Weaving the Web with XML: An Overview of the XML Family of W3C Recommendations

Henry S. Thompson

HCRC Language Technology Group
University of Edinburgh

World Wide Web Consortium

Markup Technology Ltd

XML has grown

- XML the language
- Namespaces
  - A great success
    - As long as you keep your expectations suitably low
- XSLT / XPath / DOM

- XLink / XPointer
- XML Schema
- Canonical XML / XML Signatures

- XML Query / XML Protocols

Why is XML a big deal?

- It is an official W3C Recommendation
- It is vendor-independent, platform independent, application independent,…
  - unlike Word documents, RTF documents, PDF documents, Postscript documents,…
- It is human readable
  - ditto (for most values of 'human')

Who is in charge of XML?

- XML is a W3C Recommendation
- The W3C is *The World Wide Web Consortium*, a voluntary association of companies and non-profit organisations. Membership costs serious money, confers voting rights. Complex procedures, with the Director (Tim Berners-Lee) having ultimate authority, guided by a committee of the whole called the Advisory Committee.
- The XML recommendation was written by the W3C's XML Working Group, which has since divided into a number of sub-groups

W3C Process

- Recommendations are the products of Working Groups
  - Any W3C member organisation can have representation on a WG

Becoming a Recommendation

- There are a series of public stages in the life of a Recommendation (or REC)
  - Working Draft
    - Maximum gap target: 3 months
  - Last Call
    - Public comment invited: WG must respond
  - Candidate Recommendation
    - Design is stable
    - Implementation Feedback invited
  - Proposed Recommendation
    - Advisory Committee review

Basic Concepts and Vocabulary

- What is an XML application?
  - We define an XML application as having
    - A form: what do all the documents involved in this application share?
      - A *vocabulary* (elements and attributes)
      - A *grammar* (how they are allowed to combine)
    - A function: what those elements and attributes *mean*
  - You already know the basic story about defining a syntax
    - You can use English (or French or . . .)
    - You *have* used a DTD
    - Now you can use an XML Schema

Components of the XML family

- XML Namespaces
  - Managing multiple vocabularies
- XSLT
  - Transforming XML
- XLink/XPointer
  - Connecting XML documents
- XML Schema
  - Defining XML document families
- XML Query
  - Database-style query language
- XML Protocols
  - XML-based communication

Namespaces for XML

- First, an example

```
<xh:p xmlns:xh='http://www.w3.org/1999/xhtml' >So
the result can be expressed as <!-- (a+b)² -->
 <mml:apply
   xmlns:mml='http://www.w3.org/TR/REC-MathML'>
   <mml:power/>
   <mml:apply>
    <mml:plus/>
    <mml:ci>a</mml:ci>
    <mml:ci>b</mml:ci>
   </mml:apply>
   <mml:cn>2</mml:cn>
 </mml:apply>
</xh:p>
```

Namespaces for XML, cont'd

- Where did those colons come from?
  - `xh:this, mml:that, xml:the_other`
- Two communities pushed for namespaces
  - Vendors, to manage the composition of document fragments
    - E.g. the inclusion of mathematical formulae in a document
  - Working groups, to reserve names without compromising users' freedom to name things
    - E.g. it wouldn't do for XML-link to reserve `<link>` for simple links, or XSL to reserve `<text>`

Namespaces, cont'd

- A W3C Recommendation was endorsed in January 1999
  - There was a lot of vendor pressure to get something in place, which caused political tension and at least one resignation from the WG
- The example illustrates how namespaces are declared, scoped and used

Namespaces defined

- You can use *qualified names*, consisting of two simple names separated by a colon (:)
- The namespace prefix is an abbreviation for a URI which uniquely identifies the owner/meaning/identity of the source of the name
- Using a namespace essentially cedes responsibility for the meaning of the qualified names to the owner of the URI

Declaring a namespace

- The association between namespace prefixes and URIs is declared using reserved attributes

```
<doc xmlns:mml='http://www.w3.org/TR/REC-MathML/'>
  ...</doc>
```

  - Anywhere inside the above `doc` element `mml` is a legal namespace prefix, standing for the URI given
  - There is also a mechanism for defining the default (unprefixed) namespace
- Declarations are scoped
- Prefixed names can be used for
  - Element type names
  - Attribute names

Namespace limitations

- An add-on for, not a rewrite of, the XML spec
- Validation is unchanged
  - Declarations must match instances character by character
  - Indeed there's no place for associating prefixes with URIs in DTDs
- There is no provision for merging DTDs
- The rules are confusing
  - Unprefixed attributes are never qualified
  - Unprefixed elements are qualified if and only if there is a default namespace declaration in scope

XSLT: Structure into form

- There is a stylesheet language called XSLT
  - Rules for transforming from one vocabulary to another
    - Common case: output vocabulary is HTML
    - Coming soon: HQ print-orientated vocabulary
- For example

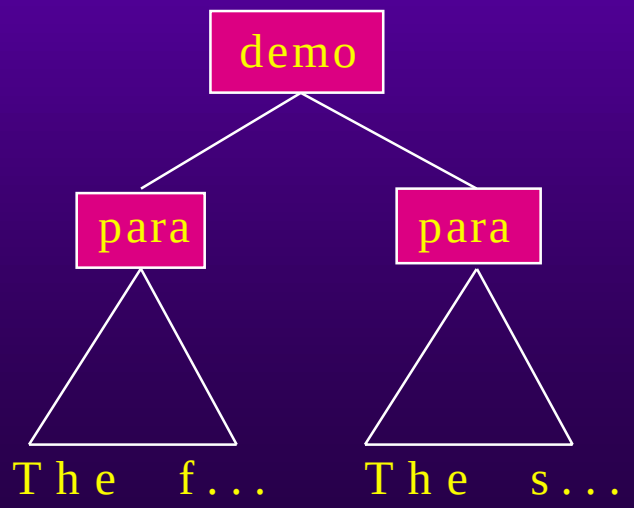
```
<xsl:template match='my:emph'>
  <xh:I><xsl:apply-templates/></xh:I>
</xsl:template>
```

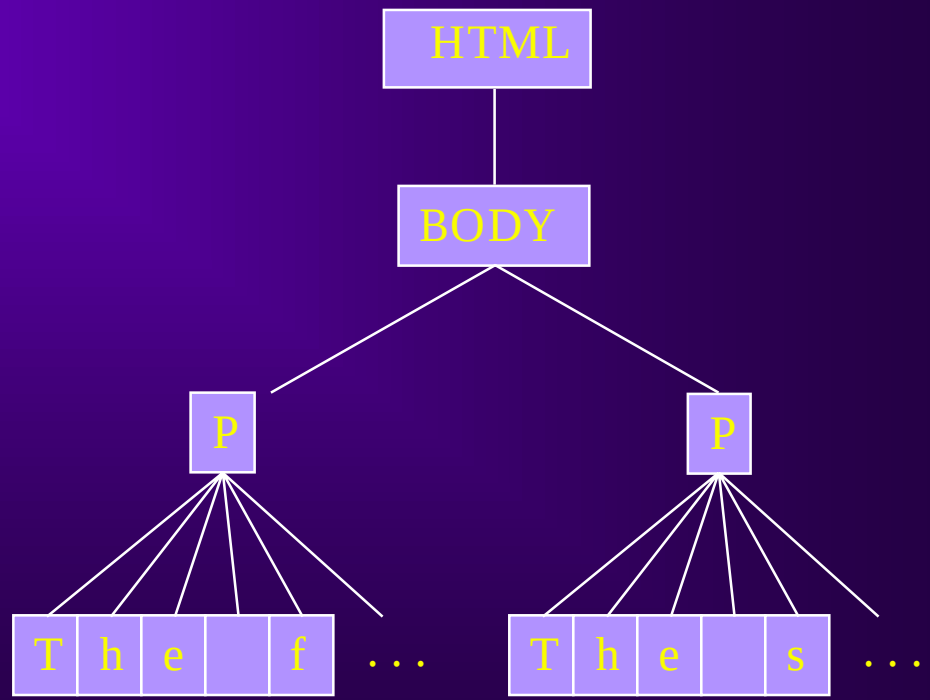
  will do part of the Transformation job
- XSLT uses iconic templates
  - You specify the result tree piece by piece

XSLT: Building the result tree

```
<x:templ match='demo'>
<HTML>
 <BODY>
  <x:apply-templates/>
 </BODY>
</HTML>
```

```
<x:templ match='para'>
<P>
 <x:apply-templates/>
</P>
```





Contrasting XSL with CSS

- CSS adds information to a document tree
  - Block or Inline
  - Font; margin/border/padding; text-form; colour
- XSL builds *and* decorates a *new* tree
  - So you can duplicate, reorder, eliminate
  - As well as provide CSS-derived rendering information
- A price list or table of contents is a good example of where the difference matters
  - Automatic numbering vs. by-hand
  - Automatic sharing of summary material and reliable connection vs. by hand

XSLT Status:

- W3C approved REC since November 1999
  - XSLT1.1 in Working Draft
- Many fully conformant implementations
  - Many are free
  - Including IE5
- Most are offline
  - Written in Java
- IE5 is online
  - Written in C++
  - file:///d:/work/xmlschema/structures/structures.xml

The Other Half of XSL

- XSL (Extensible Stylesheet Language) is intended to have two parts:
  - XSL Transformations (XSLT)
  - XSL Formatting Objects (XSL-FO)
- XSL Formatting Objects has Candidate Recommendation status
  - Comments period ended 28 February 2001
- Provides an alternative, *much* richer output vocabulary for XSL Transformations
  - Pages, columns, bidi, . . .

XML Protocols

- Replace application-specific wire protocols with XML
- Define an XML messaging story just above the transport layer
- Use the modularity of XML Schema to allow application-specific specialisation of payload
- Lack of consensus about exactly what the right level is

XML Protocol Status

- W3C Working Group recently formed
- Requirements document available
- Starting points
  - XML RPC
  - SOAP
    - Microsoft just announced a major development effort

What is XLink

- Together with XPointer, a reconstruction and enrichment of the hyperlink concept at the heart of the web
- Browsing is not the only application
  - “Follow Me” is not the only link semantics
- Take HTML’s `<A HREF=“…”/>`, and do it right
  - Not tied to a particular element type
  - Not restricted to two endpoints
  - Not restricted to be inline
- A careful separation between
  - The ontology and its notation (XLink)
  - The syntax of resource identification (XPointer/XPath)

XLink example

- By using attributes from the XLink namespace, you can make *any* element be a link

```
<ctime
  xmlns:xl='http://www.w3.org/1999/xlink'>
  The current time is
  <repl xl:type='simple'
        xl:href='ctime.xml'
        xl:show='replace'
        xl:actuate='onRequest'/>.
  </ctime>
```

- XLink gives you control where HTML freezes

XLink Status

- In Proposed Recommendation phase
- Several near-complete implementations recently announced
- Retrospective integration with e.g. XHTML and SVG underway

XPointer

- The WWW lets us point not only at whole resources, but also at parts
  - 'Fragment Identifier', the bit after the #
  - Within the http: URI scheme, meaning of a fragment identifier is Mime-type specific
  - XPointer is (part of) the story for text/xml and application/xml
- Identifying a locus in an XML document is a common requirement for XPointer and XSLT
  - There's a separate W3C Recommendation for this task, called XPath

XPointer Status

- Returned from Candidate Recommendation to Last Call
- Second Last Call period has ended
- Several implementations available

Distributed Dynamic Documents

- Ted Nelson identified a powerful link semantics over twenty years ago
  - He called it *transclusion*
  - We're only just able to implement it
- A document with transclusions in it is synthesised from the parts it points to
- The separation of form from content is crucial here
  - First you pull it together
  - Then you render it
- In the dynamic case, if what you're point at changes
  - You re-knit, and re-style
- I've used document language, but the layered story works here too

Linking vs. Messaging

- People tend to think about distributed applications at too low a level
  - RPC
  - Messages
- E-business and E-commerce are struggling to use XML versions of these technologies
  - With less success than originally expected
- I think distributed, dynamic documents are a better fit

XPath example

- Here's an XPath

```
//biblio[@author='Marx']
```

- It can be used in an XSLT template

```
<xsl:for-each
 select="//biblio[@author='Marx']">
 . . .
</xsl:for-each>
```

- Or in an a URL, via XPointer

```
<link
 xlink:href="
http://www.example.com/bibliography.xml#
  xpointer(//biblio[@author='Marx'])"/>
```

XPath Status

- Been a Recommendation since November 1999
- Many implementations, mostly as part of XSLT implementations
- Requirements out for 2.0, along with XSLT 2.0

XML Query

- Builds on XPath
- Support for following pointers and links
  - Including joins
- Support for the XML Schema type system
- Status:
  - Requirements nearly finished
  - First working draft early next year?

XML Canonicalization

- Defines a canonical character sequence for any XML document
- Became a Recommendation on 15 March
- A necessary pre-condition for XML Signature
- Makes decisions about
  - Whitespace
  - Attribute quotes
  - Entity expansions
  - Etc.

XML Signature

- Joint work with IETF
- Signing of (portions of) web resources
  - And messages
- Candidate Recommendation as of last November
  - Comments period ended in January

The great thing about standards

- Is that there are so many to choose from ☺
- OASIS vs. W3C
  - Notionally Applications vs. Infrastructure
  - Messaging an obvious area of overlap
    - ebXML (OASIS)
    - XML Protocols (W3C)

XML Schema: some details

- XML Schema is a language for defining the structure of XML documents
  - Notated in XML itself
- So there are elements defined for use in schemas to define. . .
  - Elements :-)
  - Attributes
  - Types

Terminology

- *Documents* have *structure*
  - Document *types*
  - Document *instances*
- Structure can be *defined*
  - Informally (D. S. D.)
  - SGML DTD
  - XML DTD
  - *Schema* using XML

Background

- SGML DTDs for D. S. D
  - Sperberg-McQueen
  - Others
- Considered for XML itself
- MCF, then RDF, now DCD, by Bray et al.
- *XML-Data*, two versions, now *XML-Data reduced*, by Layman et al., then Frankston and Thompson
- SOX, from Veo Corp.
- XSchema, from an *ad-hoc* group of designers

Document Structure

- Two relations are constitutive
  - Part-of
  - Kind-of
- Existing DSD mechanisms use Content Models to specify *part-of* relations
- But they only specify *kind-of* relations implicitly or informally
- Making kind-of relations *explicit* would make both understanding and maintenance easier

Why validate?

- A D. S. D. is a contract between producers and consumers
- It provides a guaranteed interface
- Producers validate to ensure they are providing what they promised
- Consumers validate to check up on producers
  - and to protect their applications
- Application authors validate to simplify their task
  - Leave error detection and analysis to the validating parser

Reconstructing DTDs

- The Schema DTD is expressed in vanilla XML
- Top level element types for declaring
  - Element types :-)
  - Entities
  - Notations
  - . . .
- Subordinate element types for declaring
  - Attributes
  - Content models
  - . . .

A simple example

```
<!ELEMENT text (#PCDATA|emph|name)*>
<!ATTLIST text
          timestamp NMTOKEN #REQUIRED>
```

```
<xs:element name="text">
 <xs:complexType content="mixed">
  <xs:choice minOccurs="0"
             maxOccurs="unbounded"
   <xs:element ref="emph"/>
   <xs:element ref="name"/>
  </xs:choice>
  <xs:attribute name="timestamp"
       type="xs:date" use="required"/>
</xs:complexType></xs:element>
```

An aside about terminology

- SGML and XML 1.0 talk about element *types*
- XML Schema to date has been more casual and just talked about elements
  - Meaning either an element in an instance
  - Or the abstraction which is described in a DTD or Schema
- Further confused by XML Schema making extensive use of *type*
- Also, *schema* means many different things to different people
  - I'll *try* always to say/write *XML* Schema. . .

The Schema Architecture: Static

- A document or an application or a user identifies a schema document
- Document and schema are well-formed XML
- The document is *schema*-valid w.r.t the schema
- (The schema is schema-valid wrt the schema for schemas)

The Schema Architecture: Dynamic

- An XML application (XSP) which schema-validates
- And augments the information with defaults, types, etc.

The state of play

- Chartered in the autumn of 1998
- Requirements document out in February of 1999
- Three component documents
  - Primer (non-normative)
  - Structures
  - Datatypes
- 8 public working drafts so far
  - May, September, November 1999
  - February, April, September, October 2000
  - March 2001:

    http://www.w3.org/TR/xmlschema-1/

    [contains pointers to previous drafts]
- Proposed Recommendation
  - Member comments due by 16 April 2001

XML Schema: Four requirements

- Reconstruct DTD functionality using XML
  - 'Eat your own cooking'
- Integrate Namespaces
  - Modular schemas for modular document types
- Provide a usable inventory of basic datatypes
  - For elements as well as attributes
- Support object-oriented design
  - *Kind-of* as well as *part-of*

Modular design

- Schemas are about elements and attributes named by *qualified names*
  - A *pair* of namespace name and local name
- A schema may include components for multiple namespaces
- Schema *documents* are primarily about one namespace
- But you can assemble multiple schema documents to build a single schema
  - `include` a schema document for the same namespace
  - `import` a schema for another namespace

Simple Type Definitions

- Treats attributes and sub-elements the same
- A frequently-expressed requirement for XML
- We need an inventory of simple types for strings

```
<xs:attribute name='birthday'
  type='xs:date'/>
```

- Other built-in simple types:
  - `boolean, decimal, uri, binary, timeInstant, timeDuration, . . .`
  - `Name, NMTOKEN, ID, IDREF, . . .`
  - `integer, NCName, QName, . . .`

Object-oriented design

- Type definitions are distinct from attribute and element declarations
  - The *tag-type* distinction
- Type definitions can be based on other definitions
  - restriction
  - extension
  - list
  - union

The XML Schema worldview

- Validity and well-formedness are XML 1.0 concepts
  - They are defined over character sequences
- Namespace-compliant is a Namespace concept
  - It's defined over character sequences too
- *Schema-validity* is the XML Schema concept
  - *It* is defined over XML document Infosets
- So the whole XML Schema exercise is predicated on and layered on top of XML 1.0 well-formedness plus Namespaces
  - Because they are constitutive of the Infoset

What's the Infoset?

- The XML 1.0 plus Namespaces abstract data model
- Defines a modest number of *information items*
  - Element, attribute, namespace declaration, ...
- Each has required and optional properties
  - Name, children, …

The Schema and the Infoset

- So crucially, schemas are about infosets, *not* character sequences
- You could schema-validate a DOM tree you built by hand!
  - Using a schema which exists only as a DOM tree ditto
- This simplifies things tremendously
  - but is hard to get your head around at first

Where did the Infoset come from?

- In the interests of time, XML 1.0 did *not* define its own data model
- So XPath had to define it
  - And XLink had to define it
  - And the DOM had to define it
- Finally, later than we'd have liked, we're about to get
  - The XML Information Set
    - Or Infoset
    - (now in Last Call)

What's the Infoset? Take two.

- The XML 1.0 plus Namespaces abstract data model
- What's an 'abstract data model'?
  - The thing that a sequence of start tags and attributes and character data represents
  - A formalization of our intuition of what it means to "be the same document"
  - The thing that's common to all the uninterestingly different ways of representing it
    - Single or double quotes
    - Whitespace inside tags
    - General entity and character references
    - Alternate forms of empty content
    - Specified vs. defaulted attribute values

What does it mean to be ‘abstract’?

- The Infoset is a description of the *information* in a document
- It’s a vocabulary for expressing requirements on XML applications
- It’s a bit like numbers
  - As opposed to numerals
- If you’re a type theorist
  - It’s just the definition of the **XML Document** type

What the Infoset isn't

- It's not the DOM
  - Much higher level
  - It's not about implementation or interfacing *at all*
- But you can think of it as a kind of fuzzy data structure if that helps
- It's not an SGML property set/grove
  - But it's close

Infoset details

- Defines a modest number of *information items*
  - Element, attribute, namespace declaration, comment, processing instruction, document ...
- Each one is composed of properties
  - Which in turn may have information items as values
- Both element and attribute information items have **[local name]** and **[namespace URI]** properties
  - Element information items have **[children]** and **[attributes]**
  - Attribute information items have a **[normalized value]**
- For more details, see my colleague Richard Tobin's talk on Thursday
  - He's the editor of the Infoset spec.

The Infoset Revolution

- We've sort of understood that XML is special because of its universality
  - Schemas and stylesheets and queries and … are all notated in XML
- But now we can understand this in a deeper way
  - The Infoset is the common currency of *all* the XML specs and languages
- XML applications can best be understood as Infoset pipelines
  - Angle brackets and equal signs are just an Infoset's way of perpetuating itself

The Infoset Pipeline begins

- An XML Parser builds an Infoset from a character stream
  - A streaming parser gives only a limited view of it
- A validating parser builds a richer Infoset than a non-validating one
  - Defaulted values
  - Whitespace normalisation
  - Ignorable whitespace
- If a document isn't well-formed, or is invalid, or isn't Namespace-conformant
  - It doesn't *have* an Infoset!

The XML Schema comes next

- Validity and well-formedness are XML 1.0 concepts
  - They are defined over character sequences
- Namespace-compliant is a Namespace concept
  - It's defined over character sequences too
- *Schema-validity* is the XML Schema concept
  - *It* is defined over Infosets

The Schema and the Infoset

- So crucially, schemas are about infosets, *not* character sequences
- You could schema-validate a DOM tree you built by hand!
  - Using a schema which exists only as data structures ditto

The Infoset grows

- Crucially, schemas are about much more than validation
  - They tell you much more than ‘yes’ or ‘no’
- They assign types to every element and attribute information item they validate
- This is done by adding properties to the Infoset
  - To produce what’s called the post schema-validation Infoset (or PSVI)
- So schema-aware processing is a mapping from Infosets to Infosets

The Infoset is transformed

- XSLT 1.0 defined its own data model
  - And distinguished between source and result models
- XSLT 2.0 will unify the two
  - And make use of the Infoset abstraction to describe them
- So XSLT will properly be understood as mapping from one Infoset to another

The Infoset is composed

- XLink resources (the things pointed to by XPointers) can now be understood as items in Infosets
- The XInclude proposal in particular fits in to my story
  - It provides for the merger of (parts of) one Infoset into another

The Infoset is accessed

- XML Query of course provides for more sophisticated access to the Infoset
- It also allows structuring of the results into new Infoset items

The Infoset is transmitted

- And finally XML Protocol can best be understood as parcelling up information items and shipping them out to be reconstructed elsewhere

A big step forward

- This is *so* much better than the alternative
  - Either
    - Pretending to talk about character sequences all the time
  - Or
    - Requiring each member of the XML standards family to define its own data model

Schemas at the heart

- I would say that, wouldn't I ☺
- Seriously, schema processing can be integrated into this story in a way DTDs could not
  - You may want to schema-process both before and after XInclude
  - Or between every step in a sequence of XSLT transformations
- We actually are missing a piece of the XML story
  - How do we describe Infoset pipelines?

Types and the Infoset

- The most important contribution to the PSVI
  - Every element and attribute information item is labelled with its type
    - Integer, date, boolean, …
    - Address, employee, purchaseOrder
- XPath 2.0 and XML Query will be type-aware
- Types will play a key role in the next generation of XML applications

The new challenge

- So how do we get back and forth between application data and the Infoset
  - Old answer
    - Write lots of script
  - New answer
    - Exploit schemas and types
- A type may be either
  - simple, for constraining string values
  - complex, for constraining elements which contain other elements

Mapping between layers

- We can think of this in two ways
  - In terms of abstract data modelling languages
    - Entity-Relation
    - UML
    - RDF
  - In concrete implementation terms
    - Tables and rows
    - Class instances and instance variables
- The first is more portable
- The second more immediately useful

Mapping between layers 2

- Regardless of what approach we take, we need
  - A vocabulary of data model components
  - An attachment of that vocabulary to types
- Sample vocabularies
  - entity, relationship, collection
  - table, row, column
  - instance, variable, list, dictionary
- Where should attachment be specified?
  - In the schema
    - convenient
  - Outside it
    - modular

Infoset Conclusion

- Think about things in terms of Infosets and Infoset pipelines
  - Modular
  - Powerful
  - Scalable
- Use XML Schema and its type system to facilitate mapping
  - Unmarshalling is easy
  - Marshalling takes a little longer

Overall Conclusions

- XML has a lot to offer e-Business and e-Commerce
  - Separating hype from reality is not easy
  - Careful requirements analysis is still the only sensible starting point
- Old paradigms are not always the right model
  - Creative exploration/exploitation of new architectures is needed
- Pilot first, before you bet the company
  - Look for help from established practitioners
- Start now, if you haven't already!