Vol. 20 no. 13 2004, pages 2075–2083
doi:10.1093/bioinformatics/bth206

Molecular Genetics Information System (MOLGENIS): alternatives in developing local experimental genomics databases

Morris A. Swertz[1,*], E. O. (Bert) de Brock[1,2], Sacha A. F. T. van Hijum[3], Anne de Jong[3], Girbe Buist[3], Richard J. S. Baerends[3], Jan Kok[3], Oscar P. Kuipers[3] and Ritsert C. Jansen[1]

[1]Groningen Bioinformatics Center (GBIC), Faculty of Medical Sciences and Faculty of Mathematics and Natural Sciences and [2]Faculty of Management and Organization, University of Groningen, P.O. Box 800, NL-9700 AV Groningen, The Netherlands and [3]Molecular Genetics, Groningen Biomolecular Sciences and Biotechnology Institute, University of Groningen, P.O. Box 14, NL-9750 AA Haren, The Netherlands

Received on October 15, 2004; revised on January 5, 2004; accepted on March 8, 2004
Advance Access publication April 1, 2004

ABSTRACT

Motivation: Genomic research laboratories need adequate infrastructure to support management of their data production and research workflow. But what makes infrastructure adequate? A lack of appropriate criteria makes any decision on buying or developing a system difficult. Here, we report on the decision process for the case of a molecular genetics group establishing a microarray laboratory.

Results: Five typical requirements for experimental genomics database systems were identified: (i) evolution ability to keep up with the fast developing genomics field; (ii) a suitable data model to deal with local diversity; (iii) suitable storage of data files in the system; (iv) easy exchange with other software; and (v) low maintenance costs. The computer scientists and the researchers of the local microarray laboratory considered alternative solutions for these five requirements and chose the following options: (i) use of automatic code generation; (ii) a customized data model based on standards; (iii) storage of datasets as black boxes instead of decomposing them in database tables; (iv) loosely linking to other programs for improved flexibility; and (v) a low-maintenance web-based user interface. Our team evaluated existing microarray databases and then decided to build a new system, Molecular Genetics Information System (MOLGENIS), implemented using code generation in a period of three months. This case can provide valuable insights and lessons to both software developers and a user community embarking on large-scale genomic projects.

Availability: http://www.molgenis.nl

Contact: m.a.swertz@cs.rug.nl

*To whom correspondence should be addressed.

INTRODUCTION

The design and implementation of information systems for experimental genomics research raises many challenges: analytical methods such as microarrays and mass spectrometry are under continuous development, a growing variety of research topics are addressed using such methods, and modern high-throughput experiments result in an increasing amount of data files and tools for processing these data. This paper summarizes the requirements for such experimental genomics databases and discusses design alternatives for dealing with these requirements in a sensible manner. The insights presented will enable 'wet' researchers and 'dry' software engineers and bioinformaticians to take better decisions regarding genomics experiment information management in their respective situations.

This is illustrated using the case of Molecular Genetics Information System (MOLGENIS), a local microarray database. The first version of MOLGENIS was developed to support the transcriptomics research workflow of the Molecular Genetics group of the University of Groningen (Fig. 1). In this group more than 20 people with varying computer skills produce and experiment with amplicon-based DNA microarrays from several bacterial species, e.g. *Lactococcus lactis*, *Bacillus subtilis* and *Streptococcus pneumoniae* (Kuipers *et al*., 2002). Versions for genomics research groups working on plants, animals and humans, using different microarray platforms, are currently under development, and proteomics extensions for mass spectrometry experiments are being considered. MOLGENIS was implemented using code generation to speed up development and maintenance of (multiple) custom microarray databases in fast-developing experimental genomics domains.

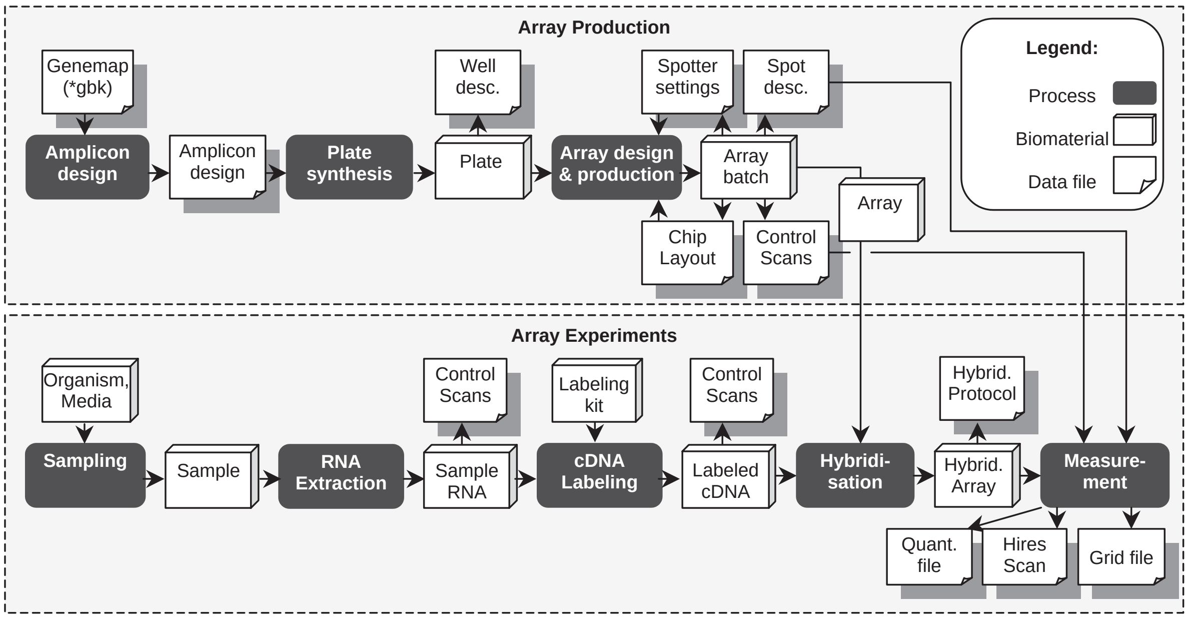


Fig. 1. Overview of the production and experiment workflow that the MOLGENIS system has to support. This figure can be viewed in colour on *Bioinformatics* online.

The decision to develop our own system instead of using an existing one was based on our specific needs. Frequently used arguments in favor of using an existing system instead of developing a new one are (1) it is often cheaper; (2) the system will be available faster; (3) one is ensured of software support and updates; and (4) the technology is proven. Typical arguments in favor of developing one's own system instead of buying a system are (1) the system can be tuned and adapted to meet specific needs and (2) there is no vendor lock-in, i.e. becoming dependently on the software vendor. The Molecular Genetics group decided on the development of a tuned system, given the risk of high costs for adaptation of an existing system (Table 1) to their specific requirements, and the low fixed-price agreed on by the systems development group: only a simple server and the equivalent of 4 person months work for a computer scientist.

REQUIREMENTS ANALYSIS AND DESIGN ISSUES

Close collaboration between the systems development group and several genomics research groups resulted in identifying the general requirements that determine the shape and structure of genomics experiment management systems. Strong interactions with the molecular genetics laboratory detailed these requirements for the MOLGENIS system. The requirements were the following: (R1) evolution ability, (R2) suitable data model, (R3) suitable treatment of data files, (R4) exchange with various software tools and (R5) low maintenance costs. For each requirement, the following topics are addressed: (i) design issues, (ii) alternative solutions, (iii) the (dis)advantage(s) of each solution and (iv) the solution chosen for the MOLGENIS case study as well as for existing software implementations (summarized in Table 1).

(R1) Evolution ability

The experimental genomics research domain is a fast-developing domain characterized by a variety of research topics, high-throughput analytical methods and accompanying datasets, exchange formats and (pre)processing software. Designers of experimental genomics databases have to anticipate diversity between groups and unpredictable changes in requirements. For example, MOLGENIS had to support frequent data model changes, modifications of the user interface and addition of reports based on complex queries if e.g. new research findings, method improvements and collaboration projects, required it. The question therefore arose of how to deal with such evolving requirements. It meant that (r)evolution ability should be the central quality for the software design and implementation process.

Evolution ability (or modifiability) can be defined as the capability of modifying a system quickly and at a low cost to keep up with changing functional requirements. Hand-coded information systems typically show a low evolution ability because specific functionality is scattered throughout the hand-coded packages. For instance, the simple addition of an extra annotation field (e.g. a 'tissue' attribute) requires costly, bug-prone and time-consuming changes in the database, user-interface programs and application logic software modules.

Table 1. This table summarizes the design details of existing systems for fulfilling our general requirements of experimental genomics information systems: (R1) evolution ability, (R2) customized data model, (R3) suitable treatment of data files, (R4) exchange with software tools and (R5) low maintenance user interface

Name	Provider	R1		R2								R3		R4		R5		Other features					Literature	URL (valid till February 2004)
		Code generation	Configuration options	Array design	Array production	Plate management for prod.	Sample information	Experiments	User authentication	Userrole based security	Projects	Black box dataset storage	Decomposed dataset storage	External data processing	Embedded data processing	Low maintenance GUI	Requires client installation	Local server installation	Free for academics	Uses free DBMS	MIAME supportive	MAGE-ML exports		
MOLGENIS	University of Groningen	•		•	•	•	•	•	•	•	•	•		•		•		•	•	•	•	p	This paper	http://www.molgenis.nl
ArrayDB	Nat. Human Genome Res. Inst.			•				•	•	•			•		•	•		•	•				Ermolaeva *et al.*, 1998	http://genome.nhgri.nih.gov/arraydb/
ArrayExpress	European Bioinformatics Inst.	○		•			•	•	•				•	•		•			•	•	•	•	Brazma *et al.*, 2003	http://www.ebi.ac.uk/arrayexpress/
BASE	Lund University			•	•	•	•	•	•	•			•		•	•		•	•	•	•	p	Saal *et al.*, 2002	http://base.thep.lu.se/
GeneDirector	BioDiscovery		•	•	•	•	•	•	•	•	•		•		•		•	•			•			http://www.biodiscovery.com/genedirector.asp
GeNet	Silicon Genetics			•			•	•	•	•	•		•		•		•	•			•			http://www.silicongenetics.com/
GeneTraffic	Iobion			•			•	•	•	•	•		•		•	•		•		•	•			http://www.iobion.com/products/products.html
GEO	Nat. Center for Biotech. Inform.			•			•	•	•			•		•		•			•	?	•		Edgar *et al.*, 2002,	http://www.ncbi.nlm.nih.gov/geo/
GeneX-Lite	National Center for Genome Resources	○		•			•	•					•		•	•		•	•	•	•			http://www.ncgr.org/genex/
LAD	Stanford University			•	•	•	•	•	•	•			•		•	•		•	•	•	•		Killion *et al.*, 2003	http://www.longhornarraydatabase.org
MADGE	Medical College of Georgia			•	•	•	•	•			•		•	•		•		•	•	•			McIndoe *et al.*, 2003	http://www.genomics.mcg.edu/niddkbtc/Software.htm
MaxdSQL	Manchester University			•			•	•	•				•	•		•		•	•	•	•	•		http://bioinf.man.ac.uk/microarray/maxd/
M-CHIPS	German Cancer Res. Centre			•	•		•	•	•				•		•	•			•	?	•		Fellenberg *et al.*, 2002	http://www.dkfz.de/tbi/services/mchips/
Partisan	Clondiag		•	•	•	•	•	•	•	•	•		•		•	•		•			•	p		http://www.clondiag.com/
QuickLIMS	German Cancer Res. Centre			•	•	•							•	•			•	•	•	•			Kokocinski *et al.*, 2003	http://www.dkfz.de/kompl_genome/Other/quicklims/
RAD/GUS	University of Pennsylvania			•	•	•	•	•	•	•	•		•		•	•		•	•		•	•	Stoeckert *et al.*, 2001	http://www.cbil.upenn.edu/RAD/
Resolver	Rosetta Bio		•	•			•	•	•	•	•		•		•		•	•			•			http://www.rosettabio.com/
SMD	Stanford University			•	•	•	•	•	•	•			•		•	•		•	•		•		Sherlock *et al.*, 2001	http://genome-www5.stanford.edu/
TM4/MADAM	The Institute for Genomic Res.			•	•		•	•					•		•		•	•	•		•	p	Saeed *et al.*, 2003	http://www.tigr.org/software/tm4/
YMD	Yale University			•	•	•	•	•	•	•	•		•	•		•			•		•		Cheung *et al.*, 2002	http://www.info.med.yale.edu/microarray/

The information comes from websites, brochures, demos, review literature (Anderle *et al.*, 2003; Gardiner-Garden and Littlejohn, 2001) and system specific literature.
Closed circles: chosen; open circles: partial; p: planned; query symbol: unknown.

To limit the need for labor-intensive reprogramming, one could add configuration options with which functionality can be enabled or disabled at runtime, e.g. the specific annotations for human samples can be enabled or some user-defined query can be added. This offers some evolution ability but only for those changes predicted by the system designers.

Finally, one can invest in code generation in which metadata descriptions are used to (re)program the system automatically. This results in short and low-cost change-to-production cycles without the need for a programming team. Then, to modify the system, only a change in metadata is needed because the generator takes care of recoding the system. A positive side-effect of generation is the uniformity of the generated application, a feature highly appreciated by the user, because all design rules are made explicit within a generator.

For MOLGENIS it was decided to invest in such a generator-based approach. Only a few databases address explicitly and solve the novel issue of evolution ability (R1; Table 1).

(R2) Suitable data model

Experimental genomics research projects using the same analytical methods have similar information needs. Therefore, standardization efforts emerged to simplify data exchange between such projects through definition of standard data models and formats. Well known and important standardization proposals such as MIAME, MAGE and PEDRo (http://www.mged.org, Brazma *et al*., 2001; Spellman *et al*., 2002; Stoeckert *et al*., 2001; Taylor *et al*., 2003) are by design of a general and comprehensive nature, while each genomics research group has unique and specific information requirements, e.g. research focusing on a human medicine has different annotation requirements compared with research on crop modification. In particular, MOLGENIS had to support traceable administration and management of data and materials describing design and production of DNA microarrays, sampling and experiments on bacterial species, and the organization of research into projects. The data concerning these projects had to be stored securely because some of the projects are with industrial partners, i.e. the data model should offer support for secure multi-user collaborations, authentication and group permissions. The main question is whether such (maturing) exchange standards should be used (unchanged) for the local experiment information management or a customized data model should be developed.

It is recognized that it is impossible to create one model that is completely expressive of the variety of microarray-based research (Spellman *et al*., 2002). Therefore, given the focus on exchange, most standards make parts of their models optional and provide extendible annotations such as name–value-type tuples to store extra data keyed by name and type (also known as triplets in MIAME) or generic annotation and description objects (MAGE). This way information irrelevant to a specific context can be left out and relevant entities that are lacking can be added. These reduction and extension possibilities are very valuable for data exchange to ensure that the specifics of all microarray facilities fit into the standard format. However, they are inadequate for data management because constraints are too loosely defined. For example, an object describing ‘temperature’ can be stored, optionally, via a triplet, while the specific research context requires this annotation to be enforced for every sample. Thus, standard models will have to be restricted, adapted and extended into a customized data model that enforces these local requirements explicitly while capitalizing on the valuable foundation of accumulated domain knowledge of standardization organizations like MGED.

For MOLGENIS, given the bacterial research context of its first implementation, some information from the standards was not applicable and has therefore been left out (e.g. tissue), some information that fit the standard but only as optional or as an ‘extensible annotation’ was made required (e.g. experiment factors like growth medium, temperature, rpm of shaker, stress) and some information that were missing altogether was added (e.g. project, plate management and user role-based security). Finally, the complexity of the model was reduced where possible to make the model more comprehensible: for instance, a sample or a sequence was represented as one table row instead of a complex of objects. As summarized in Figure 2, the system contains a total of 22 tables (collections of objects, e.g. ‘samples’) and about 90 attributes (the atomic information elements of a table, e.g. ‘sampling date’). Existing systems were compared in general terms with regard to the specific requirements of the Molecular Genetics group (R2; Table 1).

(R3) Suitable treatment of data files

High-throughput analytical methods for genomics research generate (large) data files of heterogeneous sorts and formats. For example, MOLGENIS had to store data reliably from a dozen file types describing, among others, DNA microarray designs, spotter settings, scan images and expression quantifications. The main question is whether such a file should be treated as a black box and stored as a whole, the black box option, or should be decomposed into individual data elements upon storage in the database, the decomposition option. In addition, in the case of the first option, should files be stored inside the database or stored in some directory structure to be referred to by the database? The answer may differ depending on the dataset.

The advantages of the black box option are the flexibility and simplicity of the solution: no development effort is needed to support specific and/or future formats. For most purposes, this option is sufficient because the input data are directly readable by the next device or analysis software in the DNA microarray production and experiment workflow, i.e. without format transformations. The disadvantage of the black box option

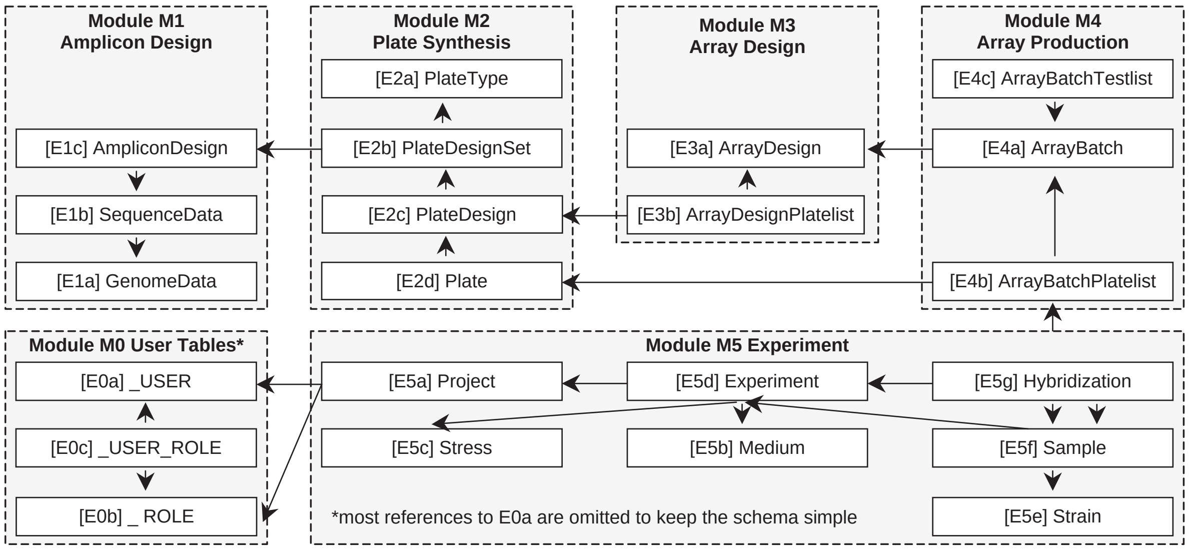


Fig. 2. MOLGENIS data model. Arrows indicate foreign key references. There are, besides an authorization module (M0), five functional modules available: (M1) Amplicon Design, to manage the sequences to be spotted and the genomes used to design them; (M2) Plate Synthesis, to produce and manage biomaterial plates of purified amplicons; (M3) Array Design, to store physical array properties as used by the spotter, and the plates of reporter sequences to be used; (M4) Array Production, to manage batch array production including the plates used and; (M5) Experiment Management, to organize hybridizations and samples into experiments and projects including lists of obligatory annotations. This figure can be viewed in colour on *Bioinformatics* online.

is that the data items inside the file are not directly searchable, which is the advantage of the decomposition option. The severe disadvantages of decomposition are its laboriousness and format specificity: a decomposer and a recomposer must be developed for each device and its associated file format and possibly it needs adaptations to accommodate future format changes depending on the software version.

The options for dealing with datasets are not mutually exclusive: one could decompose (a part of) the dataset into the database and store the file as a black box as well. However, in such mixed options, consistency between the file and its decomposition might well become a problem.

To ensure flexibility and low-maintenance costs the MOLGENIS system currently treats all files as black boxes and stores them in a protected directory structure to ensure MOLGENIS's user/project-based security rules. For example, the array designs are stored in three related files describing the spotter settings, physical layout and grid layout. Most existing systems employ the decomposition option on all datasets (R3; Table 1). This is probably because systems featuring the decomposition option also have embedded processing tools for which centralized decomposition on data entry is more convenient (which also shows that the issues presented here cannot be decided in isolation).

(R4) Exchange with software tools

Software is indispensable for manipulating experimental genomics datasets. Automatic launch of—and persistent data exchange with—these programs from within the experimental genomics database is required for reducing repetitive data management activities. For example, MOLGENIS had to exchange data with dedicated software tools in a flexible manner in order to make it possible to use multiple and/or new tools interchangeably without recurring development costs. These included tools for configuring laboratory devices (e.g. slide spotter), data visualization and manipulation (e.g. amplicon selection and design, scan image quantification) and data (pre)processing (e.g. normalization of expression data). The main question is whether to embed seamlessly such software tools programmatically into the system or accept a form of less-than-seamless integration using file-based data exchange with those tools.

Seamless embedding demands a perfect fit between the program and experiment database with regard to design and implementation, e.g. the programming language, GUI toolkit. Existing tools rarely fit the database system, which makes software embedding so time-consuming and expensive that people often completely redevelop the tool. Furthermore, this laborious embedding process has to be repeated for each new tool, and maybe even version, making the database system inflexible and maintenance costly (that can be reduced if there exists some standard interface, often labeled 'plug-in'). Alternatively, quite high levels of user-perceived integration can be achieved by using some form of less-than-seamless integration such as using the operation system feature to start software automatically based on a specific file type or extension. This form of tool integration ensures flexibility without

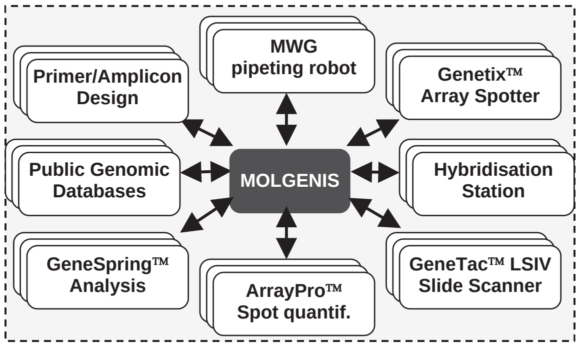


Fig. 3. MOLGENIS exchanges data with several classes of software tools without locking the researcher to a specific application or version. This figure can be viewed in colour on *Bioinformatics* online.

high design and maintenance costs, and only requires additional user interventions when (re)storing the data into the database.

For MOLGENIS this flexibility was preferred because it made it possible for the researchers to apply a best-of-breed approach when using tools without high development and maintenance costs (Fig. 3). For instance, hyperlinked Hybridization scans (*.tif) can be opened by the preferred commercial spot quantification software, and hyperlinked Genome Primer (van Hijum *et al*., 2003) amplicon design files and quantification files (*.xls) can be opened by the locally developed MicroPrep normalization software (van Hijum *et al*., 2004) (Fig. 4). The data is written back into MOLGENIS through file uploads. Many existing systems have developed their own tightly integrated solutions through (re)creation of tools, offering functions like visualization and quantification, while others are limited to data management only (R4; Table 1).

(R5) Low user interface maintenance costs

The costs of bringing the experimental genomics database to the researcher workbenches, typically distributed over multiple sites or even in multiple organizations, should be minimal during the complete lifespan of the system. For example, the molecular genetics group required low maintenance costs besides low acquirement costs (for system and DBMS). While server maintenance is covered above, the question remains as to how to lower client maintenance and administration costs.

There are several complicating factors that can make client administration and maintenance a big part of the 'total costs of ownership'. First and foremost, the collection of PC clients is often of a heterogeneous nature, e.g. various versions of Microsoft Windows, Linux and other systems. Furthermore, the experiment management system will need to be accessed across departmental borders and thus by systems not within the control of departmental computer administrators. An ideal situation would be (1) platform-independent front-end software that (2) does not require maintenance when the system is changed and (3) is available automatically for new clients when the database is accessed for the first time, to relieve the burden for the departmental computer administrators.

The first requirement can be met by using a platform-independent layer such as the Java virtual machine or .Net software. The current web browser software can also be regarded as such a layer. In addition, automated maintenance and deployment via the Internet is available for all these platforms, satisfying the second requirement, e.g. Java Web Start. However, the third requirement is still a problem with most of the above solutions because the platform software needs to be installed and maintained for each client before he or she can access the database system. Usually, only web browsers are installed for every client, which makes the web platform a good candidate for developing front-end applications (unless more advanced capabilities are needed, e.g. for visualization).

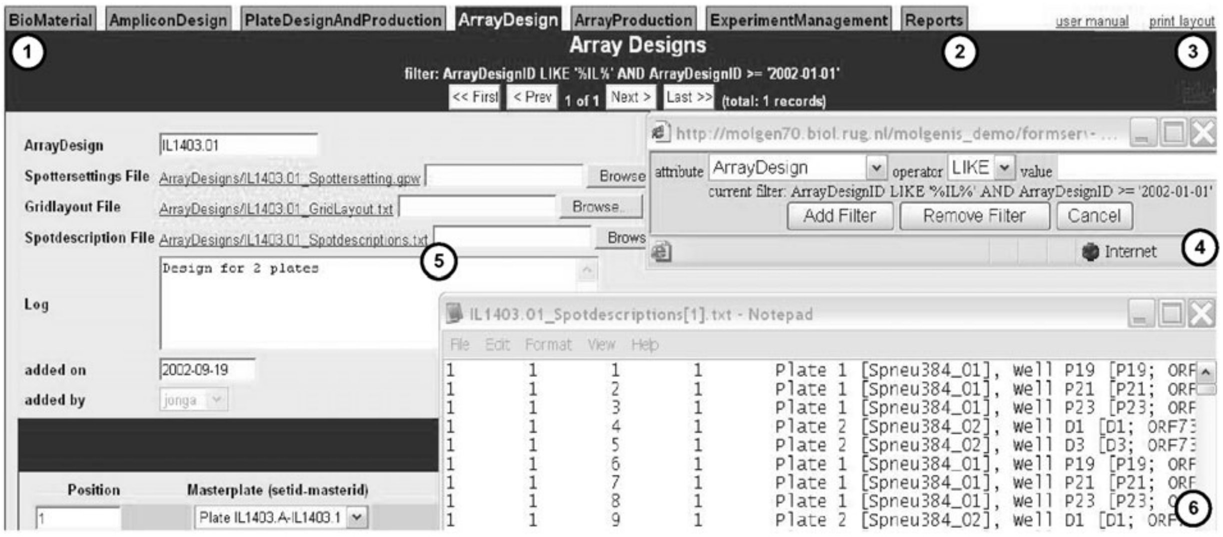
MOLGENIS uses a web user interface utilizing HTML 4.0, CSS and JavaScript capabilities. As Figure 4 illustrates, all functionality is quickly accessible via a tab menu at the top of the screen, with options for printable layout, flexible attribute based search/filtering, hyperlinks to datasets (as described above) and a report section in which the results of full-fledged, administrator-defined SQL queries can be rendered (e.g. 'how many slides were used per project' or 'list hybridizations by species'). Most existing microarray databases use Java and/or the web technology (R5; Table 1).

IMPLEMENTATION

The systems development group consisted of two computer scientists, both with a background in information system development but without an in-depth background in biology. The requirements analysis, design and implementation were completed in a period of 3 months and took about 4 person months of work. Prototypes of (parts of) the system were delivered to the researchers in an early stage of the project in order to trigger detailed feedback in terms of needs and requirements. The choice for code generation, implemented using the Invengine software (http://www.inventory.nl), allowed for rapid application development because it only required the definition of application metadata, i.e. data model, screen descriptions and layout rules. The subsequent MOLGENIS versions are generated from these meta models.

How it works

Invengine is implemented using open industry standards: it requires an SQL92 standard relational DBMS for data storage, uses XML, DOM and XSLT (http://www.w3c.org) for metadata definition and generation and has an Apache web



Array Designs

filter: ArrayDesignID LIKE '%IL%' AND ArrayDesignID >= '2002-01-01'

1 of 1 (total: 1 records)

ArrayDesign	IL1403.01
Spottersettings File	ArrayDesigns/IL1403.01_Spottersetting.gpw
Gridlayout File	ArrayDesigns/IL1403.01_GridLayout.txt
Spotdescription File	ArrayDesigns/IL1403.01_Spotdescriptions.txt
Log	Design for 2 plates
added on	2002-09-19
added by	jonga

Position	Masterplate (setid-masterid)
1	Plate IL1403.A-IL1403.1

http://molgen70.biol.rug.nl/molgenis_demo/formserv- ...

attribute ArrayDesign operator LIKE value

current filter: ArrayDesignID LIKE '%IL%' AND ArrayDesignID >= '2002-01-01'

IL1403.01_Spotdescriptions[1].txt - Notepad

```
1    1    1    1      Plate 1 [Spneu384_01], well P19 [P19; ORF
1    1    2    1      Plate 1 [Spneu384_01], well P21 [P21; ORF
1    1    3    1      Plate 1 [Spneu384_01], well P23 [P23; ORF
1    1    4    1      Plate 2 [Spneu384_02], well D1 [D1; ORF73
1    1    5    1      Plate 2 [Spneu384_02], well D3 [D3; ORF73
1    1    6    1      Plate 1 [Spneu384_01], well P19 [P19; ORF
1    1    7    1      Plate 1 [Spneu384_01], well P21 [P21; ORF
1    1    8    1      Plate 1 [Spneu384_01], well P23 [P23; O
1    1    9    1      Plate 2 [Spneu384_02], well D1 [D1; ORF
```

Fig. 4. MOLGENIS user interface: (1) navigation tabs provide quick access to all microarray information and (2) to SQL reports; (3) the layout can be switched to make the contents printable; (4) a filtering dialog offers user-definable dataset filtering; (5) datasets are hyperlinked so that they can be viewed by (6) the preferred locally installed software. This figure can be viewed in colour on *Bioinformatics* online.

server with PHP extension for the server side application generation (http://www.apache.org). Invengine uses three separate non-redundant metadata description files, which are used to generate an SQL back-end and to modify the application logic (Fig. 5):

(1) The Schema file, an XML file, defines the information structure. Addition of a table or an attribute would need changes in this file only.
(2) The GUI file, an XML file, defines the user interface and reports structure based on the Schema information. Addition of a form, a menu, or a report driven by a complex SQL query or changing navigation order would need changes in this file only.
(3) The Layout file, an XSLT file, defines how an application will be presented to the researcher, completely independent of the above. Change of corporate colors or the decision that date fields need a 'calendar control' would need changes in this file only.

The SQL relational database definitions for, currently, MySQL and PostgreSQL are generated using an XSLT transformation by which an XSL template (mapping the Schema XML to SQL) is applied on the Schema file. At runtime the Invengine software builds user screens (such as that in Fig. 4) and handles user events using the Schema, GUI and Layout files. The production server hardware consists of a Pentium 4 with 300 GB of hard disk space running a Mandrake Linux distribution and a PostgreSQL DBMS. Maintenance efforts are negligible. Code generation, also called model-driven generation, is considered part of the software engineering craft (e.g. http://www.codegeneration.net).

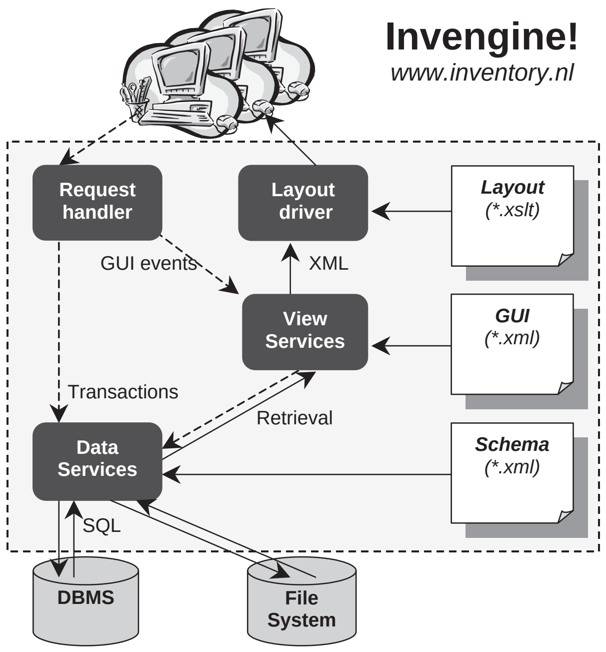


Fig. 5. Detailed architecture of the Invengine framework, which provides the logic that powers MOLGENIS. At runtime the XML metadata files are interpreted by PHP software modules: (1) a Request Handler translates HTTP user requests to database transactions or screen navigation events; (2) Data Services, generated from the Schema file, define and execute SQL92 and file system operations; (3) View Services, generated from the GUI file, build XML representations of the GUI screens; and (4) a Layout (UI) driver generates default and printable HTML layouts from the XML-defined screens using the Layout file (a layout driver to Java is possible). This figure can be viewed in colour on *Bioinformatics* online.

DISCUSSION

This paper discusses considerations concerning information management, software architecture and software engineering within the context of experimental genomic database systems. In particular, the general problem of variability and evolution is emphasized, and approaches to easing this problem are presented. The decisions cannot be taken in isolation, e.g. the decision how to integrate tools may well influence the decision on how to deal with datasets.

The development of a genomic experiment information system is not a trivial task. It requires collaboration between biologists, bioinformaticians and IT specialists. The information systems developer has to master the genomics domain to a certain level of detail, which requires much more effort than for many other application areas. Nevertheless, the MOLGENIS system was developed and tuned to the needs of genomic researchers within a period of 3 months, requiring a low budget (4 person months work, simple server) and resulting in a customizable system.

Very short prototyping cycles were enabled by the decision to use code generation technology to separate clearly the functional model from its implementation, using metadata to parameterize differences. In addition, as expected, these adaptations still go on, following new insights of researchers and new research questions. Moreover, flexibility and simplicity were gained by limiting MOLGENIS to data management only, while leaving the file decomposition and processing tasks to dedicated software. This ensures that MOLGENIS does not need to be adapted to new (versions of) data formats and tools. A plug-in system will ease the embedding processing capabilities if desired because standardization of the experiment process and a growing user community will make seamless tool development more feasible. Plug-ins for quality control, normalization and MAGE-ML exports are planned.

Projects have been initiated to adapt the MOLGENIS data model beyond bacterial DNA microarrays (building on the valuable MGED standards), producing suitable models for storage and analysis of data from medical, animal and plant transcriptomes as well. Furthermore, use of the model-driven code generation approach to experiment management is being considered for several proteomics research groups working with high-throughput liquid chromatography and mass spectrometry experiments (HPLC/MS/MS) within the University of Groningen. Finally, these insights extend to other areas as well, e.g. to biotechnology information and knowledge grid-like systems, such as the E.U. BioGRID project (BioGRID, http://www.bio-grid.net), that have to deal with similar variability and evolution problems. We are convinced that the insights discussed in this paper are also applicable to types of experiments other than DNA microarrays.

ACKNOWLEDGEMENTS

The authors thank Siger Holsappel, Aldert Zomer, and Wiep Klaas Smits for their valuable input in the development of MOLGENIS. The authors would like to thank the referees for their helpful comments and suggestions for improving the presentation and comprehensibility of the paper. Part of this work has been funded by EU-BioGRID grant IST-2002-38344.

REFERENCES

Anderle,P., Duval,M., Draghici,S., Kuklin,A., Littlejohn,T.G., Medrano,J.F., Vilanova,D. and Roberts,M.A. (2003) Gene expression databases and data mining. *Biotechniques*, 36–44.

Brazma,A., Hingamp,P., Quackenbush,J., Sherlock,G., Spellman,P., Stoeckert,C., Aach,J., Ansorge,W., Ball,C.A., Causton,H.C. *et al*. (2001) Minimum information about a microarray experiment (MIAME)—toward standards for microarray data. *Nat. Genet.*, **29**, 365–371.

Brazma,A., Parkinson,H., Sarkans,U., Shojatalab,M., Vilo,J., Abeygunawardena,N., Holloway,E., Kapushesky,M., Kemmeren,P., Lara,G.G. *et al*. (2003) ArrayExpress—a public repository for microarray gene expression data at the EBI. *Nucleic Acids Res.*, **31**, 68–71.

Cheung,K.H., White,K., Hager,J., Gerstein,M., Reinke,V., Nelson,K., Masiar,P., Srivastava,R., Li,Y., Li,J. *et al*. (2002) YMD: a microarray database for large-scale gene expression analysis. *Proc. AMIA Symp.*, 140–144.

Edgar,R., Domrachev,M. and Lash,A.E. (2002) Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. *Nucleic Acids Res.*, **30**, 207–210.

Ermolaeva,O., Rastogi,M., Pruitt,K.D., Schuler,G.D., Bittner,M.L., Chen,Y., Simon,R., Meltzer,P., Trent,J.M. and Boguski,M.S. (1998) Data management and analysis for gene expression arrays. *Nat. Genet.*, **20**, 19–23.

Fellenberg,K., Hauser,N.C., Brors,B., Hoheisel,J.D. and Vingron,M. (2002) Microarray data warehouse allowing for inclusion of experiment annotations in statistical analysis. *Bioinformatics*, **18**, 424–433.

Gardiner-Garden,M. and Littlejohn,T.G. (2001) A comparison of microarray databases. *Brief. Bioinformatics*, **2**, 143–158.

Killion,P.J., Sherlock,G. and Iyer,V.R. (2003) The Longhorn Array Database (LAD): an open-source, MIAME compliant implementation of the Stanford Microarray Database (SMD). *BMC Bioinformatics*, **20**, 32.

Kokocinski, F., Wrobel,G., Hahn,M. and Lichter,P. (2003) QuickLims: facilitating the data management for DNA-microarray production. *Bioinformatics*, **19**, 283–284.

Kuipers,O.P., de Jong,A., Baerends,R.J., van Hijum,S.A., Zomer,A.L., Karsens,H.A., den Hengst,C.D., Kramer,N.E., Buist,G. and Kok,J. (2002) Transcriptome analysis and related databases of *Lactococcus lactis*. *Antonie Van Leeuwenhoek*, **82**, 113–122.

McIndoe,R.A., Lanzen,A. and Hurtz,K. (2003) MADGE: scalable distributed data management software for cDNA microarrays. *Bioinformatics*, **19**, 87–89.

Saal,L.H., Troein,C., Vallon-Christersson,J., Gruvberger,S., Borg,A. and Peterson,C. (2002) BioArray Software Environment (BASE):

a platform for comprehensive management and analysis of microarray data. *Genome Biol.*, **3**, SOFTWARE0003.

Saeed,A.I., Sharov,V., White,J., Li,J., Liang,W., Bhagabati,N., Braisted,J., Klapa,M., Currier,T., Thiagarajan,M. *et al*. (2003) TM4: a free, open-source system for microarray data management and analysis. *Biotechniques*, **34**, 374–378.

Sherlock,G., Hernandez-Boussard,T., Kasarskis,A., Binkley,G., Matese,J.C., Dwight,S.S., Kaloper,M., Weng,S., Jin,H., Ball,C.A. *et al*. (2001) The Stanford Microarray Database. *Nucleic Acids Res.*, **29**, 152–155.

Spellman,P.T., Miller,M., Stewart,J., Troup,C., Sarkans,U., Chervitz,S., Bernhart,D., Sherlock,G., Ball,C., Lepage,M. *et al*. (2002) Design and implementation of microarray gene expression markup language (MAGE-ML). *Genome Biol.*, **3**, RESEARCH0046.

Stoeckert,C., Pizarro,A., Manduchi,E., Gibson,M., Brunk,B., Crabtree,J., Schug,J., Shen-Orr,S. and Overton,G.C. (2001) A relational schema for both array-based and SAGE gene expression experiments. *Bioinformatics*, **17**, 300–308.

Stoeckert,C.J.,Jr, Causton,H.C. and Ball,C.A. (2002) Microarray databases: standards and ontologies. *Nat. Genet.*, **32**, 469–473.

Taylor,C.F., Paton,N.W., Garwood,K.L., Kirby,P.D, Stead,D.A., Yin,Z., Deutsch,E.W., Selway,L., Walker,J., Riba-Garcia,I. *et al*. (2003) A systematic approach to modeling, capturing, and disseminating proteomics experimental data. *Nat. Biotechnol.*, **21**, 247–254.

van Hijum,S.A.F.T., de Jong,A., Buist,G., Kok,J. and Kuipers,O.P. (2003) UniFrag and GenomePrimer, programs to select primers for genome-wide production of unique amplicons. *Bioinformatics*, **19**, 1580–1582.

van Hijum,S.A.F.T. Garcia de la Nava,J., Trelles,O., Kok,J. and Kuipers,O.P. (2004) MicroPreP: a cDNA microarray data preprocessing framework. *Appl. Bioinformatics*, **2**, 241–244.