Schema Languages for XML

Hugo Manguinhas[1],

[1] INESC-ID – Instituto de Engenharia de Sistemas e Computadores,
Apartado 13069, 1000-029 Lisboa, Portugal
hugo.manguinhas@ist.utl.pt

Abstract. This paper addresses the problem of the characterization of XML schema languages. The motivation is to offer enough information for one to choose the schema language that best fits the schema constraint requirements. XML is a language developed to encode data in a structured human readable way. Its simple syntax allows the representation of any sort of information from any particular domain of interest. In order to be portable across platforms and systems, XML documents often need to be associated to a well defined structure declaration: a schema. A schema is therefore a formal way to declare the syntax and to define the relationships and semantics of the attributes of an XML document, and for that purpose XML schemas languages can be defined. This paper presents an overview of the actual state of the art in schemas languages and a comparative analysis of them. As a result of this work valuable conclusions were reached to anyone willing to formalize schemas for XML documents.

Keywords: Structured Information, Document Engineering, XML, Schema, Schema Language, Grammar, Rules, Constraints, DTD, W3C XML Schema, RELAX NG, Schematron.

1 Introduction

This paper addresses the problem of the representation of schemas using XML schema languages. The paper presents an overview and a description of the actual state of the art in schema languages, including a comparative analysis of these languages.

The results of this work can be valuable to anyone willing to express a schema using a XML schema language. The conclusions of this paper might help choosing the schema language and strategies that best fits the schema constraint requirements. XML documents are structured human readable text representations of data. Since they are structured, XML documents are efficient solutions to store and transmit information. But for that purpose, these documents need a well defined structure in order to be portable across platforms and development systems.

One way to accomplish this wellness is by developing **schemas**. The purpose of a schema is to make it possible to describe a class of XML documents using constraints to define the usage and relationship of their underlying foundations, such as elements, attributes, attribute values and text. These constraints come from the rules governing

the domain of interest that the particular XML document is supposed to represent. From an object oriented perspective, schemas can be seen as classes (template for the objects) and XML documents as objects, instantiations of those classes. Essentially, schemas can be used to catalogue classes of XML documents. Like in common object paradigms, schemas can also be composed of other schemas to achieve higher levels of abstraction.

Schema **languages** have been developed to better represent the rules and constraints contained in schemas. For that, schema languages rely on **grammars** to define, in a logical form, all the constraints contained in a schema. As a consequence, those definitions can be used to **validate** documents that are instances of these schemas. Many attempts were made to build the ultimate schema language able to make it possible to express all the required constraints of any schema. This paper will focus a greater attention in four of these languages: Document Type Definition (here mentioned in short as DTD), W3C XML Schema (here mentioned in short as WXS), RELAX NG and Schematron. These four languages were chosen because they accurately represent the approaches that have guided the evolution of schema languages over the years.

The next section presents a short description of the four schema languages that were subject of analysis (Section 2). After that, the concept of grammar and its relevance for schemas is then presented (Section 3), followed by a description of their schema constraints according to their lexicon, syntax and semantics (Section 4). Finally, the paper compares the main characteristics of these languages (Section 5), presents other complementary work (Section 6), and finishes with conclusions and future work (Section 7).

2 Schema Languages

In this section, the four languages, which will be the focus of this paper, are introduced.

XML Document Type Definition [1] (DTD in short) is a subset of the original SGML DTD [2] (the schema mechanism for SGML). It was developed along with the XML specification to be the *de facto* standard for XML schema languages. It was the first step towards the development of a schema language as we know it. XML DTD is a W3C Recommendation since February 1998 and its current version is 2.10[1]. The DTD document declaration can appear internally or externally to the XML document. The DTD schema language defines the XML document structure through a list of element and attribute declarations, entities, references, comments and notations. The last four structure entities are important to the XML document encoding and processing but are irrelevant for the schema definition.

W3C XML Schema (here WXS in short) [3][4][5], sometimes informally called XML Schema Definition (XSD), which is the name given to a WXS instance document. It was the first wide-spread attempt to replace DTDs with a new schema language that would fit the increasing requirements. WXS is a W3C recommendation

[1] http://www.w3.org/2002/xmlspec/dtd/2.10/xmlspec.dtd.

since May 2001 and was the first separate schema language for XML to achieve Recommendation status by the W3C. WXS is similar to the DTD schema language by also defining the document structure through a list of declarations of elements and attributes. Nevertheless, it adds the ability to define types or frames in an object oriented approach that can be applied throughout the source document.

RELAX NG [6] (REgular LAnguage for XML Next Generation) consists of fusion of two earlier languages; RELAX[2] (REgular Language description for XML) designed by Murata Makoto and approved by ISO/IEC Technical Report 22250-1; and TREX[3] (Tree Regular Expressions for XML) designed by James Clark which is a subset of yet another language called XDuce[4]. RELAX NG was developed to be easier to learn and use. Two relevant aspects contributed significantly to these requirements and distinguish it from the other schema languages. The first consists on the language ability to follow the XML document tree-like structure, opposed to its leveraging into a list of structure declarations. The second consists on its ability to uniformly reapply the same primitives to all of the document structures.

The **Schematron Assertion Language** [7] was developed by Rick Jelliffe at the Academia Sinica Computing Centre (ASCC). Schematron is available as Final Committee Draft for the ISO International Standard Organization (ISO/IEC FDIS 19757-3) since October 2004, as part of a broader specification named Document Structure Definition Language (DSDL) [8]. Schematron differs from the previous languages in the way it expresses the rules and constraints embodied in the schema. It uses a tree-like rule based system as opposed to the grammatical oriented constraint declaration. Inconsistencies are thus identified through the matching of patterns which encode the document rules.

3 Schemas and Grammars

A grammar is used to define, in a logical form, all the constraints contained in a schema. As we have seen before, the input is a tree-like object model to be subject of validation. Any type of grammar capable of express constraints relative to a tree-like model is suited to this task. One type of grammars capable of satisfying these requirements is **tree grammars**, which is the mechanism for describing permissible trees. Over the years many researchers used tree grammars for schema representation (DTD, WXS, RELAX NG developers and others).

Tree grammars can be divided into four main subclasses: local, single-type, restrained-competition and regular tree grammars [9]. A **local** tree grammar provides a single content model for all content of each element (terminal). A **single-type** tree grammar does not have this tight restriction, but cannot allow two production rules competing in the same terminal to have different content models. A **restrained-competition** tree grammar lifts this restriction. A **regular** tree grammar accepts any kind of production. In terms of expression and ordered from the more expressive to the less expressive (that is some grammars cannot be rewritten as other grammars) are

[2] http://www.xml.gr.jp/relax/
[3] http://www.thaiopensource.com/trex/
[4] http://xduce.sourceforge.net/

regular, restrained-competition, single-type and local tree grammars. DTD, WXS and RELAX NG are examples of respectively local, single-type and restrained-competition tree grammars.

Beside tree grammars, other grammars can be used, like **rule-based grammars** (rule-based systems). These grammars define assertions about the presence or absence of patterns in the document object tree. Schema languages using rule-based grammars are called rule-based schema languages (also called pattern-based or assertion-based) [10]. Schematron is an example of a schema language based on a rule-based grammar.

Table 1. Analysis of the schema language constraint.

Constraints		Element				Text				Attr				AttrValue			
		DTD	WXS	RNG	Sch	DTD	WXS	RNG	Sch	DTD	WXS	RNG	Sch	DTD	WXS	RNG	Sch
Lexicon	Support	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
	Namespace Support	N	Y	Y	Y	-	-	-	-	N	Y	Y	Y	-	-	-	-
Syntax	Primitive Datatypes	-	-	-	-	N	Y	Y	Y	-	-	-	-	N	Y	Y	Y
	New Datatypes	-	-	-	-	N	Y	Y	N	-	-	-	-	N	Y	Y	N
	Mandatory	Y	Y	Y	Y	Y/N	N	Y	Y	Y	Y	Y	Y	N	Y	Y	Y
	Cardinality	Y/N	Y	Y	Y	N	N	Y	Y	-	-	-	-	-	-	-	-
	Order	Y	Y	Y	N	N	N	Y	N	-	-	-	-	-	-	-	-
	Alternatives	Y	Y	Y	Y	N	N	Y	Y	N	N	Y	Y	Y	Y	Y	Y
Semantics	Co-occurrence	N	N	N	Y	N	N	N	Y	N	N	N	Y	N	N	N	Y
	Relational	-	-	-	-	N	N	N	Y	Y	N	Y	Y	-	-	-	-

Legend: supported (Y); unsupported (N); incomplete support (Y/N); not applied (-).

Document Type Definition (DTD); W3C XML Schema (WXS); RELAX NG (RNG); Schematron (Sch).

4 Schema Constraints

In this paper the schema constraints expressed in grammars are classified in three different concepts or levels of abstraction. One is the **lexicon**, which deals with the domain vocabulary (identified XML entities); other is the **syntax**, that deals with structure *per se* of the document (tree grammars are specially focus at this part); and finally, the **semantics**, also called co-occurrence constraints, which deals with constraints relative to the content of the document. In this section, this paper evaluates and compares each schema language according to these three concepts. Table 1 resumes this evaluation showing the languages' support for each identified constraint type and the corresponding XML entity.

Note that each level requires that the previous constraints (belonging to the previous level) are matched; otherwise no further constraints can be evaluated. So if a document is said to be semantically correct, it means that it is also syntactically, and consequently lexically correct.

4.1 Lexicon

The schema language must include support for elements, attributes, attribute values and text, and also the support for namespaces.

In this subsection, this paper evaluates the capacity for the language to support the lexicon present in the tree object model. For validation purposes only a subset of the XML language is used. Things like parameter entities, processing instructions, DTD and CDataSections are processed and discarded when parsing the XML document. So the tree object model is reduced to four basic entities: **Elements**, **Attributes (Attr)**, **Attribute Values (AttrValue)** and **Text** (Comments are excluded since they do not represent relevant content information). The Figure 1 represents the basic entities that are involved in an XML document and their relations.

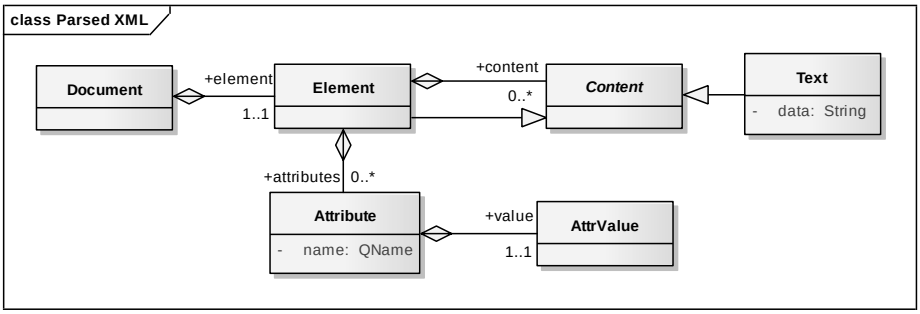

Fig. 1. Basic XML entities that are subject of validation.

All schema languages support the domain vocabulary contained in the model. WXS only recognizes text when it is the only content contained in an element (although it allows the declaration of mixed content it does not mandate where the text must appear when interleaved with elements). DTD recognizes text as PCDATA, and slightly lifts this restriction, enabling the definition of constraints with text content in between. RELAX NG and Schematron deal with text as any other kind of content. Concerning namespaces, only DTD does not have any support for this lexical entity. WXS supports namespaces naturally in the element and attribute declarations. RELAX NG allows at any time de selection of a namespace by using a specific statement (*nsName*, *name* or *anyName*). Schematron supports namespaces by relying in XPATH[5] expressions to construct expression patterns. XPATH expressions support namespaces naturally using the namespace prefix associated to the names of elements and attributes, and previously defined in the schema, using the natural XML definition of namespaces.

[5] http://www.w3.org/TR/xpath

4.2 Syntax

Syntax deals with the constraints related to the document structure. How the elements, attributes, attribute values and text are combined, placed and coded to form a meaningful environment of information.

In this kind of documents, syntax deals with the structuring of the tree; the alignment and leveling of elements, and also text, attribute placement and encoding. Tree grammars are specially focused on the definition of these constraints on tree structures. One can also use a rule-based grammar to represent structure related constraints that are missing in the tree grammar, particularly if we are using a less expressive tree grammar. Nevertheless, the use of rule-based grammars is not always straight forward and can be very hard to implement. DTD, WXS and RELAX NG are examples of respectively local, single-type and restrained-competition tree grammars. As expected RELAX NG, being a restrained-competition tree grammar, offers the more expressive approach to the representation of structure constraints. These constraints were classified in 5 different categories: **datatypes**, **mandatory**, **cardinality**, **order** and **alternatives** constraints.

Datatypes

DTD can only express the datatype of attributes in terms of explicit enumerations and a few coarse string formats. There has no support to describe numbers, dates, currency values, and so forth. Furthermore, DTD has no ability to express the datatype of character data. WXS fully supports datatypes by adding a rich set of primitive types and by enabling the declaration of new derived types from the existing primitive types. Unfortunately this is done only for attribute values but not to character data. Some authors [11] discuss the design principles that guided which types were to be defined and which were to be defined as primitives, accusing WXS for not having a well defined datatype hierarchy. RELAX NG offers a different approach to the problem. It firstly decouples the schema language from the set of datatypes, facilitating the evolution in an independent way of datatypes without compromising the language. Secondly it allows datatypes to be specified uniformly for both attribute values and element content; this is in accordance with the philosophy of uniform treatment for elements and attributes. RELAX NG introduces the concept of a Datatype Library, which provides a semantic model for a collection of data. Any collection of datatypes that can fit into the RELAX NG model can potentially be used as a RELAX NG datatype library, in particular, the datatypes defined by WXS Part 2 [4].

Mandatory and cardinality constraints

In DTD, the mandatory constraint is enforced by default simply by declaring the element in the parent element declaration. To declare cardinality of elements the operators '*' (zero or more), '?' (zero or one) and '+' (one or more) are used. For attributes the mandatory constraints is satisfied by the *REQUIRED* (no type for optional constraint) type defined in the attribute declaration. Text content can only be constrained to appear in the parent element content or not. In WXS the mandatory constraint is enforced by the declaration of elements and attributes. For attribute values this constraints is satisfied by defining a datatypes for the attribute. Cardinality

constraints for elements are enforced using the *minOccurs* and *maxOccurs* statements in particle statements (*Element*, *Choice*, *Sequence*, *Any*, *All* and *Group* statements). There are no mandatory and cardinality constraints for Text content. Like WXS, in RELAX NG, the mandatory constraint is enforced by the declaration of an element, text, attribute or value for an attribute. Cardinality constraints are partially satisfied by the *ZeroOrMore*, *OneOrMore* and *Optional* statements. As opposed to WXS, RELAX NG does not support the declaration of explicit and exact cardinality using integer boundaries. Schematron uses counting operations present in XPATH expressions to enforce mandatory and cardinality constraints. These counting operations can be applied to any type of content. Schematron relies on XPATH expressions to pattern matching and therefore is limited to the language ability to define types. XPATH only recognizes the numeric and string types but, on the other hand, can enforce these types at any type of content.

Order constraints

In DTD, order constraints for elements are supported by the ordering of elements in the children declaration. Nevertheless it stops being supported when mixed content is defined (Text and elements as content). In WXS the order of elements are defined by *Sequence* statement mandating that the order of declaration is the order in the document structure. No order for text content can be established. In RELAX NG the order of the statements and declarations mandate the order on the document. Text can be constrained anywhere in the document content just by placing the *Text* statement. Schematron can support these constraints by checking positions, in XPATH expressions, of elements or text content in the parent element. This may lead to incredible large expressions and are difficult to combine with alternative and cardinality constraints.

Alternative constraints

In DTD alternative constraints for elements are supported by the operator '|' (choice) and for attribute values with the enumeration of available datatypes (very few). For text and attributes there is no support for this kind of constraints. In WXS alternative constraints for elements are enforced by the *Choice* statement and can be combined with order constraints. Alternative for attribute values is defined with datatypes by using the *Union* (for type composition) and *Pattern* (for definition of character strings using regular expressions) statements. Like DTD, WXS does not support constraints for text and attributes. In RELAX NG, these constraints are satisfied by the *Choice* and *Interleave* statements. These statements can appear at any time in the schema enabling it to be applied to any type of content. This feature enables alternatives affecting more than one level in the object tree, making it possible to have alternative structures (skeletons). This is only possible because RELAX NG is a restrained-competition tree grammar. For AttrValues, alternatives can also be satisfied by the Datatype Library. Schematron again uses Boolean operations present in XPATH expressions to enforce alternatives to any given scenario, and therefore can be applied to any type of content. Nevertheless, this is not an immediate and simple way of defining alternatives.

4.3 Semantic constraints

Semantic constraints, also called **co-occurrence** constraints, are constraints between two or more values (content information). For example, "if an element has attribute A, it must also have attribute B" or "if an element has a parent X, then it must have an attribute Y". A co-constraint may exist between any kind of entity in the content model (elements, attributes, attribute values, and text). The occurrence of these constraints in the document model may lead to the identification of missing, illegal, duplicate, unordered, misplaced content. This kind of constraints is difficult or impossible to define using tree grammars (their primary focus is structure) and is commonly satisfied by rule-based grammars. In this comparison, Schematron is the most suited (sometimes the only) language to represent this kind of constraints. Schematron achieves this by being able to represent **co-occurrence constraints** using patterns coded in XPATH paths and expressions. Neither DTD, WXS and RELAX NG can. Some considerations about validation using rule-based constraints and particular the XPATH ability to express them are discussed in [12].

In this paper, a new subgroup of co-occurrence constraints is added, the **relational constraints**. These are a particular kind of co-occurrence constraints that deal with relational concepts like uniqueness, keys (identity) and key-references. These concepts allow systems to apply normalization according to the normal forms to schemas and benefit from them [13]. DTD (ID for identification and IDREF for key-references) and WXS (*Unique* for uniqueness, *Key* for keys and *KeyRef* for key-references) have explicit statements responsible for assuring this kind of constraints. Nevertheless, DTD only supports keys and key references for attributes and only one at a time. DTD has no support for the uniqueness constraint. On the other hand, WXS supports relational constraints for any kind of XML entities including combinations of them (e.g. a key composed of two separate attributes). RELAX NG has no support for relational constraints. Schematron can support these constraints making use of the search capacity of the language to assure uniqueness and existence of the keys.

5 Language Characteristics

In this section some of the main characteristics of the languages under analysis are discussed. Table 2 gives an overview of these characteristics, ordering each schema language from the most compliant with the feature to the less.

Expressivity

Expressivity is the measurement of the constraints enforced by the schema language. The more constraints a schema language allows the more expressive the language is. DTD is the less expressive language studied. It offers a limited set of statements allowing the enforcement of only a few constraints. Although the constraints available are able to cover most of schemas constraint requirements, it lacks the ability to define other, more refined constraints. Even though some constraints require a little more work for the developers to be defined, WXS offers a greater set of statements enabling the declaration of many constraints. One important

disadvantage of this language is the inability to deal with character data as typed content like RELAX NG. The source for the RELAX NG capacity to express constraints is the ability to apply the same statements to many different contexts. Being a restrained-competition tree grammar also contributed to wider the number of constraints being enforced. Schematron is the most expressive language of all, allowing the enforcement of almost all the constraints identified for a document.

Reusability

Reusability is the ability of a language to pack and reapply sets of statements in different contexts. DTD has no immediate way of reusing statements in the schema (only by entity replacement of attributes). Reusability in WXS is satisfied with the construction of types and attributes groups and their assignment respectively to elements and attributes. Schematron enables reusability by defining abstract patterns and rules and instantiating (with the ability to define parameter variables) them several times in the document. RELAX NG takes reusability a little bit further, allowing the definition of sets of any type of statements and reapply them to any context in the schema.

Compactness

Compactness is the ability to build the most constraints possible using the less available statements. Schematron produces very long and complex schemas due to the pattern-based approach. Schematron should be avoided when enforcing of lexical and syntactic constraints are required due to its non-structure approach. Schematron should only be used when the other languages are incapable of enforcing a constraint. WXS produces extensive schemas due to the leverage of the tree structure in a list of element declarations and the added complexity of type construction. Type construction in WXS requires an elaborate set of statements to define extensions and restrictions on content making it very extensive. With WXS the more complex the schema is the more extensive the schema becomes[6]. RELAX NG also produces very compact schemas. This is achieved by the ability of RELAX NG to follow the natural tree structure of the document combined with the huge versatility of the statements to be applied in wider contexts. Another important feature is the ability of RELAX NG to collect all types of statements and reapply them at any time and any context. Although the DTD language also leverages the document tree structure (as for WXS), it is able to produce very compact schemas given the universe of constraints possible of being enforced. The reduced expression of this language and therefore, the few statements available contributed to the compactness of the schema. The more constraints we need to enforce more difficult becomes the task of building the language primitive statements.

Complexity

Complexity is the level of knowledge required to the correct development of a schema. The reduced expression of the DTD language and the straight forward approach used in element and attribute declarations, make it very easy to understand.

[6] A more detailed comparative analysis of the language vocabulary present in WXS and RELAX NG is presented at: http://www.wyeast.net/compare.html

WXS is the most difficult language. Its approach to object oriented design makes it difficult to use and understand. Although very powerful, the type construction is not immediate for unskilled developers. The extension and restriction mechanisms are also difficult to understand and require previous knowledge and experience before using them, especially when dealing with attributes. An extended evaluation of the complexity of this language and best practices for its usage are discussed in [14]. RELAX NG is a very simple language. Again its ability to apply the same statements to many different contexts; reduced the schema vocabulary and therefore reducing the knowledge needed to develop a schema. Another aspect of RELAX NG that contributed to the reduced complexity of the schema is the ability of the language to follow the document tree structure, making it easy to develop a schema only by looking at the expected document. Some features and design characteristics of this language are discussed in detail in [15]. Again the rule-based approach of Schematron makes this language difficult to learn and even more difficult to understand the need expression for the enforcing of a given constraint

Extensibility

Extensibility is a design principle to take into consideration the language future growth. It is reflected in the ability of the schema language to add new features (increments), be enhanced or customized, without disturbing the existing features. Languages that use XML syntax for schema notation increase the ability to be extended. This is achieved by the ability of XML to be annotated with elements and attributes from other namespaces.

WXS provides a special element for this purpose (*AppInfo*), enabling the composition with other schema languages, especially Schematron [16]. Although RELAX NG does not provide specific elements or attributes to this purpose, it has an open syntax predicting this requirement. Schematron as a XML syntax language also allows schema extension. Nevertheless one of its main design principles was to be combined with other existing schema languages and not to import. Some interesting work in this area can be read in [17]. DTD has no mechanism for schema extension, mainly because it was the first effort in schema languages design. At the time DTD developers did not took this aspect in consideration because they did not predict the existence of other schema languages.

Table 2. Languages ordered by their relative characteristics.

	most	>>	>>	**less**
Expressivity	Sch	RNG	WXS	DTD
Reusability	RNG	Sch	WXS	DTD
Compactness	DTD	RNG	WXS	Sch
Complexity	Sch	WXS	RNG	DTD
Extensibility	WXS	RNG	Sch	-

Legend: Document Type Definition (DTD); W3C XML Schema (WXS); RELAX NG (RNG); Schematron (Sch).

6 Other Related Work

This paper combines the work of many different papers. It follows some of the guidelines present in [18] but introduces a different classification of the constraints being supported in each schema language. The approach used in this paper was oriented to the grammar used by the schema language to encode the schema constraints. This approach was inspired in the work of [9] based on a taxonomy of XML Schema Languages but introducing the rule-based grammar approach. Other works like [19] presented a rapid overview of the state of art of schema languages.

7 Conclusions and Future Work

As a result of this work some valuable conclusions were reached to anyone willing to formalize schemas for XML documents (some conclusions converge with the concepts guiding the DSDL project [8]).

- The schema language should be chosen carefully in order to define the most constraints possible using it. In this sense always:
  - Use a tree grammar for the definition of lexical and syntactic constraints. Consider always the most expressive tree grammar possible like restrained-competition grammars (RELAX NG is a good choice). When dealing with schemas requiring only the definition of simple vocabulary, consider the use DTD (or any other tree grammar based language).
  - Avoid using rule-based grammars as preferred schema language. Otherwise the schema becomes very extensive and complex.
  - Choose the schema language that adequately satisfies your datatype constraints. Otherwise, additional constraints will be required to restrain datatype values. Consider using a language that supports importing datatype libraries (e.g. WXS Datatype Library) like RELAX NG.
- Consider using additional schema languages when the chosen schema language does not support all the schema constraint requirements.
  - Use rule-based grammars when no tree grammar is able to satisfy the needed requirements. Otherwise the schema becomes very extensive and complex. Schematron is a good candidate since it can easily be embed in other schema languages and offers a great expressivity in defining schema constraints.
- Finally, in order to ensure that schemas remain simple and compact, always remind to choose the most appropriate schema language statement even if it means importing from other schema languages. WXS and RELAX NG offer a good choice for importing language statements for specific constraints.

For future work, it would be interesting to study the problems that users frequently face when developing schemas and provide possible solutions looking at some of the existing schema languages.

References

1. Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau, John Cowan. *Extensible Markup Language (XML) 1.1 (Second Edition)*. W3C, September 2006. See http://www.w3.org/TR/xml11/.
2. ISO 8879:1986 Information processing - Text and office systems - Standard Generalized Markup Language (SGML)
3. Fallside, David C. and Walmsley, Priscilla, editors. *XML Schema Part 0: Primer Second Edition*. W3C, October 2004. See http://www.w3.org/TR/xmlschema-0/.
4. Thompson, Henry S., Beech, David, Maloney, Murray, Mendelsohn, Noah, editors. *XML Schema Part 1: Structures Second Edition*. W3C, October 2004. See http://www.w3.org/TR/xmlschema-1/.
5. Biron, Paul V. and Malhotra, Ashok, editors. *XML Schema Part 2: Datatypes Second Edition*. W3C, October 2004. See http://www.w3.org/TR/xmlschema-2/.
6. Clark, James and Mokoto, Murata. *RELAX NG Specification*. OASIS, December 2001. See http://www.oasis-open.org/committees/relax-ng/spec.html.
7. *Document Schema Definition Languages (DSDL) — Part 3: Rule-based validation – Schematron*. ISO/IEC 2004. See http://www.schematron.com/iso/dsdl-3-fdis.pdf.
8. ISO/IEC 19757-1. *Document Schema Definition Languages (DSDL) – Part 1: Overview*. November 2004. See http://dsdl.org/0567.pdf.
9. Murata, Makoto, Lee, Dongwon, Mani, Murali. Taxonomy of XML Schema Languages using Formal Language Theory. ACM Transactions on Internet Technology (TOIT), Volume 5, Issue 4 (November 2005), pp. 660-704. See http://doi.acm.org/10.1145/1111627.1111631.
10. Dodds, Leigh. *Schemarama*. XML.com, 7 February 2001. See http://www.xml.com/pub/a/2001/02/07/schemarama.html.
11. Lewis, Amelia. *Not My Type: Sizing Up W3C XML Schema Primitives*, Julho 2002. See http://www.xml.com/pub/a/2002/07/31/wxstypes.html.
12. Provost, Will. *Beyond W3C XML Schema*, Abril, 2002. http://www.xml.com/pub/a/2002/04/10/beyondwxs.html.
13. Provost, Will. *Normalizing XML, Part 1*. Novembro 2002. http://www.xml.com/pub/a/2002/11/13/normalizing.html.
14. Obasanjo, Dare. *W3C XML Schema Design Patterns: Avoiding Complexity*. Novembro, 2002. See http://www.xml.com/pub/a/2002/11/20/schemas.html.
15. Clark, James. *The Design of RELAX NG*. See http://www.thaiopensource.com/relaxng/design.html.
16. Costello, Roger L. *Extending XML Schemas*. March 06, 2001. See http://www.xfront.com/ExtendingSchemas.html.
17. Robertsson, Eddie. *Combining Schematron with other XML Schema languages.* Junho, 2002. See http://www.topologi.com/public/Schtrn_XSD/Paper.html.
18. Lee, Dongwon, Chu, Wesley W. Comparative Analysis of Six XML Schema Languages, June 7, 2000. See http://www.cobase.cs.ucla.edu/tech-docs/dongwon/ucla-200008.pdf.
19. Rick Jelliffe. *The Current State of the Art of Schema Languages for XML*. 2001. See http://www.planetpublish.com/pdfs/RickJellif.