A LIMS to support quantitative data acquisition and numerical simulations of dynamics of segment determination in *Drosophila* early embryo

M.Samsonova[1], E.Poustelnikova[1], A.Pisarev[1], E. Myasnikova[1] , K.Kozlov[1], J. Reinitz[2]

[1]State Polytechnical University, St. Petersburg, Russia.
[2]Dept. Applied Mathematics and Statistics & Center for Developmental Genetics, Stony Brook University, New York, Stony Brook, NY, USA

Abstract

We have devised complex scenarios of image and data processing and analysis to understand the dynamical regulatory mechanisms controlling the expression of segmentation genes in fruit fly. Due to ongoing data acquisition, development of new processing and analysis methods, as well as modification and improvement of old ones we encounter serious problems with data and workflows management. Additional problems pose different geographical location of research groups. To solve these problems we have developed a Laboratory Information Management System (LIMS) known as PIPE using the technology of multiagent systems. PIPE meets most of requirements for a LIMS in systems biology studies. In particular, it is easily extendable to deal with new data processing and analysis methods, flexible in specification and modification of these methods, scalable and supports distributed processing and analysis of data.

1. Introduction

Over a period of several years we have developed different methods for quantification of segmentation gene expression in *Drosophila* embryo, as well as for processing of this information to understand the dynamical regulatory mechanisms which control the expression of segmentation genes [1, 2].

Due to ongoing data acquisition data volumes continuously increase, new data types appear and the actual version of processed data is permanently updated. This in turn entails the development of new processing and analysis methods, as well as modification and improvement of old ones according to current needs, which is very difficult to foresee in advance. As a result we encounter serious problems with data storage and managing of application programs. Additional problems pose different geographical location of performance sites (Stony Brook, Los Alamos and St.Petersburg), as often users need data or programs kept in another laboratory. These problems complicate data analysis and processing and decrease the efficiency of work as a whole.

To automate information management and analysis, as well as to integrate different types of data and application programs at all performance sites we start to develop a Laboratory Information Management System (LIMS) known as PIPE. Here we describe the system architecture and demonstrate a real-life scenario of data processing and analysis.

2. Materials and Methods

2.1 Information flow

Like all other insects, the body of the fruit fly *Drosophila* is made up of repeated units called segments. The segment determination happens during the first 3 hours of the development of fruit fly and is controlled by the network of about 16 genes [3, 4]. In our experiments the expression of segmentation genes is monitored by confocally scanning of fixed embryos stained with fluorescently tagged antibodies. Images of gene expression obtained from these embryos serve as a raw material for quantification of gene expression. The quantitative data are obtained in several steps, for each step the specialized methods for image and data processing are developed and implemented [1]. As a result the reference data on expression of segmentation genes at cellular resolution and at each

time point are constructed. Images and quantitative gene expression data from individual embryos, as well as reference gene expression data are used to study the dynamics of formation of segmentation gene expression domains, precision of development and pattern formation and the mechanisms of segment determination [5].

2.2. System requirements

PIPE is aimed to provide flexible environment, which can be used to organize the on-line collaboration of investigators from different laboratories via the Internet. The requirements to such a software can be formulated as follows:

- extendability to deal with continuously growing number of images and data volumes, introduction of new processing and analysis methods, integration with third parties tools;
- flexibility in specification and modification of analysis methods;
- support of distributed processing and analysis of data;
- provision of simultaneous access of multiple users to shared data and methods;
- support of autonomous task performance upon connection hang up, as well as notification about processing results;
- use of heterogeneous software/hardware platforms;
- provision of access through FireWall and и Proxy servers;
- scalability
- no need in programming skills or familiarization how to install special software libraries and program tools for processing and analysis of data;
- availability of powerful and friendly Web-based user interface, as well as visualization tools;
- provision of continuous work, when new components are added or old one are removed;
- sufficient response time and readiness characteristics;
- failure-resistance, if malfunction of hardware or software components happens;
- preferably based on open source software;
- portability across software platforms.

Looking at currently used client/server architectures it become immediately clear that 2-tier and 3-tier architectures cannot support the functionality required. Use of three-tier architecture allows to solve a number of problems. It makes it possible to reduce requirements to client machines and increase data protection and security as some processing functions and intra-system information (including access passwords) are moved from client interface to application servers. The systems based on 3-tier architecture have higher adaptability than ones designed in frame of two-tier paradigm. However the adaptability of these systems is restricted by predefined functions and methods. It is not possible to extend the functionality of 3-tier system without its stop. In addition 3-tier architecture does not provide for dynamic reconfiguration of the system and rational distribution of queries.

Finally therefore we decide to examine if multi-tier or multiagent architecture can meet functional requirements formulated above. The important advantage of multiagent systems is their inherent modularity. Due to modularity these systems are scalable and easy extendable. Moreover multiagent systems with redundant components (databases, agents, application programs, other services) are robust, highly adaptable to functional extensions and have high readiness and reactivity characteristics. Thus the multiagent architecture satisfies nearly all our requirements to the behavior of LIMS.

Standards and services for the implementation of multiagent architecture are currently being developed using FIPA, SOAP, GRID and REST technologies. Recently several successful tools have been developed which allow a user to search for available services over the Internet, link these services into workflows and to specify input and output parameters [6,7]. However all these architectures and standards are continuously developed and modified. Moreover they cannot currently provide for failure-resistance, sufficient response time and readiness characteristics of a

system, as well as lead to information redundancy when standard image formats are converted to XML. Therefore to develop our system we adopt a REST-like approach which makes it possible to extend the system by using locking agents to XML-RPC and SOAP.

2.3 System architecture

Figure 1 presents the system architecture. The current configuration of the system includes two servers each containing all system components. All agents are designed as multithreaded Java HTTP servers, which can actively communicate with other agents. The agents exchange messages via HTTP protocol using GET and POST commands, and hence can be used in networks with FireWall and Proxy Server. The agents implement complex scenarios of distributed interactions in heterogeneous environment.

2.3.1 System configuration

The information about agents and their functions is stored in a coordination agent (CA) database. To ensure the actuality of information about system configuration

- each agent database stores the list of counteragents and their URLs, the list of functions, reference to the monitoring program, load and authorization characteristics;
- each agent registers with CAs reporting URL, the logical names of executed services, as well as the parameters of designed load characteristics;
- each agent notifies CAs about its scheduled sign-off (e.g., due to the decrease of load on a given service or modification);
- all agents update the information about system configuration by notifying CAs about their current load;
- if any agent or service is unavailable its counteragents notify CAs about their failure to establish connection;
- functionality of the system as a whole and each registered service separately is periodically monitored;
- CAs notify registered agents about changes in the configuration of the system by sending HTTP messages;
- system administrator is notified by e-mail if malfunction of the system happens.

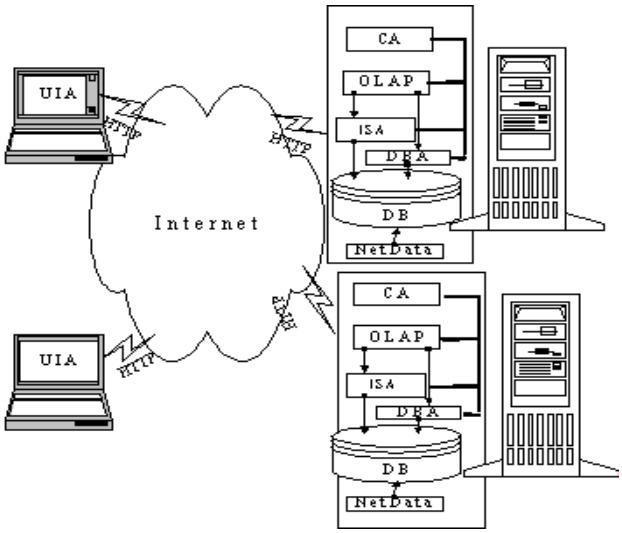

Fig. 1. System architecture.

Each agent selects a counteragent (or a required service) with regard to its availability and load, the counteragent located on the same server being selected first. This means that the interactions of agents are not static and reorganize dynamically. It is the continuous tracking of actual system configuration and the dynamic reorganization of agent interactions that ensure the capability of the

system to reconfigure. This property allows to extend the functionality of the system and to modify it in operation mode increasing the efficiency of the system use.

2.3.2 System components

In what follows we will present a detailed description of each system component.

User interface agent (UIA). This agent supports user registration, local searches on a client side and the uploading of data and images to a server. It also allows to visualize uploaded data and image files as well as to confirm user's intention to insert this data into a database. Besides UIA provides for visualization of data and images inserted into a database; selection of application program modules and specification of parameters for data analysis and processing; visualization of the results of processing and analysis; retrieval of images and data from a database as well as selection of output formats and downloading of data and images from a database.

Database access agent (DBA). DBA executes SQL queries to a database via JDBC; formats query result as TXT, HTML or XML files and converts images stored in the database into JPEG format for visualization on a client side.

Image server agent (ISA). This agent performs conversion of image formats and image scaling using ImageMagic library. It also executes standard operations on images (e.g. contrast enhancement, intensity filtering, combining of several images, etc.). In addition ISA participates in the retrieval of images from a database for processing as well as in the visualization of processed images as JEPGs.

OLAP server. The OLAP server cooperates with DBA and ISA servers to execute complex scenarios for image and data processing and analysis. The logical rules are applied to implement this cooperation. The OLAP server communicates also with the local database via JDBC and the remote database via DBA. In addition this server interacts with registered workflow modules providing for their initialization, function calls and result output.

Coordination agent (CA). CA supports agent registration, and notifies agents about current system status in respond to their request. It also monitors the functionality of the system and notifies registered agents about agent failure and other changes in the configuration of the system. CA notifies the administrator of the system about these changes by e-mail.

Database. To manage data, images, as well as the results of processing and analysis we decide to use IBM DB2 RDBMS, which supports necessary functionality and reliability of the system and requires minimum familiarization efforts.

Workflows and program modules. In general each scenario for image processing or data analysis consists of many steps. Although it is possible to write one program, which will perform all these steps, it is more convenient to write a separate program for each step and to run these programs as a workflow. This approach to the program design is known as modular programming.

A challenge in implementing modular programming environment is that each workflow should be designed with the interface that can talk to the program before and after it in the chain of calculations. This interface needs to be flexible enough to support communication with different modules in different workflows. Our system provides a very powerful way to implement such interface. Program modules communicate with each other via agents. An agent can insert data into a database, send it directly to the next program module and modify configuration files and other auxiliary data, if necessary. We implement program modules in C++ or Java.

3. Implementation

3.1 User interface

User interface supports visual construction of workflows from program modules, workflow execution and the visualization of both intermediate and final results.

The workflow is constructed by joining program modules and represents a directed acyclic graph. In this graph the nodes shown as rectangles are modules, while the edges displayed as arrows are data-dependency links, which specify that the output of one module serves as input to another one.

After a program module was visually constructed, its parameters are specified. Program modules forming one workflow may be located on one server or may be distributed among several server machines.

3.2 Real-life scenario: Rotating and cropping of an embryo image

In practice, embryo images do not come of the microscope containing only the embryo of interest in the standard orientation used by embryologists (anterior pole to the left and dorsal side up). Thus, each embryo image presents a unique image processing problem which is solved by applying the scenario described below (Fig. 2).

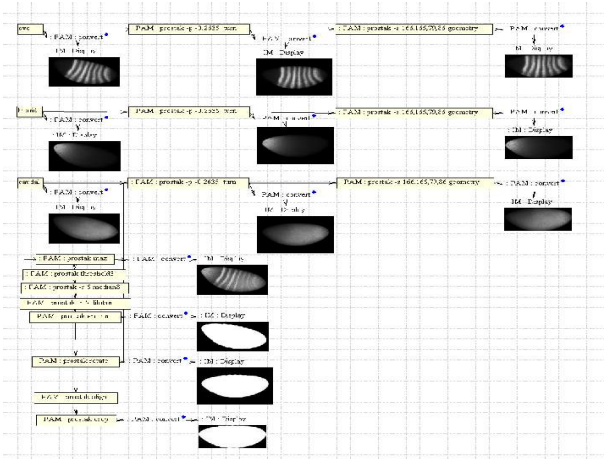

Fig. 2. PIPE in use: Rotating and cropping of an embryo image.

First the whole embryo mask is constructed by applying thresholding, median filter, erosion and dilation procedures. A central purpose of the whole embryo mask is to facilitate the rotation and cropping of an embryo image. To do this scale and translation invariant moments are calculated and used to find the rotation angle in respect to the canvas vertical axis. Next this angle is applied to rotate the embryo image. After rotation the image is cropped according to the geometry of the whole embryo mask.

4. Conclusions

The technological approach used in this work results in regulation, standardization and automation of data processing and analysis procedures, reduction of data processing errors and acceleration of system reactivity to user requests, as well as in effective use of expensive equipment and personnel skills. This increases efficiency and productivity of work, as well as expand potentialities of all performance sites to analyze and process data.

In future we will continue to link new application programs to the system. We also plan to integrate our system with several third parties tools (e.g. Khoros image processing package) and develop web-based interface.

5.References

1. E.Myasnikova, A.Samsonova, K.Kozlov, M.Samsonova, and John Reinitz (2001) Registration of the expression patterns of Drosophila segmentation genes by two independent methods. Bioinformatics, 17(1): 3 -12.

2. Jaeger, J., Surkova, S., Blagov, M., Janssens, H., Kosman, D., Kozlov, K.N., Manu, Myasnikova, E., Vanario-Alonso, C.E., Samsonova, M., Sharp, D.H.and Reinitz, J. (2004). Dynamic control of positional information in the early *Drosophila* embryo. Nature, 430: 368-371.

3. V.A. Foe and B.M. Alberts (1983) Studies of nuclear and cytoplasmic behaviour during the five mitotic cycles that precede gastrulation in Drosophila embryogenesis. Journal of Cell Science, 61: 31-70.

4. P.W. Ingham (1988) The molecular genetics of embryonic pattern formation in Drosophila. Nature, 335: 25-34.
5. Jaeger,J., Blagov, M., Kosman, D., Kozlov, K.N., Manu, Myasnikova, E., Surkova, S., Vanario-Alonso, C.E., Samsonova, M., Sharp,D.H. and Reinitz, J. (2004). Dynamical analysis of regulatory interactions in the gap gene system of *Drosophila melanogaster*. Genetics, 167: 1721-1737.
6. http://taverna.sourceforge.net/
7. http://hobit.sourceforge.net/