APPROACHES IN USING MATML AS A COMMON LANGUAGE FOR MATERIALS DATA EXCHANGE

T Ojala * and H-H Over

European Commission Joint Research Centre Institute for Energy Postbus 2, 1755 ZG Petten, Netherlands
**Email:* tauno.ojala@ec.europa.eu

ABSTRACT

Utilization of XML techniques is seen as a necessary step towards more powerful ways of incorporating semantics into data exchange used by heterogeneous systems. In this paper various techniques are studied and tried, such as XSL transformations (XSLT) and ways of extending the contents of XML Schemas, the final aim being in creating an understanding of the possibilities, and a roadmap that could possibly lead to some useful real-world applications. Based on a materials database an XML Schema is specified that defines the structure of an XML document capable of representing quite complex materials test data together with mandatory metadata. Some approaches are discussed and some of them implemented in prototypes to study the possibilities to comply with and use MatML in order to support sharing of experimentally measured materials data.

Keywords: XML, Schema, XSLT, MatML, Materials database, Materials testing, XML transformation

1 XML SCHEMAS

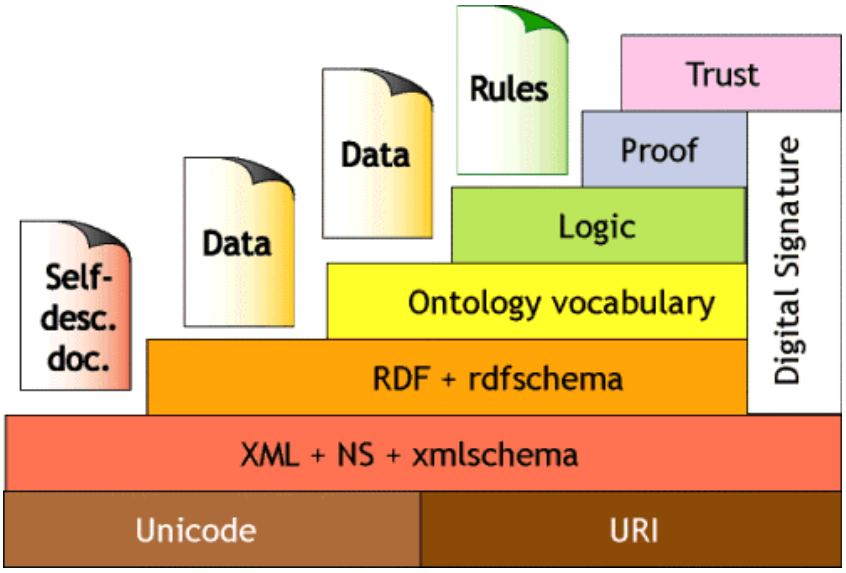
An XML schema is an XML document that defines the valid contents of a particular class of XML documents. Some well-known examples of schemas are e.g. MathML schema, which is an XML application for describing mathematical notation and capturing both its structure and content, and femML, Finite Element Modeling Markup Language, which is an effort addressing the problems of data interpretation and application interoperability in the Finite Element Modeling domain of activities for model and/or product specification portability.

The use of XML schemas has been much rarer than that of XML documents. Still one could argue that without a schema an XML document is only a little more than a comma-separated-value formatted file. It can be checked for **well-formedness** providing a good indication of the integrity of the document, and of course it can be processed much more easily than non-standard formats by standard XML parsers. Well-formedness is a prerequisite for a useful XML document, but in more sophisticated use cases it is rarely enough.

The **validity** of an XML document is always based on a schema. In other words a schema is needed by the XML processor to automatically check that the document is valid, i.e. well-formed and following the specifications made in the schema. These specifications can be very extensive and detailed and traditionally they have been implemented in the application logic or, for example, in database definitions.

One objective of this work is to better understand how well XML schemas really can support various types of metadata in a real world application. Such metadata are, for example, the structure of the data, constraints for data values, reference information, definitions based on other schemas etc.

Another objective is to lay some ground and get hands-on experience for initiatives that are aiming to achieve something that is commonly referred to as **semantic web**. XML and XML schemas are often seen as basic building blocks of the semantic web as illustrated in the figure below (Bartolo, et al., 2005; Davies, et al., 2003).



2 MATML XML SCHEMA

Materials Markup Language (MatML) has been standardized by OASIS Standardization Working Group and serves as a general purpose markup language for the exchange of material properties data (Bartolo, et al., 2005). In this paper some approaches are discussed and some of them implemented in prototypes to study the possibilities to comply with and use MatML in order to support sharing of experimentally measured materials data.

The basic structure of MatML is outlined in Monma (2005) and documented in full detail by OASIS (2006).

MatML XML schema is publicly available at http://www.matml.org/downloads/matml31.xsd

3 MAT-DB XML SCHEMA[1]

The Institute for Energy has a long history of collecting experimentally measured materials data into IE's materials database (Mat-DB) that has in the past few years evolved into a web-enabled database system running under the ODIN portal (Nagy et al., 2005; Over et al., 2008).

Here is presented an experimental XML schema that defines the structure of an XML document capable of representing quite complex materials test data together with mandatory and non-mandatory metadata. Its purpose is to support IE's research activities dealing with future energy production technologies (Over et al., 2008). For example construction of the future nuclear systems often needs a lot of collaboration between international parties. They could take benefit from the Mat-DB XML schema to achieve interoperability between their heterogeneous materials testing databases to share their own data with that of others'.

Mat-DB XML schema is currently in its early prototype phase. Its design is based on metadata that is mostly stored in the Mat-DB database. Not the entire Mat-DB contents can currently be represented in an XML document that is

[1] Not be confused with the MatDB XML schema of NIMS's New Materials Center (Ashino, *et al.*, 2007). They are different although the scope of the schemas overlaps in many places.

valid against the Mat-DB XML schema: some metadata definitions are still missing and only one test type is currently covered by it.

There has been a lot of debate about the best way to define XML schemas: whether a data value is better to represent as an element or an attribute of an element. In the design of the Mat-DB schema we adopted a pragmatic approach where data is represented as an attribute if it logically relates to the element and has a one-to-one relationship with the element it belongs to.

Experimental Mat-DB XML schema is publicly available at the JRC IE ODIN portal at http://odin.jrc.ec.europa.eu/MatDB_XMLschema/matdb.html.

3.1 Notation

The following notation and naming convention is used in the rest of this document in XML language and in the graphical representations of it:

- XML that is embedded in the text is written in italics.
- XML element names are formed by concatenating parts of element names into a string. Each word begins with a capital letter (e.g. *Material*).
- Element names are in plural where more than one element can exist.
- XML attribute names are formed by concatenating parts of attribute names into a string. Each word begins with a capital letter except the first (e.g. *companyName*).
- Standard XML schema grouping constructs (Harold, 2004, p. 687)
    - The *xs:all* group requires that each element in the group must occur at most once, order is not important. It is visualized by the symbol 
    - The *xs:choice* group requires that any one element from the group should appear. It is visualized by the symbol 
    - The *xs:sequence* group requires that each element in the group appear exactly once, in the specified order. It is visualized by the symbol 
- Optional elements are shown with dashed lines.

Cardinalities are 1:1 unless otherwise mentioned (e.g. 0..∞) .

3.2 The structure of the schema

The structure of the schema is visualized in Figure 1 (annotations included):

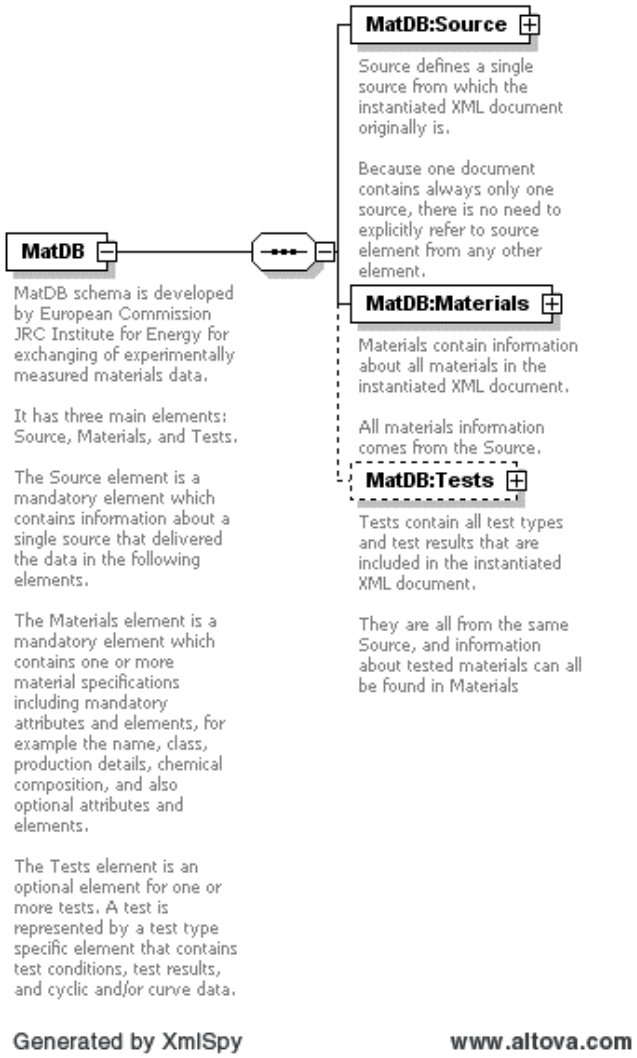


Figure 1. The high-level structure of the Mat-DB schema

An XML document that is valid against the schema consists of two mandatory elements and one optional element. The cardinality is relative to the element's parent, which is in this case the root element *MatDB*:

- *Source*: mandatory, cardinality 1:1
- *Materials*: mandatory, cardinality 1:1
- *Tests*: optional, cardinality 1:1

Note that the naming convention implies that both *Materials* and *Tests* elements may contain information for more than one occurrence whereas *Source* only contains information of a single source.

This high-level view of the schema also nicely shows how documentation can be incorporated into the document structure definition by using XML schema annotations.

More detailed view at the high level definitions of the schema reveal that in addition to the mandatory namespace definition two more namespaces are defined: MatDB namespace that refers to names that are defined in the Mat-DB

schema and MatML namespace that is used to refer to structures defined by the MatML schema. MatML schema itself is imported to be a part of the Mat-DB schema.

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:MatDB="http://odin.jrc.nl" **xmlns:MatML="http://www.matml.org"** targetNamespace="http://odin.jrc.nl" elementFormDefault="qualified" attributeFormDefault="unqualified">
<**xs:import schemaLocation="http://www.matml.org/downloads/matml31.xsd"**/>

Note, that the use of MatML inside Mat-DB schema is currently very limited and for demonstration purposes only. However, using this technique clearly demonstrates how schemas, like Mat-DB schema, can be extended using existing schema definitions.

3.2.1 Source element

The figure below presents the structure of the *Source* element. It consists of two elements:

- *Details*, mandatory, cardinality 1:1
- *ProjectResponsibilities*, optional, cardinality 1:1.

Both elements contain attributes (e.g. *sourceKey*, *sourceNumber*, and *name*) that always have a one-to-one relationship with their parent element. *Details* element also contains both simple sub-elements (e.g. *FullSourceTitle*) and complex sub-elements (e.g. *Address* and *Reports*)

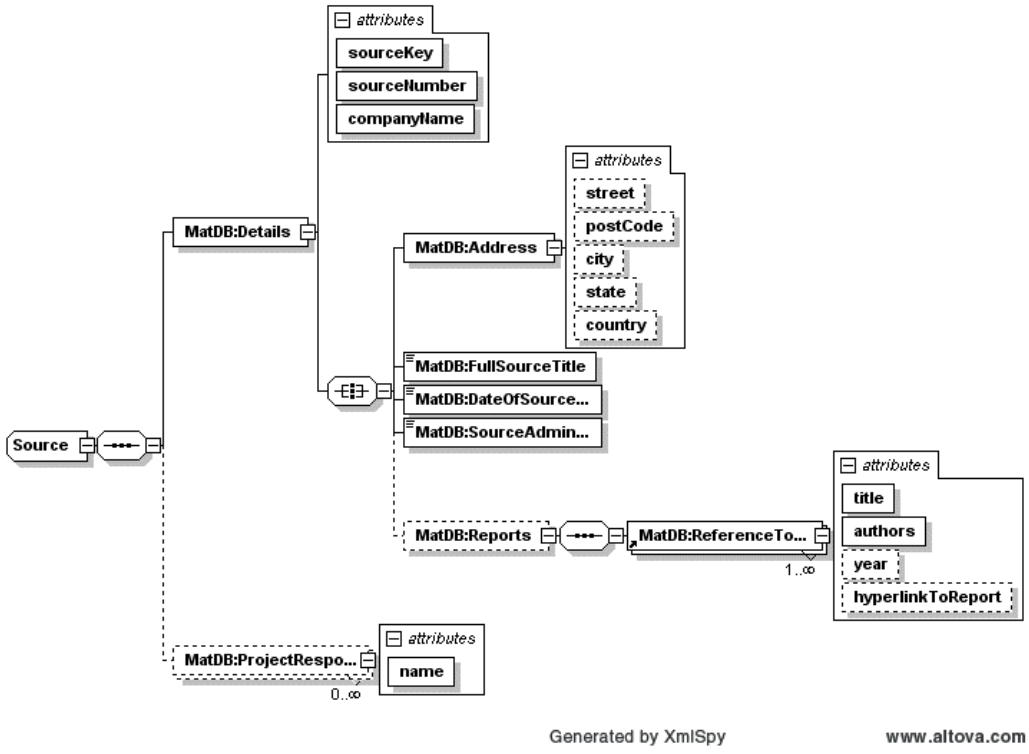


Figure 2. The structure of the *Source* element

Reports element shows a way of using a global element inside a schema: the element, if it exists, consists of one or more *ReferenceToReport*s that is a globally defined element. The same definition can be used by a reference from wherever appropriate, which makes the maintenance very easy.

3.2.2 Materials element

The mandatory *Materials* element (see figure 3) consists of one or more *Material* elements. Some new constructs are introduced in this element.

One of element's attributes is *materialKey*, which is defined as follows

```
<xs:attribute name="materialKey" type="xs:ID" use="required">
        <xs:annotation>
                <xs:documentation>Material key identifies the material in Mat-DB (RN4)</xs:documentation>
        </xs:annotation>
</xs:attribute>
```

Type *ID* is a type specified in the XML Schema definition itself. It must not appear more than once in an XML document as a value of this type and it also has some constraining facets that constrain its value space. In this schema its purpose is to uniquely identify the material in question, to make sure that a material is defined only once, and to enable references to the material using XML Schema definition's *IDREF* notation from, for example, specific test results.

The element *ChemicalComposition* is mandatory and consists of one or more references to a global element *Symbol*. This global element has four attributes, but more interestingly the value of the element itself refers to an extension called *ChemicalElementSymbol*, which is actually defined in the MatML schema and which consists of an enumerated list of all chemical element symbols (such as Au, Fe etc).

```
<xs:element name="Symbol">
        <xs:complexType>
                <xs:simpleContent>
                        <xs:extension base="ChemicalElementSymbol">
                           <xs:attribute name="minimumContent" type="xs:decimal"/>
                           <xs:attribute name="maximumContent" type="xs:decimal"/>
                           <xs:attribute name="meanContent" type="xs:decimal"/>
                           <xs:attribute name="remark" type="xs:string"/>
                        </xs:extension>
                </xs:simpleContent>
        </xs:complexType>
</xs:element>
```

This way we can be sure that any XML document that is valid against the Mat-DB schema, only uses references to chemical elements that really exist. This simple example shows nicely how we can utilize schemas defined elsewhere and offer our own schema structures to be utilized in a similar manner. This could probably be extremely useful in e.g. standardizing vocabularies.

The optional element *GrainSize* demonstrates the use of the choice structure. It consists of a single sub-element that is either *DirectionallySolidified*, *Duplex*, or *Isotropic*.

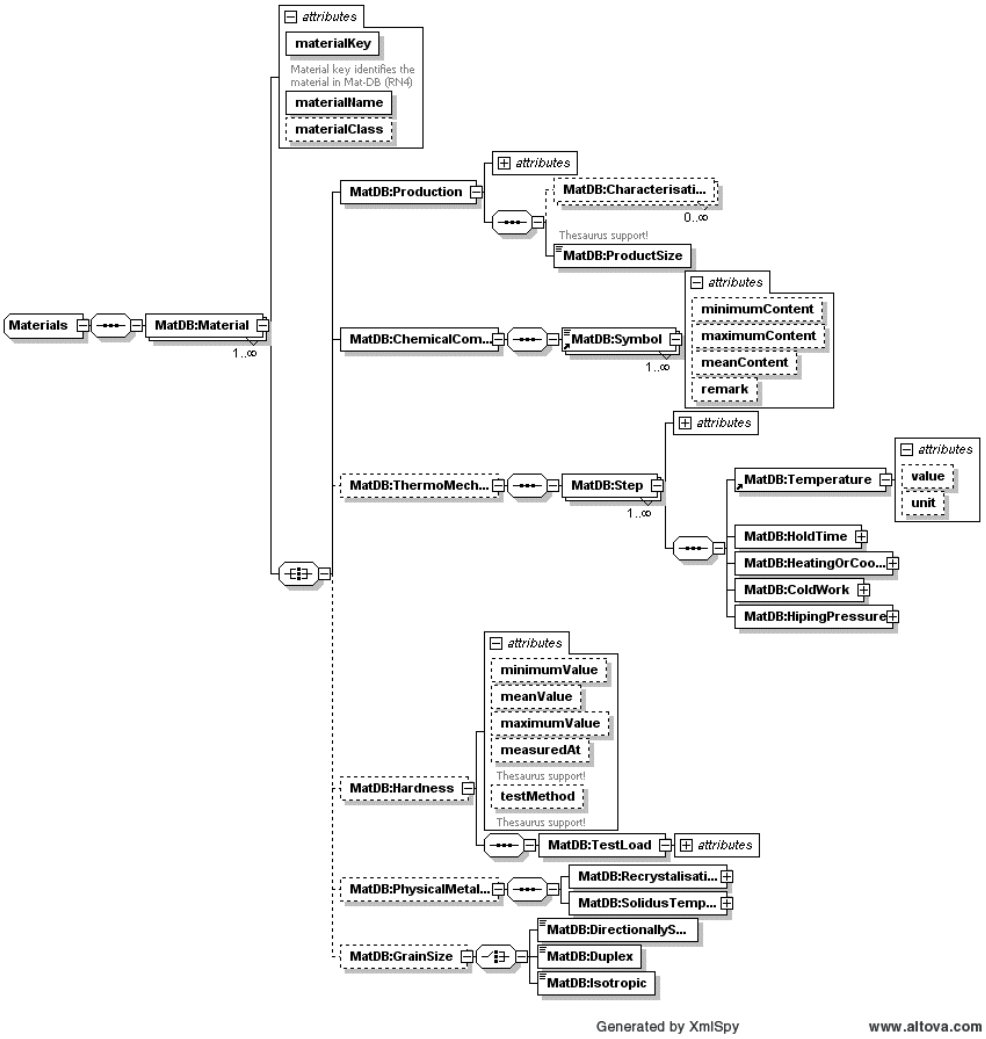


Figure 3. The structure of the *Materials* element

3.2.3 Tests element

The optional *Tests* element (see figure 4) consists of one or more *Test* elements. The optionality of *Tests* implies that materials data without any test data can also be represented using the Mat-DB schema.

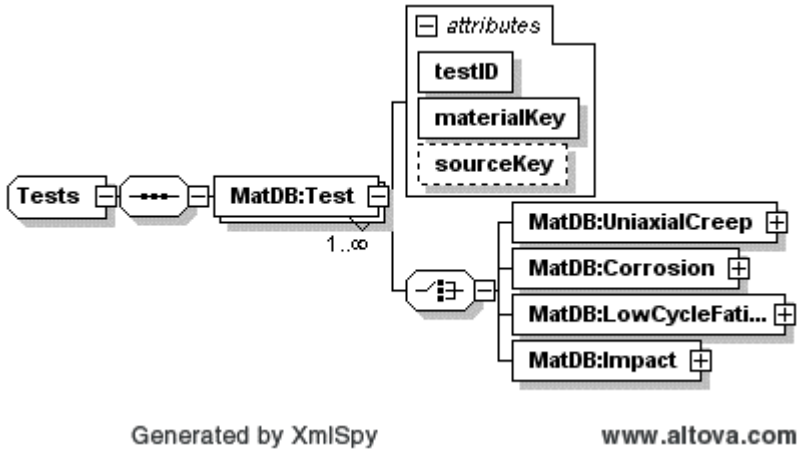


Figure 4. The structure of the *Tests* element

Each *Test* sub-element has two mandatory attributes:

- *testID*, which is an *ID* attribute type identifying the test, and
- *materialKey*, which is an *IDREF* attribute type referring to the *materialKey* attribute of the *Material* element.

Additionally each *Test* sub-element has one of several possible test types (e.g. uniaxial creep, uniaxial tensile, low cycle fatigue – strain control, etc). Only the uniaxial creep test type is currently implemented in the Mat-DB schema.

3.2.4 UniaxialCreep element

Figure 5 shows the basic structure of the *UniaxialCreep* sub-element. Its purpose is to define information that is relevant to the uniaxial creep test type. The element consists of constructs common to all test types and constructs that are specific to the test type.

Common constructs are

- the optional attribute *linkToDescription*, which is used to store an URL to the test type description.
- elements *TestCondition* and *Specimen*, (see chapter 3.2.5)

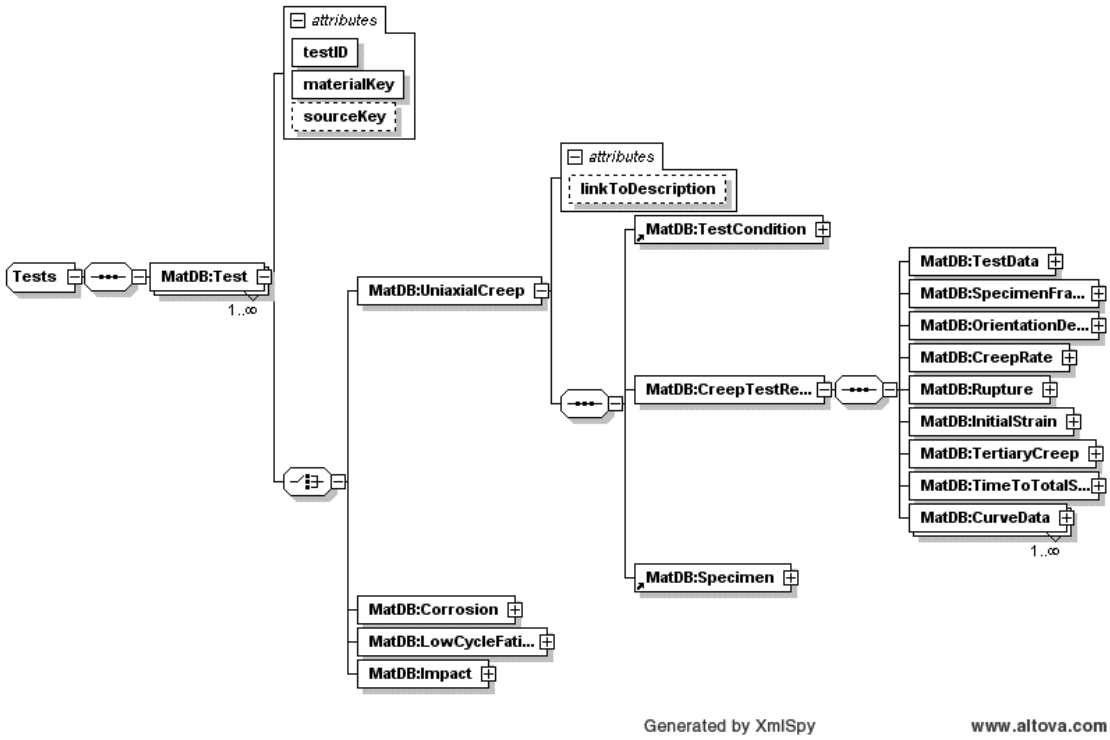


Figure 5. The structure of the *UniaxialCreep* element

Uniaxial creep test type specific content is grouped under the element *CreepTestResults*. Its structure is quite extensive, but some details may still be missing. It doesn't have any especially interesting constructs except the *CurveData* sub-element, which is used to store curve information with data points and which enables the processing of data points individually.

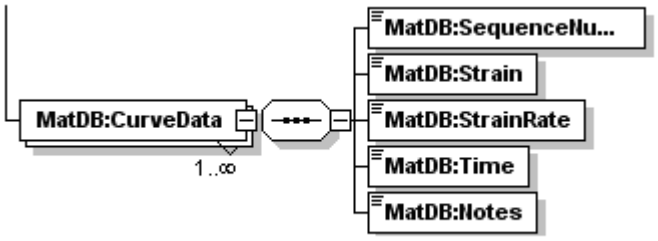


Figure 6. The structure of the *CurveData* element

The curve information for uniaxial creep test type consists of a sequence number, figures for strain, strain rate, time, and notes. An excerpt of a valid XML document's *CreepTestResults* element might therefore look as follows.

```
<CreepTestResults>
        <TestData qualityRemark=" - " specimenIdentifier="43220000" testControl="constant load" testMachine="
        - " testStandard=" - ">
                <Temperature unit="C" value="650"/>
                <Stress unit="MPa" value="139"/>
                <ReferenceToReport authors="Doe J" hyperlinkToReport=" - " title="Experimental report"
                year="2008"/>
        </TestData>
        --- a part intentionally removed ---
        <CurveData>
                <SequenceNumber>2</SequenceNumber>
                <Strain>4</Strain>
                <StrainRate> - </StrainRate>
                <Time>978</Time>
                <Notes> - </Notes>
        </CurveData>
        <CurveData>
                <SequenceNumber>3</SequenceNumber>
                <Strain>16.9</Strain>
                <StrainRate> - </StrainRate>
                <Time>1687</Time>
                <Notes> - </Notes>
        </CurveData>
```

3.2.5 Other elements and constructs

Two other important XML elements are defined: mandatory element *TestCondition*, that is used to specify test conditions, and mandatory element *Specimen*, which is used to specify the specimen used in the tests.

TestCondition information content is identical to all test types. It is implemented as a global element that can and will be referenced from all test types. Its purpose is to cover all possible test conditions which makes it quite general and which means that a single test type would only use a subset of the whole *TestCondition* element. This would mean that there are many optional sub-elements in the final *TestCondition* element. The current implementation does not take this into account.

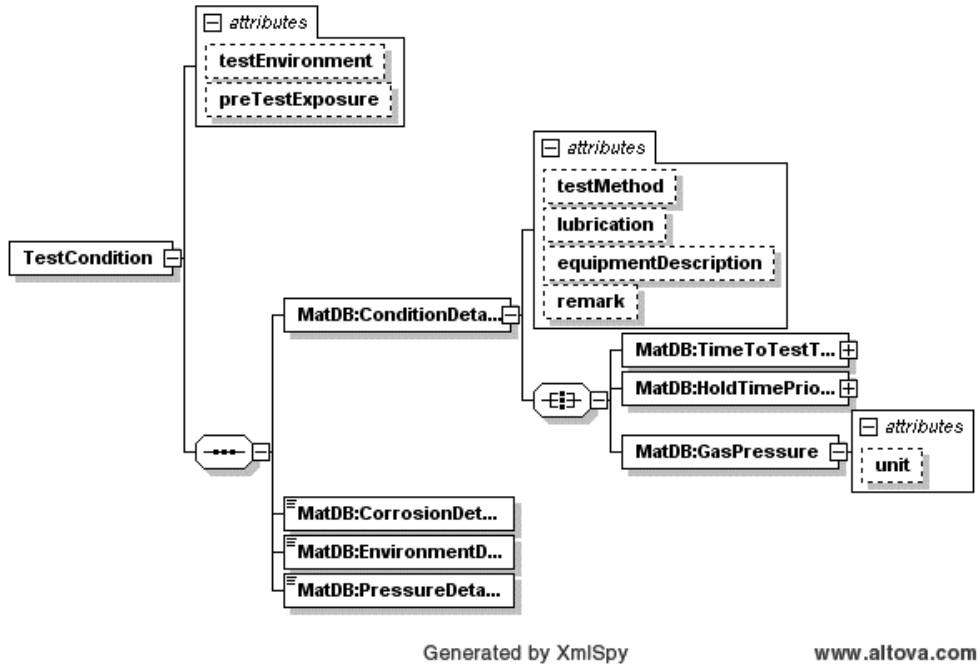


Figure 7. The structure of the *TestCondition* element

Also *Specimen* information content is identical to all test types and is implemented in the same way as *TestCondition* as a global element. Its purpose is to cover all the data that are needed to describe specimens. Again, the current implementation is by no means exhaustive.

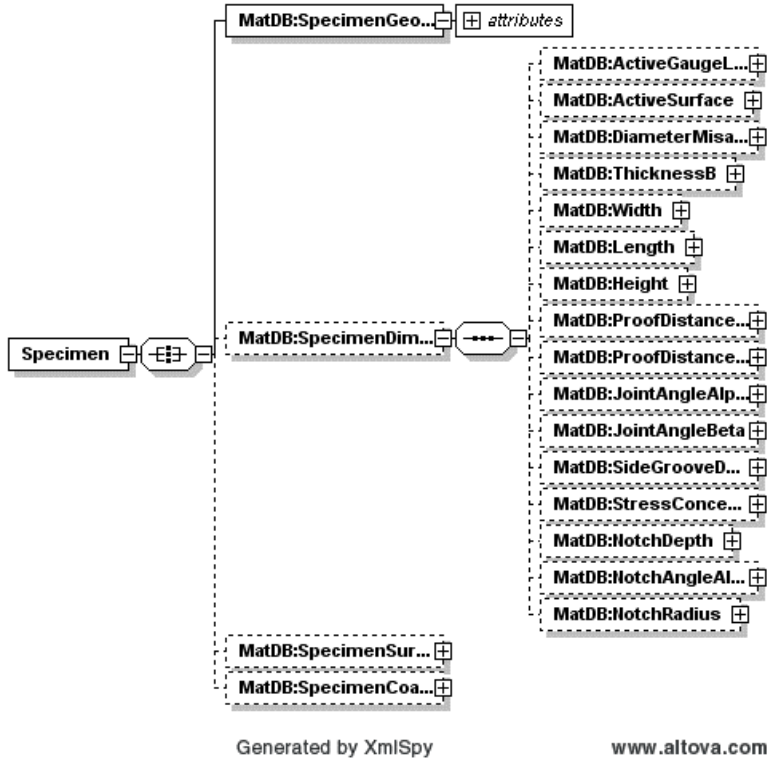


Figure 8. The structure of the *Specimen* element

Interestingly, both test conditions and specimens are addressed in the MatML *Metadata* element. Our current implementation differs from this and some further study would probably be needed in order to streamline these specifications and their use.

The current version of the Mat-DB XML schema has many other XML constructs, but they are basically repetition of the structures and ways of implementing things presented above and therefore not very interesting from the point of view of this work.

4 UTILIZATION OF THE MAT-DB XML SCHEMA

4.1 Technical environment

The technical environment consisted of tools already available so no special attention was paid to tool selection.

4.1.1 Database

The database is Oracle10g, but basically any database that is accessible via JDBC is possible to use. In the current implementation this is not important since Mat-DB is running on top of Oracle and only the export functionality is implemented.

4.1.2 XML tool

The schema editor is Altova's XmlSpy, which is a commercial product. It is used to create the schema, to validate XML documents, and to create the XSL transformations. For validations any XML tool can, of course, be used. Also for schema and XSL editing any tool can be used – even a text editor.

4.1.3 Development tools

Java programming language and Oracle JDeveloper10g were used to program the prototype implementation.

XmlSpy offers a nice feature of creating automatically Java program code to process XML documents based on the schema definitions. This feature was utilized. It creates an Altova-specific layer on top of the standard Document Object Model (DOM) interfaces specified by W3C (2004). However, the generated code is freely modifiable although generally it would be a much better idea to modify the schema specifications and re-generate. Its use is in no way restricted by Altova so the outcome is open source and free to distribute. Our experience is that the generated Java code is high-quality and easy to use especially when supported by a modern Java IDE.

4.2 Implementation

The current implementation is illustrated in the figure below.

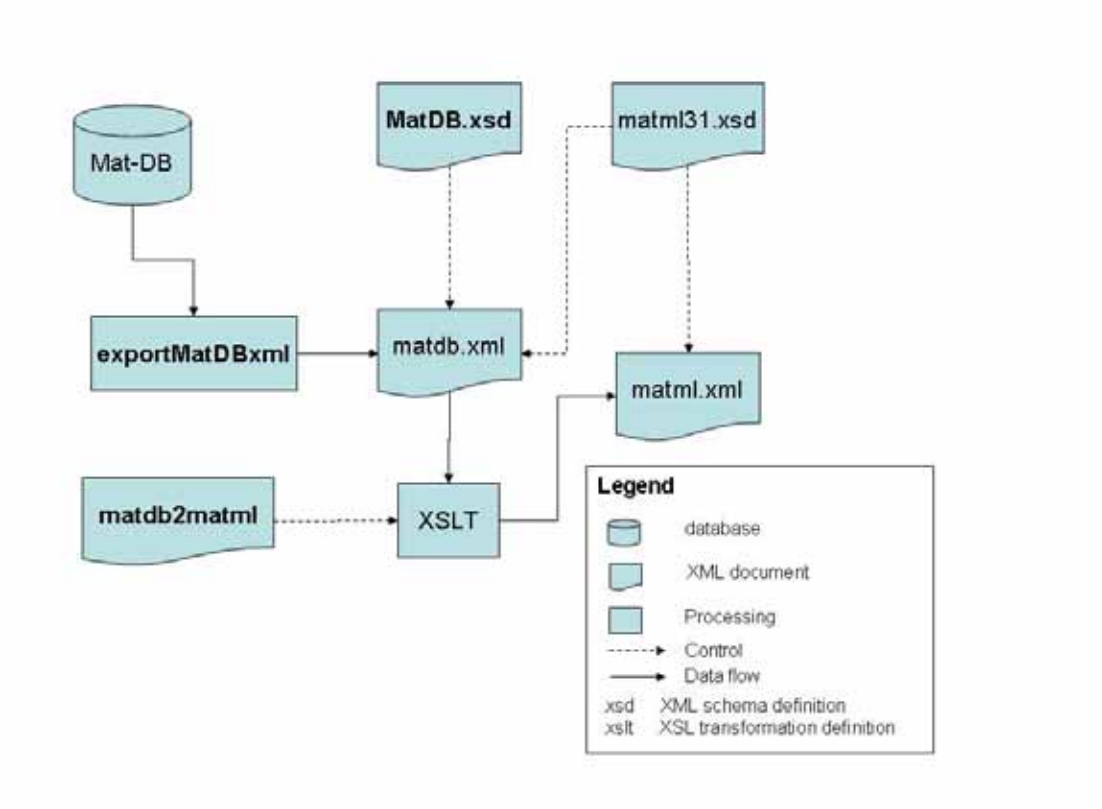


Figure 9. The architecture of the implementation

It consists of three major parts that were developed for this study:

- MatDB.xsd, which is the XML implementation of the Mat-DB schema described in chapter 3.
- exportMatDBxml, which is a Java implementation of a program that is used to export Mat-DB data from the database into a valid XML document.
- matdb2matml, which is an XSLT definition to transform a valid matdb.xml file into a valid matml.xml file.

This setup quite nicely demonstrates the capabilities and usefulness of the technologies selected for this study. Even though it is by no means complete, our assumption is that it gives a realistic picture of the possibilities and problems that are related to this kind of efforts in real world use cases, and that the experience gained is therefore valuable.

4.2.1 Schema development

Mat-DB schema development was done using XMLSpy schema editor. The schema is by no means complete at the moment, and only intended to gain some real-world experience with these technologies. Below are some of possibly most interesting experiences with XML schema development.

The utilization of pre-existing XML namespaces is a very elegant and powerful way of reusing development done elsewhere. This is demonstrated in our design by using MatML definitions for chemical elements.

The schema can be used to convey the documentation about the schema itself using annotations. These can be easily viewed in XML editors. Also, even more easily navigateable publication can be generated from the schema definition using normal editor features to produce documentation in the internet to be used by standard browsers. Therefore documentation is extremely easy to publish including referenced schemas if requested.

XML schema fits perfectly for purposes of defining and using a common vocabulary. Such needs are definitely many in the area of material research and in science in general. Instead of placing such information in databases or reference documents, one could publish these as parts of XML schemas in enumerated lists, or create a vocabulary schema. The first method is used by MatML to enumerate chemical elements and currencies. The vocabulary

schema is one which contains a single simple type definition with enumerated values and nothing else. This can then be included and used in other schemas by: <xsd:include schemaLocation = "…"/>.

Many sophisticated things that are possible to use with XML schemas were intentionally left out of this work. Indeed, designing a schema properly to model real world entities, their characteristics, and interrelationships is difficult and very well comparable to database design. XML schema language is a relatively new tool and one that has not yet been adopted very widely by the development community. This study, however, indicates that important development support can be achieved by utilizing a well-defined XML schema.

4.2.2 Java

Java program was generated automatically based on the schema implementation. The use of the generated code is quite straightforward, but in order to implement programs that really produce XML from database contents – like we did – or load XML data into a database, more effort is needed depending a lot on the complexity of the database.

Some notes on generated Java classes:

- Classes that handle member elements do not make distinction between mandatory and optional elements/attributes. Therefore by using generated Java classes it is possible to create non-valid documents. XML validation can, of course, be easily programmed.
- Iterators are automatically generated, which greatly helps the developer.
- When a member class is instantiated, all of its (sub)member classes are automatically instantiated as well.

Some advanced XML Schema features are not or not fully supported (Altova, 2007):

- content models (sequence, choice, all) are not enforced by generated classes
- default and fixed values for attributes not set or enforced by generated classes
- attribute nillable="true" and xsi:nil
- uniqueness constraints.

4.2.3 XSLT

Extensible Stylesheet Language (XSL) is used to define rules for transforming an XML document into another XML document. As defined in W3C (2007) “a transformation in the XSLT language is expressed in the form of a stylesheet, whose syntax is well-formed XML”. Essentially XSL consists of two languages: one for transformations, the other for formatting. In this paper we will only deal with the transformation language, XSLT.

We have defined a simple experimental XSLT to transform XML documents that are valid against the Mat-DB schema to another XML document that is valid against the MatML schema (see figure 9). This transformation implements a mapping of some Mat-DB elements to their MatML counterparts. The mapping is very rudimentary and only covers some material elements along with some metadata.

As defined in W3C (2007) “a transformation expressed in XSLT describes rules for transforming zero or more source trees into one or more result trees”. These rules are based on templates that define in which situations the template is applied, and what transformations will occur. XSLT template is analogous to a method in an object-oriented programming language. It allows a single XSL transformation[2] to be broken into multiple logical units, each of which performs a specific transformation.

Below are two excerpts from our XSL transformation definition.

[2] A synonym for ‘XSL transformation’ would be ‘XSL stylesheet’. In this context it is more logical to talk about XSL transformations since no real styles are applied. This terminological feature is also implemented in the XSLT specification: both <xsl:transform> and <xsl:stylesheet> tags can be used in the root element of the XSLT document (Harold, 2004, p. 429).

The first template shows how to match an XML element called *MatDB* (the root element of Mat-DB schema), and how to apply two separate templates to the result tree. Essentially these are used to create *Material* and *Metadata* elements in the MatML representation.

```
<xsl:template match="matdb:MatDB">
        <MatML_Doc xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="http://www.matml.org/downloads/matml31.xsd" >
                <xsl:apply-templates select="matdb:Materials"/>
                <xsl:apply-templates select="matdb:Source"/>
        </MatML_Doc>
</xsl:template>
```

The second is more complex and shows the mapping of the *ChemicalComposition* of Mat-DB XML schema to MatML's quite different structure. Mat-DB *ChemicalComposition* consists of one or more *Symbol*s with some attributes. MatML's *ChemicalComposition* can consist one or more *Element*s which have a much more complex structure (see figure 10). The XSLT mapping has to convert between these two structures.

```
<xsl:template match="matdb:ChemicalComposition">
  <xsl:for-each select="matdb:Symbol">
    <xsl:if test="@maximumContent &gt; 0">
      <Element>
          <Symbol><xsl:value-of select="."/></Symbol>
          <Concentration>
            <Value format="float"><xsl:value-of select="@maximumContent"/></Value>
            <Units><Unit><Name>%</Name></Unit></Units>
            <Qualifier>"max"</Qualifier>
          </Concentration>
          <Notes><xsl:value-of select="@remark"/></Notes>
      </Element>
    </xsl:if>
    <xsl:if test="@minimumContent &gt; 0">
      <Element>
          <Symbol><xsl:value-of select="."/></Symbol>
          <Concentration>
            <Value format="float"><xsl:value-of select="@minimumContent"/></Value>
            <Units><Unit><Name>%</Name></Unit></Units>
            <Qualifier>"min"</Qualifier>
          </Concentration>
          <Notes><xsl:value-of select="@remark"/></Notes>
      </Element>
    </xsl:if>
    <xsl:if test="@meanContent &gt; 0">
      <Element>
          <Symbol><xsl:value-of select="."/></Symbol>
          <Concentration>
            <Value format="float"><xsl:value-of select="@meanContent"/></Value>
            <Units><Unit><Name>%</Name></Unit></Units>
            <Qualifier>"mean"</Qualifier>
          </Concentration>
          <Notes><xsl:value-of select="@remark"/></Notes>
      </Element>
    </xsl:if>
  </xsl:for-each>
</xsl:template>
```

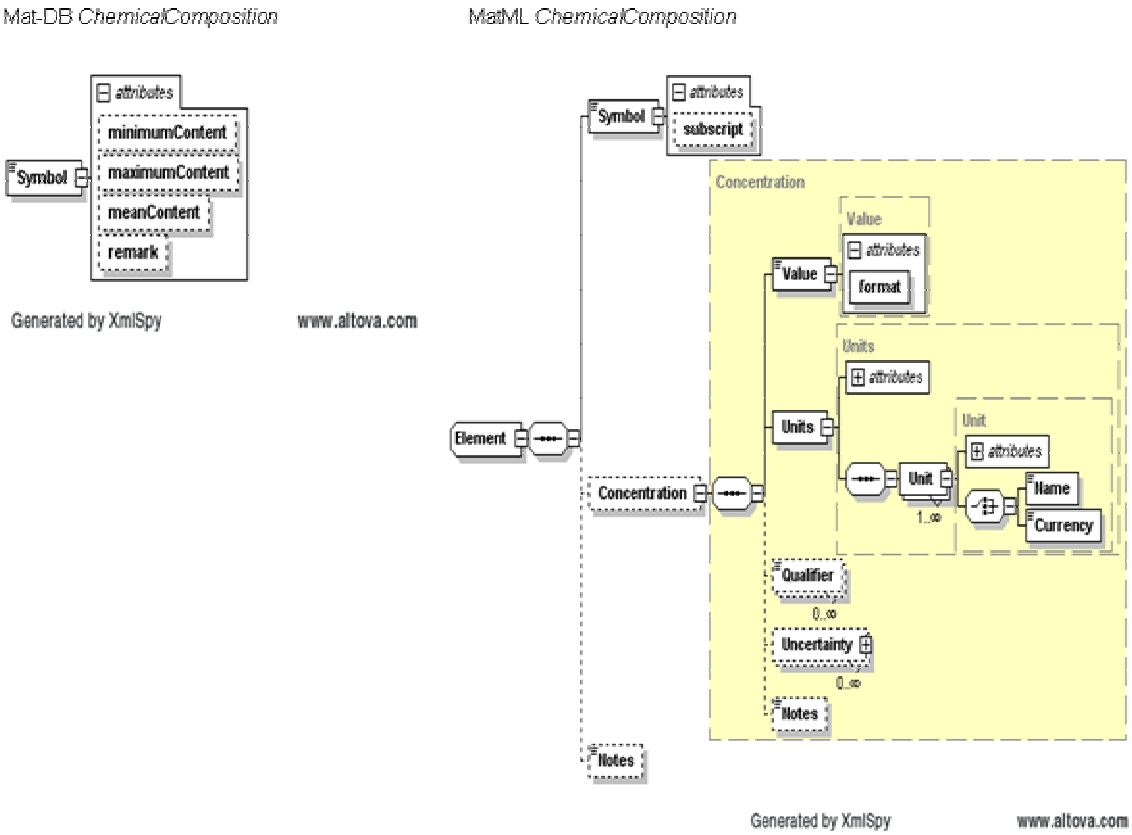


Figure 10. The different chemical composition structures of Mat-DB and MatML schemas

Our experience gives good indication that it is possible, although quite tedious to create working transformations between complex XML documents using XSLT. XSLT as such doesn't guarantee that the result document is valid against a specified schema, but combined with the schema-awareness capabilities of XSLT 2.0 such a functionality could be achieved (Gandhi, 2008). This would ensure that both the input and output documents are valid, which of course is a big data quality issue.

The full application of XSLT to handle all possible Mat-DB elements for transformations will remain a possible development task in the future if needs exists in real life.

5 CONCLUSIONS

It is obvious that the current version of Mat-DB XML schema is not sufficient to serve as a general purpose schema. However, even these preliminary experiments show how powerful XML schemas can be in designing and visualizing complex XML document structures.

Especially interesting is the ability to reuse schema designs made by others, and to offer our own schema designs for others to use. MatML represents in this respect a direction to look at: MatML definitions should be used to describe all source and material properties related information. However, it is not yet quite clear whether MatML is capable of presenting this information with respect to the needs of experimentally measured test data.

Two important areas from the Institute for Energy's point of view are clearly not sufficiently covered by MatML. The first is representing complex test conditions, such as irradiation pre-test exposures, and the other is representing

complex test results, such as thermo-mechanical fatigue. Further elaboration of the Mat-DB schema could possibly provide test result related information structures, which could then extend MatML definitions and be used if needed by the user. This could be an interesting addition, and definitely one that needs cooperation between several parties.

Attaching metadata descriptions, enumeration information, different patterns, data types, and constraints into XML document instances using well-defined XML schemas would also be an interesting possibility when interoperability and high quality data sharing are objectives. This is probably a prerequisite for joining in materials networks such as described in Ashino *et al*. (2007) and Kaji *et al.* (2004) to provide data for others to use.

A well-designed schema can also serve as a solid basis for application development. In this study we showed that tools exist that enable automatic generation of program code based on a schema definition to produce and handle corresponding XML document instances. In this case the experience has been very promising even though the generated code was only used to create an XML document from JRC-IE's Mat-DB database contents. It is seen by the experts that markup language development should be done in an evolutionary manner and that "there is an ongoing tension between static data exchange standards and the dynamic nature of science, science research and scientific data" (Bartolo *et al.* 2005). Efficient and high quality code generation could tremendously help in shortening the development cycle and enabling more dynamism.

The current Mat-DB schema needs to be extended to complete existing element definitions, to cover additional test types, and to verify that current elements and attributes are correct. Also a proper and more complete mapping of related elements and attributes between MatML and Mat-DB schemas needs to be done.

Our experiments clearly demonstrated the need to more extensively validate the contents of our Mat-DB database against both the experimental Mat-DB and especially MatML schema. The data are often incomplete and erroneous, and validation of data contents is therefore important. This is considered generally as a good idea by Monma (2005) and also addressed by Westbrook (2003).

Finally, some obvious further development tasks to enable true data exchange between databases could be

- to implement data upload from a valid XML document into the Mat-DB database,
- to create transformations for materials data between MatML and Mat-DB schemas, and
- to implement these using web services technologies for easier access in the internet.

Others, of maybe more general interest, could be the utilization of common vocabularies, supporting metadata generation with these techniques, and even taking steps towards the semantic web by filling the gap between existing legacy systems and future systems.

6 REFERENCES

Altova, (2007) *Altova XML Spy 2007 Enterprise Edition User & Reference Manual.*

Ashino, T., Oka, N., (2007) Material database syndication with RSS. *Data Science Journal, Volume 6, Supplement 2*, December 2007.

Bartolo, L.M., Cole T.W., Giersch, S., & Wright, M. (2005) *NSF / NSDL Workshop on Scientific Markup Languages*. Conference Report. D-Lib Magazine. Vol 11 Number 11.
http://www.dlib.org/dlib/november05/bartolo/11bartolo.html (accessed May 5, 2008).

Davies, J., Fensel, D., & Van Harmelen, F. (2003) *Towards the Semantic Web. Ontology-Driven Knowledge Management*. John Wiley & Sons Ltd, 288 p.

Davies, J., Studer, R., & Warren, P. (2006) *Semantic Web Technologies. Trends and research in ontology-based systems*. John Wiley & Sons Ltd, 312 p.

Gandhi, M. (2008) *Schema-aware processing with XSLT 2.0*. IBM developerworks library.
http://www.ibm.com/developerworks/xml/library/x-schemaxslt.html (accessed May 16, 2008)

Harold, E.R. (2004) *XML 1.1 Bible*, 3rd edition. Wiley Publishing Inc, 1022 p.

Kaji, Y., Tsuji, H., Fujita, M., Xu, Y., Yoshida, K., Mashiko, S., Shimura, K., Miyakawa, S., & Ashino, T. (2004) Development of a knowledge based system linked to a materials database. *Data Science Journal, Volume 3*, 21 July 2004.

Monma, Y. (2005) Implementing MatML toward sharing materials information on the Net. A presentation at the *JCDL Workshop June 10th 2005*, Denver, USA.

Nagy, M., Over, H-H., & Wolfart, E. (2005) XML related data exchange from the test machine to a web-enabled Mat-DB. *Data Science Journal, Volume 4*, 8 December 2005.

OASIS (2006) *Material Markup Language Public Review Draft 01*. http://docs.oasis-open.org/materials/ (accessed May 6, 2008).

Over, H-H. & Ojala, T. (2008) The web-enabled materials database of the European Commission with its XML related data entry part and integrated analysis tools to support GenIV nuclear power plant development. Proceedings of *PVP2008-ICPVT-13 2008 ASME Pressure Vessels and Piping Conference* July 27-31, 2008, Chicago, IL, USA.

W3C (2004) *Document Object Model (DOM) Level 3 Core Specification*. http://www.w3.org/TR/DOM-Level-3-Core/ (accessed May 6, 2008).

W3C (2007) *XSL Transformations (XSLT) Version 2.0*. W3C Recommendation 23 January 2007. http://www.w3.org/TR/xslt20/ (accessed May 20, 2008).

Westbrook, J.H. (2003) Materials data on the Internet. *Data Science Journal, Volume 2*, 25 November, 2003.