MSSML: A Molecular Spectroscopic Simulations Markup Language for Rovibrational Studies

Javier Díaz, Sebastián Reyes, Camelia Muñoz-Caro, and Alfonso Niño

Grupo de Química Computacional y Computación de Alto Rendimiento, Escuela Superior de Informática,
Universidad de Castilla-La Mancha, Paseo de la Universidad 4, 13071, Ciudad Real, Spain
{javier.diaz,sebastian.reyes,camelia.munoz,alfonso.nino}@uclm.es
http://qcycar.inf-cr.uclm.es

Abstract. This work presents the development of an XML based language for the standardization of the information needed to build molecular rovibrational hamiltonians. This Molecular Spectroscopic Simulations Markup Language (MSSML) allows to collect and represent in a unified way the information obtained from electronic structure calculations performed in heterogeneous distributed systems, such as Internet-based Grids of computers. This standard language will permit the use of a web (Grid) services approach to automate the simulation of rovibrational spectra.

Keywords: Rovibrational spectra, Spectra simulation, Data standardization, XML, MSSML.

1 Introduction

Different theoretical models can be used to perform molecular rovibrational spectra simulations. Thus, the simple rigid rotor-harmonic oscillator model can be used to get an approximate estimate of rotational and vibrational frequencies using equilibrium structure information [1]. Most of the nowadays electronic structure codes applies this model to provide fundamental vibration frequencies. In this context, it is worth mentioning the SNF program [2]. SNF allows the calculation of vibrational spectra using the harmonic approximation for frequencies and the double harmonic approximation for intensities. More realistic models can be built selecting an appropriate set of vibrational coordinates and analyzing the variation of structure and energy as these coordinates change. For instance, flexible molecules with only one accessible equilibrium position (i.e., semi-rigid molecules) can use the full set of normal modes as vibrational coordinates. The standard Watson Hamiltonian model uses this approach [3]. On the grounds of the Born-Oppenheimer separation [4] [5] the total energy of an electronic structure calculation defines the potential for the nuclear motion. On the other hand, it is possible to generate a set of molecular structures by displacing from the equilibrium position along the different vibrational coordinates. Thus, by performing electronic structure calculations for the set of structures generated we have a way to map the potential for the nuclear motion. Formally, this process represents a mapping of the molecular potential energy hypersurface. In addition, from the structures used we can build the kinetic part of the rovibrational Hamiltonian. This approach has been used to accurately simulate rovibrational spectra in semi-rigid and non-rigid molecules, see for instance [6],[7],[8], and references therein. To deal with problems of large size, a methodology for the automatic mapping of potential energy hypersurfaces has been developed for working on Internet-based Grids of computers [9]. Here, a set of different molecular structures are generated and submitted for computation in the Grid and the results are collected in a single file. Any of the available electronic structure codes can be used to perform the set of calculations. However, each software tool has a particular way to describe the input data and the results of the calculations. In other words, the specification of the information is not unified. This problem requires to know the syntax of every program to use. This situation becomes more troublesome when we need to share information among computational systems. In this case, it is necessary to identify the program generating the information before it can be used. In particular, we face the above problems when trying to represent the information needed to simulate rovibrational spectra from the molecular electronic structure package used to map the corresponding potential energy hypersurface. A standardized way to represent and collect the information generated is needed in order to unify the representation. In addition, this would allow to simplify the later automation of spectra simulations using software tools based in a web services approach.

Generalized markup languages, such as SGML (Standard Generalized Markup Language) [10] or XML (eXtensible Markup Language) [11], can be used to make explicitly the structure of a document. SGML is very powerful but also highly complex; in fact, most manufacturers could not implement it completely. XML, on the other hand, has been designed to be simpler, easier to use and smaller, being a fully conforming subset of SGML.

XML is a metamarkup language for text documents [12]. In other words, it is a language to define languages, and any XML based document can be read by any tool able to read text files. In XML documents the data are included as strings of text. These data are surrounded by text markups describing them. The XML specification defines the exact syntax these markups must follow. XML allows the definition of markup vocabularies and sets of grammatical rules to properly combine such vocabularies. The grammars specify where tags may be placed, how they must look like, which element names are legal, how attributes are attached to elements, and so forth. Given an XML based language, its associated grammar is specific enough to allow the development of XML parsers that can read documents written in that language. Documents that satisfy its intended grammar are said to be well-formed.

The markup in an XML document describes the structure of the document, specifying how its constituent elements are associated. In a well-designed XML document, the markup also describes the document semantics. The markup allowed for a particular XML application can be documented in a schema.

Particular document instances can be compared to the schema. Documents that match the schema are said to be valid. Documents that do not match are invalid. Validity depends on the schema. That is, whether a document is valid or invalid depends on which schema you compare it to. Not all documents need to be valid.

There are many different XML schema languages, with different levels of expressivity. The most broadly supported schema language, and the only one defined by the XML 1.0 specification itself, is the document type definition (DTD). A DTD lists all the legal markups and specifies where and how they can be included in a document. DTDs are optional in XML. On the other hand, DTDs may not always be enough. The DTD syntax is quite limited and does not allow to make many useful statements such as "This element contains a number" or "This string of text is a date between 1974 and 2032." The W3C XML Schema Language (which sometimes goes by the misleadingly generic label schemas) allows to express constraints of this nature [13].

Both, Document Type Definitions (DTD's) and XML Schemas (XSD's, also known as WXS) are ways to define XML-based data models. However, there are many technical benefits in the use of XML Schemas versus DTD, including:

- Support for primitive (built-in) data types (i.e.: xsd:integer, xsd:string, xsd:date, and so on), which facilitates using XML in conjunction with other typed-data, including relational data.
- The ability to define custom data types, using object-oriented data modeling principles: encapsulation, inheritance, and substitution.
- Compatibility with other XML technologies. For example, Web services, XQuery, XSLT and other technologies can optionally be schema-aware.

In the Computational Chemistry field there are several efforts to create languages which unify the document structure in different parts of the area. Therefore, some XML languages have been developed, among them CML (Chemical markup language) [14] [15]. CML is a new approach to managing molecular information. It has an extensible scope, as it covers disciplines from macromolecular sequences to inorganic molecules and quantum chemistry. CML has an extension called CMLSpect [16] for managing spectral, and other analytical, data. This language allows the user to describe different kinds of spectra. In this way, using this language, it is posible to describe the spectra of an equilibrium structure. In addition, it is possible to draw the spectra in a visual tool. Another interesting language is QC-ML (Quantum Chemistry Markup Language) [17]. This language establishes a common format for quantum chemistry information, enhancing code interoperability and communication between different programs. The goal of this language is to unify the input and output file formats for the different molecular electronic structure codes.

In this paper we present a full formal specification of the data needed for rovibrational molecular spectra simulations. These data correspond to a set of different molecular structures. The work is oriented to the sharing of information in heterogeneous distributed systems such as Internet-based Grids of computers. So, a new XML language called MSSML (Molecular Spectroscopic Simulations Markup Language) is proposed. This new language collects in a unified way the information, obtained from molecular electronic structure codes, needed to simulate rovibrational spectra. Thus, the structure, potential nuclear energy, vibrational coordinates and dipole moment (among other information) from several molecular structures can be represented. This language normalizes the input for any simulation programs that needs to use this information. Moreover, using XML parsers, error checking and validation is easy by using an appropriate XML Schema, also presented in this work. Nevertheless, from the information contained in a MSSML file, we can derive kinetic and potential energy expressions for anharmonic vibrations as shown in [18],[19].

The rest of this paper is organized as follows. In section 2 an overview of the new language is presented. Section 3 details the architecture of the language. In section 4, we present the corresponding XML Schema needed to validate documents. Finally, in section 5 we present the main conclusions of the work.

2 Molecular Spectroscopic Simulations Markup Language, MSSML

Our objective is to develop a language to standardize the format of the set of data needed for the simulation of rovibrational spectra. The data are generated from molecular electronic structure (ab initio) calculations. The goal is to collect the information obtained in molecular hypersurface scans performed in computational Grid environments as described in [9]. The different structures defining the scan are generated as a function of vibrational coordinates. Since at present we focus in semi-rigid molecules, normal modes are used as vibrational coordinates in the framework of the Watson hamiltonian model [3]. A language as the one proposed here would be a common way for representing the structural information needed to build different kinds of rovibrational spectroscopic models.

To define the language we start by establishing the basic requirements it must satisfy. Therefore, our language must be:

- Self-descriptive
- Easily and unambiguously read by both users and computational systems
- Able to logically connect its constituent elements in a clearly expressed nesting structure of statements
- Interoperable among computational systems
- Extensible

These requirements are fulfilled using the XML metalanguage because of the properties of XML based languages. First, these languages are easy to understand since they are self-descriptive, i.e., an XML based language contains markups defining the data of a document. Therefore, it is not necessary other document to explain the information contained. On the other hand, an XML based language can be structured using the markup tags. Moreover, since XML languages are plain text based, they provide interoperability between computational systems, readability of the documents for the users, and efficiency in the use of the information. Finally, an XML based language can be extended to include new kinds of information. All these characteristics make XML the most suitable option to build a language oriented to the standardized interchange of information (between users and/or computational systems). This is specially important in distributed systems such as Internet-based Grids of computers. Thus, we focus in developing a Molecular Spectroscopic Simulations Markup Language, in short MSSML.

3 MSSML Architecture

The information we wish to collect in the MSSML is obtained from an appropriate theoretical molecular structure scan. Since we are dealing with semi-rigid molecules, this scan is generated by allowing displacements on the different normal modes. In other words, we are using the set of normal modes as a complete set of vibrational coordinates. The information to collect is obtained from molecular electronic structure calculations. To properly collect and organize the information using an XML standardized language we have used the following approach. The root element, or global container, of each MSSML document describes the molecule under study. Obviously the corresponding tag element is called Molecule. The <Molecule> element is composed by lower level elements which define the molecular characteristics, the characteristics of the level of theory used and the information generated in the study. In addition, this information can be formally considered divided in two sections. The first one refers to the equilibrium information. The second corresponds to the structural scan information.

The basic information starts with an attribute (name) that defines an ID for the molecule. The kind of theoretical study is defined by the level of theory used, element <TheoryLevel>. The molecular composition is defined specifying the number of atoms with the element <Natoms>, followed by a list of the corresponding atomic numbers specified in an element called <AtomList>. These elements appear organized in a formatted document as follows:

```
<Molecule name="WaterDimer">
  <TheoryLevel>MP2 (Full) /6-311++G(2d,2p)</TheoryLevel>
  <Natoms>6</Natoms>
  <AtomList>8 8 1 1 1 1</AtomList>
  .....
  .....
</Molecule>
```

where the water dimer has been used as a case model.

The previous set of elements defines the basic information about the molecule. An independent section, identified by the element <Equilibrium>, collects the symmetry, the structure, and the harmonic vibrational data of the molecule at its equilibrium position. Thus, a <PointGroup> element identifies the point group of the molecule. On the other hand, the equilibrium structure, defined by the nuclear cartesian coordinates, is included through the element <EquilibriumCartesianCoordinates>. This element is composed by three inner elements corresponding to the x, y and z coordinates given in Angstroms, elements <x>, <y> and <z>. The three principal moments of inertia are included in the <EquilibriumInertialMoments> element. The following elements are used to define the vibrational information. Therefore, a <NormalModesNumber> element tag specifies the number of normal modes. The normal modes themselves are defined in a new section labeled with a <NormalModes> element. This element contains the information of each mode within the subelement <Mode>. The normal modes can be arbitrarily ordered. However, usually, normal modes are ordered in increasing value of their fundamental vibrational frequencies. This is the typical order in the molecular electronic structure codes. Each mode is identified by an order number, element <Number>. In addition, each mode bears its point group symmetry, harmonic frequency (in cm^{-1}), reduced mass (in AMU) and force constant (in mDyn/Angstrom). These data are found under the elements <Symmetry>, <Frequency>, <RedMass> and <ForceConstant>, respectively. Finally, the cartesian displacements defining the normal mode are also included, element <Components>. This element is organized in three groups corresponding to the displacements on the x, y and z axis, elements <x>, <y> and <z>. This organization of information will look as in the following document fragment, corresponding to the water dimer:

```
<Molecule name="WaterDimer">
  .....
  .....
  <Equilibrium>
    <PointGroup>CS</PointGroup>
    <EquilibriumCartesianCoordinates>
        <x>0.000081 0.000081 0.489136 0.489136 -0.903305
            -0.076255</x>
        <y>-1.390930 1.519775 -1.711558 -1.711558 1.833730
            0.558632</y>
        <z>0.000000 0.000000 0.759256 -0.759256 0.000000
            0.000000</z>
    </EquilibriumCartesianCoordinates>
    <EquilibriumInertialMoments>8.28831 281.32274 281.33332</
       EquilibriumInertialMoments>
    <NormalModesNumber>12</NormalModesNumber>
    <NormalModes>
        <Mode>
        <Number>1</Number>
        <Symmetry>A''</Symmetry>
        <Frequency>134.1867</Frequency>
        <RedMass>1.0694</RedMass>
        <ForceConstant>0.0114</ForceConstant>
        <Components>
            <x>0.00 0.00 0.36 -0.36 0.00 0.00</x>
            <y>0.00 0.00 0.34 -0.34 0.00 0.00</y>
            <z>0.00 0.06 -0.09 -0.09 -0.69 -0.15</z>
        </Components>
        </Mode>
        .....
        .....
        <Mode>
        <Number>12</Number>
        <Symmetry>A''</Symmetry>
        <Frequency>3974.5511</Frequency>
        <RedMass>1.0819</RedMass>
        <ForceConstant>10.0700</ForceConstant>
        <Components>
            <x>0.00 0.00 0.16 -0.30 0.00 0.00</x>
            <y>0.00 0.00 0.39 -0.24 0.00 0.00</y>
            <z>0.00 0.09 -0.04 -0.01 -0.29 -0.05</z>
        </Components>
        </Mode>
    </NormalModes>
  </Equilibrium>
  .....
  .....
</Molecule>
```

The previous <Equilibrium> section is the minimum needed in an MSSML document to perform rovibrational spectra simulations. In this case, we could only work at the rigid rotor level for rotation and the harmonic oscillator for vibration.

The next section is composed by the information arising from the molecular structure scan along the normal modes. This information is identified by the <StructureScan> element and is conformed by a set of different molecular structures. Each structure is identified by a <Structure> tag element. Each conformation has an attribute that identifies if the structure is valid (value "true") or not (value "false"). A conformation is considered valid if the molecular structure program has been able to finish the corresponding theoretical calculation providing the desired results. The elements composing a <Structure> are the displacements on the normal modes, <NormalModesIncrements>, the cartesian coordinates resulting from the displacement, <CartesianCoordinates>, the total energy of the structure, <Energy>, and the dipole moment, <DipoleMoment>. The <NormalModesIncrements> element contains a list with the increments used for each normal mode. The next element, <CartesianCoordinates>, represents the position of each nucleus, in Angstroms, after this displacement. Clearly, it should be organized in terms of the x, y and z components. Accordingly, we use again the <x>, <y> and <z> elements. On the other hand, the <Energy> element represents the total molecular energy for this molecular structure in atomic units. On the grounds of the Born-Oppenheimer approach this datum represents the potential energy for the nuclear motion. The last element, <DipoleMoment>, collects the dipole moment in Debyes, which is needed to compute intensities for vibrational transitions. This element is composed by three sub-elements corresponding to the three cartesian components: <dx>, <dy> and <dz>. When the conformation is not valid, only the <NormalModesIncrements> element appears in the document. In this way we can identify the displacement that generates the problem.

The following fragment code illustrates, for the water dimer, the case of a valid conformation:

```
<Molecule name="WaterDimer">
  .....
  .....
  <StructureScan>
    <Structure valid="true">
      <NormalModesIncrements>
        <Increment>0.453 0.000 0.000 0.000 0.000 0.000 0.000
            0.000 0.000 0.000 0.000 0.000</Increment>
      </NormalModesIncrements>
      <CartesianCoordinates>
        <x>0.000081 0.000081 0.489136 0.489136 -0.903305
            -0.076255</x>
        <y>-1.390930 1.519775 -1.711558 -1.711558 1.833730
            0.558632</y>
        <z>0.000000 0.000000 0.759256 -0.759256 0.000000
            0.000000</z>
      </CartesianCoordinates>
      <Energy>-152.109348171</Energy>
      <DipoleMoment>
        <dx>2.7636</dx>
        <dy>-0.7311</dy>
        <dz>0.4683</dz>
      </DipoleMoment>
    </Structure>
    .....
    .....
  </StructureScan>
</Molecule>
```

In case of an invalid conformation the information will appear as follows:

```
<Molecule name="WaterDimer">
  .....
  .....
  <StructureScan>
    <Structure valid="false">
      <NormalModesIncrements>
        <Increment>0.000 0.000 0.000 0.000 0.000 0.000 0.000
            0.000 0.000 0.000 0.000 0.100</Increment>
      </NormalModesIncrements>
    </Structure>
    .....
    .....
  </StructureScan>
</Molecule>
```

4 MSSML Schema

In the previous section, the architecture of the language proposed has been presented. Here, we are going to define the set of rules that our language must follow. These rules will define, among others, the characteristics of each element. This information can not be specified in the MSSML documents. However, it can be defined in an XML Schema [13]. An XML Schema is a formalization of the constraints, expressed as rules, that apply to a class of XML documents. Therefore, the main use for a Schema is the validation of documents. So, by validating documents against Schemas, we can ensure that the information of a given document is correctly structured according to the set of allowed rules. In this way, software tools can be used to read documents written in an XML language, ensuring that the information received by the software is properly formatted.

Table 1. Summary of the elements used by the Molecular Spectroscopic Simulations Markup Language (MSSML)

Element	Type	Parent
Molecule	xsd:complexType	Root
TheoryLevel	xsd:string	Molecule
Natoms	xsd:int	Molecule
AtomList	mssml:AtomListType	Molecule
Equilibrium	xsd:compexType	Molecule
PointGroup	xsd:string	Equilibrium
EquilibriumCartesian-Coordinates	mssml:xyzType	Equilibrium
EqulibriumInertial-Moments	mssml:DoubleListType	Equilibrium
NormalModesNumber	xsd:int	Equilibrium
NormalModes	xsd:compexType	Equilibrium
Mode	xsd:complexType	NormalModes
Number	xsd:int	Mode
Symmetry	xsd:string	Mode
Frequency	xsd:double	Mode
RedMass	xsd:double	Mode
ForceConstant	xsd:double	Mode
Components	mssml:xyzType	Mode
StructureScan	xsd:complexType	Molecule
Structure	xsd:complexType	StructureScan
NormalModesIncrements	xsd:complexType	Structure
Increment	mssml:DoubleListType	NormalModesIncrements
CartesianCoordinates	mssml:xyzType	Structure
Energy	xsd:double	Structure
DipoleMoment	xsd:complexType	Structure
dx, dy, dz	xsd:double	DipoleMoment
x, y, z	mssml:DoubleListType	Elements of type mssml:xyzType

Table 2. Summary of the Molecular Spectroscopic Simulations Markup Language (MSSML) element types

Element Type	Description
AtomListType	Composed by a list of int numbers
DoubleListType	Composed by a list of double numbers
xyzType	Composed by x, y and z elements

Considering the advantages of an XML Schema, we have defined one for MSSML. Following the W3C recommendation for providing uniquely named elements and attributes in an XML document [11] we encapsulate our MSSML elements defining a namespace. This namespace is used to define the Schema. The URI corresponding to the namespace is associated to out research group: http://qcycar.inf-cr.uclm.es/MSSML.xsd. The prefix used in the Schema for the namespace is mssml. In addition, there is another namespace used, defined in all Schemas, called xsd. This namespace is necessary because it defines the basic types of each element.

We have defined all the elements of the MSSML as qualified, i. e., the elements belong to the target namespace. The meaning and syntax of the elements of the MSSML have been presented in the previous section. In Table 1, we present a summary of all elements considered as well as the corresponding types.

As we can see in Table 1, we have used some basic types (namespace xsd), as for example string, int, double or complexType. We must keep in mind that, in our case, the complexType element contains a sequence of other elements. We have also used some other user defined element types (from the namespace mssml). Thus, we have defined three element types in the XML Schema in order to reuse code in the own definition of the MSSML. Moreover, these element types make the XML Schema easier to understand. Table 2 shows these element types. The first type, AtomListType, is a list of int numbers. The second type, DoubleListType, is a list of double precision real numbers. Both lists separate items by a space. The last type, xyzType, is a sequence of three DoubleListType elements. These elements are <x>, <y> and <z>.

According to these specifications, the Appendix at the end of this paper presents the Schema for MSSML in full.

5 Conclusions

We have developed an XML language for Molecular Spectroscopic Simulations called MSSML (Molecular Spectroscopic Simulations Markup Language). The language allows to collect the information generated in molecular structure scans performed with electronic structure software tools in distributed and heterogeneous systems such as Internet-based Grids of computers. Using normal modes as basic vibrational coordinates MSSML permits to store in a standardized way the information needed for building anharmonic rovibrational hamiltonians. Also, we have defined an XML Schema for MSSML, which contain the restrictions of the language. Using the Schema we can automatically test not only that an MSSML document is well-formed, but also that its has a valid structure.

As future work, we plan to extend our language to include internal or symmetry coordinates as vibrational coordinates.

Acknowledgments. This work has been cofinanced by FEDER funds and the Consejería de Educación y Ciencia de la Junta de Comunidades de Castilla-La Mancha (grant # PBI08-0008). The Ministerio de Educación y Ciencia (grant # FIS2005-00293) and the Universidad de Castilla-La Mancha are also acknowledged.

References

1. Levine, I.N.: Quantum Chemistry, 5th edn. Prentice-Hall, Englewood Cliffs (2000)
2. Neugebauer, J., Reiher, M., Kind, C., Hess, B.A.: Quantum Chemical Calculation of Vibrational Spectra of Large Molecules. Raman and IR spectra for Buckminsterfullerene. J. Comput. Chem. 23, 895–910 (2002)
3. Meyer, H.: The Molecular Hamiltonian. Ann. Rev. Phys. Chem. 53, 141–172 (2002)
4. Szabo, A., Ostlund, N.S.: Modern Quantum Chemistry. McGraw-Hill, New York (1989)
5. Jensen, F.: Introduction to Computational Chemistry, 2nd edn. John Wiley & Sons, Chichester (2007)
6. Castro, M.E., Niño, A., Muñoz-Caro, C.: Structural and Vibrational Theoretical Analysis of Protonated Formaldehyde in its $\widetilde{X}^1$ A' Ground Electronic State. Theor. Chem. Acc. (2008) DOI: 10.1007/s00214-007-0392-5
7. Niño, A., Muñoz-Caro, C., Moule, D.C.: Three Dimensional Vibrational Study of the Coupling between the Methyl Torsion and the Molecular Frame in the S_0 State of Acetaldehyde. J. Phys. Chem. 99, 8510–8515 (1995)
8. Puzzarini, C., Tarroni, R., Palmieri, P., Demaison, J., Senent, M.L.: Rovibrational Energy Levels and Equilibrium Geometry of HCP. J. Chem. Phys. 105, 3132–3141 (1996)
9. Reyes, S., Muñoz-Caro, C., Niño, A., Badia, R.M., Cela, J.M.: Performance of Computationally Intensive Parameter Sweep Applications on Internet-based Grids of Computers: the Mapping of Molecular Potential Energy Hypersurfaces. Concurr. Comput.: Pract. Exper. 19, 463–481 (2007)
10. International Standards Organization.: Standard Generalized Markup Language (SGML). ISO/IEC IS 8879 (1986)
11. W3C Recommendation.: Extensible Markup Language (XML) 1.0 (Last visit: January 2008), `http://www.w3.org/XML`
12. Rusty Harold, E., Scott Means, W.: XML in a Nutshell. O'Reilly, Sebastopol (2002)
13. Van der Vlist, E.: XML Schema. O'Really, Sebastopol (2002)
14. Murray-Rust, P., Rzepa, S.H.: Chemical Markup, XML, and the Worldwide Web. 1. Basic Principles. J. Chem. Inf. Comput. Sci. 39, 928–942 (1999)
15. Murray-Rust, P., Rzepa, S.H.: Chemical Markup, XML, and the Worldwide Web. 4. CML Schema. J. Chem. Inf. Comput. Sci. 43, 757–772 (2002)
16. Murray-Rust, P., Rzepa, S.H.: Chemical Markup, XML, and the Worldwide Web. 7. CMLSpect, an XML Vocabulary for Spectral Data. J. Chem. Inf. Comput. Sci. 43, 757–772 (2002)

17. Angeli, C., Bendazzoli, G.L., Borini, S., Cimiraglia, R., Emerson, A., Evangelisti, S., Maynau, D., Monari, A., Rossi, E., Sanchez-Marin, J., Szalay, P.G., Tajti, A.: The Problem of Interoperability: A Common Data Format for Quantum Chemistry Codes. Int. J. Quantum Chem. 104, 2082–2091 (2007)
18. Muñoz-Caro, C., Niño, A.: Vibrational energy levels and vibronic structure of electronic spectra in molecules with large amplitude motions. Comput. Chem. 18, 413–417 (1994)
19. Niño, A., Muñoz-Caro, C.: Computation of kinetic constants for large range internal motions in molecules. Comput. Chem. 18, 27–32 (1994)

Appendix: XML Schema for MSSML

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://qcycar.inf-cr.uclm.es/MSSML.xsd"
  xmlns:mssml="http://qcycar.inf-cr.uclm.es/MSSML.xsd"
  attributeFormDefault="unqualified" elementFormDefault="
      qualified">
  <xsd:element name="Molecule">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="TheoryLevel" type="xsd:string"></
            xsd:element>
        <xsd:element name="Natoms" type="xsd:int"></
            xsd:element>
        <xsd:element name="AtomList" type="mssml:AtomListType
            "></xsd:element>
        <xsd:element name="Equilibrium">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="PointGroup" type="xsd:string
                  "></xsd:element>
              <xsd:element name="
                  EquilibriumCartesianCoordinates" type="
                  mssml:xyzType"></xsd:element>
              <xsd:element name="EquilibriumInertialMoments"
                  type="mssml:DoubleListType"></xsd:element>
              <xsd:element name="NormalModesNumber" type="
                  xsd:int"></xsd:element>
              <xsd:element name="NormalModes">
                <xsd:complexType>
                  <xsd:sequence maxOccurs="unbounded">
                    <xsd:element name="Mode">
                      <xsd:complexType>
                        <xsd:sequence>
                          <xsd:element name="Number" type="
                              xsd:int"></xsd:element>
                          <xsd:element name="Symmetry" type="
                              xsd:string"></xsd:element>
                          <xsd:element name="Frequency" type=
                              "xsd:double"></xsd:element>
                          <xsd:element name="RedMass" type="
                              xsd:double"></xsd:element>
                          <xsd:element name="ForceConstant"
                              type="xsd:double"></xsd:element
                              >
                          <xsd:element name="Components" type
                              ="mssml:xyzType"></xsd:element>
                        </xsd:sequence>
                      </xsd:complexType>
                    </xsd:element>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="StructureScan" minOccurs="0">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="Structure" maxOccurs="
                  unbounded">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="NormalModesIncrements"
                        >
                      <xsd:complexType>
                        <xsd:sequence>
                          <xsd:element name="Increment" type=
                              "mssml:DoubleListType"></
                              xsd:element>
                        </xsd:sequence>
                      </xsd:complexType>
                    </xsd:element>
                    <xsd:element name="CartesianCoordinates"
                        type="mssml:xyzType" minOccurs="0"></
                        xsd:element>
                    <xsd:element name="Energy" type="
                        xsd:double" minOccurs="0"></
                        xsd:element>
                    <xsd:element name="DipolarMoment"
                        minOccurs="0">
                      <xsd:complexType>
                        <xsd:sequence>
                          <xsd:element name="dx" type="
                              xsd:double"></xsd:element>
                          <xsd:element name="dy" type="
                              xsd:double"></xsd:element>
```

```
                          <xsd:element name="dz" type="
                              xsd:double"></xsd:element>
                        </xsd:sequence>
                      </xsd:complexType>
                    </xsd:element>
                  </xsd:sequence>
                  <xsd:attribute name="valid" type="
                      xsd:boolean" use="required"></
                      xsd:attribute>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:string" use="
          required"></xsd:attribute>
    </xsd:complexType>
  </xsd:element>
  <xsd:complexType name="xyzType">
    <xsd:sequence>
      <xsd:element name="x" type="mssml:DoubleListType"></
          xsd:element>
      <xsd:element name="y" type="mssml:DoubleListType"></
          xsd:element>
      <xsd:element name="z" type="mssml:DoubleListType"></
          xsd:element>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:simpleType name="AtomListType">
    <xsd:list itemType="xsd:int"></xsd:list>
  </xsd:simpleType>
  <xsd:simpleType name="DoubleListType">
    <xsd:list itemType="xsd:double"></xsd:list>
  </xsd:simpleType>
</xsd:schema>
```