XML Schemas

- **Overcome the (many!) deficiencies of DTDs**
  - W3C states:

    "While XML 1.0 supplies a mechanism, the Document Type Definition (DTD) for declaring constraints on the use of markup, automated processing of XML documents requires more rigorous and comprehensive facilities in this area. Requirements are for constraints on how the component parts of an application fit together, the document structure, attributes, datatyping, and so on. The W3C XML Schema Working Group is addressing means for defining the structure, content and semantics of XML documents."
  - very limited capability to describe the *content* of a document
    - *only concerned with its structure*
    - *weak data typing*
      - pretty much just TEXT
    - *an important issue for a situation such as dumping/restoring an SQL database*
  - odd syntax
    - *effectively a separate language*
      - increases complexity of parsers

W3C Schemas

- **Recommendation as of May 2001**
- **Driving forces**
  - allows creation of new data types
    - *data types exhibit a hierarchy: base (float)/derived (integer)*
  - data types can be applied to attributes *and* elements
  - extensive support for facets
  - O-O modelling support
  - inline definitions of attributes/child elements are possible
  - clear definition of content model
- **Useful reference:**
  - http://www.xml.com/pub/a/2000/11/29/schemas/structuresref.html

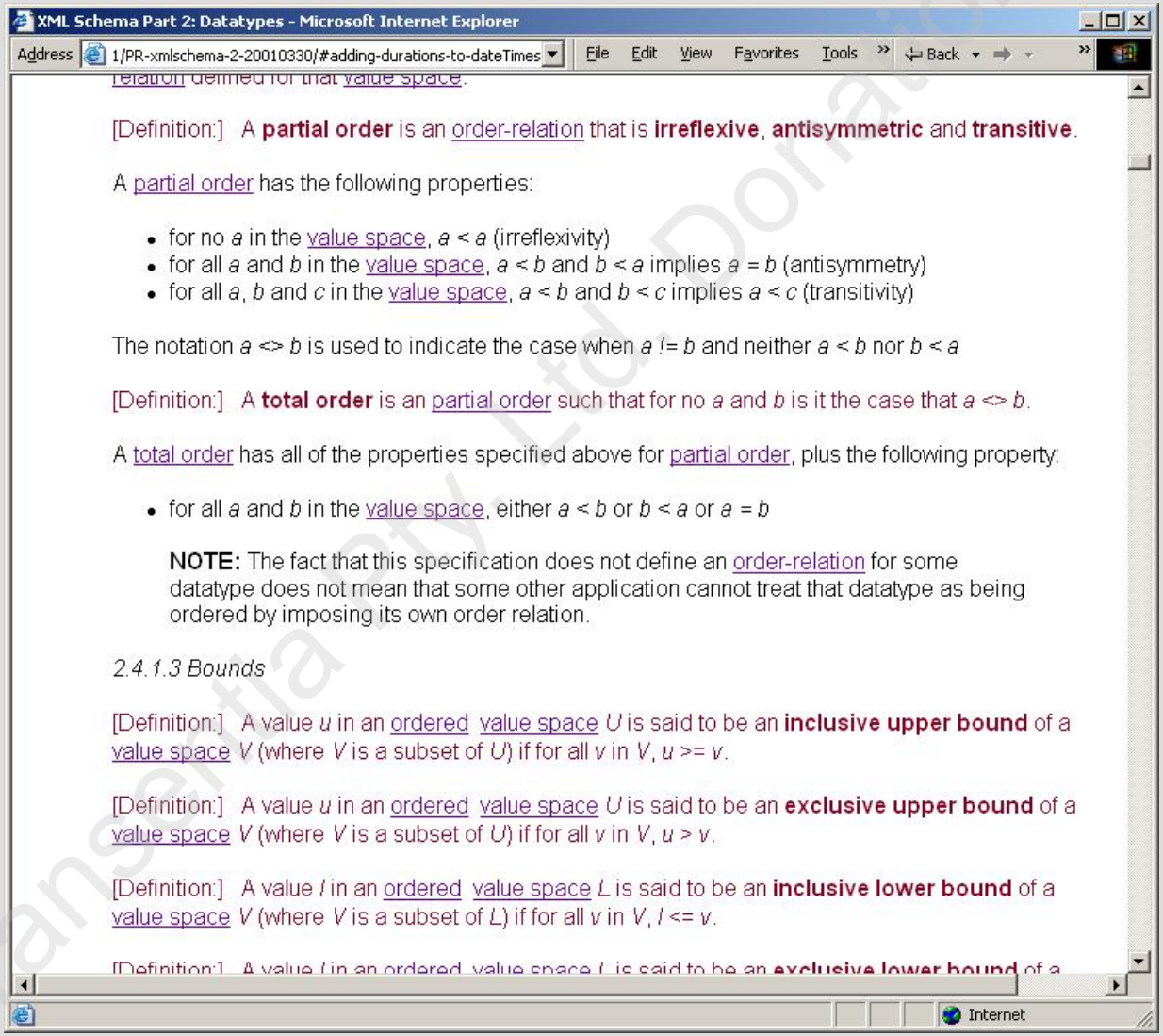
Status

- **Now available**
  - *"The XML DTD isn't dead yet—but it can only be a matter of time before it becomes obsolete..."*
    - *several contenders*
      - DCD, SOX, DDML, XML-Data (Microsoft), RDF (netscape)
    - *IE5+ provided a "preview technology" called XDR*
      - served as the foundation for much of the official standard
  - MSXML4 supports W3C Schemas
    - *'native' in WindowsXP*
- **Complex!**
  - Definition is split over three documents
    - *XML Schema Part 0: Primer*
    - *XML Schema Part 1: Structures*
    - *XML Schema Part 2: Datatypes*
  - Don't even *try* to read these!

"XDR will be relegated to historical interest…the implementation of XDR in IE5 informed the development of the XML Schema specifications, gave many developers early experience with non-DTD schemas for XML, and helped hasten the adoption of XML for non-traditional uses."

Complex Definitions

- **Gives XSD a reputation for being "too academic"**
  - *I wonder why!*



Goals

- **Authors shouldn't need to learn a special syntax**
  - the schema language should be expressed in XML; in comparison, a normal DTD has a special syntax and so requires special treatment
  - also makes life much easier for processing tools
    - *encourages uptake and allows XML to exist at "Internet speed"*
- **Schemas should be extensible**
  - allow for flexible data modelling techniques (such as OOP)
- **Schemas should meet the needs of applications requiring extensive data validation**
  - such as database interchange; need proper data types and constraints on data values, etc.

More Goals

- **A schema must be able to be built up from parts coming from many sources**
  - so that a document can be constructed that incorporates several specifications and so meets many requirements
  - achieved through support for namespaces
- **Schemas should encourage reuse**
  - DTDs can allow reuse but things quickly get messy (lots of parameter entities, etc.); schemas offer a simpler solution
- **Schemas should be upwardly compatible with XML 1.0**
  - minimise "technology churn"
  - standard DTDs can still be used, if an application has simple enough requirements

Example

```
<?xml version="1.0"?>
<MESSIER
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation = "messierschema.xsd">
  <M INDEX="1">
    <CONSTELLATION>Taurus</CONSTELLATION>
    <DESCRIPTION>Diffuse Nebula</DESCRIPTION>
  </M>
  <M INDEX="2">
    <CONSTELLATION>Aqarius</CONSTELLATION>
    <DESCRIPTION>Globular Cluster</DESCRIPTION>
  </M>
  <M INDEX="3">
    <CONSTELLATION>Canes Venatici</CONSTELLATION>
    <DESCRIPTION>Globular Cluster</DESCRIPTION>
  </M>
  …
</MESSIER>
```

```
<?xml version="1.0"?>
<!DOCTYPE MESSIER [
<!ELEMENT MESSIER (M+)>
<!ELEMENT M (CONSTELLATION*, DESCRIPTION)>
<!ATTLIST M
  INDEX CDATA #REQUIRED>
<!ELEMENT CONSTELLATION (#PCDATA)>
<!ELEMENT DESCRIPTION (#PCDATA)>
]>
<MESSIER>
  <M INDEX="1">
    <CONSTELLATION>Taurus</CONSTELLATION>
    <DESCRIPTION>Diffuse Nebula</DESCRIPTION>
  </M>
  <M INDEX="2">
…
```

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3c.org/2001/XMLSchema">
  <element name="MESSIER">
    <complexType>
      <element name="M" minOccurs="0">
        <complexType>
          <attribute name="INDEX" type="nonNegativeInteger"
                     use="required" />
          <sequence>
            <element name="CONSTELLATION" type="string"
                     minOccurs="0" maxOccurs="1"/>
            <element name="DESCRIPTION" type="string"
                     minOccurs="1" maxOccurs="1"/>
          </sequence>
        </complexType>
      </element>
    </complexType>
  </element>
</schema>
```

"Russian Doll" style

DataTypes

- **XSD permits the specification of types**
  - constrain the contents of the document
    - *not just the structure, as with DTDs*
  - exhaustive set built-in type hierarchy
    - *(next slide)*
  - also allows derivations
    - *simple*
    - *complex*

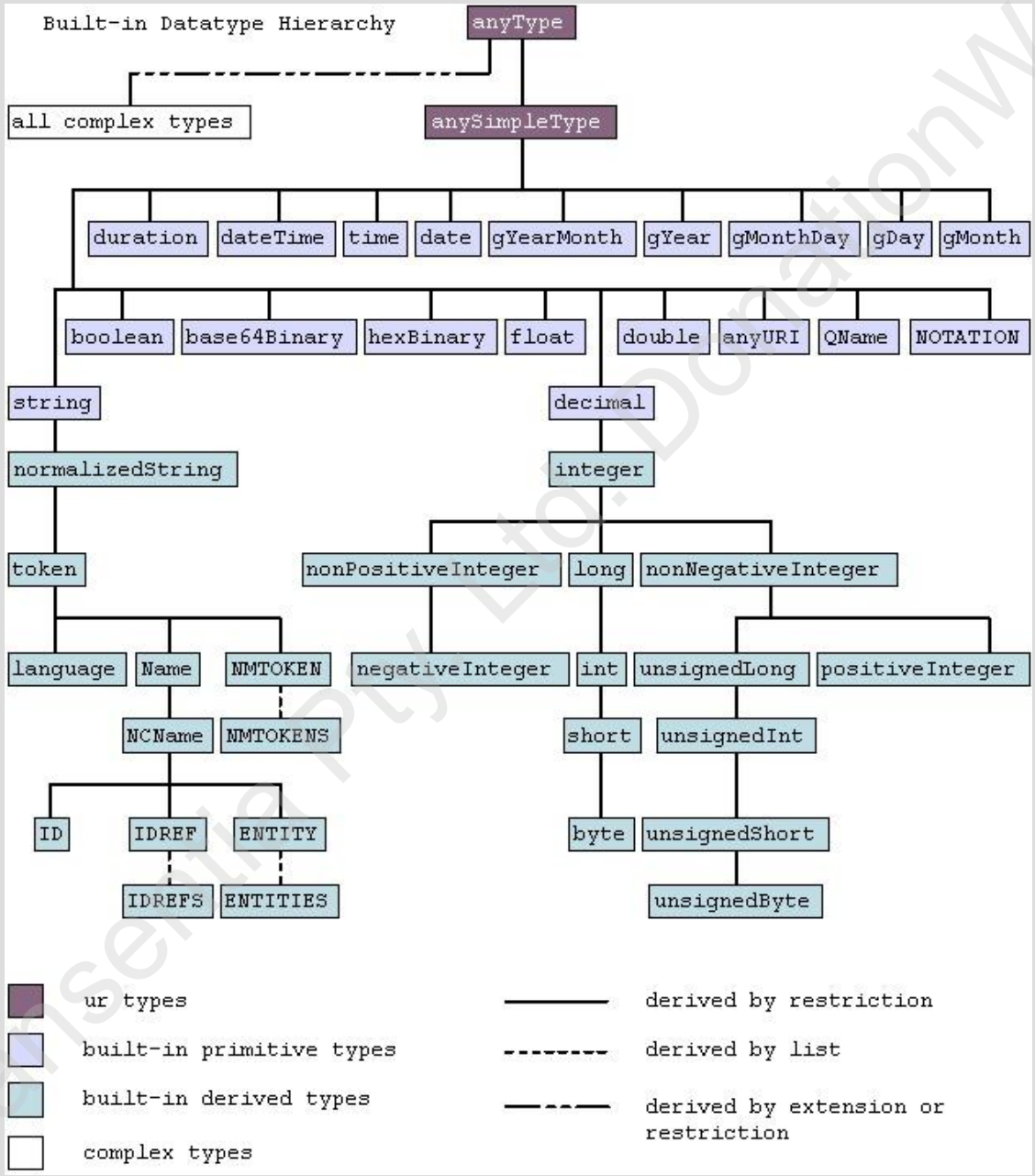
```
<xsd:simpleType name="dayOfWeek">
  <xsd:restriction base="xsd:integer">
    <xsd:minInclusive value="0"/>
    <xsd:maxInclusive value="6"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="listOfDays">
  <xsd:list itemType="dayOfWeek"/>
</xsd:simpleType>

<xsd:complexType name = "significantDOWs">
  <xsd:sequence>
   <xsd:element name = "birthDOW" type="dayOfWeek" />
   <xsd:element name = "marriageDOW" type="listOfDays" />
   <xsd:element name = "deathDOW" type="dayOfWeek" />
  </xsd:sequence>
 </xsd:complexType>
```

Datatype Hierarchy

- From the W3C



W3C-Schema Facets

- **Every data type has a set of *facets* that characterise its properties**
  - string length
  - regular expression pattern constraint
    - *based on PERL 5 regular expressions*

```
<xs:simpleType name="POSTCODE">
  <xs:restriction base="xs:string">
    <xs:pattern value="[0-9]{4}" />
  </xs;restriction>
</xs:simpleType>
```

  - total digits in float before/after decimal point
  - max/min values
    - *e.g. postcode is 4 digits long*
  - whitespace handling
    - *preserve; collapse; replace*

Composition

- **Composition is important, of course**
  - sequence/union/all
  - attributeGroup

```
<xs:simpleType name="isbnType">
   <xs:union>
     <xs:simpleType>
       <xs:restriction base="xs:string">
         <xs:pattern value="[0-9]{4}-[A-Z]{2}-[0-9]{4}"/>
       </xs:restriction>
     </xs:simpleType>
     <xs:simpleType>
       <xs:restriction base="xs:NMTOKEN">
         <xs:enumeration value="NOTYETASSIGNED"/>
         <xs:enumeration value="NONE"/>
       </xs:restriction>
     </xs:simpleType>
   </xs:union>
 </xs:simpleType>

 <xs:group name="mainBookElements">
   <xs:all>
     <xs:element name="title" type="nameType"/>
     <xs:element name="author" type="nameType"/>
   </xs:all>
 </xs:group>

 <xs:attributeGroup name="bookAttributes">
   <xs:attribute name="isbn" type="isbnType" use="required"/>
   <xs:attribute name="available" type="xs:string"/>
 </xs:attributeGroup>

 <xs:complexType name="bookType">
   <xs:sequence>
     <xs:group ref="mainBookElements"/>
     <xs:element name="chapter" type="chapterType"
          minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:attributeGroup ref="bookAttributes"/>
 </xs:complexType>
```

Uniqueness/Keys

- Constraining a document's contents

```
<xs:schema id="MyDataSet" xmlns=""
           xmlns:xsd="http://www.w3.org/2001/XMLSchema">

 <xs:element name="MyDataSet">
  <xs:complexType>
    <xs:choice maxOccurs="unbounded">
      <xs:element name="OrderDetail">
       <xs:complexType>
         <xs:sequence>
           <xs:element name="OrderNo" type="xs:integer" />
           <xs:element name="ItemNo" type="xs:string" />
         </xs:sequence>
       </xs:complexType>
      </xs:element>
      <xs:element name="Order">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="OrderNumber" type="xs:integer" />
            <xs:element name="EmpNumber" type="xs:integer" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:choice>
  </xs:complexType>

  <xs:key name="OrderNumberKey"  >
    <xs:selector xpath=".//Order" />
    <xs:field xpath="OrderNumber" />
  </xs:key>

  <xs:keyref name="OrderNoRef" refer="OrderNumberKey">
    <xs:selector xpath=".//OrderDetail" />
    <xs:field xpath="OrderNo" />
  </xs:keyref>
 </xs:element>
</xs:schema>
```

```
<xsd:unique name="everyDeptRegionDeptNameComboMustBeUnique">
    <xsd:selector xpath="department"/>
    <xsd:field xpath="deptRegion"/>
    <xsd:field xpath="deptName"/>
</xsd:unique>
```

Mixed Content

- character data can appear alongside subelements, and character data is not confined to the deepest subelements
  - W3C example

```
<xsd:element name="letterBody">
 <xsd:complexType mixed="true">
  <xsd:sequence>
   <xsd:element name="salutation">
    <xsd:complexType mixed="true">
     <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
     </xsd:sequence>
    </xsd:complexType>
   </xsd:element>
   <xsd:element name="quantity"    type="xsd:positiveInteger"/>
   <xsd:element name="productName" type="xsd:string"/>
   <xsd:element name="shipDate"    type="xsd:date" minOccurs="0"/>
   <!-- etc. -->
  </xsd:sequence>
 </xsd:complexType>
</xsd:element>
```

```
<letterBody>
<salutation>Dear Mr.<name>Robert Smith</name>.</salutation>
Your order of <quantity>1</quantity> <productName>Baby
Monitor</productName> shipped from our warehouse on
<shipDate>1999-05-21</shipDate>. ....
</letterBody>
```

Reusability

- Various facilities to foster reusability
  - Include/Abstract/Final
  - Substitution Groups

```
<xs:complexType name="characterType" final="#all">
  <xs:sequence>
    <xs:element name="name" type="nameType"/>
    <xs:element name="since" type="sinceType"/>
    <xs:element name="qualification" type="descType"/>
  </xs:sequence>
</xs:complexType>
```

```
<element name="comment" type="string" abstract="true"/>

<element name="shipComment" type="string"
         substitutionGroup="ipo:comment"/>
<element name="customerComment" type="string"
         substitutionGroup="ipo:comment"/>
```

```
<items>
  <item partNum="833-AA">
    <productName>Lapis necklace</productName>
    <quantity>1</quantity>
    <USPrice>99.95</USPrice>
    <ipo:shipComment>
      Use gold wrap if possible
    </ipo:shipComment>
    <ipo:customerComment>
      Want this for the holidays!
    </ipo:customerComment>
    <shipDate>1999-12-05</shipDate>
  </item>
</items>
```

```
<xs:include schemaLocation="extern.xsd"/>

<xs:redefine schemaLocation="extern.xsd">
  <xs:simpleType name="nameType">
    <xs:restriction base="xs:string">
      <xs:maxLength value="40"/>
    </xs:restriction>
  </xs:simpleType>
</xs:redefine>
```

```
<xs:element name="name-elt" type="xs:string" abstract="true"/>
<xs:element name="name" type="xs:string"
            substitutionGroup="name-elt"/>
<xs:element name="surname" type="xs:string"
            substitutionGroup="name-elt"/>
```

Compare

XDR Version

```
<xdr:ElementType name="Email" dt:type="dt:string" />

<xdr:ElementType name="Phone" dt:type="dt:integer" content= "textOnly">
  <xdr:attribute type="phonetype" />
</xdr:ElementType>

<xdr:ElementType name="ContactDetails" content="eltOnly" order="seq">
  <xdr:element type= "Email" minOccurs="0" maxOccurs="*" />
<xdr:ElementType>
```

W3C XML Schema Version

```
<xsd:element name="Email" type="xsd:string" />

<xsd:element name="Phone" type="xsd:integer">
  <xsd:complexType>
    <xsd:attribute name="phonetype" type="PhoneType" />
  </xsd:complexType>
</xsd:element>

<xsd:element name="ContactDetails">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="Email" minOccurs="0" maxOccurs="2" />
      <xsd:element ref="Phone" minOccurs="0" maxOccurs="4" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

XDR-W2C Schema Conversion Tool

- **From Microsoft:**
  - http://msdn.microsoft.com/msdn-files/027/001/539/xdr-xsd-converter.exe
    - *works for simple schemas*
    - *complex ones require human postprocessing*

- **Not everyone is satisfied with W3C schemas**
  - various reasons
    - *too complex*
    - *too slow*
    - *'tainted' by Microsoft*
    - *limitations*
      - can say:
        - *the Demo element should contain a sequence of elements, A followed by B*
        - *the A element contains an integer*
        - *the B element contains an integer*
      - can't express:
        - *the value of A must be greater than the value of B*
  - *much* work being done to supplement/replace schemas:
    - *RELAX, DSD, SOX, TREX, Schematron, etc.*

Schematron/Relax

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.demo.org"
        xmlns="http://www.demo.org"
        xmlns:sch="http://www.ascc.net/xml/schematron"
        elementFormDefault="qualified">
    <xsd:annotation>
        <xsd:appinfo>
            <sch:title>Schematron validation</sch:title>
            <sch:ns prefix="d" uri="http://www.demo.org"/>
        </xsd:appinfo>
    </xsd:annotation>
    <xsd:element name="Demo">
        <xsd:annotation>
            <xsd:appinfo>
                <sch:pattern name="Check A greater than B">
                    <sch:rule context="d:Demo">
                        <sch:assert test="d:A > d:B"
                                diagnostics="lessThan">
                                A should be greater than B.
                        </sch:assert>
                    </sch:rule>
                </sch:pattern>
                <sch:diagnostics>
                    <sch:diagnostic id="lessThan">
                        Error! A is less than B
                        A = <sch:value-of select="d:A"/>
                        B = <sch:value-of select="d:B"/>
                    </sch:diagnostic>
                </sch:diagnostics>
            </xsd:appinfo>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="A" type="xsd:integer"/>
                <xsd:element name="B" type="xsd:integer"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

```
<module
  moduleVersion='1.2'
  relaxCoreVersion='1.0'
  targetNamespace='http://www.wrox.com/examples/ProXML2'
  xmlns='http://www.xml.gr.jp/xmlns/relaxCore'>

  <interface>
    <export label='MailingAddress' />
  </interface>

  <elementRule role='Street' type='string' />
  <elementRule role='City' type='string' />
  <elementRule role='State' type='string' />
  <elementRule role='Zip' type='string' />
  <elementRule role='Country' type='string' />

  <tag name= 'Street' />
  <tag name='City' />
  <tag name='State' />
  <tag name='Zip' />
  <tag name='Country' />

  <elementRule role='MailingAddress' >
    <sequence>
      <ref label='Street' occurs='*' />
      <ref label= 'City' />
      <ref label= 'State' />
      <ref label= 'zip' />
      <ref label= 'Country' />
    </sequence>
  </elementRule>

  <tag name='MailingAddress' />
</module>
```

Problems

- **Why we haven't seen the ultimate schema language yet…**
  - it is generally too complicated: the spec is several hundred incomprehensible pages in a very technical language, so is hard to use
  - Practical limitations of expressibility:
    - *when describing mixed content, the character data cannot be constrained in any way (not even a set of valid characters can be specified)*
    - *content and attribute declarations cannot depend on attributes or element context*
    - *it is not 100% self-describing, even though that was an initial design requirement*
    - *defaults cannot be specified separate from the declarations (makes it hard to make families of schemas that only differ in the default values)*
    - *element defaults can only be character data (not containing markup)*
  - Technical problems:
    - *does not follow the namespace spirit*
    - *the notion of "type" adds an extra layer of confusing complexity:*
      - in instance documents, we have "elements" which have "element names"
      - in schemas, elements are described by "element definitions" which associate "element names" with "type names",
      - type definitions associate "type names" with "element descriptions" which describe the elements in the instance documents
      - to cause extra confusion, the XML 1.0 spec uses the term "element type" for the name of an element
    - *xsi:type attributes are required in instance documents when derived types are being used in place of base types (then one might as well have defined a new element and used a substitution group)*
    - *substitution groups and local declarations (with non-unique names) make it difficult to look up the description of a given element*
  - Non-minimalistic design:
    - *substitution groups and type derivation seem to be different attempts to solve the same problems*
    - *incorporation of XPath to express uniqueness and keys (neither uniqueness or keys are fundamental concepts for schemas, so dragging in a big language as XPath is overkill)*
    - *the set of built-in data types is far from minimalistic*
    - *the use of Perl-style regular expressions violates the principle of using XML syntax to describe XML syntax*

Future

- **Rosy**
  - ?
  - *Tim Bray has put forward a document describing "Extensible Markup Language – SW" (SkunkWorks)*
    - drops compatibility with SGML and refactors the standards
  - *"It seems inevitable that the W3C will eventually offer a standards document which it calls XML 2.0. … The only interesting questions which remain unanswered are what XML 2.0 will look like and how politically nasty the process that creates it will be.…Whatever XML 2.0 eventually becomes technically, the process that creates it will be more social and political than anything else, and it's that process which seems perilous and fragile at best."*
    - http://www.xml.com/pub/a/2002/02/20/deviant.html

- **Castor/JAXB**
  - "Lets you seamlessly program with normal Java objects that represent your XML documents. This is much more luxurious than writing that grueling [sic] XML parsing code."
  - Generate Java code from an XML Schema
  - More at:
    - *http://java.sun.com/xml/jaxb*
    - *http://www.castor.org/*