A Genome Databases Framework

Luiz Fernando Bessa Seibel
Sérgio Lifschitz

Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro
<seibel,lifschitz>@inf.puc-rio.br

PUC-RioInf.MCC20/01 March, 2001

Abstract: There exists many molecular biology databases, also known as Genome Databases, that need to be integrated together with all related applications and other data sources. This work proposes the use of an object-oriented framework for genome data access and manipulations. The framework approach may be an interesting solution in this context due to the flexibility, reusability and extensibility requirements of this application domain. We will present and discuss the framework in details using class and sequence diagrams that explore both the structural and dynamic parts of the architecture.

Keywords: Molecular Biology, Bioinformatics, Application Framework, Databases Integration

Resumo: Existem inúmeros Bancos de dados de Biologia Molecular, também conhecidos como Bancos de Dados de Genoma, e existe a necessidade de integração destas fontes de dados e das aplicações a elas relacionadas. Este trabalho propõe a utilização de um framework orientado a objetos para acesso e manipulação desses dados de genoma. A adoção de um framework pode ser uma abordagem interessante neste contexto devido aos requisitos de flexibilidade, reuso e extensibilidade deste domínio de aplicação. O framework será discutido e apresentado em detalhes, utilizando os diagramas de classes e de sequências que exploram os aspectos estruturais e dinâmicos da arquitetura.

Palavras-chave: Biologia Molecular, Bioinformática, Framework de Domínio de Aplicação, Integração de Bancos de Dados.

1. Introduction

Many molecular biology projects are currently active and in spite of all the benefits one may expect from them, it has become a challenging problem to deal with large volumes of DNA and protein sequences, besides other related data (such as annotations) [12, 30]. DNA and protein sequences are text strings and this is one of the reasons why molecular biologist started keeping them in text files. With new technologies available, the sequencing process and genetic code production has increased in such a way that the total volume of data became large enough, motivating the use of DBMSs [12].

Database technology is already present in this context but to a little extent, *i.e.*, even if some projects include DBMS-like software to store all the data, most does not use DBMS's functionalities [12, 3, 30]. Moreover, most users still work with flat text-based files downloaded from public repositories and data access is done through *ad-hoc* programs like BLAST search [26, 16]. There exists many Molecular Biology Databases, also known as Genome Databases, such as the GenBank Sequence Database [17], the Annotated Protein Sequence Database (Swiss-Prot) [31] and A C. elegans Database [1]. It is important to note that many so-called databases are not always complete database systems but rather file systems with own manipulation and access programs.

There are many interesting problems for the database research community, besides simply providing a way to store and give reliable and efficient access to large volumes of data [25, 11, 29]. Among them, we can mention the works on appropriate user interfaces and the interaction between different data collections [5, 33]. Some other issues involve the definition of an appropriate ontology [4], as well as dealing with efficient data access through new and faster algorithms [2], memory management [21] and much more.

We are interested here with a basic, though very important, problem that is related to the definition of a suitable data model for representing and integrating this kind of genome data. It is a well-known problem for genome and molecular biology data users that the information widely spread in different sites are not easy to deal with in a single and uniform way. Each research group that is currently generating or processing these data usually work in a independent manner, using different data models to represent and manipulate mostly the same information. For example, there are object-oriented systems (e.g., AceDB [1]), relational ones (e.g., Swiss-Prot [31]) and text-based files with a semi-structured organization (e.g., GenBank [17]).

The usual approach to handle this information integration problem is to use a specific integrated model and system that should capture all the needed data for a particular application [e.g., 9]. Since it is a research area that is often changing, with new structural information being incorporated together with new application requirements, it is very difficult to decide upon which data model should be considered in this context. Thus, every existing approach based on a chosen model may not be well adapted to all users and application needs.

In this work we propose the use of an object-oriented framework [7, 14] approach to deal with these Genome data integration problem. We claim that using a framework and the software systems instantiations it may generate, we have a better solution to most of the questions that arise in this domain, due to its flexible and extensible architecture. This framework, briefly introduced in [22], will be discussed in details here, with class and sequence diagrams that explore both the structural (static) part and the way it can be used (dynamic).

We first motivate our work in the next section, listing a sample of the existing genome data sources and tools, together with some of the most important approaches in the literature. Then, in Section 3, we give an overview of our framework, presenting its basic architecture, followed by the details of its structural part shown in Section 4. The dynamic part is represented by sequence diagrams and is discussed in Section 5, with conclusions and future work appearing in Section 6.

2. Motivation and Related Work

Molecular genome projects, through sequencing, have produced very large collections of DNA data of multiple organisms. An important problem in this research area is how to deal with gene and other genome sites in order to identify their functions. Therefore, it is important to be able to make comparisons between genome data of individuals that may be either the same or different species. Many groups have developed tools to provide integration, structuring, comparison and presentation of genome sequences and related information.

In [20, 11, 23] the authors identify and organize the most important integration strategies in 3 groups, either if they consider hyperlink navigation for joining information; or a multidatabase approach, or even those that work with data warehouses. The first one aims at allowing the users to jump through registers of different data sources, either through existing links among them [13] or navigation systems that create the links among different data sources [32]. Thus, in a first movement, the user accesses a data source register and, in what follows, the user asks for a link to another data source where the desired information is. The second group includes those that use integration tools that implement queries to the different pre-existing data sources. These queries may be formulated through special languages [8, 10] that allow representing complex data types, with an access driver implemented for each data source to be accessed. Another strategy consists of using a mediator that is responsible for determining the data sources that participate in the query, creating an access plan, translating concepts and syntax, assigning the queries to the distributed environment and integrating the results [20]. The last mentioned integration approach deal with implementation of a data instance that collects the biological information available in several sources.

When genome and molecular biology information structuring are taken into account, there exist research groups that propose and discuss data models that are suitable to represent them. We can cite the OPM semantic-based object-oriented model, also the DDBJ DNA Database presented in [27] and, more recently, the data warehouse-like approach proposed in [28]. Usually genome projects develop own system interfaces that differ in the way they show their data and results from data manipulation. For example, AceDB [1] offer a powerful graphical interface that enables the user to visualize the chromosome map in details. Last but not least, there are research groups that are mainly interested in developing tools that implement comparison and alignment algorithms, which run over specific data formats [29].

We have chosen here an approach that is based on an object-oriented framework [7, 14]. A framework is an incomplete software system, which contains many basic pre-defined components (*frozen spots*) and others that must be instantiated (*hot spots*) for the implementation of the desired and particular functionality. Our proposed framework may be considered specific to the application domain, in our case, molecular biology and genome area.

The basic idea for choosing a framework approach is that we needed a tool for integrating genome information that is spread in a distributed environment (mostly available in the web). This information is constantly changing and used by distinct (and at the same time similar), applications. So, properties like flexibility and extensibility, together with software reusability, are required. The biology data model, once initially defined, need to incrementally aggregate new information from the different existing data sources. Through a framework, it becomes possible to generate interfaces and database instances, executing the most important genome applications in a uniform and integrated way.

Our approach integrates the information through the instantiation of particular scientific data warehouses, which respond to high performance requests of the related applications. We have chosen XML Schema [34] to define the data sources schemas and XML [35] for storing data. This is due to the intrinsic characteristic of the information in the biology data sources and to the eventual adoption of XML Schema by many commercial CASE tools.

3. Framework General Description

This section describes the main characteristics of the proposed framework, besides its general architecture and main modules. In the next two sections we will detail both the framework static part, showing its classes and attributes, methods and relationships, and the dynamic part, which explores how the framework can be used.

The framework provides six basic functionalities:

(1) Schemas capture of existing and different data sources;

(2) Matching of the architecture objects to those objects in the captured schema;

(3) Capture of data belonging to the data sources;

(4) Definition of new *ad-hoc* schemas;

(5) Data generation in a format required by a molecular biology application;

(6) Execution of algorithms instantiated as methods of the molecular biology classes.

The first one assumes that there exists a converter (wrapper) for the data sources being considered. When the second functionality is not directly obtained, a new biology object is created and the pertinent associations must be created. This matching may establish relationships such as "*is_synonym_of*". The ability of capturing the data is the third functionality mentioned and is needed once the associated schema has been already captured. For this, there is another type of converter. Listed next in fourth place there is a common demand is for new schemas for specific applications, together with a new associated data set. As there are multiple stored data formats, one may need to convert them to a particular format, mandatory for the execution of a given application (e.g., FASTA file format for BLAST programs). This is what the fifth functionality is related to and, finally, the framework must be able to execute all the involved methods.

The framework being proposed is divided in four modules: *Administrator, Captor, Driver and Converter*. Their relationship and an overview of the framework architecture is depicted in Figure 1:

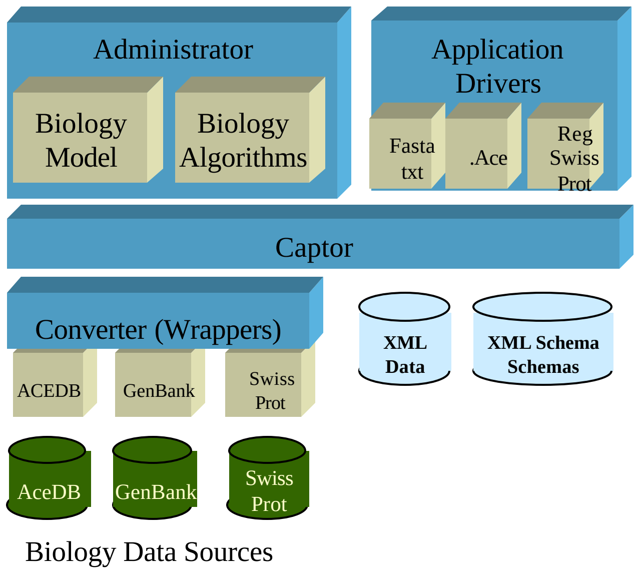

Figure 1 - The Framework Architecture

The *Biology Model* and *Algorithms*, *Wrappers* associated to biology data sources and *Application Drivers* are the framework hot spots. When instantiated, they implement a particular functionality, defining an application over the molecular biology application domain.

The *Administrator* module performs the interface with the users in a way to provide management of the biological data model; request for capturing schemas and/or data; and allow the execution of the algorithms instantiated in the framework itself. Therefore, this module contains a biology class model that is committed with the existent data sources, as well as with the methods that are associated to these classes.

The *Captor* module is responsible for the data and architecture schemas repository. The *Converter* provides access to the biology data sources, making the translation of the data sources schemas to XML Schema pattern language and data for XML. The choice of XML is quite immediate due to the known advantages of semi-structured data models in this molecular biology context [18]. Finally, the *Drivers* module implements the interface generation between the biology applications and the framework.

For example, if a user make a request to the Administrator module to capture schemas and/or data of a given data source, the Administrator sends the request to the Captor, which in turn sends it to the Wrapper of the biology data source being considered. The capture of schemas or data may only be done if the correspondent wrapper have been previously developed and instantiated in the framework. The wrapper implements the mapping of the data sources schema to XML Schema and data to XML. Both the schemas and the data captured are stored in their respective repositories.

The user may also ask the Administrator module for generation of a file for a given biology application. Much like as described before, such a request can only be done if the associated driver have been previously developed and instantiated in the architecture. The Administrator module triggers the Driver in order to execute the demanded task. The Driver then requests the data to the Captor, which manipulates the data repository. For instance, there are multiple biology applications that ask for the file name and localization in order to proceed with the execution.

The architecture also allows, via Administrator module, the execution of a biology algorithm instantiated in the architecture, which may work on the available data stored in the repository. The construction of interfaces between the framework and the existent biology applications can also be done through the Application Driver, *i.e.*, input points may be instantiated so to request the data in specific formats for a class of applications. Thus, data services can be built for applications developed externally.

4. Framework Modules

This section will detail the architecture of each module, presenting the classes' diagrams, their attributes, the correspondents' methods and their relationships. Each module has a class that represents its interface with the other modules. We will be using the *Facade* pattern [15] in their implementation.

The framework is described in the next sections, using the Unified Modeling Language [6]. We will explain each module and its classes with related functionalities. The attributes and methods will not be detailed but are quite immediate.

4.1 Administrator Module

There are 3 classes - *AdmFacade*, *BiologyModel* and *RepositoryModel* - in the Administrator module. The class *AdmFacade* is the path to all framework functionalities. The users interact with this class in order to:

1. Capture schemas (and respective exclusion) of data sources whose wrappers were previously instantiated in the architecture;
2. Obtain the matching between the biology schema, which is present in the architecture, and the schema that was captured from a data source. When this matching does not occur, new objects can be added to (or suppressed from) the model. Also, associations between the objects can be done or undone and object can be recognized (or not) as synonyms of other objects. This functionality is the one that permits the insertion of new biology concepts in the respective model. Both new objects definitions and new associations among objects in the biology model must be represented in XML Schema.
3. Create of a new schema and data instantiation that are appropriate to a given application. The new schema will be created from the objects already defined in the biology model.
4. Capture data from a biology data source;
5. Generate data for external applications;
6. Query schemas and data repositories;
7. Execute biology programs.

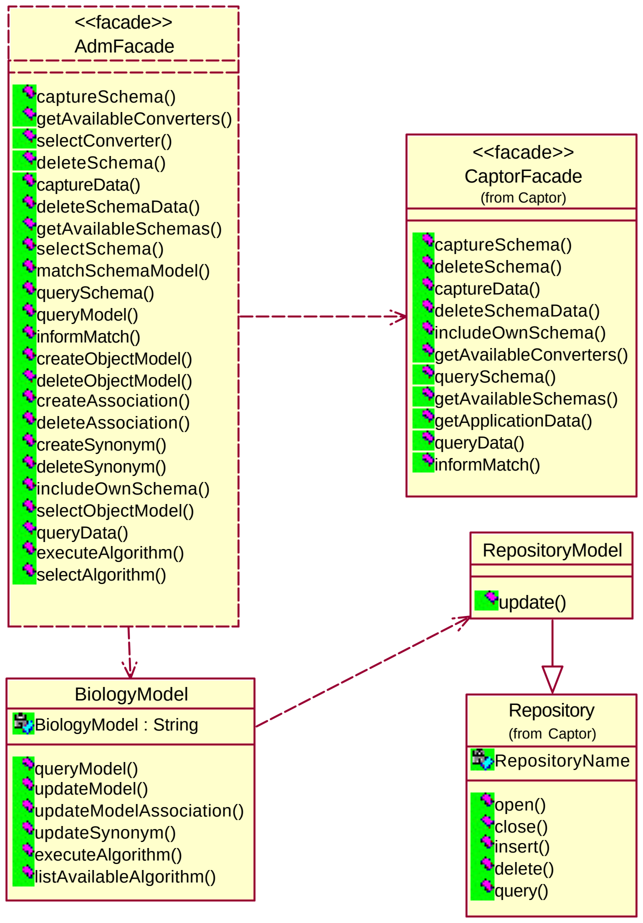

Figure 2 - Classes diagram of the Administrator module

The class *BiologyModel* manipulates the biology data model, allowing its expansion. It is important to note that classes that are part of the Administration module may be extended or modified by programmers. They are, indeed, hot spots of the proposed framework. The Strategy pattern [15] is used here to permit the creation of algorithms families associated to the biology model's classes. This way the programmers can implement variations of the available algorithms.

The class *RepositoryModel* provides persistency of the objects is the biology model, as well as their retrieval from the Repository.

4.2 Captor Module

There are 2 classes in the Captor module: the class *CaptorFacade* provides the following functionalities:

1. The capture and storage of biology data sources' schemas;
2. The management of own specific schemas, defined from the objects of the biology model, which is available in the framework;
3. The exclusion of own and/or captured schemas;

4. The capture and the storage of the data associated to a schema;
5. The exclusion of the data that were associated to a schema;
6. The query execution over the repository class, described next.

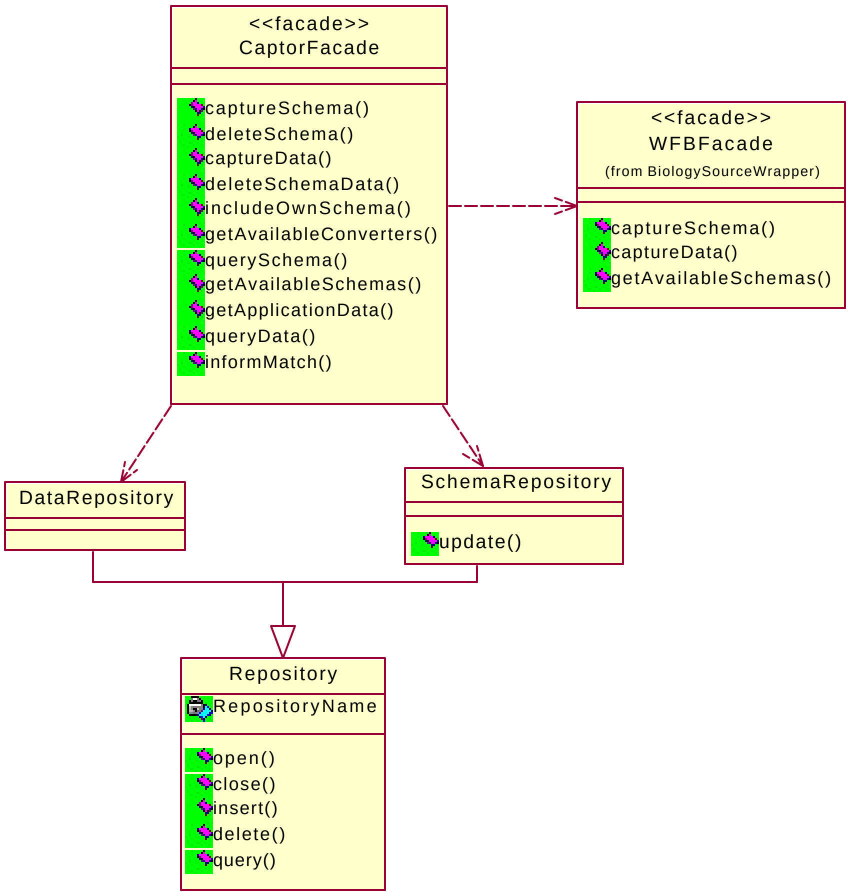

Figure 3 – Class diagram of the Captor module

The 2nd class, called *Repository*, provides persistency of schemas and data, besides enabling data retrieval.

It is worth to note that the schemas are stored in XML Schema and all data in XML. So, there is a need for access and manipulation languages, such as XQuery [36], to deal with the *Repository*.

4.3 WrapperBiologySources (WFB) module

The following classes compose the WrapperBiologySource module:

- *WFBFacade* is an interface class between the framework modules and the biology data wrappers. There will be various converters in the architecture, one for each data source.

Therefore, the relationship between the *WFBFacade* and the *Wrappers* is of the type one-to-many.

- *DataSourceWrapper* represents the implementation of each wrapper. A wrapper will contain two distinct functionalities. On one hand, it has the ability to capture the biology data source schema and, on the other hand, the capture of the source's data itself. The *DataSourceWrapper* class is a hot spot of the architecture.

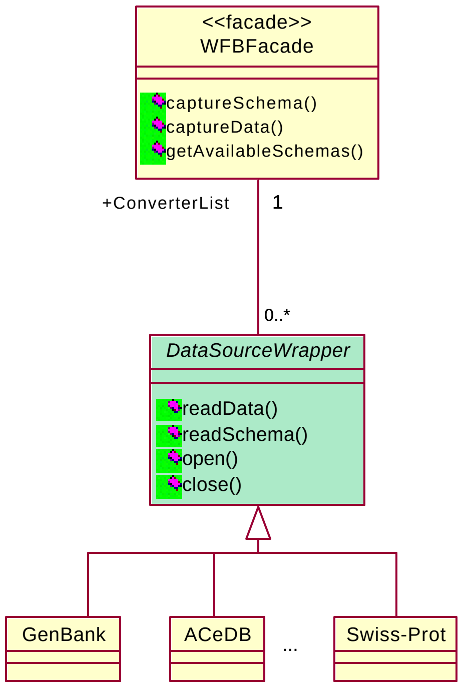

Figure 4 – Class Diagram of the WrapperBiologySource

4.4 DriverBiologyApplication Module

The following classes compose the DriverBiologyApplication module:

- *ApplicationDriverFacade* is an interface class between the modules of the framework and the drivers that generate data for the biology applications. There will exist multiple application drivers in the architecture, one for each application program to be used. Thus, the relationship between the *ApplicationDriverFacade* and the drivers is "one-to-many" type.

- *DataGenerator* represents the implementation of each driver. The driver is also a framework hot spot. For instance, a driver can generate data in a text format, according to the syntax used in GenBank or Swiss-Prot, or even in FASTA format to be used in the execution of algorithms that work on them. Moreover, a driver may be the implementation of an interface with a system available in the Web. It can send the available data in the framework repositories to a system that will execute and manipulate them. The driver can also be a data service, allowing an application to be connected to the framework, receiving the data stored there.

In the next Section, we describe the biology data model used in our framework.

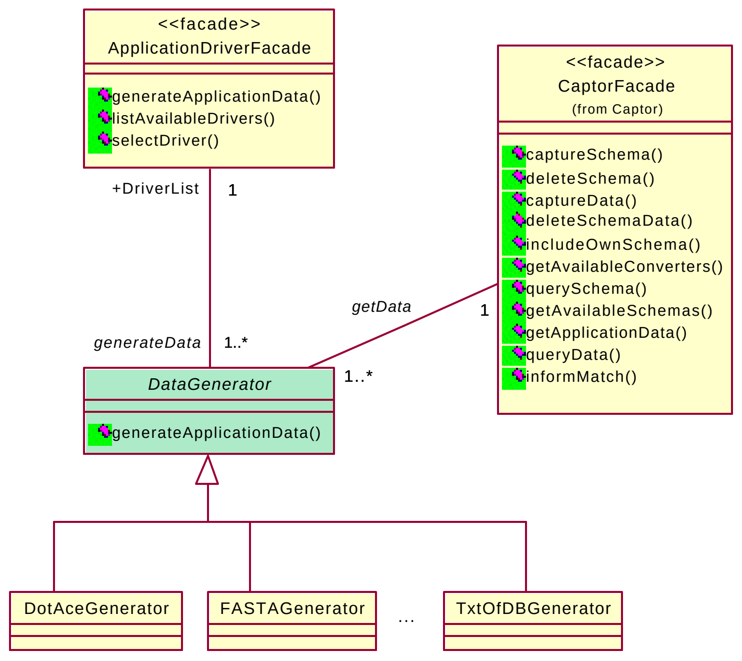

Figure 5 – Class Diagram of the DriverBiologyApplication module

4.5 An example of Biology model

The biology model is exemplified through the following class diagrams:

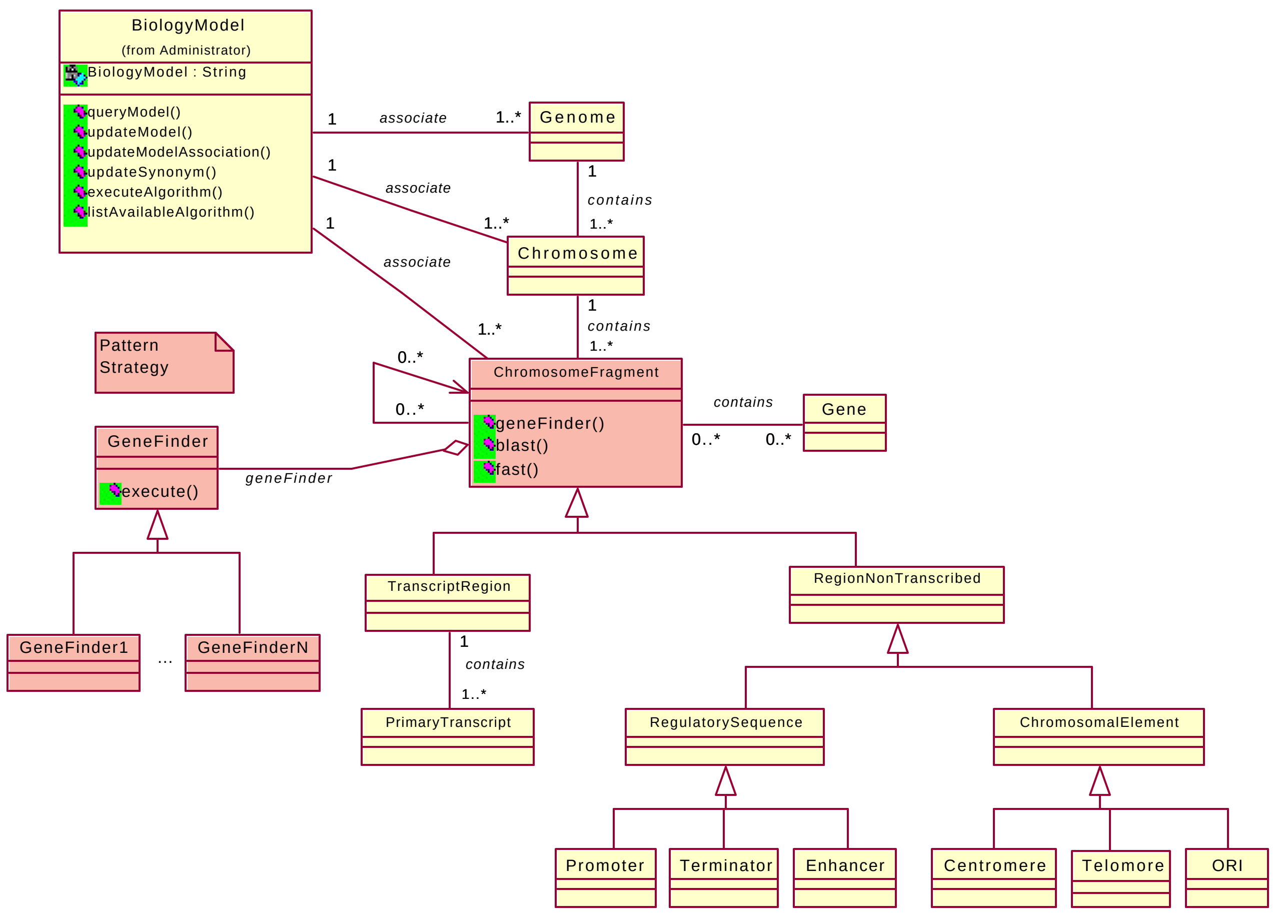

Figure 6 - Example of Genome Class diagram

The main goal for giving an example of a biology model in this point is to present a representation of the objects involved in the biology application domain, as well as the algorithms associated to each one of these objects.

The model presents only a small part of the application domain objects, specifically the pieces of information related to the genome. Other facts refer, for example, to the proteome, transcriptome and metabolone. This module will be extended so that all the information currently available in the biology data sources is considered.

In the model depicted in Figure 6, one may observe that *chromosomes* form the *genome*, and that each of them is considered a set of *Chromosome Fragments*, which consists of DNA sequences. A *Chromosome Fragment* can be either a *Transcript Region* or a *Non-Transcript Region*. The latter can be a *Regulatory Sequence* or a *Chromosomal Element*, and so forth.

In the class that represent the chromosome fragments, the types of the algorithms are defined, which may be executed over these objects that are, for example, algorithms of the type *GeneFinder*, *Blast* or *Fast* [24, 19]. The algorithms of the type *GeneFinder* are those that run discovery processes (data mining) of DNA regions in the fragments, for which genes formation is possible.

The class *ChromosomeFragment* is, then, associated to the class *GeneFinder*, which instantiates the several algorithms of this type. The classes *FragmentChromosome*, *GeneFinder*, *GeneFinder1*,... *GeneFinderN* use the *Strategy* design pattern [15]. It is worth to remember that the algorithms are also architecture hot spots.

5. Dynamic Part of the Framework

This section details the framework functionalities, presenting the existing interaction among classes. The functionalities are described through sequence diagrams [6] and are also briefly commented.

We illustrated in Figure 7 the sequence diagram for the execution of the function "Capturing Schema". Initially, available wrappers are presented to the user. Then, the user selects a wrapper for triggering the capturing of schema, which is processed by the *AdmFacade* and sent to the others. The class *WFBFacade* activates, then, the corresponding wrapper, which returns the data source schema (in XML Schema), in conformity to the biology model in the architecture.

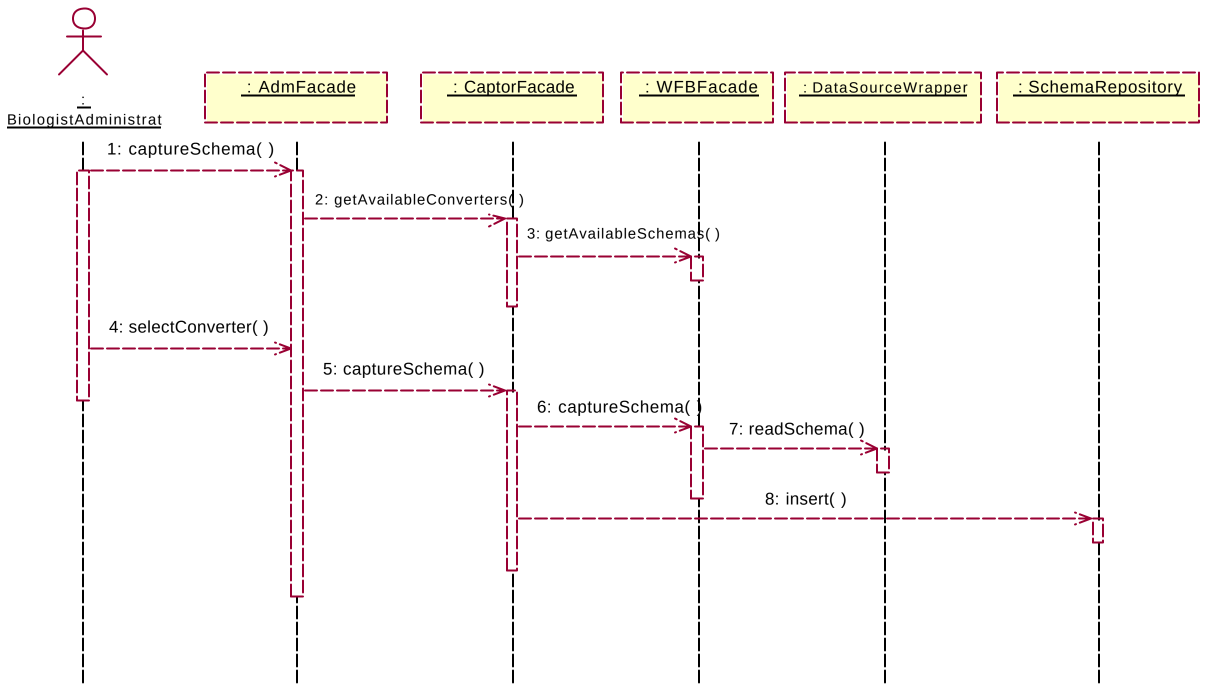

Figure 7 – Sequence Diagram for "Capturing schema"

The function "Capturing data" is illustrated with the sequence diagram in Figure 8, which is quite similar to the preceding function of schemas' capturing. Initially, the available wrappers are presented to the user. Then, the user selects a wrapper to activate the data capturing, which is processed by *AdmFacade*, and sent to the others. The class *WFBFacade* activates the corresponding wrapper, which retrieves from the data source the data already in XML format.

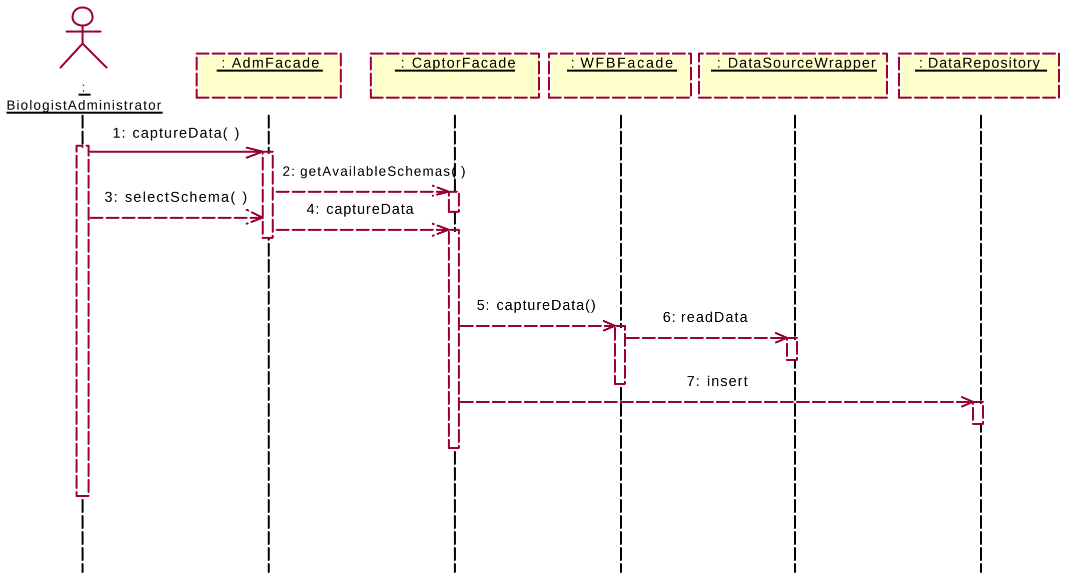

Figure 8 – Sequence Diagram for "Capturing data"

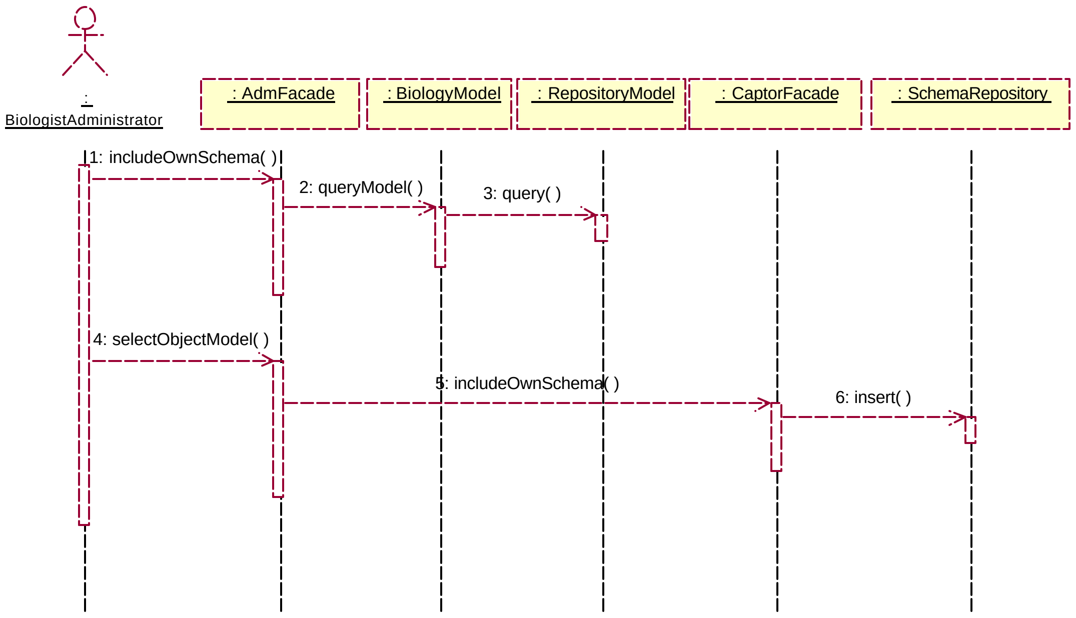

Figure 9 – Sequence Diagram for "Create own schema"

In Figure 9 we present the sequence diagram for the "Create own schema" functionality, which queries the biology data model and enable the user to select objects in the model in order to build an own schema. When the selection is ended, the built schema is stored in the Repository.

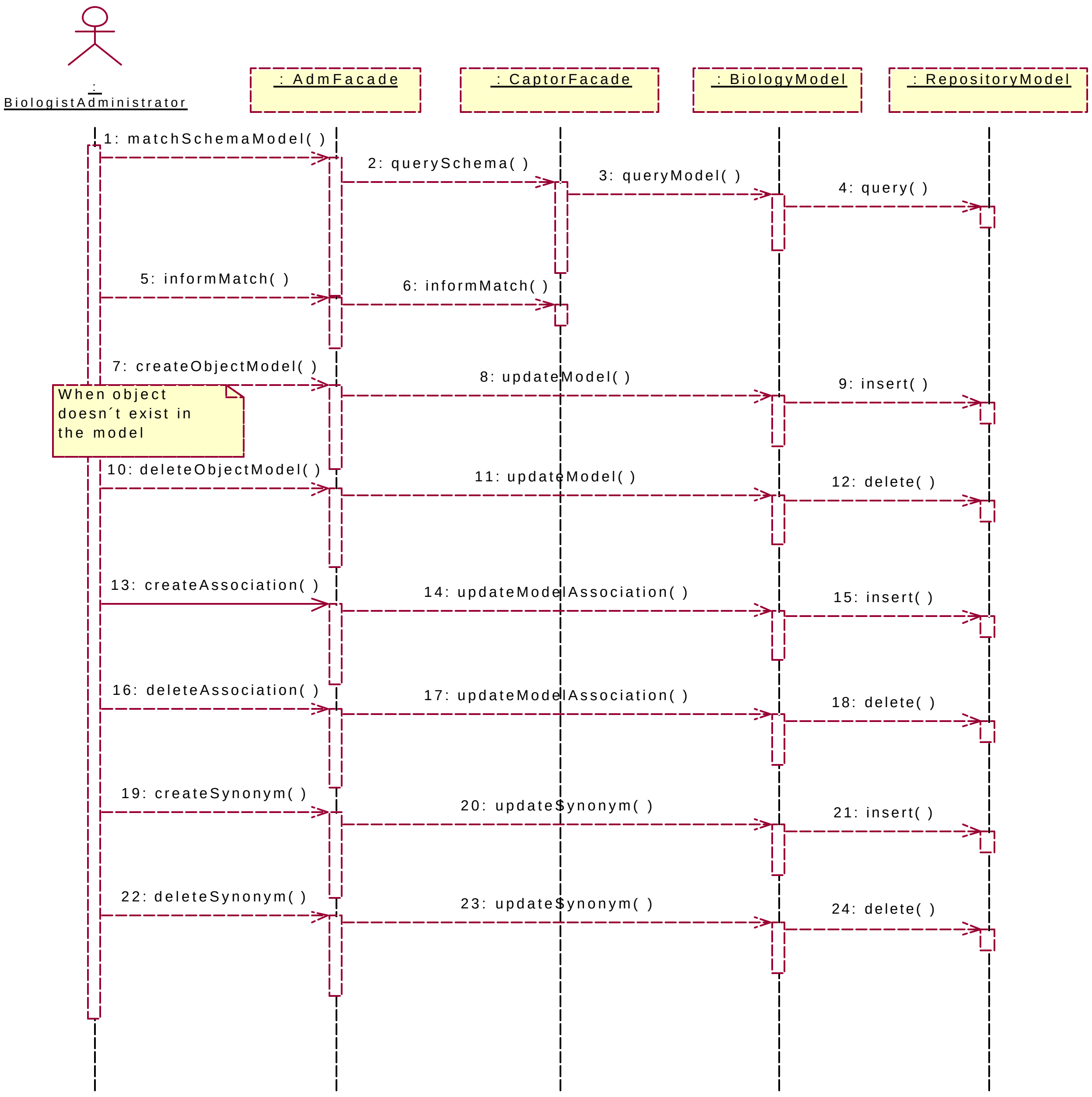

Figure 10 – Sequence Diagram for "Matching the data source schema with the biology model"

Figure 10 presents the sequence diagram for "Matching the data source schema with the biology model". Initially, both the biology data model and the data source schema one wants to match with the model are compared. If the source is already stored, the matching is annotated in the schema. If not, the user may proceed with some updates in the biology model in order to provide the desired matching. The possible manipulations are inclusion/exclusion and creation/elimination of synonyms, all these for objects and objects associations.

It is worth to emphasize that a careful maintenance of the biology model is extremely important for the generation of an ontology, which is effectively used in the molecular biology data sources.

The functionality "Generating data for a given application" is illustrated with the sequence diagram in Figure 11. When the available drivers in the architecture are presented, the user selects the one that will generate a file to be used in the execution of the application. The data are obtained from the repository and delivered to the external application in execution. It is very common to find an application in the Web, which queries the localization and the name of the file (with a pre-established format) to be processed. This functionality is the one that enables such applications to work on the available data in the architecture.

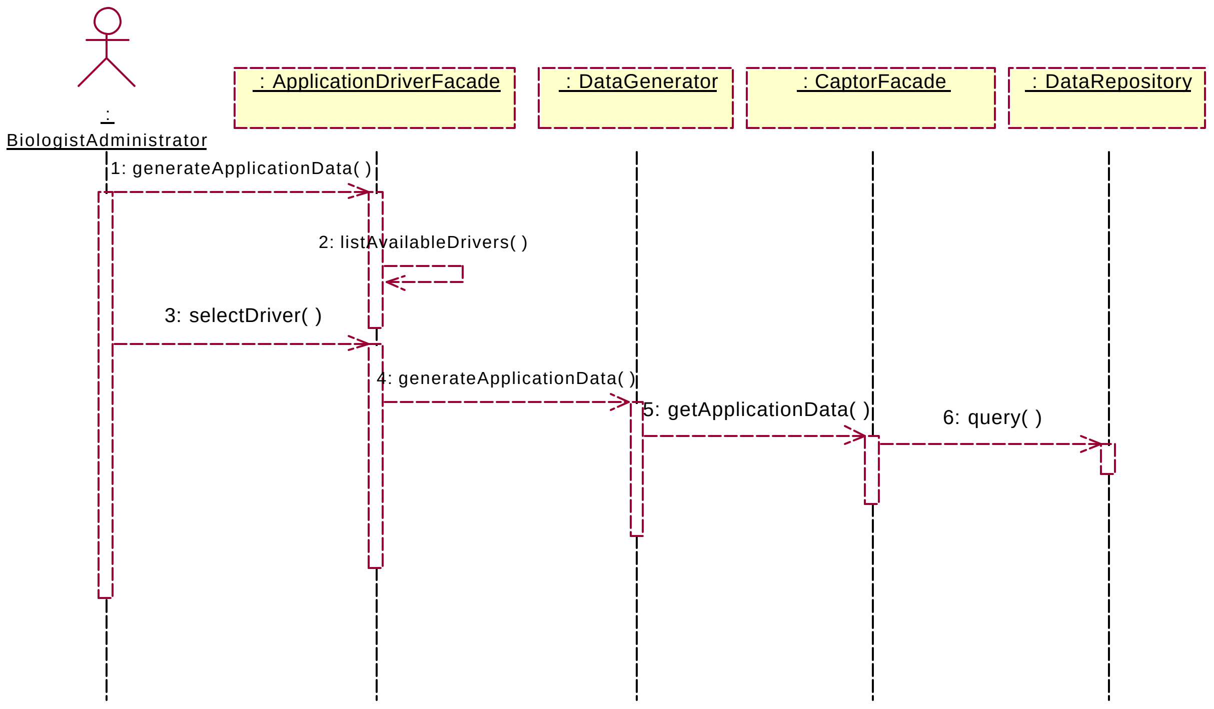

Figure 11 – Sequence Diagram of the functionality " Generate data for application"

5.6 Executing algorithm of the biology model

Finally, Figure 12 presents the sequence diagram for "Execute algorithm of the biology model". The user selects the algorithm to be executed, after they are presented, and that information is sent to the class, which represents the biology model, that triggers its execution. The algorithm runs with the data instances of the class associated to it.

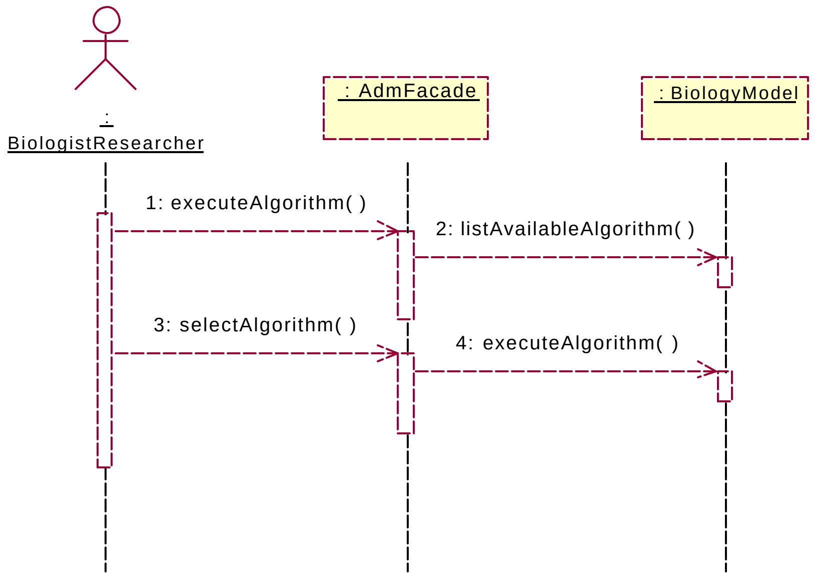

Figure 12 – Sequence diagram for "Executing algorithm instantiated in the biology model"

6. Final Comments

We proposed and detailed in this paper a genome database framework that integrates molecular biology data sources and allows the execution of programs and queries in a uniform way. This is quite different from most existing approaches, that lie on particular data models and structures, which are appropriate to specific contexts.

The main expected contribution is based on the idea that our framework works much like data warehouses do, but provide also flexibility and reusability. The users of such a tool may access a heterogeneous environment of information sources and can deal with schema evolution based on a meta-model, i.e., independent of each distinct data model used. New schemas can be built via framework instantiation, with the help of an ontology and a biology data model.

We are currently working on the framework implementation and we hope to have a prototype available soon, with all functionalities, although with restricted access to data sources. We are also interested in the definition and representation of a specific ontology for the molecular biology and genome area, which will be used in the existing data sources.

Acknowledgments: The authors would like to thank Wim Degrave and Antonio Miranda, researchers at FIOCRUZ Biology and Biochemical Department, for their help with applications requirements, and also Elvira Maria Antunes Uchôa, researcher at PUC-Rio, and José Antonio Fernandes de Macêdo, PhD candidate at PUC-Rio, for their contributions to this work.

References

[1] AceDB: http://genome.cornell.edu/acedoc/index.html

[2] S.F. Altschul, W. Gish, W. Miller, E.W. Myers and D.J. Lipman, "A basic local alignment search tool", Journal of Molecular Biology 215, 1990, pp 403-410.

[3] M. Ashburner and N. Goodman, "Informatics – Genome and Genetics Databases", Current Opinion in Genetics & Development 7, 1997, pp 750-756.

[4] P. Baker, C.A. Goble, S. Bechhofer, N.W. Patton, R. Stevens and A. Brass, "An Ontology for Bioinformatics Applications", Bioinformatics 15(6), 1999, pp 510-520.

[5] M.I. Bellgard, H.L. Hiew, A. Hunter, M. Wiebrands, "ORBIT: an integrated environment for user-customized bioinformatics tools", Bioinformatics 15(10), 1999, pp 847-851.

[6] G. Booch, J. Rumbaugh and I. Jacobson, "The Unified Modeling Language User guide", Addison-Wesley Longman, 1999.

[7] G. Booch, "Designing an Application Framework", Dr. Dobb´s Journal 19(2), 1994, pp 24-30.

[8] P. Buneman, S.B. Davidson, K. Hart, G.C. Overton and L. Wong, "A Data Transformation System for Biological Data Sources", Procs of VLDB Conference, 1995, pp 158-169.

[9] I.A. Chen and V.M. Markowitz, "An Overview of the Object Protocol Model (OPM) and the OPM Data Management Tools", Information Systems 20(5), 1995, pp 393-418.

[10] CPL/Kleisli: http://www.cis.upenn.edu/~db/home.html

[11] S.B. Davidson, C. Overton and P. Buneman, "Challenges in Integrating Biological Data Sources", Journal of Computational Biology 2(4), 1995, pp 557-572.

[12] R.F.Doolittle, ed. "Methods in Enzymology", Academic Press, 1990.

[13] Entrez: http://www.ncbi.nlm.nih.gov/Entrez/

[14] M.E. Fayad, D.C. Schmidt and R.E. Johnson, "Building Application Frameworks", Addison-Wesley, 1999.

[15] E. Gamma, R. Helm, R. Johnson and J. Vlissides, "Design Patterns: Elements of reusable object-oriented software", Addison-Wesley Longman, 1995.

[16] The Genome Database (GDB) Blast Search: http://www.gdb.org/gdb/seqBlast.html, 2000.

[17] GenBank: http://www.ncbi.nlm.nih.gov/Genbank/index.html

[18] V. Guerrinia and D. Jackson, "Bioinformatics and XML", On Line Journal of Bioinformatics, 1(1), 2000, pp 1-13.

[19] R. Karp, L. Ruzzo, M. Tompa, "Algorithms in Molecular Biology", http://www.cs.washington.edu/education/courses/590bi/96wi/, 1996.

[20]] P. Karp, "A Strategy for Database Interoperation", Journal of Computational Biology 2(4), 1995, pp 573-586.

[21] M. Lemos, "Memory Management for Sequence Comparison", MSc Dissertation (in Portuguese), Departamento.de Informática, PUC-Rio, August 2000.

[22] S. Lifschitz, L.F.B. Seibel and E.M.A. Uchôa, "A Framework for Molecular Biology Data Integration", to appear in Procs. Workshop on Information Integration on the Web (WIIW), April 2001.

[23] V.M. Markowitz and O. Ritter, “Characterizing Heterogeneous Molecular Biology Database Systems”, Journal of Computational Biology, 2(4), 1995, pp 547-556.

[24] J. Meidanis and J.C. Setúbal, “Introduction to Computational Molecular Biology”, PWS Publishing Company, 1997.

[25] F. Moussouni, N.W. Paton, A. Hayes, S. Oliver, C.A. Goble and A. Brass, “Database Challenges for Genome Information in the Post Sequencing Phase”, Procs 10th Database and Expert Systems Applications (DEXA), 1999, pp 540-549.

[26] National Center for Biotechnology Information (NCBI) Blast: http://ww.ncbi.nlm.nih.gov/BLAST/, 2000.

[27] T. Okayama, T. Tamura, T. Gojobori, Y. Tateno, K. Ikeo, S. Miyasaki, K. Fukami-Kobayashi and H. Sugawara, “Formal Design and Implementation of an Improved DDBJ DNA Database with a New Schema and Object-oriented Library”, Bioinformatics 14, 1998, pp 472-478.

[28] N.W. Patton, S.A. Khan, A. Hayes, F. Moussoni, A. Brass, K. Eilbeck, C.A. Goble, S.J. Hubbard, S. G. Oliver, “Conceptual modeling of genomic information”, Bioinformatics 16(6), 2000, pp 548-557.

[29] http://www.public.iastate.edu/research_tools.html

[30] L.F.B. Seibel, M. Lemos and S. Lifschitz, “Genome Databases” (in Portuguese), Procs. of the Brazilian Databases Symposium Tutorials, 2000, pp 514-553.

[31] Swiss-Prot: http://www.ebi.ac.uk/swissprot

[32] SRS: http://expasy.cbr.nrc.ca/srs5

[33] Tambis Project: http://img.cs.man.ac.uk/tambis/

[34] XML Schema: http://www.w3.org/XML/Schema

[35] XML: http://www.w3.org/XML/

[36] Xquery: http://www.w3.org/TandS/QL/QL98/pp/xquery.html