

Data and text mining

A framework for scientific data modeling and automated software development

Rasmus H. Fogh^{1,†}, Wayne Boucher^{1,†}, Wim F. Vranken², Anne Pajon², Tim J. Stevens¹, T. N. Bhat³, John Westbrook⁴, John M. C. Ionides² and Ernest D. Laue^{1,*}

¹Department of Biochemistry, University of Cambridge, 80 Tennis Court Road, Cambridge, CB2 1GA, UK, ²MSD group, EMBL-EBI, European Bioinformatics Institute, Wellcome Trust Genome Campus, Hinxton, Cambridge, CB10 1SD, UK, ³Biotechnology Division (831), NIST, 100 Bureau Drive, Stop 8310, Gaithersburg, MD 20899–8314, USA and ⁴Department of Chemistry and Chemical Biology, Rutgers University, Rutgers, The State University of New Jersey, 610 Taylor Road, Piscataway, NJ 08854-087, USA

Received on October 19, 2004; revised and accepted on December 15, 2004

Advance Access publication December 21, 2004

ABSTRACT

Motivation: The lack of standards for storage and exchange of data is a serious hindrance for the large-scale data deposition, data mining and program interoperability that is becoming increasingly important in bioinformatics. The problem lies not only in defining and maintaining the standards, but also in convincing scientists and application programmers with a wide variety of backgrounds and interests to adhere to them.

Results: We present a UML-based programming framework for the modeling of data and the automated production of software to manipulate that data. Our approach allows one to make an abstract description of the structure of the data used in a particular scientific field and then use it to generate fully functional computer code for data access and input/output routines for data storage, together with accompanying documentation. This code can be generated simultaneously for different programming languages from a single model, together with, for example for format descriptions and I/O libraries XML and various relational databases. The framework is entirely general and could be applied in any subject area. We have used this approach to generate a data exchange standard for structural biology and analysis software for macromolecular NMR spectroscopy.

Availability: The framework is available under the GPL license, the data exchange standard with generated subroutine libraries under the LGPL license. Both may be found at <http://www.ccpn.ac.uk>; <http://sourceforge.net/projects/ccpn>

Contact: ccpn@mole.bio.cam.ac.uk

1 INTRODUCTION

1.1 Data and program fragmentation

In the field of macromolecular NMR spectroscopy—as with many other areas of science and commerce—numerous software packages are needed for a particular project and often these do not communicate well with each other. Data in different programs tend to be

organized in different ways and stored in proprietary file formats. As a result, it is difficult to switch between different programs for different parts of a project. Often data transfer is partial and imperfect, even between programs that are used for complementary tasks, and much effort is used converting data into different formats so that it can be used in different programs. As a result, and despite the fact that the data often exist in an electronic form, large amounts of data are never carried forward to the next program used. Because of the effort required to come back, collate and convert all this data into a standard format for deposition, much of it tends not to get deposited in databases. In the wake of the Human Genome Project, many new large-scale projects for gathering information and determining experimental data are being developed. The need for data interchange between programs and for data harvesting (Fig. 1)—the automated gathering of experimental data for deposition—is thus growing in many fields.

The interaction between neighboring scientific disciplines exacerbates the problem. Often different disciplines need to describe the same subject area, e.g. molecular topology or structure, in different ways for their own specific purposes. In the absence of a mechanism for sharing a common description, each discipline will come up with its own way of doing things, which will be subtly different in content and often totally different in implementation, severely reducing the possibilities for data exchange.

There is also another side to the problem of integrating different programs. Most programs have no facility for adding plug-ins; therefore in the absence of a standard for importing and exporting data, often there is no simple way to write program extensions efficiently. This again means that software development groups cannot make full use of existing software and are often forced to duplicate programs that are already available before they can add new capabilities.

1.2 Standard software and data standards

Where sufficient resources are available, these problems can be avoided by developing integrated software packages that provide all the functionality needed for a given class of problems, or that come with well-supported interfaces to ‘neighboring’ programs. But these solutions require a large (or a rich) customer base to provide the resources, as well as a consensus on precisely what the programs

*To whom correspondence should be addressed.

†The authors wish it to be known that, in their opinion, the first two authors should be regarded as joint First Authors.

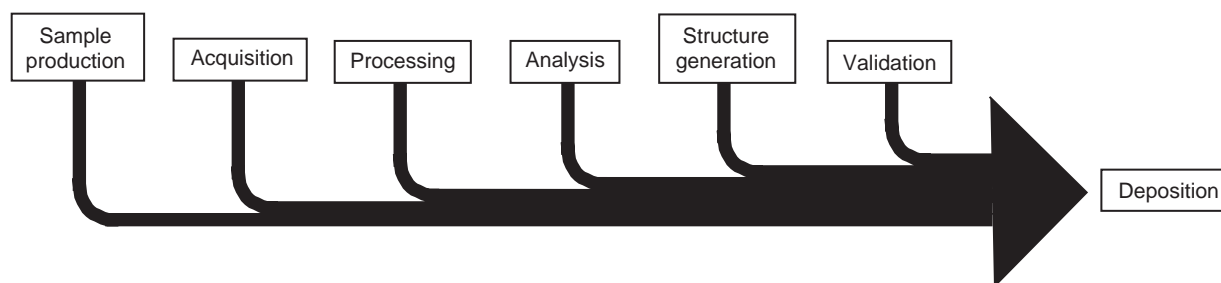


Fig. 1. Data harvesting in macromolecular structure determination. The output of each program is added to the data and carried along through successive steps prior to its being deposited in international databases.

should be doing. In scientific research where resources are limited, and where new methods are being developed continuously, solutions that rely on comprehensive well-established applications, supported by large teams of programmers distributed over the globe, are usually unrealistic.

The obvious solution to this ‘software Babel’ problem is to use an agreed data standard. A data standard provides a precise and well-defined way of describing the relevant data, and so allows programs to share and exchange them. There is no need for data conversion programs. All the relevant data are represented in a consistent way so that they can be ‘carried along’ through the course of a project, and any program that can read and write the data will be a functioning (if loosely integrated) extension of the existing software. These benefits follow from a data standard that reflects the current scientific understanding and which covers all the data needed for a given area, but to ensure long-term success more is required. As the science develops, and as new scientific disciplines share parts of the model, the data standard must change. For efficiency it must be possible to make these changes quickly, so that the standard does not hold back scientific progress, with a minimum of work, and without forcing the rewriting of programs conforming to the standard. For the standard to be widely adopted, it must overcome the natural reluctance of programmers to change the way they operate or accept the limits to their freedom of action. This requires enough support for the data standard in the form of subroutine libraries to make the change-over easy, and even attractive. Finally, it must be possible to maintain these software libraries and keep them synchronized with the data standard itself with a realistic (i.e. small) commitment of resources.

1.3 The MEta-MOdel Programming System framework

The Collaborative Computing Project for biomolecular NMR spectroscopy (CCPN) presents here a data modeling framework called MEMOPS (MEta-MOdel Programming System) that satisfies these constraints. MEMOPS provides a simple and efficient way of generating and maintaining constantly evolving data standards with powerful associated subroutine libraries in a number of programming languages simultaneously. The MEMOPS framework is completely general and could equally well be used for any other subject area.

2 SYSTEMS AND METHODS

2.1 Project strategy

The strategy adopted by the CCPN project was developed in a series of international meetings with representatives of the main deposition databases,

instrument manufacturers and research groups developing software for macromolecular NMR (Fogh *et al.*, 2002). The meetings decided that the data standard should have:

- (1) A formal specification to allow for automatic validation of data. The specification should be in Unified Modeling Language (UML), (Rumbaugh *et al.*, 1999), a computing industry standard for systems specification.
- (2) Content based on existing modeling work that had already been carried out in the field, notably by the BioMagResBank (<http://www.bmrb.wisc.edu>) and the RCSB (<http://www.rcsb.org>, Westbrook and Fitzgerald, 2003). The model should support not only final results for database deposition but also the intermediate information needed while working on a project.
- (3) An application program interface (API) rather than a data format. The choice of a single format was felt to be too limiting and too hard to maintain in the long term. Programs would interact with data only through the API, which would remain stable over time. Changes in the model, new storage formats, programming languages, etc. could then be taken care of by modifying the API implementation.
- (4) Storage format implementations for XML and SQL relational databases, as well as storage routines for existing deposition formats [PDB and mmCIF (Westbrook and Fitzgerald, 2003), and NMRSTAR (BioMagResBank, <http://www.bmrb.wisc.edu>)].
- (5) API implementations for several programming languages, e.g. Python, Java and C++.
- (6) Implementation as a single document to avoid potential synchronization problems between different versions.
- (7) All subroutine libraries, data format specifications, etc. to be, as a minimum, validated directly against the standard. Preferably they should be derived automatically from the standards document to guarantee synchronization and to make the entire set of standards, format specifications and API implementations easy to maintain.

The software organization that follows from this strategy is illustrated in Figure 2.

2.2 The Data Model

A Data Model is a description of the data for a particular subject area, how they are defined and organized and how they relate to one another. It includes the data items and their relationship, but is not adapted to any particular implementation. UML, being object-oriented, describes the model in terms of classes of objects, which in turn have attributes. In terms of relational databases, there is a rough correspondence between classes and tables, objects and rows, attributes and fields. Relations between objects are represented directly as associations in the object-oriented world; in the database world this is handled by fields serving as references to the rows of other tables (‘foreign keys’).

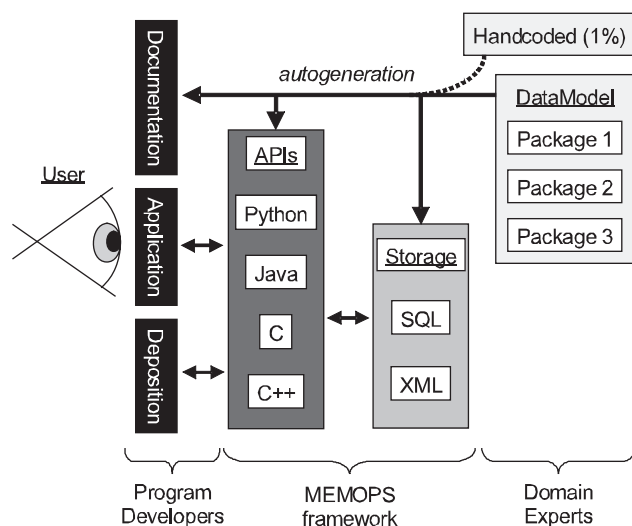


Fig. 2. Implementation of the CCPN project. Users interact with applications or deposition tools as is traditional, but software developers use the APIs to interact with the underlying data. APIs and their implementations, storage format descriptions, I/O routines and documentation are all generated automatically from the UML data model, to the extent of >99%. The APIs will remain stable over time even when the underlying data formats or data model change, thus insulating application programs from future changes.

As an example, Figure 3 shows a simplified part of the macromolecular Data Model. ChemComps (chemical components) describe the building blocks of molecules (e.g. Alanine), their topology and component atoms. Molecules describe the sequence of macromolecules, whereas MolSystems (molecular systems) describe a specific complex of interacting molecules (e.g. a protein dimer bound to a fragment of DNA). MolSystems, Molecules and ChemComps are all contained within a Project. A MolSystem, e.g. a protein dimer, in turn contains Chains, which contain Residues, which contain Atoms. A Chain is an instance of the Molecule it is linked to (as are Residues to MolResidues, and Atoms to ChemAtoms). Attributes store the properties of objects, e.g. Atom objects have names, Residues have sequence numbers and three-letter codes. All objects are part of a containment hierarchy rooted in the Project. The diagram in Figure 3 is not just an overview—this graphical UML notation has a precise meaning—it is actually a (view of a) precisely specified data model. For instance, it follows from this model that a given residue is contained within one particular chain, which in turn belongs to one molecular system. It also follows that a residue cannot be in more than one chain—even if there are two identical chains we would still be talking about two different residues. The part of the data model described here was originally developed starting from the mmCIF (Westbrook and Fitzgerald, 2003) and NMRSTAR (BioMagResBank, <http://www.bmrb.wisc.edu>) models. Further examples of features of the Data Model (residue templates) are described in the Supplementary information. A more complete description of the entire data model for NMR is given in (Vranken *et al.*, 2005).

Due to its size and complexity, the Data Model has been subdivided into packages, each containing a group of closely linked classes. The packages (shown in Fig. 4) organize the Data Model and, at the same time, the subroutine libraries generated from it and the files storing the actual data. This subdivision into packages allows loading and saving data one package at a time, so that an application dealing with sequence analysis, for example, can avoid loading code and data that are not relevant, e.g. data for structural analysis. In this way it becomes possible to model a series of neighboring areas of science so that applications for crystallography, NMR, structure analysis and bioinformatics can all use the same fundamental data model, and the same set of data files, without each application having to deal with a prohibitively

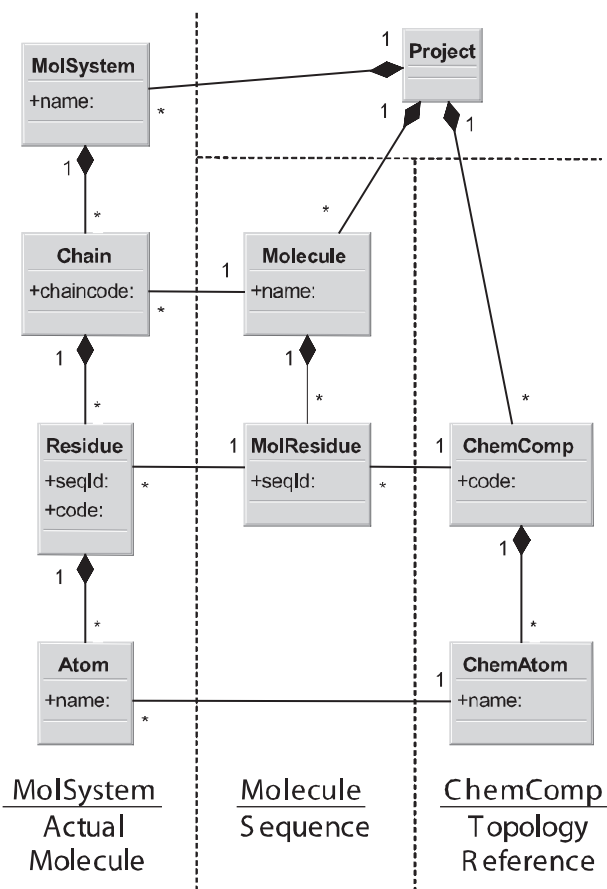


Fig. 3. A simplified part of the CCP macromolecular Data Model. Boxes show classes with their names (in bold) and attributes. Lines between classes are associations, with the filled diamond showing that the class is the container for the class at the other end. The '1' and '*' indicate, e.g. for the Atom-ChemAtom link, that each Atom must be associated with exactly one ChemAtom, while each ChemAtom can be associated with any number of Atoms. The MolSystem, Molecule and ChemComp are from three different packages in the overall Data Model. Structure determination projects, whether using NMR spectroscopy or X-ray crystallography, would make use of all the classes shown in the figure, whereas a sequence analysis program, for example, would ignore the MolSystem and the classes 'contained' in it.

large model or dataset. It also allows the independent development of different parts of the Data Model.

2.3 Automatic code generation

As illustrated in Figure 2, subroutines for data interaction (APIs), data storage and documentation are all generated automatically from the abstract data model. Very importantly, autogeneration guarantees that all of the generated documents are synchronized, offering important savings in time and effort for maintenance of the project.

The automatic code generation is a two-stage process. In the first stage the information describing the model is extracted from the UML modeling tool (ObjectDomain, a commercial program), transformed into a set of Python objects in memory and then written to a set of files. In the second stage these files are read to recreate a set of in-memory Python objects, which then form the basis for the various generating scripts. This approach decouples the generation process from the UML modeling tool, and allows the substitution of other tools at the price of changing only a single module of the generation software.

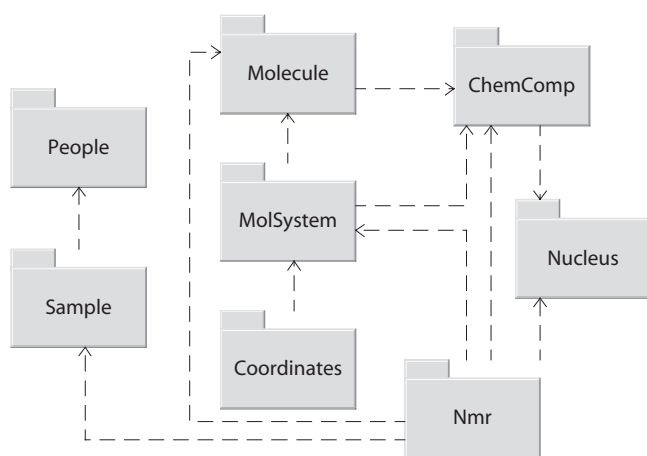


Fig. 4. Extract from the set of packages that make up the CCP Data Model. The arrows between packages describe dependencies between data packages. For instance, the NMR spectroscopy package depends on the Molecule package, which contains descriptions of sequence and covalent geometry. The effect is that programs that use the NMR package must also load the Molecule package (both code and data) whereas a sequence comparison program, for example, could load the Molecule package without having to interact with the NMR package. The division into packages allows different scientific fields to share a single package describing a common area of data.

The generation of Python or Java APIs, XML or SQL schemas, documentation, etc. is essentially a one-to-one mapping of the model. A class in the model will correspond to a Java or Python class, an XML element or an SQL table. The same name, or an automatic derivation of it, is used throughout, to avoid the need for special mapping files. Given the nature of the systems a one-to-one mapping is not, however, enough. XML requires extra elements for some attributes and links, relational databases require extra tables for many-to-many associations, etc. but in each case the extra code follows directly from the nature of the model without requiring (or allowing) extra input. There is of course an infinite number of ways of making Python APIs or XML schemas that correspond to a given data model. The goal of MEMOPS is in each case to derive one useful implementation in a simple and fully automatic way, rather than to make the process customizable by the application programmer or data model developer.

For an application programmer the impact of using the Data Model is determined mainly by the APIs. The quality and ease of use of the APIs is therefore extremely important for the wide application of our approach. Our APIs have been optimized for querying, for maintaining consistency in the presence of continuously changing data and for supporting multiple projects with multiple users using different approaches and techniques. Automatic code generation in itself reduces the potential for bugs and guarantees a consistent style across the entire body of code. In addition, the APIs have been designed to include a wide range of functionality. Comprehensive validity checking is incorporated in all operations that modify data, to ensure that the data remain in a consistent and legal state. Data loading is done automatically, and the API keeps track of which data packages are modifiable, or have been modified and thus require saving. There is also support for graphical user interfaces (GUIs) through a 'notifier' mechanism that allows 'update-screen' functions to be triggered automatically by changes in the data. The Python API, being the most mature, is presented as an example in the Supplementary information. A Java API is currently in alpha test, and C and/or C++ APIs are also planned. In addition, XML or SQL schemas can also be automatically generated.

The current API interacts with data stored in a mixture of XML files and local or remote databases. The price for this flexibility is that data must be loaded essentially one file at a time, which would hardly be a major problem,

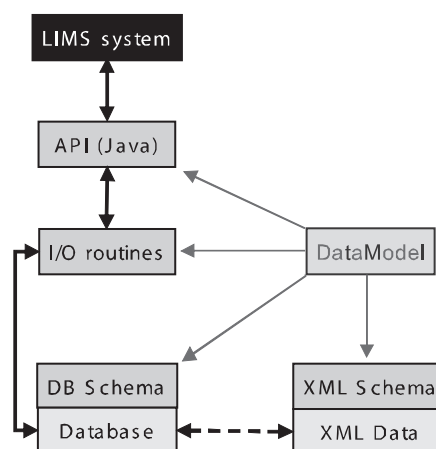


Fig. 5. Example of data flow and automatic code generation for the development of a LIMS implementation. Data are kept in a single database. The LIMS system interacts with its data through an API layer and an I/O routine layer, both autogenerated from the data model. XML is used for data export and interchange. Database and XML schemas are also generated from the model. The choice of Java for the API layer is specific to this example.

certainly for the common case where each project is worked by only one user at a time. Work is in progress on an alternative API implementation that will provide concurrency, security and fine-grained control for simultaneous, multiuser access, transaction control and roll-back, but this implementation depends on all the data being kept in a single database. Figure 5 outlines the way a specific application, in this example a Laboratory Information Management system (LIMS), could be developed based on the data model and its associated tools. An LIMS implementation would probably be based on an API interacting with a single underlying database. All data access would take place through the API in order to guarantee full data validity checking and to insulate the application from future changes in either the data model or database schema.

3 IMPLEMENTATION

3.1 Modeling with the MEMOPS framework

The MEMOPS framework allows data modeling to be separated from software implementation. Experts in a particular scientific domain can specify a data model in precise detail using UML, without having to consider storage formats, programming languages or other implementation issues. Once the model is made, APIs for different programming languages and implementations for different storage formats are generated automatically with a minimum of extra input, making the model available to application programmers with very different requirements. This feature will in itself favor the take-up of the model as a data standard.

The framework also supports the integration of models from different domains. The concept of a package, i.e. a self-contained description of data for a specific subdomain, allows different domains to share models of areas held in common. A domain expert can base his work on pre-existing packages without necessarily having to modify them or to force other users of these packages to accommodate his additions. This allows data to be shared between all domains that have an interest in a particular subdomain, and favors a situation where a number of data models for particular domains can be joined together in a single, wide-ranging model.

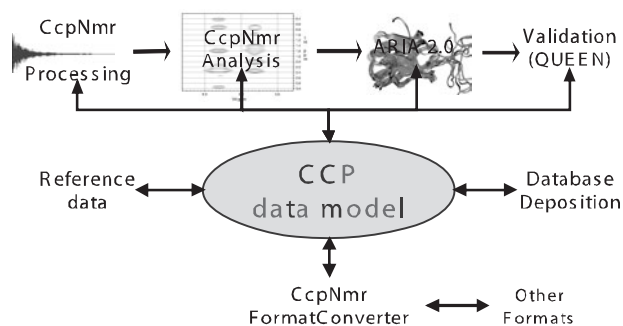


Fig. 6. Software suite for NMR structure generation. CcpNmr processing (in progress) will be based on the Azara processing package (<http://www.bio.cam.ac.uk/azara/>), while CcpNmr Analysis (W.F.Vranken *et al.*, submitted for publication) is a new analysis package for NMR. Both are based entirely on the Data Model APIs. The NMR validation program QUEEN (Nabuurs *et al.*, 2003) and the structure generation program ARIA (Nilges, 1995; Linge *et al.*, 2001) are fully integrated with the Data Model (the latter currently via conversion scripts). Data from other NMR programs can be accessed through the CcpNmr FormatConverter (W.F.Vranken *et al.*, submitted for publication). Reference data is derived from the Macromolecular Structure Database at EBI (<http://www.ebi.ac.uk/msd/>) (residue and ligand topologies) and the BioMagResBank (<http://www.bmrb.wisc.edu>) (NMR chemical shifts). Database deposition is carried out by writing data in the NMRSTAR (BioMagResBank, <http://www.bmrb.wisc.edu>) and mmCIF (Westbrook and Fitzgerald, 2003) formats.

In addition, the use of APIs (rather than data formats or models) as the invariant target for application programmers' efforts has a number of advantages for software integration and interoperability in a world where software technology and data models are continuously changing. The mere fact of programming against an API is enough to render changes in storage format transparent to the application being programmed. APIs can also be designed to be less tied to the precise detail of the underlying model than say a parser would be, as they represent a higher level of abstraction. This allows the API to protect applications that use it from having to modify their code even as the data model changes. Additions to the model are especially easy to handle, since the addition of new functions to an API does not interfere with the existing ones. Changes in names, or in which data are stored and which are calculated on the fly are also relatively unproblematic, and it will frequently be possible for the API to accommodate even more fundamental changes in the structure of a data model.

3.2 The NMR Data Model in use

Based on the CCPN NMR Data Model we are currently developing a complete suite of applications for NMR structural projects that illustrate the advantages of being able to exchange data with other programs. This development includes software for processing and analysis of NMR data, for calculation and validation of structures and for subsequent deposition of data in the public archives (Fig. 6).

In order to test and refine the Data Model and MEMOPS framework (as well as to provide a state of the art modern program) we have developed 'CcpNmr Analysis', a completely new interactive graphical analysis program for macromolecular NMR. This program has been based on the Python API implementation of the Data Model from the initial design stage. The development of this program has illustrated the significant advantages and time savings that result from

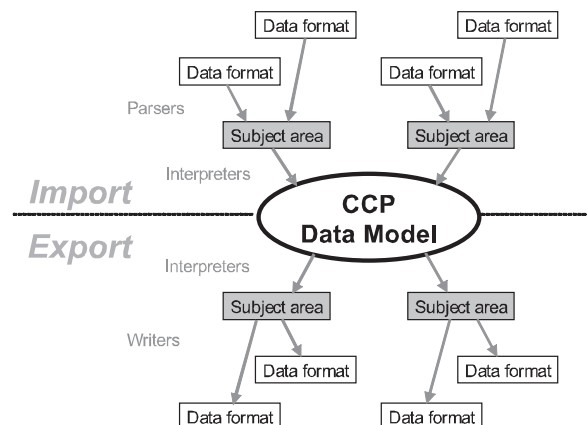


Fig. 7. The role of a central data model in format conversion. The parsers convert the information from files in particular data formats into temporary data structures separated by subject area. These are used by the interpreters to enter the data into the Data Model structure. On export, the same temporary data structures are created from the Data Model, and written out to the various format files.

the use of a well-supported data model. The automatic generation of all the code for data I/O and consistency checking results in very considerable time savings as this typically represents up to half of the (non-graphical) code for most software. In addition, it encourages the development of libraries of subroutines for highly modular code development, where each library can readily use all other existing libraries. It also allows efficient storage of data because different types of information can be readily stored in the most appropriate supported format: binary data, XML, SQL databases, etc. In fact, the data modeling framework was also used to model the graphical state of the program (window positions, colors, etc.), as this was the fastest way of generating the required code. In short, the Data Model greatly facilitates faster and easier development of new applications. CcpNmr Analysis is described in more detail in (Vranken *et al.*, 2005).

No software package, however good, can hope to provide optimal solutions to all the problems encountered in a particular field. In order to facilitate exchange of data with existing programs used by the NMR community, a series of connected format converters for macromolecular NMR and molecular structure data have been developed (Fig. 7). The CcpNmr FormatConverter program relies on the capacity of a data model for accommodating all the relevant data for a given scientific area. Data are first read into an intermediate in-memory representation using a format-specific reading routine. In some cases this requires user interaction, as the external data formats often contain less information than the Data Model, so that certain pieces of information may be missing. Once these problems have been dealt with, data are translated into an in-memory representation of the data model. This step provides a precise definition of the meaning of the data, independent of where they came from, after which it is easy to write the data directly to an arbitrary output format. Using the standard Data Model as a stepping stone implies that one can convert between all known formats while having to write (and maintain) only one conversion routine for each format rather than one for each pair of formats. In the short term, the FormatConverter provides a means of converting data into a CCP Data Model format. Of course, in the longer term, our aim is to convince other software development groups to read from and write to the Data Model in

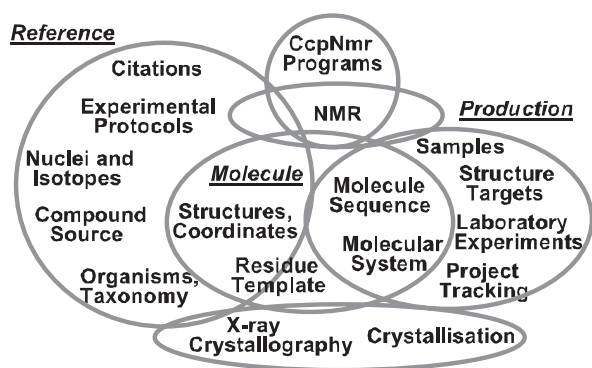


Fig. 8. Development of a UML data model for bioinformatics. The development of the data model for protein production (Pajon *et al.*, 2005) is being coordinated by the EBI as part of the e-HTPX project (<http://www.e-htpx.ac.uk/>), while that for X-ray crystallography is under discussion. The other areas have already been developed in the CCPN project.

their own programs. CcpNmr FormatConverter is also described in more detail in (Vranken *et al.*, 2005).

4 DISCUSSION

Our experience thus far is that the data exchange concept that we have developed works in practice. The applications currently being developed ‘understand one another’ and the Data Model is now being actively used both within the CCPN software suite for NMR spectroscopy and at the BioMagResBank (<http://www.bmrb.wisc.edu>) for converting existing deposited restraint lists to a standard IUPAC nomenclature. For more details see (Vranken *et al.*, 2005).

4.1 The Data Model

The different packages in the Data Model now cover many of the areas needed in bioinformatics and it can be directly extended to neighboring areas of knowledge. In fact, the development of an associated UML data model for protein production is already being coordinated by the EBI under the e-HTPX project (<http://www.e-htpx.ac.uk/>). This protein production data model (Pajon *et al.*, 2005) uses some of the packages provided by the CCPN project and has developed new packages specific to protein production (Fig. 8). The combination of data modeling carried out by the two groups now provides all the packages that are required from protein production through NMR structure determination to deposition. Preliminary efforts are under way to extend this project to X-ray crystallography and other structural methods. Using the CCPN framework one can then generate the APIs, XML and SQL schemas etc. that provide the basis for the required software applications. They also provide the basis for the development of the LIMS that play a critical role in the organization, tracking and co-ordination of data in large-scale genomics projects. One of the benefits of the unified data model that we propose is that different LIMS systems can be developed and used within a single large project—because the data is always organized in the same way.

4.2 Model driven architecture

The MEMOPS data modeling framework is related to a number of current trends in software engineering. Chief among these is the concept of a model driven architecture (MDA)

(<http://www.omg.org/mda/>) promoted by the Object Management Group (OMG) (<http://www.omg.org>). The MDA approach emphasizes the use of an abstract object-oriented model of the domain you are working on as the basis for generating software. This has the advantage of decoupling important parts of your work from particular implementations, and of allowing migration to new software technologies without losing all the effort invested in a project. A model driven architecture has a natural affinity with our concept of automatic code generation, although the two are not identical. The work of the OMG has also been important in providing a general approach to standardization, in being the source of UML, and in providing the MOF UML subset that has largely provided the basis for the metamodel used by MEMOPS.

4.3 Automatic code generation

Automatic code generation has also been presented before (Herrington, 2003), both in commercial UML modeling tools and in open source projects, such as Eclipse (an open source project supported by IBM—<http://www.eclipse.org>), and associated projects like the Kent Modeling Framework (KMF, <http://www.cs.kent.ac.uk/projects/kmf/>) and GME (<http://www.isis.vanderbilt.edu/projects/gme>) that are available as Eclipse plug-ins. There has, however, been a tendency for these tools to concentrate on partial code generation, i.e. to generate code that had to be completed by hand, and in general to operate with several layers of models (platform-independent, platform-specific, etc.) that could not be generated automatically from each other. This fits in with the general emphasis of these projects, of modeling a software application rather than a field of data, and of concentrating on producing a working application in a single language. In scientific areas related to macromolecular NMR, autogeneration has also been used e.g. for the OpenMMS toolkit (<http://openmms.sdsc.edu/>) and in the FILM server (<http://deposit.pdb.org/mmCIF/FILM/index.html>). Both efforts are based on the OMG macromolecular structure standard (http://www.omg.org/technology/documents/formal/macro_molecular.htm) and use CORBA technology (<http://www.corba.org/>) to allow interaction with macromolecular structure data.

4.4 Ontologies

The data modeling of the CCPN project is related to the development of ontologies that are being carried out in the field of biology such as the Gene Ontology Consortium (<http://www.geneontology.org/>) and the Open Biological Ontologies project under SourceForge (<http://obo.sourceforge.net/>). Indeed, depending on your definition, it could be argued that the CCPN Data Model is an ontology. However, in the biological arena, ontologies have been focused more on providing a controlled and precisely defined vocabulary, without the detailed specification of cardinalities and constraints that are an essential feature of the CCPN data modeling work. As a corollary to this, ontologies do not generally serve as precise specifications for the generation of software libraries. This fact notwithstanding, ontologies are used as important components of some recent systems (see e.g. <http://www.gmod.org/>, <http://www.flymine.org>), and there are ontology modeling programs with some of the same capabilities as CCPN, such as Protege (<http://protege.stanford.edu>).

4.5 MEMOPS

The MEMOPS code generation framework has been designed for data modeling and fully automatic code generation without reference

to a final application. The particular features of MEMOPS arise from this context. Limiting ourselves to data modeling we have been able to simplify the problem we are dealing with enough that we can achieve automatic code generation from a UML model with <1% of the final subroutine libraries having to be hand-coded. The need for supporting multiple applications with different requirements has driven the decision to provide API implementations for multiple programming languages and storage implementations for several storage formats. Our goal of using MEMOPS for generating a data standard for an entire scientific field (specifically macromolecular NMR), has led us to emphasize validity and consistency checking in our APIs, and to optimize for constantly changing rather than static data. Finally, the need to encourage the taking-up of the standard has led us to provide highly functionalized API implementations to make the product easy to use and attractive to software developers. The result is a fully generic data modeling framework that could be used in any area of knowledge.

ACKNOWLEDGEMENTS

We thank the BBSRC who funded the CCPN project, and the EU 'Quality of Life and Management of Living Resources' program (contracts QLRI-CT-2001-00015 'TEMBLOR' and QLK2-CT-2000-01313 'NMRQUAL') for additional financial support. We also thank Astra-Zeneca, Genentech, Dupont Pharma (now

Bristol-Myers-Squibb) and GlaxoSmithKline for supporting the CCPN project by providing funding for the meetings and workshops.

REFERENCES

- Fogh,R.H., Ionides,J.M.C., Ulrich,E., Boucher,W., Vranken,W.F., Linge,J.P., Habeck,M., Rieping,W., Bhat,T.N., Westbrook,J. *et al.* (2002) The CCPN project: an interim report on a data model for the NMR community, *Nat. Struct. Biol.*, **9**, 416–418.
- Herrington,J. (2003) *Code Generation in Action*. Manning Publications, Greenwich, Connecticut.
- Linge,J.P., O'Donoghue,S.I. and Nilges,M. (2001) Assigning ambiguous NOEs with ARIA, *Methods Enzymol.*, **339**, 71–90.
- Nabuurs,S.B., Spronk,C.A.E.M., Krieger,E., Maassen,H., Vriend,G. and Vuister,G.W. (2003) Quantitative evaluation of experimental NMR restraints, *J. Am. Chem. Soc.*, **125**, 12026–12034.
- Nilges,M. (1995) Calculation of protein structures with ambiguous distance restraints. Automated assignment of ambiguous NOE crosspeaks and disulphide connectivities. *J. Mol. Biol.*, **245**, 645–660.
- Pajon,A., Ionides,J., Diprose,J., Fillon,J., Fogh,R.H., Ashton,A.W., Berman,H., Boucher,W., Cygler,M., Deleury,E. *et al.* (2005) Design of a data model for developing laboratory information management and analysis systems for protein production. *Proteins*, **58**, 278–284.
- Rumbaugh,J., Booch,G. and Jacobson,I. (1999) *Unified Modeling Language Reference Manual*. Addison Wesley Longman, Amsterdam.
- Vranken,W.F., Boucher,W., Stevens,T.J., Fogh,R.H., Pajon,A., Llinas,M., Ulrich,E.L., Markley,J.L., Ionides,J. and Laue,E.D. (2005) The CCPN data model for NMR spectroscopy: development of a software pipeline. *Proteins*, in press.
- Westbrook,J. and Fitzgerald,P.M. (2003) The PDB Format, mmCIF Formats, and Other Data Formats. In Bourne,P.E. and Weissig,H. (eds) *Structural Bioinformatics*. John Wiley & Sons, Hoboken, NJ, pp. 161–179.