W3C XML Schema:

what you might not know
(and might or might not like!)

Noah Mendelsohn

Distinguished Engineer

IBM Corp.

October 10, 2002

Topics

- Quick review of XML concepts
- Why XML Schema?
- What is XML Schema?
- Where do schemas come from?
- A few validation tricks
- Wrapup

Warning!

To save screen space, some examples are simplified. Namespace decls. are omitted, only the key parts of schema declarations are shown, etc.

Quick review of XML concepts

This is an XML document

```
<?xml version="1.0"?>
  <e1>
     <e2>
        <e3  a1="123" />
     <e2>
</e1>
```

Infoset: the XML data model

```
<?xml version="1.0"?>
  <e1>
    <e2>
       <e3  a1="123" />
    <e2>
</e1>
```

Element information item:

[localname] = "e3"

[children] = Attribute info item for a1

Attribute information item:

[localname] = "a1"

[children] = "1", "2", "3"

More on XML infosets

- XML 1.0 describes only documents with angle bracket syntax “<...>”
- Infosets also describe DOM, SAX, and other representations
- *XML Schema validates infosets...applies to all of the representations*
- *XML Schema can validate from any element information item (e.g. e1 or e2)*

Why XML Schema?

What are schemas for?

- Contracts: agreeing on formats
- Tool building: know what the data will be *before* the first instance shows up
  - Database integration
  - User interface tools
  - Programming language bindings
- Validation: make sure we got what we expected

What is XML Schema?

This is an XML document

```
<?xml “version=“1.0”?>
<myns:e1
        xmlns:myns=“http://example.org/myns”
        xmlns:yourns=“http://example.org/yourns”>
    <myns:e2>
      <yourns:e1 a1=“xyz”/>
      <myns:e3   a1=“123” myns:a1=“456”/>
      <yourns:e1 myns:a1=“456”/>
    </myns:e2>
    <yourns:e4/>
</myns:e1>
```

~~This is an XML schema~~

```
<xsd:schema
</
</xsd:schema>
```

Wrong! This is not an XML Schema,
it's an XML schema *document*!

This is an XML schema document

```
<xsd:schema
        targetNamespace=“http://example.org/myns”
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        ..namespaces ommitted to protect innocent..”>
  <!-- declare element e1 ->
  <xsd:element name=“e1”>
     <xsd:sequence>
       <xsd:element name=“e2”/>
       <xsd:element ref=“yourns:e4”/>
     </xsd:sequence>
  </xsd:element
 </xsd:schema>
```

This is an XML document

```
<?xml “version=“1.0”?>
<myns:e1>
        xmlns:myns=“http://example.org/myns”
        xmlns:yourns=“http://example.org/yourns”>
    <myns:e2>
      <yourns:e1 a1=“xyz”/>
      <myns:e3   a1=“123” myns:a1=“456”/>
      <yourns:e1 myns:a1=“456”/>
    </myns:e2>
    <yourns:e4/>
</myns:e1>
```

To validate this, we need >1 schema document

Import brings in declarations for other namespaces

```
<xsd:schema
        targetNamespace="http://example.org/myns.xsd"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        ..namespace ommitted to protect innocent..">
  <import namespace="http://example.org/yourns"
      schemaLocation="http://example.org/yourns.xsd>
  <xsd:element name="e1">
     <xsd:sequence>
       <xsd:element name="e2"/>
       <xsd:element ref="yourns:e4"/>
     </xsd:sequence>
  </xsd:element
  <!-- declare element e2 ->
  <xsd:element name="e2" type="…"/>
</xsd:schema>
```

Terminology

Schema Document	An XML Document (Infoset) with definitions & decls for 1 namespace
Component	A single definition or declaration
Schema	All the components needed for validation

Cool tricks with components

- In memory schemas
- Handy tools for working with schemas
  - Build the components for you
  - Resolve subtyping across namepaces, etc.
  - Examples:
    - `http://www.eclipse.org/xsd`
    - Henry Thompson's XSV
- Conformance testing

How to read the spec.

- 3.3 Element Declarations
  - 3.3.1 The Element Declaration Schema Component
  - 3.3.2 XML Representation of Element Declaration Schema Components
  - 3.3.3 Constraints on XML Representations of Element Declarations
  - 3.3.4 Element Declaration Validation Rules
  - 3.3.5 Element Declaration Information Set Contributions
  - 3.3.6 Constraints on Element Declaration Schema Components
- Warning: the spec. never gives any rule twice!

Post-schema validation infoset (PSVI)

- Fearsome title, simple concept
- Infoset: the data model for an XML document...tells you what you can know (that matters) after a parse.
- PSVI: tells you what you can know after a validation
  - What parts of doc are valid?
  - Per which types?
  - Default values
  - Etc.

Self-describing vs. schema-described docs

- You can use xsi:type in your documents:

```
<e xsi:type="xsd:integer">123</e>
```

- Use xsi:type with built ins (and no attributes)
  - Your document is nearly self-describing
  - SOAP encoding supports this
- xsi:type with your own types
  - Partially self-describing
  - You know the type names – need schema to know what types are
  - SOAP 1.2 Encoding supports this too!

Where do schemas come from?

How are schema components found?

- *In short, wherever you want!*
- Hint from schema:
  - <xsd:import ns="..." schemaLocation="yyy.xsd"/>
- Hint from instance:
  - <myns:e1 schemaLocation=" myNSUri yyy.xsd"/>
- Processor command line or config
- Compiled into application (validating HTML editor)

Why all this flexibility?

- > 1 schema / namespace (versions, bug fixes, experiments, etc.)
- Who gets control?
  - “Docheads” want to name schema in instance
  - eCommerce: do you trust the schema named in a purchaseOrder?
  - Ultimately the application chooses
- Synthetic: DB builds it dynamically

Streaming

- Most validation can be done 1 pass
  - Id/idref, key/keyref require limited lookaside
- Problem:
  - ```
    <myinstance>
      …10Mbytes of data here…
      <!-- oops..need a new schema! -->
      <newns:a schemaLoc="newnsUri xxx">
      …lots more data…
    </mysinstance>
    ```
- Answer:
  - *Assemble schema incrementally or in advance*
  - Result must be same – can't tell which from the outside!
    ```

Our language vs. your language – why <import>?

```
<xsd:schema targetNamespace="http://example.org/ns1"
        xmlns:ns1="http://example.org/ns1"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:element name="ns1:X"/>
  …
  <xsd:element ref="ns1:X"/>
  …
</xsd:schema>
```

A fragment of a schema document...

Our language vs. your language – why <import>?

```
<xsd:schema targetNamespace="http://example.org/ns1"
        xmlns:ns1="http://example.org/ns1"
        xmlns:ns2="http://example.org/ns2"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <import namespace="http://example.org/ns2">

  <xsd:element name="X"/>
  …
  <xsd:element ref="ns1:X"/>
  <xsd:element ref="ns2:Y"/>
  …
</xsd:schema>
```

Add a reference to an external element...

...s. your ...uage – why <import>?

Unimported namespaces enhance the schema language

Imported namespaces enhance your language.

```
<xsd:sc      targetNames
      xm    :ns1="http:/
      xmln  ns2="http://exam      rg/ns2"
      xmlns xsd="http://www.      .org/2001/XMLSchema"
      xmlns:xsd2="http://www.w3.org/2004/XMLSchema">
  <import namespace="http://example.org/ns2">

  <xsd:element name="X"/>
  <xsd:element ref="ns1:X"/>
  <xsd:element ref="ns2:Y"/>
  <xsd2:betterElement name="newone" …/>
</xsd:schema>
```

Enhance the schema language!

A few validation tricks: Modeling content

How to validate this?

```
<soap:Envelope>
      <soap:Body>
            …your message here…
      </soap:Body>
</soap:Envelope>
```

- What is the content model for <soap:Body>?
- Can you validate the contents?

Inside/out vocabularies (some specific SOAP examples)

```
<!– SOAP PURCHASE ORDER -->
<soap:Envelope
  xmlns:soap="http://www.w3.org/2002/06/soap-envelope">
  <soap:Body>
    <po:purchaseOrder xmlns:po=http://example.org/po>
       …
    </po:purchaseOrder>
  </soap:Body>
</soap:Envelope>

<!– SOAP INVOICE -->
<soap:Envelope
  xmlns:soap="http://www.w3.org/2002/06/soap-envelope">
  <soap:Body>
    <inv:invoice xmlns:po=http://example.org/inv>
       …
    </inv:invoice>
  </soap:Body>
</soap:Envelope>
```

How do we write schemas
For these two cases?

Schemas for e

Putting “skip” here says: don’t validate the content of the body

```
<xsd:complexType name=“bodyType”>
      ...
            <xsd:sequence>
               <xsd:any processContents=“skip”/>
            </xsd:sequence>
      ...
</xsd:complexType>
```

Schemas for e

Putting “lax” here says: validate *only* if your schema has declarations for the contents

```
<xsd:complexType name=“bodyType”>
      ...
            <xsd:sequence>
               <xsd:any processContents=“lax”/>
            </xsd:sequence>
      ...
</xsd:complexType>
```

Schemas for e

Putting “strict” here says: you must have declarations and must successfully validate the contents of body.

```
<xsd:complexType name=“bodyType”>
      ...
        <xsd:sequence>
          <xsd:any processContents=“strict”/>
        </xsd:sequence>
      ...
</xsd:complexType>
```

Versioning vocabularies & schemas

- It’s hard!
  - Use namespaces?
  - Do 50 bug fixes give you 50 namespaces?
  - How much interop? Does old schema accept new version?
  - What about Xpath?
- For better or worse: schemas has no organized model for versioning

Inheritance: why have it?

- Allow reuse of definitions
- Model real-world inheritance and polymorphism
- Substitutability
- Mappings to programming systems w/inheritance
- *Schemas provides mechanisms offering parial solutions to these problems*

Refinement vs. Extension

- Data inheritance is different from method inheritance
- No active code: receiver “sees” everything – order matters, e.g. for multiple inheritance
- Innovation(?) in schema:
  - Restriction: subtype is a subset (supports substitutability)
  - Extension: subtype builds on base (supports modular development, some mappings to real world and programming languages.)
  - No multiple inheritance (for now)

Wrapup

Some things I learned

- No such thing as a “simple” feature
- Big committee -> big language
- Documents & data together are cool
  - But neither community gets a simple schema language
- Make realistic schedules – we didn’t make time to pull features

Thank you!