



Extreme DocBook

Norman Walsh

XML Standards Architect

Extreme Markup Languages 2004

01-06 August 2004

<http://www.sun.com/>



Version 1.0

Table of Contents

This presentation explores some of the design choices made in recasting DocBook from an XML DTD to a RELAX NG Grammar.

- What is DocBook?

- History and Purpose

- State of the Art

- DTD vs. RELAX NG

- Compatibility

- Conclusions

What is DocBook?

What is DocBook?

A DocBook Document

What is DocBook?

- DocBook is an XML vocabulary for writing documentation. It is particularly well-suited to books and papers about computer hardware and software, though it is by no means limited to them.
- It has been subset down to something that resembles HTML.
- It has been extended to do things as different as websites and, well, presentations like this one [colorized.html].

A DocBook Document

```
<book>
<bookinfo>
<title>A Book Title</title>
<author>
  <firstname>John</firstname>
  <surname>Doe</surname>
</author>
</bookinfo>
<chapter>
<title>The First Chapter</title>
<para>Some <emphasis>text</emphasis>.</para>
</chapter>
</book>
```

History and Purpose

DocBook History

DocBook's Purpose

Who's Responsible for DocBook?

DocBook NG is My Fault

DocBook Development

DocBook History

- DocBook has been actively maintained for more than a decade.
- It has always been maintained by a committee of some sort. It is now being developed by an OASIS Technical Committee.
- DocBook was an SGML DTD for many years, it is now principally an XML DTD.

DocBook's Purpose

- DocBook documents are mostly hand authored. Unlike SOAP envelopes, purchase orders, and XML/RPC invocations, humans write DocBook.
- It's mostly read by humans. DocBook documents, aren't usually consumed by unmarshalling processes building object graphs.
- DocBook contains a lot of mixed content. Very few elements have "simple content," dates, numbers, etc.

Who's Responsible for DocBook?

- Current committee members: Paul Grosso, Adam Di Carlo, Mark Johnson, Dick Hamilton, Larry Rowland, Nancy Harrison, Gary Cornelius, Jirka Kosek, Michael Smith, Robert Stayton, Steven Cogorno, Scott Hudson, Norman Walsh
- Selected “alumni”: Terry Allen, Jon Bosak, Dale Dougherty, Ralph Ferris, Dave Hollander, Eve Maler, Murray Maloney, Conleth O'Connell, Mike Rogers, Jean Tappan

DocBook NG is My Fault

- The bugs are mine.
- The current release is “Eaux-de-vie” from a few days ago.
- The Technical Committee plans to move to RELAX NG for DocBook V5.0.

DocBook Development

- There have been about 15 releases in roughly ten years.
- Four of those releases have been “major” releases.
- That means we’ve added new stuff about $\frac{1}{4}$ the time!

State of the Art

DocBook Growth

A DocBook DTD Fragment

Growing Pains

DocBook DTD Shortcomings

Design Goals

DocBook Growth

“DocBook is like a pearl, it grows by accretion.”

A DocBook DTD Fragment

```
<!ENTITY % chapter.module "INCLUDE">
<![%chapter.module;[
<!ENTITY % local.chapter.attrib "">
<!ENTITY % chapter.role.attrib "%role.attrib;">

<!ENTITY % chapter.element "INCLUDE">
<![%chapter.element;[
<!ELEMENT chapter %ho; (beginpage?,
                        chapterinfo?,
                        (%bookcomponent.title.content
                        (%nav.class;)*,
                        tocchap?,
                        (%bookcomponent.content;),
                        (%nav.class;)* )
```

A DocBook DTD Fragment (Continued)

```
    %ubiq.inclusion;>
<!--end of chapter.element-->]]>

<!ENTITY % chapter.attlist "INCLUDE">
<![%chapter.attlist;[
<!ATTLIST chapter
    %label.attrib;
    %status.attrib;
    %common.attrib;
    %chapter.role.attrib;
    %local.chapter.attrib;
>
<!--end of chapter.attlist-->]]>
<!--end of chapter.module-->]]>
```

Growing Pains

- Growth by accretion has resulted in some content models that are at best odd and at worst broken in pretty obvious ways.
- Ten years of incremental growth has also changed the scale of DocBook. Designing a schema of roughly 400 elements is different than designing a schema of roughly 100. Logically extending decisions that looked regular and consistent when DocBook had 100 elements has not always resulted in a design that continues to look regular and consistent.

DocBook DTD Shortcomings

- The DTD fails to capture some significant constraints.
- Originally designed as an exchange DTD, it has largely become an *authoring* DTD. Exchange and authoring aren't opposing design centers, but they are different.
- While DocBook is a shining example of parameter entity customization, parameter entity customization is fiendishly hard.

Design Goals

The result of recasting DocBook should...

1. “feel like” DocBook.
2. enforce as many constraints as possible.
3. clean up the content models.
4. give users the flexibility to extend or subset the schema in an easy and straightforward way.
5. be able to generate XML DTD and W3C XML Schema versions of DocBook.

DTD vs. RELAX NG

Uniform Info Elements

Uniform Info Elements

Info Elements in More Contexts

Info Elements in More Contexts

Required Titles (Valid)

Required Titles (Invalid)

Required Titles

Co-Constraints (DTD)

Co-Constraints

Untangling Tables

Untangling Tables

Untangling Tables

...

Uniform Info Elements

- DocBook V4.x has **setinfo**, **bookinfo**, **chapterinfo**, **appendixinfo**, **sectioninfo**, etc.
- Many people think it would be nicer if there was just one **info** element.
- In DTDs, this can't be done without sacrificing the ability to customize the **info** elements on a contextual basis.
- In RELAX NG, we can have different patterns that each define an element named **info**.

Uniform Info Elements

```
book.info = element info { ... }  
chapter.info = element info { ... }
```

```
book = element book { book.info, ... }  
chapter = element chapter { chapter.info, ... }
```

Notes

- RELAX NG Compact Syntax fits better on the slides
- The examples are slightly simplified from the DocBook NG schema.

Info Elements in More Contexts

It (might) be nice to have **info** elements in more contexts:

```
<para><info>  
  <indexterm>  
    <primary>Extreme Markup Languages</primary>  
  </indexterm>  
</info>Some text.</para>
```

Info Elements in More Contexts

In DTDs, we'd have to say `(#PCDATA | ... | info | ...)*` which would allow:

```
<para>Some<info>...</info> text.</para>
```

In RELAX NG, we can say: `(info?, (text | ...))*` which has the semantic we want.

Required Titles (Valid)

Some elements must have titles, but they can appear in one place or another:

```
<article>  
<title>Some Article Title</title>  
<para>Some content.</para>  
</article>
```

```
<article>  
<articleinfo>  
<title>Some Article Title</title>  
<author><firstname>Jane</firstname>  
<surname>Doe</surname></author>  
</articleinfo>  
<para>Some content.</para>  
</article>
```

Required Titles (Invalid)

I said “in one place *or* another”:

```
<article>  
<para>Some content without a title.</para>  
</article>
```

```
<article>  
<title>Is This the Title?</title>  
<articleinfo>  
<title>Or Is This?</title>  
</articleinfo>  
<para>Some content.</para>  
</article>
```

Required Titles

```
title.opt = title? & titleabbrev? & subtitle?  
title.req = title & titleabbrev? & subtitle?
```

```
info.notitle =  
  element info { (author|...)* }  
info.titlereq =  
  element info { title.req, (author|...)* }
```

```
element article {  
  (title.req, info.notitle) | info.titlereq,  
  ...  
}
```

(This isn't *exactly* the same semantic.)

Co-Constraints (DTD)

DTDs don't support co-constraints:

```
<!ENTITY biblio.class.attribute "  
    class      (doi|isbn|issn|libraryofcongress  
                |pubnumber|uri|other) #IMPLIED  
    otherclass CDATA          #IMPLIED  
">
```

The desired semantic is:

- If **class** is “other”, then **otherclass** must be specified, otherwise
- The **otherclass** must not be specified.

Co-Constraints

RELAX NG does:

```
biblio.class-enum.attribute =  
  attribute class {  
    "doi"  
    | "isbn"  
    | "issn"  
    | "libraryofcongress"  
    | "pubnumber"  
    | "uri" }?
```

```
biblio.class-other.attributes =  
  attribute class { "other" }?,  
  attribute otherclass { xsd:NMTOKEN }
```

Co-Constraints (Continued)

```
biblio.class.attrib =  
    (biblio.class-enum.attribute  
     | biblio.class-other.attributes)
```

Untangling Tables

- DocBook uses CALS Tables. In DocBook V4.3, we added HTML Tables.
- CALS and HTML tables have overlapping element names with different content models.
- CALS and HTML tables have attributes with the same name and *intentionally* disjoint enumerated values.
- In DTDs, we just make a union...

Untangling Tables

```
<!ELEMENT table ((thead?, tfoot?,  
                 (tbody|tr+)) | tgroup)>
```

```
<!ATTLIST table  
      frame      (above | all | below | ...  
                 ... | void | vsides) #IMPLIED  
>
```

```
<!ELEMENT tbody (tr+ | row+)>
```

Untangling Tables

```
html.table =  
  element table {  
    attribute frame {  
      "void"      | "above"   | "below"  
      | "hsides"  | "vsides" | "lhs"  
      | "rhs"     | "box"    | "border" }?,  
      ((html.thead?, html.tfoot?, html.tbody)  
      | html.tr+)  
    }  
  }
```

```
html.tbody =  
  element tbody { html.tr+ }
```

Untangling Tables

```
cals.table =  
  element table {  
    attribute frame {  
      "all"  
      | "bottom"  
      | "none"  
      | "sides"  
      | "top"  
      | "topbot" }?,  
    cals.tgroup  
  }  
  
cals.tbody =  
  element tbody { cals.row+ }
```

Untangling Tables

```
table = html.table | cals.table
```

This allows any HTML table or any CALS table, but no invalid mixture of the two models.

Real Datatyping

A small number of elements and attributes benefit from real datatypes.

- **date**, **pubdate**, etc. are real dates (maybe).
- **startinglinenumber** is an integer.
- **cols** on **tgroup** is a positive integer.

Extra-Grammatical Constraints

Grammar based validation technologies (like RELAX NG) and rule based validation technologies (like Schematron) are naturally complimentary. Mixing them allows us to play to the strengths of each without stretching either to enforce constraints that they aren't readily designed to enforce.

Extra-Grammatical Constraints

- Exclusions. (High on my list of features for a future version of RELAX NG.)
- A version attribute on the root element.
- Enforcing implicit constraints. (In a segmented list, the number of segments in each list item has to be the same as the number of titles specified.)
- Enforcing referential integrity constraints. (A cross-reference on a **footnoteref** must point to a **footnote**.)

RELAX NG + Schematron

```
db.book =  
  [  
    s:rule [  
      context = "/db:book"  
      s:assert [  
        test = "@version"  
        "The root element must have a version attribute"  
      ]  
    ]  
  ]  
element book {  
  book.attlist,  
  book.info,
```

RELAX NG + Schematron (Continued)

```
(navigation.components | components | division
}
```

There are validators that will enforce both sets of constraints.

Customization

- The ability to customize DocBook is *critically* important.
- Many users subset DocBook.
- Many users extend DocBook.
- To be successful, these operations must be (at least) as easy as DTD customization, preferably easier.

Two Customization Examples

1. Remove **procedure**.
2. Add an **exercise** element.

Removing Procedures from the DTD

```
<!-- DocBook XML V4.3 No Procedures Subset -->
```

```
<!ENTITY % ebnf.block.hook "">
```

```
<!ENTITY % local.compound.class "">
```

```
<!ENTITY % compound.class
```

```
  "msgset | sidebar | qandaset
```

```
    %ebnf.block.hook;
```

```
    %local.compound.class;">
```

```
<!ENTITY % procedure.content.module "IGNORE">
```

```
<!ENTITY % task.content.module "IGNORE">
```

```
<!ENTITY % sidebar.element "IGNORE">
```

```
<!ENTITY % qandaset.element "IGNORE">
```

```
<!ENTITY % qandadiv.element "IGNORE">
```



Removing Procedures from the DTD (Continued)

```
<!ENTITY % question.element "IGNORE">
```

```
<!ENTITY % answer.element "IGNORE">
```

```
<!ENTITY % revdescription.element "IGNORE">
```

```
<!ENTITY % caution.element "IGNORE">
```

```
<!ENTITY % important.element "IGNORE">
```

```
<!ENTITY % note.element "IGNORE">
```

```
<!ENTITY % tip.element "IGNORE">
```

```
<!ENTITY % warning.element "IGNORE">
```

```
<!ENTITY % docbook.dtd PUBLIC "-//OASIS//DTD DocBook  
"http://docbook.org/xml/4.
```

```
%docbook.dtd;
```

```
<!ENTITY % my.sidebar.mix
```

Removing Procedures from the DTD (Continued)

```
"%list.class; |%admon.class;
|%linespecific.class; |%synop.class;
|%para.class; |%informal.class;
|%formal.class;
|%genobj.class;
|%ndxterm.class;          |beginpage
%local.sidebar.mix;">
```

```
<!ELEMENT sidebar (sidebarinfo?,
                   (%formalobject.title.content; )
                   (%my.sidebar.mix;)+)>
```

```
<!ENTITY % my.qandaset.mix
"%list.class;          |%admon.class;
```



Removing Procedures from the DTD (Continued)

```
|%linespecific.class; |%synop.class;  
|%para.class; |%informal.class;  
|%formal.class;  
|%genobj.class;  
|%ndxterm.class;  
%local.qandaset.mix;">
```

```
<!ELEMENT qandaset (blockinfo?, (%formalobject.ti  
    (%my.qandaset.mix;)*,  
                                (qandadiv+|qandaentry+))>
```

```
<!ELEMENT qandadiv (blockinfo?, (%formalobject.ti  
    (%my.qandaset.mix;)*,  
    (qandadiv+|qandaentry+))>
```



Removing Procedures from the DTD (Continued)

```
<!ELEMENT question (label?, (%my.qandaset.mix;)+)
```

```
<!ELEMENT answer (label?, (%my.qandaset.mix;)*, q
```

```
<!ENTITY % my.revdescription.mix  
  "%list.class; |%admon.class;  
  |%linespecific.class; |%synop.class;  
  |%para.class; |%informal.class;  
  |%formal.class;  
  |%genobj.class;  
  |%ndxterm.class;  
  %local.revdescription.mix;">
```



Removing Procedures from the DTD (Continued)

```
<!ELEMENT revdescription ((%my.revdescription.mix
```

```
<!ENTITY % my.admon.mix  
    "%list.class;  
    |%linespecific.class; |%synop.class;  
    |%para.class; |%informal.class;  
    |%formal.class; |sidebar  
    |anchor|bridgehead|remark  
    |%ndxterm.class; |beginpage  
    %local.admon.mix;">
```

```
<!ELEMENT caution (title?, (%my.admon.mix;)+)  
                %admon.exclusion;>
```

Removing Procedures from the DTD (Continued)

```
<!ELEMENT important (title?, (%my.admon.mix;)+)  
                    %admon.exclusion;>
```

```
<!ELEMENT note (title?, (%my.admon.mix;)+)  
               %admon.exclusion;>
```

```
<!ELEMENT tip (title?, (%my.admon.mix;)+)  
              %admon.exclusion;>
```

```
<!ELEMENT warning (title?, (%my.admon.mix;)+)  
                  %admon.exclusion;>
```

```
# DocBook NG "Bourbon" No Procedures Subset

namespace db = "http://docbook.org/docbook-ng"
default namespace = "http://docbook.org/docbook-ng"

include "docbook.rnc" {
    db.procedure = notAllowed
}
```

Adding Exercises to the DTD

```
<!ENTITY % local.formal.class "|exercise">
```

```
<!ENTITY % docbook.dtd PUBLIC "-//OASIS//DTD DocBook  
"http://docbook.org/xml/4.  
%docbook.dtd;
```

```
<!ELEMENT exercise %ho; (blockinfo?, (%formalobject  
(%example.mix;)+)  
%formal.exclusion;>
```

```
<!ATTLIST exercise  
role CDATA #  
%common.attrib;  
>
```

Adding Exercises to the RELAX NG Schema

```
# DocBook NG "Bourbon" Exercises Extension
```

```
namespace db = "http://docbook.org/docbook-ng"
```

```
default namespace = "http://docbook.org/docbook-ng"
```

```
include "docbook.rnc" {  
    extension.blocks |= exercise  
}
```

```
exercise = element exercise { db.title, all.block
```

Converting to NG

- XSLT can convert DocBook V4.x to DocBook NG.
- It successfully converts about 94% of the DocBook test cases.
- It doesn't convert elements that use entity attributes. It also doesn't convert old style `toc` markup.
- It can't convert tests that use block elements in inline contexts.

Compatibility

Creating XML Schemas

Creating DTDs

Creating XML Schemas

Use trang:

```
<xs:element name="book">
  <xs:complexType>
    <xs:sequence>
      <xs:group ref="db:db.book.info" />
      <xs:choice maxOccurs="unbounded">
        <xs:group ref="db:db.navigation.components" />
        <xs:element ref="db:db.components" />
        <xs:element ref="db:db.divisions" />
      </xs:choice>
    </xs:sequence>
    <xs:attributeGroup ref="db:db.book.attlist" />
  </xs:complexType>
</xs:element>
```

Creating XML Schemas (Continued)

```
</xs:complexType>  
</xs:element>
```

(Approximates the RELAX NG patterns for MathML/SVG extensions because of wildcard limitations.)

Creating DTDs

- Trang can't.
- One possible solution: declarative markup in the schema and use XSLT.
- Another solution: flatten aggressively and “reconstitute”.

Conclusions

Other Approaches

Conclusions

Things I Haven't Done

References

Other Approaches

- It might be technically possible to achieve some of the goals using DTDs simply by refactoring the parameter entities (again).
- W3C XML Schema would be better than DTDs.
 - I'm not sure the abstraction is right for schemas with lots of mixed content.
 - Determinism would still be a problem.
 - Local element declarations force customizations to “cascade” up the tree.
 - No support for co-constraints.
- Schematron could do it all, but it would require a lot of rules.

Conclusions

- The DocBook RELAX NG schema satisfies the redesign goals to a large extent: it looks and feels like DocBook while at the same time having simpler, more logical content models and better constraints.
- The RELAX NG grammar is demonstrably easier to customize, at least for those applications that can use the RELAX NG grammar directly.
- DocBook NG has only 356 elements and 1,701 patterns.

Things I Haven't Done

- Decided to do it ODDly.
- Satisfactorily address the “ubiquitous linking” problem.
- Build the DTD version.
- Finish fiddling with the build system.

References

- <http://docbook.org/docbook-ng/>, the DocBook NG Schemas.
- <http://sourceforge.net/projects/docbook/>, the DocBook SourceForge project.
- <http://docbook.org/tdg/en/html-ng/>, a special edition of *DocBook: The Definitive Guide* showing the DTD and (flattened) RELAX NG content models.
- <http://www.oasis-open.org/committees/docbook/>, the DocBook Technical Committee.
- <http://norman.walsh.name/threads/refactorDocBook>, a thread through the essays I've written about redesigning DocBook.