Introduction
to
XML Schema Languages

Deborah Aleyne Lapeyre
Mulberry Technologies, Inc.
17 West Jefferson St.
Suite 207
Rockville MD 20850
Phone: 301/315-9631
Fax: 301/315-8285
info@mulberrytech.com
http://www.mulberrytech.com

July, 2005

Mulberry
Technologies, Inc.

Introduction to XML Schema Languages

Appendixes

Introduction to XML Schema Languages

slide 1

Intro to XML Schema languages

- What and why constraints
- XML schema language history
- Purpose of validation and schemas
- Constraint languages for XML (2005)
- Selecting appropriate validation and constraint languages

slide 2

The Document Model ("schema")

Major component of an XML application

- Model for one type / class of information (a "document")
  (reference book, bank transfer, journal article, credit transaction, drug monograph)
- Set of rules describing how documents of that type can be marked up
- Agreement on a common vocabulary (tag set) for an application
- Written in a formal syntax (a constraint language)
- Examples: DTD, W3C XML Schema, RELAX NG

:"A schema represents an understanding of a world-chunk"
- Kendell Grant Clark

slide 3

Models Describe Constraints

- Provides rules and constraints for
  - elements
  - attributes
  - entities (DTD only)
- Determines
  - what's inside them
  - how they relate to each other
  - datatypes (valid units of data)

slide 4

Constraint Languages Express Rules

for example structural or data type rules

- **Journal Article** =
  journal metadata followed by
  article and issue metadata followed by
  article body followed by
  back matter (which is optional)
- **paragraph** = data characters and may include any of the following:
  Brand Drug Name, *Generic Drug Name*, *Street Name*, and *italic*
- **Birth** is a date which must contain
  4-digit year followed by hyphen followed by
  2-digit month followed by hyphen followed by
  2-digit day

slide 5

Why Use the Constraints of a DTD or Schema?

(validation, documentation, automation)

- Machine validation of document structure
- DTD or schema is a contract between producers and consumers (Both can validate to see if they got / sent what they expected)
- Formal specification of information *types* allows consistent downstream processing
- Supports interoperable families of documents
  - Ensure that information conforms to model (validation)
  - Parties don't have to share software or applications

slide 6

Purpose(s) of Validation (Prescriptive)

(and thus of schemas — note lower case "s")

Validation to protect the system from data it cannot handle

- Validation of markup (document structure)
- Validation of datatypes of leaf nodes (content validation)
- Integrity validation (links, codependence)
- Validation of typing and classing
- Business rules validation (well… maybe)

slide 7

Other Functions of “schemas” (Expressive)

- Tag set building
  - Constrains component names (semantics?)
  - Codify business rules (well… maybe)
- Communicate XML structure to people and machines
- Minimize resource requirements (all is stated)
- Top-down control of data
- *Enhance basic XML document (typing, default values)*

slide 8

XML Schema Language History

- DTD (Document Type Definition)
- The schema madness of 1997–2000
- Current state of three + one
  - XML 1.0 DTD
  - W3C XML Schema (aka WXS or XSD or XML Schema)
  - Relax NG
  - Schematron

slide 9

In the Beginning was the DTD

Document Type Definition

- That is what the W3C XML Recommendation defines
  (**`http://www.w3.org/TR/REC-xml/`**)
- DTDs came from SGML
  formally, a subset of SGML [ISO 8879:1986]
- DTDs were all there was 1996–1998

slide 10

The Schema Madness of 1997–2000

- `XDR` (Microsoft XML-Data Reduced)
- `SOX` (Commerce One Schema for Object-Oriented XML)
- `TRex` (James Clark Tree Regular Expressions for XML)
- `RELAX` (Murata Makoto et al. Regular Language description for XML)
- `DCD` (Tim Bray et al. Document Content Description: an RDF vocabulary designed for describing constraints)
- `DDML` (Document Definition Markup Language; originally XSchema; logical as opposed to physical)
- `DSD` (AT&T/BRICS Document Structure Descriptions; grammar-based CSS-like)
- `D4DTD` (Datatypes for DTDs)
- `Examplatron`, `Hook`, et al.

While W3C XML Schema (`XSD`) plodded along …

See Robin Cover: **`http://xml.coverpages.org/schemas.html`**

slide 11

Constraint Languages for XML (2005)

the shake out

- XML 1.0 DTD (*Document Type Definition*)
- XSD (*XML Schema Definition* Language, i.e. W3C XML Schema)
- RELAX NG
- and the plus one: Schematron

slide 12

Features of XML 1.0 DTD

- About structure of elements
- Validation is
  - all or nothing
  - changes the information set (default attributes for example)
- Defined by regular grammar (in lay terms, a simple program can infer the relation of tokens by parsing a character stream sequentially)
- Elements
  - are declared with content models
  - data content is conceived of as strings
- Referential integrity between elements enforceable through attributes (`ID` / `IDREF`)

Anything else is considered “higher level” and pushed up to another layer

slide 13

DTD Element Modeling

- Sequence
- Choice
- Mixed content
- Occurrence (1, 1+, zero, zero or more)

(W3C XML Schema and RELAX NG provide these too)

slide 14

DTD Content Models

```
<!ELEMENT name (given, family) >
```

- A `name` element must contain a
  - `given` element followed by
  - `family` element

```
<!ELEMENT para (#PCDATA | index)* >
```

- A `para` element contains text mixed with any number of `index` elements

slide 15

DTD Content Models (2)

```
<!ELEMENT name (family | given | patronymic |
                degree | title | honorific | prefix)*     >
```

- A `name` contains any number of `family`, `given`, `patronymic` (etc.) elements, mixed arbitrarily
- No way to limit how many of each appear

(In practice, these limitations are not very onerous for most text-processing applications)

slide 16

DTD Data-typing is Weak

- Only in attributes
- Attributes are declared as a (small) set of types
  - string (any data)
  - string (constrained to the same rules as names)
  - choice of named tokens `(draft | default | dead)`
  - labeled types (*notations*)
  - Uniqueness within document scope
    - `ID` for identifiers
    - `IDREF` for pointer to an `ID`

slide 17

Many Objections to DTDs

- Syntax is *not XML*
  - unique (some say bizarre)
  - requires single-purpose processor (XML parser)
- Restricted modeling functions
  - Can't make context-dependent models
  - AND functionality (bags for example) absent
- No element data typing (therefore no type validation)
- No inheritance (except by convention `xml:lang`)
- Documentation only as comments
- Lacks other things RDBMS schemas provide: referential integrity, co-occurrence constraints, classing and derivation

slide 18

That Said, Many Organizations Use DTDs (2005)

- Easy migration from earlier SGML
- Widely understood
- Easy to learn, read, write
- Tools are/were ubiquitous
- Breaks up the problem of validation
  - use DTDs for what they're good for
  - complete by supplementing with other methods
- Many of the advanced abilities (data typing and data types) are of limited use in processing narrative text

slide 19

XSD (W3C XML Schema)

- XML Schema Working Group of the W3C (ca 40 members at times)
  - Created a schema language for XML
  - Committee work representing a range of perspectives
  - Schema 1.0 became Recommendation May 2001
- XML syntax for the model (as well as the document)
  - Uses same parser as documents
  - XML editor can create schemas
  - Standard way to integrate documentation

slide 20

The Idea of W3C XML Schema

- Nearly all of the functions of a DTD (no general entities)
- Strong data typing
- Inheritance mechanisms
- Default values for elements
- Built-in documentation elements (`xs:annotation`)
- New relationships among elements (types and derivations)
- Extensible (maybe)

How XSD works

XML documents are strings of characters and entity references

- W3C XML Schema
  - does not operate over XML documents
  - assumes document has been parsed *into a "tree"* with all entities resolved (infoset)
    - composed of structures (nodes)
    - of named types (strings, numbers, booleans, what-have-you)
- Validating with a schema produces a *PSVI* ("post-schema-validation infoset")
  - trees and their "data content"
  - can be provided with labels (*type annotations*)
  - defaults (element and attribute) added
  - validity or invalidity outcomes

slide 22

W3C XML Schema Features

- Strong OO influence
- Deterministic
  - models cannot be ambiguous
  - at any point processor knows which element
- Partial validation allowed
  - start at specified element not root
  - wildcards specify what *not* to validate (black box)
  - wildcards specify *lax* validation
    (validate if have declarations; if not, don't)

slide 23

XSD has Strong Data Typing

treats the schema as a type definition system

- Strong data typing supports
  - code generation / configuration
  - data compiling into optimized forms
  - stylesheet validation against schema
- Unifies lexical with structural datatyping
  (makes markup safe for data processing)

slide 24

W3C XML Schema Content Models

- All that DTDs can do
- Occurrences include `maxoccurs`, `minoccurs`
- Structural modeling has
  - extensible types
  - local element declarations (context-dependent models)
  - substitution groups (element and attribute)
  - default elements and attributes
- Element wildcards (`xs:any` traditional bag; all in any order)
- Attribute wildcards (any number of attributes)
- Constrain elements in mixed content
- Arbitrary elements or composite models as keys

slide 25

Content Models “simple” and “complex” types

- Content models are now *complex types*
- *Simple types* map to more familiar (data content) datatypes
- Inheritance provided
  - by *extension* (add or compound values)
  - by *restriction* (limit allowable values)
- Aims to be consistent / compatible with traditional datatyping
  - useful for web services/web-based transaction processing
  - useful for integration

XSD Complex Type

(In DTD syntax: **`<!ELEMENT name (given, family) >`** ***)***

```
<xs:element name="given" type="xs:string"/>

<xs:element name="family" type="xs:string"/>

<xs:element name="name">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="given"/>
      <xs:element ref="family"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Validates

```
<name><given>John</given><family>Wayne</family></name>
```

but not

```
<name><family>Mifune</family><given>Toshiro</given></name>
```

XSD Choice

(In DTD syntax: **`<!ELEMENT name ((given, family) | (family, given)) >`** ***)***

```
<xs:complexType name="name.type">
  <xs:choice>
    <xs:sequence>
      <xs:element ref="given"/>
      <xs:element ref="family"/>
    </xs:sequence>
    <xs:sequence>
      <xs:element ref="family"/>
      <xs:element ref="given"/>
    </xs:sequence>
  </xs:choice>
</xs:complexType>

<xs:element name="given" type="xs:string"/>

<xs:element name="family" type="xs:string"/>

<xs:element name="name" type="name.type"/>
```

Validates *both*

```
<name><given>John</given><family>Wayne</family></name>
```

and

```
<name><family>Mifune</family><given>Toshiro</given></name>
```

XSD Type Derivation by Extension

```
<xs:complexType name="name.type">
  <xs:sequence>
    <xs:element ref="given"/>
    <xs:element ref="family" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="long-name.type">
  <xs:complexContent>
    <xs:extension base="name.type">
      <!-- this sequence is appended to that of the base type -->
      <xs:sequence>
        <xs:element ref="title" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:element name="given" type="xs:string"/>

<xs:element name="family" type="xs:string"/>

<xs:element name="title" type="xs:string"/>

<xs:element name="name" type="long-name.type"/>
```

- Validates

  ```
  <name><given>Charles</given><title>Prince of Wales</title>
  ```

- In this case, the content model extension allows the `title` element to appear after the base type's content
- (`long-name.type` inherits its model from `name.type` and extends it)

slide 29

W3C XML Schema Data Types

- Both built-in and derived types
- User can extend and derive types
- A datatype is a triple
  - a set of values
  - a set of lexical representations
  - a set of facets (properties)
    - fundamental (ordered, bounded, cardinality, numeric, equal)
    - constraining (length, maxlength, pattern, totalDigits, fractionDigits, et al.)

slide 30

Examples of XSD Data types

(there are more in the specification)

A Few DTD Remnants	A Few Basic Data Types	Other Useful Types
ID	string, normalizedString	gYear, gMonthDay, gMonth, gDay
IDREF, IDREFS	boolean	name, Qname
NMTOKEN, NMTOKENS	integer	duration
ENTITY, ENTITIES	decimal	time
NOTATION	float, double	date

slide 31

RELAX NG

- Pronounced as both “relax-N-G” and “relaxing”
- Collaboration between
  - James Clark (TREX) and
  - Murata Makoto (RELAX)
- Originally defined at OASIS
- Now integrated into DSDL [ISO 19575: Document Schema Definition Language] (as *Part 2: Grammar-based validation)*

slide 32

How RELAX NG Works

- Content models are not assimilated into a type system (treats the schema as a grammar definition)
- Instead, modeling is achieved with structural *patterns*
  - Reusable patterns
  - Local declarations
  - Not as complex as XSD modeling
  - Goes beyond XML DTD
  - Can handle text constraints XSD cannot

slide 33

RELAX NG Features

- Does not change the information set (no defaults or types added)
- Conceives of XML as text, also taking advantage of its tree structure
  - partial validation
  - validation of subtrees
- Has two syntaxes (programmatically interchangeable)
  - an XML one like XSD (for processing)
  - a compact syntax (human readable)
- *De facto* reference implementation (`jing` by James Clark)
- Based on a mathematical formalism
  (adaptation of hedge automata theory to trees)

slide 34

RELAX NG Content Models

- All that DTDs can do
- Constrain elements in mixed content
- Element/attribute co-constraints
- Groups and references inside patterns
- Interleave

RELAX NG Element Definitions

XML syntax:

```
<define name="given">
  <element name="given">
    <text/>
  </element>
</define>

<define name="family">
  <element name="family">
    <text/>
  </element>
</define>

<define name="name">
  <element name="name">
    <ref name="given"/>
    <ref name="family"/>
  </element>
</define>
```

Compact syntax:

```
given = element given { text }
family = element family { text }
name = element name { given, family }
```

RELAX NG XML Interleave (1)

```
<define name="given">
  <element name="given">
    <text/>
  </element>
</define>

<define name="family">
  <element name="family">
    <text/>
  </element>
</define>

<define name="title">
  <element name="title">
    <text/>
  </element>
</define>

<define name="name">
  <element name="name">
    <interleave>
      <zeroOrMore>
        <ref name="given"/>
      </zeroOrMore>
      <ref name="family"/>
      <optional>
        <ref name="title"/>
      </optional>
    </interleave>
  </element>
</define>
```

slide 37

RELAX NG XML Interleave (2)

- Can validate

```
<name>
  <given>Charles</given><given>Philip</given>
  <given>Arthur</given><given>George</given>
  <family>Mountbatten-Windsor</family><title>Prince of
Wales</title>
</name>
```

  and also

```
<name><title>Count</title><family>Dracula</family></name>
```

- Without RELAX NG's *interleave* feature, other top-down schema languages cannot express this model (a single `family` element mixed with any number of `given` elements and an optional `title` element)
- (Schematron can express it)

slide 38

RELAX NG Compact Syntax

Here is the compact syntax for the preceding example

```
given = element given { text }
family = element family { text }
title = element title { text }
name = element name { given* & family & title? }
```

slide 39

Schematron

- Does not define a “schema”, defines rules to validate instances
- Based on path expressions rather than grammars
- Originally a “meta-application” of XSLT
  - assertions are composed in XPath
    about what is to be expected or warned against
  - a generic stylesheet processes assertions and returns stylesheet
  - resulting stylesheet is run on the instance and
    generates a validation report
- Is currently being abstracted away from XSLT/XPath
- Originated by Rick Jelliffe (Topologi in Australia)
- Now part of DSDL (ISO/IEC 19757)
  (as *Part 3: Rule-based validation - Schematron*)

slide 40

Schematron Features

- Can be loose or strict, partial or comprehensive (most flexible)
- Write your own error and warning messages
- A scalable alternative to crafting your own XSLT queries for validation
- XPath fairly expressive for defining constraints
  - looser than top-down schema languages
  - can get creative (e.g. co-occurrence constraints)

(Warning: requires firm grasp of XPath to use properly)

slide 41

How Schematron Works

- Rules (a collection of constraints)
  - put together in named patterns
  - declare a context
  - act through a series of tests
    - report
    - assertion
  - `report:` test returns a message if the test is true
  - `assert:` test returns a message if the test is false
  - XPath is generally used to express the tests

slide 42

Schematron example

Sets up the `name` element as the context

```
<pattern name="Validate name against RNC
               { given* &amp; family &amp; title? }">
  <rule context="name">
    <assert test="family">’family’ element is missing</assert>
    <report test="*[not(self::given | self:: family |
                    self::title)]">Element not allowed</report>
    <report test="text()[normalize-space()]">Text content not
       allowed</report>
    <report test="count(family) > 1">More than one ’family’
       element is not allowed</report>
    <report test="count(title) > 1">More than one ’title’
       element is not allowed</report>
  </rule>
</pattern>
```

(RNC is the RELAX NG compact syntax)

slide 43

Selecting Appropriate Constraint/Validation Languages

- Considerations in Selection
- Why limit yourself to one?
- DSDL
- Non-schema validation
- Conversions between schema languages
- Tool support

slide 44

Considerations in Selection

- Nature of application
  - the great data versus narrative text divide
  - just validation or also authoring, query, etc.
- Suitability of particular modeling features (types, typing)
- Tradeoff between expressiveness and overhead / learnability
- Namespaces and inclusion of foreign XML vocabularies
- Ability to use XSLT to extract material from schemas
- Readability
- Need particular feature in pipeline (PSVI, character entities, etc.)

slide 45

Namespaces

```
<element xmlns="some-namespace.com/namespace"/>
```

- DTD: plays poorly with namespaces
  - it can be faked, but must be hard-wired
- XSD: uses namespaces, likes namespaces, wants namespaces
- RELAX NG: uses namespaces, and respects them
  (compatible with Namespace Routing Language)
- Schematron generally fine with namespaces

slide 46

Foreign Islands

(what namespaces make possible)

- People want this!
  - other formats inside HTML (SVG, CML)
  - HTML inside other formats (as documentation)
  - Genomic stuff inside journal articles
- DTD needs vocabularies built in (many consider this major reason not to use DTDs)
- Problems with foreign islands
  - garbage in the repository (no way to validate)
  - lose layering benefits
  - I need a “what?” reader?

slide 47

Extracting Information From Schemas

- In DTDs need programming to parse strings (maybe `perl`)
- This was a major selling point for all schema development
- Extract documentation (`xs:annotation`)
  - easy to do
  - packages do it for you
- But extracting information from a schema…
  - too many ways to do any one thing (particularly XSD)
  - "practically have to implement a schema processor in XSLT"

slide 48

Readability

- RELAX NG has a short form (RELAX NG Compact)
- DTDs are math-like and concise
- Nobody cares if Schematron is readable (but it is)
- XSD can be
  - written more readably
  - viewed through tools (hierarchy diagrams, etc.)

Readability Illustrated

(In its most extreme form)

Example in DTD Syntax

```
<!-- DTD                                                    -->
<!ELEMENT  PARA      (#PCDATA | FOOTNOTE | I | B | DATE)* >
<!ELEMENT  FOOTNOTE  (#PCDATA | BIBL)* >
<!ELEMENT  BIBL      (#PCDATA) >
<!ELEMENT  I         (#PCDATA) >
<!ELEMENT  B         (#PCDATA) >
<!ELEMENT  DATE      (#PCDATA) >
```

Same example in XML Schema Syntax

```
<xsd:element name="PARA">
<xsd:complexType mixed="true">
  <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="FOOTNOTE">
      <xsd:complexType mixed="true">
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
          <xsd:element name="BIBL" type="xsd:string"/>
        </xsd:choice>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="I" type="xsd:string"/>
    <xsd:element name="B" type="xsd:string"/>
    <xsd:element name="DATE" type="xsd:date"/>
  </xsd:choice>
</xsd:complexType>
</xsd:element>
```

slide 50

W3C XML Schema

Preferred if you

- Have data not text
  - and that data really is objects
  - e-commerce, test data, some government initiatives
- Consider XML mostly as a transaction/interchange form
- Need strong typing
  - banking, credit cards, transactions of various sorts
  - can use data types to generate code, validations, etc
- Have a pipeline that needs PSVI
- Use constantly changing structure with constantly changing partial validation requirements

You are quite likely using schemas now and have been for a while

slide 51

Using Combinations of Schemas

- Multipass validation with Schematron and {XSD, DTD, RELAX NG}
- Schematron rules can be embedded *inside*
  - W3C XML Schema
  - RELAX NG

  (really good for co-occurrence constraints)
- Make one schema primary, but provide all three forms

DSDL (ISO Standardizes XML Schema Languages)

Document Schema Definition Language interoperability framework

- ISO/IEC ITC 1/SC 34 Working Group 1
- XML document-oriented applications
  (as opposed to data-oriented)
- Extensible multi-faceted validation framework
- Philosophy
  - schema validation is too complex for just *one* language
  - bring together different validation-related task in one framework
    - Framework
    - RELAX NG and Schematron
    - Data typing
    - NVDL (Namespace-based Validation Dispatching Language)
    - Namespace-aware DTDs

slide 53

Converting Between Schema Languages

- There are tools including:
  - James Clark's *Trang* (`http://www.thaiopensource.com/relaxng/trang.html`)
- Typical scenario
  - pick one language as primary (maintain in one place)
  - programmatically make others when versioning
  - Handle incompatible functions
    - limit expressiveness to lowest common denominator OR
    - throw away some of the constraints when you switch

slide 54

Non-schema-language Validation

- XPath and XSLT (see next slide)
- XForms model without the form
- CAM (Content Assembly Mechanism) validates transactions
- CliX/xlinkit (XLink-based constraints with a rule editor)
- Various APIs
- Writing application code

slide 55

XPath and XSLT for Validation

- Validation by locating invalid data
- Node selection by
  - pattern
  - node counting
  - existence tests
  - numeric, boolean, string operations
- Particularly good for:
  - calculations
  - analysis of the tree as a whole
    (schemas focus on parent/child and content)
  - multidocument constraints
    (list of legal values in a separate file)
  - relationship between values (checksum)

slide 56

Tool Support

(Why this section does not have 100s of slides)

- Changes very rapidly
- Products vary in quality and conformance
- Product claims may be “optimistic”
- June W3C XML Schema 1.0 User Experiences and Interoperability
  `http://www.w3.org/2005/03/xml-schema-user-program`

slide 57

What can be Said

- All *older* tools support DTDs
- XSD support wide but *variable*
  - individuality does not equal interoperability
  - consulting requests to Mulberry
  - a tale of DTD to XSD conversion
- RELAX NG
  - growing
  - better in Europe
- Schematron is still its own thing

slide 58

Conclusion: Validation is a Many Layered Thing

- No need to use just *one* schema language
- Different languages to build models and/or validate
  - for different purposes
  - at different times in life-cycle
  - don't forget non-schema validators (XPath, XSLT, XForms, et al.)

slide 59

Resources

More information on XML schema languages

- Overviews
  - XML Cover Pages (Robin Cover): **`http://xml.coverpages.org/schemas.html`**
  - Books by Eric van der Vlist (O'Reilly)
- W3C XML Schema
  - Home page: **`http://www.w3.org/XML/Schema`**
  - Primer: **`http://www.w3.org/TR/xmlschema-0/`**
- RELAX NG:
  - Home page: **`http://www.relaxng.org/`**
  - Tutorial (XML syntax): **`http://www.relaxng.org/tutorial-20011203.html`**
  - Tutorial (compact syntax): **`http://relaxng.org/compact-tutorial-20030326.html`**
- Schematron:
  - Home page: **`http://xml.ascc.net/resource/schematron/schematron.html`**

Extreme Markup Languages Conference

- Annual conference sponsored by IDEAlliance (**http://www.idealliance.org**)
  - in Montréal, Québec (at a cute, funky hotel)
  - August 1–5, 2005
- Devoted to markup, markup languages, markup systems, markup applications, and software for manipulating and exploiting markup
- Conference chair's (unofficial) web site: **http://www.mulberrytech.com/Extreme/**
- Official conference web site: **http://www.extrememarkup.com/extreme/**

Mulberry Technologies, Inc.

Mulberry Technologies, Inc. is a consultancy specializing in XML and SGML design and training.

Find our web site, with schedules of public classes and upcoming events, at `http://www.mulberrytech.com`.

Consulting Services

Mulberry provides consulting to assist users in many facets of the creation and implementation of XML and SGML production systems. These services may include:

- creation and maintenance of DTDs and schemas;
- software evaluation and selection;
- software customization and stylesheet support;
- output specifications and XSLT stylesheet development;
- identification, selection, and customization of shared industry applications;
- backfile conversion assistance;
- corporate and publication requirements analyses;
- hotline-style telephone support for production problems;
- wording for RFPs and specifications; and
- evaluation of proposals.

Document Design

Document analysis is the process of deciding which parts of a document type need to be identified. Mulberry Technologies, Inc., believes the people best qualified to perform this type of analysis are the users and creators of the documents.

Mulberry's consultants guide document owners through the process of document analysis and record the results of the analysis. The results are presented both in report form and in XML, as DTD or schema modules.

DTD/Schema Construction, Testing, and Verification

We design DTDs and schemas to be flexible, reusable, and practical. We base our document designs on the analysis performed by the document creators and users, enhancing the decisions made with our extensive experience in implementation.

Training

We deliver custom classes and presentations appropriate for executives, managers, authors and editors, and production and technical people. Class time is spent on the skills and knowledge people need to perform their jobs.

We also offer public classes on:

- Working with XML/SGML files
- Concepts of structured data
- Technical introductions to the basic XML and SGML standards: XML, XSLT, XSL-FO, SGML, and SVG.

Colophon

We at Mulberry not only “Talk the Talk”, we “Walk the Walk”. This presentation was:

- Written in XML
- Converted to HTML for display using XSLT
- Information and resources at:
  `http://www.mulberrytech.com/slideshow/index.html`