

Schemas, Types, Infosets and Pipes: Understanding the XML Revolution

Schemas, Types, Infosets and Pipes: Understanding the XML Revolution

Henry S. Thompson
World Wide Web Consortium,
School of Informatics
University of Edinburgh
and
Markup Technology Ltd.



© 2005 Henry S. Thompson



Introduction

- The three most important concepts in computer science are
 - abstraction
 - abstraction
 - and
 - abstraction
- XML is the newest tool for this purpose
 - But it's easy to miss the key points about XML
- In this talk my goal is to clarify the central role XML Schema plays

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

XML is ASCII for the 21st century

- ASCII (ISO 646) solved a fundamental interchange problem for flat text documents
 - What bits encode what characters
 - (For a pretty parochial definition of 'character')
- UNICODE/ISO 10646 extends that solution to the whole world
- XML thought it was doing the same for simple tree-structured documents
 - The emphasis in the XML design was on simplifying SGML to move it to the Web
 - XML didn't touch SGML's architectural vision
 - flexible linearisation/transfer syntax
 - for tree-structured documents with internal links

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

The old markup story

- Version 1
 - Hand-authored SGML and HTML
 - Marking up human-generated prose
 - For humans to read
- Version 1a
 - WYSIWYG-authored HTML
- Version 2
 - Hand-authored XML
 - Styled to HTML or print or . . .
 - Online or offline

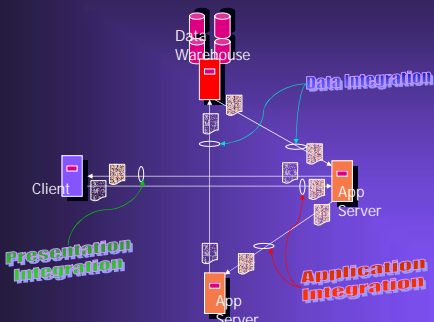
XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

The new markup story

- Machine-created XML
- Marking up application data
- For other applications to process
 - No human involvement at all

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

XML for integration



XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

Schemas, Types, Infosets and Pipes: Understanding the XML Revolution

What just happened!?

- The whole transfer syntax story just went meta, that's what happened!
- XML has been a runaway success, on a *much* greater scale than its designers anticipated
 - Not for the reason they had hoped
 - Because separation of form from content is *right*
 - But for a reason they barely thought about
 - Data must travel the web
- Tree structured documents are a useable transfer syntax for just about anything
 - So data-oriented web users think of XML as a transfer mechanism for their data

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

Catch 22

- The simplicity of the XML-as-data-transfer-mechanism story is appealing
- But it's a serious mistake to stop there
- Otherwise you're stuck with the following bad vision of an application which uses XML:



XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

XML processing as such

- We can do better than the "big lump of code" view of XML applications
- Processing with XML can often be simply achieved by just *processing XML*
 - Without writing any programs at all
- The first step towards this goal is provided by the idea of the XML Infoset

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

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, ...

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

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 got
 - The XML Information Set
 - Or Infoset
 - A W3C recommendation for how to *talk* about XML

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

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

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

Schemas, Types, Infosets and Pipes: Understanding the XML Revolution

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

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

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

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

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]**

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

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

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

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 isn't Namespace-conformant
 - It doesn't *have* an Infoset!

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

XML Schema comes next

- It's still a *generic* infoset at this point
 - XML Schema bridges over to an *application-specific* infoset
- Provides an XML document type for explicitly defining the structure of XML document types
- Much more than just validation
 - Modularity
 - Annotation
 - Type-based

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

Schemas, Types, Infosets and Pipes: Understanding the XML Revolution

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*
- We'll focus on the first and last today
 - Preparing for XML Query

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

Why validate?

- A Schema 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
 - "Validate at trust boundaries" *Dan Connolly*
- Application authors validate to simplify their task
 - Leave error detection and analysis to the validating parser

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

Reconstructing DTDs

- The Schema DTD is expressed in vanilla XML
- Elements for declaring
 - Elements :-)
 - Entities
 - Notations
 - Attributes
 - Content models
 - Types
 - ...

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

A simple example

```
<!ELEMENT text (#PCDATA|emph|xref)*>
<!ATTLIST text
    timestamp NMTOKEN #REQUIRED>
<xs:element name="text">
  <xs:complexType mixed="true">
    <xs:choice minOccurs="0"
      maxOccurs="unbounded">
      <xs:element ref="emph"/>
      <xs:element ref="xref"/>
    </xs:choice>
    <xs:attribute name="timestamp"
      type="xs:date" use="required"/>
  </xs:complexType></xs:element>
```

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

The Schema and the Infoset

- 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

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

Schemas and Types

- 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

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

Schemas, Types, Infosets and Pipes: Understanding the XML Revolution

Infoset Example

- Consider the following XML representation:

```
<department number="1" />
```
- We can present its infoset as a table:
 - [infosetExample.xls](#)
- Or after schema-validity assessment, its PSVI
 - [PSVIExample.xls](#)

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

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

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

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

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

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

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

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

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

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

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

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

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

The Schema Architecture: Static

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

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

The Schema Architecture: Dynamic

- An XML application (XSV, Xerces, MSV, MSXML4, ...) which schema-validates
- And augments the information with defaults, types, etc.
- Can be dropped in to an XML pipeline at many points
- Particularly the beginning and the end

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

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

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

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

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

Types and the Infoset

- The most important contribution to the PSVI
 - Every element and attribute information item is labelled with its type
 - Simple types for attributes (and simple elements)
 - Complex types for complex elements
- XPath 2.0 and XML Query will be type-aware
- Types will play a key role in the next generation of XML applications

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

Attribute Declarations

- The simple case
- An association between a qualified name (local name plus optional namespace names) and a simple type definition
- Determines a set of AIIs
 - [local name] and [namespace name] must match
 - [schema normalized value] must satisfy the simple type def'n
 - Two AIIs with the same name may have different types if they occur within different EIIs
- May include default/fixed value

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

Simple Type Definitions

- Treats attributes and simple 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, anyURI, hexBinary, dateTime, duration, . . .
    - QName, NOTATION, . . .
    - integer, NCName, ID, . . .

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

## Element Declarations

- An association between a qualified name and
  - A type definition (simple or complex)
  - A set of identity constraints
  - A substitution group head (optional)
- Determines a set of EIIs
  - [local name] and [namespace URI] must match\*
  - [children] must satisfy the type definition
  - [attributes] must satisfy the type definition
- May be scoped by a particular complex type definition

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

## Simple Type Definitions, cont'd

- Based on ISO 11404
- Distinguishes between *lexical* and *value* spaces
- Identifies *fundamental* and *constraining* facets
- For example, the decimal type has
  - Lexical space:  $([+-]?[0-9]*)?(\.[0-9]*)?$
  - Value space: the real numbers
  - Fundamental facets: Ordered: yes; Cardinality: countably infinite; Bounded: no; etc
  - Constraining facets: min, max, enumeration, ...

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

## Simple type definition example

```
<xs:simpleType name='fbodytemp'>
 <xs:restriction base='xs:decimal'>
 <xs:totalDigits value='4' />
 <xs:fractionDigits value='1' />
 <xs:minInclusive value='97.0' />
 <xs:maxInclusive value='105.0' />
 </xs:restriction>
</xs:simpleType>
```

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

## Complex Type Definitions

- Constrains [attributes]
  - Required/optional
  - Local or global declarations
- Constrains [children]
  - Finite-state grammar for EII sequence
  - Interpolated characters allowed or not
  - Simple type for text-only case
  - Local or global declarations

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

## Complex Type Definition, cont'd

- Membership assessment is two-part
  - Locally valid
    - All required attributes present
    - No non-declared attributes present
    - Sequence of names of EII children, if any, satisfies content model
  - Recursively valid
    - All attributes/children not exempted have known types
    - All attributes with known types satisfy them
    - All EII children with known types satisfy them

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

## Model Groups

- For content models
- One of
  - xs:choice
  - xs:sequence
  - xs:all
- With minOccurs and maxOccurs attributes
- Contents
  - Other groups
  - Element refs or inline declarations

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

## Type definition by derivation

- XML Schema makes it easy to construct type definitions which restrict or extend other type definitions, by specifying only the method of derivation and the differences between the base and derived type definitions.

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

## Derived type definition

```
<xs:simpleType
 name='healthyBodytemp'>
 <xs:restriction base='bodytemp'>
 <xs:maxInclusive value='99.5' />
 </xs:restriction>
</xs:simpleType>
```

- The **healthyBodytemp** type definition is defined by closing down the permitted range of **bodytemp**. We say it 'inherits' the other facets of **bodytemp**, so the 'effective type definition' of **healthyBodytemp** is

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

## Effective type definition

```
<xs:simpleType
 name='healthyBodytemp'>
 <xs:restriction
 base='xs:decimal'>
 <xs:maxInclusive value='99.5' />
 <xs:totalDigits value='4' />
 <xs:fractionDigits value='1' />
 <xs:minInclusive value='97.0' />
 </xs:restriction>
</xs:simpleType>
```

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

## Extension for complex types

- The next simplest case is extension for complex types
- Start with this base type

```
<xs:complexType name='name'>
 <xs:sequence>
 <xs:element name='title'
 minOccurs='0' />
 <xs:element name='forename'
 minOccurs='0'
 maxOccurs='unbounded' />
 <xs:element name='surname' />
 </xs:sequence>
</xs:complexType>
```

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

# Schemas, Types, Infosets and Pipes: Understanding the XML Revolution

## Derived type definition

```
<xs:complexType name='fullName'>
 <xs:complexContent>
 <xs:extension base='name'>
 <xs:sequence>
 <xs:element name='genMark'
 minOccurs='0' />
 </xs:sequence>
 </xs:complexContent>
 </xs:complexType>
```

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

## The effective type definition

```
<xs:complexType name='fullName'>
 <xs:sequence>
 <xs:element name='title'
 minOccurs='0' />
 <xs:element name='forename'
 minOccurs='0'
 maxOccurs='unbounded' />
 <xs:element name='surname' />
 <xs:element name='genMark'
 minOccurs='0' />
 </xs:sequence></xs:complexType>
```

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

## Restriction for complex types

- Restriction for complex types is harder to handle syntactically, because of the significance of linear order in content models, but the semantics are completely parallel to the simple type case:

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

## Restriction example

```
<xs:complexType name='simpleName'>
 <xs:complexContent>
 <xs:restriction base='name'>
 <xs:sequence>
 <xs:element name='forename'
 minOccurs='1' />
 <xs:element name='surname' />
 </xs:sequence>
 </xs:restriction>
 </xs:complexContent>
</xs:complexType>
```

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

## Restriction and Inheritance

- There must be a one-to-one line-up between the particles in the restriction and the particles in the base
- Unlike `<simpleType>` case, what you see is what you get, so the *effective type definition* of **simpleName** is just the same
- But for attributes, it works like the `<simpleType>` case, with unmentioned attributes being inherited unchanged

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

## Wildcards

- The `<any/>` content model particle, in all of its forms, allows EIs regardless of local name
  - A true 'any', i.e. any well-formed XML
- `<any/>` allows a single well-formed element information item
  - the namespace attribute allows finer control
    - ##any
    - ##other
    - ##targetNamespace
    - ##local
- `<anyAttribute/>` has a similar semantics for attributes

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

## Content Model example

```
<xs:complexType>
 <xs:sequence>
 <xs:choice>
 <xs:element ref="title"/>
 <xs:element ref="intro"/>
 </xs:choice>
 <xs:element ref="p"
 minOccurs="0"
 maxOccurs="unbounded"/>
 </xs:sequence>
</xs:complexType>
```

- DTD equivalent  
( (title|intro), p\* )

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

## Schemas and types: summary

- Why the big emphasis on types?
  - Static type checking to guarantee success
  - Smooth connections with application data
- Parallels the move towards stronger typing in computer languages over the last 20 years

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

## Type checking

- XML Query and XPath 2.0/XSLT 2.0 will support static type checking
    - If you promise to schema-validate, your queries and XPaths can be checked for correctness ahead of time
- ```
employee[@hire > date("1995-01-01") and  
@salary > 15000.00]
```
- Static checking ensures the complex type for the `employee` element has the necessary attributes and the simple types for its `hire` and `salary` attributes are of type `xs:date` and `xs:decimal` respectively

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

Declarative Data Binding

- When you finally have to bind to code, XML Schema is the natural place to manage the conversion
 - Schema spec. allows for **annotation** of element declarations and type definitions
 - We can use this to augment straight type-based marshalling and unmarshalling with information about exceptions and modifications
- At Edinburgh we've been exploring and implementing this approach
 - But it's still at the research stage

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

Mobile device example

- [thanks to Barbara Heikkinen]
- Suppose schema validation were light-weight and fast enough to embed in mobile devices
- Then type-aware processing would be easy
- You could easily find every instance of a **time** or a **dateTime** in an incoming message and
 - Convert it to the current local timezoneor
 - Check it against the diary/calendar

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

Schemas at the heart

- I would say that, wouldn't I ☺
- Seriously, schema processing can be integrated into the XML pipeline 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?

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

Schemas, Types, Infosets and Pipes: Understanding the XML Revolution

Declarative Pipelines

- Sun (and Markup and ...) proposed a simple XML document type describing straight-through sequences of XML operations
 - [demo]
- Markup has designed a rich data-flow language for XML processing
 - Cf. the "XML middleware" functionality that Markku appealed to in his talk this morning
 - [illustration]
- This approach pulls together all the different aspects of declarative XML processing we've been talking about

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

Conclusion

- So we've talked about
 - Schemas and Types
 - Infosets and Pipes
- The power of abstraction and declarative approaches to processing XML *as XML*
- This lays the groundwork for the talks that follow

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson

Conclusion, cont'd

- I've focussed on what you can do now, with mature standards which are well understood
 - And with many implementations you can pull together today
- No Java, No AppServer, Just XML
- Next time you think about an application, try thinking of code as the *last* thing you'll write, not the first
- XML may well be *all* you need to get the job done

XML Schema overview W3C, Brisbane, 2005-01-14 Henry S. Thompson