BIOINFORMATICS REVIEW

Vol. 17 no. 2 2001
Pages 115–125

XML, bioinformatics and data integration

Frédéric Achard [1,3], Guy Vaysseix [1,2] and Emmanuel Barillot [1,2,*]

[1]CRI Infobiogen, 523 place des terrasses de l'agora, 91000 Evry, France and
[2]Généthon, 1 bis rue de l'Internationale, 91000 Evry, France

Received on June 9, 2000; revised on September 20, 2000; accepted on October 31, 2000

ABSTRACT

Motivation: The eXtensible Markup Language (XML) is an emerging standard for structuring documents, notably for the World Wide Web. In this paper, the authors present XML and examine its use as a data language for bioinformatics. In particular, XML is compared to other languages, and some of the potential uses of XML in bioinformatics applications are presented. The authors propose to adopt XML for data interchange between databases and other sources of data. Finally the discussion is illustrated by a test case of a pedigree data model in XML.

Contact: Emmanuel.Barillot@infobiogen.fr

INTRODUCTION

For those who joined the research community recently it is hard to imagine how genome/biology research was conducted before the advent of the Web (World Wide Web) and the HyperText Markup Language (HTML). Indeed HTML has been, and still is, overwhelmingly successful. In biology, it is used for database browsing, data publishing, data gathering, data submission or data analysis. The main reasons behind this success are:

- it is very user-friendly: browsing the Web is almost instinctive and requires very minimal training time;
- it is very easy to learn the language†: the knowledge of a few self-explanatory tag names and the understanding of a very simple syntax are enough to write surprisingly good looking HTML pages;
- it is very easy to program: the CGI mechanisms allow fast hooking of programs to an HTML page.

However, HTML has a number of well-known limitations, largely because it is dedicated to human browsing. It does a good job for rendering simple documents, but complex structured documents are difficult to model because HTML does not provide arbitrary structure. For example, HTML defines six levels of headings, from <H1> to <H6>. What if a document has more than six levels of headings? Workarounds are usually easy to implement but they hinder rigorous modeling of the document. Another weakness of HTML is that it allows for tags to occur in a completely arbitrary order. Nothing prevents a Web designer using <H1> as a sub-header for <H2> sections (besides common sense...). The structure of such documents is hard to decipher because one is expecting <H2> to be a sub-section of <H1>.

Moreover, HTML says nothing about the semantics of a document. For example the <blockquote> tag announces a list, but no difference can be made in HTML between a list of theaters in Roma and a list of metallo-proteins. The so-called HTML META tags only indicate vaguely the content of an HTML page, are not widely used and lack standardization (Lawrence and Giles, 1999). As a result, it is difficult to programatically extract information from HTML pages. Search engines such as AltaVista (http://www.altavista.com/) or Infoseek (http://www.infoseek.com/) produce a number of false-positive hits for some types of queries because HTML is not capable of capturing the semantics of a document.

The idea of the eXtensible Markup Language (XML) is to overcome the limitations of HTML. XML is derived from the Standard Generalized Markup Language (SGML), the international standard for defining descriptions of the structure and content of different types of electronic documents. SGML allows you to define your own markup language, that consists of your own tags. It is therefore a meta-language to design markup languages. For example, the Request For Comment 1886 (http://www.ietf.org/rfc/rfc1866.txt) described formally for the first time HTML 2.0 (to distinguish it from the previous informal specifications) as an SGML application. It would be overly complex (and not very entertaining) to go into the differences between SGML and XML. Suffice to say that XML could be described as a Web dedicated, lightweight SGML, without the more complex and less used parts of SGML.

*To whom correspondence should be addressed.
[3]Present address: Cereon Genomics, 45 Sidney St, Cambridge, MA 02139, USA.
† See a 10-min guide for newcomers to HTML: http://www.w3.org/MarkUp/Guide/.

XML is said to *'open the way to a new Internet that would be easier to search and exploit'* (Mackenzie, 1998). Like SGML, XML allows the definition of a set of tags to be applied to one or many documents. These tags generally identify different types of elements in the document, with the possibility for recursion and referencing. The element identification can be used for presentation, in the same way HTML is used to present a document via a Web browser. XML gives the means for defining strongly structured documents so computer programs can easily navigate through them and access relevant pieces of information. XML is often presented as a smart successor of HTML, although it is clear that, because of its many qualities, the ease of use being the most prominent one, HTML will be around for a long time.

It is important to keep in mind that, even though XML was designed for the Web, its goal is to describe the structure of a document, which may or may not be on the Web. For example, XML is often used as a data exchange language (as in GAME, see Section XML: who is using it?), independently from any Web related application.

XML: HOW DOES IT WORK?

The World Wide Web Consortium (W3C) has supervised the specifications of XML (http://www.w3.org/XML/) since its inception in 1996. In short, XML documents consist of elements, that are textual data structured by tags. An element consists of a start/end tag pair, some optional attributes defined as key/value pairs, and the data between the two tags, for example:

```
<roman_theater year='0120' type='outdoor'
          location='Roma'>Colosseo</roman_theater>
```

is an XML element.

A Document Type Definition (commonly called a DTD) describes the structure of the elements of an XML document. The element declarations have the form of a tree corresponding to the document structure: elements may have child elements, data (text) or may be empty. One can also mix elements and text. For example, the DTD describing the XML element above would be (the text between '$<!--$' and '$-->$' are commented out):

```
<!-- element definition as a character string        -->
<!-- PCDATA stands for Parseable Character DATA)     -->
<!ELEMENT roman_theater (#PCDATA)>

<!-- attributes attached to the roman_theater element -->
<!-- CDATA are strings,                               -->
<!--  #REQUIRED and #IMPLIED constraint the attribute -->
<!ATTLIST roman_theater
    year  CDATA     #REQUIRED
    type  CDATA     #IMPLIED
    location CDATA  #REQUIRED>
```

A more complex example of XML data with its DTD file is given in Appendix An example: using XML for pedigree data.

An XML document is either:

- well-formed: obeying the syntax of XML;
- or valid: a well-formed XML document conforming to the logical structure as defined separately in a DTD.

The XML Linking Language (http://www.w3.org/TR/NOTE-xlink-req/) (XLL), currently under development, introduces a standard linking model for XML. It will provide linking capabilities that go beyond the hyperlinks provided by HTML. Some of the key features of XLL are: (i) semantics of links; (ii) extended links to define relationships between more than two documents; (iii) extended pointers to give direct access to part of a structured document; and (iv) bi-directional links.

With XML also comes the XML Stylesheet Language (XSL), currently under development, which specifies how an XML document should be rendered. This information is to be used for example by a browser. The look and feel of a Web site can be updated by simply modifying the XSL file(s). XSL provides for text only rendering of documents, but tools already exist that extend the capabilities to graphical rendering (Ciancarini *et al.*, 1999).

One of the ten design goals of XML, as described in the reference document (http://www.w3.org/TR/REC-xml) for XML 1.0, is: *'It shall be easy to write programs which process XML documents'*. For that purpose, programmers use an XML processor which is responsible for making the content of the document available to the application. There are two types of XML processor: event-driven and tree-driven. SAX (Simple API for XML (http://www.megginson.com/SAX/index.html)) is an example of an event-driven API. An XML document is read as a data stream and markup is interpreted as encountered. The Document Object Model (DOM) is an example of a tree-driven API. The whole document is accessible to the application as a tree, facilitating search and content manipulation. It has an object design that assumes the use of an object oriented programming or scripting language. Like XML, DOM is being developed by W3C (http://www.w3.org/TR/REC-DOM-Level-1/).

Most of the well known software companies have announced major commitments into the XML technology. We list below, in no particular order, some products that were either used by the authors, or that we found worth mentioning.

- Microsoft with Internet Explorer 5 released a browser which does a good job at displaying XML files. It includes a validating parser and some XSL support. The default stylesheet for displaying XML files shows the structure of the document in a straightforward way, but this is easy to configure with a customed stylesheet,

for example an XSL file. A number of XML resources and documentations are available at the Microsoft developer site (http://www.microsoft.com/xml/default.asp).

- Expat (http://www.jclark.com/xml/expat.html) is an XML 1.0 parser written in C. It aims to be fully conforming, but it is currently not a validating XML processor. This is the supporting parser for Netscape (Mozilla 5) and Perl 5 module (see below).
- Perl 5 has a number of XML libraries and tools (http://www.perlxml.com/modules/perl-xml-modules.html), notably XML::Parser and XML::DOM. We found that it was easy to manipulate XML documents by means of those libraries, with the facilities that Perl provides for text manipulation.
- Java Project X is the code name for the experimental XML programming package released by Sun (http://java.sun.com/xml). It includes an XML parser with optional validation. This package is currently used by one of the authors with good results.
- IBM (http://www.alphaworks.ibm.com) also released an experimental XML programming package, including a Java parser and a collection of tools, notably an XML editor called Xeena.
- The Apache XML project (http://xml.apache.org) goal is to provide commercial-quality standards-based XML solutions that are developed in an open and cooperative fashion. For example, IBM contributed to the project with its XML parser.

For further details, we refer the reader to the abundant literature on XML (e.g. Bradley, 1998) and to the Web site maintained by Robin Cover for a very complete list of URLs on XML (http://www.oasis-open.org/cover/xml.html).

XML: WHO IS USING IT?

Many commercial and academic actors are now adopting XML as a standard for their data management, if not for their Web site. There is no doubt that within a few years it will be as widespread as HTML is today.

The chemistry and the mathematics community benefited from early developments in XML. The main goal is to facilitate the writing and the exchange of scientific information by the adoption of a common language in XML, namely: CML and MathML. The DTDs defining those two languages were agreed on by members from each community, and are now freely available.

- The Chemical Markup Language (http://www.xml-cml.org/) (CML, see Murray-Rust and Rzepa, 1999) aims at managing chemical information (atomic, molecular and crystallographic information, compounds, structures, publications ...), and has several associated tools such as the popular Jumbo browser (http://www.venus.co.uk/omf/cml).
- The MathML (http://www.w3.org/Math/) specification has been adopted by the W3C last year as a DTD of about one hundred elements for the representation of mathematical formulas. MathML has also its specialized browser: WebEQ (http://www.webeq.com).

More recently, there have been similar attempts in biology where XML is used as a general framework for annotating sequence data.

- The Bioinformatic Sequence Markup Language (http://www.visualgenomics.com/products/index.html) (BSML): the DTD is aimed at representing DNA, RNA, protein sequences and their graphic properties. We found the structuration of the information to be similar to the one used in the EMBL/GenBank/DDBJ databases (http://www.ebi.ac.uk/embl.html; http://www.ncbi.nlm.nih.gov; http://www.ddbj.nig.ac.jp).
- The BIOpolymer Markup Language (BioML (http://www.proteometrics.com/BIOML/)): the approach is somewhat different to BSMLs approach. Quoting the authors (Fenyö, 1999), BioMLs goal is to *'allow the expression of complex annotation for protein and nucleotide sequence information. BioML was designed to mimic the hierarchical structure of a living organism.'* It results in a DTD which achieves some level of data integration in combining information from different sources (e.g. nucleotide and protein sequence).

The business logic of the two companies (Visual Genomics Inc. for BSML, Proteometrics, LLC for BioML) that developed those languages are similar: they propose a public domain standard for the encoding of biological information, while they sell software developed around their language, such as browsers, data integration and data management tools.

- The taxonomic markup language consists of a DTD for the description of taxonomic relationships between organisms (Gilmour, 2000).
- XML is also adopted by the gene ontology consortium (http://www.geneontology.org), whose objective is to provide controlled vocabularies for the description of the molecular functions, biological processes and cellular locations of gene products (Gene Ontology Consortium, 2000).

Additionally, XML and domain specific DTDs are starting to be used to facilitate the transfer and the publication of biological information. GAME (http://www.bioxml.org/Projects/game) was first developed to exchange data between members of the Berkeley Drosophilia Genome Project (Lewis *et al.*, 1998) and Celera. It is now being extended as a common language for annotating biosequence 'features'.

A second example is the BlastXML DTD (http://workingobjects.com/blastxml/blastxml.dtd) that aims to model NCBI Blast output. Around this DTD, WorkingObject (http://workingobjects.com/) developed two components: BlastXML-SDK, a Java framework for creating and processing BlastXML documents, and the Blast Parsing Framework for processing NCBI Blast native and web-server generated reports. Finally, the European Bioinformatics Institute has also announced that they will use XML for the storage of DNA array data (Editorial, 1999).

For technical discussions on the use of XML in bioinformatics, we recommend subscribing to the BioXML mailing list (http://www.bioxml.org/). The following Web sites will also list some of the XML resources available for each languages: bioperl, biojava and biopython (http://www.bioperl.org/; http://www.biojava.org/; http://www.biopython.org/).

DATA MANAGEMENT IN BIOINFORMATICS

Today a bioinformatics information system typically deals with large data sets reaching a total volume of about one terabyte[‡]. The problem of managing (that is, modeling, storing and querying) this information is not solved satisfactorily, although it has been recognized as a key component of today's genome/biology research (Robbins, 1994). It is now starting to be a bottleneck in many biological projects (Reichhardt, 1999). One terabyte is indeed a very large volume of data but nothing extraordinary. For example particle physics deals with petabytes of data and the exabyte is envisioned at the European Laboratory for particle physics (See http://www.objectivity.com/Releases/CERN.htm). In fact the difficulties in dealing with the bioinformatics data come more from some of its idiosyncrasies than from its quantity:

- data are complex to model. There are many different types of data presenting numerous relationships;
- new types of data emerge regularly: complete sequences of genomes, microarrays, interaction maps of proteins, proteomics .... Not only must these new types of data be modeled properly but they also modify our perception of the old types of data. In particular new relationships appear between concepts that had previously no or only weak links. This means that the semantics of the whole must be updated and that sources of information that were formerly independent must be integrated;
- data analysis generates new data that also have to be modeled and integrated;
- raw data must be archived, because scientists often need to return to them to confirm computer generated results;
- granularity of data is rather fine, and the terabyte of bioinformatics data consists of a large number of objects. For example, at Infobiogen, our object-oriented database (Viara *et al.*, 1999) containing the four million entries of the EMBL/GenBank/DDBJ database includes more than 200 million objects;
- data are updated very frequently, accessed intensively and exchanged very often by researchers on the Internet;
- all kinds of users (biologists, programmers, database managers ...) need to issue complex queries.

These peculiarities of bioinformatics data have already been outlined (Robbins, 1994; Letovsky, 1995; Karp, 1995; Davidson *et al.*, 1995; Markowitz and Ritter, 1995). In summary, bioinformatics has to deal with exponentially growing sets of highly inter-related and rapidly evolving types of data that are heavily used by humans and computers. A bioinformatics language should therefore offer power and scalability at run time and should be based on a flexible and expressive model.

In addition it is important to consider the following technical issues when handling data in bioinformatics.

- The volume of data grows exponentially, doubling in less than two years for the EMBL/GenBank/DDBJ database for instance. This growth has been sustained since the beginning of molecular biology 20 years ago and everything indicates that this will continue for some time. This poses serious problems in terms of data management and requires very scalable solutions (Letovsky, 1995).
- Data are disseminated in a myriad of different databases (Discala *et al.*, 1999) that are duplicated in several repositories.
- These databases have heterogeneous formats, see for example (Markowitz and Ritter, 1995). Most of them are accessible under the form of flat files structured

[‡] 1 terabyte = 1000 gigabytes, 1 petabyte = 1000 terabytes, 1 exabyte = 1000 petabytes.

by a field/value convention. They are weakly interconnected with indexing systems such as SRS (Etzold *et al.*, 1996) or HTML links.

These considerations are relevant for interconnection-oriented languages.

XML IN BIOINFORMATICS: THE PROS AND CONS

It is clear that XML has interesting features addressing some of the problems of the bioinformatics community exposed above.

- XML is highly flexible: it is simple to modify a DTD. Since it is interpreted, adding new elements or attributes may not require modification of the XML data; the XML and DTD files are human readable and thus can be easily edited by people with only few computer skills. Updating a data model is, therefore, straightforward (at least from a technical point of view). This capability was named *content scalability* in the terminology of Letovsky (1995).
- XML is Internet-oriented and has very rich capabilities for linking data; this can be used for interconnecting databases. For example cross-references between databases (such as the DR field in Swiss-Prot (http://www.ebi.ac.uk/sprot) or EMBL/GenBank/DDBJ) could be replaced by a so-called `ENTITY` with a direct access to the information under a structured format; one could also use the XLL one-to-many links with $n > 2$ if several cross-references refer to the same biological object. This would avoid the long process of cross-indexing the database entries at each new release.
- XML provides an open framework for defining standard specifications. This is an important point because bioinformatics clearly lacks standardization. For example, querying on multiple molecular biology databases could be greatly facilitated if each database would offer an XML view of their content.

On the other hand, XML has some weaknesses.

- The overhead of a text based format in data parsing, storage and transmission needs to be evaluated before adopting XML as a general solution. However, a text format means that the source code can be read and edited with any text editor. We believe a binary format like ASN.1 never really took off because of the need for specialized tools to see or manipulate the data.
- It is not clear whether XML satisfactorily addresses the problems of *technological scalability*. Indeed if XML data are stored in flat files, queries on XML files will not scale because XML in itself does not provide scalable facilities such as indexing or data clustering. This means that parsing should be done on the fly which leads to poor performances. One solution could be to have query optimizations done externally for example using a DataBase Management System (DBMS).
- In addition, the expressiveness of the XML data model would probably not be sufficient for molecular biology. Indeed, as mentioned before, the semantics of biological data is very rich and requires a very expressive data model: object—oriented technologies are often employed such as in (Barillot *et al.*, 1999a; Médigue *et al.*, 1999; Baker *et al.*, 1999) and are thought to be the only satisfactory solution (Aberer, 1994). Although doable, modeling of biological data with XML is limited because:
  - — XML has no inheritance mechanism, no methods on objects (although external programs can be hooked);
  - — the concept of relationship can be mimicked through lazy referencing with `ID` and `IDREF(S)` but does not exist as such;
  - — only not null, unicity and cardinality constraints can be specified; there is no symmetry, no elaborated constraints, no triggers;
  - — the basic types are mainly strings (`CDATA`, `NMTOKEN(S)`), enumerations and various references to XML elements, XML documents or non-XML data types such as TIFF images; and
  - — XML has no support for numerical values, tables and matrices.

  The XML Schema language will address many of these concerns. It provides XML with a much better data modeling capability. Quoted from the public working draft of XML Schema 1.0: *'Requirements are for constraints on how the component parts of an application fit together, the document structure, attributes, data-typing, and so on. The XML Schema Working Group is addressing means for defining the structure, content and semantics of XML documents.'* (http://www.w3.org/TR/xmlschema-0/).

At this point the question to be answered is whether the pros prevail over the cons. To provide the reader with a bigger picture, the next section presents a comparison of XML with the alternative solutions.

XML VERSUS OTHER SOLUTIONS

In this section, XML is compared to some of the most popular solutions that are used for the management and exchange of bioinformatics data.

- Flat files structured by a field/value convention, such as the flat file libraries from EMBL/GenBank/DDBJ or Swiss-Prot.
- Abstract Syntax Notation One (ASN.1) (Steedman, 1993).
- The Common Object Request Broker Architecture (CORBA) (Siegel, 1996; Achard and Barillot, 1997; Seetharaman, 1998; Siegel, 1998; Vinoski, 1998).
- The Java Remote Method Invocations (Java RMI) (http://java.sun.com/products/jdk/rmi/).
- OODBMS (Object–Oriented DBMS) (Cattell, 1996).

It may seem a little odd to compare such different things. However, the fact is that all approaches are used in bioinformatics for the same purpose of data management: for defining data structures, capturing a semantic, exchanging or distributing data. They have all been used as the foundation of a data integration system, for example: with SRS (Etzold *et al.*, 1996) for flat files, at the National Center for Biotechnology Infomation (http://www.ncbi.nlm.nih.gov/) for ASN.1, at the European Bioinformatics Institute for CORBA (http://corba.ebi.ac.uk) and Java RMI (Möller *et al.*, 1999), and at Infobiogen (Barillot *et al.*, 1999b) for OODBMS.

- Field/value based flat files are very commonly used in bioinformatics. Of course this is a very limited solution, because it lacks referencing, typed values, vocabulary control, constraints .... Often fields are ambiguous and their content is contextual: one has to refer to the user manual, if not to a human expert, to understand fully the semantics. This hinders considerably their use by programs without human interaction. Moreover, access concurrency can only be controlled at the file level, which is a very coarse granularity.
- ASN.1 is heavily used at the National Center for Biological Information (http://www.ncbi.nlm.nih.gov) as a format for exporting the GenBank data (Benson *et al.*, 1999). ASN.1 can be seen as a means for exchanging binary data with a description of its structure. Its data model offers a wide variety of basic types with the possibility of aggregation. It is very compact. Of course it is not as convivial as XML since the data and its structure are binary and can only be read by parsers. Since ASN.1 files convey the description of their structure, it offers some flexibility: the client side does not necessarily need to know in advance the structure of data. However, ASN.1 is not really scalable and there is no support for queries. Again, access concurrency is manageable at the file level only.
- CORBA has already been used in many bioinformatics projects (Achard and Barillot, 1997; Hu *et al.*, 1998; Barillot *et al.*, 1999a,b; Rodriguez-Tomé and Lijnzaad, 1999). CORBA is a standard that provides an intermediary object-oriented layer that handles access to the data. The interface between the clients and the objects on the server is described in an Interface Definition Language (IDL) which isolates the former from the latter and offers language independence (client and server parts can be written in different languages, such as C++, Java, SmallTalk ...). A standard protocol, the Internet Inter-ORB Protocol ensures a platform independence, and allows interaction on the Internet between clients and servers on heterogeneous architectures. CORBA also offers a variety of services such as yellow pages, lifecycle, trading .... The interface definition is also stored in an Interface Repository which can be accessed by clients and used to discover on the fly the object definition. Of course, developing a CORBA environment is a heavy task that requires highly skilled computer scientists.
- Java RMI has been proposed by the Javasoft division of Sun microsystems. It provides a mechanism to invoke methods on Java objects, that may reside on a local or a remote server. Both the client and the server must be written in Java. The interface between server objects and clients is specified in the Java language, unlike CORBA which has a dedicated Interface Definition Language (IDL). The invocation appears local to the programmer. Java RMI offers the rich data model of Java. Like CORBA, its scalability depends greatly on the network bandwidth. Java RMI has already been used in bioinformatics, for example (Möller *et al.*, 1999; Saqi *et al.*, 1999).
- OODBMS have the advantage of offering a very rich data model well suited to biology (Aberer, 1994; Médigue *et al.*, 1999). They are designed to be highly scalable: they offer indexing, object clustering, query optimization .... Their data definition and query languages have been standardized (Cattell, 1996), although not all the OODBMS have fully implemented the standard query language yet. OODBMS also guarantee security, concurrent accesses, integrity, consistency and reliability. They offer Application Programming Interfaces in different languages. Of course using them supposes a higher computer science background than for XML.

All these considerations are summarized in the Table 1.

It is clear from this synoptic representation that XML has some weaknesses that match the OODBMS strengths. As mentioned earlier these weaknesses could be overcome by the joint use of OODBMS and XML. This has already

Table 1. Summary of comparison of different alternatives to XML. Each one is rated with one to four stars for different criteria: the higher the number of stars, the better the solution with regard to the criteria. The scores represent a subjective evaluation, that are based on our personal experience. Details are given in the text

	XML	Field/value	ASN.1	CORBA	Java RMI	OODBMS
Model expressiveness	**	*	***	***	***	****
Constraints	**	*	*	**	***	****
Self-descriptive	yes	no	yes	yes	yes	yes
Query language	soon[a]	no	no	soon[b]	no	yes
Flexibility	****	*	***	***	***	****
Simplicity	****	****	***	*	**	**
Scalability	**	*	**	***	***	****
Interoperability	****	*	**	****	****	***

[a] Two proposals for an XML query language have been issued: XQL (Microsoft, http://www.w3.org/TandS/QL/QL98/pp/xql.html) and XML-QL (AT&T, INRIA, Universities of Pennsylvania and Washington, http://www.w3.org/TR/NOTE-xml-ql/).
[b] This is a CORBA service but it is not yet implemented.

been implemented by some OODBMS (e.g. http://www.poet.com/products/cms/white_papers/xml/index.html). The idea is to use OODBMS where they are the best, that is data storing and retrieval, and XML where it is excellent, that is, as the interface language. Queries on the database would then return XML formatted results that could then be exchanged between users, databases or displayed by adequate XML browsers.

The use of XML as an intermediate medium would be really efficient only if all databases share common or very similar DTDs. Whatever language is used, it is always difficult to find an agreement on a common semantics, and when one is found, it is often revised. However, XML would be an excellent candidate for this role because of its flexibility.

CONCLUSION

XML is a promising standard that will undoubtedly impact positively on the bioinformatics community. Although the expressiveness of its data model is limited and it poses scalability issues, its flexibility, its simplicity and its interconnection capabilities make it an excellent candidate as a language for data exchange. It is anticipated that XML will be used in informatics as a universal hub for database interchange. We suggest adopting this powerful standard in bioinformatics for the same purpose, in combination with the OODBMS technology to which XML is complementary.

As has already happened in chemistry and mathematics, we stress the necessity for the database managers to define common DTDs. Indeed semantics does not spring magically from XML, but from a standard on which the community agrees. This task is already being undertaken in some domains of bioinformatics (Barillot *et al.*, 1999a; Gene Ontology Consortium, 2000) as it is crucial for the future of biology, notably for interconnecting databases. Although it is too early to speculate on the outcome of these initiatives, a key to success certainly lies on the promotion made by consortium with a big momentum, such as the Object Management Group (Barillot *et al.*, 1999a) or the Gene Ontology Consortium (Gene Ontology Consortium, 2000).

We also propose to replace field/value based flat files by XML Interchange Data Dumps (XIDD) and to base cross-references on the rich link model of XML. This would give to the programmers a uniform access to the data both from a syntactic and a semantic perspective. The cost of generating such XIDD would be very low and it would save tedious and often sub-optimal parsing by use of standard and generic parsers.

Data outputs of analysis programs are another important source of information in bioinformatics. These results need to be stored for reference; at the same time they are often used as input for further analysis. Here again we recommend the use of XML for describing the output of the analysis programs, similarly to BlastXML (see above).

Of course, XML is not completely mature yet, and several related functionalities have not been implemented yet (e.g. XQL) or are still being drafted (e.g. XSL), but they should be finalized soon and will reinforce the XML possibilities. Therefore we believe that in the very near future, XML will be ubiquitous in the bioinformatics community.

APPENDIX

An example: using XML for pedigree data

Why using XML for pedigree data? Pedigree data are a good test case for the use of XML in bioinformatics for several reasons:

- pedigree are not very complex to model, and the limited expressiveness of XML should not hinder their representation in this language;
- they are often shared between several groups collabo-

rating on the genetic dissection of a trait. The network orientation of XML will be useful here;

- they are manipulated by clinicians, epidemiologists and other people who do not necessarily have highly skilled computer scientists at their disposal and therefore prefer simple, intuitive languages;
- the format most commonly used for pedigree exchanged is difficult to read for a human as illustrated in Figure 5;
- various epidemiological information may be attached to the pedigree data, and their types are potentially infinite, because they largely depend on the trait under study: any risk factor, blood pressure, size, weight, smoking habits, enzyme dosage ....

The choice of pedigree is also the result of our experience in a project aiming at developing a collaborative environment for pedigree data management: CoPE (Brun-Samarcq *et al.*, 1999).

Example of a DTD. We have designed a DTD for pedigree data (http://www.infobiogen.fr/services/CoPE/XML/pedigree.dtd) that takes inspiration from the data schema of CoPE Brun-Samarcq *et al.* (1999) and of Bryant (1998). Figures 1 and 2 give a textual version and a tree representation of this DTD. Pedigree data typically consist of a set of pedigrees, each pedigree being a group of relatives that have been diagnosed for a given trait (e.g. a disease). Of course, epidemiologists and geneticists need to know the relationships between persons (parents/children, sibs, mates). But they are also interested in some other characteristics that may be correlated to the occurence of the trait under study and that could be interpreted as a risk factor (age, sex, clinical observations, and other epidemiological data that depend on the trait: smoking habit, blood pressure...).

The root of the DTD tree structure is the element `pedigree_set` which is defined as a sequence of one or more (`+`) `pedigree`(s) and has two optional (`#IMPLIED`) attributes, its `title` and the `trait` under study, both strings of characters (`CDATA`).

A `pedigree` is then declared as a sequence of one or more `person`(s); `pedigrees` have an optional attribute `title` and a mandatory (`#REQUIRED`) identifier `id` whose unicity within an XML document is enforced by the fact that its type is `ID`.

The third level of the tree structure is `person`, which has as child element zero or more (`*`) `epid_data` and as attributes a mandatory identifier `id`, a name `name`, a reference (`IDREF`) to the `pedigree` he/she belongs to, to his/her father and mother, a sex to be chosen in the enumerated list (`male | female | unknown`) with `unknown` as default value and a status as to the `trait` under study.

The element `epid_data` is defined as one of (like a union in C) the elements `qualitative_data` and `quantitative_data`; `epid_data` refers to any epidemiological data that could be correlated to the occurence

```
<!ELEMENT  pedigree_set  (pedigree+)>
<!ATTLIST pedigree_set
  title  CDATA  #REQUIRED
  trait  CDATA  #IMPLIED
>

<!ELEMENT  pedigree  (person+)>
<!ATTLIST  pedigree
  title  CDATA  #IMPLIED
  id  ID  #REQUIRED
>

<!ELEMENT  person  ( epid_data)* >
<!ATTLIST  person
  id  ID  #REQUIRED
  name  CDATA
  pedigree  IDREF  #REQUIRED
  father  IDREF  #IMPLIED
  mother  IDREF  #IMPLIED
  sex   ( male  | female  |  unknown )  'unknown'
  status  (  affected  |  unaffected |  unknown  )  #REQUIRED
  birth_year CDATA
>

<!ELEMENT  epid_data  ( qualitative_data | quantitative_data ) >
<!ATTLIST  epid_data
  name  CDATA  #REQUIRED
>

<!ELEMENT qualitative_data  (#PCDATA)>

<!ELEMENT quantitative_data (#PCDATA)>
```

Fig. 1. An example of a DTD for pedigrees: the file pedigree.dtd.

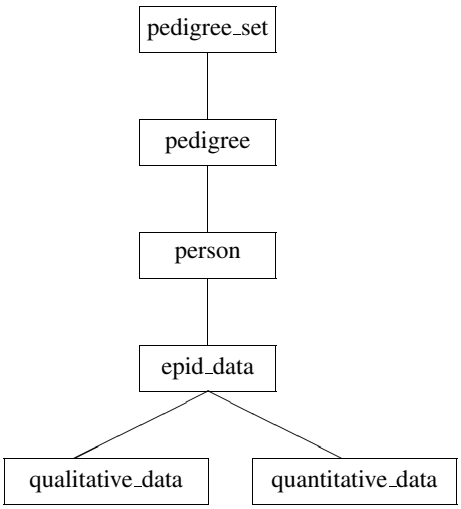

Fig. 2. The tree structure of the pedigree DTD of Figure 1.

```
class Pedigree_set {
  set <Pedigree *> pedigrees (notnull, card > 0);
  string title (notnull, unique);
  string trait;
};

class Pedigree {
  set <Person *> persons (notnull, card > 1, inverse<Person:pedigree>);
  string title (notnull);
};

enum Sex {
  male;
  female;
  unknown;
};

enum Status {
  affected;
  unaffected;
  undetermined;
};

class Person {
  string name;
  Pedigree * pedigree (notnull, inverse<Pedigree:persons>);
  Person * father;
  Person * mother;
  Sex sex = unknown;
  Status status = undetermined;
  int birth_year;
  Epid_data epid_data[];
};

class Epid_data {
  string name (notnull);
};

class Qualitative_data extends Epid_data {
  string value;
};

class Quantitative_data extends Epid_data {
  float value;
};
```

Fig. 3. An example of a object schema (ODL) for pedigrees.

of the trait, as the examples given in the previous section. Both `qualitative_data` and `quantitative_data` have no attribute but may contain free text to describe the values measured for each individual. This example of `epid_data` illustrates clearly that XML has a limited expressiveness: here inheritance and handling of numerical values would improve the data modeling greatly.

Example of an object schema. Figure 3 represents an OODBMS schema for pedigrees in Object Definition Language (ODL). Classes replace the XML elements, and the superiority of expressiveness of ODL versus XML is attested by the better treatment of numerical data (such as the attribute `birth_year` in `Person`, which is now an integer instead of a string of characters) or by the better modeling of the different types of epidemiological data through an inheritance mechanism, instead of a union of subelements in XML. Also in ODL no explicit object identifier is needed in the classes `Pedigree` and

```
<?xml  version="1.0"?>

<!DOCTYPE  document  SYSTEM  "pedigree.dtd">

<document title="An example of pedigree" trait="music genius">

    <pedigree title="Bach family" id="p1">
        <person id="1" name="Johann Sebastian" pedigree="p1"
         sex="male" status="affected" birth_year="1685">
           <epid_data name = "pitch">
               <qualitative_data> absolute pitch
               </qualitative_data>
           </epid_data>
        </person>
        <person id="2" name="Maria Barbara" pedigree="p1"
         sex="female" status="unknown">
        </person>
        <person id="3" name="Catharina Dorothea" pedigree="p1"
         father = "Johann Sebastian" mother = "Maria Barbara"
         sex="female" status="unknown">
        </person>
        <person id="4" name="Wilhelm Friedemann" pedigree="p1"
         father = "Johann Sebastian" mother = "Maria Barbara"
         sex="male" status="unknown">
        </person>
        <person id="5" name="Carl Philip Emmanuel" pedigree="p1"
         father = "Johann Sebastian" mother = "Maria Barbara"
         sex="male" status="affected">
        </person>
        <person id="6" name="Johann Gottfried Bernhard" pedigree="p1"
         father = "Johann Sebastian" mother = "Maria Barbara"
         sex="male" status="unknown">
        </person>
    </pedigree>
</document>
```

Fig. 4. An example of pedigree data in XML.

```
1  1  0  0  1  1
1  2  0  0  2  0
1  3  1  2  2  0
1  4  1  2  1  0
1  5  1  2  1  1
1  6  1  2  1  0
```

Fig. 5. Same pedigree data as in Figure 4 now in linkage format.

`Person` (as declared with `ID id` in the DTD): the object identifiers are assigned and managed automatically by the OODBMS.

Example of a pedigree in XML. Figure 4 shows an example of a pedigree in XML which describes the first marriage of Johann Sebastian Bach. This XML document conforms to the DTD exposed above (`DOCTYPE` declaration). The trait under study is the musical genius and a qualitative epidemiological data has been collected for one person: the absolute pitch.

Figure 5 shows the same pedigree data as in Figure 4, now in the linkage format (Lathrop *et al.*, 1985). It illustrates perfectly how XML can improve data structuration and readability.

Example of a query in XQL. Here we simply give some examples of queries in XQL, a specification of an XML Query Language already mentioned above. The syntax is declarative and straightforward :

```
# retrieve all persons in pedigree
pedigree/person

# retrieve the name of all persons
person/@name

# retrieve all persons that have a name
person[name]

# retrieve all pedigrees that contain
# some persons with epid_data
pedigree[person/epid_data]

# retrieve all persons with absolute
# pitch and born before 1700.
person[epid_data/qualitative_data!text() =
'absolute pitch' $and$ birth_date $lt$ 1700]
```

ACKNOWLEDGEMENTS

We thank our colleagues Laurence Samarcq, Éric Viara and Frédéric Guyon for their help in this work. This work was supported in part by the European Union contract BIO4-CT98-0030.

REFERENCES

Aberer,K. (1994) The use of object-oriented data models in biomolecular databases. *Conference on Object-Oriented Computing in the Natural Sciences*. Heidelberg, Germany, pp. 3–13.

Achard,F. and Barillot,E. (1997) Ubiquitous distributed objects with CORBA. In Altman,R., Dunker,K., Hunter,L. and Klein,T. (eds), *Pacific Symposium on Biocomputing '97*. World Scientific, Singapore, pp. 39–50.

Baker,P.G., Goble,C.A., Paton,S.B.N.W., Stevens,R. and Brass,A. (1999) An ontology for bioinformatics applications. *Bioinformatics*, **15**, 510–520.

Barillot,E., Leser,U., Lijnzaad,P., Cussat-Blanc,C., Jungfer,K., Guyon,F., Vaysseix,G., Helgesen,C. and Rodriguez-Tomé,P. (1999a) A proposal for a CORBA interface for genome maps. *Bioinformatics*, **15**, 157–169.

Barillot,E., Pook,S., Guyon,F., Cussat-Blanc,C., Viara,E. and Vaysseix,G. (1999b) The HuGeMap database: interconnection and visualisation of human genome maps. *Nucleic Acids Res.*, **27**, 119–122.

Benson,D.A., Boguski,M.S., Lipman,D.J., Ostell,J., Ouellette,B.F.F., Rapp,B.A. and Wheeler,D.L. (1999) GenBank. *Nucleic Acids Res.*, **27**, 12–17.

Bradley,N. (1998) *The XML Companion*. Addison-Wesley, Reading, MA.

Brun-Samarcq,L., Gallina,S., Philippi,A., Demenais,F., Vaysseix,G. and Barillot,E. (1999) Cope: a collaborative pedigree drawing environment. *Bioinformatics*, **15**, 345–346.

Bryant,S.P. (1998) *Managing Pedigree and Genotype Data*. Academic Press, New York, pp. 49–75.

Cattell,R.G.G. (1996) *The Object Database Standard : ODMG–93*. Morgan Kaufman, San Francisco.

Ciancarini,P., Vitali,F. and Mascolo,C. (1999) Managing complex documents over the WWW: a case study for XML. *IEEE Trans. Knowl. Data Eng.*, **11**, 629–638.

Davidson,S.B., Overton,C. and Buneman,P. (1995) Challenges in integrating biological data sources. *J. Comput. Biol.*, **2**, 557–572.

Discala,C., Ninnin,M., Achard,F., Barillot,E. and Vaysseix,G. (1999) DBcat: a catalog of biological databases. *Nucleic Acids Res.*, **27**, 10–11.

Editorial (1999) Array data go public. *Nature Genet.*, **22**, 211–212.

Etzold,T., Ulyanov,A. and Argos,P. (1996) SRS: Information retrieval system for molecular biology data banks. *Methods in Enzymology*. Vol. 266, Academic Press, New York, pp. 114–128.

Fenyö,D. (1999) The biopolymer markup language. *Bioinformatics*, **15**, 339–340.

Gene Ontology Consortium (2000) Gene ontology: tool for the unification of biology. *Nature Genet.*, **25**, 25–29.

Gilmour,R. (2000) Taxonomic markup language: applying XML to systematic data. *Bioinformatics*, **16**, 406–407.

Hu,J., Mungall,C., Nicholson,D. and Archibald,A. (1998) Design and implementation of a CORBA-based genome mapping system prototype. *Bioinformatics*, **14**, 112–120.

Karp,P.D. (1995) A strategy for database interoperation. *J. Comput. Biol.*, **2**, 573–586.

Lathrop,G.M., Lalouel,J.M., Julier,C. and Ott,J. (1985) Multilocus linkage analysis in humans: detection of linkage and estimation of recombination. *Am. J. Hum. Genet.*, **37**, 482–498.

Lawrence,S. and Giles,C.L. (1999) Accessibility of information on the web. *Nature*, **400**, 107–109.

Letovsky,S.I. (1995) Beyond the information maze. *J. Comput. Biol.*, **2**, 539–546.

Lewis,S.E., Harris,N.L., Helt,G.A., Misra,S. and Rubin,G.M. (1998) An intelligent system for interpretation of sequence alignment results. *The Institute for Genomic Research: Second Annual Conference on Computational Genomics*. Reston, Virginia, pp. 9.

Mackenzie,D. (1998) New language could meld the web into a seamless database. *Science*, **280**, 1840–1841.

Markowitz,V.M. and Ritter,O. (1995) Characterizing heterogeneous molecular biology database systems. *J. Comput. Biol.*, **2**, 547–556.

Médigue,C., Rechenmann,F., Danchin,A. and Viari,A. (1999) Imagene: an integrated computer environment for sequence annotation and analysis. *Bioinformatics*, **15**, 2–15.

Möller,S., Leser,U., Fleischmann,W. and Apweiler,R. (1999) DITtoTrEMBL: a distributed approach to high-quality automated protein sequence annotation. *Bioinformatics*, **15**, 219–227.

Murray-Rust,P. and Rzepa,H.S. (1999) Chemical markup, XML, and the Worldwide Web. 1. Basic principles. *J. Chem. Inf. Comput. Sci.*, **39**, 928–942.

Reichhardt,T. (1999) It's sink or swim as a tidal wave of data approaches. *Nature*, **399**, 517–520.

Robbins,R.J. (1994) Report of the invitational DOE workshop on genome informatics, 26–27 April 1993, Baltimore, Maryland. genome informatics I: community databases. *J. Comput. Biol.*, **1**, 173–190.

Rodriguez-Tomé,P. and Lijnzaad,P. (1999) The radiation hybrid database. *Nucleic Acids Res.*, **27**, 115–118.

Saqi,M.A.S., Wild,D.L. and Hartshorn,M.J. (1999) Protein Analyst—a distributed object environment for protein sequence and structure analysis. *Bioinformatics*, **15**, 521–522.

Seetharaman,K. (1998) The CORBA Connection. *Communi. ACM*, **41**, 34–36.

Siegel,J. (1996) *CORBA, Fundamentals and Programming*. Wiley.

Siegel,J. (1998) OMG overview: CORBA and the OMA in enterprise computing. *Communi. ACM*, **41**, 37–43.

Steedman,D. (1993) *ASN.1 The Tutorial and Reference*. Technology Appraisals, Twickenham, UK.

Viara,E., Barillot,E. and Vaysseix,G. (1999) The EyeDB OODBMS. *International Database Engineering and Applications Symposium*, pp. 390–402. IEEE Publications

Vinoski,S. (1998) New features for CORBA 3.0. *Communi. ACM*, **41**, 44–52.