# JUMBO – An XML infrastructure for eScience

**Yong Zhang** [a], Peter Murray-Rust [a], Martin T Dove [c], Robert C Glen [a], Henry S Rzepa [b], Joe A Townsend [a], Simon Tyrrell [a], Jon Wakelin [c], Egon L Willighagen [d]

[a] Unilever Centre for Molecular Science Informatics, Chemistry Department, University of Cambridge, Lensfield Road, Cambridge, CB2 1EW, UK,
[b] Chemistry Department, Imperial College, London, SW7 2AY, UK ,
[c] Department of Earth Sciences, University of Cambridge, Downing Street, Cambridge CB2 3EQ,
[d] Laboratory of Analytical Chemistry, Toernooiveld 1, 6525 ED Nijmegen, NL

## Abstract

JUMBO is an OpenSource toolkit addressing the semantic and ontological impedances that are major barriers to interoperability in computational chemistry and physics. Users build XMLSchemas from generic XML components to support particular computational tasks, such as high-throughput chemistry. JUMBO components provide a complete semantic description of information to or from a code such as MOPAC or GAMESS. Codes are edited to use JUMBO libraries as adapters to program-independent XML objects, or output is transduced using a generic parser, JUMBOMarker. The JUMBO system is designed for flexible collaborative contributions.

## 1. Introduction

Existing codes in quantum mechanics or crystal/molecular mechanics/dynamics rely on "FORTRAN-like" input and output with variable and misunderstandable semantics and ontology. Typical examples are the lack of explicit scientific units or fuzziness over whether a molecule has a charge. It is difficult to chain the output of one program into the input of the next so human "cut-and-paste" is frequent. JUMBO allows authors and users to wrap these codes in XML and convert them to workflow components.

JUMBO uses context-independent XMLSchema components to assemble problem-specific bespoke schemas. It has been developed for computational chemistry and crystal physics, with ca 250 components based on CML (Chemical Markup Language[3][4][5]). However many of these are generic to physical science (matrices, units, geometry, etc.) and users in other domains can create extend these for their own purposes. Many information components in physical science require semantic support through procedural code (e.g. interconversions of units, calculation of molecular mass). JUMBO is therefore designed to create fully functional code automatically from the schemas. Although some Java-based systems (Castor[8], JAXB[9], XMLBeans[10], etc.) can do this in Java, we require greater language support (e.g. FORTRAN) and customisability. Many components such as `<scalar>` are abstract and rely on dictionaries (v.i.) to add semantics and ontology. In practice many of the concepts in a program are not in the schema but provided by domain-specific dictionaries.

The ultimate goal is to provide simple and flexible tools to XMLise existing and future computational codes so they can be used as "black-boxes" in eScience workflows. Some authors have adopted this approach and embedded calls to JUMBO libraries in their code for input and output. However in some cases native legacy methods must still be used (for example where we do not have access or permission to modify source). In these cases we use XSLT to create legacy inputs and have developed a generic parser, JUMBOMarker, to translate "logfiles" to marked-up XML.

## 2. JUMBO strategy and infrastructure

The JUMBO process consists of:

- Assembling a schema from components. These components can include additional support such as validation protocols, examples and declarative semantics.
- Translating the schema to the target language (Java, C++, Python, FORTRAN).
- Compiling the schema to libraries and creating documentation (javadoc, HTML, PDF, Wiki).
- Compiling the examples and validating the code against them.

- Linking the libraries into the problem-specific code(s) *or*
- Writing a JUMBOMarker template to parse the output of the code to XML.

Additionally users create dictionary entries for their code and domain. A code may give rise to several hundred dictionary entries (e.g. for input and output quantities, strategies).

The final problem-specific code, libraries, converters, and templates are then integrated into an XML-aware "black-box" component, often run as an ant procedure (Figure 1).


Figure 1. Computation flow for a molecule-based code, "foo". The main XML flow is down the center, and JUMBO-based XMLization of the code is shown by blue dots. Many components are semantically enhanced by the foo-specific and the generic CMLComp dictionaries.

## 3. Architecture and Implementation

### 3.1 Schema

The schema is built from four types of components:

- **elements**. Each is defined in a separate `xsd:element`.
- **attributeGroups**. Every attribute in the schema is defined in a separate `xsd:attribute` contained in an `xsd:attributeGroup`. Many attributes are used by several elements and although some are only used by one element, all are potentially re-usable.
- **types**. All types in the schema are defined as separate elements, either `xsd:simpleTypes` or `xsd:complexTypes`.
- **examples**. These exercise the functionality of the element or attribute, validate the schema and can be compiled into procedural code for testing.

A user can create a schema from all components (`cmlAll`) or a small subset to avoid schema bloat. Compilation is rapid so that iterative design and testing is supported, and the approach used protects users from the most forbidding aspects of XMLSchema.

### 3.2 Linking JUMBO

The schema-derived input/output libraries (Java, C++, Python, F90) then provide complete access to the compiled information objects (domain-specific DOMs). Each code can then use these libraries as adapters to program-independent XML objects (Figure 2).

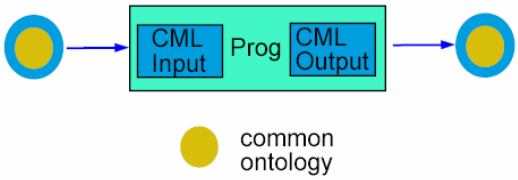


Figure 2. Code with embedded JUMBO libraries

### 3.3 Libraries

Besides the automatic creation of schema-derived libraries JUMBO uses a range of libraries to provide semantic behaviour, and the Java version is shown in Figure 3

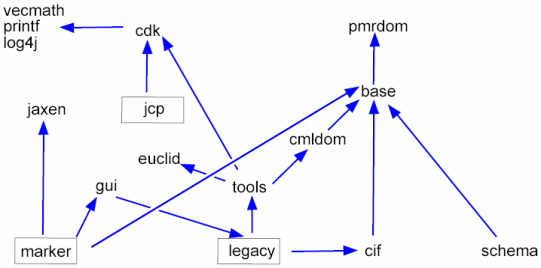


Figure 3 The modules and dependencies in JUMBO

The distribution includes

- `euclid` - general maths, arrays, matrices, 2- and 3-D geometry
- `pmrdom` A wrapper (delegation) for a W3C DOM (such as Xerces or Java1.4) to provide subclassable public constructors.
- `base`. The CMLDOM design creates an interface (`CMLFoo.java`) and an implementation (`FooImpl.java`) for each `xsd:element` subclassed from `base`.
- `schema` - code to convert the custom Schema to target languages.

- `gui`. A generic GUI builder.
- `marker` (JUMBOMarker). An XML-based language supporting structured regular expressions for parsing semi-structured documents to XML
- `legacy`. Converters between legacy formats and CML.
- `jcp` (JChempaint), `cdk` (Chemistry Development Kit), `cif` are OpenSource chemistry/crystallography toolkits.
- `vecmath`, `printf`, `log4j` and `jaxen` are generic OpenSource libraries.
- `tools`: per-element classes adding non-schema functionality to a CMLDOM element (v.i.)
- `xsl`. Stylesheets for generic rendering.

The JUMBO system uses:

- ant script(s) (`build.xml`) to compile and run all functions.
- XSLT stylesheets (`xsl`) for initial assembly of the schema.
- Java code (`nonschema`) to process the schema into Java (C++, etc.)
- XSLT and FOP to create documentation

### 3.4 Tools

The user also has access to a library of tools to provide problem-independent functionality. Thus `MoleculeTool` has over 100 methods for managing and computing molecular properties and wraps instances of (the schema-derived) `CMLMolecule`. Tool functionality is often complex and ours uses other OpenSource libraries such as CDK and JCP.

### 3.5 Dictionaries

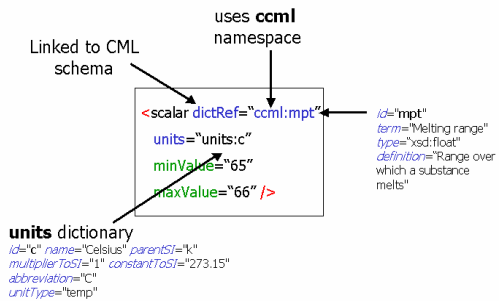


Figure 4. Dictionaries

Dictionaries (Figure 4) can be created and enhanced independently of the compilation of the schema into libraries. This allows users to add ontological functionality without recompiling the code. The community can then rationalise these ontologies, providing more general dictionaries and this has worked well in computational chemistry where concepts are well understood. By extension the system can support any discipline where the components can be captured as XMLSchemas.

### 3.6 JUMBOMarker

Where the code cannot be edited (technical or legal reasons) JUMBO provides generic adapters (Figure 5). To convert XML to legacy input is straightforward using XSLT. Transforming "flat-file" output to XML is more challenging and a template-based parsing mechanism, JUMBOMarker, is provided. The user defines a set of structured Schema-like XML templates as a grammar to model the output of the code. These are populated with regular expressions and work fast and efficiently for codes such as MOPAC[7], GULP[1] and GAMESS[6]. .
Retrieval and recall are very high – often 99%+ due to the formulaic output. The templates also define potential data structures for re-use by other programs or rendering agents.

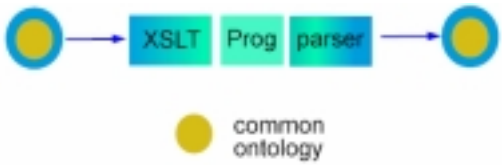


Figure 5. Generic adapters for a specific code

### 3.7 Experience

The schema-derived input/output libraries (Java, C++, Python, F90) then provide complete access to the compiled information objects (domain-specific DOMs). Each code can then use these libraries as adapters to program-independent XML objects (Figure 2).

The JUMBO strategy has so far been used for 3 codes:

- GULP (crystal physics). This is in FORTRAN and the JUMBO libraries have been integrated using an F95 DOM (see ref). A GULP-specific dictionary has been created[2].
- MOPAC (semi-empirical quantum mechanics). This is closed source so input and output is in legacy managed by XSLT and JUMBOMarker templates.
- GAMESS (general purpose computational chemistry). We are working with the authors to develop a black-box approach to computation, using XSLT for evaluating and transforming the input and either F95 libraries or wrapping for the output.

These have encouraged us to believe that most computational chemistry workflow can be

assembled from components, with extensions for each particular code. As more codes are converted to or wrapped in CMLComp we shall discover what concepts are common to all codes. The dictionaries are a flexible way of customising each code since it can be done by the authors or users, who often understand the code at the deepest level. Later it should be possible to merge large parts of the dictionaries into a common XML ontology for computational chemistry and physics.

## 4. Summary

The JUMBO system is well suited to communal distributed development where independent groups create their dictionaries and add JUMBO calls to some or all of the parts of their codes.
We are designing the next generation of code generators on common pseudocode. Then with language-specific serializers (Java, C++, Python and F90) are used to create the appropriate library for the application. The generic design of the system makes it useful for a wide range of tasks in physical sciences.

Because the results are in XML any generic technologies can be used to manage and enhance these We have used RSS to create automatic "news feeds" for the results of calculations, and Xindice to store and search the archive. There are many uses of XSLT combined with SVG or applets for rendering and reusing the results.

We acknowledge support from DTI/eScience project(YZ, SMT), NERC(JW), Unilever Research (YZ, SMT, JAT).

We thank many members of the OpenSource community, including CDK, JChemPaint; and Dan Zaharevitz (NCI) for support for JUMBO.

## 6. References

[1] J.D. Gale, *GULP - a computer program for the symmetry adapted simulation of solids*, JCS Faraday Trans., 93, 629, 1997.

[2] A. Garcia, P. Murray-Rust, J. Wakelin, *The use of XML and CML in Computational Chemistry and Physics Programs*, UK e-Science All Hands Meeting, September 2004.

[3] P. Murray-Rust and H. S. Rzepa, *CML Schema*, J. Chem. Inf. Comp. Sci., 43, 2003.

[4] P. Murray-Rust, R. C. Glen, Y. Zhang and J. Harter, *The World Wide Molecular Matrix - a peer-to-peer XML repository for molecules and properties*, 163-164 "EuroWeb2002, The Web and the GRID: from e-science to e-business", Editors: B. Matthews, B. Hopgood, M. Wilson, 2002 The British Computer Society.

[5] P. Murray-Rust, R. C. Glen, H. S Rzepa, J. J. P. Stewart, J. A. Townsend, E. L. Willighagen, Y. Zhang, *A semantic GRID for molecular science*, UK e-Science All Hands Meeting, September 802-809, 2003.

[6] M.W. Schmidt, K.K. Baldridge, J.A. Boatz, J.H. Jensen, S. Koseki, M.S. Gordon, K.A. Nguyen, T.L. Windus, S.T. Elbert, *General Atomic and Molecular Electronic Structure System (GAMESS)*, QCPE Bulletin, volume 10, 1990.

[7] J. J. P. Stewart., *MOPAC: A General Molecular Orbital Package*, Quant. Chem. Prog. Exch., 10:86, 1990.

[8] Castor, http://www.castor.org/.

[9] JAXB, http://java.sun.com/xml/jaxb/

[10] XMLBeans, http://xml.apache.org/xmlbeans/