A Review on XML Document Schemas and Methods for Schema Design

Eliisa Jauhiainen, Anne Honkaranta
University of Jyväskylä
raelurja@cc.jyu.fi,
anne.honkaranta@it.jyu.fi

Abstract

Due to wide-spread use of XML in applications and the future arrival of XML to the popular office software XML shall become the standard de facto for organizational documents. In order to realize novel, intelligent XML-based document applications organizations need to pay attention on schema desing. This paper reviews three differing schema languages and two potential methods for document-oriented schema design. The paper also exhorts that schema desing for document-oriented XML-documents requires differing methods than those applicable for data-oriented XML document type schema design.

1. Introduction

Organizations use documents as a means for information management [Salminen, Lyytikäinen & Tiitinen, 2000, 624). *Document* is a unit of content meant for human perception with one or more external representations [Salminen 2003]. A digital document is electronically recorded information flexibly structured for human consumption though documents dependent upon a computers for their use [Murphy, 2001]. A *structured document* is a digital document, which information is defined and presented by XML (Extensible Markup Language; [Bray 2000] [Salminen 2003].

XML was initially developed for document management, but it is becoming increasingly used for storing and exchanging all kinds of data on the internet [Elmasri 2002]. Even though many kinds of documents and messages may utilize XML, one may make a distinction between "document-oriented" and "data-oriented" XML. "Document-oriented" XML may also be called as "narrative" or "content-oriented" XML. Correspondingly "data-oriented" XML may be considered as "transactional" or "transaction-oriented" XML. These two are not rigid categories since there is no technical difference between these two.

However the distinction is important for XML system analysis and implementation. [DuCharme 2004]

XML is becoming a standard to represent any kind of content in any application domain, because it is able to represent any kind of structured or semi-structured documents, as papers, manuals, technical reports, web pages, database schemas, style-sheets, etc. [Psaila 2000]. XML has been widely adapted to enterprise system integration, as e-Business format, and for Web Services. Along the introduction of XML into common-purpose office software such as Microsoft Office 2003 Professional [Goldfarb and Walmsley 2004] and becoming Microsoft Office 12 and OpenOffice the office documents become XML-based. The growing use of XML shall make large amount of heterogeneous XML documents available. Each organization or application creates its own document applications by defining schemas to meet their requirements.

Document schema may describe the organization of a document into graphical constituents like sections, paragraphs lists, and figures [Power 2003]. Logical structure of a XML document describes the subjects covered on a document. For example, an agenda may consist of logical components of "list-of-participants", and "time-of-meeting". In addition to logical and layout structure, each XML document may have also a physical structure. Physically the document is composed of units called entities, which are storage units that are identified by entity name and which have some kind of content. The logical structure of the XML document is hierarchical i.e. tree-structured [Elmasri 2002] and this structure is described by a schema. A schema for a class of documents, such as DTD [Maler and El Andaloussi 1996] or XML Schema [Fallside and Walmsley 2004] describes allowed structure, or the order and names of elements and attributes for that peculiar class of XML documents. [Boukottaya 2004]

This paper is focused on the "document-oriented" XML. These kinds of documents are, for example, agendas and reports. The paper reviews three distinctic schema languages – DTD [Maler and El Andaloussi 1996], XMLSchema [Bray 2000] and RelaxNg [Clark 2001], and compare their features. We also introduce two schema design methods; the Maler and El Andaloussi method [Maler and El Andaloussi 1996], and the Kennedy method [Kennedy 2003] and discuss their similarities and differences. The aim is to provide information for those considering the implementation of XML for office documents.

The Maler and El Andaloussi method [1996] is generally recognized as the "best practice" for DTD design [Thompson 2000]. However, DTDs differ from schemas. When new implementation techniques appear, methods in use are usually outdated or provide a poor match with practices and platforms in use [Rossi 2000]. Therefore this paper also considers if the Maler & El Andaloussi method (developed for DTD design) is still the best practice for designing schemas?

The paper is organized as follows. Chapter 2 reviews the schema languages. Chapter 3 describes the two methods for schema design, namely the Maler & El Andaloussi method and the Kennedy method. Chapter 4 discusses the findings and sums up the paper.

2. Schema languages

According to Marinelli, Sacerdoti Coen & Vitali [2004, 164], schema languages can be divided in two kinds of types:

- Grammar-based languages
- Rule-based languages

XML Schema, RELAX NG, and also DTD belong to grammar-based languages. Grammar-based schema languages are created by document engineers who construct context-free grammar according to top-down production rules in a specified form. An example of rule-based schema language is Schematron. A rule-based schema language lists the rules that XML document has to meet, providing either an open or closed specification. [Marinelli, 2004, 164-165] Schematron is intented to be used for checking up specific parts of document markup for applications by using for example *assertions*. It is not intented to be utilized as a schema for a whole document. Therefore we do not consider the Schematron in this review.

Following subsections introduce the most popular grammar-based schema languages; XML DTD, XML Schema and RELAX NG.

2.1. DTD

The first and probably the best known schema language for XML documents is DTD (Document Type Definition). XML DTD is almost a direct derivation of its counterpart in SGML, only simplified from its original form. [Marinelli 2004]

DTDs provide a simple, document-specific grammar for schema [Jelliffe 2001]. DTD's main building blocks consist of an *element* and an *attribute*. The real world is represented with hierarchical element structures [Lee 2000]. DTDs also provide a sophisticated regular expression language for imposing constraints on elements and subelements (i.e. *content model*) However DTD is limited in the control of attribute and element values. [Marinelli 2004]

DTDs have been originally introduced for validating SGML structures and they have been found useful in a publishing domain where most rules deal with the explicit structure of the document, not so much with the contents of the element values. [Marinelli 2004]

2.2. XML Schema

XML Schema is a W3C recommendation aimed to replace DTDs as the official schema language for XML documents. The first major improvement compared to DTDs is that XML Schema is declared by using XML-based syntax. This made the readability of the schema worse but improved the flexibility and automatic processability of the schema. [Marinelli 2004]

XML Schema takes the namespace, not the document, as the fundamental unit of interest in validation [Jelliffe 2001]. It borrows many features from object oriented languages. W3C XML Schema is a strongly typed schema language. It

is generally considered complex. [van der Vlist 2002] XML Schema is the most wildely used among all the schema languages [Bex 2004].

XML Schema's expressive power is higher than that of DTD's. XML Schema has additional features over DTDs, like namespaces, import facilities and possibility to define elements as data types. In practice only small amount of schemas use these features. This means that majority of the schemas found in web are equivalent to a DTDs even though the modelling power of the XML Schema is notably higher. [Bex 2005, 712]

2.3. RELAX NG

RELAX NG [Clark 2001b] is a schema language based, developed by a small working group at the OASIS organization [Jelliffe 2001]. It is built on two preceding schema languages, TREX (Tree regular expression for XML) and RELAX (Regular language of description for XML). It was first published in March 2000 as a Japanese ISO Standard (JIS) Technical Report written by Murata Makoto. [van der Vlist 2002] The central concepts of RELAX NG are the *patterns*. While in DTD a content model is an expression over elements, in RELAX NG a pattern is an expression over elements, text nodes and attributes. [Marinelli 2004] A RELAX NG schema is itself an XML document [Clark 2001b].

RELAX NG represents both schemas and their instances by using an abstract data model. (Clark 2001b). The RELAX NG schema is a pattern that a document must match, and other patterns may appear as components of the main schema pattern. In RELAX NG, an attribute list is apart of its content model, which makes it possible to specify dependencies between elements and attribute presence and even between element and attribute values. This feature is a significant difference between RELAX NG and any other schema language. [DuCharme 2004] One of the limitations of the RELAX NG is its inability to define default values for element and attributes [Marinelli 2004].

RELAX NG is considered to be simpler than W3C XML Schema and that is why it might be a serious alternative for it. It even seems to be technically superior to W3C XML Schema, but the support by the software vendor and XML developers has not reached to the level of W3C XML Schema.

2.4. Comparison of Schema Languages

Lee and Chu [2000] have compared six schema languages including DTD and XMLSchema. We appended the Lee and Chu's (ibid.) comparison with RELAX NG properties. Following tables provide comparison of DTD, XML Schema and RELAX NG with respect to properties of schemas, datatypes and attributes, and properties of elements.

Table 1. Comparison of four schema languages concerning properties of schemas, datatypes and attributes (XSD = XML Schema; RNG = RELAX NG).

	XML Schema			Datatype		Attribute		
	XML syntax	Name-space	Include & import	Built-in type	User-defined type	De-fault value	Choice among attri-butes	Optio-nal vs. requi-red
DTD	No	No	No	10	No	Yes	No	Yes
XSD	Yes	Yes	Yes	44	Yes	Yes	No	Yes
RNG	Yes	Yes	Yes	2 + 44	-	No	Yes	Yes

Data types can be categorized into two classes: *simple* types and *complex* types. A simple type cannot have element content nor carry attributes, but complex can. The support of complex types varies between schema languages. Built-in type is either a primitive or derived simple type provided by the schema language specification. User-defined types are defined by schema designers. In Table 1 the number of datatypes supported by RelaxNG is 2+44. There are two built-in datatypes in RelaxNG but also all 44 simple datatypes used by XMLSchema may be imported.

DTDs have a built-in set of data types, which can be applied only to the values of attributes and not to the element content, which in DTD is either parseable data (PCDATA), non-parseable data (NDATA) or EMPTY. In RELAX NG there are two major differences. First RELAX NG allows data types to be specified uniformly for both attribute values and element content. Secondly RELAX NG decouples the schema language from the set of data types. [Clark 2001a]

Table 2 provides a comparison of the schema languages with respect to their use of elements.

Table 2. Comparison of four schema languages concerning properties of elements. (XSD = XML Schema; RNG = RELAX NG).

	ELEMENT						
	Default value	Content model	Ordered sequence	Unord. sequence	Choice among elements	Min & Max occur-ence	Open model
DTD	No	Yes	Yes	No	Yes	Partial	No
XSD	Partial	Yes	Yes	Yes	Yes	Yes	No
RNG	No	Partial	Yes	Yes	Yes	Yes	No

In Table 2 the default value of the element concerns both simple and complex default values. In XML Schema the content model concerns the fact that element content model can be empty, text, element or mixed (text + element). [Lee 2000]

The element property of "open model" on the right-hand column means that an open content model enables additional elements or attributes to be present regardless they are not declared in the schema. In Schematron, the content model is open by default. [Lee 2000, 6-7]

Following example describes a content model as DTD and its counterparts in XML Schema and RELAX NG. The example considers a schema for a memo document type, which consists of one or more items to be discussed in a meeting. Each item has a title, reference for the department and contents. Content is build up with one of more paragraphs and a motion. After the meeting it is possible to add a decision element.

```
<?xml version=”1.0” encoding=”ISO-8859-1” ?>
<!ELEMENT memo (item+)>
   <!ELEMENT item (department-ref*, title, content)>
<!ATTLIST item id ID #REQUIRED>
   <!ELEMENT department-ref EMPTY>
<!ATTLIST department-ref id IDREF #REQUIRED>
   <!ELEMENT title (#PCDATA)>
<!ELEMENT content (paragraph+, motion, decision?)>
<!ELEMENT paragraph (#PCDATA)>
<!ELEMENT motion (#PCDATA)>
<!ELEMENT decision (#PCDATA)>
```

Following XML document is valid with the the DTD presented above:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<memo>
<item id="all">
<department-ref id="all"/>
<title>Nomination for the CEO</title>
<content>
   <paragraph>Authority memebers of the organization have familiarize
themselves with the applications of the applicants.</paragraph>
   <paragraph>According the statements of the authority members, it is noted
that Jane Doe is the only qualified applicant for the post of the
CEO.</paragraph>
<motion>It is proposed that Jane Doe is nominated as a CEO.</motion>
<decision>Accordant with the motion.</decision>
</content>
</item>
</memo>
```

Table 3 provides both XML Schema and RELAX NG schema matching the structure of the DTD represented above.

Table 3. Example of memo-element from memo DTD in XML Schema and RELAX NG schema languages.

XML Schema	RELAX NG
<?xml version=1.0" encoding="ISO-8859-1" ?> <xs:schema xmlns:xs="http://www.w3.org/2001/ XMLSchema"> <xs:element name="memo"> <xs:complexType> <xs:sequence> <xs:element name="item" maxOccurs="unbounded"> <xs:complexType> <xs:sequence> <xs:element name="department-ref" minOccurs="0" maxOccurs="unbounded"> <xs:complexType> <xs:attribute name="id" type="xs:IDREF" use="required"/> </xs:complexType> </xs:element> </xs:schema>	<?xml version=1.0" encoding="ISO-8859-1" ?> <grammar xmlns="http://relaxng.org/ ns/structure/1.0" xmlns:a="http://relaxng.org/ns/comp atibility/annotations/1.0" datatypeLibrary="http://www.w3.org /2001/XMLSchema-datatypes"> <start> <choice> <ref name="memo" /> </choice> </start> <define name ="memo"> <element name="memo"> <oneOrMore> <ref name="item" /> </oneOrMore> </element> </define> </grammar>

DTD nor XML Schema provides a choice between attributes, but in RELAX NG this kind of choice is possible to make. For example if element "student" has two attributes "department" and "major", the choice between these attributes may be defined as follows:

<element name="student">
<choice>

<attribute name="department" />
<attribute name="major" />
</choice>
</element>

This imposes that element "student" must have either a "department" attribute or a "major" attribute, but not both of them.

DTD and XML Schema have both same strengths and weaknesses, which are broadness as strength and complexity as weakness. RELAX NG's strength is its expressiveness, but its weakness is its limited focus [Jelliffe 2001].

As seen above, a DTD schema is easier to read than XML-based schema definition even though DTD itself does not adhere to XML-syntax. The example also shows that the structure of "document-oriented" XML document is usually hierarchial, i.e. contains a number of nested elements which is comonly not the case with "data-oriented" XML documents such as those storing content for application integration. In a case of "document-oriented" XML document, the structure also contains more variance. There are, for example, optional or repeating elements or element groups whereas "data-oriented" XML data is usually organized according to table structures of relational databases or as application-to-application messages having one root element and a number of child elements that occur exactly once. These differences exhort that desing for such a differing kinds of schemas should also have differing approaches.

3. Methods for schema design

This section describes the two potential schema design methods; the Maler and El Andaloussi [1996] method and the Kennedy method.

3.1. The Maler & El Andaloussi method

Maler and El Andaloussi [1996 methodology has been considered as a best practice on the field (Maler & El Andaloussi 1996, Thompson 2000). The text in this subsection follows the content of the Maler & El Andaloussi "Developing SGML DTDs" book [1996]. The basic idea of the method is to study the text, model the components and finally to define a DTD.

The methodology can be divided in three phases, which are

- Document analysis
- Document type modelling
- Producing a document analysis report

Document analysists should be familiar with basic XML concepts and the Tree Diagram notation. Especially important is to learn to identify the differing kinds of components of XML documents, which are:

- Content-based components (for example addresses, quantities, numbers etc.)
- Structural components (for example paragraphs, lists etc.), and
- Presentational components (for example phrases, regions etc. with special font or size).

First three steps of the method contain the document analysis. They are as follows:

1. Identifying potential components
2. Classifying components
3. Validating the needs

First task is to identify all the potential semantic components of the document. All found potential components should be named in an unique way. Every potential component should also have an example of its use recorded. After step 1 there should be a list of all the potential components.

In step 2 the *potential components* identified are classified. Classification can be based on content-based and structural similarities. After step two the classes of the components should be marked on the component list.

In the third step the existing other schemas for similar kinds of document types are inspected in order to cross-analyze the design and identify useful portions of design. It is important that this step is conducted after the component identification and classification, so earlier works do not effect on the search of potential components. Every domain and its documentation is unique and one must understand it in terms of finding existing logical components.

The next phase of the method is document type modeling. During this phase the seven remaining steps of the method are conducted. These steps are:

1. Selecting semantic components
2. Building the document hierarchy
3. Building the information units
4. Building the data-level elements
5. Populating the branches
6. Making connections (links, special features, entities)
7. Validating the design.

In step 4 the component lists are revisited. The aim is to drop out unnecessary components. Finally the list of accepted classified components should be made.

Step 5 considers modelling the components into *document hierarchy* model (Figure 1), which represents the class of components building the upper part of the document type. Hierarchy ends where “unspecified text” begins to appear. These parts are represented in a “cloud” figures.

In Figure 1, components are depicted by rectangles. Three dots underneath components depict that that the content model is represented in another diagram. Symbol “+” indicates that the component has to occur at least once and it may be repeatable whereas “?” defines that the component is optional. Symbol “*” defines an optional and repeatable component. The sequence of the components is written from left to right.

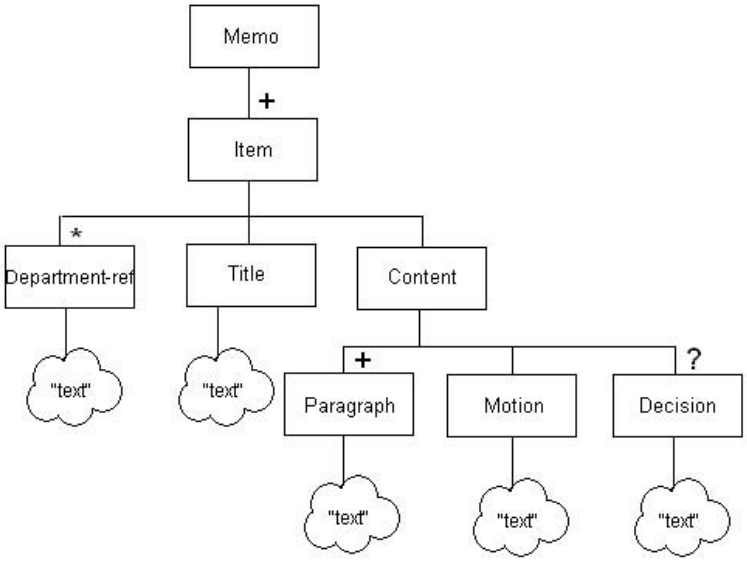

Figure 1. An Example of document hierarchy model for a memo.

When moving on the step 6, all the component hierarchies have been finished. During this step there can still be components left, which did not end up in the hierarchies. The remaining components can be divided in two layers:

- Information units
- Data-level components

Information units are groups of elements that should be modelled as a whole; such as list with list items and figure and its caption. Data-level components are small information bits that need to be processed differently for the surrounding data for some reason. Information units can be considered as "block" information and data-level components as "in-line" information. During the step 6, information units should be identified and their internal structures should be modelled by using the tree diagrams. The information of information units should be filled into the component forms.

Step 7 considers data-level element modelling. The list of remaining components is studied for refining the classification of the data-level components. At this phase the components should be mapped to elements and attributes. Finally the list of data-level components is built as well as the tree diagrams for the data-level elements.

Information units and data-level elements are revisited on step 8 for populating the branches of the tree diagrams with information units and data-level elements defined.

In step 9 the list of remaining components is again studied for identifying potential link components. *Link components* are components that record the relationship of two or more internal or external pieces of information. Link components as well as their potential source and target points are listed.

Finally in step 10 the whole design is revisited. One should check all the documents made and tie up loose end and add all missing information. This step

end the second phase of the Maler & El Andaloussi method and only one phase remains, which is producing the document analysis report.

The next phase is to conduct DTD or schema design. During this phase the models are replaced with actual DTD or schema textual models.

3.2. The Kennedy method

Kennedy [2003, 92-94] lists nine rules for defining XML schemas. The rules consider:

1. Data analysis
2. Coding the data
3. Use of container elements
4. Use of group or block elements
5. Use of subelements for multi-value data
6. Avoiding mixed content
7. Use of meaningful names
8. Correct use attributes and elements
9. Reviewing the desing

First rule, data analysis, concentrates following Maler & El Andaloussi methodology. According to Kennedy, one should

a) identify the components and
b) classify the components into logical groups.

However, the data analysis for XML is more than identifying the components, because it is also about defining the structures. During the data analysis the following questions (among the others) should be asked:

- Can this element repeat?
- Is this element optional?
- Is order important?

The second rule is about coding the data. This rule means that any information about the presentation should be left for the style sheet, because XML is about separating the data from the presentation. Can this be done? Is it possible to design XML document's structure without thinking the result?

Third rule is about using collection elements. By this Kennedy means that if an element can occur multiple times at the same level, one should create a container element for them, because it makes the XML more human readable and more easily to process with loop instructions (like xsl:for-each- loop in the XSLT transformation language). For example, if element <author> repeats multiple times, one should define container element <authors> to group the <author> elements.

Fourth rule is about remembering that XML mark-up is a tree structure. However the tree should be wide instead of tall. Fifth element reminds that the best way to represent multi-valued data is via subelements.

Sixth rule means that one should avoid mixed element content. The seventh rule reminds us to use meaningful names for elements. The eight rule emphasizes the correct use of elements and attributes. According to Kennedy, XML structures commonly contain lots of elements and fewer attributes. However,

schemas for data-oriented XML documents and data transfer may consist a dearth of attributes and a small number of elements. The application type for a schema usually dictates the approach to use.

The final rule is about recognizing that defining the schema is an iterative process. As Kennedy writes, an XML structure is a data structure. Like in a case of databases, the analysis is also in that case iterative, because often the structures of the tables are tested when the data is inserted in the tables and the possible design flaws occur.

Kennedy summaries the XML design as follows:

- Identify the components and their attributes.
- Identify the structure, identify component groups and look for collections.
- Model the structure by Elm tree diagram then use diagrams to write schema.

4. Discussion and conclusion

The logical structure of document-oriented XML document is hierarchical and may have a lots of optional and repeating elements whereas the logical structure for data-oriented XML documents is commonly more straight-forward. There are several schema languages for XML document schemas. The most common one is W3C's XML Schema - a rich schema language with multiple features and datatypes. However, XML Schema definition is considered as complex. Even though many applications support the use of XML Schema, only small amount of defined schemas in practice use its complex features. In fact many schemas are still equivalent to the expressiveness of DTDs, which are still often used even though XML Schema was designed to displace them.

RELAX NG is considered as XML Schema's competitor even though it is not as widely supported because its syntax is simpler and it is considered as easier to learn than XML Schema.

XML Schema with its many data types is ideal for the "data-oriented" use of XML. DTDs have always been a good practice in publishing area, where its ancestors, SGML DTDs have been used. That is why DTDs are more suitable in the use of "document-oriented" XML. RELAX NG falls in between, because its syntax is quite simple and according to Clark [2001], it is an evolution and generalization of XML DTDs. How ever with RELAX NG there is no need to flatten the natural hierarchical structure of the document into a list of element declarations as one would have to do with DTDs. In addition, RELAX NG uses XML syntax and with its grammar-element it is possible to use the datatypes of XML Schema if needed.

The Maler & El Andaloussi method is a profound method for DTD design and implementation. It was defined for SGML DTD design whereas the Kennedy method was targeted for RELAX NG schema design. There are many similarities in between the Kennedy and Maler & El Andaloussi method. The Kennedy method adopts the first two steps of the Maler & El Andaloussi method. Kennedy's method also applies Maler & El Andaloussi method on defining container elements, which are typical for document-oriented XML-document

types. Both Kennedy and Maler & El Andaloussi emphasize the importance of domain-oriented, meaningful element names. Both methods also remind DTD or schema designers that schema design is usually an iterative process and the document structures should be tested before implementation.

It may be concluded that both the Kennedy and the Maler & El Andaloussi method may be adoptable for schema design.

Why one should go all the trouble on analyzing documents and defining components when current XML editors can convert a schema in few seconds from a model XML document?

Kennedy [2003] reports that if data analysis with modelling is made before mark-up, the techniques can be learnt better and more quickly and even the produces XML structures were more efficient. This seems natural since in information system development a requirement analysis is usually done before an implementation of the system so same should apply to implementing document systems. In addition, Elm tree diagrams are easy to convert into schemas. Even though the DTDs are nowadays often replaced by schemas the Maler & El Andaloussi method may prove to be one of the best practices for schema design also.

An automatic conversion of the schema by XML editor does not support the identification and analysis of the schema element containers as schema components. Therefore automatically created schema definitions may fail to meet the document management needs of an organization. Schema provides a base for XML processing, content filtering, and content organization. A good schema also supports finding and using relevant information from the structured documents. Therefore schema design is a base for document-oriented XML document management and its importance should not be overlooked. .

5. References

[Bex 2004] Bex G.J., Neven F. & Van den Bussche J.*DTDs versus XML Schema: A practical study*. In WebDB 2004, pp 79–84.

[Bex 2005] Bex G.J, Martens M., Neven F. & Schwentick T. *Expressiveness of XSDs: From Practice to Theory, There and Back Again*. In *Proceedings of the Fourteenth International World Wide Web Conference*, Chiba, Japan, May 2005, pp-.712–721.

[Boukottaya 2004] Boukottaya A., Vanoirbeek C., Paganelli F. & Abou Khaled O., *Automating XML documents transformations: a conceptual modelling based approach*. 1st Asia-Pacific Conference on Conceptual Modelling, ACSW 2004, Dunedin, New Zealand, January 2004.

[Bray 2000]Bray T., Paoli J., Sperberg-McQueen C.M & Maler E. *Extensible Markup Language (XML) 1.0* (Third Edition) [online], W3C Recommendation. <http://www.w3.org/TR/ 2004/REC-xml-20040204/>

[Clark 2001a]Clark J. (2001) The design of RELAX NG. [Online]Available at http://www.thaiopensource.com/relaxng/design.html [16.1.2006]

[Clark 2001b] Clark J. & Murata M. *RELAX NG Specification*.

[DuCharme 2004]DuCharme B. *Documents vs. data, schemas vs. schemas. XML 2004 conference, Washington D.C.*

[Elmasri 2002] Elmasri R., Wu Y-C., Hojabri B., Li C. & Fu J. *Conceptual modeling for customized XML schemas*. In *21st International Conference on Conceptual Modeling (ER), Tampere, Finland*, volume 2503 of *Springer LNCS*, Springer, 2002, pp. 429–443.

[Fallside and Walmsley 2004] Fallside, D. C., & Walmsley, P. e. (2004, 2 May). *XML Schema Part 0: Primer. 2nd Edition. W3C Recommendation, 28 Oct. 2004.* Retrieved 16 June, 2005, from http://www.w3.org/TR/xmlschema-0/

[Goldfarb 1990] Goldfarb C.F. *The SGML handbook*. Oxford, UK. Oxford University Press, 1990.

[Goldfarb and Walmsley 2003] Goldfarb, C.F. and P. Walmsley, *XML in Office 2003. Information Sharing with Desktop XML*, Prentice Hall, Upper Saddle River: Pearson Education, 2004.

[Jelliffe 2001] Jelliffe R. *The current state of the art of schema languages for XML*. Presentation at XML Asia Pacific, Sidney, Australia, 2001.

[Kennedy 2003] Kennedy D. *Relax NG with XML data structures*. National Advisory Committee on Computing Qualifications, Palmerston Nth, July, 2003.

[Lee 2000] Lee D. & Chu W. *Comparative analysis of six XML schema languages*. *SIGMOD Record*, 29(3):76–87, 2000.

[Maler and El Andaloussi 1996] Maler E., El Andaloussi J. *Developing SGML DTDs. From text to markup*. Upper Saddle River NJ: Prentice Hall, 1996.

[Marinelli 2004] Marinelli P., Sacerdoti Coen C. & Vitali F. (2004). *SchemaPath, a minimal extension to XML schema for conditional constraints.* In Proceedings of the 13th International Conference on the World Wide Web (WWW13), New York City, U.S.A. ACM, 2004.

[Murphy 2001] Murphy, L.D. *Digital documents in organizational communities of practice: A first look.* In R.H. Sprague, Jr. (Ed.), *Proceedings of the 34th Hawaii International Conference on System Sciences.* Los Alamitos CA: IEEE Computer Society, 2001.

[Power 2003] Power R., Scott D. & Boyayad-Agha N. *Document structure*. Computational Linguistics, 29(3), 2003 pp. 211–260.

[Psaila 2000] Psaila G. ERX: *A conceptual model for XML documents*. In In Proc. of ACM Symposium on Applied Computing (SAC), Villa Olmo, Italy, 2000, pp. 898-903.

[Rossi 2000] Rossi M., Tolvanen J-P., Ramesh B., Lyytinen K. & Kaipala J. *Method rationale in method engineering*. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*. Maui, HI: IEEE Press, 2000.

[Salminen 2000] Salminen A., Lyytikäinen V. & Tiitinen P. *Putting documents into their work context in document analysis*. Information Processing and Management, 36(4), 2000, pp. 623-641.

[Salminen 2003] Salminen A. *Document analysis methods*. In Bernie C.L. (Ed.) Encyclopedia of Library and Information Science, Second Edition, Revised and Expanded. New York: Marcel Dekker, 2003, pp. 916-927.

[Thompson 2000a] Thompson H. S. *XML schema types and equivalence classes reconstructing DTD best practice*. XML Europe 2000 conference, 2000.

[Thompson 200b] Thompson H.S., Beech D., Murray M. & Mendelsohn N. *XML Schema part 1: structures*. W3C, Cambridge, MA, USA, 2000. Also available as http://www.w3.org/TR/ xmlschema-1

[van der Vlist 2002] van der Vlist E. *XML Schema languages.* In Proceedings of XML Europe 2002, Barcelona, Spain, May 2002.