A Comparative Study of Schema Languages for XML Documents

Jean-Michel HUFFLEN
LIFC — University of Franche-Comté
BachoTEX, 1st May 2011

Contents

A small example

Just parsing such a file.

A small example

Just parsing such a file.

$\mathrm{DTD} \Longleftarrow \mathrm{SGML}$.

A small example

Just parsing such a file.

$\mathrm{DTD} \Longleftarrow \mathrm{SGML}$.

Simplified form. OK, but. . .

Drawbacks

Two syntaxes: XML and EBNF.

Drawbacks

Two syntaxes: XML and EBNF.

No modularity.

Drawbacks

Two syntaxes: XML and EBNF.

No modularity.

Elements: either a structure, or a string, or both.

Drawbacks

Two syntaxes: XML and EBNF.

No modularity.

Elements: either a structure, or a string, or both.

No support for namespaces.

Drawbacks

Two syntaxes: XML and EBNF.

No modularity.

Elements: either a structure, or a string, or both.

No support for namespaces.

But also. . .

Elements: only arbitrary repetitions are allowed.

Keys $\Longleftarrow$ `ID`/`IDREF`(`S`) attributes.

Schemas' common points

XML texts themselves.

Namespace-aware.

Schemas' common points

XML texts themselves.

Namespace-aware.

DOCTYPE tags? Dummy if you use *entities*.

Schemas' common points

XML texts themselves.

Namespace-aware.

DOCTYPE tags? Dummy if you use *entities*.

XInclude

XML Schema

Verbose, but complete.

XML Schema

Verbose, but complete.

Two extreme styles: *Russian-doll* and *flattened*.

XML Schema

Verbose, but complete.

Two extreme styles: *Russian-doll* and *flattened*.

`minOccurs`/`maxOccurs` attributes.

XML Schema

Verbose, but complete.

Two extreme styles: *Russian-doll* and *flattened*.

`minOccurs`/`maxOccurs` attributes.

Unordered collections (`xsd:all`).

XML Schema

Verbose, but complete.

Two extreme styles: *Russian-doll* and *flattened*.

`minOccurs`/`maxOccurs` attributes.

Unordered collections (`xsd:all`).

Nillable elements.

XML Schema's types

Rich predefined type library.

XML Schema's types

Rich predefined type library.

Simple type $\Longrightarrow$ simple content.

XML Schema's types

Rich predefined type library.

Simple type $\Longrightarrow$ simple content.

Complex type $\Longrightarrow$ simple or complex content.

XML Schema's types

Rich predefined type library.

Simple type $\Longrightarrow$ simple content.

Complex type $\Longrightarrow$ simple or complex content.

Building new types, by extension or restriction.

Uniqueness

`xsd:unique` $\Longleftarrow$ fields may be optional.

Uniqueness

`xsd:unique` $\Longleftarrow$ fields may be optional.

`xsd:key` $\Longleftarrow$ all the fields are required.

Uniqueness

`xsd:unique` $\Longleftarrow$ fields may be optional.

`xsd:key` $\Longleftarrow$ all the fields are required.

`xsd:keyref` $\Longleftarrow$ values related to an element `xsd:unique` or `xsd:key`.

Relax NG

Designed to be less verbose than XML Schema. Regular expressions are specified using XML-like syntax.

Relax NG

Designed to be less verbose than XML Schema. Regular expressions are specified using XML-like syntax.

(Compact—C-like—syntax exists.)

Relax NG

Designed to be less verbose than XML Schema. Regular expressions are specified using XML-like syntax.

(Compact—C-like—syntax exists.)

Interleaving operator.

Relax NG

Designed to be less verbose than XML Schema. Regular expressions are specified using XML-like syntax.

(Compact—C-like—syntax exists.)

Interleaving operator.

One way for definitions of elements, attributes, groups, contents.

Relax NG (con'd)

Simple types $\Longleftarrow$ XML Schema's.

Relax NG (con'd)

Simple types $\Longleftarrow$ XML Schema's.

Often users redefine the default namespace $\Longrightarrow$ no prefix.

Relax NG (con'd)

Simple types $\Longleftarrow$ XML Schema's.

Often users redefine the default namespace $\Longrightarrow$ no prefix.

(For me, that is bad technique.)

Relax NG (con'd)

Simple types $\Longleftarrow$ XML Schema's.

Often users redefine the default namespace $\Longrightarrow$ no prefix.

(For me, that is bad technique.)

Less verbose, actually?

Relax NG (con'd)

Simple types $\Longleftarrow$ XML Schema's.

Often users redefine the default namespace $\Longrightarrow$ no prefix.

(For me, that is bad technique.)

Less verbose, actually? In fact, that depends on the number of your *definitions*.

Schematron

Completely dynamic: you attack the XML document by means of XPath expressions.

Schematron

Completely dynamic: you attack the XML document by means of XPath expressions.

For example, expressing that a work is dated between the birth and death of its author.

Schematron

Completely dynamic: you attack the XML document by means of XPath expressions.

For example, expressing that a work is dated between the birth and death of its author.

Testing a structure with optional subparts $\Longleftarrow$ punishment.

Schematron (con'd)

Often used in conjunction with another schema language $\Longleftarrow$ embedded Schematron subschemas.

Schematron (con'd)

Often used in conjunction with another schema language $\Longleftarrow$ embedded Schematron subschemas.

Usually compiled into XSLT stylesheets.

DTD's fall

Forthcoming. . . Betcha.

DTD's fall

Forthcoming. . . Betcha.

Other schema languages?

DTD's fall

Forthcoming. . . Betcha.

Other schema languages? Why not?

DTD's fall

Forthcoming. . . Betcha.

Other schema languages? Why not? But lack of tools.

Ending

What do I recommend?

Ending

What do I recommend?

If you do not need uniqueness property, you can use Relax NG.

Ending

What do I recommend?

If you do not need uniqueness property, you can use Relax NG.

For big-sized examples, the most suitable is probably XML Schema $\Longleftarrow$ importing schemas, possibly with redefinitions.