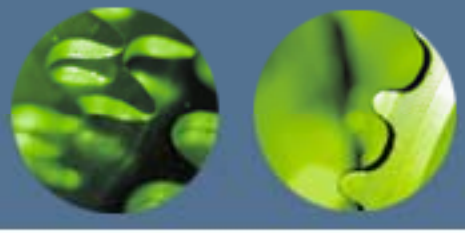
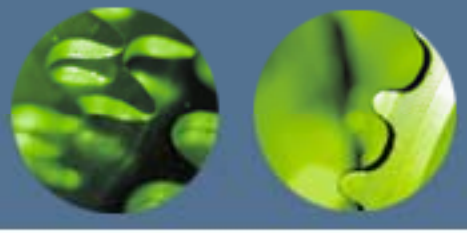
XML Schema 1.1
The Good Parts

Priscilla Walmsley
pwalmsley@datypic.com
http://www.datypic.com



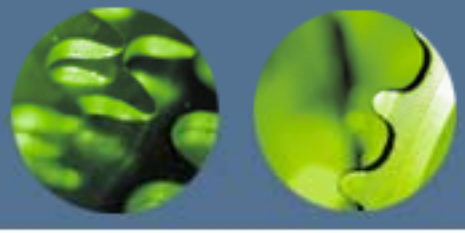
XML Schema 1.1

- Became a W3C Rec on April 5, 2012
- Processor support:
  - Yes: Xerces, Saxon, oXygen
  - Not yet: XMLSpy
- Namespace is the same:
  - http://www.w3.org/2001/XMLSchema
- Pretty much backward compatible



Assertions

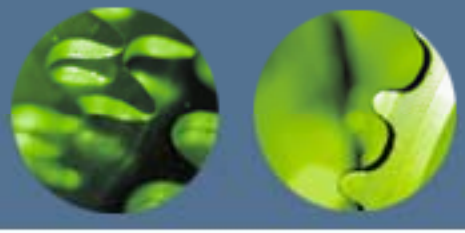
- Additional constraints on types
  - Both complex and simple types
- Expressed in XPath 2.0
- Only allows access to attributes and content
  - (not parents, root element, other documents, etc.)



Assertion on a Complex Type

- Uses the **`assert`** element

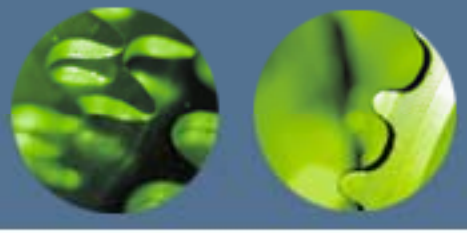
```
<xs:complexType name="ProductType">
  <xs:sequence>
    <xs:element name="number" type="xs:integer"/>
    <xs:element name="name" type="xs:string"/>
    <xs:element name="size" type="xs:integer"/>
  </xs:sequence>
  <xs:attribute name="dept" type="xs:string"/>
  <xs:assert test="(@dept = 'ACC' and number > 500) or
                   (number &lt; 300)"/>
  <xs:assert test="if (@dept = 'ACC')
                     then not(size)
                     else true()"/>
</xs:complexType>
```



Assertion on a Simple Type

- Uses the **`assertion`** element
- **`$value`** variable (not ".") used to represent the simple content value

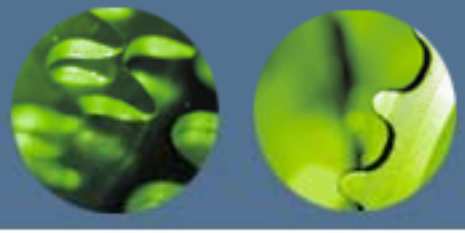
```
<xs:simpleType name="DepartmentCodeType">
  <xs:restriction base="xs:token">
    <xs:assertion test="not(contains($value,'XYZ'))"/>
    <xs:assertion test="substring($value,2,2) != '00'"/>
    <xs:length value="3"/>
  </xs:restriction>
</xs:simpleType>
```



Conditional Type Assignment

- Assigns a type for an element based on the values of its attributes
- Use an **`alternative`** element
  - can specify a default, error conditions

```
<xs:element name="product" type="ProductType">
  <xs:alternative test="@dept='ACC'"
                  type="AccessoryType"/>
  <xs:alternative test="@dept='WMN' or @dept='MEN'"
                  type="ClothingType"/>
  <xs:alternative test="@dept='N/A'"
                  type="xs:error"/>
</xs:element>
```

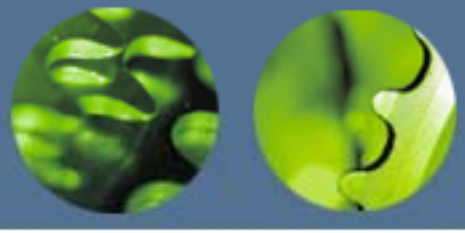


Open Content

- Allows wildcard to apply anywhere in a type

```
<xs:element name="product" type="OpenProductType"/>
<xs:complexType name="OpenProductType">
  <xs:openContent>
    <xs:any namespace="##other" processContents="lax"/>
  </xs:openContent>
  <xs:sequence>
    <xs:element name="number" type="xs:integer"/>
    <xs:element name="name" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

```
<product xmlns:ext="http://datypic.com/extension">
  <ext:el1>foo</ext:el1>
  <number>557</number>
  <ext:el2>bar</ext:el2>
  <name>Short-Sleeved Linen Blouse</name>
  <ext:el1>foo</ext:el1>
</product>
```

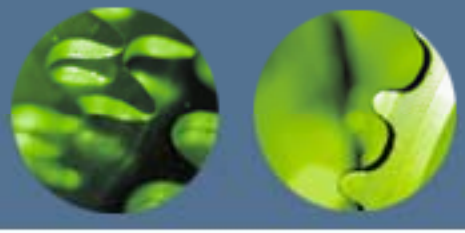


Less Constrained **all** Groups

- In 1.1, **all** groups can:
  - have element declarations where maxOccurs > 1
  - contain wildcards
  - contain group references (to groups that also use **all**)
  - be extended (by other types that also use **all**)

```
<xs:complexType name="ProductType">
  <xs:all>
    <xs:element name="number" type="xs:integer"/>
    <xs:element name="color" type="xs:string"
                             maxOccurs="unbounded"/>
    <xs:any namespace="##other" processContents="lax"/>
  </xs:all>
</xs:complexType>
```

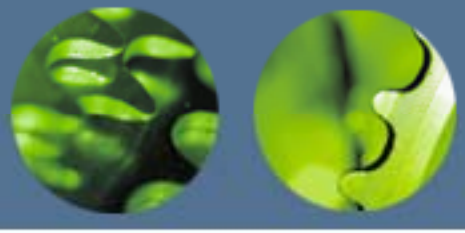
  - still can't be combined with **sequence** and **choice**



New Wildcard Attributes

- Two new attributes for wildcards (both element and attribute wildcards)
- **`notNamespace`**
  - list of namespace that are *not* allowed for replacement elements
- **`notQName`**
  - list of names of elements that are not allowed for replacement elements, with two additional keywords:
    - **`##defined`** - any name that is defined in the schema
    - **`##definedSibling`** - any name that defined that could be a sibling

```
<xs:any minOccurs="0" maxOccurs="unbounded"
                      notNamespace="http://www.w3.org/1999/xhtml"
                      notQName="##definedSibling desc size"
                      processContents="lax"/>
```

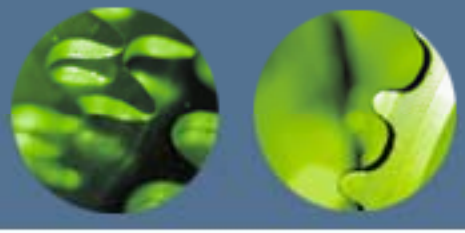


Relaxed UPA for Wildcards

- No longer a UPA violation to follow an optional element with a wildcard that could also apply.

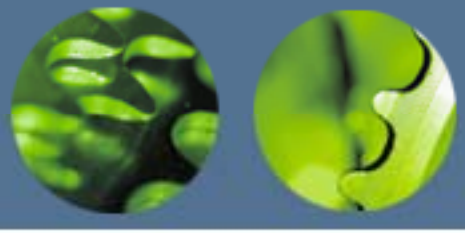
```
<xs:complexType name="ProductType">
  <xs:sequence>
    <xs:element name="number" type="xs:integer"/>
    <xs:element name="color" type="xs:string"
                             minOccurs="0"/>
    <xs:any namespace="##any" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

- Makes forward compatibility easier to achieve



Simplified Restrictions

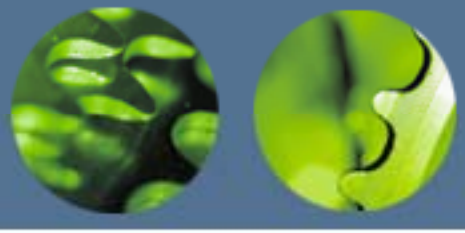
- No more complex mapping rules for what's allowed in a restriction
  - General statement that the restriction must only allow a subset
- Ability to restrict types in another namespace
  - Use **`targetNamespace`** on local element/attribute declarations
- Ability to "ref" identity constraints
  - Allows reuse of **`unique/key/keyref`**
  - Makes restriction easier because specifies a clear subset



Multiple Substitution Groups

- An element can be in more than one substitution group
  - use space-separated list of names in **`substitutionGroup`**

```
<xs:element name="product" type="ProductType"/>
<xs:element name="discontinuedProduct" type="ProductType"/>
<xs:element name="hat" type="HatType"
            substitutionGroup="product"/>
<xs:element name="umbrella" type="UmbrellaType"
            substitutionGroup="product discontinuedProduct"/>
```

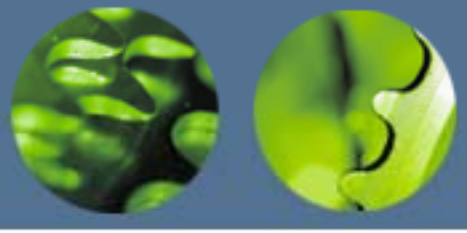


Default Attributes

- Specify a particular attribute group as the default
  - it applies to all complex types
  - unless you turn it off using **`defaultAttributesApply="false"`** on the type

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           defaultAttributes="IdentifierGroup">

  <xs:attributeGroup name="IdentifierGroup">
    <xs:attribute name="id" type="xs:ID" use="required"/>
    <xs:attribute name="version" type="xs:decimal"/>
  </xs:attributeGroup>

  <xs:complexType name="ProductType">
      <!--.....-->
  </xs:complexType>
</xs:schema>
```

```
<product id="A123" version="12.0">
  ...
</product>
```



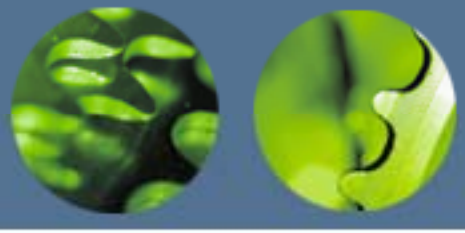
Inheritable Attributes

- Can specify that attribute values are inheritable by descendants

```
<xs:element name="work" type="WorkType"/>
<xs:complexType name="WorkType">
  <xs:sequence>
    <xs:element name="title" type="TitleType" maxOccurs="3"/>
  </xs:sequence>
  <xs:attribute name="language" type="xs:language"
                inheritable="true"/>
</xs:complexType>
<!-- TitleType still needs to declare language attribute -->
```

```
<work language="en">
  <title>This is not a pipe.</title>
  <title language="fr">Ceci n'est pas une pipe.</title>
</work>
```

work/title[1]/@language='en'



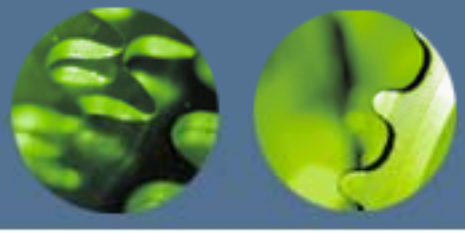
Overrides

- **`redefine`** is deprecated, and replaced by **`override`**
- Unlike **`redefine`**, **`override`**:
  - does not require/allow reference to the original component
  - can apply to element and attribute declarations

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="size" type="xs:string"/>
</xs:schema>
```

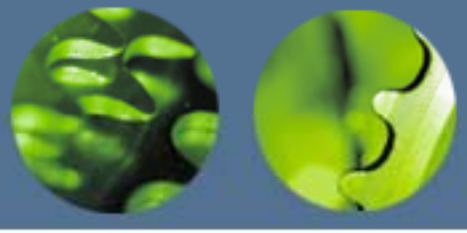


```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:override schemaLocation="schema1.xsd">
    <xs:element name="size" type="xs:integer"/>
  </xs:override>
</xs:schema>
```



New Types and Facets

- New types:
  - **`yearMonthDuration`** and **`dayTimeDuration`**
    - ordered subtypes of duration
  - **`dateTimeStamp`**
    - restriction of dateTime that requires a time zone
  - **`error`**
    - really only used in conditional type assignment
- New facets:
  - **`explicitTimezone`**
    - whether a time zone is required
  - **`assertion`**

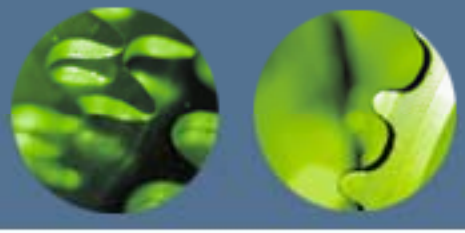


Implementation-Defined Types and Facets

- Saxon has an implementation-defined facet:
  - **`preprocess`** allows XPath processing on value before validation

```
<xs:simpleType name="SMLXSizeType">
   <xs:restriction base="xs:token">
      <saxon:preprocess action="upper-case($value)"/>
      <xs:enumeration value="SMALL"/>
      <xs:enumeration value="MEDIUM"/>
      <xs:enumeration value="LARGE"/>
   </xs:restriction>
</xs:simpleType>
```

- Can have fallback definition if type/facet is unavailable

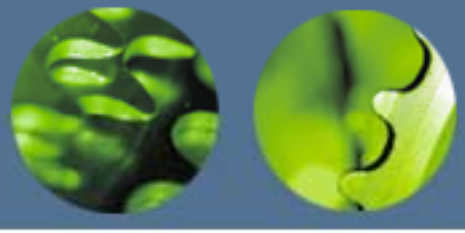


Version Control

- Version control attributes allow you to say what to do if a processor doesn't support a version of XML Schema.

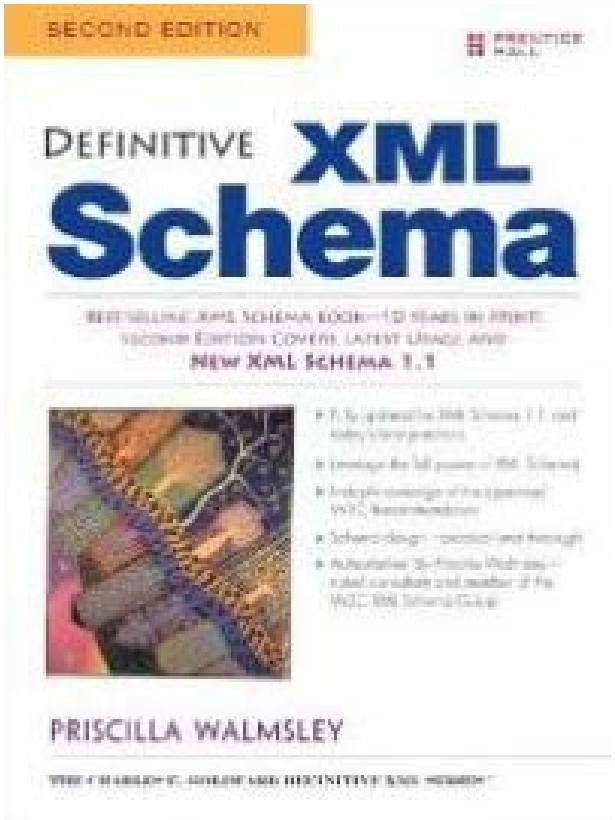
```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:vc="http://www.w3.org/2007/XMLSchema-versioning">

  <xs:element name="product" vc:minVersion="1.3">
    <!-- a declaration that uses XML Schema 1.3 constructs -->
  </xs:element>

  <xs:element name="product" vc:minVersion="1.1"
                             vc:maxVersion="1.3">
    <!-- a declaration conformant to versions 1.1 and 1.2 -->
  </xs:element>
</xs:schema>
```

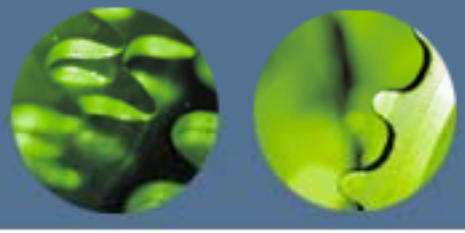
- Really only useful for future versions



Shameless Plug

- *Definitive XML Schema*, 2nd edition came out last week
- Covers 1.1, but still useful for 1.0
  - changes are marked and explained
- Includes what I've learned over the last 10 years about designing schemas





The End

- Please use the next 15 seconds for quiet reflection on the exquisite beauty of XML Schema 1.1.