VAMDC Workshop – Vienna 16 – 18 February 2011

Summary of proposed changes to the XSAMS Schema

1. Refer to "sources" by element rather than attribute.
   e.g.

```
<SomeData>
     <Value>0.</Value>
     <SourceRef>Bref1</SourceRef>
     <SourceRef>Bref2</SourceRef>
     …
</SomeData>
```

   instead of

```
<SomeData sourceRef="Bref1">
     <Value>0.</Value>
</SomeData>
```

   This apparently makes it easier to refer to more than one source than an `xs:IDREFS` attribute type would and is easier for users to loop over.

2. Xsams.xsd in the vamdc-working subversion branch now has a targetNamespace:
   http://xsams.svn.sourceforge.net/viewvc/xsams/branches/vamdc-working
   To make this work and play nicely with unqualified attributes, all the attribute definitions have to be converted into ordinary simpleTypes. e.g.

```
<xs:simpleType name="methodRefType">
     <xs:restriction base="xs:IDREF">
          <xs:pattern value="M.+"/>
     </xs:restriction>
</xs:simpleType>
```

   instead of

```
<xs:attribute name="methodRef">
     <xs:simpleType>
          <xs:restriction base="xs:IDREF">
               <xs:pattern value="M.+"/>
          </xs:restriction>
     </xs:simpleType>
</xs:attribute>
```

3. `DataType` is now a choice between a `<Value>`, `<Accuracy>` pair or `<FitParameters>`, in order that an item of data which is to be calculated from a given function may be specified. For example, a particular quantity, *D*, might be calculated from a function

$$D = F(x, y; a, b, c) = ax^2 + bxy + cy^2,$$

in which *x, y* are the function *arguments* and *a, b, c* function *parameters*. The function itself may be defined within the existing `<Function>` element; the `DataType` definition of *D* then takes the form:

```
<D>
     <FitParameters functionRef="Ffunc">
          <FitArgument units="…">
               <Name>x</Name>
               <LowerLimit units="…">…</LowerLimit>
               <UpperLimit units="…">…</UpperLimit>
          </FitArgument>
          <FitArgument units="…">
               <Name>y</Name>
               <LowerLimit units="…">…</LowerLimit>
               <UpperLimit units="…">…</UpperLimit>
          </FitArgument>
          <FitParameter name="a">
               <SourceRef>…</SourceRef>
               <Value units="…">…</Value>
               <Accuracy>…</Accuracy>
          </FitParameter>
          <FitParameter name="b">
               <SourceRef>…</SourceRef>
               <Value units="…">…</Value>
               <Accuracy>…</Accuracy>
          </FitParameter>
          <FitParameter name="c">
               <SourceRef>…</SourceRef>
               <Value units="…">…</Value>
               <Accuracy>…</Accuracy>
          </FitParameter>
     </FitParameters>
</D>
```

4. ~~A `<NamedDataType>` has been introduced, which extends `<DataType>` by adding an attribute, `name`.~~
   `PrimaryType` now contains an element, `<Name>`, in addition to `<Comments>` which contains a string identifying the data point or set of data.

5. The `<States>` tag has been changed to `<Species>`. And there was much rejoicing.

6. `AtomicIonType` and `MoleculeType` now carry a mandatory attribute, `speciesID`, matching `/X.+/` which can be referenced elsewhere in the XSAMS document with the `speciesRef` attribute.

7. `MolecularChemicalSpecies` now contains the optional element `InChI` and mandatory element `InChIKey` for identification.

8. `MolecularChemicalSpecies` now contains an element `MoleculeStructure` containing the CML elements `cml:atomArray` and `cml:bondArray` identifying the molecular structure (connectivity and/or equilibrium atom *(x,y,z)* coordinates) and labelling the atoms for reference when giving e.g. hyperfine coupling quantum numbers. The CML elements reside in their own namespace (http://www.xml-cml.org/schema) which should be declared if used.

9. Two more `cases' have been introduced within the case-by-case formulation for molecular states: those for open-shell, non-linear triatomic molecules (nltos) and open-shell, linear polyatomic molecules (lpos). The attribute `group` has been added to some symmetry elements for which it is relevant. Full documentation of the case-by-case Schemata is at http://xsams.svn.sourceforge.net/viewvc/xsams/branches/ucl-branch/docs/cbc-doc-0.2.1.pdf

10. An Environments element, at the top level of the XSAMS tree, has been introduced to describe one or more conditions of temperature, pressure and composition for reference. For example,

```
<Environment envID="Eair">
    <Temperature>
        <Value units="K">298.</Value>
    </Temperature>
    <TotalPressure>
        <Value units="atm">1.</Value>
    </TotalPressure>
    <Composition>
        <Species name="N2">
            <MoleFraction>
                <Value units="unitless">0.79</Value>
            </MoleFraction>
        </Species>
        <Species name="O2">
            <MoleFraction>
                <Value units="unitless">0.21</Value>
            </MoleFraction>
        </Species>
    </Composition>
</Environment>
```

    The species may also be referred to by `speciesRef` (see above), and composition specified by `PartialPressure`, `MoleFraction`, or `Concentration`.

11. Line broadening is contained within the `Broadenings` element, and consists of one or more of

    a. `DopplerBroadening`
    b. `StarkBroadening`
    c. `VanDerWaalsBroadening`
    d. `NaturalBroadening`

e. InstrumentBroadening

Each `Broadening` references an environment (through the `envRef` attribute) and contains one or more `Lineshape` elements. Lineshapes are identified through the ~~`name attribute`~~ `Name` element and contain one or more `LineshapeParameter` ~~(of type `NamedDataType`)~~ also named ~~by attribute~~. Examples:

```
<StarkBroadening envRef="EHe">
    <Lineshape name="Lorentzian">
        <LineshapeParameter name="gammaL">
            <SourceRef>BStarkB</SourceRef>
            <Value units="A">0.008</Value>
            <Accuracy>0.001</Accuracy>
        </LineshapeParameter>
    </Lineshape>
</StarkBroadening>
```

Here, **`EHe`** refers to some environment of He atoms defined elsewhere in the XML tree.
For the Lorentzian HWHM given as a function of pressure and temperature:

$$\gamma_{\mathrm{air}}(p, T) = \gamma_{\mathrm{air}}^{\mathrm{ref}}(p_{\mathrm{ref}}, T_{\mathrm{ref}}).(p/p_{\mathrm{ref}}).(T_{\mathrm{ref}}/T)^{\mathrm{n\_air}}$$

```
<VanDerWaalsBroadening envRef="Eair">
   <Lineshape name="Lorentzian">
      <Comments>The temperature-dependent pressure broadening
                Lorentzian lineshape</Comments>
      <LineshapeParameter name="gammaL">
          <FitParameters functionRef="FgammaL">
              <FitArgument units="K">
                  <Name>T</Name>
                  <LowerLimit units="K">240</LowerLimit>
                  <UpperLimit units="K">350</UpperLimit>
              </FitArgument>
              <FitArgument units="K">
                  <Name>p</Name>
                  <LowerLimit units="atm">0.</LowerLimit>
                  <UpperLimit units="atm">1.2</UpperLimit>
              </FitArgument>
              <FitParameter name="gammaL_ref">
                  <SourceRef>B_HIT-HF-g_air-2</SourceRef>
                  <Value units="1/cm">0.105</Value>
                  <Accuracy>0.0021</Accuracy>
              </FitParameter>
              <FitParameter name="n">
                  <SourceRef>B_HIT-HF-n_air-0</SourceRef>
                  <Value units="unitless">0.55</Value>
                  <Accuracy>None</Accuracy>
              </FitParameter>
          </FitParameters>
      </LineshapeParameter>
   </Lineshape>
</VanDerWaalsBroadening>
```

12. Line shifting data is placed within a `Shiftings` element, which contains one or more `Shifting` elements, each containing one or more `ShiftingParameter` elements. Each `Shifting` element carries and `envRef` attribute, to reference the shifting environment.

13. Basic types, `VectorType` and `MatrixType` have been introduced to represent Cartesian vectors and general matrices. e.g.

```
<Vector ref="…" x3="0." y3="1." z3="-2."/>

<SomeMatrix units="…" nrows="n" ncols="n" form="symmetric"
            values="real">
   <RowRefs>row_1 row_2 row_3 … row_n</RowRefs>
   <ColRefs>col_1 col_2 col_3 …col_n</ColRefs>
   <Matrix>0. 0. 1. -4. 2. … -3. 0.</Matrix>
</SomeMatrix>
```

`values` can be `real`, `imaginary` or `complex`. `form` identifies the type of matrix (`symmetric`, `skew-symmetric`, `diagonal`, etc.) and hence the number of (whitespace-delimited) values it is necessary to give within the `<Matrix>` tag (e.g. $n(n+1)/2$, n, n^2, etc.)

14. Basic types `DataListType` and `LinearSequenceType` represent simple lists of numerical data:

```
<DataList n="100" units="Mb">0. 0.13 0.12 0.08 …
                              1.20</DataList>
```

or:

```
<LinearSequence n="101" a0="500." a1="0.1"
                    units="1/cm"/>
```

This latter example represents the sequence (arithmetic progression):
`500. 500.1 500.2 … 510.0`

15. A `DataSeriesType` contains one of `DataList`, `LinearSequence`, or the name of an external file (containing the whitespace-delimited data in text format) along with either an `ErrorList` (giving accuracies for each data point) or a single `Error` (if the same accuracy applies to every data point).

16. A `SimpleDataTableType` contains one or more `DataSeriesTypes` called `X` and one or more `DataSeriesTypes` called `Y`.

17. An element, `CrossSection`, has been added to `RadiativeType`. It is an extension of `SimpleDataTableType`, with additional attributes to refer to species (`speciesRef`) and environment (`envRef`), and assigns vibrational bands through a `BandAssignment` element. E.g

```
<CrossSection id="azulene-paxs">
    <SourceRef>B_TCSD1</SourceRef>
    <SpeciesRef>X-CUFNKYGDVFVPHO-UHFFFAOYAT</SpeciesRef>
    <Description>Photoabsorption cross section of
                azulene</Description>
    <X parameter="photon energy" units="eV" id="D-eV-series1">
        <LinearSequence n="601" units="eV" a0="0." a1="0.05"/>
    </X>
    <Y parameter="sigma" units="Mb">
        <DataList n="601" units="Mb">
0 -1.3588884e-05 0.00029940086 0.0018259565 …84.611003
        </DataList>
        <Error>1.e-7</Error>
    </Y>
   </CrossSection>
```

18. One or more `BandAssignment` elements identify broad features in the spectrum by their central position and width, and gives assignments by reference to the normal modes listed in `MolecularChemicalSpecies`. e.g.

```
<CrossSection>
   ... [Cross section data] ...
   <BandAssignment name="2v1+v2">
     <BandCentre>
       <Value units="1/cm">410</Value>
       <Accuracy>2</Accuracy>
     </BandCentre>
     <BandWidth>
       <Value units="1/cm">40</Value>
       <Accuracy>5</Accuracy>
     </BandWidth>
     <Modes>
       <DeltaV modeID="v1">2</DeltaV>
       <DeltaV modeID="v2">1</DeltaV>
     </Modes>
   </BandAssignment>
</CrossSection>
```

19. `NormalModes` resides within `MolecularChemicalSpecies` and lists the molecule's normal mode frequencies, intensities, and displacement vectors. e.g.

```
<NormalModes electronicStateRef="SX_Azulene-1">
  <NormalMode id="v1" pointGroupSymmetry="A1">
     <HarmonicFrequency>
        <Value units="1/cm">162</Value>
           <Accuracy>1</Accuracy>
     </HarmonicFrequency>
     <Intensity>
        <Value units="km/mol">0</Value>
     </Intensity>
    <DisplacementVectors units="Å">
       <Vector ref="C1" x3="0." y3="0.001" z3="0.0005"/>
       <Vector ref="C2" x3="0.01" y3="-0.001" z3="0.0005"/>
       <Vector ref="C3" x3="-0.005" y3="0.001" z3="0."/>
       <!-- etc... -->
```

```
        </DisplacementVectors>
      </NormalMode>
    </NormalModes>
```

20. The anharmonicity tensor is given within a MatrixData element, within `MolecularChemicalSpecies`. It lives inside the `StableMolecularProperties/OtherProperties` tag. e.g.

```
<StableMolecularProperties>
  <OtherProperties>
    <Name>Anharmonicity Tensor</Name>
    <MatrixData units="1/cm" nrows="48" ncols="48"
                form="symmetric" values="real">
      <RowRefs>v1 v2 v3 ... v48</RowRefs>
      <ColRefs>v1 v2 v3 ... v48</ColRefs>
      <Matrix>
          10.43 2.45 -4.01 0.49 -78.9 ... 12.4
      </Matrix>
     </MatrixData>
  </OtherProperties>
</StableMolecularProperties>
```

21. The `PartitionFunctionType` defines tables of temperature-dependent partition functions for atoms and molecules. The `PartitionFunction` element may be used within `MolcularChemicalSpecies`. e.g.

```
<PartitionFunction>
  <T parameter="Temperature" units="K">
    <DataList n="15" units="K">100. .. 1500.</DataList>
  </T>
  <Q parameter="Partition Function" units="unitless">
    <DataList n="15" units="unitless">56.432 154.342 ...
              3445.321</DataList>
  </Q>
</PartitionFunction>
```

22. `MolecularState` also has a `StateExpansion` element to give the coefficients of a linear expansion in terms of some basis. e.g.

```
<StateExpansion>
  <Coeff stateRef="Sbb1">0.7071</Coeff>
  <Coeff stateRef="Sbb2">0.7071</Coeff>
</StateExpansion>
```

The basis itself is a list of `MolecularState` elements within `Molecule`. e.g.

```
<BasisStates>
    <MolecularState stateID="Sbb1">
    <dcs:QNs>
      <dcs:v>0</dcs:v>
      <dcs:J>3</dcs:J>
```

```
    ...
  </dcs:QNs>
  </MolecularState>
  <MolecularState stateID="Sbb2">
    <dcs:QNs>
      <dcs:v>0</dcs:v>
      <dcs:J>4</dcs:J>
      ...
    </dcs:QNs>
  </MolecularState>
</BasisStates>
```