

5. APPLICATIONS

previous and next elements to allow iteration. The *StarAssoc* class contains methods for manipulating STAR objects through setting and retrieval of data names and assignment of values. *DataBlock* is the class containing all the lower-order items and associations, and one or more such *DataBlock* classes comprise the *StarFile* class.

The *OOSTAR* approach also provides *Dictionary* and *DictionaryElement* classes so that STAR applications that do describe element attributes in external dictionary files can make use of such information.

Some sample applications were built with the class libraries of the *OOSTAR* toolset, and they are described in Chang & Bourne (1998). They include: simple converters of STAR data files to HTML pages with hyperlinks to associated dictionary entries; a query tool for retrieving the data values for a specific item specified by name within a specific data block; and a query mechanism that retrieves a set of items from a STAR loop structure and represents their values in an array, *i.e.* flattening if necessary any nested loops into a purely tabular presentation. These query tools are different from *Star_Base* primarily in that they do not return the context in which the data are found in the original file – their interpretation depends on a detailed *a priori* understanding of the target file structure.

This approach is rather different from the canonical description of STAR given in this volume, and was never developed into full-blown applications (in part because the developers recognized that the Java language would provide a preferable platform for further development). Nevertheless, it provides interesting ideas for the developer considering building object-oriented applications for STAR Files.

5.2.6.2. CIF++

CIF++ was a small library of classes designed by Peter Murray-Rust during the time that DDL was being developed as a language for representing the properties and attributes of STAR data items (*e.g.* Murray-Rust, 1993). The purpose of this project was to demonstrate the design of data classes that represented real-world objects such as molecules and crystal cells. At a time when the structure of CIF was under intense discussion, many of the classes were potentially extensible to STAR features that were a superset of those found in CIF.

While these classes were never developed into fully functional applications, they demonstrated a potentially fruitful approach to model representation and gave rise to the idea of including methods in STAR dictionary definitions, thus allowing STAR applications to dynamically associate algorithmic relationships and operations with data objects by parsing the methods description in the dictionary definitions. This approach is currently being developed into a relational expression language for STAR called dREL (Spadaccini *et al.*, 2000).

The ideas behind *CIF++* were further developed in a class library for molecular representation called *Democritos*, and have informed the representation of molecular and crystal structure in Chemical Markup Language (CML) and in the structured document browser *Jumbo* (Murray-Rust, 1998).

The *CIF++* classes have been refactored into Java using the W3C document object model (W3C, 2004) and other DOM-like models. They are now based on an XML schema which is the abstraction of the formal DDL1 specification (Chapter 2.5).

These are available as part of the *Jumbo* distribution at <http://cml.sf.net>. The design involves an interface that would allow the C++ classes to be recreated from the Java.

5.2.6.3. CIFOBJ

The *CIFOBJ* class library (Schirripa & Westbrook, 1996) was developed to provide an object view of the mmCIF dictionary and to complement the relational *CIFLIB* class library (Westbrook *et al.*, 1997) for handling mmCIF data. As such, it is very much tuned to crystallographic applications and it handles only the subset of STAR features used in CIF dictionaries. This does, however, include some limited handling of save frames, and is therefore a little more complex than applications interested only in CIF data input/output processing.

CIFOBJ has two components. The first builds a persistent store of objects of types item, subcategory, category and dictionary. Each such object is a container for all relevant attributes permitted for that object type. The object store is populated from the mmCIF dictionary by the *CIFOBJ* loader class (using methods provided by *CIFLIB*). This loader class assembles the dictionary objects and passes them to an object-storage manager. The second component of the *CIFOBJ* class library provides the methods necessary for building dictionary objects from the persistent store and for passing attribute strings to procedures concerned with establishing the integrity of data values.

The main reason for discussing this implementation here is to indicate the rapid growth in complexity needed to impose an application-specific object view on even a relatively simple STAR data structure. In generic prototype STAR projects such as *OOSTAR*, half a dozen or so classes and associated methods suffice to represent the highest-level abstract concepts implied in STAR constructions. In *CIFOBJ*, however, dozens of methods are associated with dictionary access. Dictionary-driven validation of an mmCIF numerical data value can involve:

- (i) retrieval from the dictionary of the extended type declaration associated with the data item;
- (ii) validation of the basic type (*i.e.* that it is indeed numeric) against the primitive data types supported by the dictionary;
- (iii) validation of the extended type by regular-expression matching of the string representation of the value against the allowed patterns stored in the dictionary;
- (iv) retrieval of any existing range constraints specified in the dictionary and comparison with the data value;
- (v) location and evaluation of any associated standard uncertainty;
- (vi) identification of the units in which the physical quantity is expressed;
- (vii) if necessary, conversion of the units according to the conversion tables stored in the dictionary.

More complexity arises from the relationships between data items expressed through dictionary attributes such as name aliasing, parent–child dependencies and category membership.

Nevertheless, the complexity of these relationships is an indication of the richness of the metadata available through the dictionary approach, and the availability of well defined object representations simplifies the construction of well designed large-scale application frameworks, such as underpin the Protein Data Bank (Berman, Battistuz *et al.*, 2002) and Nucleic Acid Database (Berman, Westbrook *et al.*, 2002).

5.2.6.4. starlib

BioMagResBank (BMRB) is a repository for NMR spectroscopy data on proteins, peptides and nucleic acids at the University of Wisconsin – Madison (Ulrich *et al.*, 1989). For some time, NMR data sets have been exchanged within this environment using STAR Files; an NMRStar data dictionary to define the data names used for tagging NMR data is under development

(BioMagResBank, 2004). The *starlib* class library was developed at BMRB for handling NMRStar files, but its initial application to such files independently of the prototype data dictionary means that it is applicable to any STAR File. It does not provide a relational database paradigm (although this is a long-term goal). However, it does provide objects and methods suitable for searching and manipulating STAR data.

Table 5.2.6.1 lists the top-level classes used in *starlib*. *ASTnode* is a formal base class, providing the types and methods that can be used in other derived classes. *StarFileNode* is the root parent of all other objects contained in an in-memory representation of a STAR File; in practice it contains a single *StarListNode*, which is the list of all items contained in the file. *BlockNode* is a class which contains a partition of the STAR File: the class handles both data blocks and global blocks. Data-block names are stored in instances of the *HeadingNode* object, which also holds save-frame identification codes and is therefore useful for accessing named portions of the file.

DataNode is a virtual class representing the types of data objects handled by the library (accessed directly as *DataItemNode*, *DataLoopNode* and *SaveFrameNode*).

Looped data items are handled by a number of objects. *DataLoopNameListNode* is a list of lists of names in a loop. The first list of names is the list of names for the outermost loop, the second list of names is the list of names for the next nesting level and so on. *LoopNameListNode* is a list of tag names representing one single nesting level of a loop's definition. *LoopTableNode* is a table of rows in a *DataLoopNode* (not itemized in Table 5.2.6.1; it is an object representing a list of tag names and their associated values, a particular case of *DataNode*). *starlib* views a loop in a STAR file as a table of values, with each iteration of the loop being a row of the table. Each row of the table can have another table under it (another nesting level), but such tables are the same structure as the outermost one. Thus *LoopTableNode* stores a table at some arbitrary nesting level in the loop. A simple singly nested loop will have only one loop table node, but a multiply nested loop will have a whole tree of loop tables. *LoopRowNode* is a single row of values in a loop.

DataNameNode holds the name of a tag/value pair or a loop tag name. *DataValueNode* is the type that holds a single string value from the STAR file and the delimiter type that is used to quote it.

DataListNode and *SaveFrameListNode* store lists of data within higher-order data objects or save frames, and are internal classes rarely invoked directly by a programmer.

A number of observations may be made regarding this approach. Firstly, the objects can be mapped with reasonable fidelity to the high-level Backus–Naur form representation of STAR (Chapter 2.1). Secondly, it is computationally convenient to abstract common features into parent classes, so that, for example, individual data items, looped data and save frames are represented as child objects of the *DataNode* object, and not themselves as first-generation children of the base class. Thirdly, the handling of nested loops may be achieved in different ways; *starlib* has chosen a particular view that is perhaps well suited to relational data models.

As expected within a programming toolkit, *starlib* offers a large number of methods for retrieving STAR data values, adding new data items, extending or re-ordering list structures, and performing structural transformations of the in-memory data representation. Unlike the stand-alone *Star_Base* application, it does not guarantee that output data will be in a STAR-conformant format; and the programmer is left with the responsibility of validating transformed data at a low level.

Table 5.2.6.1. Object classes for manipulating STAR data in *starlib*

<i>ASTnode</i>	The base class from which all other classes are derived
<i>StarFileNode</i>	The STAR File object
<i>StarListNode</i>	List of items contained in the STAR File
<i>BlockNode</i>	A data or global block
<i>HeadingNode</i>	Labels for major STAR File components
<i>DataNode</i>	General class for data objects
<i>DataLoopNameListNode</i>	List of lists of names in a loop
<i>LoopNameListNode</i>	List of tag names representing one nesting level
<i>LoopTableNode</i>	Table of rows in a loop
<i>LoopRowNode</i>	Single row of values in a loop
<i>DataNameNode</i>	A data name
<i>DataValueNode</i>	A single string value
<i>DataListNode</i>	List of data within a higher-order data object
<i>SaveFrameListNode</i>	List of data items allowed in a save frame

Nevertheless, this is a substantial and important library which, as with *CIFOBJ*, has played an important role in the functioning of a major public data repository. Development of the class libraries continues, with a Java version now available.

5.2.6.5. StarDOM

A convenience of well designed object representations is that effective transformation between different data representations may be possible. The *StarDOM* package (Linge *et al.*, 1999) demonstrates a transformation from STAR Files to an XML representation, where the tree structure of a STAR File as interpreted in the *starlib* view above is mapped to a document object model (DOM; W3C, 2004). This approach is similar to *Jumbo*, mentioned above in Section 5.2.6.2.

A demonstration of *StarDOM* is the transformation of the complete set of NMR data files at BioMagResBank to XML. The resultant files can then be interrogated using the *XQL* query language (Robie *et al.*, 1998). In this example implementation, the target XML document type definition (DTD) includes a small number of XML elements matching the STAR objects *global* and *data block*, *save frame*, *list*, *data item*, *data name* and *data value*. Particular data names are recorded as values of the <NAME> element. The authors of the *StarDOM* package are considering an extension in which named data items map directly to separate XML elements; the goal is to develop an NMR-specific DTD that is isomorphous to the emerging NMRStar data dictionary.

References

- Berman, H. M., Battistuz, T., Bhat, T. N., Bluhm, W. F., Bourne, P. E., Burkhardt, K., Feng, Z., Gilliland, G. L., Iype, L., Jain, S., Fagan, P., Marvin, J., Padilla, D., Ravichandran, V., Schneider, B., Thanki, N., Weissig, H., Westbrook, J. D. & Zardecki, C. (2002). *The Protein Data Bank*. *Acta Cryst.* **D58**, 899–907.
- Berman, H. M., Westbrook, J., Feng, Z., Iype, L., Schneider, B. & Zardecki, C. (2002). *The Nucleic Acid Database*. *Acta Cryst.* **D58**, 889–898.
- BioMagResBank (2004). *NMR-STAR Dictionary*. Version 3.0. (Under development.) <http://www.bmrwisc.edu/dictionary/3.0.html/>.
- Chang, W. & Bourne, P. E. (1998). *CIF applications. IX. A new approach for representing and manipulating STAR files*. *J. Appl. Cryst.* **31**, 505–509.
- Knuth, D. E. (1973). *The art of computer programming*. Vol. 3, *Sorting and searching*, pp. 422–447. Reading, MA: Addison-Wesley.
- Linge, J. P., Nilges, M. & Ehrlich, L. (1999). *StarDOM: from STAR format to XML*. *J. Biomol. NMR*, **15**, 169–172.
- Murray-Rust, P. (1993). *Analysis of the DDL dictionary parsing problem*. *Proc. First Macromolecular Crystallographic Information File (CIF) Tools Workshop*, ch. 12. New York: Columbia University.