Towards Ubiquitous Bio-Information Computing: Data Protocols, Middleware, and Web Services for Heterogeneous Biological Information Integration and Retrieval

Pengyu Hong
Department of Statistics
Harvard University
hong@stat.harvard.edu

Sheng Zhong
Department of Biostatistics
Harvard University
szhong@hsph.harvard.edu

Wing H. Wong
Department of Statistics and
Department of Biostatistics
Harvard University
wwong@hsph.harvard.edu

Abstract

Biological information computing is rapidly advancing from homogeneous data computation to large-scale heterogeneous data computation. However, the development of data specification protocols, software middleware, and Web services, which support large-scale heterogeneous data exchange, integration, and computation, generally falls behind data expansion rates and bioinformatics demands. The Ubiquitous Bio-Information Computing (UBIC^2) project aims to disseminate software packages to assist the development of heterogeneous bio-information computing applications that are interoperable and may run distributedly. UBIC^2 lays down the software architecture for integrating, retrieving, and manipulating heterogeneous biological information so that data behave like being stored in a unified database. UBIC^2 programming library implements the software architecture and provides application programming interfaces (APIs) to facilitate the development of heterogeneous bio-information computing applications. To achieve interoperability, UBIC^2 Web services use XML-based data communication means, which allow distributed applications to consume heterogeneous bio-information regardless of platforms. The documents and software package of UBIC^2 are available at http://www.ubic2.org.

Keywords: Data Integration, Middleware, Interoperability, Ubiquitous Bio-Information Computing.

1 Introduction

Along with the emergence of various high throughput biological experimental techniques, researchers have generated and keep generating large amounts of diverse biological data, such as sequence data, gene expression data, phenotype data, protein interaction data, etc. Each type of biological information alone may only provide a partial and often noisy view of the underlying biological systems and require different processing methods. Systematically utilizing heterogeneous information could provide deeper and more comprehensive understanding of the underlying biological facts. It has been shown that biology studies could benefit more from large-scale heterogeneous bio-information computing. For example, Hanisch *et al.* proposed a method for coupled analysis of biological network information and gene expression data [13]. Conlon *et al.* proposed to combine DNA sequences and gene expression profiles for regulatory motif discovery [6]. Stuart *et al*. used multiple species' gene sequences and gene expression data to discover conserved genetic modules [37].

However, there are substantial barriers to carrying out large-scale heterogeneous bio-information computing. First, data are widely distributed in many remote databases. Currently, there are more than 500 databases related to molecular biology [11]. Second, while all databases allow users to retrieve information manually via their web sites, automating information retrieval from those remote databases are non-programmable or require non-trivial programming. Third, there has been a great diversity of data specification conventions. Each database defines its own data formats, which are often not semantically sound. In addition, the data formats may change from time to time. As the results, researchers need to write their own parsers to interpret downloaded data and keep updating their parsers to catch up with the changes of the remote databases. Fourth, to compute bio-information, researchers need to write codes to load data into computer memory as programmable structured elements. Programming libraries for common data operations can be developed and shared to

Proceedings of the Fourth IEEE Symposium on Bioinformatics and Bioengineering (BIBE'04)
0-7695-2173-8/04 $ 20.00 © 2004 IEEE

avoiding duplicate efforts. The bioinformatics community has developed many open-source programming libraries. Nevertheless, such libraries are developed passively instead of actively and coordinately with the development data specification standards.

To industrialize the development of large-scale heterogeneous bio-information computing applications/systems, unified endeavors should be devoted to developing semantic bio-information specification protocols, software middleware, and interoperable computing applications.

2 Backgrounds

2.1 Data specification protocols

XML (the eXtensible Markup Language) is one of the main of the international efforts to promote the evolution and interoperability of World Wide Web. XML provides a mechanism to describe and exchange structure-rich data in computer understandable ways on the Internet. Biological database research community has realized the value of XML and been working on formatting biological data semantically via XML DTD (document-type definition) or XML Schema. Outstanding achievements include Distributed Annotation System (DAS) [9] for decentralized sequence annotation, XEMBL [39] for distributing EMBL data in XML format, RNAML [40] for exchanging RNA information, Systems Biology Markup Language [25] for representing biochemical reaction networks, and Microarray and Gene Expression Markup Language for microarray data annotation and exchange [34]. Such efforts are often specialized for certain types of data. They excel in data formation but do little for integrating heterogeneous databases.

2.2 Heterogeneous data integration

Works on database middleware for large-scale integration of heterogeneous biological data mainly take the following two approaches: database federation and data warehousing.

A federation approach builds a middle layer on tope of a collection of distributed databases and provides a query facility for the client applications to access remote data. Examples of database federation approaches include BioKleisli [7], TINet [10], DiscoveryLink [12], SEMEDA [28], etc. The federated approach has the advantages. Distributed databases appear as an integrated one to users. It allows users to access up-to-date distributed data at run-time. On the other hand, the performance of distributed data retrieval at run-time depends mainly on the reliability and speed of the underlying network connectivity, which often turns out to be the bottle neck. Moreover, as pointed earlier, most of the current biological databases only allow users to manually retrieve information via their web sites. Retrieved information is returned as a web page. Although the retrieval procedure can be automated, it requires non-trivial programming to parse HTML pages, and the programs are vulnerable to the format change of the web sites of the databases.

A data warehousing approach periodically batch downloads data from the ftp sites of remote databases, then extracts, re-arranges, and manages data locally. Since warehousing maintains a centralized local data, it is more efficient for repeatedly retrieving a large amount of heterogeneous data. Usually, different programs should be written to serve different data sources. However, most of the programs follow a similar procedure. An inheritable and expandable template can be designed and implemented to reduce the cost of programming. Examples of warehousing systems include TAMBIS [31], SLAD [33], BioMolQuest [4], InterPro [30], JXP4BIGI [26], SRS [41], SOURCE [8], AnnBuilder [42], ChipInfo [43], etc.

Most of the above systems use mature database management systems. On one hand, mature database management systems make information integration and retrieval easier and more efficient. On the other hand, the applications built on top of these database systems are tightened to the underlying database management systems, and hence lose certain degree of mobility.

2.3 Application development

There are some excellent efforts for developing software middleware to automate common bioinformatics tasks and facilitate the development of bioinformatics applications in different programming languages. Existing open-source programming toolkits including Bioperl [35], Biopython [5], Biojava [16], Bioruby [17], and BioConductor [15]. However, they provide limited supports for heterogeneous bio-information integration, data exchange, and interoperability between/among applications and databases.

2.4 Interoperability

Currently, biological databases are mainly designed to be used by human instead of machines. The main interfaces provided by the databases are web pages. Many of the programming libraries described in Section 2.3 provide methods to invoke certain computation on the remote databases, fetch the results which are embedded in a web page, parse the HTML source to extract information, and return the reformatted data. However, any format changes of the web pages of the databases could force developers to change their programs immediately. Stein advocated Web services as a solution to achieve interoperability among online databases [36]. Applications can be developed to weave together XML Web services from a variety of sources in the same way to create a distributed application using many components.

3 UBIC^2 Overview

The vision of UBIC^2 is to see the creation and thriving of thin bio-computing servers, which are more active, have more mobility, and assist every step in the progress of a biology research project in a nearly unnoticeable way.

UBIC^2 comprehensively considers data specification standards, data integration, application development, and interoperability. UBIC^2 adopts some existing XML data specifications and initiates new ones when needed. All data are managed in XML formats. UBIC^2 software architecture designs a pipeline that fetches, encapsulates, and manages data from different sources, so that the data behave like being stored in a unified database. The data integration approach of UBIC^2 is a mixture of data warehousing and database federation. Some remote databases, which are frequently used, have local snapshots. Some databases, which are too large to be managed locally, are integrated via federation. The programs can decide to warehouse which databases in run time. UBIC^2 does not use any existing database management systems. It provides simple data management mechanism that are efficient enough. Hence, the software and data of UBIC^2 are light and easy to disseminate.

Based on the UBIC^2 software architecture, the UBIC^2 programming library is designed and implemented using object-oriented technology. A set of generic classes is designed and implemented for tasks that share similar or common procedures. The programming library is inheritable and expandable. It is easy for developers to extend the library to cover new databases and develop new ways to manipulate data. The programming library enables developers to quickly develop bioinformatics applications and Web services. UBIC^2 also initiates an XML schema to standardize the communications between distributed bio-information computing elements and data servers through Web services.

In the following sections, we will present the current status of the UBIC^2 project in details.

4 XML

4.1 Data specifications

Many biological data have already been specified in XML formats, which are defined by XML DTDs or schemas such as DAS XML specification [9], XEMBL [39], NCBI Document Type Definitions [19], and Swiss-Prot XML Schema [23]. Every XML DTD or schema defines a data encapsulation format, specifically data names, types and lengths, document structure and semantics. As the first step to integrate heterogeneous biological data, UBIC^2 collects, categorizes, and organizes DTDs and schemas into a unified data specification set.

UBIC^2 XML data specification has four categories: Molecule annotation information, Sequence, Ontology, and Literature. Molecule information schemas cover annotation information of genes and proteins. Sequence schemas define the formats to represent nucleotide sequences and amino acid sequences. Currently, the Ontology category only includes specification of Gene Ontology [1]. We adopt PubMed DTD [20] for literature data specification. We designed XML schemas for gene annotation information (name, symbol, synonym, product, Gene Ontology, literature link, etc.), gene homology information, and gene upstream sequences.

4.2 Data query specification

Data managed by UBIC^2 contain various types of biological data. Users could retrieve the information in combinatorial explosive ways. For example, some may query the annotation information and the homologous genes of a set of genes; some may query both the annotation information of a set of genes and the annotation

information of their proteins; and so on. A naïve way to meet various information demands is to design and implement various corresponding functions that can be called from UBIC^2 applications. However, this requires designing and implementing a huge number of functions. The other way is to use exiting relational database systems to manage data and use SQL for data query. Nevertheless, this makes UBIC^2 applications bound to a database system. UBIC^2 intends to follow a query language XQuery [24], which is designed to be broadly applicable to XML data sources. As an initiative, UBIC^2 defines a simple XML schema for data query. Currently, the data query XML schema only covers gene annotation information. Query results are returned in XML formats defined by UBIC^2 XML data specifications. Figure 1 shows a query example, which retrieves the following information of two genes using their LocusLink IDs: official names, official symbols, synonyms, RefSeq IDs, UniGene IDs, and PubMed IDs of papers related to the genes.

5 Software Architecture

UBIC^2 software architecture contains six software layers and several local XML databases (Figure 2). The six layers are (1) Internet interfaces, (2) Remote database conductors, (3) Species-specific data integrators, (4) Application programming interfaces, (5) Applications, and (6) Web services. The following subsections will provide the details of each component.

5.1 Internet interface

The Internet interface layer provides programmable means for retrieving data from remote databases. To fetch a remote file, the program should specify the Internet network protocol, the Internet address (e.g., IP address, URI, etc.), and the file name. Currently, two network protocols, HTTP and FTP, are supported. If the database supports, clients can choose to download a chunk of data instead of the whole data file with the HTTP protocol. This layer also allows users to login to a database with a user id and password. By extending the Internet interface layer, developers can accommodate other network protocols.

5.2 Remote database conductors

This layer consists of a set of remote database conductors. Each remote database conductor deal with one remote database. A generic class is designed and implemented for the remote database conductors. The generic class contains common member variables and functions.

```
<UBIC2_GeneInfoQuery>
  <GeneIDType>LocusLinkID</GeneIDType>
  <GeneIDs>1,16999</GeneIDs>
  <GetOfficialName />
  <GetOfficialSymbol />
  <GetSynonyms />
  <GetPubMedID />
  <GetRefseqID />
  <GetUniGeneID />
  <GetLocusLinkID />
</UBIC2_GeneInfoQuery>
```

```
<GeneInfo>
  <Gene>
    <LocusLinkID>1</LocusLinkID>
    <Gene_Name>alpha-1-B glycoprotein</Gene_Name>
    <Gene_Symbol>A1BG</Gene_Symbol>
    <Gene_Synonym>A1B</Gene_Synonym>
    <Gene_Synonym>ABG</Gene_Synonym>
    <Gene_Synonym>GAB</Gene_Synonym>
    <RefSeqID>NM_130786</RefSeqID>
    <RefSeqID>NP_570602</RefSeqID>
    <UniGeneID>Hs.390608</UniGeneID>
    <PubMedID>2591067</PubMedID>
    <PubMedID>3458201</PubMedID>
    <PubMedID>8889549</PubMedID>
    <PubMedID>12477932</PubMedID>
  </Gene>
  <Gene>
    <LocusLinkID>16999</LocusLinkID>
    <Gene_Name>latent transforming growth factor beta binding protein
    4</Gene_Name>
    <Gene_Symbol>Ltbp4</Gene_Symbol>
    <Gene_Synonym>2310046A13Rik</Gene_Synonym>
    <RefSeqID>NM_175641</RefSeqID>
    <RefSeqID>NP_783572</RefSeqID>
    <UniGeneID>Mm.272251</UniGeneID>
    <PubMedID>10349636</PubMedID>
    <PubMedID>11042159</PubMedID>
    <PubMedID>11076861</PubMedID>
    <PubMedID>11217851</PubMedID>
    <PubMedID>12208849</PubMedID>
    <PubMedID>12477932</PubMedID>
  </Gene>
</GeneInfo>
```

Figure 1. A query example. Left: query. Right: query results.

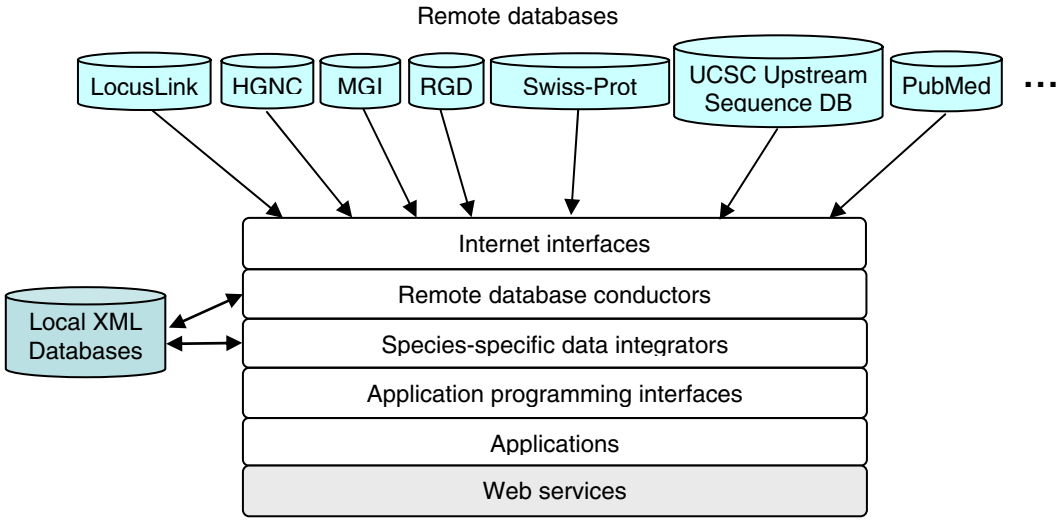

Figure 2. UBIC^2 Software Architecture.

Specific remote database conductors are derived from the generic class.

One function of remote database conductors is to download the whole remote data file. Many remote databases are very large and take long time to download. For the purpose of fair public access, many remote databases limit the amount of time that a thread can access. They will automatically disconnect threads that break the constraints or even block the IP addresses of the clients. A generic function is designed and implemented for the generic remote database conductor class to download all the required data while conform the constraints. This function can recover disconnected sessions and continue downloading data after certain time period until all data is downloaded.

Another function of remote database conductors is to convert the downloaded data files into XML formats. To make later information retrieval efficient, the local XML databases are indexed by major IDs (e.g., LocusLink ID, RefSeq ID, UniGene ID, etc.). Some of these XML databases are used by species-specific data integrators, which will be described later.

To date, the UBIC^2 software package supports a number of remote databases including LocusLink [22], HUGO Gene Nomenclature Committee (HGNC) [32], Mouse Genome Database (MGD) [2], Rat Genome Database (RGD) [38], Swiss-Prot [3], HomoloGene [21], and upstream sequence databases from UCSC Genome Bioinformatics [27].

5.3 Species-specific data integrators

This layer defines a set of species-specific data integrators. Different remote databases may have information for the same species. For example, both LocusLink and HGNC contain annotations for human genes. Since remote biological databases were developed cooperatively as well as independently, the information in those databases could be complementary, overlapping, and inconsistent. Each species has an integrator. A species-specific data integrator combines information from different databases, detects inconsistency, reconciles data discrepancy, and handles heterogeneous data query that are related to the specific species. Through this layer, all previously miscellaneous data are as if coming from one unified database to the user.

The work flow of this layer is described as the following. First, the integrators retrieve information from the databases downloaded and reformatted by remote database conductors, and then combine information for the same species from different databases. Currently, the information is categorized into four types: gene annotation, protein annotation, gene homology annotation, and upstream sequence. Gene annotation include the following information of a gene: LocusLink ID, RefSeq ID, UniGene ID, GenBank ID, Swiss-Prot ID, Gene Ontology annotation, official name, official symbol, synonyms, PubMed IDs of related literature. Protein annotation includes: Swiss-Prot ID,

protein name, EMBL IDs, HSSP IDs, InterPro IDs, Pfam IDs, PIR IDs, PRINTS IDs, ProDom IDs, PROSITE IDs, SMART IDs, and TRANSFAC IDs. Each type of information is stored in an XML database, which is indexed by major IDs (e.g., LocusLink ID, Swiss-Prot ID, UniGene ID, etc.).

In addition to information integration, the integrators also detect data inconsistency across databases, reconcile data discrepancies, and track obsolete information. The integrated databases are stored as local species-specific XML databases, which will be addressed later. The detected inconsistency is stored and reported to users. Currently, the integrators invoke a simple method to automatically reconcile such discrepancies. This method uses a designated database as the correct information source. Developers can replace this method by simply overriding the discrepancy reconciliation function of the integrator.

After finishing integration, the integrators index the integrated databases and handle query operations. Gene annotation databases are indexed by LocusLink IDs, RefSeq IDs, UniGene IDs, and GenBank IDs. Protein annotation is indexed by Swiss-Prot IDs. Upstream sequence database is indexed by RefSeq IDs. When a query is received, a data integrator first tries to retrieve the information from local databases. If local databases do not have the required information, it will perform information query to remote databases through remote database conductors. Online query of remote databases is useful when a remote database (e.g., PubMed) is too big for a local machine. If users use obsolete information, e.g., obsolete LocusLink IDs, to query other information, Data integrators will direct users to the latest information, e.g., new LocusLink IDs, and return the update-to-date information.

For example, if the user uses a set of LocusLink IDs to retrieve gene annotations, upstream sequences, genes' protein annotations, and PubMed abstracts. The system will first retrieve gene annotation information from species-specific data integrators. Gene annotation information contains RefSeq IDs and Swiss-Prot IDs of genes and PubMed IDs of the gene related to papers. Then, the integrators will retrieve upstream sequences from upstream sequence databases using RefSeq IDs, protein annotations from protein annotation database using Swiss-Prot IDs, PubMed abstracts from the PubMed conductor using PubMed IDs.

This layer now consists of integrators for *Homo sapiens*, *Mus musculus* and *Rattus norvegicus*. The developers can also add new ways of heterogeneous data integration by adding new integrators, which can be derived from the generic class of integrators.

5.4 Application programming interfaces

This layer provides a set of basic data structures (e.g., basic data elements for genes and proteins, lists, hash-table, and so on). Using the application programming interfaces, the application developers can quickly develop applications that are capable of dynamically retrieving heterogeneous biological information. The retrieved information is organized as objects in computer memory, and hence is computable. Miscellaneous data computing functions should be designed and implemented in this layer. This layer also contains of a query generation object. Users can specify the desired combination of different information by instantiating the member variables of the query generation object. The query generation object will generate queries in XML format, which can be passed to the query functions of species-specific data integrators.

5.5 Applications

Built on top of the API layer, UBIC^2 applications manipulate heterogeneous data. The applications aim at serving end-users who consume the results of heterogeneous information integration, retrieval, and computation. End-users can get the results by interacting with the graphical interfaces or command line executions of applications.

5.6 Web services

Web services support interoperable computer-to-computer interaction over a network. UBIC^2 Web services provide interfaces, which are described in a machine-processable format, as standard means of interoperating between different biological data management and computation applications, which run on various platforms. The customers of UBIC^2 Web services invoke the Web service using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards. This layer allows different groups to work flexibly, distributedly,

and cooperatively using incongruent component models, operating systems, and programming languages.

6 UBIC^2 Version 1.0

In the current release of UBIC^2 (version 1.0), we implemented the UBIC^2 software architecture using the Microsoft® C# programming language and the Microsoft® .Net technology, which supports FreeBSD OS and Mac OS X 10.2. The UBIC^2 concept can be implemented using other programming languages and platforms, e.g., Java and Apache/SOAP. At the level of Web services, remote clients can be implemented in any programming languages and run on different operating systems.

To illustrate how to use the UBIC^2 software package, UBIC^2 1.0 is distributed with a demo application. The demo application mainly has three features: (1) download remote databases; (2) integrate downloaded databases and store the results in local XML databases; (3) real-time batch retrieval of heterogeneous information. UBIC^2 1.0 implemented a Web service that provides gene annotation query service [14]. The Web service is deployed on a Window Server 2003 installed with Microsoft Internet Information Service 6.0. The query request should be described using UBIC^2 XML Schema for data query. The UBIC^2 programming library, demo, documents, and manuals are available at *http://www.ubic2.org*.

7 Discussions

The current version of UBIC^2 has the following features that favor small research groups and individual researchers. Firstly, it is an economical way for data management. The data is maintained in local lightweight XML databases and can be manipulated without being tied up to any database systems. Secondly, it provides a set of objects and APIs that enable researchers to quickly build up their own applications, which can be delivered without any requirements on database systems. Thirdly, it is easy to expand the coverage to new types of data by designing new objects.

To support high throughput data desired by larger group of users, we are considering incorporating relational database management system (e.g., MySQL [18]). In the future, it should allow users to specify if they desire XML databases or relational databases. Mechanisms will be designed and implemented to transfer data between XML databases and relational databases.

References

[1] Ashburner, M., C. A. Ball, et al. (2000). "Gene ontology: tool for the unification of biology. The Gene Ontology Consortium." Nat. Genet. 25(1): 25-9.

[2] Blake, J. A., J. E. Richardson, et al. (2003). "MGD: the Mouse Genome Database." Nucleic Acids Res. 31(1): 193-5.

[3] Boeckmann, B., A. Bairoch, et al. (2003). "The SWISS-PROT protein knowledge base and its supplement TrEMBL in 2003." Nucleic Acids Res. 31(1): 365-70.

[4] Bukhman Y. V. and J. Skolnick. (2001) "BioMolQuest: integrated database-based retrieval of protein structural and functional information." Bioinformatics. 17(5):468-78.

[5] Chapman, B. and J. Chang (2000). Biopython: Python tools for computation biology. ACM SIGBIO.

[6] Conlon, E. M., X. S. Liu, et al. (2003). "Integrating regulatory motif discovery and genome-wide expression analysis." Proc. Natl. Acad. Sci. USA 100(6): 3339-44.

[7] Davidson, S. B., C. Overton, et al. (1997). "BioKleisli: A Digital Library for Bio-medical Researchers." International Journal of Digital Libraries 1(1): 36-53.

[8] Diehn, M., G. Sherlock, et al. (2003). "SOURCE: a unified genomic resource of functional annotations, ontologies, and gene expression data." Nucleic Acids Res. 31(1): 219-23.

[9] Dowell, R. D., R. M. Jokerst, et al. (2001). "The distributed annotation system." BMC Bioinformatics 2(1): 7.

[10] Eckman, B. A., A. S. Kosky, et al. (2001). "Extending traditional query-based integration approaches for functional characterization of post-genomic data." Bioinformatics 17(7): 587-601.

[11] Galperin, M. Y. (2004) "The molecular biology database collection: 2004 update." Nucleic Acids Research 32:D3-D22.

[12] Haas, L. M., P. M. Schwarz, et al. (2001). "DiscoveryLink: A system for integrated access to life sciences data sources." IBM Systems Journal 40(2): 489-511.

[13] Hanisch, D., A. Zien, et al. (2002). "Co-clustering of biological networks and gene expression data." Bioinformatics 18 Suppl. 1: S145-S154.

[14] http://bayes.fas.harvard.edu/Webservice/BioWebservice/BioWebservice.asmx

[15] http://www.bioconductor.org/

[16] http://biojava.org

[17] http://bioruby.org/

[18] http://www.mysql.com

[19] http://www.ncbi.nih.gov/dtd/

[20] http://www.ncbi.nlm.nih.gov/entrez/query/static/PubMed.dtd

[21] http://www.ncbi.nlm.nih.gov/HomoloGene/

[22] http://www.ncbi.nlm.nih.gov/LocusLink/

[23] http://www.ebi.ac.uk/swissprot/SP-ML/

[24] http://www.w3.org/TR/xquery/

[25] Hucka, M., A. Finney, et al. (2003). "The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models." Bioinformatics 19(4): 524-31.

[26] Huang, Y., T. Ni, et al. (2003). "JXP4BIGI: a generalized, Java XML-based approach for biological information gathering and integration." Bioinformatics 19(18): 2351-2358.

[27] Kent, W. J., C. W. Sugnet, et al. (2002). "The human genome browser at UCSC." Genome Res. 12(6): 996-1006.

[28] Kohler, J. and S. Schulze-Kremer (2002). "The semantic metadatabase (SEMEDA): ontology based integration of federated molecular biological data sources." In Silico. Biol. 2(3): 219-31.

[29] Miyazaki, S., H. Sugawara, et al. (2003). "DNA Data Bank of Japan (DDBJ) in XML." Nucleic Acids Res. 31(1): 13-6.

[30] Mulder, N. J., R. Apweiler, et al. (2002) "InterPro: an integrated documentation resource for protein families, domains and functional sites." Briefings in Bioinformatics. 3(3):225-35.

[31] Paton, N. W., R. Stevens, et al. (1999). Query Processing in the TAMBIS Bio-informatics Source Integration System. Proceedings of the 11th International Conference on Scientific and Statistical Database Management, New York, IEEE.

[32] Povey, S., R. Lovering, et al. (2001). "The HUGO Gene Nomenclature Committee (HGNC)." Hum Genet 109(6): 678-80.

[33] Schonbach C., P. Kowalski-Saunders, and V. Brusic. (2000) “Data warehousing in molecular biology.” Brief. Bioinform. 1(2):190-8.

[34] Spellman, P. T., et al. (2002). "Design and implementation of microarray gene expression markup language (MAGE-ML)." Genome Biol. 3(9): research0046.1-0046.9.

[35] Stajich, J. E., D. Block, et al. (2002). "The Bioperl toolkit: Perl modules for the life sciences." Genome Res. 12(10): 1611-8.

[36] Stein, L. (2002). "Creating a Bioinformatics Nation." Nature 417: 119-120.

[37] Stuart, J. M., E. Segal, et al. (2003). "A gene-coexpression network for global discovery of conserved genetic modules." Science 302(5643): 249-55.

[38] Twigger, S., J. Lu, et al. (2002). "Rat Genome Database (RGD): mapping disease onto the genome." Nucleic Acids Res. 30(1): 125-8.

[39] Wang, L., J. J. Riethoven, et al. (2002). "XEMBL: distributing EMBL data in XML format." Bioinformatics 18(8): 1147-8.

[40] Waugh, A., P. Gendron, et al. (2002). "RNAML: a standard syntax for ex-changing RNA information." Rna 8(6): 707-17.

[41] Zdobnov, E. M., R. Lopez, et al. (2002). "The EBI SRS server-new features." Bioinformatics 18(8): 1149-50.

[42] Zhang, J., V. Carey, et al. (2003). "An extensible application for assembling annotation for genomic data." Bioinformatics 19(1): 155-6.

[43] Zhong, S., C. Li, et al. (2003). "ChipInfo: Software for extracting gene annotation and gene ontology information for microarray analysis." Nucleic Acids Res. 31(13): 3483-6.