5.1. General considerations in programming CIF applications

By H. J. Bernstein

5.1.1. Introduction

There are many ways to create new 'CIF-aware' applications and to adapt existing applications to make them CIF-aware. This chapter reviews general considerations in programming CIF-aware applications, ranging from leaving an application CIF-unaware and relying on external filter utilities to do the job, through engineering an existing application to directly read and write CIFs, to writing a new CIF-aware application from scratch. The adaptation of applications to CIF does not happen in isolation. There are many other data representations and metadata frameworks relevant to crystallography. In Chapter 1.1, the CIF format was placed in the historical context of the development of data representation and metadata languages. In this chapter, we deal with that context from the perspective of software design.

The major issues in making an application CIF-aware are:

(1) Are CIFs to be read?
- (i) Are these CIFs produced externally?
- (ii) Does the organization of information required by the application conform to the organization of information specified in the relevant dictionaries?
- (iii) Is maximal performance in reading important?

(2) Are CIFs to be written?
- (i) Are these CIFs to be used externally?
- (ii) Does the organization of information used by the application internally conform to the organization of information specified by the relevant dictionaries?
- (iii) Is maximal performance in writing important?

Reading a CIF is a much more complex task than writing a CIF. Two equally valid CIF presentations of exactly the same information may be ordered differently. An application that reads a CIF must be prepared to accept tags and columns of tables in any order. The same order independence means that an application may write out tags and columns of tables in any convenient order, simplifying the design of CIF write logic.

When CIFs are only to be used internally, it is tempting to adjust the format to fit the application, *e.g.* by imposing order dependence. However, caution is advised if there is any possibility that such an application might eventually have to deal with CIFs coming from or going to other groups. In designing the CIF interface for an application, it is prudent to assume that the read logic will eventually have to deal with externally produced CIFs and that CIFs produced by the write logic will eventually be processed by software at other sites.

If performance is not a major issue, then it can be easy to make an application CIF-aware simply by the use of external filter programs. However, when performance *is* an issue, it becomes necessary to integrate the CIF reading and writing logic with the application. This can be done with reasonable efficiency by the use of existing CIF-aware libraries, but such libraries can impose a cost by their use of private internal data structures to hold the information from a CIF. Integrated design from scratch may be needed for maximal performance.

Creating or adapting software for CIF is an example of creating or adapting software for an agreed format, a format to be adhered to in the creation of multiple applications. There are different levels of 'agreement'. The agreement may apply to the representation of data, the representation of information about data ('metadata') or both (making a 'data framework'). The agreement may loosely specify the style of presentation of information or may specify details of presentation with great precision, or anything in between. The effort one needs to make in adapting an application to an agreed format depends to a large part on the level of agreement. If the agreed format is sufficiently detailed, one can comply strictly with the agreed format as a standard.

If one's goals are the widest possible interchange of data and the longest possible survival time of data in archives, it is important to achieve strict adherence to the agreed format as a standard. If one's goals are shorter-term and do not raise issues of interchange with independent groups, use of an agreed format as a checklist or style guide may help to avoid redundant effort. Even when one has longer-term goals, system requirements may not mesh with the agreed format and the development of new formats may be needed. CIF, as an agreed format, involves specification of metadata for a specific data format and of ontologies of domain-specific definitions that could be used not only in CIF but also in other formats. The use of an agreed format as a base can help to avoid redundant effort in this case as well. Within the domain of small-molecule crystallography, CIF has achieved its powerful impact on crystallographic publication and archiving by being used as a strict standard both for the data format and for definitions of terms. Within other domains or for certain applications other approaches may be appropriate. For example, it has proven productive to make use of CIF data names within XML (Bray *et al.*, 1998) formatted documents (Bernstein & Bernstein, 2002).

5.1.2. Background

There have been many efforts at creating agreed formats for data to be used in crystallography (see Chapter 1.1). We need to consider how software has been created to make use of such formats, especially software to make use of CIF.

Agreement on formats evolved from the earliest efforts at collaboration among research groups. Within crystallography, recognition of the need to use data formats as standards and to adapt applications to agreed formats, rather than to adapt formats to the caprices of particular applications or diffractometers or graphics engines, began in the late 1960s and early 1970s with the establishment of computerized data resources for the chemical and crystallographic community and the increasing availability of computer networks (Lykos, 1975). We will discuss three early data-resource efforts: the Cambridge Crystallographic Data Centre Structural Database File (CSD) (Allen *et al.*, 1973), the Brookhaven National Laboratory Protein Data Bank (PDB) (Bernstein *et al.*, 1977) and the NIH/EPA Chemical Information System (CIS) (Heller *et al.*, 1977). The differences and similarities among application development efforts related to these resources illustrate some of the issues

Affiliation: Herbert J. Bernstein, Department of Mathematics and Computer Science, Kramer Science Center, Dowling College, Idle Hour Blvd, Oakdale, NY 11769, USA.

references