The Current State of the Art of Schema Languages for XML

Rick Jelliffe
Topologi Pty. Ltd., Sydney, Australia, `ricko@topologi.com`

Abstract

A characterization and comments on schema languages for XML at the end of 2001.

Keywords
SGML, XML, Schemas, DTDs, RELAX, Schematron, Hook, Examplotron, document validation, PSVI

I. Introduction

The years 1999 to 2001 saw much activity in the area of schema languages for XML[2]. I released Schematron, W3C released XML Schemas, and Makoto Murata and James Clark combined their RELAX and TREX validation languages into RELAX Next Generation. Several other interesting approaches became available.

This paper attempts to characterize the capabilities and approaches of these schema languages. It uses a layered model of the different constraints that apply to an XML document:

- the *encoding* rules;
- *linking* rules allow the physical composition of the document from parts to be described and, potentially, constrained;
- the well-formedness and namespace rules which produce an *infoset* (XML information set);
- constraints on *structures* which can be expressed by regular grammars;
- *static datatyping*, which are the constraints on values that are independent of any other values;
- *local reference integrity*, which are the constraints about references and uniqueness within a document, such as the XML ID/IDREF rules;
- *web reference integrity*, which are the constraints on references between resources over the WWW;
- *co-occurrence constraints*, which are constraints on structures or values based on data values (e.g., if the “pet” attribute has value “dog” then the “legs” attribute must have a value less than four) or based on the occurrence of other element structures that cannot be expressed by basic regular grammars (e.g., if there has already been a “subject” element in a previous element in the document, the current element cannot contain it);
- *value-defaulting* is where the schema language provides the values to be used when an attribute or element is not explicitly specified.

This layered model is not the only one possible, but reflects a common perception among users of documents.

One other very important consideration, in particular, is how the schema language supports the construction and maintenance of complex, large-scale and long-term schemas. In the DBMS world, these issues are dealt with at the tools level; but open text-based schema languages require tool-independent support: explicit mechanisms.

The following diagram shows the layers. In the subsequent sections, this diagram will be used to show:

- the kind of constraints the schema language is focussed on;
- the leading strength; and
- the leading weakness.

The strength and weakness should not be taken as indications of the usefulness of the Schema language for any purpose, since power is not the same thing as usability or suitability.

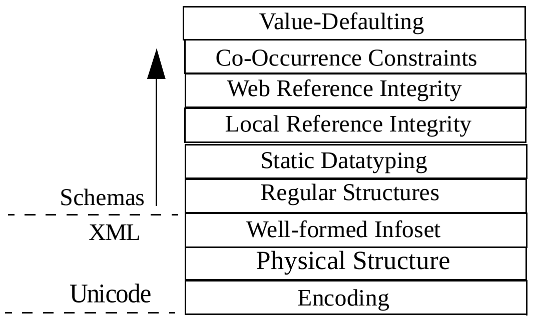

Figure 1: A layered model of schema domains

This paper gives the schema languages known to the author as *living* at the time of writing. The major omission is consideration of ISO ASN.1. Readers may also care to consider how systems such as the Unified Modeling Language’s Object Constraint Language might fit in.

II. DTDs

XML[2] provides a stripped-version of SGML’s markup declarations. XML DTDs provides a simple grammar mechanism, with a low-level inclusion mechanism (*parameter entities* and *marked sections*) to allow the structure rules to be parameterized and customized, per document. Only validating XML systems can make full use of this capability. SGML DTDs allow more powerful grammars to be specified: some set operations are allowed (*include* and *exclude*) and the positions in a content model in which text is valid can be specified.

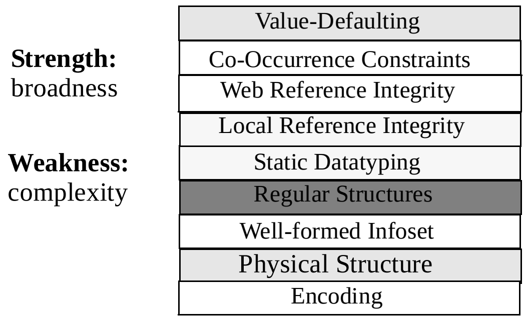

Figure 2: Focus of DTDs

The datatyping of DTDs is very simple and limited to enumerations and simple kinds of tokens. The local reference checking (ID/IDREF) is basic but useful.

III. W3C XML Schema

W3C XML Schema[1][16] use XML as its expression language. It takes the namespace, not the document, as the fundamental unit of interest in validation. It is made of two normative specifications, one for Datatypes and the other for Structures and everything else. All typing uses type hierarchies, by which one can *restrict* and, in some cases, *extend* other types.

The Datatypes specification has found a good level of acceptance as a thorough and elegant approach to specifying datatypes. For each of its primitive types, it defines a set of *facets*, such the *maximum value* of a number; types are derived from these primitive types by restricting facets.

The intent of W3C XML Schema is to reconstruct the facilities provided by DTD's parameter entities and marked sections into a full *type-lattice* system with type inheritance, type extension and type restriction. However, many of the uses of parameter entities and marked sections could not be reconciled with element or attribute "types" and so a basic module system of *import* and *include* declarations is also provided.

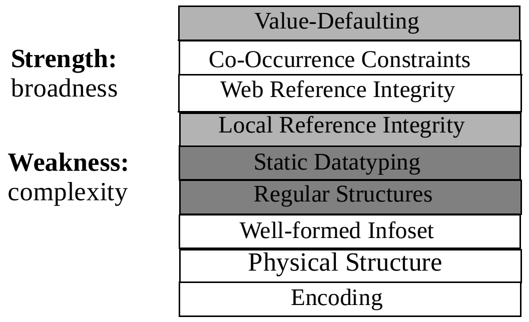

Figure 3: Focus of XML Schemas

The Datatypes specification is uncontroversial enough other schema languages can be expected to merely adopt it.

The Structures specification is rather more controversial, and perhaps we should expect further development.

IV. ISO/JIS/OASIS RELAX

The RELAX NG[4] is a grammar-based schema language developed specifically for data validation. It relaxes some of the ambiguity constraints of DTDs and XML Schemas, and to be powerful without providing any inheritance mechanism.

RELAX NG is being developed by a small working group at the OASIS organization, building on earlier schema languages by Makoto Murata and James Clark. The intellectual program of the former is concerned with discovering how to allow documents of different types to be combined: if a fragment of one document is pasted into another, their schemas must also allow combination; this issue not only concerns documents of different namespaces, but combining schemas which are variants of each other, where one document conforms to an earlier version of a schema and another conforms to a later version. The intellectual program of the latter is more concerned with developing implementation primitives with which schema languages as powerful as SGML DTDs can be implemented simply.

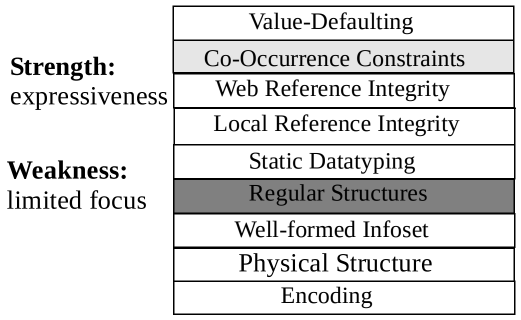

Figure 4: Focus of RELAX NG

The RELAX NG specification is nearing completion at time of writing. It has been mooted to become an ISO standard itself (i.e., to be used as the basis of DSDL, see below) and at least it may inform the development of future versions of W3C XML Schemas.

V. Schematron

The Schematron Assertion Language[7][9] is a language and toolkit for making assertions about patterns found in XML documents. It can be used as a friendly validation language and for automatically generating external annotation (links, RDF, perhaps Topic Maps). Because it uses XPath[5] paths and expressions rather than grammars, it can be used to assert many constraints that cannot be expressed by DTDs or XML Schemas.

I developed Schematron at the Academia Sinica Computing Centre, Taiwan, during 2001 as part of research on XML schema languages[15] and on internationalization.[6]

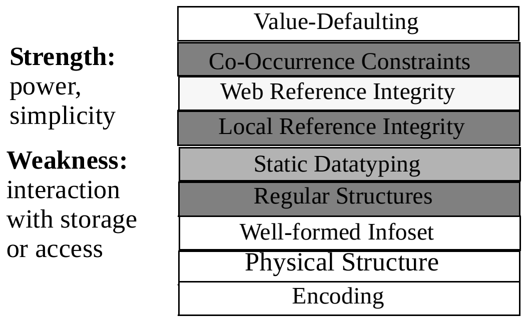
Value-Defaulting
Co-Occurrence Constraints
Web Reference Integrity
Local Reference Integrity
Static Datatyping
Regular Structures
Well-formed Infoset
Physical Structure
Encoding
Strength: power, simplicity
Weakness: interaction with storage or access

Figure 5: Focus of Schematron

To characterize assertions informally, following are some of the kinds of assertions that can be made. These can be made, practically, by the grammar-based schema languages in this paper:

A year contains twelve months, a month contains 28 to 31 days; a year has 355 or 354 days; every day has a name, and all the names must cycle Monday, Tuesday, Wednesday, etc. correctly across month boundaries; if December 25 is a Sunday then December 26 must have attribute holiday *as true.*

This kind of collection of assertions is a *pattern*, in Schematron terminology: various rules collected together, describing some extended structure in the document. The language is being exploring how to represent abstract non-regular patterns in documents[11].

Schematron, like RELAX, was developed in part as a response to the size and complexity of XML Schemas. The Schematron reference implementation only takes about two pages of XSLT code.

Because Schematron and rule-based schema languages represent a new paradigm with many unexplored possibilities, I am delaying presenting Schematron for standardization, although it has been mooted for ISO standardization. I believe it needs some abstraction mechanism for patterns in order to make them first-class objects for expressing schemas; some parameterization mechanism should be enough.

Another lack has been a standard data-typing library; I am working on such a library, implementing equivalents of the XML Schemas Datatypes built-in types.

VI. ISO Document Structure Definition Language

DSDL[3] is a proposed schema language for XML and SGML, being championed at ISO by the British national standards body. Its main goal seems to be to provide and upgrade all the current capabilities of full SGML DTDs.

It would add some data-typing and occurrence features from the other schema languages, but have much more emphasis on modularity, physical storage issues, tag minimization, change tracking, and embedded non-XML sections as part of the same data stream.

From the emphasis on parsing and physical construction issues, it seems that DSDL will be most useful in the publishing areas in which SGML has proved useful, but perhaps which have been neglected during XML's rise, rather than the XML Schema's areas of database-connectivity and XML's area of small messages.

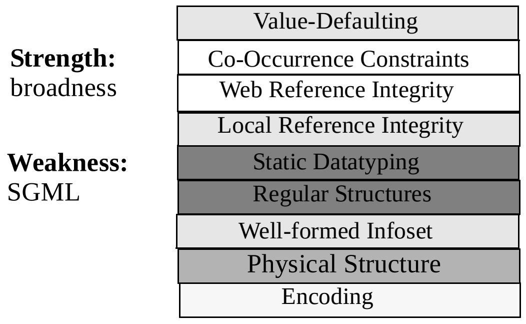

Figure 6: Focus of DSDL

VII. Xlinkit

Xlinkit is a first-order logic-based rule language for specifying link consistency, part of a *rule-based hypertext generation and consistency-checking service.*[12] Xlinkit was developed at University College, London by Professor Anthony Finkelstein, Christian Nentwich and others.

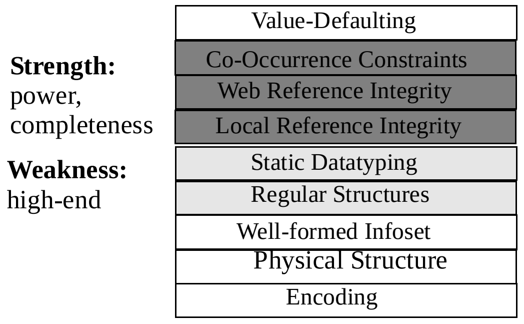

Figure 7: Focus of xlinkit

To understand xlinkit, let us consider a general use-case: we very often transform XML documents from one structure to another; sometimes the XML document goes through multiple transformations. There is a school of software engineering that says we need to be able to prove our systems; this may take place by unit tests on parts of the system or be put in place permanently for quality control. Xlinkit provides tools for these kinds of checks: it allows links to be implied between different documents. The links might also be between documents in the same generation, rather than just pre- and post-transformation documents: some information in one document may presume the existence of some other information in another document.

Xlinkit shares many features in common with the author's Schematron Assertion Language: the use of XPath expressions, a rule-based system, the use schema languages for automatic link generation, and the positioning of schema languages in the Software Engineering discipline (rather than, say, schema languages for application generation).

However, the two were developed independently and as part of different intellectual programs: xlinkit as a way to check link consistency, Schematron as a way to describe abstract patterns in documents. Xlinkit is perhaps best targeted at larger sites which can use a consistency-checking service, while Schematron is intended as more a tool for giving end-users diagnostic information. Xlinkit pre-dates Schematron by a small period and has some patent governing it. However, implementations of both XLinkIt and Schematron source code are available as Open Source.

VIII. Examplotron

Examplotron[17] uses exemplars to specify a schema: a document with all the elements and attributes that will be required, with some additional elements to specify optionality. Examplotron was developed by Eric van der Vlist.

Schematron-like assertions are also available.

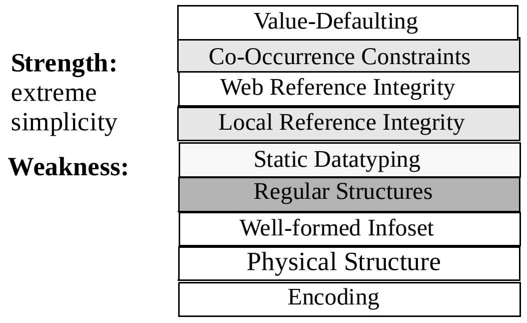

Figure 8: Focus of Examplotron

IX. Hook

Hook[8] is an experiment in minimalism. It is based on *partial ordering* rather than grammars or paths. Hook was developed by the author at Academia Sinica and Geotempo, Inc.

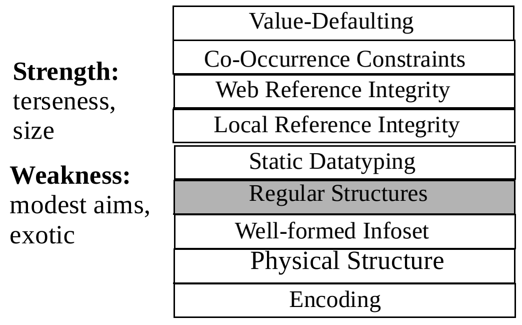

Figure 9: Focus of Hook

The intent is a schema language which is more terse than DTDs, but which allows several important constraints to be validated by a receiving web client using a SAX filter. The constraints validated are:

- element names,
- possible top-level elements,
- elements that cannot contain other elements,
- some kinds of containment or sequence relations.

A Hook schema is an ordered list of element names, with various brackets and punctuation to indicate grouping and containment properties. Below are examples of three Hook schemas for a simple HTML.

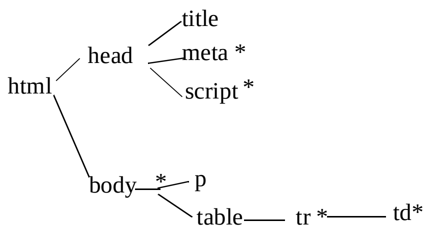

Figure 10: A grammar tree for a simple HTML

The first example below merely lists all elements (here in alphabetical order). This is adequate for check element name spelling, but not much else. The square brackets indicate that the elements in the group may contain each other.

```
[body head html meta p script table td title tr]
```

Figure 11: A basic Hook schema for simple HTML

The second example below groups the elements by level: the top-level element must be “html”. The second level elements must be “head” or body”, and so on. The full-stop on title indicates that it cannot have children.

```
html [ head body ] [ title. meta script; p table] [tr] [td]
```

Figure 12: A possible Hook schema for the simple HTML by level

The third example below introduces provides quite a lot of ordering and containment information. The semi-colon on the meta element indicates that the element cannot contain elements from the same group.

```
html head title. [meta; script;] body [p; table][tr][td;]
```

Figure 13: A better Hook schema for the simple HTML

The reason for the name Hook is that it is the *Hook path* for each element in a document that is checked for its order. The Hook path is the path created by concatenating the *previous-sibling* elements of the current element with its *ancestor* elements, using XPath[5] axis terminology.

The rationale for using Hook paths to unify axes is that in SGML document, which allow omit-tag minimization, the class of errors where one element is wrongly embedded in another rather than following it, and *vice versa*, is important; however, it is less important in XML due to the well-formedness rules. It may not be an important error to test for.

X. Embedded Schematron

One distinguishing feature of Schematron is that it has been designed to be embeddable in other schema languages.

Various levels of assertions are possible:

- simple assertions (e.g. single `sch:assert` elements) which must be true at some context provided by the host schema;

- rules, which are collections of assertions, which provide their own contexts;
- full patterns, which collect rules together.

An open-source preprocessor exists[14] by which Schematron rules may be extracted from a W3C XML Schema appinfo element and assembled into an full Schematron schema. Another open-source preprocessor exists[13] which can decorate a document with the type names from an XML Schema schema; the conjunction of these two pre-processors allows a Schematron schema to act, in large part, using the PSVI without requiring any special API.

Sun Microsystem's Multi-Schema Validator has an add-on component[10] to interpret Schematron rules attached to RELAX elements.

Examplotron includes simple assertions.

The provision of simple assertions is so simple to implement, given the widespread availability of XSLT implementations, that it may well become a common feature of future schema languages.

XI. Controversies

Here are some of the current technical issues.

A. Post-Schema Validation Infoset

W3C XML Schema introduces a concept of the *Post-Schema-Validation Infoset* (PSVI). This is the information set of an XML document after it has been validated by an XML Schema processor. The information set is decorated with various type information and validation outcomes, which are enumerated in the XML Schemas specifications.

This in turn leads us to a brave new world of PSVI-aware tools and standards. For example, an updated version of XPath which allows us to locate nodes according to their type. A proponent of PSVI processing might claim that it will allow more efficient and generic software.

An opponent of PSVI processing might claim that not much information is suited for type-based processing, that controlled vocabularies of element names and content models are entirely adequate, that PSVI processing adds a level of complexity and abstraction that is not needed, that much efficiency can be gained by casting the type of items in queries rather than relying on a PSVI.

One of the strongest objections is that, for web use, it is very desirable to have schema languages that can be downloaded at the same time as the document: this requires a fairly terse schema language so that the download time of the schema does not dominate the download time of the document. Terseness is a feature of DTDs.

The XML Schema line to answer this is that an XML Schema can be compiled into efficient code, rather than being downloaded for each document being validated. This is certainly an issue where the developers of a data-interchange system will have a different opinion from someone trying to open a document over the WWW and edit it.

The PSVI is a very disruptive force. It entails having data accessible in a form for which there is no straightforward XML equivalent. It will require a new generation of all XML standards or specifications. It is significantly increases the complexity of XML systems, though certainly for some users this may be functionality they require.

B. Relation to Query Languages

One of the considerations of XML Schemas was to allow efficient querying. In particular, a content model can only be extended by *suffixing* particles to its end. This means that if the data is being accessed using some index into a structure, queries that are defined on a base element structure will also work against elements of a type derived from the base.

A proponent of this might say that this can allow XML data access to be closer to that achieved by table-based implementations.

An opponent might say that querying is better left to relational database systems, and XML Schemas seems to hijack XML away from being a simple format for transmitting topic-oriented data, documents and reports over the WWW to being a next-generation interface for semi-structured database management systems. In that case, the requirement to allow efficient querying over large documents is entirely spurious, and as a result XML Schema's limited suffixing provides little utility for XML documents over the WWW.

C. Scoping: Documents or something else?

SGML and XML have a strong concept of *document*, a logical collection of information. The document can be composed of different physical entities. Unique identifiers (IDs) are scoped by documents.

However, document scoping introduces several problems. If elements from one document in some namespace are cut and pasted into another document which uses some other namespace, we will not get element name clashes, but we may have ID clashes. Consequently XML Schemas provides a way to scope IDs to elements in different namespace. The key and uniqueness mechanisms in XML Schemas adopt a limited version of XPath to do this; there is not enough evidence yet to know that the mechanism provided by XML Schemas is adequate.

Neither DTDs nor XML Schemas provide any way to define what the top-level element is. An XML Schemas schema instead provides the constraints applicable to a single namespace. A document is validated by finding the appropriate schema for each namespace as it is encountered in the document instance.

XII. References

Many of these papers will also have archive copies available through the Robin Cover's *Cover Pages* archive, `http://xml.coverpages.org/`

There is Open Source code available for all the Schema languages considered in this paper, except Hook which is experimental.

[1] Paul V. Biron, Ashok Malhotra, editors. *XML Schema Part 2: Datatypes*. W3C (World Wide Web Consortium), 2001. See `http://www.w3.org/TR/xmlschema-2/`

[2] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, editor, *Extensible Markup Language (XML) 1.0 (Second Edition)*, W3C (World Wide Web Consortium), 2001. See `http://www.w3.org/TR/REC-xml`

[3] Martin Bryant, *Use Cases for DSDL*, web page, The SGML Centre, U.K., 2001. See `http://xml.coverpages.org/BryanDSDL-UseCases20011006.html`

[4] James Clark, MURATA Makoto, editors. *RELAX NG Specification*, technical report, OASIS, 2001. See `http://www.oasis-open.org/committees/relax-ng/spec.html`

[5] James Clark, Steve DeRose, editors, *XML Path Language (XPath) Version 1.0*, W3C (World Wide Web Consortium), 2001. See `http://www.w3.org/TR/xpath`

[6] Rick Jelliffe, *Chinese XML Now!*, website, Academia Sinica Computing Centre, Taiwan, 1999. See `http://www.ascc.net/xml`

[7] Rick Jelliffe, *The Schematron Assertion Language 1.5*, technical report, Academia Sinica Computing Centre, Taiwan, 2000. See `http://www.ascc.net/xml/resource/schematron/Schematron2000.html`

[8] Rick Jelliffe, *Resource Directory (RDDL) for Hook 0.2, A One-Element Language for Validation of XML Documents based on Partial Order*, web page, Academia Sinica Computing Centre, Taiwan, 2001. See `http://www.ascc.net/xml/hook/`

[9] Rick Jelliffe, *Resource Directory (RDDL) for Schematron 1.5*, web page, Academia Sinica Computing Centre, Taiwan, 2001. See `http://www.ascc.net/xml/schematron/`

[10] K. Kawaguchi, *Sun Multi-Schema XML Validator Schematron add-on*, software, Sun Microsystems, 2001. See `http://www.sun.com/software/xml/developers/schematronaddon/`

[11] Toivo Lainevool, *XML Patterns*, website, `http://www.xmlpatterns.com/`

[12] Christian Nentwich, Wolfgang Emmerich and Anthony Finkelstein. *xlinkit: links that make sense*, technical report, Department of Computer Science, University College London, 2001. See `http://www.cs.ucl.ac.uk/staff/A.Finkelstein/papers/htxlinkit.pdf`

[13] Francis Norton, *Type-Tagger*, software, 2001. See `http://www.schemavalid.com/utils/typeTagger.zip`

[14] Francis Norton, Eddie Robertsson. *xsd2sch.xsl*, software, 2001. Available as part distribution of *Topologi Schematron Validator*, Topologi, 2001. See `http://www.topologi.com/`

[15] Simon St. Laurent, *Schematron: an Interview with Rick Jelliffe*, webpage, XML.com, 2000. See `http://www.xmlhack.com/read.php?item=121`

[16] Henry S. Thompson, David Beech, Murray Maloney, Noah Mendelsohn, editors. *XML Schema Part 1: Structures*. W3C (World Wide Web Consortium), 2001. See `http://www.w3.org/TR/xmlschema-1/`

[17] Eric van der Vlist, *Examplotron*, web page, 2001. See `http://www.examplotron.org/`