-Württemberg"/>
                                        <xs:enumeration va-
   lue="Bayern"/>
                                        <xs:enumeration va-
   lue="Berlin"/>
                                        <xs:enumeration va-
   lue="Brandenburg"/>
                                        <xs:enumeration va-
   lue="Bremen"/>
                                        <xs:enumeration va-
   lue="Hamburg"/>
                                        <xs:enumeration va-
   lue="Hessen"/>
                                        <xs:enumeration va-
   lue="Mecklenburg-Vorpommern"/>
                                        <xs:enumeration va-
   lue="Niedersachsen"/>
                                        <xs:enumeration va-
   lue="Nordrhein-Westfahlen"/>
                                        <xs:enumeration va-
   lue="Rheinland-Pfalz"/>
                                        <xs:enumeration va-
   lue="Saarland"/>
                                        <xs:enumeration va-
   lue="Sachsen"/>
                                        <xs:enumeration va-
   lue="Sachsen-Anhalt"/>
                                        <xs:enumeration va-
   lue="Schleswig-Holstein"/>
                                        <xs:enumeration va-
   lue="Thüringen"/>
                                </xs:restriction>
                        </xs:simpleType>
```

```
                </xs:element>
                <xs:element name="zip">
                        <xs:complexType>
                                <xs:simpleContent>
                                        <xs:restriction ba-
se="xs:integer">
                                                <xs:minLength
value="5"/>
                                                <xs:maxLength
value="5"/>
                                        </xs:restriction>
                                </xs:simpleContent>
                        </xs:complexType>
                </xs:element>
                <xs:element name="country" type="xs:string"/>
                <xs:element name="email">
                        <xs:complexType>
                                <xs:simpleContent>
                                        <xs:restriction
base="xs:string">
                                                <xs:pattern va-
lue=".+@.+\.\w{2,6}"/>
                                        </xs:restriction>
                                </xs:simpleContent>
                        </xs:complexType>
                </xs:element>
                <xs:element name="telephone" type="xs:string"/>
                <xs:element name="fax" type="xs:string"/>
                <xs:element name="URL" minOccurs="0"
type="xs:anyURL"/>
        </xs:sequence>
</xs:complexType>
```

Listing 8: Manually Changed Bit 1

On the basis of second abstract (listing 7) it will be explained how one can insert, for example, restrictions for a date:

```
<xs:element name="name" type="xs:string"/>
<xs:element name="day">
      <xs:simpleType>
           <xs:restriction base="xs:string">
```

```
                    <xs:pattern value="(0?[1-9])|([1,2][0-
9])|(3[0,1])"/>
            </xs:restriction>
      </xs:simpleType>
</xs:element>
<xs:element name="month">
      <xs:simpleType>
             <xs:restriction base="xs:string">
                          <xs:pattern value="(0?[1-
9])|(1[0-2])"/>
                    </xs:restriction>
             </xs:simpleType>
</xs:element>
<xs:element name="year">
      <xs:simpleType>
                    <xs:restriction base="xs:string">
                  <xs:pattern value="[0-9]{4}"/>
             </xs:restriction>
      </xs:simpleType>
</xs:element>
   <xs:element name="org-unit" type="xs:string"/>
```

Listing 9: Manually Changed Bit 2

The changes made, e.g. in the first bit are valid obviously only for Germany and should only show the possible changes. Also in the second bit it would be possible to change the XML Schema so that, according to ISO Standard[25], instead of 3 elements for day, month and year a single element of type `date` is used.

Step 3:

In the 3rd step, a new development of the XML Schema and, among other things, internationalization should take place. One could realize this, e.g. by means of „`substitutionGroup`“, as follows:

```
<xs:element name="abstract_head-element" type="xs:string" ab-
   stract="true"/>
<xs:element name="head-element" type="xs:string" substitution-
   Group="abstract_head-element" xml:lang=”en”/>
<xs:element name="kopf-element" type="xs:string" substitution-
   Group="abstract_head-element" xml:lang=”de”/>
<xs:element name="element-principal" type="xs:string" substitu-
   tionGroup="abstract_head-element" xml:lang=”fr”/>
```

[25]ISO 8601 Date and Time Formats (§D): http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/#isoformats

Listing 10: Possibility of Multilingualism with XML Schema

Therefore, one would define an element `abstract_head-element`. Within this element, each of the three following elements can be used according to the desired language.

Furthermore, the new XML Schema should be splitted by means of `xsd:include` into small components. The splitting will help to receive clearness and will simplify the serviceability, as the provided XML Schema is too complex. At this point the XML Schema documentation can also contribute by means of `<documentation>`.

5 Test and Validation of XML Schema

There are a lot of tools for testing of XML environmental reports according to provided XML Schema[26]. They enable to carry out the validation. For the purposes of a lower-cost solution only Open Source / Freeware solutions will be considered here. In the industrial environment the validation can occur and occurs at various work places. The following 3 tested functional variants are examined here:

1. XML Parser for Java of IBM (xml4j)[27]
2. Oracle XDK[28]
3. Sun Multi-Schema XML Validator[29]

It was possible to validate instances of the Hasseroeder environmental report towards provided XML Schema with all 3 variants.

In the following chapter it will be explained in detail how XML files can be validated according to the provided XML Schema with the help of the XML Parser for Java developed by IBM. A necessary requirement for the usage is an installed Java 1.1[30] runtime.

By means of xml4j, the sample program XMLGrammarBuilder provided by IBM, demonstrates the possibilities of validating instances of XML documents against XML Schemas. This happens by an easy call of the program with several parameters:

```
C:\...\xml4j-4_2_2>java XMLGrammarBuilder

usage: java xni.XMLGrammarBuilder [-p config_file] -d uri ... |
   [-f|-F] -a uri .

.. [-i uri ...]
```

[26] see Van der Flist (2002)
[27] http://www.alphaworks.ibm.com/tech/xml4j
[28] http://otn.oracle.com/tech/xml/xdkhome.html
[29] http://wwws.sun.com/software/xml/developers/multischema/
[30] For testing Java 1.4 was used. See http://java.sun.com/j2se/

```
options:
  -p config_file:   configuration to use for instance validation
  -d    grammars to preparse are DTD external subsets
  -f  | -F    Turn on/off Schema full checking (default off)
  -a uri ...  Provide a list of schema documents
  -i uri ...  Provide a list of instance documents to validate

NOTE:  both -d and -a cannot be specified!
C:\...\xml4j-4_2_2>
```

Program Output 1: Output of XMLGrammarBuilder of IBM

Now, the environmental report of Hasseroeder Brewery can be validated without errors according to the provided XML Schema:

```
C:\...\xml4j-4_2_2>java XMLGrammarBuilder -f -a
   C:\...\environmental-report.xsd -i C:\...\hasseroeder.xml
C:\...\xml4j-4_2_2>
```

Program Output 2: Validation of XML Schema

If the schema is not valid, an error message will appear like. in this case with an intentional mistake:

```
C:\...\xml4j-4_2_2>java XMLGrammarBuilder -f -a
   C:\...\environmental-report-part.xsd -i
   C:\...\hasseroeder_introduction. xml
[Error] environmental-report-part.xsd:828:34: cos-element-
   consistent: Fehler für
 Typ 'tableType'.  Mehrere Elemente mit dem Namen 'table-
   header,null' und mit unterschiedlichem Typ werden in der Mo-
   dellgruppe verwendet.
[Error] environmental-report-part.xsd:828:34: cos-nonambig:
   "":table-header und "":table-header (oder Elemente ihrer Sub-
   stitutionsgruppe) verletzen die Regel "Unique Particle Attri-
   bution".
```

Program Output 3: Error Message During the Validation of Incorrect Schema

The source code for parsing and validation of the XML file instances according to the provided XML Schema is quite simple designed. The most important code elements of IBM file `XMLGrammarBuilder.java` are presented here:

```
// Symboltabelle und Parser vorbereiten:
SymbolTable sym = new SymbolTable(BIG_PRIME);
XMLGrammarPreparser preparser = new XMLGrammarPreparser(sym);
XMLGrammarPoolImpl grammarPool = new XMLGrammarPoolImpl();
```

```
preparser.registerPreparser(XMLGrammarDescription.XML_SCHEMA,
   null);


// Jedes angegebene Schema parsen

Grammar g = prepar-
   ser.preparseGrammar(XMLGrammarDescription.XML_SCHEMA, strin-
   gToXIS((String)schemas.elementAt(i)));


// Konfiguration erstellen

parserConfiguration = new IntegratedParserConfiguration(sym,
   grammarPool);


//jede der angegebenen XML Dateien validieren

parserConfigura-
   tion.parse(stringToXIS((String)ifiles.elementAt(i)));
```

Source Code 1: Parts of IBM XMLGrammarBuilder

Thus, with the provided parser and few lines of code it is easier to integrate a validation of XML documents into the own application in order to validate without external programs. This can be done, e.g. on server side applications on Java Servlets basis in order to allow the validation and to avoid mistakes in the data.

The validation with Oracles XDK is done similarly to IBM's xml4j. It can be simply carried out with help of the provided class XSDSetSchema, which in principle offers the same functionality as XMLGrammarBuilder.java of IBM:

```
C:\...\oracle>java XSDSetSchema

Usage: java XSDSetSchema <schema_file> <xml_file>


C:\...\oracle>java XSDSetSchema C:\...\environmental-report-
   part.xsd C:\...\hasseroeder_introduction.xml

Parsing C:\...\hasseroeder_introduction.xml

The input file <C:\...\hasseroeder_introduction.xml> parsed
   without errors


C:\...\oracle>
```

Program Output 4: Output of XSDSetSchema of Oracle

Unlike the tools already introduced here, the Multi-Schema XML Validator (MSV) developed by Sun is not a complete implementation of a parser etc. to process XML files. Instead, it is a command line tool and Java library, which can be used for validation of XML documents according to different types of XML Schemas. MSV supports the following types of XML Schemas: RELAX NG, RELAX Namespace, RELAX Core, TREX, XML DTDs, and a part of XML Schema Part1.

Nevertheless, a validation occurs comparably:

```
C:\...\sun>javac JAXPWithMSVDemo.java


C:\...\sun>java JAXPWithMSVDemo C:\...\environmental-report-
   part.xsd C:\...\hasseroeder-organisation.xml
C:\...\hasseroeder-organisation.xml is valid


C:\xml\sun>
```

Program Output 5: Command Line Call of MSV of Sun

Because of the incomplete implementation of the W3C XML Schema standards this tool cannot be fully recommended.

With the given tools it is possible in an easy way to validate instances of XML documents on the basis of XML Schema[31]. Now this can be used by environmental reporting with XML to guarantee that the information need is covered by the instances of environmental report and also is correct. Through the use of Java by the provided parser it is also possible to carry out the validation on different operating systems with different addressees of the environmental reports.

The most recommendable is xml4j from IBM because this parser, based on an open source project[32], completely implements the XML Schema Standard of the W3C. Furthermore, a big developer community enhances it.

5 Summary and Perspectives

With the introduced methods it is possible to perform a semi-automatic transformation of present DTDs for environmental reports to XML Schemas. This is certainly the first step for using XML Schemas in order to take advantages of the technology. However, research should not stop at this point.

A further step would be to develop an entirely new XML Schema for a using of all advantages on the basis of the gained findings. As presented here, DTDs are not the only ones that exist and by a new XML Schema development also a standardized model for environmental reports can be created. Besides, as a basis for environmental reports, a harmonized schema should be generated. It could be done by a merge of all already available DTD/XML Schemas for environmental reports. In this case, a new developed XML Schema should offer a possibility for internationalization, among other things the multilingualism.

Also XML Schema should be unitized i.e. divided into small parts to make the support and possible extensions easier.

[31] Walmsley (2002)

[32] http://xml.apache.org/xerces-j/

If those and all other already mentioned advantages of XML Schema are implemented, one can create XML Schema, which can be a standard model for an environmental reporting with XML, which comes up to technological progress and development of XML Schema[33].

References

Binstock, C. (2003): The XML schema complete reference, Addison-Wesley, Boston.

Krueger, M. (2001): Konzeption und Entwicklung einer automatisierten Umweltberichterstattung am Beispiel der Hasseroeder Brauerei GmbH. Master Thesis. Otto-von-Guericke-Universität Magdeburg.

Niedermair, E. and Niedermair, M.: (2002): XML fuer Print und Screen.

Rusty Harold, E. and Scott Means, W.: (2nd edition 2003): XML in a Nutshell.

Schraml, T. (1997): Operationalisierung der oekologiebezogenen Berichterstattung aus Sicht des Informationsmanagements: Konzeption eines Vorgehensmodells zur formalisierten Explikation logischer Dokumenttypmodelle im Rahmen der Umweltkommunikation von Unternehmen, Dissertation, Technische Universität Dresden.

Van der Vlist, E. (2002): XML schema - The W3C's object-oriented descriptions for XML, O'Reilly, Beijing.

Walmsley, P. (2002): Definitive XML schema, Prentice Hall, Upper Saddle River.

Wyke, A. and Watt, A. (2002): XML Schema Essentials, John Wiley, New York.

Internet Addresses

NikKlever(2001):http://klever.multimedia.fh-augsburg.de/publications/talks/DECUS2001.pdf

Thiemo Fetzer (2003): http://www.devmag.net/xml/xml_schema_einfuehrung.htm

W3C (2001): http://www.w3.org/TR/xmlschema-0/

W3C (2000): http://www.w3.org/TR/REC-xml

W3C (2001): http://www.w3.org/XML/Schema

W3C (2000): http://www.w3.org/MarkUp/SGML/

W3C (2002): http://www.w3.org/TR/xmlschema-0/#simpleTypesTable

Alcova (2003): http://www.xmlspy.com/

http://xml.apache.org/xerces-j

http://www.jeckle.de/files/UStuttgart2002.pdf

http://www.linux-magazin.de/Artikel/ausgabe/2002/01/xml/xml.html

Hasseroeder: http://www.hasseroeder.de/

IBM (2003): http://www.alphaworks.ibm.com/tech/xml4j

Oracle (2003): http://otn.oracle.com/tech/xml/xdkhome.html

Sun (2003): http://wwws.sun.com/software/xml/developers/multischema/

Sun (2003): http://java.sun.com/j2se/

[33] see Rusty Harold (2003)