XML Schema for unique values using key and keyref

Sometimes simple data requirements take too much time, much more than expectations. I have spent more than 2 days to sort out tightly coupled XML Schema for Xia2 and validating them with XMLBeans although validation and binding can also be done through JAXB. Here are the three simple requirements of Xia2.

1. Consider this simple XML fragment from Xia 2. (Here prefix "xsd" is attached with URL http://xia2.e-htpx.ac.uk/xsd rather than http://www.w3.org/2001/XMLSchema).

   ```
   <xsd:wavelength name="wavelength2">
     <xsd:WAVELENGTH>1.1345</xsd:WAVELENGTH>
     <xsd:F1>1.2345</xsd:F1>
     <xsd:F11>1.3345</xsd:F11>
   </xsd:wavelength>
   ```

   In the above XML fragment all three elements are optional but if there is F1 (it should be F' but F' is not in accordance with XML naming restrictions) then there should be F11 and vice versa. Thus following XML fragments are valid:

   - ```
     <xsd:wavelength name="wavelength2">
             <xsd:WAVELENGTH>1.1345</xsd:WAVELENGTH>
     </xsd:wavelength>
     ```
     (F1 and F11 both are missing)

   - ```
     <xsd:wavelength name="wavelength2">
              <xsd:F1>1.2345</xsd:F1>
             <xsd:F11>1.3345</xsd:F11>
     </xsd:wavelength>
     ```
     (F1 and F11 both are present; optional WAVELENGTH is missing)

   But the following XML possible XML fragments are not valid:

   - ```
     <xsd:wavelength name="wavelength2">
              <xsd:WAVELENGTH>1.1345</xsd:WAVELENGTH>
             <xsd:F1>1.2345</xsd:F1>
     </xsd:wavelength>
     ```
     (WAVELENGTH and F1 are present but if F1 is present then F11 should be there)

   - ```
     <xsd:wavelength name="wavelength2">
             <xsd:WAVELENGTH>1.1345</xsd:WAVELENGTH>
             <xsd:F11>1.3345</xsd:F11>
     </xsd:wavelength>
     ```
     (WAVELENGTH and F11 are present but if F11 is present then F1 should be there)

   - ```
     <xsd:wavelength name="wavelength2">
             <xsd:F11>1.3345</xsd:F11>
     </xsd:wavelength>
     ```
     (Only F11 is present but if F11 is present then F1 should be there).

   Simple XML Schema like the one shown below will not work:

   ```
   <xs:complexType name="wavelengthType">
           <xs:sequence>
                   <xs:element name="WAVELENGTH" type="xs:float" minOccurs="0"/>
                   <xs:element name="F1" type="xs:float" nillable="false" minOccurs="0"/>
   ```
   ```

```
            <xs:element name="F11" type="xs:float" nillable="false" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="required"/>
</xs:complexType>
```

Solution:

- The easy solution was to define complexType with following structure:

```
<xs:complexType name="frequencyType">
        <xs:all>
            <xs:element name="F1" type="xs:float"/>
            <xs:element name="F11" type="xs:float"/>
        </xs:all>
</xs:complexType>
```

<xs:all> means both inner elements should be present if < frequencyType > available and in <waveLengthType> we can have something similar to the following structure:

```
<xs:complexType name="wavelengthType">
        <xs:sequence>
            <xs:element name="WAVELENGTH" type="xs:float" minOccurs="0"/>
            <xs:element name="frequencey" type=" frequencyType " minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="required"/>
</xs:complexType>
```

This solution will work but the only problem is the extra element <xsd: frequencey > generated in XML instance.

```
<xsd:wavelength name="wavelength2">
   <xsd:WAVELENGTH>1.1345</xsd:WAVELENGTH>
   <xsd: frequencey >
        <xsd:F1>1.2345</xsd:F1>
        <xsd:F11>1.3345</xsd:F11>
   <xsd: frequencey >
</xsd:wavelength>
```

- The most elegant solution can be achieved through nested <sequence>. The trick here is the use *minOccurs="0"* for the nested <sequence>.

```
<xs:complexType name="wavelengthType">
        <xs:sequence>
            <xs:element name="WAVELENGTH" type="xs:float" minOccurs="0"/>
            <xs:sequence minOccurs="0">
                <xs:element name="F1" type="xs:float" nillable="false"/>
                <xs:element name="F11" type="xs:float" nillable="false"/>
            </xs:sequence>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="required"/>
</xs:complexType>
```

2. The <wavelength> element discussed earlier is immediate child element of root element <project> and single <project> can have multiple <wavelength> elements but each of them should have unique value for attribute “name”. Uniqueness of any element and attribute can be enforced in XML Schema by using <xs:unique>.

```
<xs:unique name="wavelengthName">
        <xs:selector xpath="wavelength "/>
        <xs:field xpath="@name"/>
</xs:unique >
```

In the above code snippet “name” attribute is required. The **selector** element contains an XML Path Language (XPath) expression specifying the set of elements across which the values specified by the **field** elements must be unique. In other words the XPath expression which leads to required elements/nodes. There must be one and only one **selector** element.
Each **field** element contains an XPath expression specifying the values (*attribute or element values*) that must be unique for the set of elements specified by the **selector** element. If there is more than one **field** element, the combination of the **field** elements must be unique. In this case, the values for a single **field** element may or may not be unique across the selected elements, but the combination of all the fields must be unique. There must be one or more **field** element.

<xs:unique> element or attribute specified by combination of selector and filed XPath should be unique but it can be nil also. Alternative to this is the use of <xs:key> which guarantees uniqueness and nil value is not acceptable value. One main advantage of using <xs:key> is that it can be re-used in other elements and in most cases is the best solution.

```
<xs:key name="waveLengthName">
        <xs:selector xpath="wavelength"/>
        <xs:field xpath="@name"/>
</xs:key>
```

3. The root element <project> also has <sweep> element. The cardinality of sweep element is 0-N. One of the child elements of the <sweep> refers to the “name” attribute of <wavelength>. The value of that child element should be valid and must also matches one of the unique “name” attribute of <wavelength>. Below is the structure of <sweep>:

```
<xs:complexType name="sweepType">
  <xs:sequence>
    <xs:element name="WAVELENGTH">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:string">
            <xs:attribute name="name" type="xs:string" use="required"/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
    <xs:element name="BEAM_X" type="xs:int"/>
    <xs:element name="BEAM_Y" type="xs:int"/>
    <xs:element name="IMAGE" type="xs:anyURI"/>
    <xs:element name="DIRECTORY" type="xs:anyURI"/>
    <xs:element name="EPOCH" type="xs:int"/>
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="required"/>
</xs:complexType>
```

The precise explanation of the problem with respect to Xia2 schema is that value of “name” attribute of the < *WAVELENGTH*> in the <sweep> should match the unique “name” attribute of the <wavelength>. We have already imposed uniqueness restriction on the “name” attribute of the <wavelength> using <xs:key>. We can use the < xs:keyref> to couple the value of “name” attribute of < *WAVELENGTH*>

(child element of <sweep>) with the unique “name” attribute of the <wavelength>. Below is the XML Schema where “refer” is the reference to existing <xs:key> element:

```
<xs:keyref name="waveLengthRefConstraint" refer="waveLengthName">
        <xs:selector xpath="sweep/WAVELENGTH"/>
        <xs:field xpath="@name"/>
</xs:keyref>
```

Validation of XML Schema using key and keyref with XMLBeans

The last part of the puzzle is where to put the information related unique “key” and “keyref”. The complexType can’t have sub element of type either <key> or <keyRef>. One possibility is to impose restriction for uniqueness in elements of our custom complexType.

```
<xs:element name="wavelength" type="wavelengthType" nillable="true">
        <xs:key name="waveLengthName">
                <xs:selector xpath=" wavelength"/>
                <xs:field xpath="@name"/>
        </xs:key>
</xs:element>

<xs:element name="sweep" type="sweepType" nillable="true">
        <xs:key name="sweepName">
                <xs:selector xpath=" sweep"/>
                <xs:field xpath="@name"/>
        </xs:key>
        <xs:keyref name="waveLengthRefConstraint" refer="waveLengthName">
                <xs:selector xpath="sweep/WAVELENGTH"/>
                <xs:field xpath="@name"/>
        </xs:keyref>
</xs:element>
```

Semantically and technically the above XML schema is valid but it will not meet our requirements. The reason why the above snippet will not work is due to logical error. In the above XML Schema each <wavelength> element of type "wavelengthType" is separate entity and restriction will be imposed on them in isolation. In other words two <wavelength> in the <project> with identical value for name attribute is not a conflict; same applies to <sweep>.

Our problem was to achieve unique <wavelength> elements within the <project> and each <sweep> referencing any existing <wavelength> within the <project>, thus these restrictions should be defined in the <project> element. Below is the code which is both technically and logically correct.

```
<xs:element name="project">
        <xs:complexType>
                <xs:sequence>
                        <xs:element ref="heavyAtomInformation"/>
                        <xs:element ref="wavelength" maxOccurs="unbounded"/>
                        <xs:element ref=" sweep" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>

        <xs:key name="waveLengthName">
                <xs:selector xpath="wavelength"/>
                <xs:field xpath="@name"/>
```

```
            </xs:key>

            <xs:key name="sweepName">
                  <xs:selector xpath="sweep"/>
                  <xs:field xpath="@name"/>
            </xs:key>

            <xs:keyref name="waveLengthRefConstraint" refer=" waveLengthName">
                  <xs:selector xpath="sweep/WAVELENGTH"/>
                  <xs:field xpath="@name"/>
            </xs:keyref>
      </xs:element>
```

The XML Schema shown above can be compiled using XMLBeans but it doesn't impose the restrictions mentioned in the schema. To achieve the desired result with XMLBeans, one has to use qualified name within the schema. Qualified name means use of Namespace. Below is the complete working XML Schema can be compiled with XMLBeans and which maintains the uniqueness and dependency of elements. Please note the prefix "xsd" which refers to http://xia2.e-htpx.ac.uk/xsd.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2007 sp2 (http://www.altova.com) by Asif Akram (Imperial College) -->
<xs:schema xmlns:xia2="http://xia2.e-htpx.ac.uk/xsd"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://xia2.e-htpx.ac.uk/xsd"
      elementFormDefault="qualified" attributeFormDefault="unqualified">

      <xs:complexType name="heavyAtomInformationType">
            <xs:sequence>
                  <xs:element name="ATOM" type="xs:string"/>
                  <xs:choice>
                        <xs:element name="NUMBER_PER_MOMOMER" type="xs:int"/>
                        <xs:element name="NUMBER_TOTAL" type="xs:int"/>
                  </xs:choice>
            </xs:sequence>
            <xs:attribute name="name" type="xs:string" use="required"/>
      </xs:complexType>
      <xs:complexType name="wavelengthType">
            <xs:sequence>
                  <xs:element name="WAVELENGTH" type="xs:float" minOccurs="0"/>
                  <xs:sequence minOccurs="0">
                        <xs:element name="F1" type="xs:float" nillable="false"/>
                        <xs:element name="F11" type="xs:float" nillable="false"/>
                  </xs:sequence>
            </xs:sequence>
            <xs:attribute name="name" type="xs:string" use="required"/>
      </xs:complexType>
      <xs:complexType name="sweepType">
        <xs:sequence>
          <xs:element name="WAVELENGTH">
            <xs:complexType>
              <xs:simpleContent>
                <xs:extension base="xs:string">
                  <xs:attribute name="name" type="xs:string" use="required"/>
                </xs:extension>
              </xs:simpleContent>
```

```
            </xs:complexType>
        </xs:element>
        <xs:element name="BEAM_X" type="xs:int"/>
        <xs:element name="BEAM_Y" type="xs:int"/>
        <xs:element name="IMAGE" type="xs:anyURI"/>
        <xs:element name="DIRECTORY" type="xs:anyURI"/>
        <xs:element name="EPOCH" type="xs:int"/>
      </xs:sequence>
      <xs:attribute name="name" type="xs:string" use="required"/>
    </xs:complexType>

    <!-- Elements -->
    <xs:element name="heavyAtomInformation" type="xia2:heavyAtomInformationType" nillable="true"/>
    <xs:element name="wavelength" type="xia2:wavelengthType" nillable="true"/>
    <xs:element name="sweep" type="xia2:sweepType" nillable="true"/>
    <xs:element name="project">
        <xs:annotation>
            <xs:documentation>Description of Xia2 project</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="xia2:heavyAtomInformation"/>
                <xs:element ref="xia2:wavelength" maxOccurs="unbounded"/>
                <xs:element ref="xia2:sweep" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
        <xs:key name="waveLengthName">
            <xs:selector xpath="xia2:wavelength"/>
            <xs:field xpath="@name"/>
        </xs:key>
        <xs:key name="sweepName">
            <xs:selector xpath="xia2:sweep"/>
            <xs:field xpath="@name"/>
        </xs:key>
        <xs:keyref name="waveLengthRefConstraint" refer="xia2:waveLengthName">
            <xs:selector xpath="xia2:sweep/xia2:WAVELENGTH"/>
            <xs:field xpath="@name"/>
        </xs:keyref>
    </xs:element>
</xs:schema>
```

Various validation errors generated by XMLBeans in case of non-valid of XML instance:

When two different <wavelength> elements have same value for the “name” attribute.
>> error: cvc-identity-constraint.4.2.2: Duplicate key 'wavelength1' for key constraint 'waveLengthName@http://xia2.e-htpx.ac.uk/xsd'

When <WAVELENGTH> of <sweep> has value not found for “name” attribute of <wavelength>
>> error: cvc-identity-constraint.4.3: Key 'wavelength3' not found for keyref constraint 'waveLengthRefConstraint@http://xia2.e-htpx.ac.uk/xsd'