Computers in Biology and Medicine 42 (2012) 474–479

Contents lists available at SciVerse ScienceDirect

Computers in Biology and Medicine

journal homepage: www.elsevier.com/locate/cbm

LabSystem Gen, a tool for structuring and analyzing genetic data in histocompatibility laboratories

Luiz Cláudio Demes da Mata Sousa [a,*], Pedro de Alcântara dos Santos Neto [a], Fernando da Fonseca de Souza [b], Semiramis Jamil Hadad do Monte [c]

[a] Federal University of Piauí, DIE, Brazil
[b] Federal University of Pernambuco, CIN, Brazil
[c] Federal University of Piauí, LIB, Brazil

ARTICLE INFO

Article history:
Received 12 March 2010
Accepted 27 December 2011

Keywords:
Transplantation
Histocompatibility
Automation
Data analyses
Framework
Data crossing
Biological data

ABSTRACT

Analysis of HLA data involves queries on web portals, whose search parameters are data stored in laboratories' databases. In order to automate these queries, one approach is to structure laboratory data into a database and to develop bioinformatic tools to perform the data mapping. In this context, we developed the LabSystem Gen tool that allows users to create a Laboratory Information System, without programming. Additionally we implemented a framework that provides bioinformatic tools, transparent access to public HLA (human leukocyte antigen) information resources. We demonstrated the LabSystemGen system by implementing BMDdb, which is a LIMS that manages data of recipients and donors of organ transplant.

1. Introduction

The molecular age provided biomedical researchers with so much information that nowadays one of the problems that biologists face is a too large data set to be fully analyzed. This scenario allowed the development of robust computational programs designed to manage such biologic data sets, an approach that bridged biomedical and computational sciences. The above partnership allowed tremendous advances in the field of immunogenetics. Immunogenetics deals with the Major Histocompatibility Complex (MHC), considered one of the most polymorphic and polygenic sequences known [1,3]. Since HLA differences lead to an alloreactive immune response, transplant program's physicians attempt to match donor and recipient HLA at a molecular level. It is imposed upon histocompatibility laboratories the need for using different resources, including bioinformatic tools, public domain databases and HLA data stored in local repositories.

Small-sized histocompatibility labs generally use spreadsheets to manage data because they are easy to use and do not require researchers to have deep technical knowledge on computer sciences. However, data version control and data sharing are challenging due to the lack of a centralized management system. In this scenario, searching the data requires meticulous hand trimming and picking of data sets from multiple MS-Excel [4,5] sheets and information systems. Such searches are becoming more and more difficult to be controlled as the number of donors and recipients increases.

A Laboratory Information Systems Manager (LIMS) is a good solution for storing a large amount of centralized and accessible data. Furthermore, to store genetic data into relational databases allows automated and complex analysis to be performed by bioinformatic tools. However, the lack of a standardized communication between tools and genetic databases requires complex and expensive integration solutions. Therefore, the researcher needs to act as a bridge (ad hoc) between the various tools and data sources in order to perform their analysis, manipulating several temporary files and copying data from one location to another [6].

In order to solve these difficulties in histocompatibility labs, we proposed the user-friendly LabSystem Gen. Its model driven approach lets individual users to create their own LIMS through graphical specification of models. The LIMS generated by the tool has the following features: (i) it allows complex biological data analysis, by automated tools, involving public and private databases; and (ii) it provides special biological database types to facilitate studies related to HLA.

* Correspondence to: Universidade Federal do Piauí/DIE SG-9, Av. Nossa Senhora de Fátima, s/n Ininga, Teresina-PI 64048-901, Brazil. Fax: +55 86 3215 5690.
E-mail addresses: claudio.demes@ufpi.edu.br (L.C.D.M. Sousa), pasn@ufpi.edu.br (P.d.A.d. Santos Neto), fdfd@cin.ufpe.br (F.d.F.d. Souza), semiramis@ufpi.edu.br (S.J.H.d. Monte).

0010-4825/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compbiomed.2011.12.016

2. eDAframework

The eDAframework is a framework developed in our work to provide applications a simplified access to local databases and public resources on the weband to perform the transformation described in Section 2.1. The framework is composed of three major modules: (1) transformation module (Designer tool); (2) Data Access module; and (3) importing and exporting module.

2.1. Designer tool

The Designer is a knowledge management tool developed in Object Pascal language [7] in order to allow researchers to design data and application models. The data model includes a domain-specific data model describing the entities (tables) and relationships (associations between tables) that the scientist wants to manage. The application model describes properties that allow the scientist to customize the look and feel of the LIMS graphical user interface. Data and application models are manually designed by researchers using the functionalities provided by the Designer tool. However, as a way of speeding up development, data models can be generated from csv files (comma-separated values files); spreadsheets (requires a special format), or from a relational schema (reverse engineering). In the current implementation, only MySQL [8] and Access [2] relational schemas can be read, but the extension to support other databases is simple in the proposed architecture.

A model in Designer consists of a set of named forms associated with each other in three possible ways: one-to-one, one-to-many and many-to-many. Each form instance is associated with a set of *field-objects*. Each *field-object* has properties and a value, that can be: (i) a primitive data type such as String, Integer, Boolean, etc., (ii) a reference to a list of values, (iii) a reference to another form, or (iv) A *bio data type*, such as a nucleotide sequence (ftNucleotide data type), an amino acid sequence (ftAminoacid data type) or an allele code (ftHLA data type).

The properties set of a *field-object* determine its appearance, behavior and how it is linked to other *field-objects*. By changing their properties, the researcher determines how data will be stored in the relational database and the look and feel of the GUI.

A Designer Model form may contain a table associated with it and tables associated with its fields. Additionally, forms may be associated to simulate a *master-detail* form. In this case, the researcher should indicate which forms are details and which form is master. The Designer automatically creates a primary key in the table associated with the master form and a foreign key in the tables associated with the detail forms, in a one-to-many association (Fig. 1—*master screen* area).

This Designer model differs from an entity-relational (ER) model [9] by not using technical terms or ER diagrams that are too complex to researchers. Instead, *field-objects* are added to *forms* and have their properties set determining both the appearance and behavior of the application and the schema of the relational database.

Fig. 2 depicts the Designer's main window showing two *forms* of the Bone Marrow Donor Project's data and application model (BMDdb). The Object inspector floating tool bar (right of the figure) allows user access to *field-objects*' more complex properties, if needed.

A LabSystem Gen application stores its experiment data into a relational database, and it is important to notice that each project developed using Designer will generate a different database schema to handle its data. Any given laboratory or researcher group will have its own specific data model, which is transformed into a specific relational schema. The method that performs the transformation of a model into a specific relational schema is an implementation of the algorithm for ER-to-relational mapping [10]. The differences are only in the terminology: instead of entities, attributes and associations between entities, the Designer provides *forms*, *field-objects* and *associations* between *forms*. Regarding *bio data types*, they are mapped as String database type into a relational schema.

The algorithm for ER-to-relational mapping [10] maps sets of entities and their attributes for relations and columns, respectively. Furthermore, associations with cardinality many-to-many between entities and multivalued attributes are mapped as relations. The eDAFramework uses this algorithm to map *forms* and associations between *forms* to relations, and to map *field-objects* to attributes or relations. *Bio data* types are mapped to String-type columns in the relation generated for the *form* into which it is inserted. Metadata referred to the *bio data* types are stored into a table of the System called *Biologicalfields_table*. These metadata (URL, files, FTP directories, fields, etc.) are used by the eDAframework in recovering the data stored into public databases. Information about appearance, positioning, behavior, and cardinality of *forms*, *field-objects* and relations are stored in three metadata tables of the System called *forms*, *field-objects* and *associations*. If a *field-object* is associated to anprimitivedata type, it is mapped to a column of the relation mapped from the *form* into which it is inserted, but if the *field-object* is associated to a list of values or to another project form, the System generates a relation for the *field-object* and associates it to the relation generated for the *form* into which the *field-object* is inserted (association one-to-many).

The algorithmis implemented in Object Pascal language and PHP [11]. It takes a Designer Model file as input, and outputs a text file with the relational schema in SQL [12]statements. Such a file is then executed by the relational database engine to create the database schema containing the data and metadata tables of the System. The System's metadata tables are queried by the Web

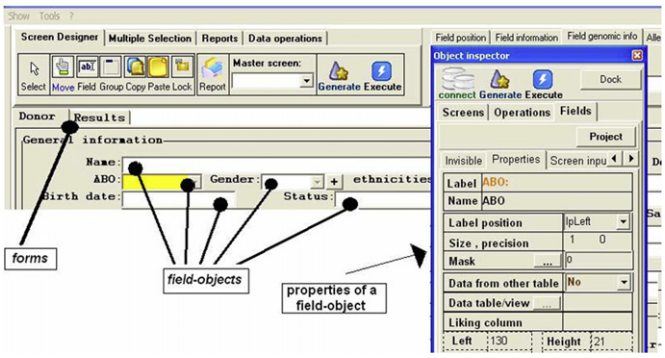

Fig. 1. The Designer's main window.

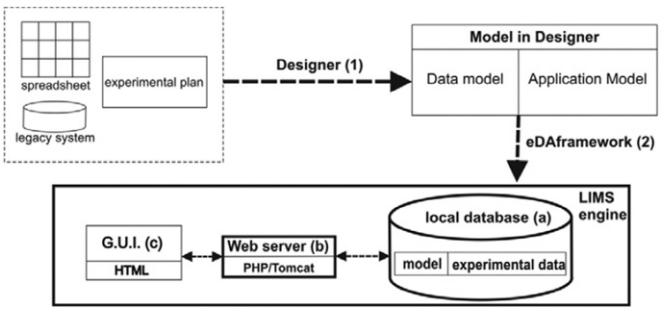

Fig. 2. Overview of the LabSystem Gen architecture.