COMPUTATIONAL ENVIRONMENT FOR CELLULAR IMAGING BY FLUORESCENCE MICROSCOPY

GERNOT STOCKER

DOCTORAL THESIS

Graz University of Technology
Institute for Genomics and Bioinformatics
Petersgasse 14, 8010 Graz, Austria

Graz, February 2007

Abstract

In the postgenomic era, high-throughput screening methods have identified a large number of genes which still lack functional characterization. Fluorescence microscopy can be used to localize labeled proteins from those genes in determining their functional role in cellular processes. In order to perform a systematic analysis of the generated microscopy images, structured storage of the data and organized documentation of its experiments are inevitable.
To address this problem a platform independent computational environment for handling microscopy data, named Scientific MIcroscopy Laboratory Environment (SMILE), was developed. The project oriented workflow of a typical experiment guides the design of the software for data storage, data retrieval and consistent chronological documentation. This approach inherently leads to structured recording of the experiment, conserving the complete experimental context of data records at any time. State-of-the-art software technology was used to implement a multi-tiered J2EE application offering an intuitive Web-based user interface. The SMILE application back-end can be accessed through a generic programming interface and is open for integration of external imaging tools, such as ImageJ. Computationally intensive tasks for server-side image processing and analysis are transparently delegated to an adequate high performance computing infrastructure, and results are stored within the underlying relational database management system.
The SMILE computational environment offers researchers an integrative software platform for scientific collaboration, while hiding the necessary complexity of data management and IT infrastructure. Because of its modular and flexible architecture, SMILE is ready to face the future challenges of microscopy data management.

Keywords: microscopy, experiment workflow, SMILE database, J2EE, high performance computing

Publications

This thesis was based on the following publications, as well as upon unpublished work:

Papers

J. Rainer, F. Sanchez-Cabo, G. Stocker, A. Sturn, Z. Trajanoski. CARMAweb: comprehensive R- and BioConductor-based Web service for microarray data analysis. Nucleic Acids Res. 34: W498-W503 (2006) PMID: 16845058

C. Vogl, F. Sanchez-Cabo, G. Stocker, S. Hubbard, O. Wolkenhauer, Z. Trajanoski. A fully Bayesian model to cluster gene-expression profiles. Bioinformatics. 21 Suppl 2: ii130-ii136 (2005) PMID: 16204092

M. Maurer, R. Molidor, A. Sturn, J. Hartler, H. Hackl, G. Stocker, A. Prokesch, M. Scheideler, Z. Trajanoski. MARS: microarray analysis, retrieval, and storage system. BMC Bioinformatics. 6: 101-101 (2005) PMID: 15836795

H. Hackl, M. Maurer, B. Mlecnik, J. Hartler, G. Stocker, D. Miranda-Saavedra, Z. Trajanoski. GOLDdb: Genomics of lipid-associated disorders database. BMC Genomics. 5: 93-93 (2004) PMID: 15588328

G. Stocker, D. Rieder, Z. Trajanoski. ClusterControl: a Web interface for distributing and monitoring bioinformatics applications on a Linux cluster. Bioinformatics. 20: 805-807 (2004) PMID: 14751976

G. G. Thallinger, S. Trajanoski, G. Stocker, Z. Trajanoski. Information management systems for pharmacogenomics. Pharmacogenomics. 3: 651-667 (2002) PMID: 12223050

Conference proceedings and poster presentations

J. Hartler, G. G. Thallinger, G. Stocker, A. Sturn, T. Burkard, E. Körner, K. Mechtler, Z. Trajanoski, Z. Management and Analysis of Proteomics LC-MS/MS Data. Fourth International Symposium of the Austrian Proteomics Platform, Seefeld in Tirol. 2007 Jan 28.

J. Hartler, G. G. Thallinger, G. Stocker, A. Sturn, T. Burkard, E. Körner, T. Fuchs, K. Mechtler, Z. Trajanoski, Z. MASPECTRAS: Web-based System for Storage, Retrieval, Quantification and Analysis of Proteomic LC MS/MS Data. Third International Symposium of the Austrian Pro-

teomics Platform, Seefeld in Tirol. 2006 Jan 16.

J. Hartler, G. G. Thallinger, G. Stocker, A. Sturn, T. Burkard, E Körner, T. Fuchs, K. Mechtler, Z. Trajanoski. MASPECTRAS: Web-based System for Storage, Retrieval, and Analysis of Proteomic LC MS/MS Data. HUPO 4th Annual World Congress, Munich. 2005 Aug 29.

F. Sanchez-Cabo, H. Hackl, S. Hubbard, G. Stocker, Z. Trajanoski, O. Wolkenhauer, C. Vogl. A Fully Bayesian Model to Cluster Gene Expression Profiles. ISMB/ECCB 2004, Glasgow. 2004 Jul 31- Aug 04.

G. Stocker, J. McNally, Z. Trajanoski. SMILE: Scientific microscopy lab environment. OEGBMT Symposium für Biomedizinische Technik, Graz, Austria. 2004 Nov 12-13.

H. Hackl, T. Burkard, C. Paar, R. Fiedler, A. Sturn, G. Stocker, R. M. Rubio, J. Quackenbush, A. Schleiffer, F. Eisenhaber, Z. Trajanoski. Large scale gene expression analysis and functional annotation of adipocyte differentiation. Keystone Symposia: Molecular Control of Adipogenesis and Obesity, Banff, Canada. 2004 Mar 04-10.

H. Hackl, T. Burkard, C. Paar, R. Fiedler, A. Sturn, G. Stocker, R. Rubio, J. Quackenbush, A. Schleiffer, F. Eisenhaber, Z. Trajanoski. Large Scale Expression Profiling and Functional Annotation of Adipocyte Differentiation. First International Symposium of the Austrian Proteomics Platform. Seefeld, Austria. 2004 Jan 26.

Contents

Chapter 1

Introduction

The sequencing of the human genome [International Human Genome Sequencing Consortium, 2004; Venter et al., 2001] identified thousands of as yet uncharacterized genes. With the localization of their transcribed proteins in subcellular components, it is possible to draw conclusions about the functional roles of these genes [O'Rourke et al., 2005]. Due to the steady development of fluorescence microscopy, reliable tools for assessing protein locations have become available [Giepmans et al., 2006].

Experiments using such methods to address significant biological questions result in image data which must be processed, analyzed, quantified and stored over long time periods on an adequate infrastructure. The necessity for a data management system is clear, and several initiatives were started to standardize the way of describing and storing microscopy data output [Goldberg et al., 2005; Swedlow et al., 2003, 2006]. Several database systems for specific applications [Carazo et al., 1999; Gonzalez-Couto et al., 2001; Kals et al., 2005; Liang et al., 2002] have been developed, but most of these focus on a targeted experimental setup and are not universally applicable.

The Open Microscopy Environment (OME) project [Swedlow et al., 2003] is the most actively developed scientific database project for universal microscopy data management, devoting noticeable effort into standardizing methods of image storage [Goldberg et al., 2005]. OME has introduced a new tiff-based file format, which contains XML formatted data about the creation and development of an image. Several microscopy software packages from industry and academia are already able to read the format and to access OME's server back-end. Technologically, OME is offering a Perl interface, used for back-end image storage and the implementation of a data-centric lightweight Web interface. Recently, OME was extended by the J2EE-based server application OMERO, which also handles the management of image data and is accessed by a standalone Java client, named Schoola. OME is particularly focused on handling image data but lacks an easy-to-use Web interface for data acquisition during the development of experi-

mental protocols.
To the best of our knowledge, there is no academic microscopy data management system available, that is able to generically store experimental data in a workflow-oriented manner independent of the file format. This includes primarily image files, but also experiment-related documents from literature research over analysis documentation to comparison and experimental notes. Therefore, the idea of an intuitive software package, named Scientific Microscopy Lab Environment (SMILE), was born. SMILE should follow the typical workflow of an experiment and should map every working step to a discrete step in the software. If SMILE is used on a regular basis, the organization of data in a chronological and experiment-oriented way is continuously built up as data is acquired. Such acquired data always remains in the original experimental context. The workflow is the central starting point of this thesis, and will be analyzed and described in more detail in the next section.
Furthermore, recent initiatives defining minimum standards for describing microscopy experiments (e.g. MISFISHIE [Deutsch et al., 2006]) need to be supported in a manner where existing researchers are aware of the specific parameters and settings, which are required for standardization of data exchange. In this way, data sharing between groups and the reproducibility of data should be guaranteed.
Additional requirements should be fulfilled by an experiment management system
[Jakobovits et al., 2000]: it should be possible to integrate the system in a heterogeneous environment. Remote collaboration over the Internet between working groups, intuitive navigation within the system, advanced file format support and their data integration all need to be considered. SMILE tries to address these requirements with a Web-based interface, which allows platform independent access to the system. For more sophisticated tasks like multiple file transfer, which can't be handled with regular Web-form based interfaces, SMILE uses Java and its Applet features. The system has an integrative characteristic and offers simple interfaces in a necessarily complex IT infrastructure (e.g. numerically intensive analysis performed on a high performance computing (HPC) cluster). The real-world application of such a system will show to what degree this workflow oriented and integrative approach is accepted by the scientific community. The first biological projects exploring the nuclear organization of cells, and a large scale screening project in yeast, are already in the planning phase which will demonstrate the potential value of the concept.

1.1 Workflow analysis

In order to define software requirements, the lab work and its different stages have been analyzed to establish a complete workflow oriented model. Observations suggest that the lab work can be divided into five principal steps:

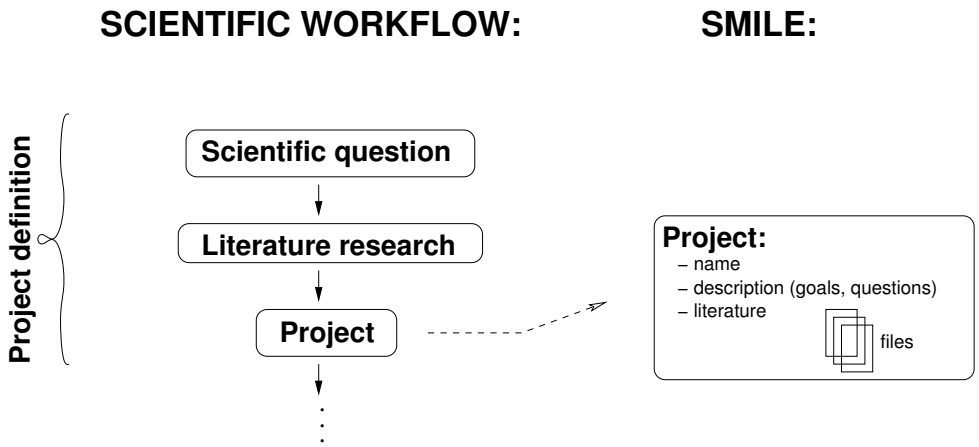


Figure 1.1: Workflow phase: project and experiment definition

- project definition
- data acquisition
- data analysis and processing
- data comparison
- data retrieval

These five steps explain how SMILE can be used to track the data evolution in the experimental stage.

Project definition In general, scientists start by asking biological questions, to which the answers are of special interest but largely unknown. A detailed literature research should determine whether others have already satisfactorily solved the specific questions. If not, the next stage is set up a hypotheses that can be tested and thus proven or disproven.
Starting from such considerations, a new project should be defined and captured in digital form. It is necessary to describe the new project by: giving it a descriptive name; defining which questions should be answered and establishing what the solution strategy should look like. Documents collected during the literature research stage should be collated with the project definition for later review or for sharing with other researchers.

Data acquisition With the establishment of a hypothesis, the detailed design of experiments can begin. Experimental work follows well established (standard) protocols which may have to be optimized for specific conditions. It must be possible to define well established protocols as a sequence of steps, each of which is defined just once but reused each time the same protocol is followed. However, well established protocols must be kept flexible in a way

that particular conditions can be changed. Those changing parameters of standard protocol steps (e.g. fixation times, temperature changes etc.) are important to record as they are used to improve the biological reliability of a protocol.

Armed with a collection of standard protocols, an experiment can be started and defined in SMILE at the start of the workflow. The name of the experiment, its description and specific aims, must all be provided in order to be able to distinguish different experiments. The experiment itself is described by its current working protocol (CWP). If the experiment is following a standard protocol exactly, the CWP needs not to be defined again but can be created by copying the steps- and parameter definitions of a corresponding standard protocol. A sequence of input masks (wizard) should assist the input of the CWP and its step parameters.

Sometimes, however, after completing a sequence of protocol steps, experiments split up into different sub-branches (e.g. cells are treated with different drugs in order to study their response). Therefore, software like SMILE must be able to handle the insertion of such "splitting steps", leading to different branches of the experiment workflow.

Cell treatments or time-series experiments result in one or more different digital files that are acquired with a special device and at a special step during the protocol. As a lab has only a limited number of devices used during experiments, the definition of the devices needs to be performed just once and can be reused by every user within SMILE. The facility manager should insert the devices with their name, their descriptions, and their documentation files. Facility managers are also the persons with the most experience of using the devices. They should be able to define the mandatory and optional device parameters that must be inserted during an experiment. In this way, all necessary information, such as the objective magnification, etc., can not be lost.

For data acquisition, SMILE offers the possibility to store all generated files acquired by different devices. The file transfer of one or more results files should be as simple as possible. By selecting a specific file or directory, it should be possible to connect these files to specific steps or entire protocols. All the internal linkage to the steps should be handled internally without confronting the user with these details.

Finally, data acquisition comprises not only files but also observations made during the experiments. Often, these disappear into notebooks and are lost to digital systems. Therefore, for every protocol step and every file obtained during the experiments, the ability to file observations in textual form should exist. To improve the search functionality, an observation should consist of some keywords, which are automatically added to a dictionary, along with a detailed description of the observation. As the files are attached to a protocol and an experiment, the overall context is conserved and the reproducibility of the same conditions and the same results is maintained.

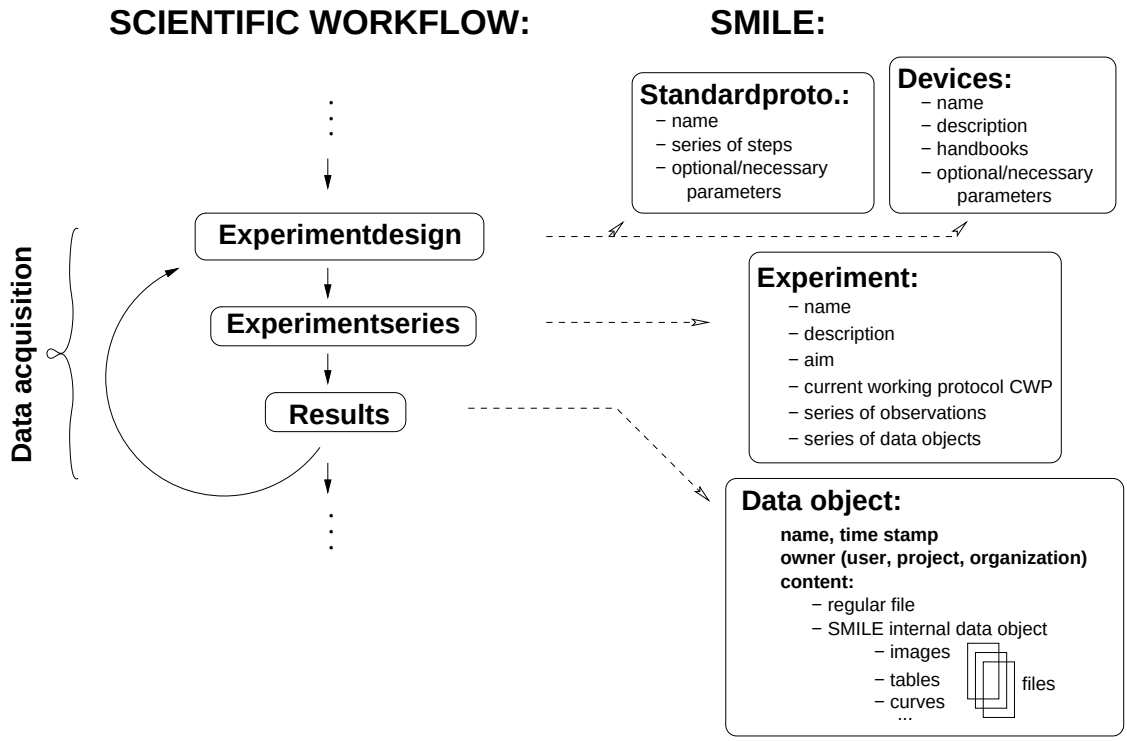


Figure 1.2: Workflow phase: data acquisition

Data analysis and processing After storing the raw result files, their analysis and post-processing steps are performed. In order to extract useful information and to combine it in a statistically meaningful manner, multiple microscopy images of different cells have to be acquired. The data files, for example from a Fluorescence Recovery After Photobleaching (FRAP) experiment, are image stacks. For example, the temporal increase of intensity values in areas with Green Fluorescence Protein (GFP)-tagged proteins must be quantified after bleaching. This can be done with an external program that is optimized for this measurement. The output result file is a table with values of intensities from the bleach-spot, background, etc. With SMILE it is possible to describe such external analytical steps, so that the files resulting from the external analysis software can be assigned to the original raw-data files. The aim of the external analysis steps is to keep track of the development of data collected using third party software. Such analyses should have a suitable name, a short description and detail the parameters used during the operation. Another example of the use of external programs for post-processing are deconvolution packages for improving the image quality by numerically eliminating out-of-focus light from images. During these post-processing steps parameters must be adjusted to obtain improved results. These can be stored with the analysis steps as optional or required parameters. The data generated at the analysis stage can be easily connected to the raw data by using a platform independent upload client, which allows connection of the data files with the

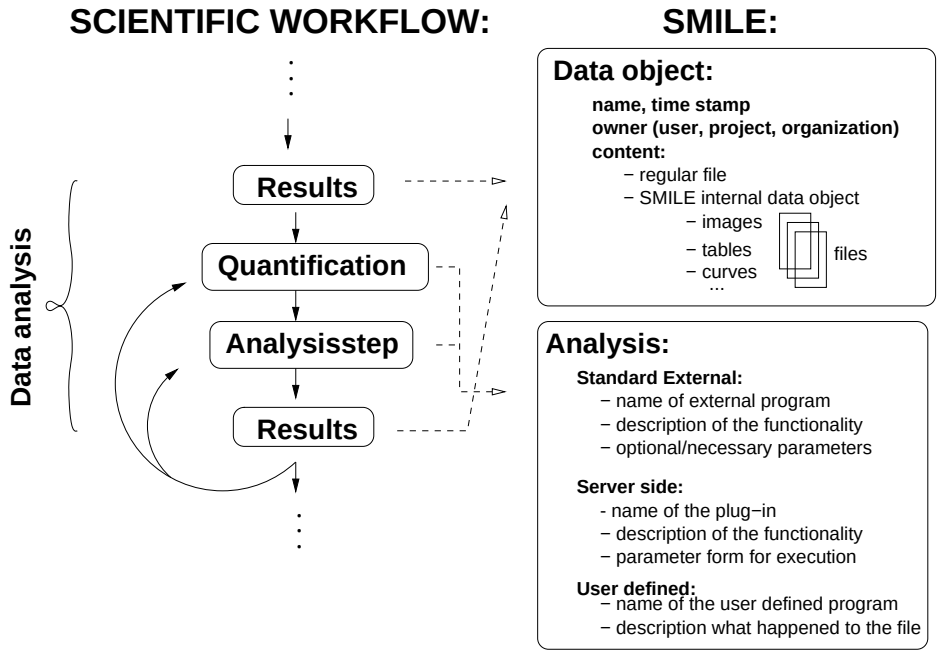


Figure 1.3: Workflow phase: data analysis

original experimental context.

To improve the productivity of frequently performed analytical tasks, SMILE offers the possibility to integrate analysis modules (e.g. simple numerical operations like background corrections, etc.). Such analytical steps should be accessible from the server-side via the Web interface. SMILE offers a plug-in system so that additional server-side analysis steps can be added easily. The plug-in should implement a well-defined interface, enabling communication with SMILE (e.g. transferring the input file, running the analysis, and returning the result as a SMILE data object).

Data comparison Following data analysis, different results must be aggregated and compared before any interpretation of results can begin. Unexplained differences or unexpected similarities form the basis for new directions in planning further experiments. Sometimes it may not be possible to recognize connections between experiments, but it is important to keep notes of the comparisons already performed. SMILE should also be able to handle such notes for comparisons.

To perform a comparison within SMILE, the comparison must be first described. A descriptive name and the objective should be entered at the beginning of the comparison. Then one should be able to select the appropriate files or data objects to compare, and these should appear in a visual way. Images, either appearing next to each other, or curves, displayed on the same plot, should give an informative overview enabling pertinent conclusions to be drawn. Such

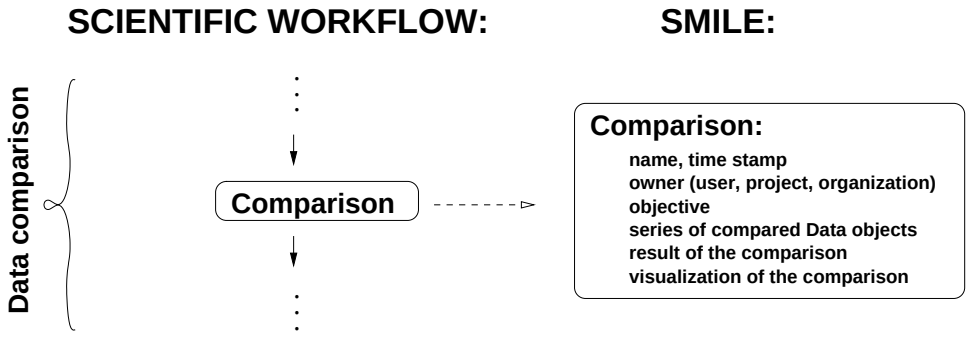


Figure 1.4: Workflow phase: data comparison

conclusions must be entered in SMILE to keep a record of the comparisons. Keywords should also be entered to enable comparisons via a searchable keyword index. These comparisons are usually parts of the same project and link together different experiments within a project.

Data retrieval If SMILE is used regularly, the organization of data in a chronological and project-oriented way is continuously registered during daily acquisition. An additional advantage is the ability to search through the data. Researchers frequently have to search through notebooks to find previously uninterpretable observations. Subsequently, as the project develops, the researchers gain a different perspective and recognize that prior observations could lead to new discoveries. SMILE offers easy to use interfaces that allow searches through observation notes, comparison notes or even changes performed during protocol observations. However, full text searches commonly result in numerous hits and the information disappears in the noise of the data. Searches with keywords, which are inserted during acquisition and analysis, should partly address this. The user should get an index of existing keywords, similar to a dictionary, with which one can search the stored data for experiments with similar observation or comparison notes. Experimental comparison and keyword-specific data retrieval is

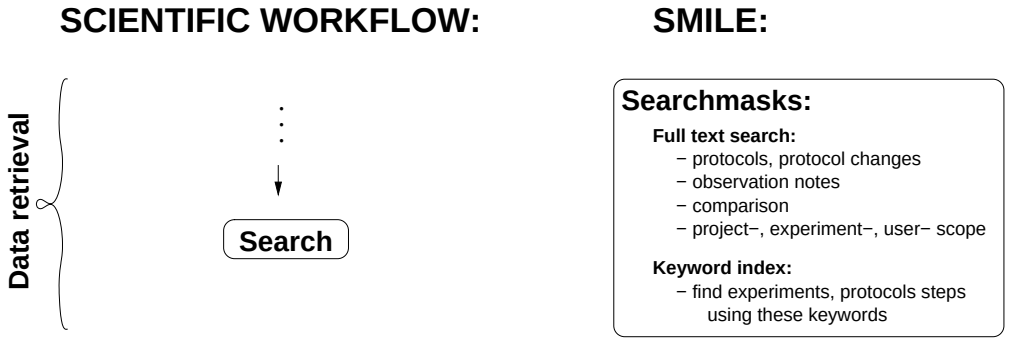


Figure 1.5: Workflow phase: data retrieval

intended to be a future extension, but is not part of this thesis. Nevertheless, the data model

behind SMILE must be able to accommodate these future extensions.

Other aspects Particularly during the first productive use of SMILE, it is absolutely necessary to gain the user's trust in the software. Consequently, it should be possible to export all acquired data from SMILE into text form, so it can be printed and inserted into a conventional notebook folder. The most convenient form of export is the PDF-format, where documents, protocols, etc., can be exchanged independent of the platform or operating-system.

User management Scientific work is commonly conducted in a team that needs to share protocols, experiments and results internally, but not necessarily with a wider community. Different people should be able to work together in a group, but should see only the projects that they belong to. To realize this protection from unauthorized access, a user management system must be introduced in SMILE.
To implement this, SMILE should offer the possibility to manage different independent groups which have completely separate file and operation spaces. These groups are termed organizations, which use SMILE on the same server but do not want to share anything with other organisations. Within an organization, different people work together on different projects. Therefore, every user belongs to at least one organization within which shared projects are worked on. The owner of the project and project administrator can assign additional people within an organization to a project.
Stored files should be also accessible directly via file system operations on a shared volume. To keep consistency within the SMILE database, only read-only file system access is allowed. Additionally, privacy between organizations must be preserved.

1.2 Objectives

The aim of this thesis is to develop a workflow-oriented computational environment for universally handling data in a microscopy laboratory. This comprises the structured storage of experimental imaging data and its development, including associated data generated by literature research, experimental design, quantification, analysis and observation. The project oriented workflow of a scientific experiment should be used as a guideline to achieve a consistent and context-specific work documentation. Microscopy experiments must be handled from the beginning of planning through the analysis, comparison and data retrieval stages.
Therefore, it is necessary to evaluate innovative technologies and to combine them with the proven J2EE platform. The architectural design should create a modular and extendable system. The required IT infrastructure, from enterprise-scale storage systems, to centralized data management within a relational database, to the high performance computing infrastructure, needs to be integrated into a single environment. Nevertheless, the resulting complexities should be hidden, as best as possible, from typical users of the system.
Finally, the resulting system needs to a be platform-independent environment, which organizes data for researchers collaborating within the biological microscopy community.

Chapter 2

Results

2.1 Overview

In order to achieve the previously described objectives, the following software components were implemented:

- ***Scientific MIcroscopy Laboratory Environment (SMILE)*** is a database centric J2EE application for handling experimental data based on the workflow of typical microscopy experiments. SMILE offers an intuitive Web interface for acquiring and accessing accumulating data, and additionally gives the possibility to access stored data from third party applications via a platform independent Simple Object Application Protocol (SOAP) Web service.

- ***JClusterService*** is a multipurpose component framework for the integration of high performance computing infrastructure into a Java centric IT environment. Challenging requirements regarding computational power and storage requirements, forced this transparent delegation of resource intensive tasks from server- and standalone applications to adequate external HPC systems.

- ***Back-end Infrastructure*** For a clean integration of the aforementioned services into an existing bioinformatics environment, it was necessary to implement back-end software components (e.g. integration in a central user management system with Single Sign On Support or tools for intuitive and platform independent data exchange).

Every module with its application-specific capabilities contributes to the resulting computational environment. This environment builds a collaboration platform for local laboratory users as well as for scientific collaborators over the Internet.

2.2 Scientific MIcroscopy Laboratory Environment (SMILE)

2.2.1 Functionality overview

In order to demonstrate the functionality of SMILE, a real microscopy experiment detailed in 2.1 was used. The aim of this experiment was to localize and estimate the number of actively transcribing RNA polymerase II foci within the nucleus of the hMads cell system [Rodriguez et al., 2004]. A basic protocol for immunofluorescence labeling was applied to tissue culture cells, which started by fixating the cells with para-formaldehyde and permeabilizing the cell membrane for primary antibody labeling. The secondary antibody was added afterwards and fluorescence images were taken from the prepared cells. In the course of the experiment parallel protocol steps were performed for realizing positive and negative controls. The experiment ended by performing post-processing steps using 3D deconvolution to enhance the visual quality of the image.

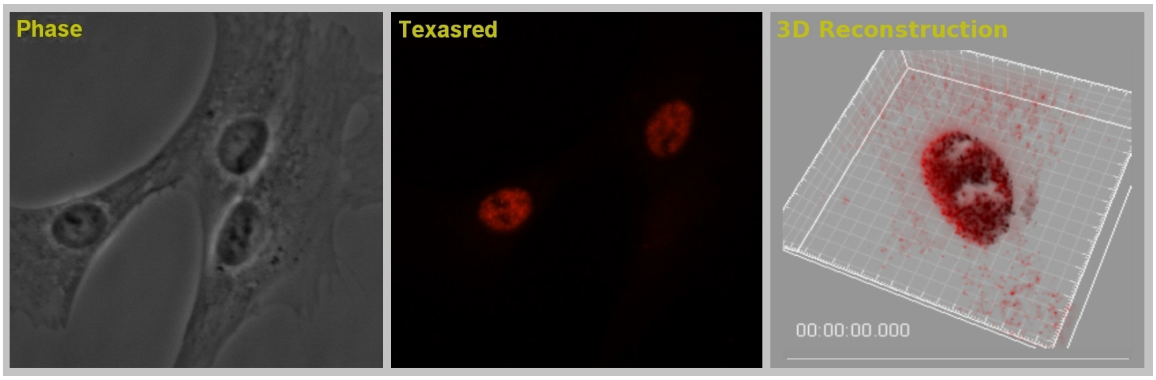


Figure 2.1: SMILE: experimental image data kindly provided by Dietmar Rieder

The functionality offered by the SMILE Web interface can be classified by the following main activities:

- **General organization** This part of the user interface, which covers the first step of the experimental workflow described in the introduction. The definition of projects and experiments consists solely in inserting the required descriptive parameters via a Web form. In doing so, a hierarchical structure with projects, sub-projects and experiments is created and displayed in the SMILE overview. This overview is the main starting point of SMILE, from which almost every activity can be initiated. By navigating through the tree, an information box appears alongside. This box details information about the current node in the tree and the operations which can be performed on the database managed object represented by the node. Already in this early stage, collateral files derived from literature research can be uploaded to projects and experiments, and ongoing observations can be

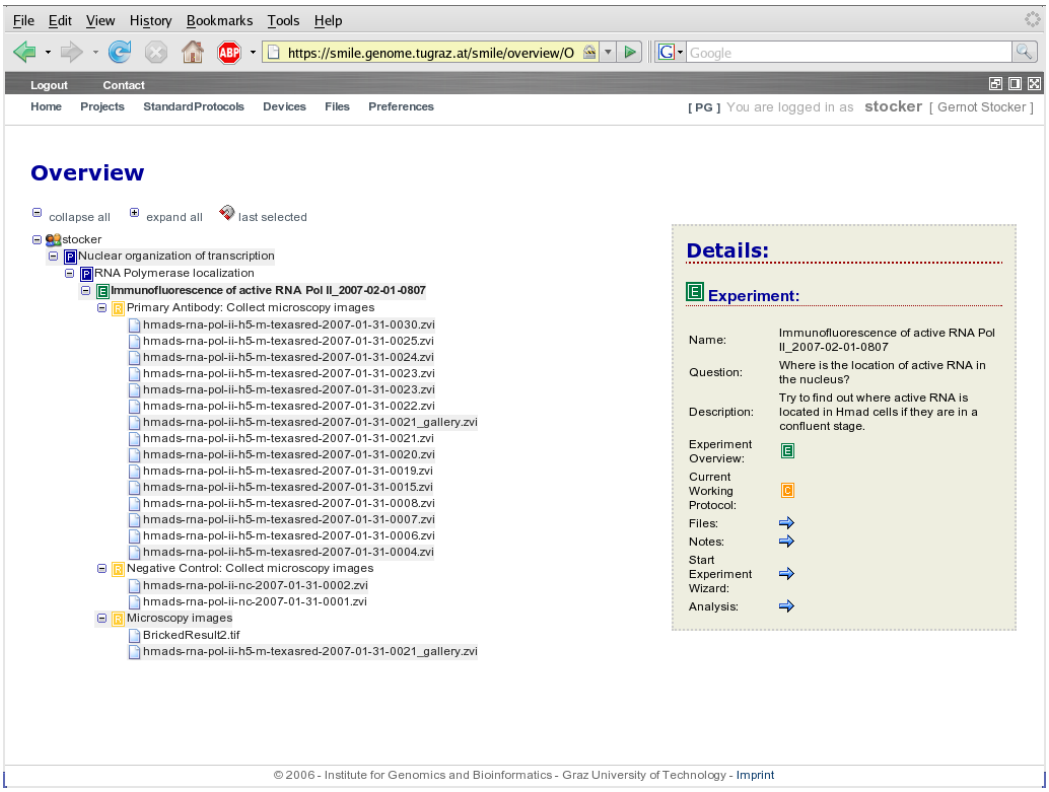


Figure 2.2: SMILE screenshot: main overview

stored using the general note dialog. If multiple files must be associated with projects and experiments, a Java applet can be used to conveniently upload the files to the generated project/experiment structure. SMILE can manage all files independent of their file type, and can thus be considered as a generic document management system. File types only need to be considered for subsequent processing and data extraction.

- **Standard protocol definition** When starting experimental work, the SMILE facility manager should define commonly used standard protocols using the protocol creation masks. Therefore, a sequence of steps must be defined which describes the typical ongoing experiment in detail. Dynamic parameters, which may be adapted for protocol optimization during the experiment, can be associated with the pre-defined steps. These parameters can be either numerical values, descriptive text or predefined enumeration types, all of which can be preset by default values and marked with appropriate units. In order to force the acquisition of critical parameters in the wizard, parameters can be marked as required. The initial definition is fairly laborious but necessary to improve the creation of working protocols within experiments. It is recommended rather to define small and reusable standard protocol units, which can be used as building blocks during the experiment-specific protocol assembly. Additionally, automatic internal versioning guarantees that changes to

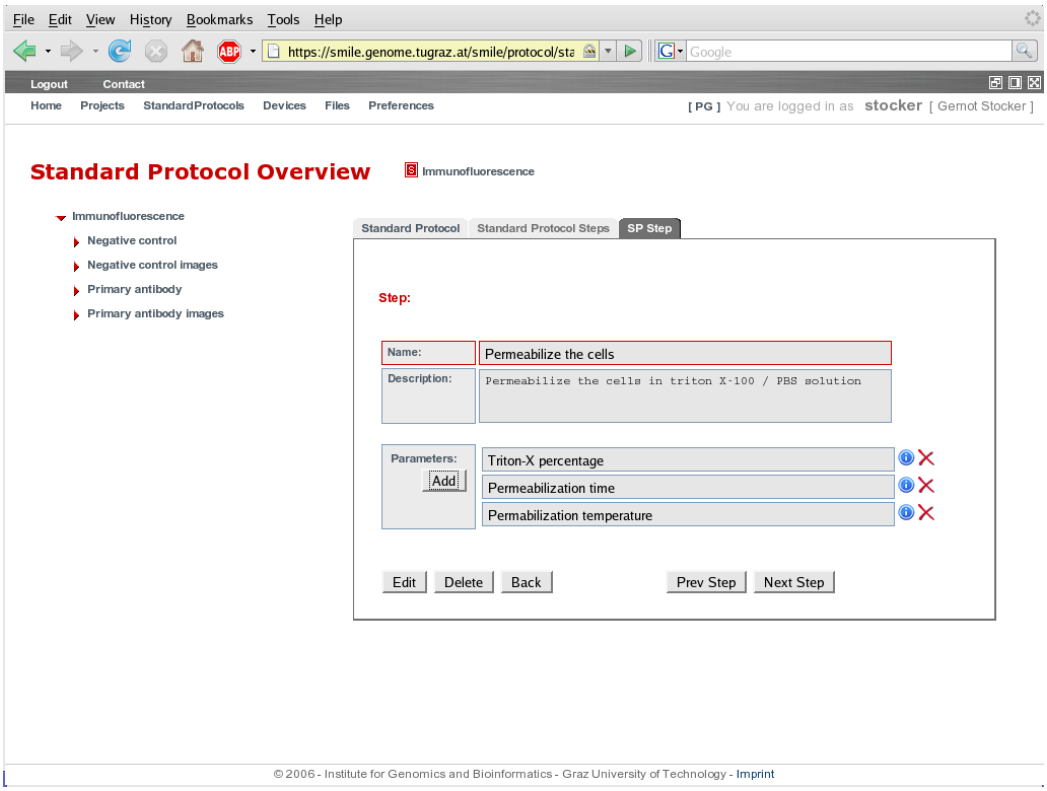


Figure 2.3: SMILE screenshot: standard protocol step definition mask

standard protocols do not affect dependent protocols used in previous experiments.

- **Current working protocol** As soon as the standard protocol templates are defined they can be used to define the current working protocol, describing the precise way the experiment is accomplished. Similar to the definition masks of the standard protocols, steps can be created as single steps, step groups or split steps. The latter step-type describes a situation in which the experiment splits into different sub-steps that must be performed in parallel. Parallel ongoing step sequences are again grouped together.
  Once the current working protocol is specified according the scheduled experiment, the protocol can be exported as a PDF file and printed. This document can be used as a guideline in the lab and has the advantage that the experimenter already knows, which step parameters are required for later experiment description within the wizard [Fischer, 2006].

- **Acquisition wizard** As images are collected throughout the course of the experiment, the necessary data for describing the project must be inserted using a comfortable wizard. This Web interface leads the user through the current working protocol. On every step the user has to fill out the value fields for required parameters and can attach files and notes

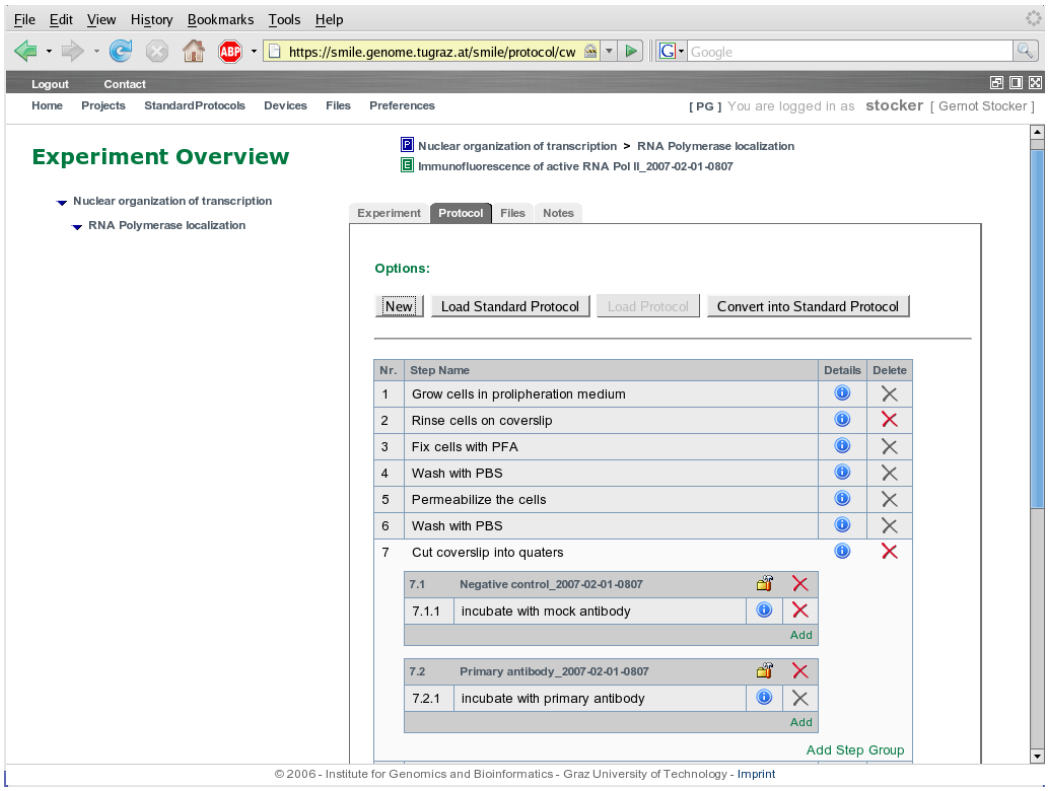


Figure 2.4: SMILE screenshot: current working protocol of an experiment

to each of the steps. During the CWP definition activity, it is important to name those steps to which files are attached in a descriptive way. The name of the file-containing steps appear as named node in the SMILE overview. Files that are directly connected to experimental steps are considered as raw-files and are protected against deletion, as further analytical steps can be easily repeated to produce the same results. The effort in reproducing experimental raw files is very high.

- **Analysis and post-processing** The analysis and post-processing steps are recorded in SMILE by either reusing existing analysis templates or describing new analysis steps applied to the previously uploaded raw-files. According to the file type, internally implemented analysis steps or the description of externally performed analysis steps are associated with the raw-files. Results files from analysis programs together with the used parameters can be easily attached to analysis definitions. As an example, a server side analysis tool was implemented for deconvolving three dimensional stacks, executed on a remote HPC cluster node using the JClusterService as application link.

- **Integration of external programs** In order to create a proof of concept implementation for externally accessing programs using the Web service API, a custom plugin for the

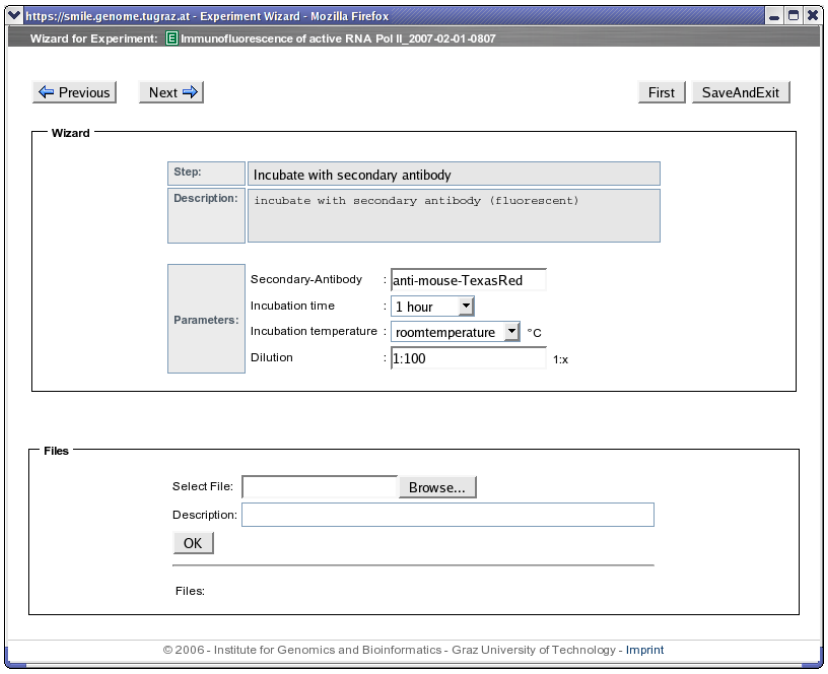


Figure 2.5: SMILE screenshot: data acquisition wizard with one step open

widely used image processing software, ImageJ, was implemented [ImageJ, 2007]. This Java plugin enables ImageJ to transfer the image files directly from SMILE to the client machine. This functionality appears as regular dialog in the graphical user interface of ImageJ, and allows upload of results files back into SMILE in a transparent manner.

Additional Web interfaces, such as the management of facility internal devices and the ability to specify user defined display preferences, are rudimentary implementations, but need further integration into the existing application modules. For improved usability, icons and colors are used consistently to signal in which part of the program the user is currently active. The data model in the back-end considers the persistence of comparison steps and advanced ontology-based keyword search, although the Web interfaces for accessing these functionalities are not yet implemented.

2.2.2 Architectural design and implementation

SMILE is a multi-tier client-server application and can be subdivided into different functional modules which interact as self-contained units according their defined responsibilities. The presentation tier within SMILE is formed by a Web interface, using Tapestry as the model view controller and an Axis Web service, which allows programming access to parts of the application logic. Thus, on the client side, a user requires an Internet connection and a recent Web browser with Java support, available for almost every platform. In order to provide a simple,

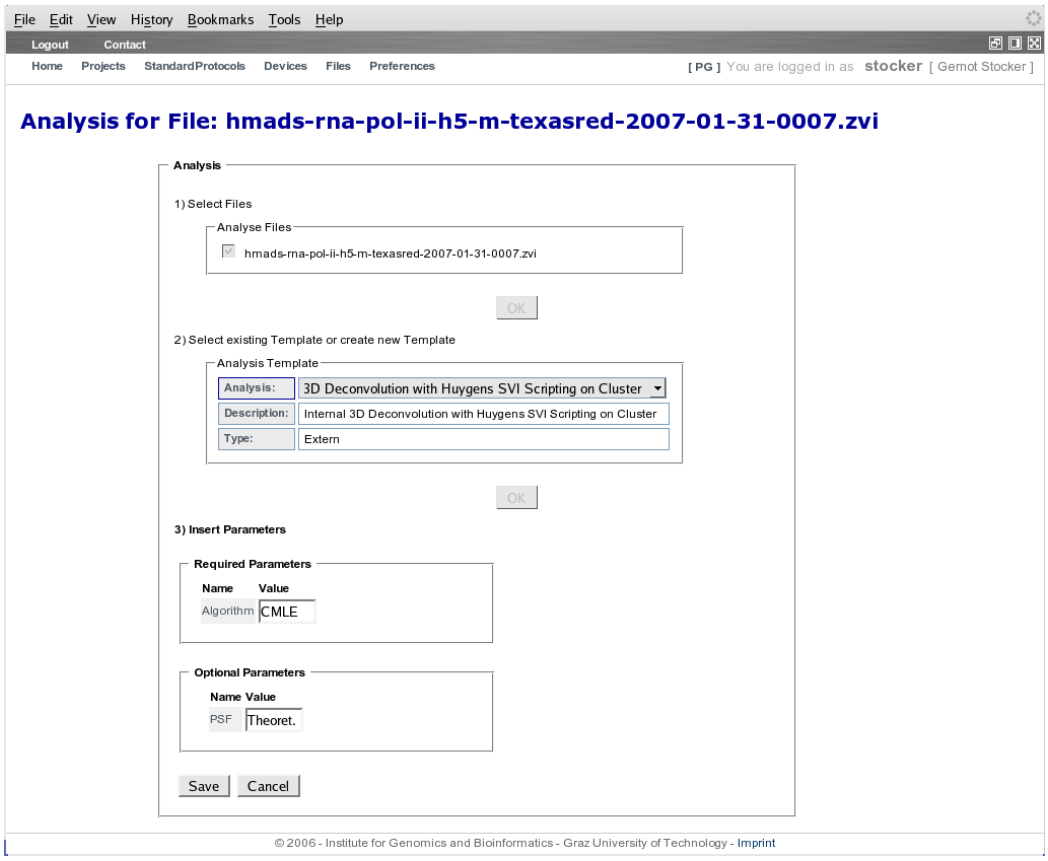


Figure 2.6: SMILE screenshot: mask for defining an analysis for one specific file

consistent but also attractive Web interface, SMILE follows internal institute guidelines for corporate design and uses Web 2.0 technologies for dynamic content generation.

The presentation tier exclusively accesses objects of a service facade, which hides the concrete realization of the application functionality. In between the presentation and business tier, a transparent ACEGI security layer prevents unauthorized calls on externally accessed methods. The service facade acts as a translator of view-related requests to the business tier and returns the results of the requests as serializable value objects. By doing this, the presentation tier does not directly communicate with instance objects of the persistence tier, and thus persistence technology can be replaced without changing code in the presentation tier.

The business tier is realized as view-independent application logic, which stores and retrieves datasets by communicating with the persistence layer. The internal management of files is also handled from a central service component, which persists the meta-information for acquired files to the database, and stores the file content in a file-system-based data hierarchy. The business layer also holds asynchronous services for application-internal JMS messaging and for integration of external computing resources, such as an HPC cluster. All services of this layer are implemented as Spring beans, for which the Spring-internal interceptor classes pro-

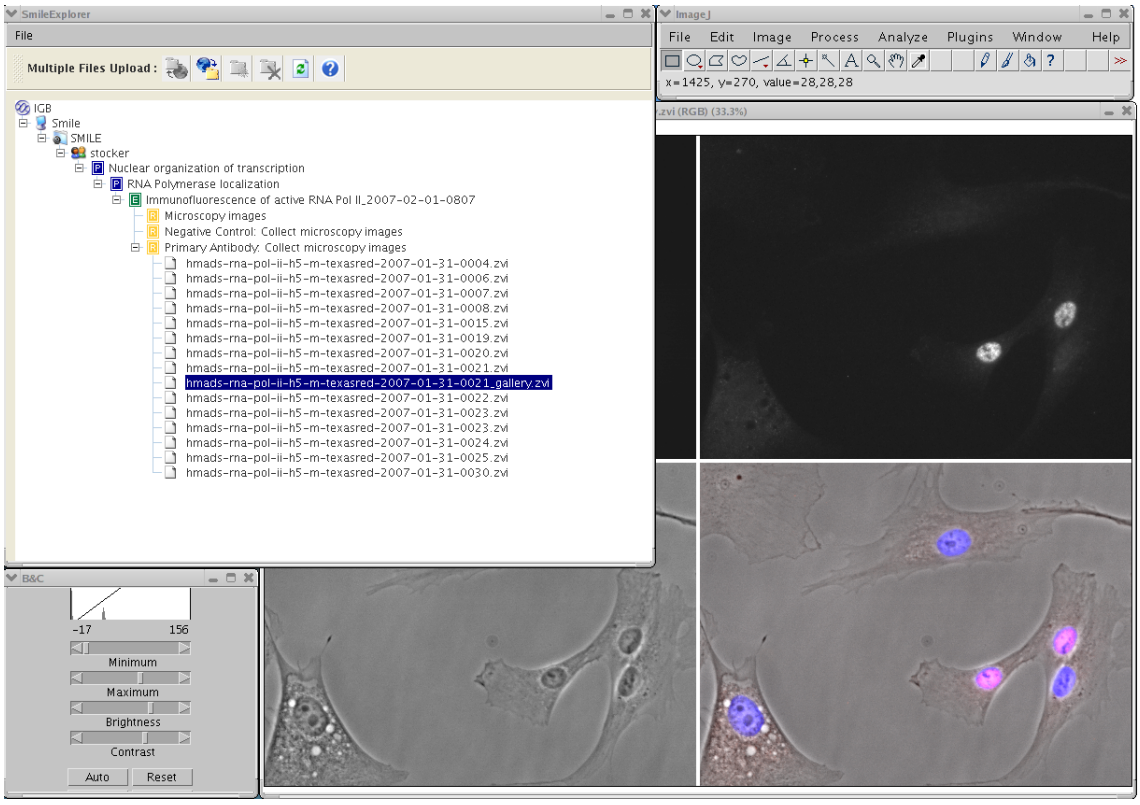


Figure 2.7: SMILE screenshot: ImageJ while accessing SMILE using the transfer plugin

vide transactional integrity.

Datasets that must be stored in the database are passed to the persistence tier, which offers data access object (DAO) classes for creating, retrieving, and updating the internal state of non-volatile data objects. The concrete implementation of the DAOs are derived from Spring DAOs, which access via Hibernate a PostgreSQL database. RDBMS specific SQL code was strictly avoided during the development process. Thus, the support for additional RDBMS should be possible without causing problem.

The business tier and the persistence tier are bound by the Spring J2EE lightweight container, which manages the component-object life cycle. Furthermore, the Spring context is transparently integrated into the Servlet context of Tapestry using the HiveMind container backend. Hence, Spring service beans are managed by Spring and can be directly accessed from the Tapestry pages by calling adequate annotated getter methods. Integrative glue code for lookups into the Spring container was avoided. Since SMILE uses Spring instead of EJB related components, the deployment of the application only requires a standard conformed Servlet container. Therefore, the Servlet container Tomcat is used, which offers not only Servlet functionality but J2EE infrastructure services such as centrally configured datasources and transaction management. This makes the deployment of SMILE on different servers easier,

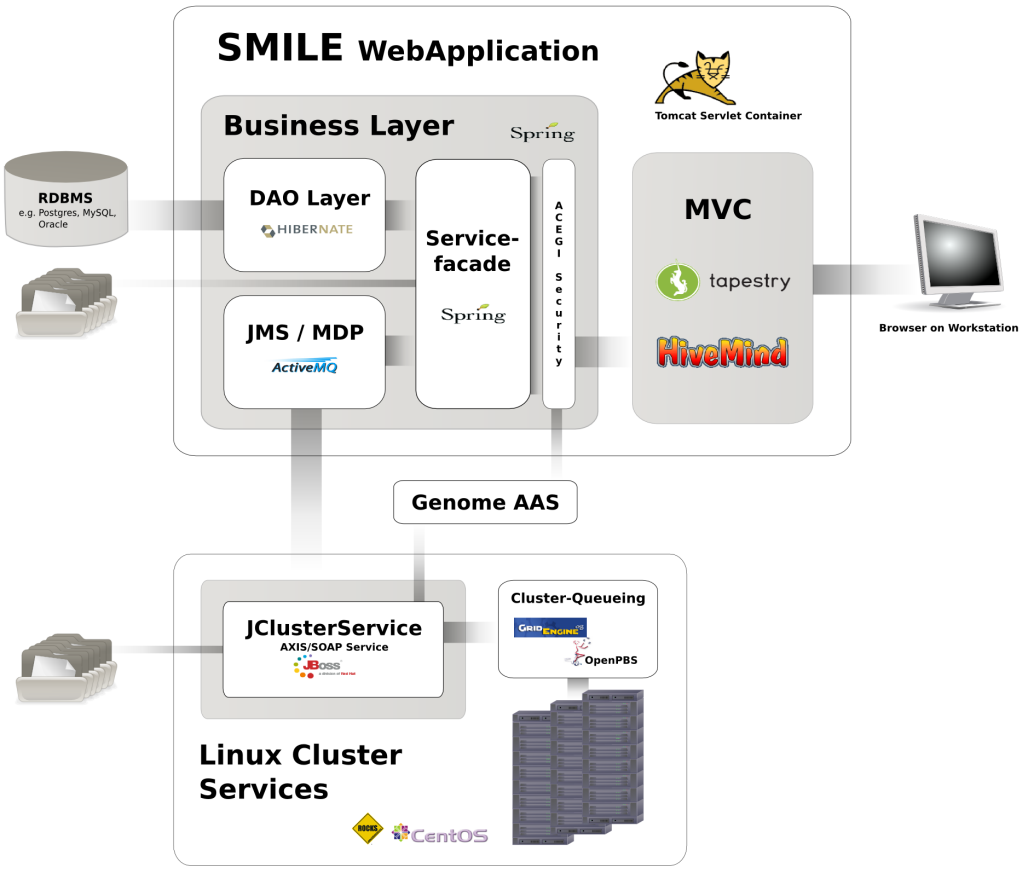


Figure 2.8: SMILE overview: software architecture

because machine-specific settings for different production environments are kept outside the application configuration. Internally, Spring takes care of acquiring the proper connection and configuration parameters from the J2EE environment. During startup of the application, the Spring container and the Tapestry Servlet are initialized automatically by the Servlet container and provide immediate Web access.

The SOAP Web service interface for external programmatic access is realized by combining the Web service framework Axis with corresponding SMILE components. These inherit the basic Web service functionality from the corresponding Spring Web service endpoint implementation, and have solely to consider the business functionality. The Web service operates as an external access point for Java Applets within the Web application, as well as for external analysis and processing applications such as ImageJ.

In order to reduce coding and to increase the long term maintainability, AndroMDA is used to generate almost all components of the persistence layer and recurrent parts from the service facade. The annotated SMILE UML-model is translated into Spring/Hibernate DAO classes, value objects, Hibernate configuration files and Spring configuration files. Due to the flexibil-

ity of AndroMDA, also application external services, such as the user management system, have a clean integration in the model. Dependencies of internal service components on externally defined services are cleanly managed by the build system. The portability of SMILE to other RDBMS is additionally improved by the fact that data retrieval is modeled with the OCL language. The OCL query statements are translated by AndroMDA into platform specific query statements. By changing the build parameters in the AndroMDA configuration, different RDBMS specific code for data retrieval is generated. Furthermore, technology lock-in regarding the implementation of the service facade was also addressed by using AndroMDA, as the implementation of the service facade can be switched during the build process from Spring based components to EJBs. Limited effort is required to deploy the business and persistence layer into a EJB container, which could be necessary for large-scale business environments. At present, SMILE is operating on one local machine and, providing the usage scenarios don't demand it, this architectural configuration will remain. However, chosen technologies are known to work on Web server farms and the crucial distribution of HTTP sessions among server nodes is transparently performed by Tapestry MVC.
The asynchronous handling of business processes is realized in SMILE with message-driven Plain Old Java Objects (POJOs). Hence, application tasks, such as the generation of image previews, can be performed asynchronously. If performed immediately, these would unnecessarily block the responsiveness of the Web front-end. SMILE delegates tasks via JMS messages to back-end components, which perform the necessary processing actions in the background. These services are UML-modeled service components and receive messages handled by the JMS provider, ActiveMQ. ActiveMQ is manually integrated in the Spring context and persists automatically the message states into the same database, which handles the application data. If back-end tasks consume too many calculation resources, the separation of Web front-end and JMS message receiving services can be realized by copying the applications onto two different servers and changing the Spring JMS configuration.

2.2.3 Universal back-end infrastructure

During the implementation phase, some common features for Web interface development had to be implemented which can be reused in almost every J2EE application.

2.2.3.1 Multiple file transfer

The transfer of multiple files between a client machine and a server in an efficient manner is a frequently demanded use-case. An exclusively HTTP-based solution would consist as a repeated HTTP-form upload, which is unacceptable for transferring, for example, 30 to 50 microscopy images at one time. Therefore, a Java based client has been developed, which

allows universal transfer of files from a client machine to a server endpoint using the SOAP Web service technology Axis, combined with MIME encoded attachments. The client back-end uses this technology in a special streaming mode, so that even the transfer of large files in the gigabyte range remains efficient and does not demand a large amount of memory from the client and server side. The Graphical User Interface (GUI) of the multiple file transfer client can be used as a Java Applet, which is started within a Java enabled browser or as a Java standalone application, started in a normal Java virtual machine. The third field of

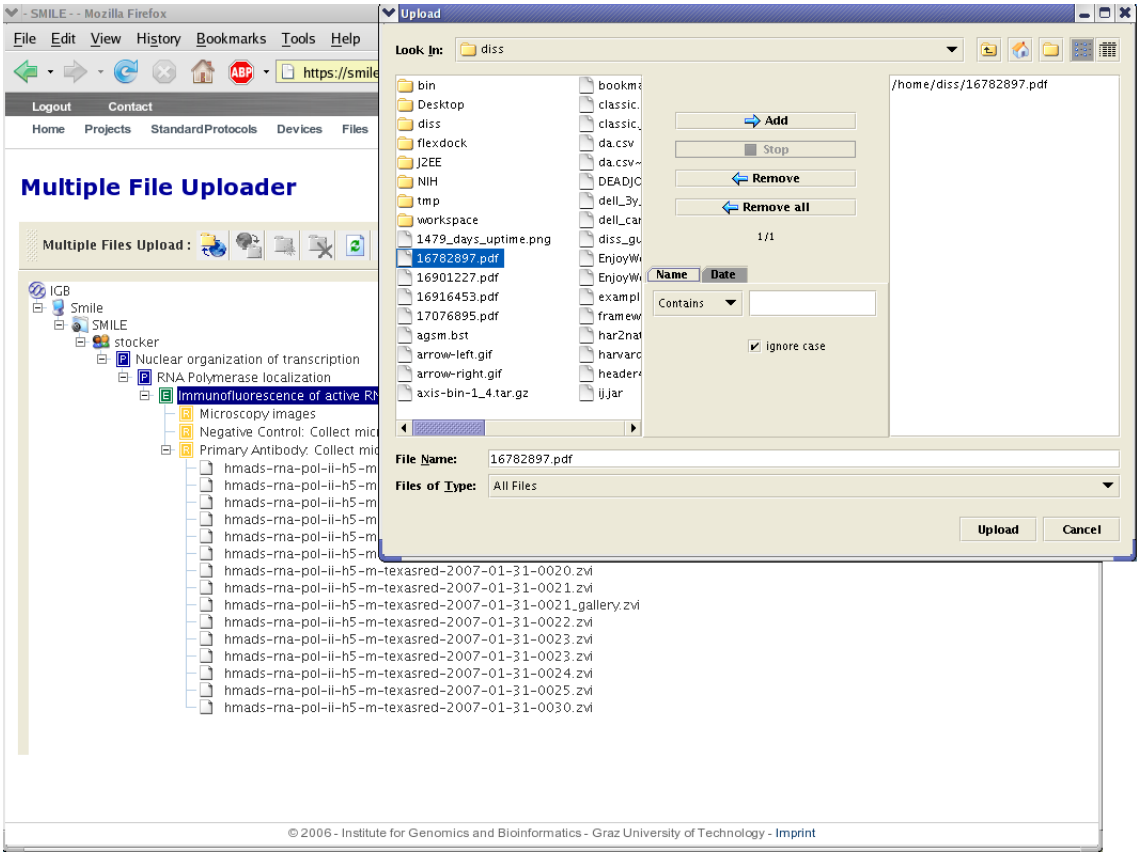


Figure 2.9: SMILE screenshot: embedded multiple file transfer Java applet

application for exactly the same GUI component can be found within the plugin code of third party applications such as ImageJ. In order to react to the changing requirements of each application, the core behavior behind the GUI is designed in such a modular manner that it can be replaced during startup. Hence, the same GUI is opening a regular “save dialog” for storing a file into a local directory, if the “Download” button is pressed in the applet mode, and a ImageJ-specific “image opening dialog” is appearing, if the same button is pressed in the ImageJ plugin mode.

On the server side, a Servlet container is necessary to house the gateway application, which acts as a dispatcher between the client software and multiple J2EE business applications.

The gateway application passes the requests of the multiple file transfer client to a specified destination application. The destination application has to implement a simple Java interface and to expose it in a Spring-remote-able manner (RMI, Hessiod, Burlap etc.) to the gateway application. There is no necessity for a developer who wants to integrate multiple file transfer functionality in the own Web application to worry about Java Applet technology or Axis Web service technology, which is completely hidden behind the gateway Servlet.
The customized interface for selecting files to upload is an adapted Java file chooser dialog, which allows users to add and remove multiple files from multiple directory hierarchies while having the possibility to apply filtering patterns to filenames and date attributes.

2.2.3.2 Spring ACEGI integration into Genome authentication and authorization system

At the Institute for Genomics and Bioinformatics user related data, credentials and authorization rules are centrally handled by the user management system Genome Authentication Authorzation System [Maurer et al., 2005]. This user management system covers all ranges of credential checks from Windows/Samba domain service authentication to a Single Sign On solution for Web authentication and authorization. Its integration into the SMILE and JClusterService, described later, was performed using a thin abstraction layer, named UsermanagementUtils. This layer allows the application administrator to switch the implementation of user management by changing one configuration parameter. This additional encapsulation of gathering user and authentication information guarantees that SMILE can be smoothly integrated in an existing user management infrastructure. The necessary data is retrieved from the external system during the first login into SMILE and is stored for performance reasons in a local database table. The authentication process itself is independent from this cached information and is performed using the original external user management system.
The Web interface and the service facade within SMILE uses the security library ACEGI for authorization. In order to combine the ACEGI technology with Tapestry and the Single-Sign-On functionality of Genome AAS, it was necessary to implement a HTTP-form based authentication filter. Consequently a Servlet filter was developed which universally handles the automatic SSO login, ACEGI context integration and the integration of user credentials into the Tapestry HTTP session. This can be reused for every Servlet-based Web application which uses ACEGI for security enforcement. Therefore, different implementations of back-end user management are provided by the UsermanagementUtils layer, and these are described in a later section.

2.3 JClusterService

JClusterService is a Java Application Programming Interface (API) developed in the course of this thesis for delegating the execution of resource intensive tasks (numerical or storage) to a remote high performance computing (HPC) cluster system. JClusterService consists of a client and a server component that are used to communicate with the HPC cluster. The server is a J2EE application implemented with Enterprise Java Beans technology. A pooled stateless session bean is remotely accessed as a SOAP Web service, using the aforementioned library Axis. A central XML based configuration file defines which applications are offered for swarmed execution on the HPC cluster. Because of the flexible implementation of this configuration, every command line oriented tool, which can be started without a graphical display on a Linux/Unix system, can be integrated as a remote service within minutes. The definition of a service comprises the definition of possible execution parameters, but also the definition of multiple required input- and output files. Dependencies between multiple jobs can be specified in such a manner that the sequential execution of multiple dependent applications can be combined to an execution pipeline (e.g. output files from the first task can be used as input files for the next application, and so on).
The client-API part of the JClusterService submits an execution task for running an application on a cluster node to the server side. Therefore, the programmer needs to specify the parameters and the input files which are necessary for the remote execution. The client transfers transparently all files to the server over a secured HTTP connection. The server retrieves the files, stores them in a central file-based directory structure and composes the queuing system-specific job execution file by using the XML job definition, the received parameters and the input files. The job execution file is committed to the locally installed queuing system, which passes the computation task to a particular compute node. The client API offers the possibility to continuously poll for the current status of the job execution. As soon as the application has finished its calculation on the node, the client can retrieve the result and output files. Finally, the application programmer using the client API is responsible for cleaning up the files previously generated during the submit and execution process.
The queuing systems Sun Grid Engine, Torque and a JMS based in-house developed system are supported by the JClusterService, and support for additional queuing systems can be added on demand. The installation of the service does not require a database management system as it manages the jobs internally in a file-based manner. By doing this, JClusterService is completely uncoupled from the queuing system and, if a job finishes the execution, the appropriate files are generated on the node. To authenticate and authorize users to the service, the aforementioned UsermanagementUtils library was developed. Because the cluster system

is completely decoupled from the internal system, it was necessary to implement a XML-based "SimpleUsermanagement", which can hold all user data in a locally stored file. In addition, the Genome AAS offers an export functionality in order to generate a data file with which the SimpleUsermanagement can be synchronized on a regular basis. If JClusterService has direct access to a Genome AAS instance, it can be switched to this authentication system solely by changing a configuration parameter. The UsermanagementUtils library was also integrated in the J2EE application CarmaWeb [Rainer et al., 2006], which offers a comprehensive R- and bioconductor-based Web service for microarray data analysis.
The ease of use of this remote procedure call approach has already convinced several programmers to use the HPC cluster within their J2EE- or Java-Stand-alone applications. Users do not have to deal with the complexity of an HPC cluster infrastructure and can perform the execution of external applications using the HPC resources as it would be performed locally. The "separation of concern" is completely fulfilled. The following sections describe use-case scenarios for the transparent integration of JClusterService within SMILE and within other in-house applications.

2.3.1 Analysis module for SMILE

JClusterService is used to delegate computationally intensive tasks of image post processing and analysis from the Web-server hosting the SMILE Web application to the attached HPC cluster. The remote deconvolution of 3D image stacks by using the text-oriented scripting functionality of the software package "Scientific Volume Imaging, Huygens" is a good example for such an analysis module. SMILE submits the two necessary image-related files, the runtime parameters and the scripting file to the cluster master, which in turn submits the task to a specific cluster node using the queuing system. This node has Huygens software installed and can start the calculation. During the calculation, a JMS based software component surveys the execution. After the job is finished, results files are transferred back to SMILE. In this way, the user does not need to install the software locally, and can proceed normally with the work. The results are calculated in the background. Results files can be re-opened within SMILE (e.g. with the ImageJ plug-in).

2.3.2 Application in comparative transcriptomics and genomics

The JClusterService has proven to be a stable and effective HPC back-end library in a large scale comparative genomics study [Sturn, 2005]. Therefore, the standalone Java application Genesis [Sturn et al., 2002], which originally offered an intuitive tool for cDNA microarray analysis, was extended by a comparative genomics pipeline. A principal objective of this thesis was to develop a comprehensive and efficient bioinformatics platform for large-scale transcriptomic

studies. The platform facilitates comparative analyses of human diseases and corresponding mouse models by integrating gene expression data with genome sequence information.
Numerous standalone bioinformatics applications, such as NCBI-BLAST [Altschul et al., 1990], BLAT [Kent, 2002], REPEATMASKER [Smit, 1993; Smit et al., 2007], GENEWISE / PROMOTERWISE [Birney et al., 2004], CLUSTALW [Thompson et al., 1994], PROF [King and M., 2000] and CHROMOMAPPER [Rieder, 2005], have been integrated as remotely executable applications through JClusterService. Additionally, biological sequence databases had to be available on the HPC storage infrastructure to run these algorithms and to perform high-throughput sequence comparisons. In order to get an idea about the stability and scalability of the JClusterService API, Sturn states that 4.965.990.400 sequence comparisons for human and 4.712.591.872 sequence comparisons for mouse could be conducted in a little over one hour on the 48 processor computing cluster. This equates to more than 1.300.000 sequence comparisons per second.

2.3.3 Application in proteomics

During the aforementioned study, it was frequently necessary to manage different biological sequence databases comprising multiple versions of the same database. This requirement was addressed in the course of development of the J2EE application MAss SPECTRometry Analysis System (MASPECTRAS) [Hartler et al., 2006], which is a platform for management and analysis of proteomic masspectrometry data. The JClusterService contributes to this application with a flexible Java-based management service for flat-file sequence databases, for example Swiss/Uni-Prot [Apweiler et al., 2004] and MSDB [MSDB, 2006], which can be remotely accessed, enabling fast sequence retrieval operations. “Regular expression”-based identifier extraction and large scale indexing mechanisms were implemented in order to achieve acceptable flexibility and responsiveness of the system.
In addition, computationally intensive tasks, such as distributed protein quantification using a modified ASAPRatio algorithm [Li et al., 2003], protein clustering using Markov Clustering and multiple alignment algorithms, were implemented using the JClusterService as the back-end infrastructure.

2.4 Production environment

Currently, the production environment in which SMILE is integrated consists of the following hardware components:

- A database server, a V880 with 4 Sparc CPUs and 8 GB of RAM from Sun Microsystems, handles the central RDBMS infrastructure using the operating system Solaris 9. Besides the PostgreSQL database, this machine additionally hosts an instance of Oracle RDBMS, which could also be used as a database back-end for SMILE. The portability will be tested, after the first release of SMILE. The server is heavily used by other J2EE applications that manage the data from the in-house microarray facility within MARS [Maurer et al., 2005] and masspectrometry data within MASPECTRAS [Hartler et al., 2006].

- The production instance of the SMILE J2EE application is deployed on a V20z dual AMD Opteron machine from Sun Microsystems, and runs under Linux CentOS 4.0 in the Java Virtual Machine 1.6 equipped with 8 GB of RAM. The RAM consumption of SMILE is around 500MB. However, as the production server handles several productive J2EE applications, the hardware must be adequately adapted to the requirements. Similar hardware is used for the central work group server that hosts the central Genome AAS.

- The storage infrastructure, on which the database and external application data is stored, resides on a SAN Enterprise Virtual Array from HP, which currently offers a capacity of 4 TB. The capacity will be extended by an additional 4 TB in the near future.

- The HPC cluster consists of one master node, with 2 Intel Xeon 2.60GHz processors and 4 GB of RAM, which serves to 24 computing nodes with the same CPU equipment as a central file server for user data over gigabit Ethernet. The main NFS data storage is a NetApp Filer F820 that supplies 1TB of disk space for bioinformatics applications as well as for image processing to the compute nodes. The master and the compute nodes are installed with Rocks Clusters Linux. Fast interprocess communication can be performed via the high speed Myrinet network. The master holds the installation of the JClusterService production instance, which is accessed from the SMILE application server via gigabit Ethernet using an encrypted connection via HTTPS. The queuing system Sun Grid Engine (SGE) handles the distribution of the jobs on the HPC cluster.

- Within the cluster compound, an additional V20z AMD Opteron system is responsible for remote 3D deconvolution, accessed from SMILE using the scripting Version of Huygens VSI software.

Chapter 3

Discussion

In the postgenomic era, high-throughput screening methods have identified a large number of genes which still lack functional characterization [O'Rourke et al., 2005]. Fluorescence microscopy can be used to localize labeled proteins for determining their functional role in cellular processes [Giepmans et al., 2006]. In order to perform a systematic analysis of generated microscopy images, structured storage of the data and organized documentation of experiments are inevitable [Eisenstein, 2006]. Therefore a platform independent computational environment for handling microscopy data was developed.
The project oriented workflow of a typical experiment guides the design of the software for data storage, data retrieval and consistent chronological documentation. This approach inherently leads to structured recording of experiments, conserving the complete experimental context of data records at all times. State-of-the-art software technology was used to implement a multi-tiered J2EE application, offering an intuitive Web-based user interface. The SMILE application back-end can be accessed through a generic programming interface, and enables the integration of external imaging tools such as ImageJ. Computational intensive tasks for server-side image processing and analysis are transparently delegated to a suitably high performance computing infrastructure, and results are stored back into the underlying relational database management system. In the field of microscopy, database management systems for structured data storage have been been established for many years. Such management systems can be subdivided into two different main categories:

- *Specialized databases for specific experimental applications* These databases focus on presenting and archiving image data produced by a specific homogeneous and well established experimental setup. High-throughput microscopy studies on the "Omics" level [Kumar et al., 2002; O'Rourke et al., 2005] and protein localization screening methods [Glory and Murphy, 2007] demand the creation of such specific data retrieval systems for optimized access and processing. These are most often targeted to a specific application

such as IMRIS [Liang et al., 2002] or [Bonetto et al., 2004; Dai et al., 2003], which focus on the high-resolution 3D reconstruction and its data management. Additionally, archiving storages like [Gonzalez-Couto et al., 2001; Habeler et al., 2002; Kals et al., 2005] are very useful for making finalized results publically available so that published data may be used as a basis for further research.

- *Generic databases for managing image data within a microscopy facility*
  Generic database systems such as OME [Swedlow et al., 2003], BioImage Database [Carazo et al., 1999] or [Lindek et al., 1999], focus on the generic description and management of microscopy images. Besides commercial products, the Open Microscopy Environment (OME) is the most vital scientific initiative in this field, which attempts to establish a standardized system for the universal description of microscopy experiments [Goldberg et al., 2005; Swedlow et al., 2006].

SMILE fits best into the latter category of databases, and can be seen as a workflow-oriented experiment management system with a particular focus on microscopy. SMILE absolutely does

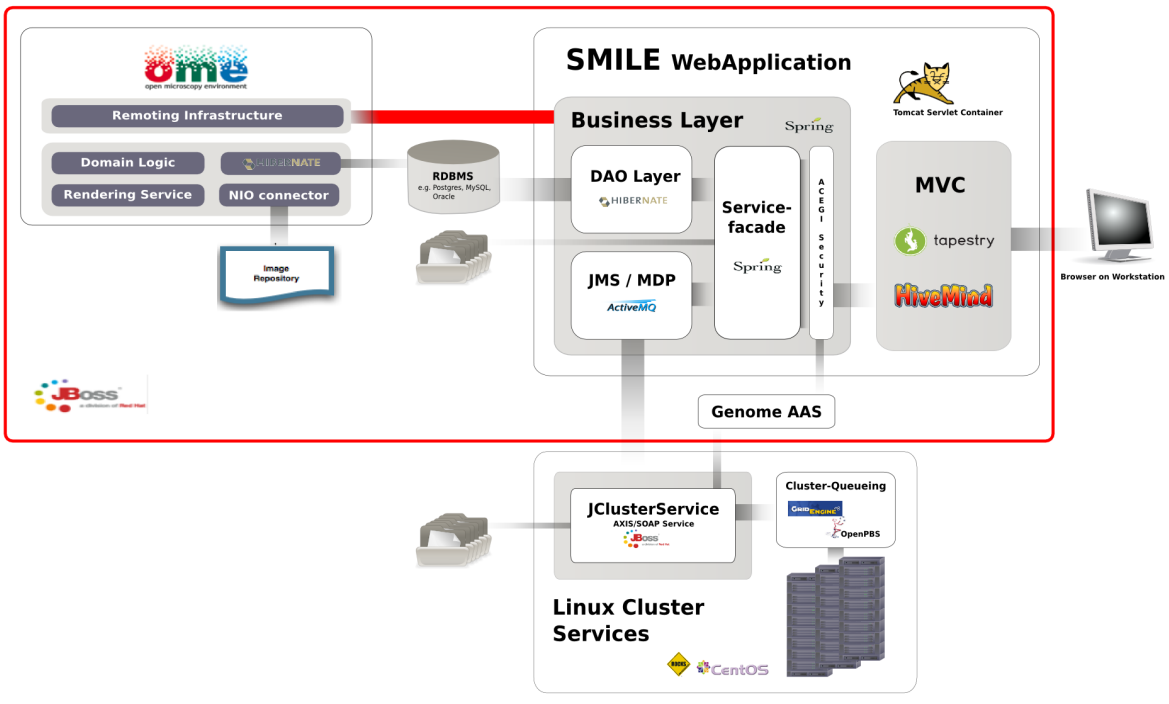


Figure 3.1: SMILE and OMERO: a possible combination to unite both projects into one J2EE environment

not compete with OME and a convergence of both projects could be achieved. This is possible as OME changed the back-end architecture to a Java based J2EE application named OMERO. Synergy effects of using both projects at the same time could be achieved in the merged config-

uration described in figure 3.1. Because of its modularity, SMILE could use the OMERO server application as a possible image storage back-end for files which are recognized as images and were, until now, handled in a hierarchical file system structure. This would result in image files stored in a standardized OME-conforming manner, although the workflow oriented acquisition and the exact conservation of experimental context is performed with SMILE.

In order to support recent initiatives for standardizing the description of microscopy experiments, such as Minimum Information Specification For In Situ Hybridization and Immunohistochemistry Experiments (MISFISHIE) [Deutsch et al., 2006], SMILE can be used to force the acquisition of the necessary data. This can be attained by defining a standard protocol, which defines the experimental workflow according to the mandatory information. For further data to be exported in a standard conformable manner, it is only necessary to define the proper mapping between mandatory protocol step parameters and the parameters demanded by the standard.

Currently, SMILE is optimized for evolving exploratory experiments in a microscopy facility, although these experiments can end up in large-scale high-throughput studies. In order to provide optimized access to the high-throughput data, adequate extensions in the direction of a data warehouse are necessary, and these can be added.

The software architecture of SMILE is ready to address issues for implementing aforementioned extensions, as it uses the model driven code generation framework, AndroMDA in central parts of its development. By extending the existing UML model with new service and persistence classes, SMILE can offer additional storage functionality for new use-cases. Frequently used functionality can be introduced by revising code templates. After rerunning the build process of AndroMDA, the centrally performed changes affect all dependent Java classes. This model-based generative development process introduces additional complexity, but subsequent development benefits from the software maintainability and flexibility. A switch from a Spring-oriented platform to Enterprise Java Beans could be realized, but is not necessary.

The Spring programming model introduces much less complexity than Enterprise Java Beans (EJB) in Version 2.1, and demands cleaner programming against custom interfaces. Support for “InversionOfControl”- and “Aspect Oriented Programming” patterns, the possible use of regular plain old Java objects, support for transparent transaction handling, increased testability outside the J2EE container, close but clean integration of frequently used Java libraries and consistent persistence exception handling over different object relational mapping frameworks are just some of the reasons why Spring technology was favored over EJB technology. Additionally, Spring profits from a large and agile developer community, which offers for every enterprise use-cases adequate solutions (e.g. security framework ACEGI, workflow engine, etc.). For persistence, Hibernate convinced because of its better performance compared to

container managed bean persistence.
Solely in distributed transactional environments like the JClusterService the introduced overhead of EJB technology could be justified and could be seen as complementary technology for special applications. With the introduction of the new EJB 3.0 standard [Sriganesh et al., 2006], several of the advantages of Spring are reduced, although the development of this standard was strongly influenced by the advantages of Spring over older Versions of EJB. For reasons of reduced complexity, functionality and expense, we favored the use of Spring as opposed to EJB in the development of SMILE.
Before beginning the development of SMILE, a careful evaluation of freely available Web technology frameworks was performed. The Web framework Struts, the component based Web framework Echo, the novel Web technology Java Server Faces (JSF) and Tapestry were all taken into consideration. However, Struts did not offer a flexible component based programming model and is based on an outdated low level request and response programming model. JSF and Echo, new frameworks at the time of development, lacked stability and support for an integrated development environment. The reasons for the decision to favor the Web framework Tapestry are: its underlying component-based programming model; the reasonable amount of pre-existing dynamic Web 2.0 components and; its long history as a component-oriented Web framework with a very active developer community. By necessity, the decision was a trade off between innovation, stability and usability.

Chapter 4

Outlook

Due to the ongoing development of biological- and IT technologies, there are continuously changing needs and requirements from the users of a computational environment. The implementation of such systems can never be seen as finished, and ideas for future improvements are already in development.

Initial requirements for experimental and analytical comparison, as well as sophisticated keyword- and ontology- based searching will be implemented. The persistence layer and the database scheme already provides the necessary back-end functionality for realizing these aims; appealing graphical Web interfaces will follow. Additionally, future versions of SMILE should contain a read-only publishing functionality so that complementary material for scientific publications, documented within SMILE, can be offered directly to the scientific community. Adequate access rights for published data must be specified. Published data should be migrated in a separate extension of the data model, which should grant optimized read access comparable to a data warehouse solution.

Server-side analysis modules within SMILE must be continuously maintained and adapted according to future evolving biological questions. Nevertheless, the flexible framework for generically storing server-side and third party analysis steps in the database could be extended by a customizable plug-in system. Similar to the plug-in system of ImageJ [ImageJ, 2007], it should be possible for a researcher versed in Java to implement their own analyzing classes. After uploading the Java classes to the server, these custom plug-ins should appear as additional analysis modules in the Web interface of the user. Concerns about security related issues for such extensions must be reviewed thoroughly.

The integration of Matlab [Matlab, 2007] as an image processing engine for analysis purpose is also a feature which could be realized on demand within a reasonable time frame. Integrative code and experience with this software were gathered in a pilot project, offering a Web frontend for classifying FRAP data with the algorithm described in [Sprague et al., 2004].

For future extensions of the image-specific data storage, the capabilities of existing standardized storage solutions should be evaluated, such as the OMERO server back-end of the OME project or the more generic approach using implementations of [JSR 170: Content Repository for JavaTM technology API, 2004]. The first possibility would inherently address the desired OME-XML export functionality. The latter approach could grant additional flexible access to the files via the WebDAV access protocol. The modular design of the storage strategy within SMILE enable the replacement or augmentation of the existing file-system-based approach used by these solutions.

Read-only accessibility of the stored files from Windows clients are already conceptually supported for the Linux platform. This solution for Linux-based SMILE server systems will use the SAMBA server software [Samba, 2007] for sharing internally stored files over the Common Internet File System (CIFS) protocol. A “shadow tree”, which is created by SMILE with Unix-native symbolic links, needs to be implemented for this purpose. This feature introduces a platform-dependency towards Unix systems and must be kept outside the core application as an optional and configurable extension.

From the long-term point of view, SMILE will support the export of stored data in order to archive unused data. This can reduce the rising and frequently underestimated efforts in managing a production system concerning storage capabilities, database performance and backup strategy. A possible archive format could be the OME-XML [Goldberg et al., 2005] format. Further extensions and changes will be strongly influenced by the field test in the in-house microscopy facility and in facilities of collaboration partners. Ongoing projects using SMILE are exploring the influence of nuclear organization on gene expression, high throughput protein screening within yeast populations and the management of Fluorescence Recovery After Photobleaching (FRAP) experiments.

Chapter 5

Methods

5.1 Computational system components

5.1.1 Java 2 Enterprise Edition J2EE

Java 2 Enterprise Edition J2EE [Armstrong et al., 2005], or the new incarnation JavaEE [Ball et al., 2006], consists of a set of specifications and technologies which define a software architecture for a more standardized development of stable, platform independent, secure and portable business applications [Stark, 2005].
The basis of J2EE/JavaEE is its application model, which is a modular and component based system. Its components interact with each other through standardized interfaces which establish a flexible and complex network of interacting modules. Because of the loose coupling of the components, further extensions and services can be easily integrated and the distribution of services on different dedicated systems can be implemented almost transparently. To reduce increasing system complexity, the J2EE application model introduces multiple tiers and suggests a three-tier architecture [Ball et al., 2006]:

- *Presentation tier* is the application layer which handles communication with the exterior of the system. This can be a Web application accessed by the clients browser or a Web service accessed by external collaboration partners. The components of the presentation tier must be mostly implemented by the programmer using additional programming frameworks or J2EE internal APIs to avoid low level implementation (e.g. HTTP communication with the Servlet API, Java Server Pages, Web service APIs etc.)

- *Business tier* components run in a J2EE environment and execute actions initiated by internal business-process-specific components or directly by external client actions transmitted through the presentation tier. The application logic is implemented by the programmer and this accesses other components within or outside the container, as well as

standard system services provided by the J2EE environment.

- *Enterprise information system (EIS) tier* comprises enterprise internal back-end systems which are essentially relational data management systems for data persistence. These can also act as resource planning systems (ERP) or other legacy systems, which must be informed by the actions implemented in the components of the business tier.

In the presentation tier, as well as in the business tier, the developer can use J2EE internal standard services provided by the J2EE runtime environment; i.e. the containers. Each container has its type-specific tasks to fulfill [Ball et al., 2006]:

- *J2EE server* is the basic container which configures and starts the Web container, eventually the Enterprise Java Bean (EJB) container. This server also manages the central resources and J2EE services described below.
- *Enterprise Java Beans (EJB) container* implements the EJB specification and takes care of the EJB specific life cycle of EJB business components.
- *Web container* manages the Web tier by deploying and running the servlet components for J2EE applications.
- *Application client container* is the container located on the client side of an application. For example, if a business partner requires direct access to internal business components, they can access the components directly from their own application client container. This container always operates outside the business application.
- *Applet container* is the container which accesses the application and its business components within the Java runtime environment of the client browser plug-in.

It is good practice to separate the latter two containers from the internal business application with a service facade in order to hide business internals from the outside. By separating application parts into different layers, the modularity, maintainability and scalability of the system can also be discretely handled (“separation of concerns”).
The primary target platform and programming language of J2EE is Java. Several open source [Glassfish, 2006; JBossAS, 2006; JOnAS, 2006] and commercial [Bea, 2006; OracleAS, 2006; Websphere, 2006] implementations of the J2EE standard are produced by different container providers. These offer more or less strict implementations of J2EE services according to the J2EE specification. The aim of these services is to relieve the developers from often repeated and complex but necessary code for low level actions, such as transaction management, resource pooling or multi-threading. Container services, described later in more detail, can be easily used within the business components, in which developers can concentrate exclusively

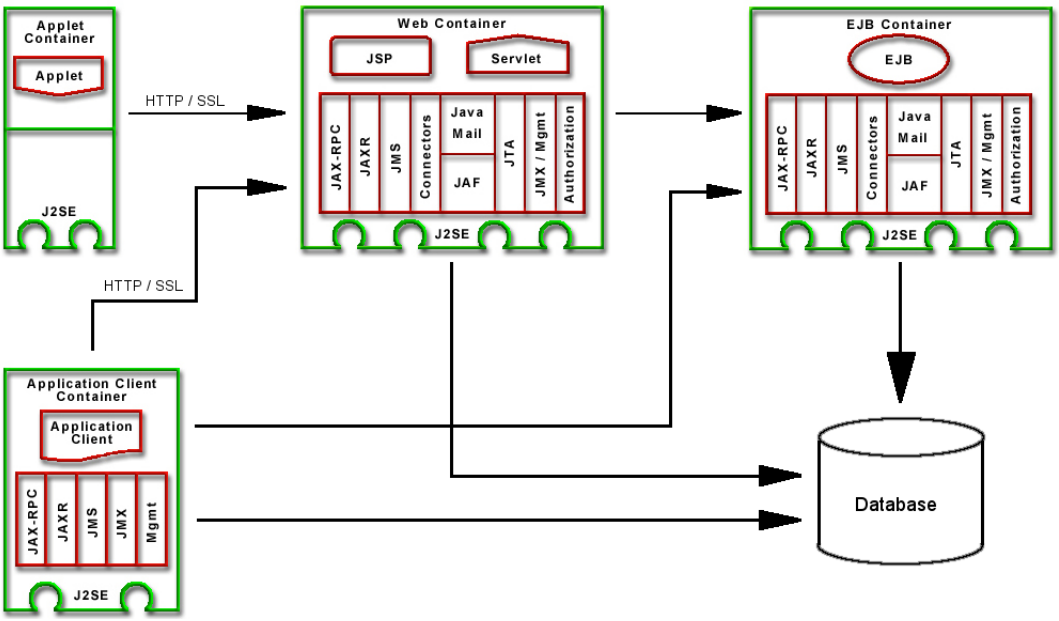


Figure 5.1: Schematic overview on the J2EE architecture, kindly provided by Daniel S. Haischt Creative Commons

on business-process-specific actions. Creation of services, dependencies on other services as well as central resource pooling are all performed by the container. Each can be centrally configured once for every application in the J2EE runtime environment. The most common J2EE services and APIs are:

- *Java Naming and Directory Service (JNDI)* is used as a directory service for registering and locating components as well as shared resources within the J2EE environment by using unified names.
- The *transaction management* via Java Transaction API (JTA) is crucial for distributed components to coordinate actions which must be performed as a single unit. If multiple actions have to be performed within one transaction, this service allows the monitoring of the ongoing actions. In case of failure, role back actions of inner transactional changes can be performed.
- *J2EE Security* is a service which offers authentication and access control mechanisms to its components using the Java Authentication and Authorization API (JAAS) for granting restricted access to dedicated users. This declarative access control system allows developers to reuse the same component within different applications, applying different access restrictions depending on the area of application. This is a typical example where the reusability and modularity of components can be demonstrated.

- *Java Management Extension (JMX)* is an API which allows the on-line management of components such as stopping or starting without restarting the complete J2EE environment. This is particularly important for high-available enterprise applications for which long down times cannot be accepted.

- *Java Messaging Service (JMS)* deals with the processing of asynchronous messages (e.g. if the duration of an action takes more than a client is willing to wait). The client can proceed without having finished the task, which can be performed asynchronously in the background. Therefore, a message can be sent from one component to a queue of message-receiving components, which process the messages depending on the message type and content. The queuing of messages is necessary as multiple components can send messages at the same time and a coordinated processing of messages can be attained.

- In this context, J2EE defines the *J2EE Connector Architecture*, the standard for integrating J2EE external components such as JMS providers into application servers, without the underlying restrictive constraints applied, for example, to EJB containers.

- *Management of components*, such as business objects during their life cycle, is a central part of J2EE from initialization during instantiation, through activation, to pooling and destroying of reusable component instances. The strictly-defined life cycle of Enterprise Java Beans in EJB containers is a good example of this J2EE task.

- The *support for automatic deployment* is another central part of the J2EE container, which demands strict directives on how a project is structured and where the internal configuration of accessed resources and libraries are found.

- The J2EE container also offers *access encapsulation* for local resources provided by the operating system, such as database connectors over Java Database Connectivity (JDBC) or file system access.

- One important characteristic of J2EE components is the possibility of distribution over multiple hosts within different Java virtual machines. Therefore, J2EE offers several technologies of more-or-less transparent inter-component communication. The most often used communication technology in J2EE 2.1 is Java Remote Method Invocation (Java RMI), which is responsible for direct inter-EJB-communication and remoting. Methods of remote Java business components can be invoked directly from other components using serialized parameters without modifying or truncating their Java types.
  Additional remoting functionality via XML and HTTP were considered by introducing new

or adopting existing Web service standards, technologies, APIs and integrated Web service services:

  - Simple Object Access Protocol (SOAP)
  - Java Architecture for XML Binding (JAXB) for mapping XML schemata to Java classes
  - Java API for XML processing (JAXP)
  - Java API for Remote Procedure Calls (JAX-RPC) for standardized remote access of RPC services
  - Java API for XML Registries (JAXR) for transparent access to Web service directory services, such as UDDI, etc.

- For HTTP communication within the Web container, J2EE defines the *Java Servlet API*, which relies on a request response programming model. On this basis, Java Server Pages, which are Java Servlets compiled from templates, and Java Server Faces support easier Web application development.
- Another commonly used service is the *Java Mail services*. This facilitates email communication from within business applications in a standardized and coordinated way.
- *Enterprise Java Beans* (EJB) are described in more detail in the next section .

The commonly held opinion that J2EE consists only of EJB and its life cycle management was inherent over many years but has changed with time [Johnson, 2002; Johnson and Hoeller, 2004]. Several frameworks and light-weight J2EE containers [HiveMind, 2006; Spring, 2006] use and partially implement J2EE standard services. In so doing, the container developers avoided some of the accepted drawbacks of EJB containers described in the following section. The strategy of just using EJBs when are reasonable and necessary was applied during the implementation of SMILE. To conclude this overview of the J2EE platform, Stark describes J2EE as the following [Stark, 2005]:

J2EE is the most spread "operating system" for (Web) applications.

5.1.2 Enterprise Java Beans (EJB)

The Enterprise Java Beans (EJB) specification [EJB, 2006] describes a framework which enables the distributed processing of transaction-oriented business logic [Stark, 2005]. The EJBs depend on an EJB container that provides the necessary runtime environment with the aforementioned J2EE services and APIs. Without the container, the EJB can not be instantiated.

The basic EJB-object consists of a Java class that holds the functionality and runs in the memory of a local or remote back-end server. Directly associated with the Java bean class is a Remote-Interface, which describes the method functionality for externally-accessing applications and a Local-Interface for faster Java-Virtual-Machine internal access. To be instantiated and initialized properly by the container, an EJB object must also implement a predefined Home-Interface to offer the container pre- or self-defined life cycle methods. In order to inform the container of the existence of a bean, a deployment descriptor must be defined for each EJB in a XML-formatted configuration file. The interface between external clients and the container is built by the bean context, which works as mediator between the client application or component and the EJB-object. Instance pooling, the monitoring of the EJB life cycle, the transition between different activation states of EJB-instances, low level remote communication, transaction management and internal service lookup details, for example, are all hidden behind the container and its application context and need not to be considered by the business logic developer. However, the EJB programming model introduces a reasonable communication and programming overhead, which slows down not only the runtime performance but also the development progress of an application.
Depending on the necessary functionalities of EJB and their application area, the EJB specification distinguishes the following basic types of distributed components:

- *Entity Beans* are used to facilitate the persistence and retrieval of business data from and to relational databases. Java Objects as well as composite objects and their relations can be persisted directly into database tables, whereas the low level JDBC communication, creation of tables and instance caching, for example, can be managed by the container. Unfortunately, this type of bean accessed remotely decreases application performance due to the communication overhead, and, unless it is hidden behind adequate access patterns, it should be avoided [Bien, 2003; Fowler, 2003].

- *Stateless Session Beans (SLSB)* are business logic centric components accessed from outside that offer business functionality through their (remote) method invocation. The typical application for this type of beans are business processes with multiple operations on different persisted objects and executed within one transaction. It is important to note that a SLSB is not associated with a server side state and can not hold session dependent status variables. It performs the execution of the remotely called methods using the applied parameters and keeping transactional integrity. Increased scalability and the ability to pool multiple instances of the same SLSB makes them very efficient and ideal for use in stateless session facades [Bien, 2003].

- *Stateful Session Beans (SFSB)* are service objects which are associated with a special

client communication. For each client, a dedicated stateful session bean must be instantiated. In this way, status variables and frequently used datasets can be cached and reused without fetching them again from the database. The drawback of this bean is the increased memory consumption.

- *Message Driven Beans (MDB)* behave similar as stateless session beans, but execute a sequence of business relevant actions asynchronously. The event driven communication between client and MDB is done via Java Message Service (JMS). A client, for example a facading SLSB, sends a message to the JMS message queue in order to start an action executed by one of the MDBs. The client SLSB ends its execution and its associated transaction after having submitted the message to the JMS Service and is ready for the next request. The JMS service delivers the message from the queue to the MDB and guarantees, due to internal persistence, that messages do not get lost while awaiting processing in the JMS queue. This special type of SLSB is very useful in increasing the system response for a user while executing long lasting jobs or resources that can be accessed sequentially.

All of the described EJB types are used in three-tier applications such as MARS [Maurer et al., 2005] and GOLD.db [Hackl et al., 2004], and partially within SMILE and the JClusterService described in detail in result section. The specification of EJB 3.0, with all its changes and improvements, are not detailed here as there was no final specification or implementation at the initiation of SMILE. EJB version 2.1 is described herein.

5.1.3 Spring, Java/J2EE Application Framework

Spring [Spring, 2006] is a lightweight programming framework that mainly focuses on the easier development of multi-tiered J2EE applications but can be used for the development of standalone Java applications as well. The advertising motto of its developers is shortly:
"*Spring aims to help structure whole applications in a consistent, productive manner, pulling together best-of-breed single-tier frameworks to create a coherent architecture.*" [Johnson et al., 2005]
Spring can be seen as a open source lightweight container which describes a universally usable, non-invasive and integrative meta-framework. *Non-invasive* means that the Spring programming model does not introduce mandatory code conventions like the EJB-programming model is doing it with its Remote- / Home- / Local interfaces. Reusable application code implemented as Plain Old Java Objects (POJOs) can become transparently a managed business object or enterprise service and is treated within the Spring environment as such. This includes life cycle management, declarative transaction management and dependency lookup as well as automatic dependency injection achieved by separating the configuration management from the business code. More details about this topic follow in the section "Inversion of Control".
Integrative meta-framework means in this context, that Spring does not reimplement application specific features which can be provided by already existing and well proven frameworks but integrates them in a consistent way. This is achieved by introducing implementation-independent abstraction layers most often in form of Java-interfaces which define the basic functionality e.g. data access objects (DAOs) [Johnson, 2002; Johnson and Hoeller, 2004]. Additionally Spring offers different implementations of the interfaces realizing them with existing Object Relational Mapping (ORM) frameworks like Hibernate [Hibernate, 2006], iBatis [iBATIS, 2006], etc. Cleaner programming against interfaces and modular "separation of concerns" as form of good programming practice is introduced by Spring almost automatically due to the implementation of its configuration management.

5.1.3.1 Inversion of Control (IoC), dependency injection

The central paradigm within most common lightweight containers like Spring is the "Inversion of Control" which describes a way of separating the creation and configuration of application components from the utilization within the components.
An EJB 2.1 container would also fulfill this definition because it also cares about the life cycle of business objects and services but it does it in an intrusive way demanding the EJB management methods from the application programmer. Additionally the EJB programmer has to care within the business code how to reach dependent EJB business objects defined in the container deployment descriptor. Using the Service Locator pattern can abstract the container-API-specific

lookup from the the business code but it still must be implemented by the developer.
In this context Martin Fowler coined the term of “dependency injection” in his article [Fowler, 2004] narrowing down the meaning of generic “Inversion of Control” to the fact that business object dependencies are configured externally within the container configuration. The business code itself just accesses the dependent external business objects or services via component internal methods or variables. The lightweight container takes care about the initialization of every business component according to its configuration mechanisms. The most used forms of dependency injection are:

- *Setter Injection* While using setter injection the component programmer must implement an empty object constructor and public property-access-methods (getters and setter) for dependent components. These methods follow the JavaBean naming convention “get-PropertyName” and “setPropertyName”. The property variables are populated during the initialization process of the lightweight container via calling the setter methods without and using any container specific API. This kind of dependency injection finds broad support from Integrated Development Environments (IDEs) because getter and setter creation for instance-variables can be performed automatically. Compared to the constructors Java can also inherit the getter and setter methods to child classes. On the other hand setter injection has the drawback that the programmer can not influence the order of variable initialization by the container and an incomplete configuration leads to unrecognized undefined variables which are only detected during runtime.

- For *constructor injection* every container managed component must implement a Java constructor with all its external component dependencies as constructor arguments. The lightweight container passes the external dependencies to the constructor and there the developer can decide how to deal with the external components during the instantiation process. The main advantage of constructor injection is that every dependency is set during construction time and every business object is in a consistent initialized state. Additionally “writing less code” compared to the setter injection can be also mentioned as an advantage. If the amount of dependencies of one module is reasonable high, this kind of injection leads to unmanageable multi-argument constructors. Further on constructor argument names can not be accessed via Java introspection and optional fields result in multiple constructors which are not inherited to sub classes.

Spring supports both kind of dependency injection which are used in SMILE dependent on their advantages.

5.1.3.2 Spring modules

The Spring application framework can be divided into several functional distinct units which are tied together via the Spring Core. The typical modular and layered structure was consistently put into practice during the implementation of Spring itself. Thus it uses internally the same techniques for defining, creating, initializing and managing business components and services.

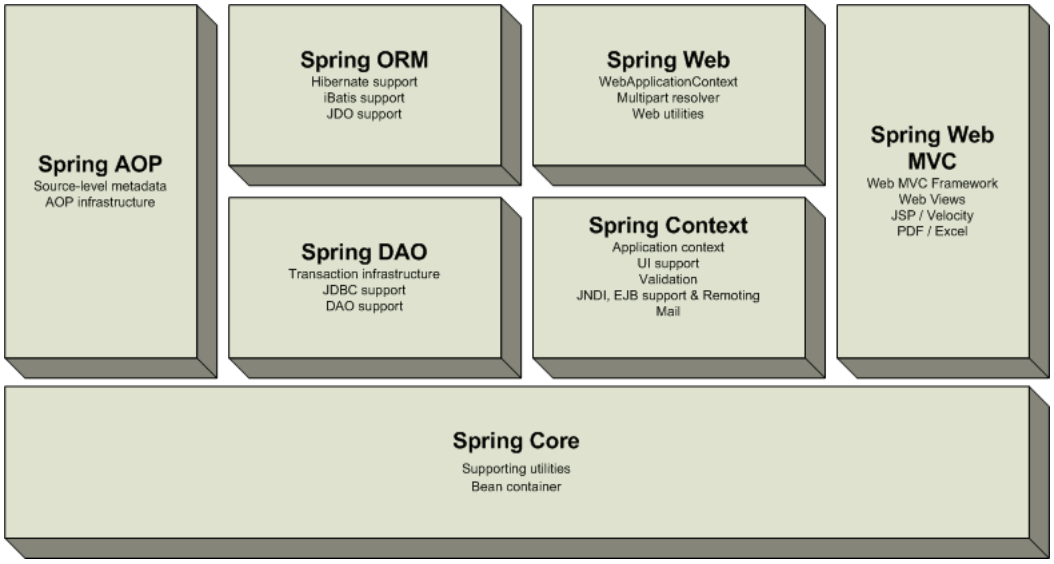


Figure 5.2: Overview on the Spring framework, from Spring reference [Johnson et al., 2006]

- The *Spring Core* is the central "Inversion of Control" container facilitating the flexible configuration management for custom-defined POJOs as well as for services provided by additional Spring internal modules. The complexity of the individual POJOs and its granularity depends on the developers. The framework just resolves the external dependencies and passes them to the dependent POJOs via "Dependency Injection". [Johnson and Hoeller, 2004]
  During the startup of the container Spring uses a BeanDefinitionReader to get an internal meta-configuration used by a BeanFactory to instantiate the defined POJOs, Spring beans. The meta-configuration model is useful to provide different methods of configuration like XML-file or property-file configuration but also code integrated annotation configuration. Each bean can be instantiated either as singleton, what defines the Spring bean as factory-wide unique object, or as prototype object, which is a poolable object that can be instantiated multiple times to improve performance reducing instantiation overhead.
  Different implementations of BeanFactories can be instantiated in every tier of a multi-tier application and introduce in that way automatically a layered structure.

- *Spring Application Context* extends the BeanFactory and introduces in Spring J2EE container qualities. Integrative message lookup for internationalization (i18n), eventing mechanisms, capsuled file- and resource access [Johnson et al., 2006] are some features which this module provides.
  Further on the Spring context offers consistent integration of enterprise services like JavaMail, data source management for databases, JNDI-access infrastructure, external-framework-based scheduling, but also sophisticated remoting possibilities. This is the basis for accessing distributed components running on different servers like EJBs in a local or remote container. The communication specific part of remoting is hidden from the developer behind a generic abstraction layer which allows to exchange transparently the underlying communication technology. The application developer gets from a communication technology specific proxy factory [Johnson et al., 2005] a proxying object which implements the remote service interface and delegates each method call to the accessed service. Method arguments, return values but also exceptions are passed back and forth between the proxy interface and the calling client object.
  Spring offers RMI communication integration for remote EJB access, but also more lightweight communication frameworks like the binary protocol Burlap, or the XML-based protocol Hessiod are available. Additionally a simple HTTP invoker interface, which transfers Java-serialized objects via HTTP, and acts as a simple replacement for Hessiod implementation, can be used. Finally Spring has also integrated SOAP/Web service remoting using the JAX-RPC specification.
  Due to the flexible proxy implementation Spring can transparently integrate remote services and creates a flexible access strategy for components like EJBs. Both technologies must not be harshly competitive but can be used complementary to separate service implementation and its access in a elegant way [Wengatz, 2005].

- *Spring Aspect Oriented Programming (AOP)* takes most of the responsibility for Spring not being intrusive. By implementing AOP-“method interception” Spring enables the programmer to add additional functionality before and after the call of an object method. This additional functionality might be logging- or performance-tracking-code but can also be more complex like declarative transaction management, or method/rule-based access control (see section with ACEGI). The latter two functionalities add transparently the typical J2EE service aspects to simple POJOs and are therefore very important for replacing the EJB programming model by the lightweight Spring approach. The basic functionality of POJOs can be simply tested outside the J2EE environment as a separate unit. After that the integration into Spring makes a service bean with transactional integrity etc. out of the regular bean. AOP used with method interception can be seen as a powerful

completion of the object oriented way of thinking. Spring uses internally dynamic proxies and CGLIB byte code generation at runtime for the internal AOP class enhancements. It integrates also AspectJ, AspectWerkz and JBossAOP as external frameworks.

- *Spring DAO* One strength of Spring is its way of offering advanced abstraction and help on the persistence layer which should separate the business logic from the underlying implementation of persisting. Therefore Spring has a clean implementation of the Data Access Object pattern described by Johnson [Johnson, 2002]. The business components just use the basic data-persistence and -retrieval methods on a business object basis and are not getting in touch with relational databases and their exceptions. This unique data access exception hierarchy is used in a very consistent way for all methods of Spring accessing a relational database. From JDBC, to Hibernate, from iBatis to JDO, the principal way of persistent data management remains the same. The different implementation of DAOs are using the optimized framework strength just behind the DAO layer.

- *Spring Object Relational Mapping ORM* One commonly used strategy of persisting data using an object oriented programming language like Java is the object relational mapping. This method cares primarily about mapping Java objects and their inheritance hierarchy into a relational database schema but also about mapping Java datatypes into vendor specific SQL datatypes of different relational database management systems. This functionality is offered by several ORM frameworks like Hibernate, JDO, Oracle TopLink, Apache OJB, etc. but they are all hidden behind a common OR mapping layer. Session management for efficient and proper transaction management, data source resource management integrated transaction management exception wrapping and easier testability are some reasons that justify the significant added value of this layer [Harrop and Machacek, 2005]. In a J2EE environment the combination of Spring and Hibernate is used very frequently and emerged to be the persistence stack of choice for SMILE.

- *Spring Web and Model View Controller (MVC)* offers out of the box Web related utility for the Web presentation layer and a complete implementation of the Model View Controller pattern evolved to address the limitations of former de facto standard Struts. Spring integrates also different additional frameworks like velocity template engine for template based JSP generation, iText for PDF generation etc. for easier handling with different presentation requirements. Features from this modules are just used for integrating the Spring container into the Tapestry presentation layer of SMILE.

5.1.3.3 ACEGI Spring security and Genome authentication and authorization system

The ACEGI Security project [Acegi, 2006] is the Spring security framework and takes care about most of the security precautions an application has to provide. It features the basic requirements like user authentication and authorization, but also more advanced requirements like service layer security, domain object instance security and its access control, Web request security, channel security and human detection capabilities.
The authentication mechanism within ACEGI introduces three core interfaces:

- The *Authentication* interface is a Java interface which defines a security class, containing identity specific data like unique user names, session IDs and its credentials but also the application specific authority attributes the authenticated user has obtained from the administration.
- The *Authentication Manager* is the central interface for a class which processes an authentication request. This authentication manager gets an authentication object and checks the credentials using an implementation specific authentication provider. If the authentication process was successful the authentication manager populates also the authority attributes of the authentication object.
- The *Authentication Provider* decouples the authentication implementation from the ACEGI authentication process itself. ACEGI offers out of the box several types of authentication providers accessing different types of user back-ends:
  - DAO Authentication Provider using predefined Spring internal data access services with locally stored users from database tables (using JDBC) or from XML-files.
  - Container based authentication using JAAS and JBoss authentication
  - Common Authentication Service (CAS) Provider from Yale University which provides a widely used and full featured user management system with broad platform support.
  - Principal Authentication Provider
  - The Lightweight Data Application Protocol (LDAP) provider offers a generic integration of users stored in de facto standard and universally usable LDAP repositories like Active Directory from Microsoft or Novell E-Directory.

  If this existing authentication back-ends are not sufficient or an application has to authenticate its users against a legacy user management not internally supported by ACEGI e.g. Genome AAS [Maurer et al., 2005] the developer can implement the Authentication-Provider Interface for authentication purpose. If authorization is also managed centrally by

an external user management, an AuthenticationManager must be additionally extended with the appropriate features.

This back-end authentication providers can be used for several authentication mechanisms like HTTP-form-based authentication, basic HTTP authentication, HTTP digest authentication, anonymous authentication (if no user is really required), X509-certificate based authentication and CAS-single-sign-on authentication. Depending on the access technology ACEGI offers its direct integration with wrapper classes, HTTP transparent authentication filters and context integration filters.
When the authentication process is finished successfully through one of the above mentioned mechanisms, the authorization is also supplied with ACEGI mechanisms. It can be distinguished between pre-invocation handling and post-invocation handling. Pre-invocation handling means when security constraints are checked and, if not allowed, denied before executing a method call. Post-invocation handling means when after a method call the instances of object instances are checked according internally managed Access Control Lists (ACL). Object instances which should not be accessible by the authenticated user are filtered out after the method is called.
The integration of both handling types is realized Spring-typically in an non-invasive manner. Method security on a service access level can be achieved via security interception, where the business object is enhanced by security relevant aspects. For each method the external Spring configuration can define access constraints which guarantee that only authorized users can access the method. Therefore the business object code must not be adapted if security relevant changes have to be implemented. Other forms of ACEGI security integration in technology specific manners, like tag-libraries for the model-view-controller Struts or flexible Java5 security annotations but also SSL channel security, are directly provided by ACEGI as well.

5.1.3.4 Message driven POJOs

The concept of Message Driven POJOs (MDP) can be seen as Spring replacement for Message Driven Beans described in section “Enterprise Java Beans”. Spring uses technologically the same concepts of Java Message Service API in order to realize a loosely coupled communication between business components. The communication is handled by a JMSProvider which represents a Java Message Oriented Middleware (MOM) service and offers two models of message communication [Johnson et al., 2005]:

- The *JMSQueue* is a point to point communication channel which receives JMSMessages from a JMSProducer. After receiving the message the JMSQueue queues the JMSMessages persistently for being delivered sequentially to a JMSConsumer. Queued JMSMessages remain in the JMSQueue until a JMSConsumer is reading them out.

- On the other hand the *JMSTopic* is a "publish- and subscribe"-based communication channel where JMSProducer can publish JMSMessages. The JMSTopic works on a subscription basis and if there are multiple JMSConsumers registered, they all will get delivered the published JMSMessages. If no JMSConsumer is registered the JMSMessage gets lost.

Message Driven POJOs are JMSConsumers which have not to implement any Business Interface like Message Driven Beans and can be easily developed and tested outside the container. They accept Java serializable objects used as JMSMessages and react on them. This is frequently used in a Web environment when the user interfaces submit long lasting or computational intensive actions to the background business objects. Hence the user interface is not blocked and can regularly continue the work while the message receiving MDP is processing the JMSMessage. This can happen immediately, at a later time point or even externally. In earlier versions Spring supports the developer just with offering JMS client functionality where a JMS template facade is hiding the real JMS communication from the business code. The server side of JMSProvider services [Harrop and Machacek, 2005] has do be integrated into Spring via third party lightweight JCA containers like Jencks [Jencks, 2007] with embedded JMSProviders like ActiveMQ [ActiveMQ, 2007]. J2EE platform specific transaction management can be used directly via transaction interception and realizes therefor a full featured replacement for Message Driven Beans.
Since version 2.0 a clean and direct integration of JMSProviders is supported within Spring and will be used within SMILE as soon as the version upgrade is possible. This integration of JMSProvider services would not have been necessary if a typical J2EE application server would have been used, because it would come with an already integrated JMSProvider service.

5.1.4 Relational database management system

The basic aim of J2EE multi-tiered business application like SMILE, MARS [Maurer et al., 2005] or GOLD.DB [Hackl et al., 2004] is collecting and permanently persisting data in a structured and organized way. Sophisticated relational database management systems (RDBMS) are used as back-end for managing the efficient storage and retrieval of data. An RDBMS is a software package which allows to create and delete databases, to define their structure via a Data Definition Language (DDL) and manipulate the containing data via a Data Manipulation Language (DML). The third task of an RDBMS is providing a Data Control Language (DCL) in order to control data access and to define data storage and commit strategy [Kuhlmann and Müllmerstadt, 1999]. The American National Standards Institute has defined the Structured Query Language (SQL) to unify these manipulation languages for RDBMS from different manufacturers. Unfortunately, almost every RDBMS product uses its own SQL dialect and makes

database independent programming difficult. Beside offering a SQL-based data access, the RDBMS has also to care about data integrity, data security and business constraint safety for independent and often concurrently accessing database clients.
A database defined by the DDL comprises a collection of two-dimensional tables so called relations, which represent real world objects mapped with its describing attributes into a relational data model. The columns of a table describe the attributes of a mapped object called entity and the table rows describe stored entities identified by unique keys. The tables can be linked together according to their real world dependencies as one-to-one, one-to-many or many-to-many relations. The links between this tables are established by adequate primary keys, foreign key rows or reference tables.
To avoid redundancy during data storage the data derived from real objects must be analyzed and transformed in normalized form. The basis of this normalization process and its resulting relational data model was developed by E. F. Codd [Codd, 1990]. He applied mathematical principals like set theory and predicate logic to achieve flexible and systematic data collection and functional data retrieval. The original relational model approach was realized by different commercial manufacturer [MSSQL, 2006; Oracle, 2006] and open source projects [MySQL, 2006; PostgreSQL, 2006] and is still state of the art.
The internal representation of data, the implementation of indexing for faster search processing and the key generation management differs from RDBMS to RDBMS and has to be addressed either by the database accessing application itself or by an intermediate data access layer.
Java solves the generic relational database access and datatype conversion by using the Java DataBase Connectivity (JDBC) API. The JDBC API offers relational database access classes generically written for arbitrary Java client applications. Each RDBMS manufacturer implements customized JDBC drivers for the optimized access to their specific back-end implementation. The application developer on the other hand has not care about the implementation details and can use transparently the dynamically plugged in JDBC Driver through the JDBC Driver Manager. The JDBC architecture is strongly inspired by the Open DataBase Connectivity (ODBC) API which can be seen as the de facto standard for platform and programming language independent, SQL-query-based database access.

5.1.5 Object relational persistence with Hibernate

JDBC solves the problem of generic database support very well but offers only SQL-query based access to database entities. Simple persistence functionality as Creating, Reading, Deleting and Updating (CRUD) has to be reimplemented by hand coded JDBC individually for every persistent Java class. To reduce the manual programming effort Object Relational Mapping (ORM) frameworks were introduced. Johnson describes the underlying principal

as follows: *"O/R mapping is the attempt to map the state of Java objects onto data in an RDBMS providing transparent persistence"* [Johnson, 2002]. In this context Johnson coins also the term "Object/Relational Impedance Mismatch" which describes the discrepancies between the object-oriented programming model of Java and the table-oriented model of relational databases. Questions for the degree of granularity for object mappings into tables (when to use just additional columns or new tables), questions for the mapping from SQL-query results to result objects, but also questions for the mapping of object relations and their inheritance have to be answered [Bauer and King, 2004]. Caching and retrieval-deepness strategies on object level but also aggregate functions and transparent persistence of contemporaneously accessed objects are some more problems which implementations of ORM tools have to face. The criteria for choosing an ORM tool out of the range of persistence frameworks depend mainly on how the manufacturers have solved the problems of paradigm mismatch and its intuitive usage.
Hibernate [Hibernate, 2006] is one of the wide spread ORM frameworks, which has solved the problem of automatic and transparent persistence. Also the problems of paradigm mismatch to a large degree avoiding the drawbacks of bean- or container managed persistence are considered. It is an open source project, which has realized the mapping of the data representation from arbitrary POJO objects to relational data representation by using descriptive XML mapping files. Out of these meta data Hibernate creates individually for almost every RDBMS optimized SQL statements which are taking advantage of the specific capabilities of the underlying management system. Differences between SQL dialects can be observed during the selection of adequate SQL datatypes for class attribute mapping, in the various strategies for primary key generation and at non-SQL-conform peculiarities of different RDBMS. The developer just specifies in the central configuration file hibernate.xml the SQL dialect and the framework is taking care of the SQL specifics. Also on the Java side the POJOs don't have to fulfill special requirements like derivation from specific persistence super classes or implementation of special interfaces. Various inheritance mapping strategies are supported as well as different association types like one-to-one, one-to-many or many-to-many object relations. The developer can even specify the directionality of associations, different caching mechanisms for repeatedly accessed objects and their associations (lazy initialization). Hibernate can operate in J2EE environments and in standalone applications without any restrictions and considers nevertheless different kinds of environment-dependent transaction management for conserving data integrity.
Data retrieval can be performed in Hibernate by using three different approaches:

- The *Hibernate Query Language (HQL)* is a SQL-like query language which represents the consequent continuation of the object oriented approach for data retrieval. The complete persistence cycle of the internal state of objects can be performed directly on an

object basis without knowing the underlying relational database details like tables and columns. Objects are populated and persisted automatically using SQL-close keywords and clauses. Select conditions on objects and their properties return again result objects or object sets on which SQL-typical functions like aggregate functions or sorting clauses can be additionally applied. Advanced features like result pagination, object fetch joins, cartesian products or stored procedure calls are implemented in HQL. The range of regular, SQL-typical functionalities are extended by polymorphic queries which integrate the object oriented inheritance idea in the retrieval strategy of Hibernate.

- The second data access strategy can be seen as *Query by Criteria (QBC)* and *Query by Examples (QBE)* which represents a powerful template based retrieval system using the Hibernate internal criteria API. Partially set properties of a template object are used internally as criteria for the object selection.

- Finally also *native SQL* can be used to access objects from the database via Hibernate.

Performance, scalability, reliability and a huge user community of Hibernate influenced strongly the further development of other persistence frameworks in the J2EE environment. Even some implementations [JBossAS, 2006] of the new EJB 3.0 bean persistence rely on the capabilities of Hibernate.

5.1.6 J2EE Web services

A Web service can be described as a communication method using XML documents to exchange information between loosely coupled applications over the Internet. The big advantage of Web services relys on the fact that Web service consumer and producer operate on arbitrary XML documents which can be processed independently from its runtime platform and programming language. To guarantee a standardized interaction between Web services the *Simple Object Access Protocol (SOAP)* [SOAP, 2003] defines the formal packaging structure of the exchanged XML messages and its processing model. SOAP wraps all data into an XML envelop which again contains a header and a body section. The header contains contextual information how the processing pipeline at the document consumer side should treat the body of the message. This includes information about Web service routing, transactional process integrity but also information about authentication, integrity via digital signatures and authorization. The body section of a SOAP message contains the data to exchange, which must be encoded according the SOAP standard. Erroneous and misdirected messages and the mismatch of protocol version lead to fault messages, which can contain numerical error codes, textually described error details and involved actors [Snell et al., 2002].
Typical application scenarios for Web services are Remote Procedure Call (RPC) constellations

and Electronic Document Interchange (EDI) processes. During the Java based RPC communication the Web service client invokes methods on remote Web services by sending SOAP messages formatted according the JAX-RPC standard and containing method name, method arguments and their type description. Results of such remote method calls are sent back to the client via SOAP messages and conclude the procedure calls. JAX-RPC is extensively used in the high performance Grid computing field but also in the area of business process communication and customer integration. The EDI scenario on the other hand focuses on the exchange of electronic documents which must be transferred in a platform independent way from the producer to the consumer Web service. The exchange of automatically generated contracts or of cryptographic key files between public key infrastructure providers are examples for the application of EDI [Snell et al., 2002].

Web services are typically using the Hyper Text Transport Protocol (HTTP) for the XML document exchange, but the SOAP document content is independent from the underlying transport protocol. Therefore FTP, SMTP but also Instant Messaging Protocols can be used for transferring the SOAP XML documents. Beside SOAP there are additional programming technologies and standards involved in Web services: the *Web Service Description Language (WSDL)*, the *Universal Description, Discovery and Integration (UDDI)*, the *SOAP with Attachments API for Java (SAAJ)*, the *Java API for XML Registries (JAXR)* and *the Java API for XML Processing (JAXP)* [Monson-Haefel, 2004]. WSDL is used for describing platform independently the data content transfered in the body between Web services. This ranges from descriptions of method requests and their responses, over argument descriptions to default and custom defined datatypes. Starting from the WSDL description SOAP clients can be generated almost automatically in different programming languages like Java, .Net or Perl. In dynamic business environments UDDI defines standardized methods to publish location information about public Web services to central Web service registries. The access of UDDI registries is facilitated by using the JAXR programming interface. The process of SOAP conform message generation is supported by the SAAJ API which is a low level SOAP programming interface and additionally allows to attach Multipurpose Internet Mail Extensions (MIME)-encoded files to the SOAP message envelop. This feature can be used for transferring files of reasonable size in an efficient way bypassing the memory intensive XML parsing effort.

The open source project Apache eXtensible Interaction System (Axis) [Axis, 2007] was one of the first Java frameworks which supported the Java Web service development regardless of implementing the server side or the client side. On the server side Axis can be integrated in every Java Web container which supports the Java servlet specification. Once integrated in the server Axis can easily expose business objects as SOAP Web services. Just by defining a Web Service Deployment Descriptor (WSDD) Axis can directly access existing business code which

can be implemented with numerous different back-end technologies like EJB Stateless Session Beans or Spring Beans. Axis offers a stable API hiding the SOAP internal communication details from the application developer. Beside the unproblematic integration on the server side Axis offers also the possibility to extract the WSDL definition of the SOAP exported business object. This generated WSDL file can be used as a input file for the Axis utility WSDL2Java which automatically generates the client code for accessing the defined Web service. Changes in the API of the exposed Web services can be considered on the client side by rebuilding the artifacts of the generated client classes. Once the developer has overcome the steep learning curve of creating the WSDD for the Web services, Axis improves the productivity and reduces the effort of API changes in the development process appreciably. Finally it must be mentioned that Axis supports the transfer of files via SOAP RPC using a powerful implementation of MIME encoded file attachments. The JavaMail API is internally used to recode the file content into MIME encoding and the file is added as attachment to the SOAP envelop of the method call. The use of this RPC/attachment implementation leads to a memory optimized high-performance file transfer which can handle files in the range of gigabytes without reaching memory or timeout limits.

5.1.7 Tapestry, a component oriented Web framework

Tapestry is a component oriented Java framework for the development of dynamic Web applications [Tapestry, 2006]. It can be simply integrated into a J2EE environment via the Java Servlet API and can therefore operate in a full-featured J2EE application server as well as in a standalone servlet container like Tomcat [Tomcat, 2007]. By using one central servlet which takes all browser requests the Tapestry servlet acts as a typical *Front controller* [Fowler, 2003, pp. 344–349] which separates the Web tier of an application from its Business tier. The literature describes this distinct separation of presentation tier (view) and the presentation-independent application tier (model) via a controller as *Model View Controller (MVC)* pattern [Fowler, 2003, pp. 330–332]. Unlike other Web frameworks [Struts, 2006] Tapestry hides the typical request/response programming model originated from the Java Servlet API behind an event driven component model [Springmann and Tiggers, 2005].
The front controller servlet takes an incoming request and extracts all necessary information for further processing from the request URL, the request parameters and the persisted HTTP session values. With the help of this information the servlet identifies internally the responsible Web page, its containing components and the necessary services which are instantiated and initialized dynamically according to the request. Components are the basic units of Tapestry and are objects which inherit internally used life cycle methods from a common super class. This is necessary to hide the completely transparent component pooling and guarantees increased performance and scalability of the framework due to the reuse of pre-instantiated components. In addition, each component object itself is responsible for its individual rendering behavior and has to implement a rendering interface. The methods of this interface are called recursively from the Tapestry servlet on each involved component during the creation of the HTTP response. The idea of the reusable component oriented approach, which has its roots in the development of rich client Java applications, is realized consistently within the framework. Even the Web pages internally are components which can recursively contain predefined or self-defined components [Lewis Ship, 2004].
The rendering of Tapestry pages and its components is supported by the implementation of the *Template View* pattern [Fowler, 2003, pp. 350–360]. A page but also a component can be defined by an HTML template, an XML-specification file and a Java class. The HTML template defines the layout of the Web page and can be created by regular Web designers without knowing about the dynamic content. In this template file HTML tags are augmented by the naming attribute “jwcid” in order to be replaced dynamically by the framework. The name used for the jwcid attribute is used in the XML-specification file as the link between the component instance object and the layout. The specification XML-file is used during the initialization phase for “binding” component arguments. It defines the assembly of dynamic components, static text

and above mentioned component objects. Even the browser events are routed to the responsible components and listener methods using the binding information from the specification file. The method arguments of the view-related page/component objects are bound during the request processing cycle by the Tapestry servlet and the execution of the event listener methods is performed. Within these methods the component objects access the presentation-independent model classes and retrieve all necessary business information for rendering the Web page. A clear separation of concerns with a distinct tier structure (browser view, servlet-based view controller and business model) can be realized inherently without getting in touch with HTTP requests and responses. The developers have not to care about typical tasks in the Web application field like URL generation, session state persistence or page dispatching according URL parameters. An other advantage especially in projects with bigger teams brings the separation of design and page logic where Web designers can concentrate on the layout and the application logic developer can concentrate on the programming domain.
A set of predefined base components form the necessary tools to build regular HTML Web applications. They can be separated by their functionality in two main classes:

- The layout components are responsible for producing visual output. Typical examples for layout components are the Tapestry component equivalents for almost all HTML tags. The Shell component can be seen as envelop of a typical Web page where commonly used HTML header settings like JavaScript inclusion and CSS layout referencing takes place. Within the Shell, the Body and its derived components can include simple HTML components like Table-, Image-, TextField-, TextArea-, PropertySelection-, RadioGroup- or Checkbox structures and more complex dynamic structures like DatePicker-, Rollover-, Tree-, Datagrid- or Pallette components.
  In addition to HTML components Tapestry offers also Wireless Markup Language (WML) components for applications accessed via the Wireless Application Protocol (WAP). This protocol is especially designed for the use with mobile devices and can be used within Tapestry to offer an additional view on the same model.

- The page control components introduce loop constructs like For/Foreach- loops, conditional constructs like the If/Else- components, or RenderBlock / RenderBody for grouping and dynamically updating of multiple layout components. In addition view logic for page navigation can be directly integrated via special HTTP links, which call listener methods on Component or Page objects. Besides calling listener methods dynamic Form components offer an universal validation of form values on client and server side. Validation is automatically performed by the framework just by specifying value constraints for the input fields.

Besides the framework internal components Tapestry can additionally resort to several third party libraries provided by the strong user community. These libraries offer ready to use static components [JSCookMenu, 2007] but also dynamic components relying on Asynchronous JavaScript and XML (AJAX) [Tacos, 2007]. Due to the long history [Onstine, 2006, chap.1] of Tapestry the developer can find for almost every problem an already existing component which can be simply reused and adapted to the requirements of the Web application. The power of custom defined components can be demonstrated when self-defined components are defined for repeatedly used layout groups. Code and layout maintenance is quite simple because changes on those frequently used layout components can be performed once and the changes are taking effect in every place where the component is used.
The back-end infrastructure in Tapestry is managed by HiveMind [HiveMind, 2006], an Inversion of Control container which was initially created to meet the special requirements of the Web framework. Meanwhile it evolved to a separate Apache Foundation project which can be universally used for the configuration and management of Java application modules. Different instantiation strategies (singleton instantiation, just in time instantiation and instance pooling), Inversion of Control-based object initialization and non intrusive thread safety are some of the strongest features of this lightweight Java container framework. Starting from XML configuration files the HiveMind micro kernel combines simple, almost independent Java modules to an interacting POJO service environment which is designed for the deployment within a J2EE environment. The creator of HiveMind describes the framework as follows: “HiveMind allows you to create more complex applications, yet keep the individual pieces (the individual services) simple and testable. HiveMind encourages the use of common best practices, such as coding to interfaces, separation of concerns, and keeping code highly testable without a special container.” [HiveMind, 2006].
Even the integration of an additional Inversion of Control container like Spring [Spring, 2006] can be achieved by adding a bean provider service definition to one of the configuration files. Thus HiveMind unites the advantages of both IoC container frameworks and integrates them transparently into the Tapestry controller servlet. Pages and components can access Spring service objects from the business tier either via a binding prefix (“spring:”) in specification files or within the program code via service-injecting Java annotations. Also Spring ACEGI method security can be assured in Tapestry’s listener method calls just by adding Java annotations to the method code.
The development process with Tapestry is supported by returning always clear feedback with line precise error reporting [Tong, 2005]. The automatic integration of internationalization, JSR 168 Portlet support, Object Graph Navigation Language (OGNL)-support, automatic variable persistence on the client and server side, and bookmarkable URLs are additional Tapestry

features. OGNL is used for directly accessing Java objects and their properties within the text-based Tapestry templates and specification files. Further details in [Lewis Ship, 2004; Tong, 2005; Tapestry, 2006].

5.1.8 Model Driven Architecture

The Model Driven Architecture (MDA) standard initiative was created by the Object Management Group (OMG) [OMG, 2006] to handle the continuous change in technology and business requirements over time. The concept of MDA has been established on the basis of a clear separation of layers which describe the same software project on different abstraction levels. This method promises better maintainability and easier technology migration because functionality and application behavior is defined on higher abstraction level. Technological implementation details on the other hand are hidden behind lower abstraction levels.
The basis of Model Driven Architecture are models which describe the target system on different abstraction levels [Seidewitz, 2003]:

- The **Computational Independent Model (CIM)** is the mental model out of the mind of software architects and put down on paper. It represents a textual description of the system and its basic functionalities.
- **Platform Independent Model (PIM)** On the next abstraction level the PIM describes the business processes of the system and its functional units without any platform specific implementations details using UML as structured language.
- **Platform Specific Model (PSM)** The PSM goes into the details of platforms and provides already a platform dependent view for architectural aspects and services.
- The **Code Model (target Platform)** represents the final implementation of a system within a platform, using the platform specific programming language and deployment configuration of the target runtime environment e.g. Java EE [J2EE, 2006] or .Net [MSDOTNET, 2006].

The MDA development process [Mellor et al., 2003] comprises the generation and the successive transformation [Sendall and Kozaczynski, 2003] of these models from a higher abstraction level to a lower ending up in a coded application. In order to build consistent, exchangeable and vendor neutral standards for the application of model-driven software development OMG has released an MDA Guide [Miller and Mukerji, 2003]. In this guide OMG uses internally defined standards like Unified Modeling Language (UML), the Meta Object Facility (MOF), XML Metadata Interchange (XMI), and the Common Warehouse Metamodel (CWM) to describe this complex evolutionary process.
Current research is going on in finding standardized ways how to describe and perform transformations of models using different intermediate meta models and transformation languages [Seidewitz, 2003]. One recent standard from OMG, the Query/View/Transformation (QVT) [Ignjatovic, 2006; MOFQVT, 2006], addresses this problem in a generic way. There exist already

implementations of these transformation languages and their applications are in preparation [Bohlen, 2006].

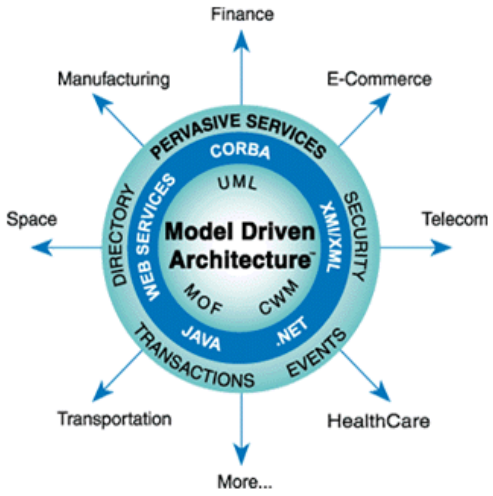


Figure 5.3: MDA overview by OMG from [OMG, 2006]

The model- transformation- process can be realized to some extent in an automated way and results in partial or complete code generation. By combining the model-driven and the generative software development, MDA reaches its aim of increased reusability, portability and longevity to a high degree. During recurring tasks it additionally reduces the error source “human” and increases the productivity of code generation because of the higher level of automation. Refactoring during code maintenance is started on the model level and is performed due to the automated generation process in a consequent way. Finally, by continuously changing the model an up-to-date description of the current system is permanently available and can be seen as part of the documentation for developers maintaining projects. This advantages of MDA have been convincing over the last years in small software projects as well as in more complex projects [Friese and Bohlen, 2006]. Despite the MDA standard, recent literature is using in this context the term Model-driven Software Development (MDSD) which describes a more generic but practice oriented way of building software based on models. Regarding this, MDA can be seen as a a special case of MDSD and is according to [Stahl and Völker, 2006, p. 4] “a standardization initiative of the OMG focusing on MDSD”.

5.1.9 AndroMDA

AndroMDA is a software development environment which combines the model driven programming paradigm with generative software development. The result of this combination is a MDA-based code generation framework which creates artifacts for multiple target platforms. Similar to the MDA approach, software developers start with describing the architectural structure of a software on higher abstraction level using UML. Already existing Computer-Aided Software Engineering (CASE) tools like MagicDraw [MagicDraw, 2007] are used to design the Platform Independent Model (PIM). With the help of AndroMDA-specific UML profiles the collaborating components within the PIM are set into a functional context. By adding UML stereotypes and tagged values a component gets a special functionality which is still independent from the real

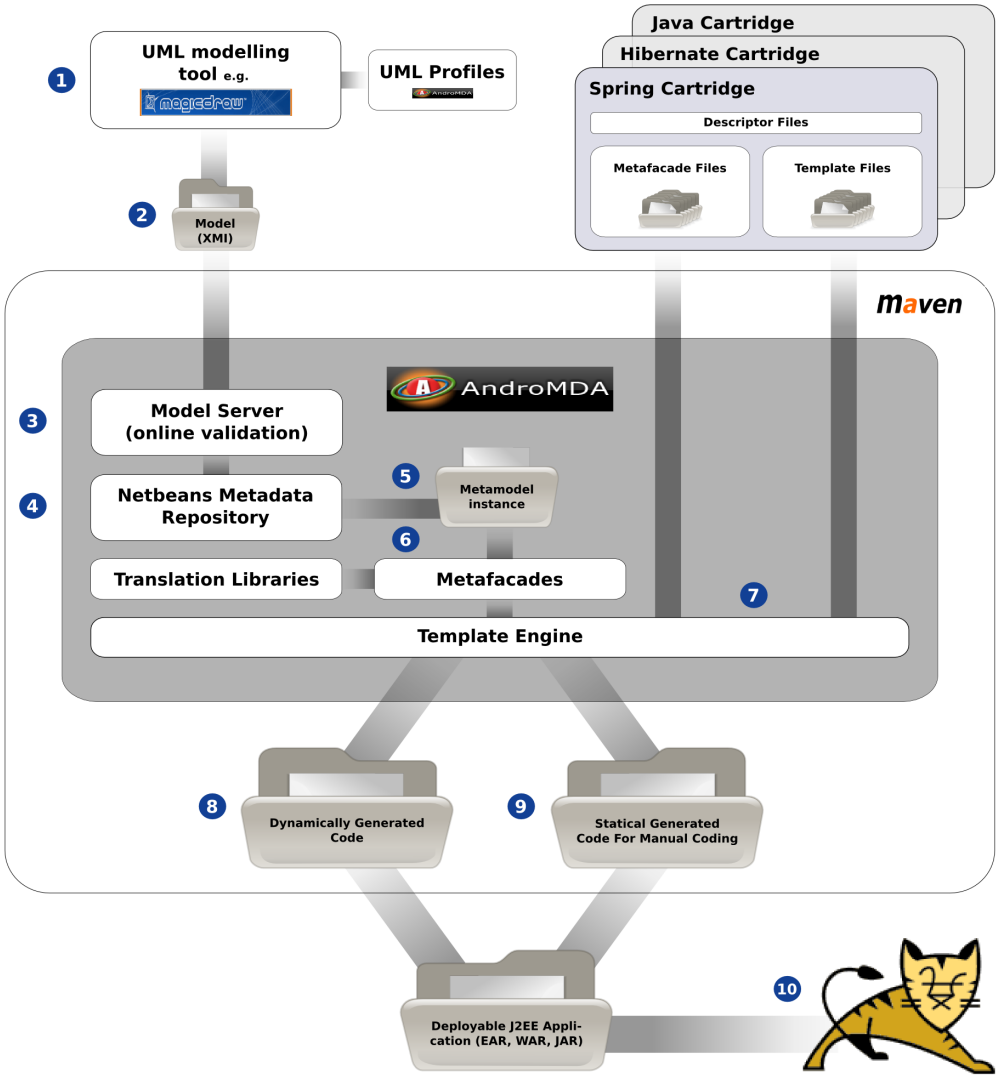


Figure 5.4: AndroMDA build process, derived and extended from [Schulz, 2006, pp. 4]

implementation and platform. If a UML class gets the stereotype $\ll Entity \gg$, the attributes of the class are "somehow" persisted and can be retrieved afterwards. It is not specified which technology, which database or which framework is used. A model annotated in this manner can be automatically translated with AndroMDA into a Code Model skipping the manual creation of a Platform Specific Model proposed by the MDA specifications. Thus the author of AndroMDA describes this kind of enhanced modeling and code generation as the "MDA light" approach [Bohlen and Starke, 2003].

AndroMDA build process Figure 5.4 gives a detailed overview on how the build process is realized in AndroMDA version 3.1. The starting point is the annotated model, which is cre-

ated by software architects, using an UML tool of their choice. For better collaboration within a developer team the model can be divided in logical sub units, called modules, which together form the PIM. The UML tool must support external UML profiles for component annotation and it must be possible to save the model modules in XML Model Interchange (XMI) files. From that moment the build process is taken over by the project- and build-manager Maven 1 which is steering the successive build steps using the core engine classes of AndroMDA. At the entry point the UML model is read and validated to guarantee its formal correctness and completeness. For this purpose a model server can be started which performs an online validation simultaneously to the modeling process. Detailed information about invalid components in the model are reported immediately to the software architects. Internally this server provides the already parsed UML model as Metadata Repository (MDR) [MDR, 2007] to the build process. This repository is compliant to the MOF repository definition from OMG, which is part of the MDA standard.

Out of the MDR repository the AndroMDA core engine creates internally used metamodel instances which are processed by cartridge units. Cartridges are responsible for the platform specific generation of source code and configuration files. They consist of a bundle of Java classes (metafacades) and template files. Out of the metamodel the metafacade classes extract all necessary build information for the artifacts of a specific platform. This includes the evaluation of relational expressions defined with the Object Constraint Language (OCL) on the UML class level. OCL expressions are processed with translation libraries in order to generate, for example platform specific object query code with the Hibernate Query Language or EJB Query Language. Through the metafacades the gathered and translated build information is offered to a template engine which generates the target source- and configuration-files according the build instructions in the code templates. By decoupling the templates from the metamodel with metafacades a clear “separation of concerns” is achieved and the templates get their necessary information without knowing how it is gathered.

Different template engines can be used within AndroMDA but existing cartridges are using mainly the template engine Velocity [Velocity, 2007]. This is because AndroMDA is implemented completely modular and almost all modules within the AndroMDA core engine can be replaced by customized implementations. The build process can be influenced by the developer using a central configuration file that defines for which platform which cartridge must be used. Additionally this configuration file contains platform and framework specific configuration parameters for the cartridges like SQL dialect of the productive RDBMS environment etc. The outcome of this build process is to one part dynamically generated code and to one part statically generated code artifacts. The last named code frames are just generated during their first build process and remain untouched by successive build cycles. The domain specific business

logic, which must be still manually programmed, can be coded within these classes. Subsequent changes in the UML model lead to a complete rebuild of the dynamically generated code artifacts which have the changes incorporated automatically. The manual classes are derived from the continuously generated dynamic classes using the inheritance mechanisms of the specific platform and programming language. Thus the changes in the dynamically generated super classes influence the sub classes and their capabilities through the Java class inheritance. If the changes influence also parts of already existing manual code, the developers have to adapt the code by hand.
During the last years this mixed concept of using MDA-based code generation for repeated code and manual implementations for application specific business code turned out to be a productive and successful approach. Big projects (for example with the German airline Lufthansa) could be realized within reasonable time, producing less costs and cleaner coding style [Friese and Bohlen, 2006].

Predefined implemented patterns With AndroMDA one can generate clean and ready to use architectural structures (patterns) for several use cases. One typical use case is a data centric application like SMILE which has to deal with multiple real world objects that must be stored in a database. Developers can realize the complete persistence layer using modeled data entities. Out of this class definitions marked as $\ll Entity \gg$, AndroMDA generates the complete palette of components which are necessary for a typical persistence layer. Classes which realize the Data Access Object (DAO) pattern [Bien, 2003, pp. 180] [Johnson and Hoeller, 2004, pp. 285] offer a clean access to data in database tables on an object basis. Besides modeled accessors and finder methods defined with the Object Constraint Language (OCL), stored object entities can be converted transparently to generated Value Objects [Bien, 2003, pp. 85]. The Value Object pattern is very frequently used for transfer objects [Fowler, 2003, pp. 401] which are assembled in a DAO from one or multiple database entities and sent to a external view tier. The code generation comprises not only Java code but also configuration files, deployment descriptors and SQL database definition scripts, considering always platform specific features of the target environment. Another feature of the persistence cartridges are persisted enumeration type objects, which are stored in the database and can be used as regular enumeration object during program execution and persistence.
The Service Facade [Johnson, 2002, pp. 373] pattern is also realized within AndroMDA and can be seen as hiding front-end for the business tier. Internal implementation details are not directly accessible from the outside and requests to the business tier must be passed through a service facade object. Transaction management within this service facades, security access restriction but also Web service implementations are generated automatically by adding the appropriate stereotypes like $\ll Service \gg$ to the model components. The platform specific realization

can be almost transparently switched from one target platform to another. Just by changing the target platform the service facade can be either implemented as Stateless Session Beans (SLSB) or as Spring Beans. The persistence layer can be realized either with DAOs using Entity Beans or Hibernate.

Supported Platforms and Technologies AndroMDA has its historical roots in the J2EE environment (UML2EJB) and was initially designed to generate different well proven J2EE patterns with Enterprise Java Beans technology. The range of supported platforms was steadily extended and comprises now cartridges for Enterprise Java Bean implementations, Spring with Hibernate persistence and also Microsofts .Net implementations. The main focus of the project is still on enterprise applications in the J2EE environment. It shows up with solutions for particular enterprise use cases like remote services protected with distributed transactions, Web services realized with the Axis framework, generation of CRUD pages with Web front-end using Model View Controller like Struts, Java Server Faces and business process modeling with jBPM and BPM4Struts. Within the SMILE project AndroMDA is mainly used for the implementation of the persistence tier with Spring and Hibernate and for the implementation of Service facades with ACEGI Security integration.

5.1.10 Software configuration management

The software development process can be massively supported by the use of suitable development tools. The following tools are used for the development of SMILE:

- The Computer-Aided Software Engineering (CASE) tool *MagicDraw* [MagicDraw, 2007] is used for the visual creation of UML models which act as input files for the automatic code generation process of the AndroMDA generation framework. Its commercial version supports the modular separation of UML models into sub packages and therefore permits easier collaboration within a heterogeneous development team. The models created by the modeling software are stored in XMI format. Version 9.6 is the last version supporting the XMI version 1.1 demanded by AndroMDA 3.1. The platform independence of MagicDraw is an additional criteria which was considered by the selection of the UML modeling tool.

- The open source project *Maven 1* is the central software package driving the automated build process of AndroMDA. Outgoing from a centrally configured project object model (POM) Maven runs the model integrity checks, generates the code according the cartridges, compiles and packages the project artifacts.

- Manual coding is performed with the support of the Integrated Development Environment (IDE) Eclipse [Eclipse, 2007] which consists of a flexible core application and its extensions. This core application alone offers basic project management, editing and debugging functionality. It additionally provides a very flexible plugin system for custom extension. The flood of modular plugins unfolds the application core to a powerful development tool integrating development functionality for different programming languages like Java, C/C++ or Perl. J2EE extensions like MyEclipse [MyEclipse, 2007] take care about the integration of most incarnations of J2EE containers, lightweight container frameworks, Web based MVCs etc. into Eclipse IDE. Eclipse plugins improve the development process and productivity to a large extend e.g. by providing syntax highlighting, code validation, code completion, simple creation wizard for complex J2EE framework applications or version control.

- Tomcat [Tomcat, 2007] is a servlet container released under the Apache open source license and can be configured or extended on demand with additional J2EE functionality like central database connection pooling, transaction management etc. Because it is the reference implementation of Suns Java Servlet API several EJB application server integrate Tomcat as their internal Web container. On the other hand Tomcat can be also used as standalone Web container for the realization of multi- tiered Web applications

using J2EE lightweight container like Spring and HiveMind for transaction management-, remoting- or messaging- services. Tomcat is technically mature and its short startup, deployment and shutdown cycles increase the productivity during the development compared to full-featured application servers.

- PostgreSQL [PostgreSQL, 2006] is a wide spread representative of open source relational database systems which supports most of today's operating systems. It turns out to be a reliable Atomicity Consistency Isolation and Durability (ACID) compliant database environment which tries to be ANSI-SQL 92/99 standard conform regarding datatypes and data manipulation functionality. Besides regular datatypes also binary large objects are supported within tables. Some more data definition and retrieval features are: sub queries, different transaction isolation levels, sequences for primary key generation, key constraints and references etc. For more details about its functionality see [Douglas and Douglas, 2005]. The PostgreSQL database comes with native client support for most of the commonly used programming languages like Java, C/C++, .Net, Perl, Python, PHP, Ruby, Tcl. From the administrative view necessary enterprise features for mission critical production environments are also part of the PostgreSQL capabilities e.g. Multi-Version Concurrency Control (MVCC), online backup functionality, asynchronous replication and nested transactions.

- Subversion (SVN) [Subversion, 2007] is a free version control system which can be used for centrally tracking the change of files and directories within a project, over time. Changes on SVN shared projects are stored as numbered revisions by which older status of files and directories can be reconstructed. Its main field of application lies in the management of source code especially if teams of multiple programmers have to collaborate. Subversion offers local and remote repository access using either the own binary protocol or the more common WebDAV protocol [Collins-Sussman et al., 2004]. File locking for locking shared access to files and renaming of files and directories by keeping their history are important characterisitcs of this system.

5.1.11 High performance computing with Rocks Clusters Linux

The methodologies used for studying biological processes changed during the last years from predominantly manual to semi automatic and automatic investigations. Besides holistic approaches like whole genome sequencing, which have to deal with huge amount of sequence data, also the microscopy field experienced a noticeable increase of image data. This can be explained by improved sensitivity of devices but also by automatic acquisition of experiments from three dimensional to five dimensional space [Andrews et al., 2002]. Sophisticated programs to process and analyze hundreds of images consume computational power which can't be provided by a single machine within reasonable time.
This problem can be solved by integrating a High Performance Computing (HPC) infrastructure [Thallinger et al., 2002] into the experiment and analysis workflow. A HPC cluster consists in a PC-based compound of compute nodes which communicate over a high speed interconnection. One PC in the cluster takes over the functionality of the dedicated cluster master, which is the gateway for remote access to the nodes. Administrative installation, configuration and user management but also the development and submission of parallel programs are done on the master system. The programs running on a cluster can be separated into two categories:

- Special custom made *parallel programs* are designed for parallel execution using dedicated HPC communication libraries for interprocess communication like MPICH [Gropp et al., 1996], LAM/MPI [G. et al., 1994] or PVM [Geist et al., 1994]. The program logic itself has to partition the tasks into subtasks by following the paradigms of either domain decomposition or functional decomposition [Thallinger et al., 2002]. The implementation of such programs needs in depth scientific domain experience as well as programming know-how, but returns fast and optimized applications.

- Unmodified regular programs, which can be executed on a single machine, can be run also on a cluster system. The dividing of the computational problem into subtasks happens here during job submission, e.g. the overall processing task of multiple Microscopy images can be run in parallel by submitting one image to each compute node. There the calculation is performed as it would run on a single machine sequentially and the results are collected after all images are processed. This kind of parallel execution is called swarming and is the easiest way of using an HPC cluster. The user must not have an in depth domain-specific knowledge to perform legitimate computing decomposition. The submission of program execution to the nodes but also the load balancing on different nodes is supported by queuing systems which guarantee an optimized usage of HPC cluster.

The main argument for using HPC computing cluster compared to monolithic Symmetric Multi Processing (SMP) machines with the same amount of CPUs is the reasonable prize for hardware. On the other hand the costs for the necessary administrative effort and know-how to manage this kind of infrastructure especially regarding the software installation should not be underestimated.
The San Diego Supercomputing Center has developed a free Linux distribution named “Rocks Clusters” [Papadopoulos et al., 2001; Rocks, 2006], which is dedicated and optimized for large scale HPC cluster systems. Rocks Clusters in the version 4.0 uses the free Linux distribution CentOS [CentOS, 2007] or, if commercial support is necessary, the binary compatible counterpart Redhat Enterprise Server as basis for the operating system. The installation process must be performed manually on the cluster master from CD and for cluster nodes automatically over the network. The Preboot Execution Environment (PXE) offers a comfortable way to boot a basic operating system from remote servers and to start the installation on compute nodes. The Redhat KickStart installer is running completely unattended and without any user interaction whereas necessary packages are gathered via the HTTP protocol from the cluster master. The installation of the cluster master and its nodes results in a fully configured and ready to use cluster system where user- and node specific data is stored in a central MySQL [MySQL, 2006] database. Cluster users are created on the cluster master and the user information is distributed to the nodes by the internally developed password system 411 [Sacerdoti et al., 2005]. Rocks Clusters basic installation can be extended with application specific software bundles, so called rolls [Bruno et al., 2004]. Rocks Clusters comes with HPC rolls for different queuing systems like SGE [SGE, 2006] or Torque [Torque, 2006], parallelization libraries and with more domain specific software for visualization, bioinformatics, grid computing [Foster and Kesselman, 1997] or Java development. From the hardware site Rocks Clusters supports natively x86-PC systems with one or more 32- and 64-bit CPUs from AMD and Intel. Special high speed interconnect networks with low latency for fast interprocess communication like Myrinet [Boden et al., 1995] or Infiniband can be integrated smoothly by just adding rolls from the different hardware manufacturers. No additional configuration for Myrinet nodes is necessary and the programmer can just use different node names in order to route the communication traffic over the Myrinet network. In addition to this Ethernet tunneling over Myrinet special hardware optimized communication libraries for the MPI are available, which don’t use the Ethernet stack for communication but direct Myrinet interconnect functionality. The cluster hardware and its load can be online-monitored by the pre-installed Ganglia [Sacerdoti et al., 2003] system which offers a Web interface for statistical evaluation of the work load. Special customizations for local environment conditions to compute nodes can be integrated by adapting the XML configuration files of KickStart installer and by copying the additionally necessary self made packages for the

Redhat Package Manager to the installation directory. Changes of the configuration are forcing the new installation of all nodes which takes for one node approximately 5 to 15 minutes. Additional services like Web services for accessing the cluster can be easily installed according the Linux installation procedures.

5.2 Basic overview of fluorescence microscopy

During the last years fluorescence microscopy has undergone a remarkable revival in life sciences [Yuste, 2005]. Strong evidence of this trend-setting development can be adduced by searching the central publications database PubMed for the term “fluorescence” and “fluorescence microscopy” [PubMed, 2006]. Since February 2005 more than 27.000 articles containing the term “fluorescence” either in title or in the abstract and more than 6.200 articles with “fluorescence microscopy” were published (search performed in February 2007).

The underlying physical principle, discovered already in 1852 by Sr. G. G. Stokes, is based on the fact that fluorescent molecules can be excited from a lower energy state to a higher energy state by absorbing light of a particular wavelength. Fluorescent molecules are not able to rest in this excited state and transform immediately the absorbed light in vibration energy and emitted light of a longer wavelength. Excitation and emission wavelength depend on the chemical and physical composition of the fluorescent molecules. By filtering out the excitation light and detecting solely the emission light, the region of interest marked with fluorophores is shining in high contrast to the dark background [reviewed by Conchello and Lichtman, 2005]. Drawbacks like bleaching is addressed by developing modified fluorophores with more stable fluorescence behavior. The principle of fluorescence has found broad application in life science, where various biological substances including DNA, RNA and proteins can be detected by labeling with fluorophores.

A major mile stone for fluorescence microscopy was the isolation of the green fluorescent protein GFP from the jellyfish *Aequoria vitoria* and it’s encoding sequence [Chalfie et al., 1994; Tsien, 1998]. The coding sequence of GFP or of a member of the homologous fluorescent protein family [Lippincott-Schwartz and Patterson, 2003] is cloned into gene regions where mRNA for target proteins is transcribed. In this manner proteins translated from the resulting modified RNA are genetically tagged with fluorescent regions and can be localized in fixed cells as well as in living cells. The analysis of protein localization within subcellular compartments can lead immediately to a better understanding of the protein function [reviewed by Andrews et al., 2002].

By recording time series during life cell imaging, even dynamic processes like protein trafficking between subcellular compartments [McNally et al., 2000] and between different cells can be traced. Numerous different techniques have been developed for characterizing different behavior of proteins. E.g. using Fluorescence Recovery After Photobleaching (FRAP) or Fluorescence Loss in Photobleaching (FLIP) the kinetic behavior and mobility of tagged proteins within living cells can be measured [Giepmans et al., 2006]. Fluorescence Energy Transfer (FRET) is used for examining the binding behavior of interacting proteins [Vogel et al., 2006].

Fluorescence microscopy grants a unique global perspective on the specimen under *in vivo* conditions using noninvasive methods.

A second possibility for transferring fluorescent dyes to a cellular region of interest is using immunolabeling techniques in fixed cells [Allan, 2000]. It is necessary to fix and permeabilize cell membranes in order to create entrance points for antibodies targeted against proteins of interest. For the detection of a specific protein, a two-step method is commonly used. A primary antibody detects specifically the protein of interest and contains an additional binding site for a secondary antibody. This second antibody is binding to the primary antibody and carries the detectable fluorophore region to the target location. This procedure can be applied for localizing multiple different proteins at the same time in the same cells. To achieve this it is necessary to use different fluorescent dyes with non overlapping excitation/emission spectra to avoid cross talk excitation between the different fluorophores.

Futher on, in order to increase the efficiency of immunofluorescence methods quantum dots, which are inorganic nanocrystals with good fluorescence behaviour, have been attached to antibodies. This results in a sharp and discrete wavelength of the fluorescence light by higher extinction coefficient and good quantum yield [Giepmans et al., 2006].

The emitted fluorescence light can be collected with different types of microscopes and digital detection systems [Inoué and Spring, 1997, chapter 6] which produce images and store them in files. Existing technologies like video microscopy [Evanko, 2005; Inoué and Spring, 1997] were initially applied and optimized but also novel techniques have evolved. Some highlights of the arising microscopy disciplines with different technological focus are: the confocal microscopy [Conchello and Lichtman, 2005], the two-photon microscopy [Denk et al., 1990; Helmchen and Denk, 2005], computational approaches like three-dimensional (3D) deconvolution microscopy [McNally et al., 1999] and 5D quantitative fluorescence microscopy [Andrews et al., 2002].

Recent developments in fluorescence microscopy are going into the direction of holistic approaches where Proteomics initiatives try to systematically combine localization information, gathered by large scale localization studies, with typical Proteomics datasets [Davis, 2004; O'Rourke et al., 2005]. This is only possible if modern methods of high-throughput microscopy [Glory and Murphy, 2007] are applied for automatically generating and analyzing resulting microscopy images.

Appendix A

Bibliography

Article references

[Altschul et al., 1990] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J Mol Biol*, 215:403–410, 1990.

[Andrews et al., 2002] P. D. Andrews, I. S. Harper, and J. R. Swedlow. To 5D and beyond: quantitative fluorescence microscopy in the postgenomic era. *Traffic*, 3:29–36, 2002.

[Apweiler et al., 2004] R. Apweiler, A. Bairoch, C. H. Wu, W. C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, M. J. Martin, D. A. Natale, O Donovan, C. N. Redaschi, and L. L. Yeh. UniProt: the Universal Protein knowledgebase. *Nucleic Acids Res*, 32:D115–D119, 2004.

[Birney et al., 2004] E. Birney, M. Clamp, and R. Durbin. GeneWise and Genomewise. *Genome Res*, 14:988–995, 2004.

[Boden et al., 1995] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W.-K. Su. Myrinet: A Gigabit-per-Second Local Area Network. *IEEE Micro*, 15(1):29–36, 1995. ISSN 0272-1732. doi: http://doi.ieeecomputersociety.org/10.1109/40.342015.

[Bohlen, 2006] M. Bohlen. QVT und MultiI-Metamodell-Transformationen in MDA. *OBJECTspektrum*, 02:51–57, 2006.

[Bohlen and Starke, 2003] M. Bohlen and G. Starke. MDA entzaubert. *OBJECTspektrum*, 03:52–57, 2003.

[Bonetto et al., 2004] P. Bonetto, P. Boccacci, M. Scarito, M. Davolio, M. Epifani, G. Vicidomini, C. Tacchetti, P. Ramoino, C. Usai, and A. Diaspro. Three-dimensional microscopy

migrates to the web with "PowerUp Your Microscope". *Microsc Res Tech*, 64:196–203, 2004.

[Bruno et al., 2004] G. Bruno, M. J. Katz, F. D. Sacerdoti, and P. M. Papadopoulos. Rolls: Modifying a Standard System Installer to Support User-Customizable Cluster Frontend Appliances. In *IEEE International Conference on Cluster Computing, Los Angeles USA*, 2004.

[Carazo et al., 1999] J. M. Carazo, E. H. Stelzer, A. Engel, I. Fita, C. Henn, J. Machtynger, P. McNeil, D. M. Shotton, M. Chagoyen, P. A. de Alarcon, R. Fritsch, J. B. Heymann, S. Kalko, J. J. Pittet, P. Rodriguez-Tome, and T. Boudier. Organising multi-dimensional biological image information: the BioImage Database. *Nucleic Acids Res*, 27:280–283, 1999.

[Chalfie et al., 1994] M. Chalfie, Y. Tu, G. Euskirchen, W. W. Ward, and D. C. Prasher. Green fluorescent protein as a marker for gene expression. *Science*, 263:802–805, 1994.

[Conchello and Lichtman, 2005] J. A. Conchello and J. W. Lichtman. Optical sectioning microscopy. *Nat Methods*, 2:920–931, 2005.

[Dai et al., 2003] W. Dai, Y. Liang, and Z. H. Zhou. Web portal to an image database for high-resolution three-dimensional reconstruction. *J Struct Biol*, 144:238–245, 2003.

[Davis, 2004] T. N. Davis. Protein localization in proteomics. *Curr Opin Chem Biol*, 8:49–53, 2004.

[Denk et al., 1990] W. Denk, J. H. Strickler, and W. W. Webb. Two-photon laser scanning fluorescence microscopy. *Science*, 248:73–76, 1990.

[Deutsch et al., 2006] E. W. Deutsch et al. Development of the Minimum Information Specification for In Situ Hybridization and Immunohistochemistry Experiments (MISFISHIE). *OMICS*, 10:205–208, 2006.

[Eisenstein, 2006] M. Eisenstein. Something to see. *Nature*, 443:1017–1021, 10 2006. Technology Feature.

[Evanko, 2005] D. Evanko. Focus on fluorescence imaging, principle and practice of microscope techniques. *Nat Methods*, 2(12):901, 2005.

[Foster and Kesselman, 1997] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, Summer 1997. URL `citeseer.ist.psu.edu/foster96globus.html`.

[Friese and Bohlen, 2006] P. Friese and M. Bohlen. Erfolgreicher Einsatz von AndroMDA bei Lufthansa Systems. *OBJECTspektrum*, 03:62–68, 2006.

[G. et al., 1994] Burns G., R. Daoud, and J. Vaigl. LAM: An Open Cluster Environment for MPI. *Proceedings of Supercomputing Symposium*, 1:379–386, 1994. URL `http://www.lam-mpi.org/download/files/lam-papers.tar.gz`.

[Giepmans et al., 2006] B. N. Giepmans, S. R. Adams, M. H. Ellisman, and R. Y. Tsien. The fluorescent toolbox for assessing protein location and function. *Science*, 312:217–224, 2006.

[Glory and Murphy, 2007] E. Glory and R. F. Murphy. Automated subcellular location determination and high-throughput microscopy. *Dev Cell*, 12:7–16, 2007.

[Goldberg et al., 2005] I. G. Goldberg, C. Allan, J. M. Burel, D. Creager, A. Falconi, H. Hochheiser, J. Johnston, J. Mellen, P. K. Sorger, and J. R. Swedlow. The Open Microscopy Environment (OME) Data Model and XML file: open tools for informatics and quantitative analysis in biological imaging. *Genome Biol*, 6:R47–R47, 2005.

[Gonzalez-Couto et al., 2001] E. Gonzalez-Couto, B. Hayes, and A. Danckaert. The life sciences Global Image Database (GID). *Nucleic Acids Res*, 29:336–339, 2001.

[Gropp et al., 1996] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Computing*, 22(6):789–828, 09 1996. URL `http://www.sciencedirect.com/science/article/B6V12-3WP2DJ1-1/2/a7fcb79b332859e16f4da400c5b79cf7`.

[Habeler et al., 2002] G. Habeler, K. Natter, G. G. Thallinger, M. E. Crawford, S. D. Kohlwein, and Z. Trajanoski. YPL.db: the Yeast Protein Localization database. *Nucleic Acids Res*, 30:80–83, 2002.

[Hackl et al., 2004] H. Hackl, M. Maurer, B. Mlecnik, J. Hartler, G. Stocker, D. Miranda-Saavedra, and Z. Trajanoski. GOLDdb: Genomics of lipid-associated disorders database. *BMC Genomics*, 5:93–93, 2004.

[Helmchen and Denk, 2005] F. Helmchen and W. Denk. Deep tissue two-photon microscopy. *Nat Methods*, 2:932–940, 2005.

[Ignjatovic, 2006] M. Ignjatovic. ”Query/View/T”ransformation: Ein neuer OMG-standard. *OBJECTspektrum*, 02:46–50, 2006.

[International Human Genome Sequencing Consortium, 2004] International Human Genome Sequencing Consortium. Finishing the euchromatic sequence of the human genome. *Nature*, 431:931–945, 2004.

[Jakobovits et al., 2000] R. Jakobovits, S. G. Soderland, R. K. Taira, and J. F. Brinkley. Requirements of a Web-based experiment management system. *Proc AMIA Symp*, 1:374–378, 2000.

[Kals et al., 2005] M. Kals, K. Natter, G. G. Thallinger, Z. Trajanoski, and S. D. Kohlwein. YPLdb(2): the yeast protein localization database, version 20. *Yeast*, 22:213–218, 2005.

[Kent, 2002] W. J. Kent. BLAT–the BLAST-like alignment tool. *Genome Res*, 12:656–664, 2002.

[King and M., 2000] R. D. King and Ouali M. Cascaded Mutiple Classifers for Secondary Structure Prediction. *Prot. Sci*, 9:1162–1176, 2000.

[Kumar et al., 2002] A. Kumar, S. Agarwal, J. A. Heyman, S. Matson, M. Heidtman, S. Piccirillo, L. Umansky, A. Drawid, R. Jansen, Y. Liu, K. H. Cheung, P. Miller, M. Gerstein, G. S. Roeder, and M. Snyder. Subcellular localization of the yeast proteome. *Genes Dev*, 16: 707–719, 2002.

[Li et al., 2003] X. J. Li, H. Zhang, J. A. Ranish, and R. Aebersold. Automated statistical analysis of protein abundance ratios from data generated by stable-isotope dilution and tandem mass spectrometry. *Anal Chem*, 75:6648–6657, 2003.

[Liang et al., 2002] Y. Liang, E. Y. Ke, and Z. H. Zhou. IMIRS: a high-resolution 3D reconstruction package integrated with a relational image database. *J Struct Biol*, 137:292–304, 2002.

[Lindek et al., 1999] S. Lindek, R. Fritsch, J. Machtynger, P. A. de Alarcon, and M. Chagoyen. Design and realization of an on-line database for multidimensional microscopic images of biological specimens. *J Struct Biol*, 125:103–111, 1999.

[Lippincott-Schwartz and Patterson, 2003] J. Lippincott-Schwartz and G. H. Patterson. Development and use of fluorescent protein markers in living cells. *Science*, 300:87–91, 2003.

[Maurer et al., 2005] M. Maurer, R. Molidor, A. Sturn, J. Hartler, H. Hackl, G. Stocker, A. Prokesch, M. Scheideler, and Z. Trajanoski. MARS: microarray analysis, retrieval, and storage system. *BMC Bioinformatics*, 6:101–101, 2005.

[McNally et al., 1999] J. G. McNally, T. Karpova, J. Cooper, and J. A. Conchello. Three-dimensional imaging by deconvolution microscopy. *Methods*, 19:373–385, 1999.

[McNally et al., 2000] J. G. McNally, W. G. Muller, D. Walker, R. Wolford, and G. L. Hager. The glucocorticoid receptor: rapid exchange with regulatory sites in living cells. *Science*, 287: 1262–1265, 2000.

[Mellor et al., 2003] S. J. Mellor, A. N. Clark, and T. Futagami. Guest Editors' Introduction: Model-Driven Development. *IEEE Software*, 20(5):14–18, 2003. ISSN 0740-7459. doi: http://doi.ieeecomputersociety.org/10.1109/MS.2003.1231145.

[O'Rourke et al., 2005] N. A. O'Rourke, T. Meyer, and G. Chandy. Protein localization studies in the age of 'Omics'. *Curr Opin Chem Biol*, 9:82–87, 2005.

[Papadopoulos et al., 2001] P. M. Papadopoulos, M. J. Katz, and G. Bruno. NPACI Rocks: Tools and Techniques for Easily Deploying Manageable Linux Clusters. In *Cluster 2001: IEEE International Conference on Cluster Computing. Los Angeles USA*, 10 2001.

[Rainer et al., 2006] J. Rainer, F. Sanchez-Cabo, G. Stocker, A. Sturn, and Z. Trajanoski. CARMAweb: comprehensive R- and bioconductor-based web service for microarray data analysis. *Nucleic Acids Res*, 34:W498–W503, 2006.

[Rodriguez et al., 2004] A. M. Rodriguez, C. Elabd, F. Delteil, J. Astier, C. Vernochet, P. Saint-Marc, J. Guesnet, A. Guezennec, E. Z. Amri, C Dani, and G Ailhaud. Adipocyte differentiation of multipotent cells established from human adipose tissue. *Biochem Biophys Res Commun*, 315:255–263, 2004.

[Sacerdoti et al., 2003] F. D. Sacerdoti, M. J. Katz, and P. M. Papadopoulos. Wide Area Cluster Monitoring with Ganglia. In *IEEE International Conference on Cluster Computing, Hong Kong*, 12 2003.

[Sacerdoti et al., 2005] F. D. Sacerdoti, M. J. Katz, and P. M. Papadopoulos. 411 on Scalable Password Service. In *IEEE High Performance Distributed Computing Conference, North Carolina*, 06 2005.

[Seidewitz, 2003] E. Seidewitz. What Models Mean. *IEEE Software*, 20:26–32, 2003.

[Sendall and Kozaczynski, 2003] S. Sendall and W. Kozaczynski. Model transformation: the heart and soul of model-driven software development. *IEEE Software*, 20:42–45, 2003.

[Smit, 1993] A. F. Smit. Identification of a new, abundant superfamily of mammalian LTR-transposons. *Nucleic Acids Res*, 21:1863–72, 1993.

[Sprague et al., 2004] B. L. Sprague, R. L. Pego, D. A. Stavreva, and J. G. McNally. Analysis of binding reactions by fluorescence recovery after photobleaching. *Biophys J*, 86:3473–3495, 2004.

[Springmann and Tiggers, 2005] L. Springmann and O. Tiggers. Das Webframework Tapestry (The webframework Tapestry). *Java spektrum*, 1(1):18–25, 03 2005.

[Sturn et al., 2002] A. Sturn, J. Quackenbush, and Z. Trajanoski. Genesis: cluster analysis of microarray data. *Bioinformatics*, 18:207–208, 2002.

[Swedlow et al., 2003] J. R. Swedlow, I. Goldberg, E. Brauner, and P. K. Sorger. Informatics and quantitative analysis in biological imaging. *Science*, 300:100–102, 2003.

[Swedlow et al., 2006] J. R. Swedlow, S. E. Lewis, and I. G. Goldberg. Modelling data across labs, genomes, space and time. *Nat Cell Biol*, 8:1190–1194, 2006.

[Thallinger et al., 2002] G. G. Thallinger, S. Trajanoski, G. Stocker, and Z. Trajanoski. Information management systems for pharmacogenomics. *Pharmacogenomics*, 3:651–667, 2002.

[Thompson et al., 1994] J. D. Thompson, D. G. Higgins, and T. J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res*, 22: 4673–4680, 1994.

[Tsien, 1998] R. Y. Tsien. The green fluorescent protein. *Annu Rev Biochem*, 67:509–544, 1998.

[Venter et al., 2001] J. C. Venter, M. D. Adams, E. W. Myers, P. W. Li, R. J. Mural, G. G. Sutton, H. O. Smith, M. Yandell, C. A. Evans, R. A. Holt, J. D. Gocayne, P. Amanatides, R. M. Ballew, D. H. Huson, J. R. Wortman, Q. Zhang, C. D. Kodira, X. H. Zheng, L. Chen, M. Skupski, G. Subramanian, P. D. Thomas, J. Zhang, G. L. Gabor, Miklos, C. Nelson, S. Broder, A. G. Clark, J. Nadeau, V. A. McKusick, N. Zinder, A. J. Levine, R. J. Roberts, M. Simon, C. Slayman, M. Hunkapiller, R. Bolanos, A. Delcher, I. Dew, D. Fasulo, M. Flanigan, L. Florea, A. Halpern, S. Hannenhalli, S. Kravitz, S. Levy, C. Mobarry, K. Reinert, K. Remington, J. Abu-Threideh, E. Beasley, K. Biddick, V. Bonazzi, R. Brandon, M. Cargill, I. Chandramouliswaran, R. Charlab, K. Chaturvedi, Z. Deng, F. Di, V., P. Dunn, K. Eilbeck, C. Evangelista, A. E. Gabrielian, W. Gan, W. Ge, F. Gong, Z. Gu, P. Guan, T. J. Heiman, M. E. Higgins, R. R. Ji, Z. Ke, K. A. Ketchum, Z. Lai, Y. Lei, Z. Li, J. Li, Y. Liang, X. Lin, F. Lu, G. V. Merkulov, N. Milshina, H. M. Moore, A. K. Naik, V. A. Narayan, B. Neelam, D. Nusskern, D. B. Rusch, S. Salzberg, W. Shao, B. Shue, J. Sun, Z. Wang, A. Wang, X. Wang, J. Wang, M. Wei, R. Wides, C. Xiao, C. Yan, A. Yao, J. Ye, M. Zhan, W. Zhang, H. Zhang, Q. Zhao, L. Zheng, F. Zhong, W. Zhong, S. Zhu, S. Zhao, D. Gilbert, S. Baumhueter, G. Spier, C. Carter, A. Cravchik, T. Woodage, F. Ali, H. An,

A. Awe, D. Baldwin, H. Baden, M. Barnstead, I. Barrow, K. Beeson, D. Busam, A. Carver, A. Center, M. L. Cheng, L. Curry, S. Danaher, L. Davenport, R. Desilets, S. Dietz, K. Dodson, L. Doup, S. Ferriera, N. Garg, A. Gluecksmann, B. Hart, J. Haynes, C. Haynes, C. Heiner, S. Hladun, D. Hostin, J. Houck, T. Howland, C. Ibegwam, J. Johnson, F. Kalush, L. Kline, S. Koduru, A. Love, F. Mann, D. May, S. McCawley, T. McIntosh, I. McMullen, M. Moy, L. Moy, B. Murphy, K. Nelson, C. Pfannkoch, E. Pratts, V. Puri, H. Qureshi, M. Reardon, R. Rodriguez, Y. H. Rogers, D. Romblad, B. Ruhfel, R. Scott, C. Sitter, M. Smallwood, E. Stewart, R. Strong, E. Suh, R. Thomas, N. N. Tint, S. Tse, C. Vech, G. Wang, J. Wetter, S. Williams, M. Williams, S. Windsor, E. Winn-Deen, K. Wolfe, J. Zaveri, K. Zaveri, J. F. Abril, R. Guigo, M. J. Campbell, K. V. Sjolander, B. Karlak, A. Kejariwal, H. Mi, B. Lazareva, T. Hatton, A. Narechania, K. Diemer, A. Muruganujan, N. Guo, S. Sato, V. Bafna, S. Istrail, R. Lippert, R. Schwartz, B. Walenz, S. Yooseph, D. Allen, A. Basu, J. Baxendale, L. Blick, M. Caminha, J. Carnes-Stine, P. Caulk, Y. H. Chiang, M. Coyne, C. Dahlke, A. Mays, M. Dombroski, M. Donnelly, D. Ely, S. Esparham, C. Fosler, H. Gire, S. Glanowski, K. Glasser, A. Glodek, M. Gorokhov, K. Graham, B. Gropman, M. Harris, J. Heil, S. Henderson, J. Hoover, D. Jennings, C. Jordan, J. Jordan, J. Kasha, L. Kagan, C. Kraft, A. Levitsky, M. Lewis, X. Liu, J. Lopez, D. Ma, W. Majoros, J. McDaniel, S. Murphy, M. Newman, T. Nguyen, N. Nguyen, and M. Nodell. The sequence of the human genome. *Science*, 291:1304–1351, 2001.

[Vogel et al., 2006] S. S. Vogel, C. Thaler, and S. V. Koushik. Fanciful FRET. *Sci STKE*, 2006: re2–re2, 2006.

[Wengatz, 2005] N. Wengatz. Spring und EJB, Das Beste aus zwei Welten (Spring and EJB, the best of two worlds). *Java spektrum*, 1(1):26–32, 03 2005.

[Yuste, 2005] R. Yuste. Fluorescence microscopy today. *Nat Methods*, 2:902–904, 2005.

Book references and manuals

[Allan, 2000] V. J. Allan. *Protein Localization by Fluorescence Microscopy*. Oxford Univeristy Press Inc., 2000. ISBN 0-19-963740-7.

[Armstrong et al., 2005] E. Armstrong, J. Ball, S. Bodoff, D. B. Carson, I. Evans, D. Green, K. Haase, and E. Jendrock. *The J2EE™1.4 Tutorial*. Sun Microsystems, 8.2 edition, 12 2005. URL `http://java.sun.com/j2ee/1.4/docs/tutorial/doc/J2EETutorial.pdf`.

[Ball et al., 2006] J. Ball, D. Carson, I. Evans, S. Fordin, K. Haase, and E. Jendrock. *The Java™EE 5 Tutorial*. Sun Microsystems, 9 edition, 06 2006. URL `http://java.sun.com/javaee/5/docs/tutorial/doc/JavaEETutorial.pdf`.

[Bauer and King, 2004] C. Bauer and G. King. *Hibernate in Action*. Manning Publications, 2004. ISBN 193239415X.

[Bien, 2003] A. Bien. *J2EE Patterns, Entwurfsmuster für J2EE*. Addison-Wesley, 2003. ISBN 3827321247.

[Codd, 1990] E. F. Codd. *The relational model for database management: version 2*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990. ISBN 0-201-14192-2.

[Collins-Sussman et al., 2004] B. Collins-Sussman, B. W. Fitzpatrick, and C. M. Pilato. *Version Control with Subversion*. O'Reilly, 2004. ISBN 0-596-00448-6. URL `http://svnbook.red-bean.com/`. Fulltext of the newest version is online available.

[Douglas and Douglas, 2005] K. Douglas and S. Douglas. *PostgreSQL (2nd Edition)*. Sams Publishing, 2nd edition, 2005. ISBN 0672327562.

[Fowler, 2003] M. Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2003. ISBN 0321127420.

[Geist et al., 1994] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. *PVM Parallel Virtual Machine, A User's Guide and Tutorial for Networked Parallel Computing*, volume 3. MIT Press, Cambridge, Mass., 1994. URL `citeseer.ist.psu.edu/article/geist94pvm.html`.

[Harrop and Machacek, 2005] R. Harrop and J. Machacek. *Pro Spring*. Apress, 2005. ISBN 1-59059-461-4.

[Inoué and Spring, 1997] S. Inoué and K. R. Spring. *Video microscopy : the fundamentals 2nd ed.* Plenum Press, New York, 1997. ISBN 0-306-455531-5.

[Johnson, 2002] R. Johnson. *Expert One-on-One J2EE Design and Development*. Wrox Press Ltd, 2002. ISBN 1-86100-784-1.

[Johnson and Hoeller, 2004] R. Johnson and J. Hoeller. *Expert One-on-One J2EE Development without EJB*. WROX, Wiley Publishing, Inc., 2004. ISBN 0-7645-5831-5.

[Johnson et al., 2005] R. Johnson, J. Hoeller, A. Andersen, T. Risberg, and C. Sampaleanu. *Professional Java Development wit the Spring Framework*. WROX, Wiley Publishing, Inc., 2005. ISBN 0-7645-7483-3.

[Johnson et al., 2006] R. Johnson, J. Hoeller, A. Arendsen, C. Sampaleanu, R. Harrop, T. Risberg, D. Davison, D. Kopylenko, M. Pollack, T. Templier, and Vervaet E.

Spring java/j2ee Application Framework Reference Documentation. Interface21, 2006. http://static.springframework.org/spring/docs/1.2.x/spring-reference.pdf.

[Kuhlmann and Müllmerstadt, 1999] G. Kuhlmann and F. Müllmerstadt. *SQL Der Schlüssel zu relationalen Datenbanken*. Rowohlt Taschenbuch Verlag, 1999. ISBN 3-499-60063-3.

[Lewis Ship, 2004] H. M. Lewis Ship. *Tapestry in Action*. Manning Publications Co., 2004. ISBN 1932394117.

[Miller and Mukerji, 2003] J. Miller and J. Mukerji. *MDA Guide Version 1.0.1*. OMG, Object Management Group, 1.0.1 edition, 06 2003. URL `http://www.omg.org/cgi-bin/doc?omg/03-06-01`.

[MOFQVT, 2006] MOFQVT. *MOF QVT Final Adopted Specification*. Object Management Group (OMG), 2006. URL `http://www.omg.org/docs/ptc/05-11-01.pdf`. Website last visited 30/10/2006.

[Monson-Haefel, 2004] R. Monson-Haefel. *J2EE ™Web Services*. Addison Wesley, 2004. ISBN 0-321-14618-2.

[Onstine, 2006] W. Onstine. *Tapestry 101*. SourceBeat, LLC., 12 2006. ISBN 0-9748843-8-3.

[Snell et al., 2002] J. Snell, D. Tidwell, and Kulcheko P. *Webservice-Programmierung mit SOAP*. O'Reilly, 2002. ISBN 3-89721-159-9.

[Sriganesh et al., 2006] R. P. Sriganesh, G. Brose, and M. Silverman. *Mastering Enterprise JavaBeans 3.0*. Wiley Publishing Incorporation, 2006. ISBN 0-471-78541-5. URL `http://www.theserverside.com/tt/books/wiley/masteringEJB3`.

[Stahl and Völker, 2006] T. Stahl and M. Völker. *Model-Driven Software Development*. Jon Wiley & Sons, Ltd., 2006. ISBN 0-470-02570-0.

[Stark, 2005] T. Stark. *J2EE, Einstieg für Anspruchsvolle*. Addison-Wesley in cooperation with Pearson Studium, 2005. ISBN 3-8273-2184-0.

[Tong, 2005] K. I. Tong. *Enjoying Web Development with Tapestry*. Lulu Press, 2005. ISBN 1411649133.

Unpublished references

[Fischer, 2006] M. Fischer. Digital Labbook. Master's thesis, University of Applied Sciences Hagenberg, 2006.

[Fowler, 2004] M. Fowler. Inversion of Control Containers and the Dependency Injection pattern. URL http://www.martinfowler.com/articles/injection.html. Article series, 01 2004.

[Hartler et al., 2006] J. Hartler, G. G. Thallinger, G. Stocker, A. Sturn, T. R. Burkard, E. Korner, A. Scheucher, R. Rader, A. Schmidt, K. Mechtler, and Z. Trajanoski. MASPECTRAS: a platform for management and analysis of proteomic LC-MS/MS data. submitted to Genome Biology, 09 2006.

[Rieder, 2005] D. Rieder. ChromoMapper, a tool for sequence information mapping onto chromosomes. URL http://mcluster.tu-graz.ac.at/clustercontrol/modules/ChromoMapper. Webbased, 2005.

[Schulz, 2006] D. Schulz. MDA-Frameworks: AndroMDA. URL http://www.wi.uni-muenster.de/pi/lehre/ws0506/seminar/02_andromda.pdf. term work from "Ausgewählte Kapitel des Software Engineering", 01 2006.

[Smit et al., 2007] A. F. A. Smit, R. Hubley, and P Green. RepeatMasker. URL http://repeatmasker.org. A program that screens DNA sequences for interspersed repeats and low complexity DNA sequences, 2007.

[Sturn, 2005] A. Sturn. *Comparative Analysis of Human and Mouse Transcriptomes*. PhD thesis, Graz University of Technology, 2005.

Web link references

[Acegi, 2006] ACEGI Security Framework. Last visited on 08/12/2006, 2006. URL http://www.acegisecurity.org.

[ActiveMQ, 2007] ActiveMQ JMSProvider. Last visited on 05/01/2007, 2007. URL http://www.activemq.org.

[Axis, 2007] Axis Webservice Framework. Last visited on 11/01/2007, 2007. URL http://ws.apache.org/axis.

[Bea, 2006] Bea Application Server. Last visited on 08/12/2006, 2006. URL http://www.bea.com.

[CentOS, 2007] ContOS, Linux distribution. Last visited on 21/01/2007, 2007. URL http://www.centos.org/.

[Eclipse, 2007] Eclipse Integrated Java Development Environment. Last visited on 17/01/2007, 2007. URL http://www.eclipse.org/.

[EJB, 2006] Enterprise Java Technology Specification. Last visited on 08/12/2006, 2006. URL http://java.sun.com/products/ejb/docs.html.

[Glassfish, 2006] Glassfish. Last visited on 08/12/2006, 2006. URL http://glassfish.dev.java.net/.

[Hibernate, 2006] Hibernate Persistence Framework. Last visited on 08/12/2006, 2006. URL http://www.hibernate.org/.

[HiveMind, 2006] HiveMind application framework. Last visited on 08/12/2006, 2006. URL http://hivemind.apache.org.

[iBATIS, 2006] iBATIS Data Mapper framework. Last visited on 08/12/2006, 2006. URL http://ibatis.apache.org/.

[ImageJ, 2007] ImageJ, a flexible image processing Java software devloped at the National Institute of Health, Bethesda. Last visited on 02/03/2007, 2007. URL http://rsb.info.nih.gov/ij/.

[J2EE, 2006] JSR-000244 JavaTM Platform, Enterprise Edition 5 Specification. Last visited on 08/12/2006, 2006. URL http://jcp.org/aboutJava/communityprocess/pfd/jsr244/.

[JBossAS, 2006] Jboss Application Server. Last visited on 08/12/2006, 2006. URL http://www.jboss.org.

[Jencks, 2007] Jencks JCA Container. Last visited on 05/01/2007, 2007. URL http://www.jencks.org/.

[JOnAS, 2006] JOnAS Application Server. Last visited on 08/12/2006, 2006. URL http://jonas.objectweb.org/.

[JSCookMenu, 2007] JSCookMenu for Tapestry. Last visited on 17/01/2007, 2007. URL http://tomcat.apache.org/.

[JSR 170: Content Repository for JavaTM technology API, 2004] JSR 170: Content Repository for JavaTM technology API. Last visited on 31/01/2007, 2004. URL http://jcp.org/en/jsr/detail?id=170.

[MagicDraw, 2007] MagicDraw, UML modelling and CASE tool from. Last visited on 12/01/2007, 2007. URL http://www.magicdraw.com.

[Matlab, 2007] Multipurpose Mathematic Software from "The MathWorks, Inc.". Last visited on 31/01/2007, 2007. URL http://www.mathworks.com.

[MDR, 2007] Netbeans MetaData Repository. Last visited on 28/01/2007, 2007. URL http://mdr.netbeans.org/.

[MSDB, 2006] MSDB Protein Database. Last visited on 08/12/2006, 2006. URL http://csc-fserve.hh.med.ic.ac.uk/msdb.html.

[MSDOTNET, 2006] Microsoft .net. Last visited on 08/12/2006, 2006. URL http://www.microsoft.com/net.

[MSSQL, 2006] Microsoft SQL server. Last visited on 06/01/2007, 2006. URL http://www.microsoft.com/sql.

[MyEclipse, 2007] MyEclipse, J2EE plugin for Eclipse. Last visited on 17/01/2007, 2007. URL http://www.myeclipseide.com/.

[MySQL, 2006] MySQL RDBMS server. Last visited on 06/01/2007, 2006. URL http://www.mysql.com/.

[OMG, 2006] Object Management Group. Last visited on 08/12/2006, 2006. URL http://www.omg.org/mda/.

[Oracle, 2006] Oracle Relational database management system. Last visited on 06/01/2007, 2006. URL http://www.oracle.com.

[OracleAS, 2006] Oracle Application Server. Last visited on 08/12/2006, 2006. URL http://www.oracle.com/appserver.

[PostgreSQL, 2006] PostgreSQL RDBMS server. Last visited on 06/01/2007, 2006. URL http://www.postgresql.org/.

[PubMed, 2006] Central database for citations from MEDLINE and other life science journals for biomedical articles. Last visited on 11/02/2007, 2006. URL http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=PubMed.

[Rocks, 2006] Rocks Clusters, a Linux distribution for cluster installations. Last visited on 08/12/2006, 2006. URL http://www.rocksclusters.org.

[Samba, 2007] server software for file and print services to SMB/CIFS clients. Last visited on 31/01/2007, 2007. URL http://www.samba.org.

[SGE, 2006] Sun Grid Engine. Last visited on 01.12.2006, 2006. URL http://gridengine.sunsource.net.

[SOAP, 2003] W3C Specification for SOAP Version 1.2. Last visited on 12/01/2007, 2003. URL http://www.w3.org/TR/soap12.

[Spring, 2006] Spring J2EE Framework. Last visited on 08/12/2006, 2006. URL http://www.springframework.org/.

[Struts, 2006] Apache Struts Web framework. Last visited on 08/12/2006, 2006. URL http://struts.apache.org/.

[Subversion, 2007] Subversion Code Version Management System. Last visited on 17/01/2007, 2007. URL http://subversion.tigris.org/.

[Tacos, 2007] Tacos, AJAX component library for Tapestry. Last visited on 17/01/2007, 2007. URL http://tomcat.apache.org/.

[Tapestry, 2006] Apache Tapesty Framework. Last visited on 08/12/2006, 2006. URL http://tapestry.apache.org.

[Tomcat, 2007] Tomcat servlet container. Last visited on 15/01/2007, 2007. URL http://tomcat.apache.org/.

[Torque, 2006] TORQUE Resource Manager. Last visited on 08/12/2006, 2006. URL http://www.clusterresources.com/pages/products/torque-resource-manager.php.

[Velocity, 2007] Velocity Template Engine. Last visited on 28/01/2007, 2007. URL http://velocity.apache.org/.

[Websphere, 2006] Websphere Application Server. Last visited on 08/12/2006, 2006. URL http://www.ibm.com/software/websphere/.

List of Figures

Appendix B

Glossary

AAS	Authentication and Authorization System
ACEGI	Acegi Security framework (1st,3rd,5th,7th,9th character in the alphabet)
ACIS	Atomicity, Consistency, Isolation and Durability
ACL	Access Control Lists
AOP	Aspect Oriented Programming
API	Application Programming Interface
Axis	Apache eXtensible Interaction System
CAS	Common Authentication Service
CASE	Computer-Aided Software Engineering
CIMP	Computational Independent Model
CRUD	Creating, Reading, Deleting and Updating
CWP	Current Working Protocol
DAO	Data Access Object
DC	Dependency Injection
DCL	Data Control Language
DDL	Data Definition Language
DML	Data Manipulation Language
EIS	Enterprise Information System
EJB	Enterprise Java Beans
ERP	Enterprise Resource Planning systems
FISH	Fluorescence In-Situ Hybridization
FLIP	Fluorescence Loss in Photobleaching
FRAP	Fluorescence Recovery After Photobleaching
FRET	Fluorescence Resonance Energy Transfer
GFP	Green Fluorescence Protein

GID	Global Image Database
GUI	Graphical User Interface
HQL	Hibernate Query Language
HTTP	Hypertext Transport Protocol
IDE	Integrated Development Environment
IoC	Inversion of Control
IT	Information Technology
J2EE	Java 2 Enterprise Edition
JAAS	Java Authentication and Authorization API
JAXB	Java Architecture for XML Binding
JAXP	Java API for XML processing
JAXR	Java API for XML Registries
JAXRPC	Java API for Remote Procedure Calls
JDBC	Java Database Connectivity
JDO	Java Data Objects
JMS	Java Messaging Service
JMX	Java Management Extension
JNDI	Java Naming and Directory Interface
JSP	Java Server Pages
JTA	Java Transaction API
LAM	Local Area Multicomputer Messa Passing Interface
LDAP	Light Weight Application Protocol
MDA	Model Driven Architecture
MDP	Message Driven POJO
MDR	Metadata Repository
MIME	Multipurpose Internet Mail Extensions
MOF	Meta Object Facility
MOM	Java Message Oriented Middleware
MPI	Message Passing Interface
MVC	Model View Controller
ODBC	Open DataBase Connectivity
OGNL	Object Graph Navigation Languag
OME	Open Microscopy Environment
ORM	Object Relational Mapping
PDF	Portable Document Format
PIM	Platform Independent Model

POJO	Plain Old Java Object
POM	Project Object Model
PSM	Platform Specific Model
PVM	Parallel Virtual Machine
QBC	Query By Criteria
QBE	Query By Example
RDBMS	Relational Database Management System
RMI	Remote Method Invocation
RPC	Remote Procedure Call
SAAJ	SOAP with Attachments API for Java
SFSB	StateFull Session Bean
SLSB	StateLess Session Bean
SMILE	Scientific Microscopy Lab Environment
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
UDDI	Universal Description, Discovery and Integration
UML	Unified Modelling Language
WML	Wireless Markup Language
WSDL	Web Service Description Language
XMI	XML Metadata Interchange

Appendix C

Acknowledgments

This work was supported by GENAU: BIN, Bioinformatics Integration Network. I would like to express my deepest gratitude to my mentor, Zlatko Trajanoski, for his encouragement, vision, and belief in me.
Further thanks go to former members of the ”Bioinformatics group” and those at the ”Institute for Genomics and Bioinformatics” for their fruitful discussions, support and friendship. A special thank is owed to all those people who has actively contributed to this work in alphabetical order: Gabriella Bindea, Maria Fischer, Hubert Hackl, Michael Hanscho, Juergen Hartler, Michael Maurer, Robert Molidor, Dietmar Rieder, Gerhard Thallinger, Elmar Trost, Fatima Sanchez-Cabo, Guenter Schwann and Alexander Sturn.
I would also like to acknowledge James McNally and colleagues at the ”Laboratory of Receptor Biology and Gene Expression” (NCI, Bethesda) for giving me the opportunity to gain insights into the exciting world of microscopy. Special thanks go to Hillary Mueller, who had the patience to introduce a ”computa guy” to the ”art of cell culture”, as well as the friendly support in Washington DC.
I am indebted to my parents for their unfailing support, and to my companion in life, Sigi, for being with me and for her love, encouragement and understanding.

Appendix D

Publications

G. G. Thallinger, S. Trajanoski, G. Stocker, Z. Trajanoski. Information management systems for pharmacogenomics. Pharmacogenomics. 3: 651-667 (2002) PMID: 12223050

G. Stocker, D. Rieder, Z. Trajanoski. ClusterControl: a Web interface for distributing and monitoring bioinformatics applications on a Linux cluster. Bioinformatics. 20: 805-807 (2004) PMID: 14751976

H. Hackl, M. Maurer, B. Mlecnik, J. Hartler, G. Stocker, D. Miranda-Saavedra, Z. Trajanoski. GOLDdb: Genomics of lipid-associated disorders database. BMC Genomics. 5: 93-93 (2004) PMID: 15588328

M. Maurer, R. Molidor, A. Sturn, J. Hartler, H. Hackl, G. Stocker, A. Prokesch, M. Scheideler, Z. Trajanoski. MARS: microarray analysis, retrieval, and storage system. BMC Bioinformatics. 6: 101-101 (2005) PMID: 15836795

C. Vogl, F. Sanchez-Cabo, G. Stocker, S. Hubbard, O. Wolkenhauer, Z. Trajanoski. A fully Bayesian model to cluster gene-expression profiles. Bioinformatics. 21 Sup. 2: ii130-ii136 (2005) PMID: 16204092

J. Rainer, F. Sanchez-Cabo, G. Stocker, A. Sturn, Z. Trajanoski. CARMAweb: comprehensive R- and BioConductor-based Web service for microarray data analysis. Nucleic Acids Res. 34: W498-W503 (2006) PMID: 16845058

REVIEW

Information management systems for pharmacogenomics

Gerhard G Thallinger, Slave Trajanoski, Gernot Stocker, Zlatko Trajanoski†

†Author for correspondance Institute of Biomedical Engineering, Graz University of Technology, Krenngasse 37, A-8010 Graz, Austria Tel: +43 316 873 5332; Fax: +43 316 873 5340; E-mail: zlatko.trajanoski@tugraz.at

The value of high-throughput genomic research is dramatically enhanced by association with key patient data. These data are generally available but of disparate quality and not typically directly associated. A system that could bring these disparate data sources into a common resource connected with functional genomic data would be tremendously advantageous. However, the integration of clinical and accurate interpretation of the generated functional genomic data requires the development of information management systems capable of effectively capturing the data as well as tools to make that data accessible to the laboratory scientist or to the clinician. In this review these challenges and current information technology solutions associated with the management, storage and analysis of high-throughput data are highlighted. It is suggested that the development of a pharmacogenomic data management system which integrates public and proprietary databases, clinical datasets, and data mining tools embedded in a high-performance computing environment should include the following components: parallel processing systems, storage technologies, network technologies, databases and database management systems (DBMS), and application services.

Keywords: application services, data warehouse, information management system, network technologies, parallel algorithms

Ashley Publications Ltd www.ashley-pub.com

Disease phenotypes arise from complex interactions of organisms with their environments. While there is a long history of associating genes and gene defects with a large number of diseases, a growing body of data suggest that many disease phenotypes arise from gene–gene interactions and the interactions of genes with their environments – including the genetic background in which those genes are expressed. With the sequencing of the human genome [1,2], the genomes of model organisms, and development of high-throughput technologies, for the first time there is the means to dissect complex phenotypes and develop novel diagnostic, prognostic, therapeutic and preventive strategies.

While the principles of the underlying high-throughput technologies (e.g., cDNA microarrays [3]) are relatively simple, the challenges associated with examining thousands of data points and presenting those data in a usable form are substantial. Further, the true value of the data lies not in the result of any single experiment or study but rather in examining correlations across multiple experiments and study groups. Moreover, the value of high-throughput genomic research is dramatically enhanced by association with:

- key patient data
- clinical lab results [4]
- X-ray and magnetic resonance imaging (MRI) data
- serum enzyme levels
- patient history including lifestyle

These data are generally available but of disparate quality and not typically directly associated. If the patient name can be coded to ensure confidentiality, a system that could bring these disparate data sources into a common resource connected with functional genomic data would be tremendously advantageous. The prerequisite for personalized medicine or pharmacogenomics will be achieved by combining these clinical data sets with genome information management systems. However, disparate data sources, like public or proprietary molecular biology databases, laboratory information management systems (LIMS) and clinical information management systems (CIS) pose significant challenges to integrating and querying these data. Furthermore, high-throughput genomic technologies require the nontrivial development of information management systems embedded in a high-performance computing environment. An integration of clinical and accurate interpretation of the generated functional genomic data requires the development of information management systems capable of effectively capturing the data as well as tools to make that data accessible to the laboratory scientist or to the clinician.

In this review these challenges and current IT solutions associated with the storage, manage-

 ISSN 1462-2416

ment and analysis of high-throughput data are highlighted. Having in mind a heterogeneous audience of clinicians, R&D professionals and decision-makers in the pharmaceutical and biotech industries and not technology specialists, the focus is on general systems rather than specific turnkey pharmacogenomic solutions. It should be pointed out that both information technologies and high-throughput genomic technologies are rapidly developing areas. Thus, rather than being a comprehensive overview of the current IT solutions, this review represents a guide for better communicating and understanding the IT infrastructure needs for a pharmacogenomic laboratory.

The amount of data generated by modern molecular biology is precedented only by high-energy physics, climate modeling, geophysics and astronomy [34]. The efforts associated with adequately planned and properly structured resources for a professional bioinformatic practice are not always fully appreciated. As of June 2002, 90 GB of disk space is required solely for the uncompressed GenBank files and the volume doubles every 14–18 months [5,29]. The volume of a small selection of public databases that the authors are using in their laboratory [101] including their index files and different formats is 250 GB. Although a number of public and private websites offer access to the databases, it is argued that a local installation and maintenance of the most important databanks including GenBank [5], SWISSPROT [6] and PDB [7] is required in order to enable:

- integration with the clinical data
- integration of functional genomic data generated in the local institution
- complex queries across the various databases

Local installation and maintenance is necessary for the three reasons:

- First, public websites have certain limitations regarding the size of the files that can be down- or uploaded and usually do not allow processing of hundreds or thousands of jobs in a batch mode.
- Second, integration of proprietary local and public remote databases is not easily possible.
- Third, optimization of certain computational processes that are relevant to the problem of interest can only be achieved if access at the level of system administrator is guaranteed.

Over and above local installation and maintenance of public databases, a number of standard sequence analysis algorithms (e.g., BLAST [8], FASTA [9], HMM [10]) and specific bioinformatic tools (e.g., analytical pipeline for gene expression profiling or mass spectrometry data analyzes) also have to be set up on-site and administered. Clearly, setting up a pharmacogenomic data management system that integrates public and proprietary databases, clinical data sets and data mining tools requires a high-performance computing environment including:

- parallel processing systems
- storage technologies
- network technologies
- databases and database management systems (DBMS)
- application services

In the following paragraphs the necessary components for storage, management and analyses of high-throughput data generated in genomic studies are highlighted.

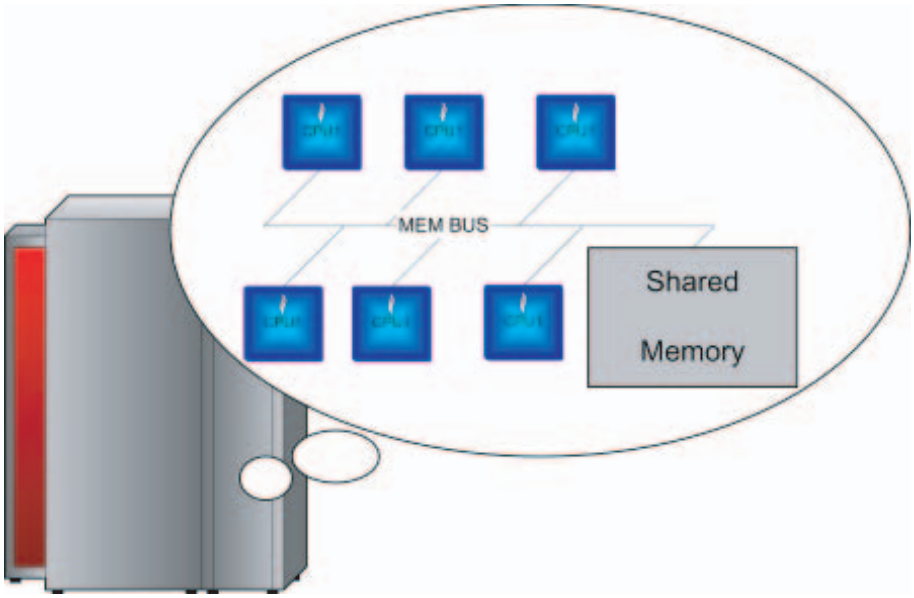
Parallel processing systems

The analysis of the enormous amounts of data requires parallel architectures [23] and methods [20] to solve the computational problems of genomic applications in a reasonable time. Three different architectures are distinguished to implement parallel processing on current computer platforms:

- shared memory systems
- distributed memory systems
- combination of both systems

The main characteristic of shared memory systems is a central memory (e.g., 16–32 GB) that is accessed directly through a fast bus system from multiple processors (**Figure 1**). Therefore, significant parts of the data (e.g., databases) can be loaded into that global memory at once, and all processors connected to this memory can solve the numerical problems sharing the same database at the same time. The communication between the processors is performed using the shared memory pool with efficient synchronization mechanisms. This kind of systems is limited by the number of processors that can be combined (generally 4–16) and the bandwidth of the memory bus used for interprocess communication. The high costs of such systems are limiting factors for organizations with restricted financial possibilities. If there is need for high computing performance for an application, which can not be distributed, these systems are the first choice.

Figure 1. Shared memory architecture.



A shared memory architecture consists of several microprocessors (CPUs) processing data that is stored in a large randomly accessed memory pool.

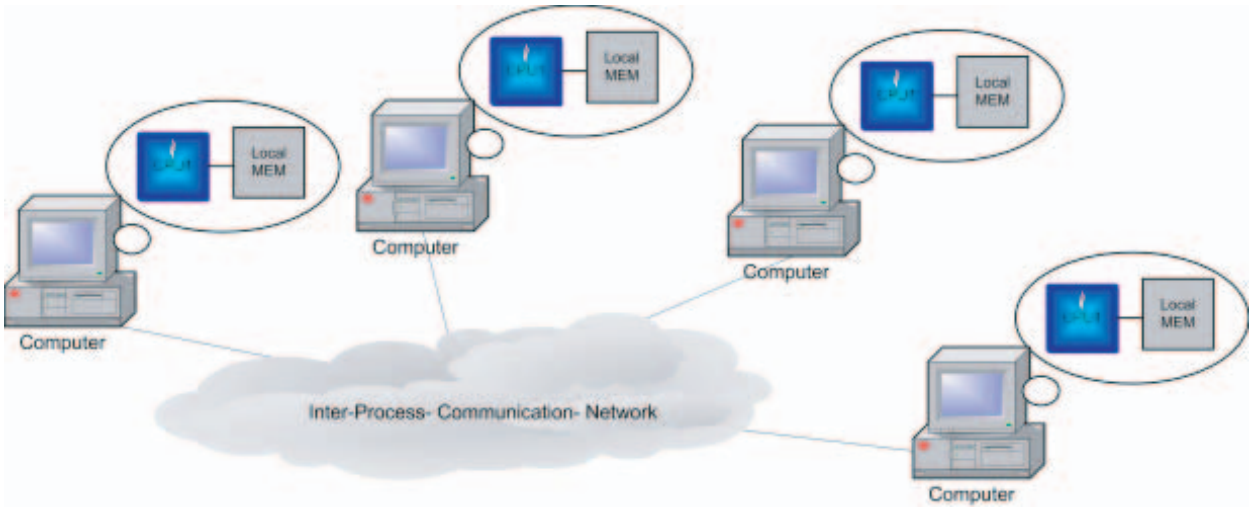
The second possibility to achieve parallel computation with lower costs is the distributed memory solution (**Figure 2**). In general, these systems consist of a cluster of serial computers, so-called nodes, that solve dividable numerical problems. These nodes normally run under open source operating systems (e.g., Linux, FreeBSD) that can be easily extended by special cluster environments, like the BeoWulf-Project [102] or MOSIX [103], running parallel applications with communication libraries like PVM (Parallel Virtual Machine) [58-60] or MPI (Message Passing Interface) [61,82]. Every node contains:

- an off the shelf processor (32-bit or 64 bit)
- local memory in the order of 1–4 GB
- a local hard disk (for intermediate storage of input data or results)

Data required for the computation, like intermediate results from other nodes, has to be requested via a high-performance communication network. This is the characteristic property of distributed memory systems and can also be their bottleneck. If an algorithm needs to access centrally stored data, the nodes of the distributed memory system have to communicate permanently via network-messages. This can result in network congestion, decreasing the overall system performance. As a direct consequence, distributed memory systems are ideal for atomic numerical computational intensive problems that have a low communication/calculation time ratio. To increase the performance, this type of architecture is simply extended by connecting additional nodes to the computing network. An additional improvement is to keep genomic databases or at least parts of them locally stored on the disk of every node with the disadvantage of increased administrative effort.

A new approach of distributed parallel-computing was recently developed: GRID-Computing [16], i.e., a network of IT-resources placed over various locations, working together on numerical problems via the Internet. For every computational problem the appropriate computing facility in a worldwide resource pool can be harnessed to contribute to its solution. Management of work distribution and the combination of multiple administrative computing domains depending on their availability, capabilities, performance and costs are topics of ongoing research [33]. A computing GRID differs from the earlier described cluster topology mainly by the fact that there is no central resource management system. In a GRID every node has its own resource management system and its own distribution policy. Conventional cluster systems can be nodes of the computing GRID but are not

Figure 2. Distributed memory architecture.



In a distributed memory architecture the various computing devices (e.g. PCs) have their own local memory, and perform calculations on distributed problems. Input data and results are exchanged via an inter-process communication network.

GRIDs *per se*. Currently ongoing projects like 'The Virtual Lab' [35], 'Virtual screening project' [36], 'Folding@Home' [37] and 'BioGrid' [104] try to use globally available computing power by using different GRID software like Globus, Cactus, GridBus etc. [33].

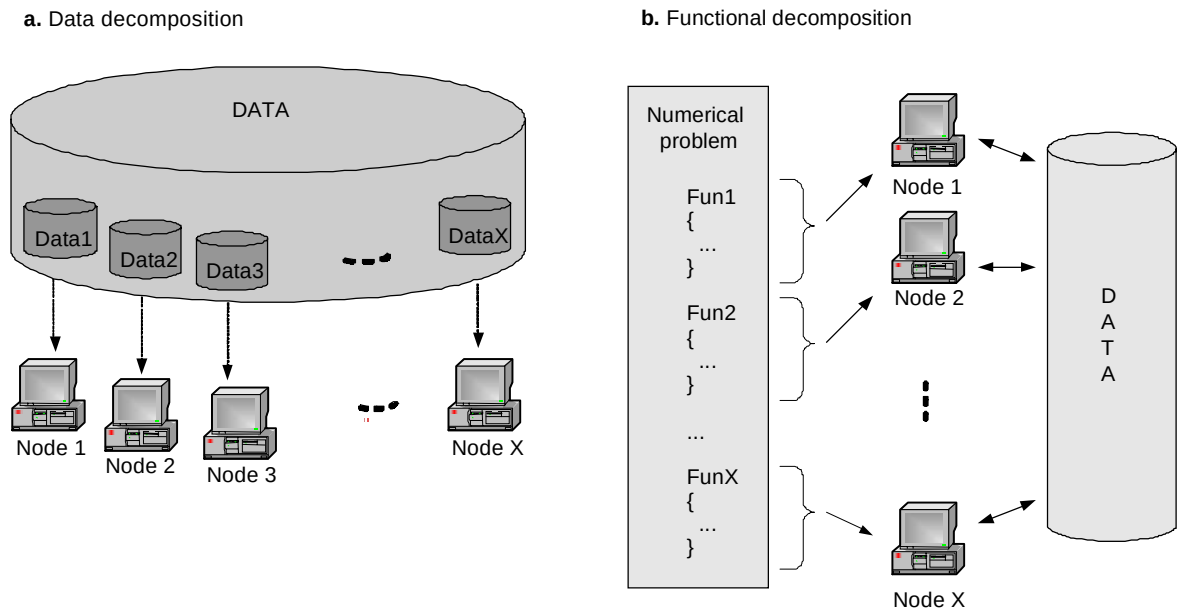
All systems mentioned represent multiple instruction multiple data (MIMD) devices. These systems run multiple instructions on different processors working with multiple data. A different approach represents single instruction multiple data (SIMD) systems. A typical SIMD machine consists of 100–1000 simple processors with small local memory calculating at the same cycle of the processor, the same instruction on different data [14,18]. In this way algorithms are hardware-implemented on processors and data can be processed in parallel [38]. Such a SIMD system has the drawback that it is dedicated only for special purposes (e.g., calculation of Smith-Watherman algorithm) and cannot be used for other calculations.

On MIMD parallel hardware every serial program can be run sequentially. However, in order to use the parallel features of the computing facility, the software has to meet parallel demands too. A numerical problem that has to be solved in parallel must be divided into subproblems that can be subsequently delegated to different processors. This partitioning procedure can be done either with so-called 'domain decomposition' or 'functional decomposition' (**Figure 3**).

The term 'domain decomposition' describes the approach to partition the input data and process the same calculation on each available processor. Most of the parallel-implemented algorithms are based on this approach dividing the genomic databases into pieces and calculating the sequence alignment of a sequence on a subpart. The second and simplest way to implement the domain decomposition on a cluster system is to take sequentially programmed applications and run them on different nodes with different parameters for example, run the well known BLAST with different sequences against one database by giving every node another sequence. This form of application parallelization is called swarming and does not need any adaptation of existing programs. In this context it is very useful to have a survey program that manages the job distribution automatically (see the swarming applications).

On the other hand 'functional decomposition' is based on the decomposition of the computation process. This can be done by discovering disjoint functional units in a program or algorithm and sending these subtasks on to different processors. Interactions between the processing units and exchange of intermediate values have to be implemented with interprocess communication either via a network or global shared memory pool. Finally in some parallel implementations combinations of both techniques are frequently used, so that 'functional-decomposed units' are calculating 'domain-parallelized' sub-

Figure 3. Data and functional composition.



a. Data decomposition is a computational paradigm where data to process is distributed and processed on different nodes. **b.** Functional decomposition divides the computational problem in functional units, which are distributed onto different working nodes processing the same data.

tasks. An overview of the widely used parallel applications in bioinformatics concludes the section about parallel processing.

FASTA

FASTA [9] is one of the frequently used sequence homology search tools based either on heuristic search protocol (fasta, fastx, etc.) or Smith-Waterman local alignment algorithm (SSearch). The standard FASTA distribution contains a parallel implementation of both alignment algorithms, which supports the communication libraries MPI and PVM. It is sufficient to compile the alignment programs with parallel support and run them in the prepared parallel MPI or PVM environment. It is noteworthy that only the master process must have direct access to the database sources and the nodes get the data from this controling process. This reduces the network file system (NFS) load on the fileserver, which holds the databases but can congest the communication network [105].

ClustalW

This multiple sequence alignment tool [21,106] has a parallel implementation supported only on SGI platforms [106]. Current development targets into the direction of parallel implementation for all platforms. A new dynamic-programming-improved and multithreaded implementation of ClustalW is already available for several platforms [107] and an MPI-version should follow soon.

HMMER

HMMER is an implementation of profile 'Hidden Markov Models' (HMM) methods for sensitive database searches using multiple sequence alignments as queries [108]. This application is programmed with multithreading support and can also be run on PVM-Cluster compounds.

BLAST

BLAST is the most famous algorithm for sequence alignment [8,19], it has two main implementations, one developed at the National Center for Biotechnology Information (NCBI) [109] and the other at Washington University, USA [110]. Both of them have an option for multithreading support and the NCBI-version is used by several commercial solutions to build distributed systems and cluster enabled turnkey solutions. Gene or protein databases and query sequences are split into pieces and 'blasted' on distributed nodes. After the calculations are finished the results are collected and combined into

one result similar to the one that is calculated by one single BLAST on the whole database. Some MPI and PVM solutions for BLAST were in development while this article was in preparation, but there was no public release available. Known parallel approaches (BeoBLAST [15], TurboBLAST [39]) implement parallel BLAST by swarming single BLAST jobs through different job-queuing systems on distributed memory systems. SGI has augmented BLAST in the form of HT-BLAST [40] targeting an increased throughput of BLAST alignment by adapting the standard implementation from NCBI to their shared memory environment. Different instances of BLAST get 10–25 input sequences from a distribution system and align sequences against shared, memory mapped databases within different threads. In this way a continuous stream of sequences can be BLAST-aligned without the penalty of the setup delay during the start of a BLAST process.

Swarming

The simplest way of processing the vast amount of input-files on a cluster consists in swarming the atomic serial- or multithreaded-programmed applications onto the cluster. The following applications can be used.

Ppmake

Ppmake is a PVM enabled parallel make clone, that spreads makefile execution over a PVM cluster. It does not require any changes to normal makefiles or the make executable but it must have access to all source files and executables from every node [111].

Multiblast

Multiblast is used for the efficient processing of a large number of query sequences against a BLAST database. It consists of a collection of scripts that split the input sequences into directories. An MPI-control program assigns to every working node a directory and surveys the execution of BLAST for all entries in the directory [112].

MPI-swarm

This MPI swarm program, like multiblast, takes all files from an input directory. The job dispatcher acquires working nodes via MPI and tells the node which program to invoke, together with the input file and destination for the output. After the worker finishes one job it gets a new input file from the master process until all files from the input directory are processed [113].

These algorithms produce, especially for large-scale approaches on big databases, megabytes of output that is manually not manageable. Therefore, preprocessing of the sequences to reduce redundant information and postprocessing of the results is inevitable and will be the challenge in the future. Benchmarks on some of the above can be found elsewhere [41-43].

Storage technologies

Within the last few years there has been a dramatic growth in storage data. Industry experts estimate that the amount of data stored is increasing at a rate of 70–80% every year [31]. With development of IT and the use of computers in every segment of life, science, government and corporate world generate terabytes of user data. This is the reason why in the last few years new storage architectures have been emerging on the market. Currently, the following solutions are in use:

- Directly attached storage (DAS)
- Network attached storage (NAS)
- Storage area networks (SAN)
- Internet SCSI (iSCSI)

Directly attached storage

This is historically the first and very basic method for attaching storage to a computer system. Such storage can be seen today in every PC, like hard disk, floppy disk or CD-ROM attached with short internal cables with the main host. In the mainframe arena storage devices, hard disk drives or tape drives are separate boxes connected with cables to the host. From a functional perspective this is the same configuration as for a PC.

Network attached storage

Storage systems, which optimize the concept of file sharing across the network, are known as NAS. NAS is defined by the Storage Networking Industry Association (SNIA) [121] as storage elements that connect to a network providing file access services to computer systems. NAS devices are connected directly to the existing local area networks (LAN) using standard TCP/IP protocols. Therefore, the storage is not restricted to attach to a specific server and any-to-any connectivity is provided. NAS devices exploit the existing LAN topology and any user running any operating system can access them. To provide all these services NAS has a built in intelligent controller, which is actually a small server with a striped operating system running on it (for

Table 1. Advantages and disadvantages of NAS.

Advantages	Disadvantages
As one box solution, NAS is easy to configure and install. It is plugged in the existing LAN and configured with IP address from the subnet.	Because of high-level access to NAS devices, there is software overhead impact on the performance. TCP/IP as a protocol is designed to bring data integrity in Ethernet networks, which consumes processing power.
Exploits existing IP network infrastructure. Since it is connected to the existing LAN infrastructure there are minimal costs of implementation and staff with existing skills can carry out the installation and configuration.	LAN networks, especially Ethernet, are tuned for short burst transmissions and not large continuous data transmissions. For NAS devices in the LAN sufficient spare bandwidth should exists.
NAS devices can be easily upgraded in capacity and performance within the appliance's configuration limits.	Backup and restore can generate increased traffic in the network and thus, implicate significant performance downsides on user applications.
Storage decision is separated from server decision, thus providing users with more choices for their needs.	Because of its file I/O access NAS devices are not usually suitable and often not supported for large database models that rely on block level access.
NAS backup is common support for major software backup programs.	
NAS devices support multiple client file systems. Many organizations have diverse and mixed platform environments and with NAS it is not necessary to obtain server for each environment. Heterogeneous file sharing is enabled with translation between NFS and CIFS.	
Because of its LAN connection, NAS can provide any-to-any connectivity.	

I/O: Input/output; IP: Internet protocol; LAN: Local area networks; NAS: Network attached storage; TCP: Transmission control protocol.

example Linux or Windows). Storage is directly connected to this server, usually with a SCSI connection and at present time, Fibre Channel connections are becoming more popular. The advantages and disadvantages of NAS are outlined in **Table 1**.

Storage area networks

A SAN is a specialized, dedicated high-speed network (**Figure 4**). A SAN is defined by SNIA as a network whose primary purpose is the transfer of data between and among computer systems and storage elements. Like a LAN, a SAN allows any-to-any connections across the network, using interconnect-elements, such as routers, gateways, hubs and switches. Fibre Channel is the *de facto* SAN networking architecture, although other network standards could be used. The advantages and disadvantages of SAN are outlined in **Table 2**. Communication inside SAN happens between these elements:

- Server–storage: communication as in direct access storage configuration, the only difference is the distance that can bridged by the connection.
- Server–server: providing high-speed communication between servers.
- Storage–storage: very convenient for serverless backup and storage mirroring.

Internet SCSI

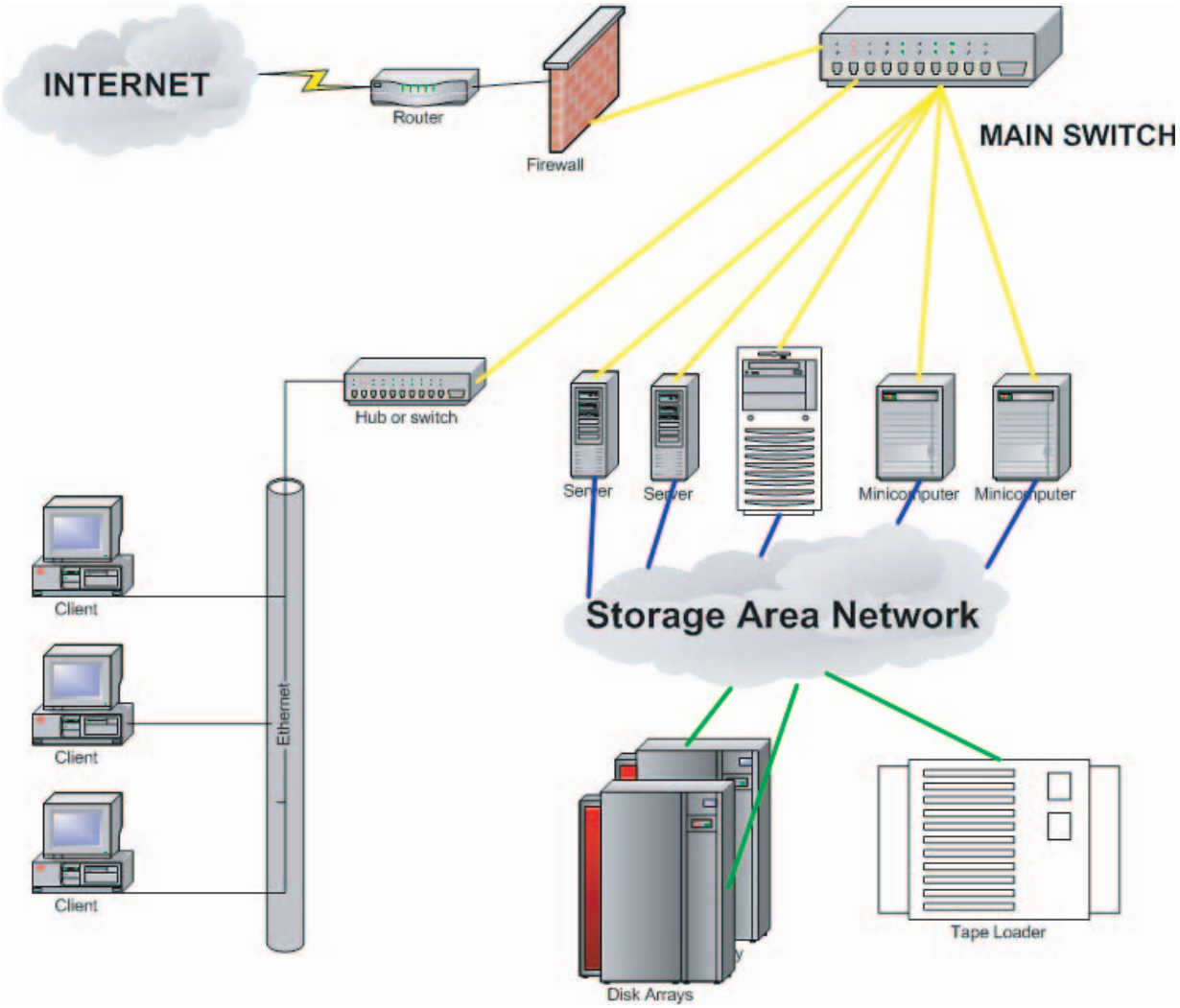
iSCSI is a new technology, which has been proposed as an industry standard by IBM and Cisco to the Internet Engineering Task Force (IETF). The idea behind iSCSI is the encapsulation SCSI commands in TCP/IP packets and sending them through standard IP based networks (Gigabit Ethernet, FDDI or ATM – see more on this in the network technologies section). With this approach iSCSI storage elements can exist anywhere on the LAN and any server with an iSCSI host adapter can access them.

Which storage network?

Each of the storage network solutions is optimized for a differing but sometimes overlapping environment:

- DAS is optimized for single, isolated processors, delivering good performance at a low initial cost.
- SAN is a robust storage infrastructure, optimized for high performance and enterprise-wide scalability.
- Integrated NAS appliances are discrete pooled

Figure 4. Management system infrastructure using SAN.



The SAN requires dedicated networking architecture, usually a Fibre Channel network, and allows any–any connections across the network, using routers, hubs and switches.
SAN: Storage area network.

disk storage subsystems, optimized for ease-of-management and file sharing, using lower-cost IP-based networks.

- iSCSI is very new technology and not many products are present on the market, but it may play a bigger role in the future.

Networking technologies

Modern organizations depend on their LANs to provide connectivity for a growing number of complex, mission-critical desktop computing applications (**Figure 4**). However, as the volume of network traffic increases, the bandwidth offered by a typical 10 Mbps Ethernet LAN quickly becomes inadequate to maintain an acceptable performance. Currently, two network technologies can accommodate the need for high-throughput and high-bandwidth in a LAN:

- Gigabit Ethernet
- ATM

Fiber distributed data interface (FDDI) as the first high-throughput LAN technology, will not be discussed in this review, since it has lost its market share significantly in the past several years.

Table 2. Advantages and disadvantages of SAN.

Advantages	Disadvantages
Single storage solution for all companies' data with access from multiple servers.	Expensive solution, since a new dedicated network has to be built from scratch.
Nondisruptive scalability, easy storage upgrade with no downtime independent of servers.	Lack of standards in comparison with Ethernet, which is long and proofed technology.
High performance through the use of SCSI-3 protocol and current speeds of up to 80 MBps. With new 2 Gbps Fibre Channel hardware emerging on the market even higher transfer rates are possible.	Special knowledge and skills are needed for implementing, configuring and maintaining SAN.
Tape devices in SANs can be used by any device connected to the SAN.	
SAN enables for server clustering as single storage accessible from all connected cluster nodes.	
Possibility for disaster tolerance and site mirroring.	

SAN: Storage area network.

Gigabit Ethernet

Behind Gigabit Ethernet is one of the most used and proven technologies for LAN – Ethernet. According to Interntional Data Corporation (IDC) [123] by the end of 1997 >85% of all installed connections were Ethernet. The remaining network connections are a combination of Token Ring, FDDI, ATM and other protocols. Gigabit Ethernet is an evolution in LAN technologies, continuing the development of Ethernet, beginning with 10Base-T and Fast Ethernet. It employs the same Carrier Sense Multiple Access with Collision Detection (CSMA/CD) protocol, same frame format and same frame size as its predecessors. This means easy and low cost upgrade path to Gigabit Ethernet, while keeping the existing networking equipment. Gigabit Ethernet supports new full duplex operating modes for switch–switch and switch–end–station connections. By itself, Gigabit Ethernet does not provide quality of service (QoS), as Ethernet in general defines only data link Layer 2 of the OSI protocol model. Since QoS is becoming an important issue in LAN environment RSVP (resource reservation protocol) has been defined as a new standard by the Internet Engineering Task Force (IETF) [120].

Asynchronous transfer mode

Asynchronous transfer mode (ATM) is a replacement of the previous, inefficient synchronous transfer mode (STM). ATM is based on switching of fixed sized cells, in contrast to the variable length frame used by Ethernet, Token Ring and FDDI. The switching of ATM cells can be done in hardware instead of software resulting in a much higher transmission speed. Since individual cells can be positioned anywhere within the datastream they have to be reassembled at the receiving station. ATM currently works with speeds between 155 Mbps– 2.5 Gbps. ATM is very flexible and allows transmission of various media types such as voice, video, data files etc. As each of the media types requires different network settings, ATM provides individual QoS settings at the connection stage. ATM allows dedicated circuits with guaranteed bandwidths and differing requirements to be set up simultaneously. This is achieved by dividing the network traffic into low and high priority traffic. During network or switching congestion low priority traffic is discarded in favor of high priority traffic.

A key advantage of ATM is that it can carry traffic of different types, like voice, video, data etc. Traffic management in ATM supports this by appropriately providing QoS for different types of traffic resulting in an efficient bandwidth utilization of the network [114].

Gigabit Ethernet versus ATM

Gigabit Ethernet is the network of choice for easy installation and configuration. It is identical with it predecessors using same frame format, so training for users and skilled administrators to establish the new network is not necessary. Gigabit Ethernet equipment fits easily and coexists with the current fast Ethernet or Ethernet switches, hubs and routers. In contrast, ATM is more costly, more complex and needs educated network administrators to install and manage it. Currently, > 80% of existing LANs are Ethernet

based, meaning that by most installations when ATM is chosen, upgrade should be performed. ATM cards are more expensive then Gigabit cards as well as other networking equipment and it is not compatible with the existing Ethernet or fast Ethernet topology. However, if audio and video streaming in the network is used, ATM is the technology of choice. ATM has built in support for defining reserved and guaranteed bandwidth and there are many mechanisms inside the protocol that can increase efficient bandwidth utilization. Another of ATMs strong points is its scalability: it can be used either as wide area network (WAN) or LAN solution or both. This means seamless integration of ATM LAN networks into future WAN configuration.

In conclusion, Gigabit Ethernet is the easiest and most affordable network upgrade, whereas when more sophisticated, controlled and dedicated network traffic is needed, ATM is the better choice.

Databases and database management systems

Genomic data has to be stored in and needs to be easily retrieved from appropriate databases, which are either publicly available and/or proprietary. The major problem is the integration of the heterogeneous data sources, whose number is steadily increasing [30]. Flat files (GenBank, SWISSPROT, EBML-Bank [29]) with widely varying formats and relational databases based on disparate relational models and/or different object semantics (e.g., PDB, ArrayExpress [27,115]) need to be integrated. Several efforts are underway to standardize either relational models and/or objects semantics. In the area of microarray databases [27,28], the MGED (Microarray Gene Expression Data) Society [115] proposes with MAGE an object model [22] and with MIAMI [24] minimum information required to unambiguously interpret and verify microarray experiments. The Gene Ontology Consortium [117] tries to provide a structured and standardized vocabulary to describe gene products in any organism [25,26]. Recently, an object model for the data stored in PDB has been finalized [52].

There are two main approaches for data storage in a genomic information management system, one using a data warehouse with data marts and the other uses a federated database system (FDS).

Data warehouse

A data warehouse is a copy of transaction data specifically structured for querying and reporting [11]. It acts as a centralized repository for all information related to the management system. Data has to be imported from various sources to the data warehouse at regular intervals to keep information up-to-date. Applications query the data warehouse and present the results to the user. A data warehouse is accompanied by smaller 'data marts', which contain information related to special task. Data marts duplicate information already present in the data warehouse, which are structured in a different way to support faster responses for specialized queries than the data warehouse itself. There are several limitations to the data warehouse approach:

- The timeliness of the information depends on the interval the external sources are imported to the data warehouse. Changes to the data sources are reflected only after a successful data import.
- Data imports can be time consuming since the data has to be indexed heavily to allow fast responses to queries.
- Information is replicated and stored multiple times in different locations, resulting in higher storage requirements and operating costs.
- Searches already present for data sources have to be reimplemented in the data warehouse.

Federated database system

In a FDS 'external' data sources are not imported to a centralized information repository but accessed directly through a federated database server. No replication of data is necessary; the integration of data sources can either be complete (all data can be accessed) or partial (only information needed is available through the server). The basic components of a FDS [12] are:

- a common data model that should be flexible enough to represent different kinds of information from different sources and cope with missing or wrong information
- a common language that could be used to submit queries to the integrated system

Shortcomings of a FDS are that queries spanning different data sources can be slow and the federated database server is quite complex. In spite of these shortcomings, a FDS is the architecture of choice for a genomic information management system, since it avoids data duplication and overheads of a relational DBMS in the case of flat files. The sequence retrieval system (SRS) [53,54] initially developed at EMBL and EBI uses this approach in a modified form by heavily indexing

the genomic flat file databases. This leads to retrieval times, which are 10–100 times faster [55] compared to the retrieval of the same data using an RDBMS. SRS is installed at academic institutions and utilized in commercial companies [56].

Application services

Some of the less complex tasks in pharmacogenomics (e.g., microarray image analysis or gene expression clustering with < 10,000 genes), can be performed directly on a user workstation. The applications are installed locally on the client and required databases are accessed directly through the different database servers. As soon as the tasks become more complex and time consuming or the number of users increases substantially, this approach (often called the 'fat client' approach) has serious drawbacks:

- Workstation power has to be increased to achieve reasonable response times for complex tasks.
- The throughput of each of the different database servers has to be increased either by load balancing and/or clustering.
- Applications and database clients have to be deployed and kept up-to-date locally on each workstation.
- Parallelization of applications is not easily possible.

Most of these drawbacks can be avoided by moving from this two-tier client-server architecture to a three-tier system with an application server(s) as its central component (**Figure 5**). The application server is the 'middleware' between the clients and the backend database servers. It provides all the necessary business logic to perform the tasks required at a central location. Clients interface the application server either through a Web browser or a thin client application basically providing the user interface. This client application can access services either using an application programming interface (API) or a standardized interface like CORBA. Confining the business logic of the information system in the application server leads to several advantages:

- Complex queries over several databases can be optimized.
- Access to database servers can be rescheduled depending on the priority of a task.
- It is possible to split tasks and distribute them on a computing cluster.
- Maintenance is much easier and upgrades to business logic have to be deployed on the application server only.

Application server architecture provides inherent scalability to support an increasing number of users and more demanding tasks, by clustering the application servers and distribution of the server load through load balancers. Load balancing can be performed by front-end servers, which distribute the client requests to the available application servers in the cluster.

Web Services

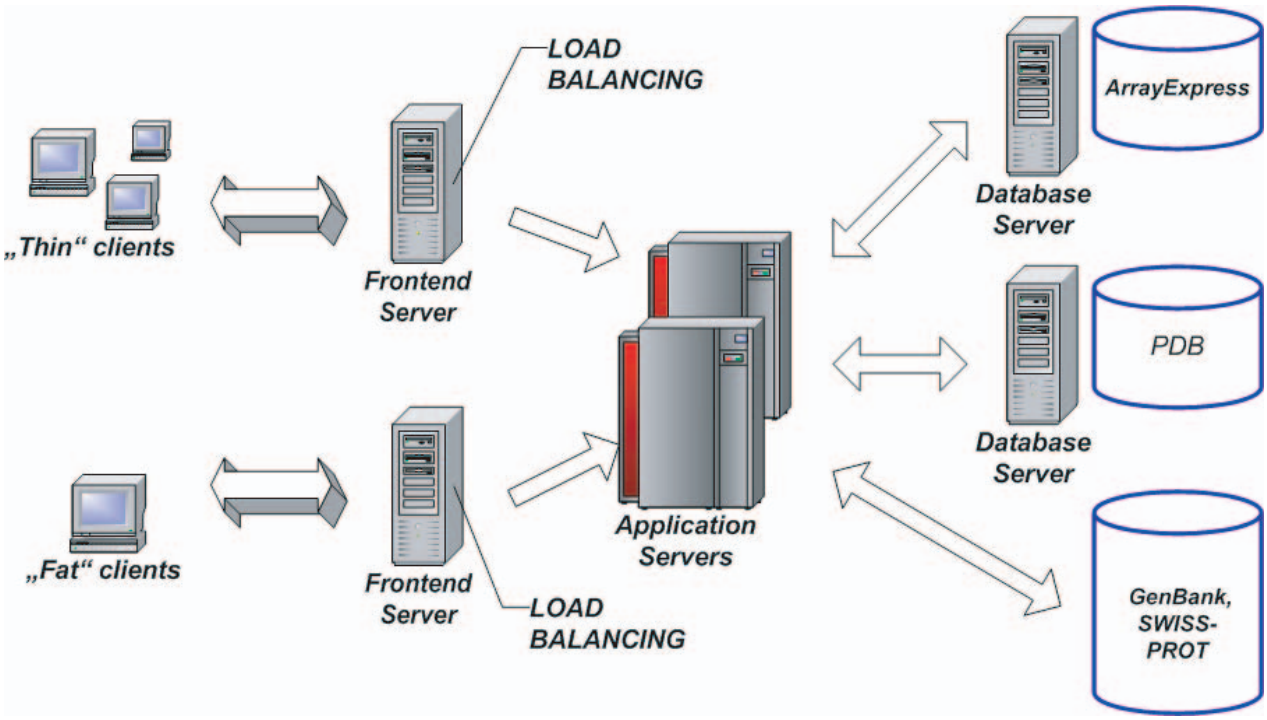
A logical extension of the application services over the Internet are web services [44,17]. While access to application services is in general limited to clients on the LAN, web services can be utilized from anywhere on the Internet. A web service is a 'self-contained, modular applications that can be described, published, located and invoked over a network, generally, the Worldwide Web' [45]. They are based the following standards:

- Extensible markup language (XML) [46] as the formatting language for the data.
- Simple object access protocol (SOAP) [47] as the protocol between the client using the service and the server providing the service.
- Universal description, discovery and integration (UDDI) [48,49] the 'yellow pages' for web services.
- Web services description language (WSDL) [50] providing descriptions of the Web services, their location and specification on how to invoke them.

The main limitation of web services is the network speed, especially when the transfer of a large amount of data with a request or response is necessary, which is quite common for genomic applications. An additional limitation is the use of SOAP as the protocol, since it is based on XML and HTTP, which degrades performance compared to other protocols like CORBA [51]. As soon as the average network speed over the Internet increases, applications can take advantage of the well-defined functionality web services can provide.

Finally, the distributed annotation system (DAS) [13], which allows decentralized sequence annotation, can be seen as an example for web services in the genomics field. The DAS viewer retrieves annotations from various annotation servers and displays them together with the sequence map from the reference sequence server. While based at the moment on a proprietary protocol utilizing HTTP and XML, future versions are likely to use SOAP as the underlying protocol.

Figure 5. Load balanced genomic information management system utilizing a cluster of application servers.



In the case of thin clients the bulk of the data processing occurs on the server whereas fat client is a client that performs th e bulk of the data processing operations. The data itself is stored on the server.

Expert opinion

A pharmacogenomic data management system has to combine public and proprietary genomic databases, clinical data sets and results from high-throughput screening technologies. Data mining tools have to be provided, which require high-performance computing environment. As far as networking infrastructure and storage solutions are concerned, the requirements are satisfied by existing or emerging technologies like Gigabit Ethernet or SAN and NAS solutions from various vendors. Shipped storage capacity increases by 70–80% per year [31], data density on hard disks doubles every year [32] and network speed increases by an order of magnitude every 5 years. In the authors' opinion, the IT challenges in designing and implementing a pharmacogenomic information management system are faced on the software side. Database integration and access has to be optimized and new algorithms for improved search capabilities have to be designed and implemented. Existing algorithms have to be parallelized to take advantage of high performance computing clusters. Although a variety of promising approaches exist already, they have to be combined to fulfill the demands made on a data management system.

An even bigger challenge represents the integration of key patient and genomic data. IT systems for storing, managing and analyzing clinical data are heterogeneous to a great extent, even within a single clinical setting. For example, due to the high degree of automation, blood chemistry data is readily available in a digital form whereas digitally available imaging data coming from NMR or CT sources is difficult to handle. Over and above, regionally varying regulatory, administrative and financial requirements make it difficult to implement a single standardized system. Thus, it will be necessary to design, test, implement and manage customized pharmacogenomic IT infrastructure for a certain location and/or a certain disease.

The efforts and costs associated with the design, implementation and maintenance of an IT infrastructure for pharmacogenomic applications is not adequately appreciated. In general, the following rough rule can be applied: setting up an IT infrastructure is equivalent to setting up a genomic laboratory, i.e., the costs for hardware and housing (including appropriate air con-

Highlights

- High-throughput genomic technologies require high-performance computing environment and appropriate information management systems to store, manage, and analyze the generated data.
- A pharmacogenomic information management system includes: parallel processing systems, storage technologies, network technologies, databases and database management systems, and application services.
- An information management system in pharmacogenomics has to combine public genomic databases with propriatary databases containing patient related data.
- The main challenges in the integration are faced on the software side of the integration, since the requirements on the storage and networking side are satisfied by existing and emerging technologies.
- Standardization efforts on the object model of genomic databases will ease the integration of these databases
- Due to the varying data sources and requirements as well as lack of standards, it will be necessary to develop customized pharmacogenomic IT infrastructure for a certain location and/or a certain disease.
- Design and testing of information management systems for pharmacogenomics and laborious and long-term commitment is necessary to fully exploit the promises of the postgenomic era.

ditioning, power supply and uninterrupted power supply) are similar to the costs of the corresponding data generators using high-throughput technologies (microarray and proteomic facilities). The cost for software and database licenses are equivalent to the material costs, i.e., microarray slides and reagents. However, assuming that Moore's law (the number of elements in an integrated circuit doubles every 18 months) directly influences the half-life time of the hardware (18 months), IT investments pose a significant challenge for the private sector and funding agencies.

Outlook

Current estimates suggest that the proteome will require Petabytes (1000 Tbytes) of storage space and that a system for personalized medicine will be in the range of Exabytes (1,000,000 TBytes) [57]. Currently, small genomic laboratories work with several Terabytes and only a few large laboratories can handle hundreds of Terabytes of data. Since technology is the driving force for modern molecular biosciences, it is believed that personalized medicine, i.e., the development and use of patient-tailored therapy, will soon be a reality. Assuming that storage capacity continues to double every year, in 10 years working with Petabytes will be standard procedure in many institutions and some will even hit the Exabyte range.

It is evident that high-throughput genomic technologies rely on high-performance computing infrastructure. An increasing use of high-end computational infrastructure in a clinical setting will be seen. Integration of patient archiving systems for imaging data (PACS), genomic and pharmacogenomic databases, as well as other laboratory and patient-relevant data will require novel solutions. Integration of patient databases will represent significant challenges for designers and administrators of pharmacogenomic information management systems. The lack of international standards in CIS, patient care and management and different accounting systems will require the development and installation of country-specific (or even regional-specific) systems. Security issues arising from the sensitivity of certain types of information need to be addressed and solved in a proper manner. All these problems can be solved with current and emerging hardware and software technology. However, the design and testing of complex pharmacogenomic systems will be laborious and the long-term commitment of the decision-makers will be necessary to fully exploit the promises of the postgenomic era.

Acknowledgments

The authors would like to acknowledge the valuable comments of the anonymous reviewers. This work was supported by the Austrian Science Fund, Project SFB Biomembranes F718.

Explanation of terms

Term used	Explanation
10Base-T	One of several adaptations of the Ethernet (IEEE 802.3) standard for LANs. The 10Base-T standard (also called Twisted-Pair Ethernet) uses a twisted-pair cable with maximum lengths of 100 meters.
Application server	Also called an appserver: a program that handles all application operations between users and an organization's backend business applications or databases. Application servers are typically used for complex transaction-based applications. To support high-end needs, an application server has to have built-in redundancy, monitors for high-availability, high-performance distributed application services and support for complex database access.
Bandwidth	The amount of data that can be transmitted in a fixed amount of time. For digital devices, the bandwidth is usually expressed in bits per second (bps) or bytes per second. For analog devices, the bandwidth is expressed in cycles per second or Hz. The bandwidth is particularly important for I/O devices. For example, a fast disk drive can be hampered by a bus with a low bandwidth.
Bus	A collection of wires through which data are transmitted from one part of a computer to another. When used in reference to personal computers, the term bus usually refers to internal bus. This is a bus that connects all the internal computer components to the CPU and main memory.
CIFS	A network file system access protocol originally designed and implemented by the Microsoft Corp. under the name Server Message Block protocol and primarily used by Windows clients to communicate file access requests to Windows servers. Today, other implementations of the CIFS protocol allow other clients and servers to use it for intercommunication and interoperation with Microsoft operating systems.
Clustering	Connecting two or more computers together in such a way that they behave like a single computer. Clustering is used for parallel processing, load balancing and fault tolerance. Clustering is a popular strategy for implementing parallel processing applications because it enables companies to leverage the investment already made in PCs and workstations. In addition, it is relatively easy to add new CPUs simply by adding a new PC to the network.
CSMA/CD	A physical layer data transmission protocol used in Ethernet and fast Ethernet networks. Carrier sense refers to arbitration for a shared link. Unlike 'always on' physical protocols, carrier sense protocols require a node wishing to transmit to wait for the absence of carrier (indicating that another node is transmitting) on the link. Multiple access refers to the party line nature of the link. A large number of nodes (up to 500 in the case of Ethernet) share access to a single link. Collision detection refers to the possibility that two nodes will simultaneously sense absence of carrier and begin to transmit, interfering with each other. Nodes are required to detect this interference and cease transmitting. In the case of Ethernet, each node detecting a collision is required to wait for a random interval before attempting to transmit again [118].
Database	A collection of information organized in such a way that a computer program can quickly select desired pieces of data (electronic filing system).
Data mart	A database or collection of databases. Whereas a data warehouse combines databases across an entire enterprise, data marts are usually smaller and focus on a particular subject or department. Some data marts, dependent data marts, are subsets of larger data warehouses.
Data mining	A buzzword for a class of database applications that look for hidden patterns in a group of data. The term is commonly misused to describe software that presents data in new ways. True data mining software does not just change the presentation but discovers previously unknown relationships among the data.
Data warehouse	A collection of data designed to support management decision making. Data warehouses contain a wide variety of data that present a coherent picture of business conditions at a single point in time. The term data warehousing generally refers to combine many different databases across an entire enterprise.
Ethernet	A LAN architecture developed by the Xerox Corp. in cooperation with DEC and Intel in 1976. Ethernet uses a bus or star topology and supports data transfer rates of 10 Mbps. The Ethernet specification served as the basis for the IEEE 802.3 [119] standard, which specifies the physical and lower software layers. Ethernet uses the CSMA/CD access method to handle simultaneous demands. It is one of the most widely implemented LAN standards.
Fat client	In client/server architecture, a client that performs the bulk of the data processing operations. The data itself is stored on the server.
Fibre Channel	A set of standards for a serial I/O bus capable of transferring data between two ports at up to 100 MBytes/second, with standards proposals to go to higher speeds. Fibre Channel supports point to point, arbitrated loop and switched topologies. Fibre Channel was completely developed through industry cooperation, unlike SCSI, which was developed by a vendor and submitted for standardization after the fact [119].
Flat file database	A relatively simple database system in which each database is contained in a single file.
Hub	A communications infrastructure device to which nodes on a multi-point bus or loop are physically connected. Hubs are commonly used in Ethernet and Fibre Channel networks to improve the manageability of physical cables. Hubs maintain the logical loop topology of the network of which they are a part, while creating a 'hub and spoke' physical star layout. Unlike switches, hubs do not aggregate bandwidth. Hubs typically support the addition or removal of nodes from the bus while it is operating.
I/O	The term I/O is used to describe any program, operation or device that transfers data to or from a computer and to or from a peripheral device. Every transfer is an output from one device and an input into another. Devices, such as keyboards and mouse are input-only devices while devices, such as printers are output-only. A writable CD-ROM is both an I/O device.

Explanation of terms

IP	IP specifies the format of packets, also called datagrams and the addressing scheme. Most networks combine IP with a higher-level protocol called TCP), which establishes a virtual connection between a destination and a source. IP by itself is something like the postal system. It allows you to address a package and drop it in the system but there is no direct link between the sender and the recipient. TCP/IP, on the other hand, establishes a connection between two hosts so that they can send messages back and forth for a period of time.
Latency	The amount of time it takes a packet to travel from source to destination. Together, latency and bandwidth define the speed and capacity of a network.
Memory	Internal storage areas in the computer. The term memory identifies data storage that comes in the form of chips and the word storage is used for memory that exists on tapes or disks. Moreover, the term memory is usually used as shorthand for physical memory, which refers to the actual chips capable of holding data. Some computers also use virtual memory, which expands physical memory onto a hard disk. Every computer comes with a certain amount of physical memory, usually referred to as main memory or RAM.
Network	A group of two or more computer systems linked together.
NFS	A distributed file system and its associated network protocol originally developed by Sun Microsystem Computer Corporation and commonly implemented in UNIX systems, although most other computer systems have implemented NFS clients and/or servers. The IETF is responsible for the NFS standard [120].
Processor, microprocessor (CPU)	A silicon chip that contains a CPU. In the world of personal computers, the terms microprocessor and CPU are used interchangeably. At the heart of all personal computers and most workstations sits a microprocessor.
QoS	A technique for managing computer system resources such as bandwidth by specifying user visible parameters such as message delivery time. Policy rules are used to describe the operation of network elements to make these guarantees. Relevant standards for QoS in the IETF are the RSVP and COPS protocols. RSVP allows for the reservation of bandwidth in advance, while COPS allows routers and switches to obtain policy rules from a server [120].
RDBMS	A type of DBMS that stores data in the form of related tables. Relational databases are powerful because they require few assumptions about how data are related or how it will be extracted from the database. As a result, the same database can be viewed in many different ways. An important feature of relational systems is that a single database can be spread across several tables. This differs from flat-file databases, in which each database is self-contained in a single table.
Router	A device that connects any number of LANs.
SCSI	A collection of ANSI standards and proposed standards which define I/O buses primarily intended for connecting storage subsystems or devices to hosts through host bus adapters. Originally intended primarily for use with small (desktop and desk-side workstation) computers, SCSI has been extended to serve most computing needs and is arguably the most widely implemented I/O bus in use today.
Server	A computer or device on a network that manages network resources. For example, a file server is a computer and storage device dedicated to storing files. Any user on the network can store files on the server. A print server is a computer that manages one or more printers and a network server is a computer that manages network traffic. A database server is a computer system that processes database queries.
SNIA	Storage Networking Industry Association [122].
Switch	A network infrastructure component to which multiple nodes attach. Unlike hubs, switches typically have internal bandwidth that is a multiple of link bandwidth and the ability to rapidly switch node connections from one to another. A typical switch can accommodate several simultaneous full link bandwidth transmissions between different pairs of nodes.
TCP	TCP is one of the main protocols in TCP/IP networks. Whereas the IP protocol deals only with packets, TCP enables two hosts to establish a connection and exchange streams of data. TCP guarantees delivery of data and also guarantees that packets will be delivered in the same order in which they were sent.
Thin client	In client/server applications, a client designed to be especially small so that the bulk of the data processing occurs on the server.
Token ring	A type of computer network in which all the computers are arranged (schematically) in a circle. A token, which is a special bit pattern, travels around the circle. To send a message, a computer catches the token, attaches a message to it and then lets it continue to travel around the network.

COPS: Common open policy service; DBMS: Database management system; Hz: Hertz; I/O: Input/output; IP: Internet protocol; LAN: Local area networks; NFS: Network file system; QoS: Quality of service; RDBMS: Relational database management system; RSVP: Resource reservation protocol; TCP: Transmission control protocol.

Bibliography

Papers of special note have been highlighted as either of interest (•) or of considerable interest (••) to readers.

1. Lander ES, Linton LM, Birren B *et al.*: Initial sequencing and analysis of the human genome. *Nature* 409, 860-921 (2001).
2. Venter JC, Adams MD, Myers EW *et al.*: The sequence of the human genome. *Science* 291, 1304-1351 (2001).
3. Shalon D, Smith SJ, Brown PO: A DNA microarray system for analyzing complex DNA samples using two-color fluorescent probe hybridization. *Genome Res.* 6, 639-645 (1996).
4. van't Veer LJ, Dai H, van de Vijver MJ *et al.*: Gene expression profiling predicts clinical outcome of breast cancer. *Nature* 415, 530-536 (2002).

•• **Emphasizes the need to inegrate patient data with genomic databases.**

5. Benson DA, Karsch-Mizrachi I, Lipman DJ, Ostell J, Rapp BA, Wheeler DL: GenBank. *Nucleic Acids Res.* 30, 17-20 (2002).
6. Bairoch A, Apweiler R: The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000. *Nucleic Acids Res.* 28, 45-48 (2000).
7. Berman HM, Bhat TN, Bourne PE *et al.*: The Protein Data Bank and the challenge of structural genomics. *Nat. Struct. Biol.* 7, 957-959 (2000).
8. Altschul SF, Gish W, MillerW, Myers EW, Lipman DJ: Basic local alignment search tool. *J. Mol. Biol.* 215, 403-410 (1990).
9. Pearson WR: Searching protein sequence libraries: comparison of the sensitivity and selectivity of the Smith-Waterman and FASTA algorithms. *Genomics* 11, 635-650 (1991).
10. Eddy SR: Profile hidden Markov models. *Bioinformatics* 14, 755-763 (1998).
11. Kimball R: *The Data Warehouse Toolkit: Practical Techniques For Building Dimensional Data Warehouses*. John Wiley & Sons, New York, USA 310 (1996).
12. Sheth AP, Larson JA: Federated Database Systems for managing distributed, heterogeneous and autonomous databases. *ACM Computing Survey* 22, 183-236 (1990).
13. Dowell RD, Jokerst RM, Day A, Eddy SR, Stein L: The distributed annotation system. *BMC Bioinformatics* 2(1), 7 (2001).
14. Trelles O: On the parallelisation of bioinformatics applications. *Brief Bioinform.* 2(2), 181-194 (2001).

• **Introduction about possibilities and methods of parallelization in the bioinformatic field.**

15. Grant JD, Dunbrack RL, Manion FJ, Ochs MF: BeoBLAST: distributed BLAST and PSI-BLAST on a Beowulf cluster. *Bioinformatics* 18(5), 765-766 (2002).
16. Foster I, Kesselman C: *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, USA (1999).
17. Graham S, Simeonov S, Boubez T *et al.*: Building web services with Java: making sense of XML, SOAP, WSDL and UDDI. SAMS (2001).
18. Flynn MJ: Some computer organizations and their effectiveness. *IEEE Trans. Computer* C-21, 948-960 (1972).
19. Altschul SF, Madden TL, Schaffer AA *et al.*: Gapped BLAST and PSI-BLAST: a new generation of protein DB search programs. *Nucleic Acids Res.* 25(17), 3389-3402 (1997).
20. Foster I: *Designing and Building parallel programs: concepts and tools for parallel software engineering*. Addison-Wesley Publishing Company, Inc., USA (1994).
21. Thompson JD, Higgins DG, Gibson TJ: CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.* 22, 4673-4680 (1994).
22. Gene expression specification. *OMG Final Adopted Specification*. http://www.omg.org/cgi-bin/doc?dtc/02-02-05.pdf (2002).
23. Buyya R: *High Performance Cluster Computing: Architectures and Systems (Vol. 1 & 2).* Prentice Hall, NJ, USA (1999).
24. Brazma A, Hingamp P, Quackenbush J *et al.*: Minimum information about a microarray experiment (MIAME)-toward standards for microarray data. *Nat Genet.* 29, 365-371 (2001).

• **Gives an overview about information which has to be present for microarray experiments.**

25. The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nat. Genet.* 25, 25-29. (2000).
26. The Gene Ontology Consortium. Creating the gene ontology resource: design and implementation. *Genome Res.* 11, 1425-1433 (2001).
27. Brazma A, Sarkans U, Robinson A *et al.*: Microarray data representation, annotation and storage. *Adv. Biochem. Eng. Biotechnol.* 77(Chip Technology), 113-139 (2002).
28. Gardiner-Garden M, Littlejohn TG: A comparison of microarray databases. *Brief Bioinform.* 2(2), 143-158 (2001).
29. Stoesser G, Baker W, van den Broek A *et al.*: The EMBL nucleotide sequence database. *Nucleic Acids Res.* 30(1), 21-26 (2002).
30. Baxevanis AD: The molecular biology database collection: 2002 update. *Nucleic Acids Res.* 30(1), 1-12 (2002).
31. Porter J: Disk Trend 1998 Report, Proceedings of the 100th Anniversary conference: Magnetic Recording and Information Storage Technological Milestones and Future Outlook Conference (1998).
32. Hayes B: Terabyte territory. *American Scientist* 90(3), 212-216 (2002).
33. Buyya R: Economic-based Distributed Resource Management and Scheduling for Grid Computing., PhD Thesis, Monash University, Melbourne, Australia (2002).

• **Good overview about GRID computing.**

34. Allcock B, Bester J, Bresnahan J *et al.*: Data management and transfer in high-performance computational grid environments. *Parallel Computing* 28, 749-771 (2002).
35. Buyya R, Branson K, Giddy J, Abramson D: The virtual laboratory: A toolset to enable distributed molecular modeling for drug design on the world-wide grid. *J. Concurrency Computations: Practice Experience* In press (2002).
36. Richards WG: Virtual screening using grid computing: the screensaver project. *Nat. Rev. Drug Discov.* 1(7), 551-555 (2002).
37. Larson S, Snow C, Shirts M, Pande V: Folding@Home and Genome@Home: using distributed computing to tackle previously intractable problems in computational biology. To appear in Computational Genomics, Richard Grant, Horizon Press (2002).
38. Rognes T: ParAlign: a parallel sequence alignment algorithm for rapid and sensitive database searches. *Nucleic Acids Res.* 29(7), 1647-1652 (2001).
39. Bjornson RD, Sherman AH, Weston SB, Willard N, Wing J: TurboBLAST®: a parallel implementation of BLAST based on the TurboHub® process integration architecture. *Turbogenomics white paper.* http://www.turbogenomics.com/products/turboblast.pdf (2002)
40. Camp N, Cofer H, Gomperts R: 1998: high-throughput BLAST. *SGI White Paper.* http://www.sgi.com/industries/sciences/chembio/resources/papers/HTBlast/HT_Whitepaper.pdf (1998).
41. Chen R, Taaffe-Hedglin C, Willard N, Sherman A: Benchmark and performance analysis of TurboBLAST on IBM® xSeries™ server cluster. *IBM Performance*

Technical Report. (http://www.redbooks.ibm.com/redpapers/pdfs/redp0443.pdf) (2002).

42. SGI high-throughput computational environment performance sheet, (http://www.sgi.com/industries/sciences/chembio/pdf/htc_perf.pdf) (2001).
43. SGI Bioinformatics Performance Report (http://www.sgi.com/industries/sciences/chembio/pdf/bioperf01.pdf) (2001).
44. Newcomer E: *Understanding Web Services: XML, WSDL, SOAP and UDDI.* Addison Wesley Professional (2002).

• **Excellent introduction to Web Service technology.**

45. IBM Web Services Architecture Team: Web Services architecture overview. ftp://www6.software.ibm.com/software/developer/library/w-ovr.pdf (2000).
46. W3C: Extensible Markup Language (XML) 1.0 (Second Edition). *W3C Recommendation*. http://www.w3.org/TR/2000/REC-xml-20001006.pdf (2000).
47. W3C: Simple Object Access Protocol (SOAP) 1.1. *W3C Note*. http://www.w3.org/TR/2000/NOTE-SOAP-20000508 (2000).
48. UDDI.org: UDDI Version 2.01 Operator's Specification. *UDDI Published Specification.* http://uddi.org/pubs/Operators_v2.pdf (2002).
49. UDDI.org: UDDI Version 2.04 API Specification. *UDDI Published Specification.* http://uddi.org/pubs/ProgrammersAPI_v2.pdf (2002).
50. W3C: Web Services Description Language (WSDL) 1.1. *W3C Note.* http://www.w3.org/TR/2001/NOTE-wsdl-20010315 (2001).
51. Anbazhagan M, Nagarajan A: Understanding quality of service for Web services. *IBM White Paper.* ftp://www6.software.ibm.com/software/developer/library/ws-quality.pdf (2002).
52. Macromolecular Structure Specification. *OMG Specification*. http://cgi.omg.org/cgi-bin/doc?formal/02-05-01.pdf (2002).
53. Etzold T, Ulyanov A, Argos P: SRS: information retrieval system for molecular biology data banks. *Methods Enzymol.* 266, 114-128 (1996).
54. Etzold T, Verde G: Using views for retrieving data from extremely heterogeneous databanks. *Pac. Symp. Biocomput.* 134-141 (1997).
55. Zdobnov EM, Lopez R, Apweiler R, Etzold T: The EBI SRS server-recent developments. *Bioinformatics* 18(2), 368-373 (2002).

•• **Describes the structure of SRS and its recent enhacements.**

56. Kerlavage A, Bonazzi V, di Tommaso M *et al.*: The Celera Discovery System. *Nucleic Acids Res.* 30, 129-136 (2002).
57. Venter JC: Plenary lecture, RECOMB2002 Conference, Washington DC, April 2002.

Websites

101. https://cluster.tu-graz.ac.at/srs/
SRS at Graz University of Technology
102. www.beowulf.org/
BeoWulf
103. www.mosix.org
MOSIX
104. http://.biogrid.icm.edu.pl/
BioGrid
105. www.people.virginia.edu/~wrp/pearson.html
Fasta
106. http://bimas.dcrt.nih.gov/clustalw/clustalw.html
ClustalW
107. www.sgi.com/industries/sciences/chembio/resources/clustalw/parallel_clustalw.html,
Parallel ClustalW
108. http://bioinfo.pbi.nrc.ca/clustalw-smp/
ClustalW SMP
109. http://hmmer.wustl.edu/
HMMER
110. www.ncbi.nlm.nih.gov/BLAST/
NCBI-Blast
111. http://blast.wustl.edu/
WU-Blast
112. www3.informatik.tu-muenchen.de/~zimmerms/ppmake
Ppmake
113. http://biowulf.nih.gov/blast.html
Multiblast
114. http://genome.tugraz.at/Software/MPI-Swarm/
MPI-Swarm
115. www.atmforum.org/
ATM Forum
116. www.ei.ac.uk/ArrayExpress
ArrayExpress
117. www.mged.org
Microarray Gene Expression Data Society
118. www.geneontology.org
Gene Ontology Consortium
119. http://standards.ieee.org/getieee802/
IEEE 802.x Standards
120. www.fibrechannel.com/
Fibre Channel Industry Association
121. www.ietf.org/
Internet Engineering Task Force
122. www.snia.org/
Storage Networking Industry Association
123. www.webopedia.com/
Online dictionary for computer and Internet technology
124. www.idc.org/
Interntional Data Corporation (IDC).

BIOINFORMATICS APPLICATIONS NOTE Vol. 20 no. 5 2004, pages 805–807
DOI: 10.1093/bioinformatics/bth014

ClusterControl: a web interface for distributing and monitoring bioinformatics applications on a Linux cluster

*Gernot Stocker, Dietmar Rieder and Zlatko Trajanoski**

Institute of Biomedical Engineering and Christian-Doppler-Laboratory for Genomics and Bioinformatics, Graz University of Technology, Krenngasse 37, 8010 Graz, Austria

Received on June 2, 2003; accepted on August 19, 2003
Advance Access publication January 29, 2004

ABSTRACT

Summary: ClusterControl is a web interface to simplify distributing and monitoring bioinformatics applications on Linux cluster systems. We have developed a modular concept that enables integration of command line oriented program into the application framework of ClusterControl. The systems facilitate integration of different applications accessed through one interface and executed on a distributed cluster system. The package is based on freely available technologies like Apache as web server, PHP as server-side scripting language and OpenPBS as queuing system and is available free of charge for academic and non-profit institutions.

Availability: http://genome.tugraz.at/Software/ClusterControl

Contact: zlatko.trajanoski@tugraz.at

INTRODUCTION

With the introduction of high-throughput technologies, e.g. sequencing and microarrays for expression profiling the amount of data that has to be stored, managed and analyzed increased dramatically. Very soon, it became necessary to implement computer-intensive bioinformatics applications for analyzing the flood of data on multi-processor computing systems. As most of these applications are not parallelized, they have to be distributed and monitored using a queuing system. Several web interfaces (Jenuth, 2000; Ferlanti *et al*., 1999; Stoesser *et al*., 2003; Blanchet *et al*., 2000) and more generalized approaches (Letondal, 2001) for universal generation of web interfaces based on textual xml descriptions have been developed previously to enable delivery of calculations on a computing server. However, since these web interfaces were developed for execution on the same server on which the web server is installed, they are not able to use efficiently the power of computing cluster systems.

The first approach to combine distributed calculations using a low-cost PC computing cluster and a web server has been implemented by the BeoBLAST (Grant *et al*., 2001) project. However, this system is limited to the BLAST (Altschul *et al*., 1990) applications and there is currently no freely available tool, which enables integration of other command line oriented programs like HMMer (Durbin *et al*., 1998) or FASTA (Deshpande *et al*., 1995). Therefore, the objective of this work was to develop a platform-independent web interface for distributing and monitoring bioinformatics applications on PC-based cluster systems.

PROGRAM OVERVIEW

The modular framework of this system enables integration of every command line-driven tool (Fig. 1). By adding a web-form (that includes all required parameters of the program which should be integrated) and an additional PHP file (that assembles the appropriate command line), the framework can be extended by any user with moderate programming skills. The delivery to the queuing system and collection of the results are managed by the framework in the background. Hereby, single process oriented applications are supported as well as real parallel applications programmed with special communication libraries like MessagePassingInterface (MPICH). After obtaining the results, it is also possible to run a post-processing tool to improve the visualization of the result files. An example for an integrated application of this framework is the existing NCBI-Blast module that can be tested on http://genome.tugraz.at/Software/ClusterControl/Demo. Currently, the list of supported application modules contains NCBI-Blast, Fasta, WU-Blast and HMMer. Customizations of existing modules can be easily done by using simple but well-defined interface classes and procedures. Unsupported applications can be added immediately and can extend the functionality of ClusterControl by following the short instructions on how to write new modules.

For time-consuming applications, the user can log out from the web interface and can login later to monitor the status of the calculation. If the calculation is still running, an automatically refreshing web page will show the current status

*To whom correspondence should be addressed.

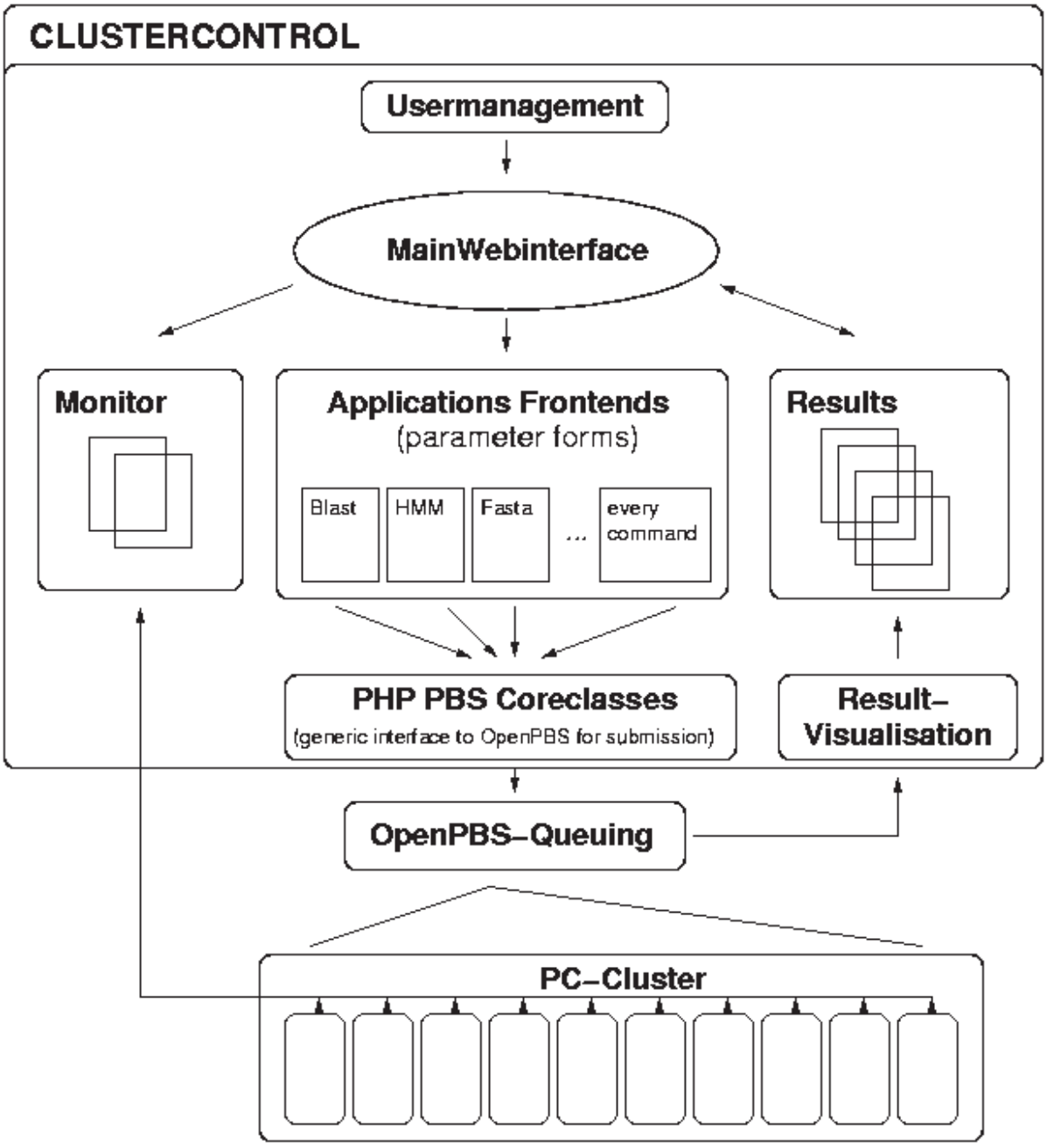


Fig. 1. Schematic overview of ClusterControl. The user can submit jobs through the applications front-ends and monitor the current status of the cluster. After submission, the job will be distributed through the backend PHP classes and the OpenPBS queuing system on the cluster. The job also appears in the Results section and shows its current status. After calculation, a visual improvement of the results can be performed and the final result will be shown in the Results section.

of this submitted job. Otherwise, the processed result will be presented immediately to the user.

An additional feature of this web interface is the ability to monitor all cluster nodes attached to this cluster system. To provide the real-time status and functionality of every calculation node, a lightweight program is installed on every calculation node within the cluster. For the transfer of this status information to the web interface a text protocol was defined, which allows request for information, such as system-, processor-, memory-load, etc.

For the management of the users two different systems are supported: a Lightweight Data Application Protocol (LDAP)-Server or an Unix-like password file that can be used as source of user-accounts. Thus, local cluster user and virtual user

stored in the password file can access the cluster system simultaneously without having direct access to the data, sequences, etc. of each other.

The object-oriented application programming interface (API) of ClusterControl can also be used in PHP scripts for farming out program execution outside the web server context. Similar to the Mollusc-API (Hokamp *et al*., 2003), which is programmed in Perl and wraps up distributed program execution, the backend library of ClusterControl can be used to submit calculation jobs from a PHP script run in a regular Unix shell.

INSTALLATION

For local installation, ClusterControl requires a running installation of the web server Apache (http://www.apache.org) with PHP (Version 4) and LDAP support, that comes with nearly every Linux distribution and a local installation of the queuing system OpenPortableBatchSystem (PBS), which can be downloaded from www.openpbs.org. If an LDAP-Server is already available for user management of the cluster, it can be easily integrated by modifying the configuration file. To activate the monitoring mechanism of ClusterControl, a status collecting server-process must be started on every cluster node. With the adaptation of one well-documented configuration file the installation process is completed. This web application is freely available and can be downloaded from http://genome.tugraz.at/Software/ClusterControl. It is licensed under the Gnu General Public License (GPL; http://www.gnu.org/licenses/gpl.html).

FUTURE DEVELOPMENT

Future development of ClusterControl will provide extensions of the already existing application set, improvements in integration of features of OpenPBS for job management and extension to other queuing systems. Beside already supported MPI implementations, some parallel bioinformatics applications are still using the parallelization software PVM. Therefore, support for this platform is also in preparation. Additional enhancements in the form of new modules by the scientific user community are encouraged and will be integrated in the open source distribution of ClusterControl.

ACKNOWLEDGEMENT

This work was supported by a grant Bioinformatics Integration Network from the Austrian GEN-AU Programme.

REFERENCES

Jenuth,J.P. (2000) The NCBI. Publicly available tools and resources on the Web. *Methods Mol. Biol.*, **132**, 301–312.

Ferlanti,E.S., Ryan,J.F., Makalowska,I. and Baxevanis,A.D. (1999) WebBLAST 2.0: an integrated solution for organizing and analyzing sequence data. *Bioinformatics*, **15**, 422–423.

Stoesser,G., Baker,W., van den Broek,A., Garcia-Pastor,M., Kanz,C., Kulikova,T., Leinonen,R., Lin,Q., Lombard,V., Lopez,R. *et al*. (2003) The EMBL Nucleotide Sequence Database: major new developments. *Nucleic Acids Res.*, **31**, 17–22.

Blanchet,C., Combet,C., Geourjon,C. and Deleage,G. (2000) MPSA: integrated system for multiple protein sequence analysis with client/server capabilities. *Bioinformatics*, **16**, 286–287.

Letondal,C. (2001) A Web interface generator for molecular biology programs in Unix. *Bioinformatics*, **17**, 73–82.

Grant,J.D., Dunbrack,R.L., Manion,F.J. and Ochs.M.F. (2001) BeoBLAST: distributed BLAST and PSI-BLAST on a Beowulf cluster. *Bioinformatics*, **18**, 765–766.

Altschul,S.F., Gish,W., Miller,W., Myers,E.W. and Lipman,D.J. (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.

Durbin,R., Eddy,S., Krogh,A. and Mitchison,G. (1998) *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge.

Deshpande,A.S., Richards,D.S. and Pearson,W.R. (1995) Searching protein sequence libraries: comparison of the sensitivity and selectivity of the Smith–Waterman and FASTA algorithms. *Protein Sci.*, **4**, 1145–1160.

Hokamp,K., Shields,D.C., Wolfe,K.H. and Caffrey,D.R. (2003). Wrapping up BLAST and other applications on Unix clusters. *Bioinformatics*, **19**, 441–442.

BMC Genomics

BioMed Central

Database

Open Access

GOLD.db: genomics of lipid-associated disorders database

Hubert Hackl, Michael Maurer, Bernhard Mlecnik, Jürgen Hartler, Gernot Stocker, Diego Miranda-Saavedra and Zlatko Trajanoski*

Address: Institute for Genomics and Bioinformatics and Christian-Doppler-Laboratory for Genomics and Bioinformatics, Graz University of Technology, Petersgasse 14, 8010 Graz, Austria

Email: Hubert Hackl - hubert.hackl@tugraz.at; Michael Maurer - michael.maurer@tugraz.at; Bernhard Mlecnik - bernhard.mlecnik@tugraz.at; Jürgen Hartler - juergen.hartler@tugraz.at; Gernot Stocker - gernot.stocker@tugraz.at; Diego Miranda-Saavedra - diego@compbio.dundee.ac.uk; Zlatko Trajanoski* - zlatko.trajanoski@tugraz.at

* Corresponding author

Published: 10 December 2004

BMC Genomics 2004, **5**:93 doi:10.1186/1471-2164-5-93

Received: 06 September 2004

Accepted: 10 December 2004

This article is available from: http://www.biomedcentral.com/1471-2164/5/93

Abstract

Background: The GOLD.db (Genomics of Lipid-Associated Disorders Database) was developed to address the need for integrating disparate information on the function and properties of genes and their products that are particularly relevant to the biology, diagnosis management, treatment, and prevention of lipid-associated disorders.

Description: The GOLD.db http://gold.tugraz.at provides a reference for pathways and information about the relevant genes and proteins in an efficiently organized way. The main focus was to provide biological pathways with image maps and visual pathway information for lipid metabolism and obesity-related research. This database provides also the possibility to map gene expression data individually to each pathway. Gene expression at different experimental conditions can be viewed sequentially in context of the pathway. Related large scale gene expression data sets were provided and can be searched for specific genes to integrate information regarding their expression levels in different studies and conditions. Analytic and data mining tools, reagents, protocols, references, and links to relevant genomic resources were included in the database. Finally, the usability of the database was demonstrated using an example about the regulation of Pten mRNA during adipocyte differentiation in the context of relevant pathways.

Conclusions: The GOLD.db will be a valuable tool that allow researchers to efficiently analyze patterns of gene expression and to display them in a variety of useful and informative ways, allowing outside researchers to perform queries pertaining to gene expression results in the context of biological processes and pathways.

Background

The excessive consumption of high calorie, high fat diets and the adoption of a sedentary life style have made obesity and atherosclerosis major health problems in Western societies. In the USA, over 50% of the population are overweight (BMI > 25) and close to 25% are considered obese (BMI > 30) [1,2]. As a consequence, a large fraction of the population is at risk to develop a broad range of common, life-threatening diseases including non-insulin dependent diabetes, various hyperlipidemias, high blood pressure and atherosclerosis. Vascular disease including coronary heart disease and stroke is currently the major cause of

death in the United States and in other industrialized nations.

At the root of obesity and atherosclerosis is an excessive deposition of neutral lipids. Adipose tissue accumulates predominantly triglycerides, whereas macrophages along the blood vessel wall mainly accumulate cholesterol and cholesteryl esters. Accordingly, a detailed understanding of the molecular mechanisms that govern the balance between lipid deposition and mobilization is fundamentally important for the prevention and improved treatment of disease. In addition to the apparent environmental components involved in the pathogenesis of disorders related to lipid and energy metabolism, a large number of studies have provided undisputed evidence that susceptibility genes contribute around 50% of the phenotype. These genes encode products involved in the cellular uptake, synthesis, deposition and/or mobilization of lipids. However, characterization of many if not most of these genes and their products remains rudimentary. Deficiencies in the current level of understanding extend to key enzymes such as important triglyceride hydrolases in adipose tissue [3] or cholesteryl ester hydrolases in macrophages, hormones, signal transduction pathways, and the regulation of the transcription of relevant genes.

While medical molecular biology traditionally associates single genes and gene products with diseases, a growing body of evidence suggests that several common disease phenotypes arise from the delicate interaction of many genes as well as gene-environment interactions. To elucidate the development of obesity and atherosclerosis, it will be necessary to analyze patterns of gene expression and relate them to various metabolic states. To discover novel genes, processes and pathways that regulate lipid deposition and mobilization, a departure from hypothesis-driven research and turn to a discovery-driven approach is necessary. The application of high-throughput technologies and genome-based analysis will provide the tools for the analysis of gene-gene and gene-environment interactions in a systematic and comprehensive manner.

To facilitate genomic research we have initiated the development of a system for storing, integrating, and analyzing relevant data needed to decipher the molecular anatomy of lipid associated disorders. In order to provide a reference for pathways and information of the relevant genes and proteins in an efficiently organized way, we have created the Genomics Of Lipid-Associated Disorders database (GOLD.db). The GOLD.db integrates disparate information on the function and properties of genes and their protein products that are particularly relevant to the biology, diagnosis management, treatment, and prevention of lipid-associated disorders.

Construction and content

The main goal of the GOLD.db was to provide biological pathways with image maps and visual pathway information. For each element in the pathway, specific information exists including structured information about a gene, protein, function, literature, and links. The GOLD.db provides also the possibility to map gene expression data individually to each pathway. Additionally, analytic and data mining tools, reagents, protocols, references, and links to relevant genomic resources were included in the database.

The GOLD.db was implemented in Java technology [4]. Hence, the pathway editor, as well the web application are platform independent. The web application of GOLD.db is build in Java Servlets and JavaServer Pages technology based on the Model-View-Controller Architecture [5]. For the implementation, the freely available struts framework [6] was used. This code can be easily deployed in any Servlet Container. We used the Servlet Container Tomcat (also freely available at [7]) which is accessible from all web browsers. Oracle 9i was used as database management system. The interface between the Java and the Database management system was established using Java database connectivity (JDBC) 2.0. Therefore, migration to other freely available DBMSs like mySQL can be easily done. For additional storage and communication between the pathway-editor components, the markup language XML containing structured, human readable information, was used. The provided pathways can be downloaded as Scalable Vector Graphics (SVG) [8], a standard for describing two-dimensional graphics in XML, and can be visualized in this format on the client side with the web browser using a plug-in for SVG.

For tracking the repository of the reagents like clone resources and libraries which can be used for microarray studies, we have developed a relational database. Information about the vector, the sequence and length of the clone insert, primers for the PCR amplification, tissue, organism, accession number, library, container, storage information, date and person and access to other clone bases (e.g. IMAGE Consortium) can be stored. Users of the GOLD.db can list these clones and get all the information about each available clone. With restricted access, clone information or even clone lists can be uploaded and selection lists can be created and deleted. The input mask is designed in such way that the user can choose one of the elements of the created selection lists.

In order to deal with the huge amount of data associated with large scale studies and to perform sequence based

and microarray analysis, several bioinformatic tools were integrated or can be downloaded. Sequence similarity search against databases can be performed with BLAST (Basic Local Alignment Search Tool) [9], FASTA [10] or HMM (Hiden Markov Models) [11] on a 50 CPU Myrinet Cluster. The sequence retrieval system SRS (LION Bioscience AG, Heidelberg, Germany) was included to enable rapid, easy and user friendly access to the large volumes of diverse and heterogeneous data [12]. The latest version of the PathwayEditor for the construction of biological pathway diagrams can be downloaded. For microarray analysis the platform independent JAVA tools ArrayNorm [13] for normalization of microarray data and Genesis [14] for clustering and analysis of large scale gene expression datasets were made available.

Utility and discussion

Pathways

In order to construct the biological pathways of interest, we have developed a pathway editor [15] and an extended version to map gene expression data (pathway mapper). This drawing tool provides the possibility to draw elements – typically representing a gene as part of the pathway – and the connection between those elements. The benefit of this tool is that information can be appended to each element via an input mask. This information can be accessed by clicking on the corresponding element in the image map within the pathway mapper or when saved and uploaded via the web interface to the GOLD.db. To design this pathway service as flexible as possible, features are provided for the remove, up- and download of relevant pathways (image maps) including the underlying additional information of the elements. However, this service is on a restricted basis to prohibit unauthorized access. Since some pathways tend to become very detailed an option to search for genes or gene accession number, respectively, within the pathways was built in. The pathway editor is executable as a standalone application and is available from [16]. Currently annotated pathways are the insulin signaling pathway, the IGF-1 pathway and the adipogenesis regulatory network. Other pathways of lipid metabolism will follow in the near future. Available KEGG pathways can also be adapted with the pathway editor based on the provided XML files [17] and uploaded in the same way. All relevant KEGG pathways for different organisms are provided. Moreover, pathways from BioCarta were made available within the GOLD.db and HTML files [18] were parsed to provide additional meta-information of the pathway elements.

For each element in the pathways a specific information field exists. The field includes structured information about a gene, protein, function, literature, and links to well-curated and annotated databases. Besides the gene name and the symbol name – for human the HUGO symbols and gene names and for mouse the MGI nomenclature were used – RefSeq numbers for the transcript and the protein as well as a link to SwissProt/UniProt and LocusLink is available. For the elements of the KEGG pathways a link to the provided enzyme or product information was given. The description, localization and classification of the factors are entered by the annotator in plain text and are accessed in the same format. The references used to generate the content of the database entries can be appended, including a link to the PubMed entry. There is also the possibility to create a list of reference entries for the pathway. If a clone for a specific gene is available in the clone resources, the clone name will be displayed automatically and a link with optional information about this clone is provided.

Mapping of gene expression data sets to pathways

Through the integration of several types of biological information deeper insights into the molecular mechanisms and biological processes can be gained than just by the analysis of one type of experimental results. In the GOLD.db it is possible to map gene expression data (for instance results of microarray studies) to the corresponding elements of the available pathways similar to previous efforts [19]. Either an individual or a provided gene expression data set can be used to visualize the gene expression at different experimental conditions sequentially or all at once in the context of a pathway. If an element (gene) of the pathway is included in the data set, the related symbol in the image map is color coded according to the relative gene expression or the log ratio in two color microarray experiments, respectively.

As key for the mapped relation the RefSeq number [20] is used. Hence, only those elements in the data set file are mapped, where the RefSeq number in the data set is specified. For the KEGG pathways each element classified by the enzyme classification number (EC) is virtually subdivided into different corresponding RefSeq entries, since one EC is represented by one or more RefSeq entries.

Curated gene expression data sets

Analysis of gene expression patterns in animal and cell models for lipid-associated disorders will help to understand the fundamental gene relations and regulatory mechanisms responsible for the development of obesity related diseases. The huge amount of data associated with the analysis of large scale gene expression analysis raises the demand of tools for storing, processing and retrieving complex information. Although a number of studies have been published and despite the requirements of some journals to deposit microarray data in public databases like GEO http://www.ncbi.nlm.nih.gov/geo/ or ArrayExpress http://www.ebi.ac.uk/arrayexpress/, it is still very difficult for researchers to obtain the original data. Web

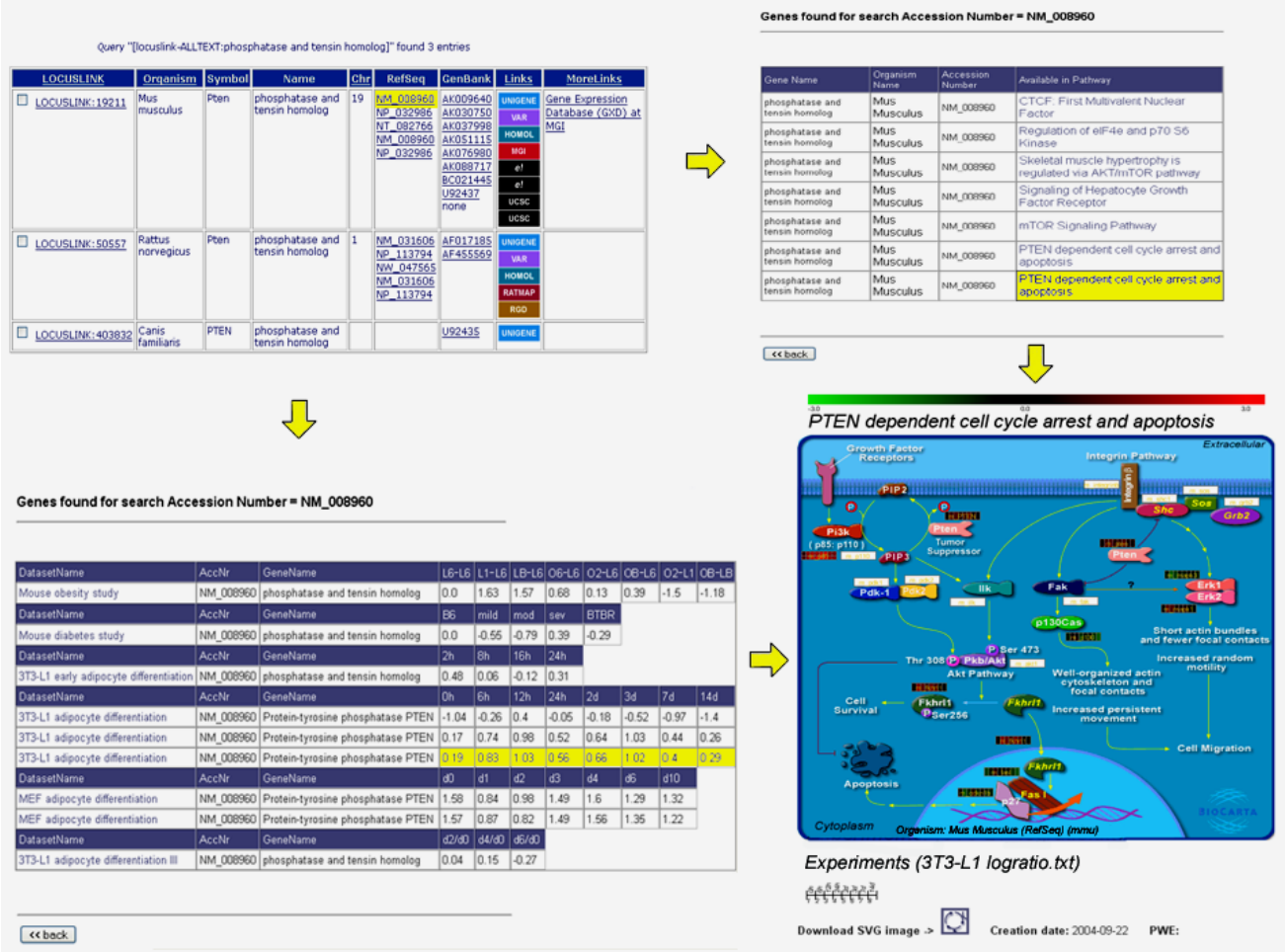


Figure 1
Various result tables from using GOLD.db to address the question how is PTEN regulated during adipocyte differentiation (*top left:* result of search in SRS for phosphatase and tensin homolog; *top right:* pathways, in which PTEN is involved; *bottom left:* relative gene expression levels of PTEN in different datasets; *bottom right:* PTEN dependent cell cyle pathway with mapped gene expression levels)

sites with Supplementary information are not maintained and/or not further developed. Hence, a database with a large collection of curated datasets will be enormously valuable for the community. Approaches to upload and retrieve gene expression data were pursued within the GOLD.db. Large scale gene expression data sets can be uploaded in form of tab delimited text files (Stanford file format) [21] as used for cluster analysis programs together with additional information about the experimental conditions and the citation for already published data sets. Within those data sets the search for specific genes is possible to provide integrated visualization of gene expression levels in different studies and experimental conditions.

Example for using GOLD.db: regulation of Pten during adipocyte differentiation

Recently, it was shown that insulin sensitivity, energy expenditure, and thermogenesis were enhanced in adipose-specific Pten-deficient (AdipoPten-KO) mice. Body and adipose tissues weight in these mice were significantly lower than those of control mice in spite of a larger food intake [22]. We addressed the question how is the expression of the Pten gene regulated during adipocyte differentiation in different models and experimental setups and in which pathways is PTEN involved. The workflow for the analysis is described in Figure 1. Pten (phosphatase and tensin homolog deleted on chromosome 10) is known as tumor suppressor gene and is a protein and lipid phos-

phatase with the major substrate phosphatiylinositol 3,4,5-triphosphate (PIP3), as indicated in the annotated insulin signaling pathway within the GOLD.db. In fact, Pten regulates negatively the insulin signaling pathway in 3T3-L1 adipocytes [23].

During adipocyte differentiation cyclin dependent kinase inhibitors, like p21 leads to a hypophosphorylation of the Retinoblastoma protein (Rb) which allows binding to the E2F transcription factor, causing cells to permanently exit the cell cycle – a required step in adipocyte differentiation called mitotic clonal expansion – before entering the terminal differentiation state. pRb interacts physically with adipogenic CCAAT/enhancer-binding proteins and positively regulates transactivation by C/EBPβ and therefore plays a pivotal role in adipocyte differentiation [24,25]. Hence, since a) PTEN is expressed during adipogenesis (Figure 1), b) is involved in the regulation of Rb [22], a major player in adipogenesis, and c) is an important component in cell cycle arrest and apoptosis (Figure 1), it can be postulated that PTEN plays an important role in fat cell development.

Thus, using recently identified key player for food intake and weight control and using the GOLD.db, it is possible to address relevant questions and generate testable hypotheses on the molecular mechanisms of fat cell development.

Conclusions

The vast quantity of gene expression data generated in genomic studies presents a number of challenges for their effective analysis and interpretation. In order to fully understand the changes in expression that will be observed, we must correlate these data with phenotype, genotype, metabolism and other information including the tissue distribution and time course expression data gleaned from previous studies. The goal of our work was the development of a specialized database and tools that allow researchers to efficiently analyze patterns of gene expression and to display them in a variety of useful and informative ways, allowing outside researchers to perform queries pertaining to gene expression results in the context of biological processes and pathways. The uniqueness of the GOLDdb database we have developed is threefold: 1) the inclusion of annotated pathways, 2) the availability of curated datasets and 3) the possibility to map experimental data on biological pathways. The upcoming challenges will be to include data from functional analysis and proteomics data, which will give us new opportunities in understanding mechanisms of different applications and lipid-associated disorders in particular.

Availability and requirements

The GOLD.db database should be cited with the present publication as a reference. Access to GOLD.db is possible through the world wide web at http://gold.tugraz.at. The pathway editor and the clone tracker are available free of charge to academic, government, and other nonprofit institutions.

Author's contributions

HH was responsible for the content, the annotation process, webdesign, and processing of data sets. MM was responsible for the implementation of the database and web application as well as the relational database for the clone tracker. BM and JH had implemented the mapping of expression data to pathways. GS is involved in providing of sequence analysis tools and server software. DMS has annotated the insulin signaling pathway. ZT was responsible for the design of the study and for overall project coordination.

Acknowledgements

This work was supported by the Austrian Science Fund, Project SFB Biomembranes F718, the GEN-AU projects Bioinformatics Integration Network (BIN) and Genomics of Lipid-Associated Disorders (GOLD). Diego Miranda-Saavedra was supported by an EU Marie Curie Training Site program "Genomics of Lipid Metabolism". Michael Maurer was supported by a grant from the Austrian Academy of Sciences.

References

1. Flegal KM, Carroll MD, Kuczmarski RJ, Johnson CL: **Overweight and obesity in the United States: prevalence and trends, 1960-1994.** *Int J Obes Relat Metab Disord* 1998, **22:**39-47.
2. Must A, Spadano J, Coakley EH, Field AE, Colditz G, Dietz WH: **The disease burden associated with overweight and obesity.** *JAMA* 1999, **282:**1523-1529.
3. Zechner R, Strauss J, Frank S, Wagner E, Hofmann W, Kratky D, Hiden M, Levak-Frank S: **The role of lipoprotein lipase in adipose tissue development and metabolism.** *Int J Obes Relat Metab Disord* 2000, **24:**S53-S56.
4. **Java Technology** 2004 [http://java.sun.com].
5. Gamma E, Helm R, Johnson R, Vlissides J: *Design Patterns - Elements of Reusable Object-Oriented Software* Addison-Wesley; 1995.
6. **Struts framework** 2004 [http://jakarta.apache.org/struts/].
7. **Tomcat** 2004 [http://jakarta.apache.org/tomcat/].
8. **SVG** 2004 [http://www.w3.org/TR/SVG].
9. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ: **Basic local alignment search tool.** *J Mol Biol* 1990, **215:**403-410.
10. Pearson WR, Lipman DJ: **Improved tools for biological sequence comparison.** *Proc Natl Acad Sci U S A* 1988, **85:**2444-2448.
11. Eddy SR: **Profile hidden Markov models.** *Bioinformatics* 1998, **14:**755-763.
12. Etzold T, Ulyanov A, Argos P: **SRS: information retrieval system for molecular biology data banks.** *Methods Enzymol* 1996, **266:**114-128.
13. Pieler R, Sanchez-Cabo F, Hackl H, Thallinger GG, Trajanoski Z: **ArrayNorm: comprehensive normalization and analysis of microarray data.** *Bioinformatics* 2004, **20:**1971-3.
14. Sturn A, Quackenbush J, Trajanoski Z: **Genesis: cluster analysis of microarray data.** *Bioinformatics* 2002, **18:**207-208.
15. Trost E, Hackl H, Maurer M, Trajanoski Z: **Java editor for biological pathways.** *Bioinformatics* 2003, **19:**786-787.
16. **Institute for Genomics and Bioinformatics, Graz University of Technology** 2004 [http://genome.tugraz.at].
17. Kanehisa M, Goto S, Kawashima S, Nakaya A: **The KEGG databases at GenomeNet.** *Nucleic Acids Res* 2002, **30:**42-46.

18. **Biocarta Pathways** 2004 [http://biocarta.com/genes/allPathways.asp].
19. Dahlquist KD, Salomonis N, Vranizan K, Lawlor SC, Conklin BR: **GenMAPP, a new tool for viewing and analyzing microarray data on biological pathways.** *Nat Genet* 2002, **31:**19-20.
20. Pruitt KD, Maglott DR: **RefSeq and LocusLink: NCBI gene-centerd resources.** *Nucleic Acids Res* 2001, **29:**137-140.
21. Eisen MB, Spellman PT, Brown PO, Botstein D: **Cluster analysis and display of genome-wide expression patterns.** *Proc Natl Acad Sci U S A* 1998, **95:**14863-14868.
22. Komazawa N, Matsuda M, Kondoh G, Mizunoya W, Iwaki M, Takagi T, Sumikawa Y, Inoue K, Suzuki A, Mak TW, Nakano T, Fushiki T, Takeda J, Shimomura I: **Enhanced insulin sensitivity, energy expenditure and thermogenesis in adipose-specific Pten suppression in mice 2.** *Nat Med* 2004, **10:**1208-1215.
23. Nakashima N, Sharma PM, Imamura T, Bookstein R, Olefsky JM: **The tumor suppressor PTEN negatively regulates insulin signaling in 3T3-L1 adipocytes.** *J Biol Chem* 2000, **275:**12889-12895.
24. Hansen JB, Petersen RK, Larsen BM, Bartkova J, Alsner J, Kristiansen K: **Activation of peroxisome proliferator-activated receptor gamma bypasses the function of the retinoblastoma protein in adipocyte differentiation.** *J Biol Chem* 1999, **274:**2386-2393.
25. Chen PL, Riley DJ, Chen Y, Lee WH: **Retinoblastoma protein positively regulates terminal adipocyte differentiation through direct interaction with C/EBPs.** *Genes Dev* 1996, **10:**2794-2804.

BMC Bioinformatics

BioMed Central

Software

Open Access

MARS: Microarray analysis, retrieval, and storage system

Michael Maurer†, Robert Molidor†, Alexander Sturn†, Juergen Hartler, Hubert Hackl, Gernot Stocker, Andreas Prokesch, Marcel Scheideler and Zlatko Trajanoski*

Address: Institute for Genomics and Bioinformatics and Christian Doppler Laboratory for Genomics and Bioinformatics, Graz University of Technology, Petersgasse 14, 8010 Graz, Austria

Email: Michael Maurer - michael.maurer@gmail.com; Robert Molidor - robert.molidor@tugraz.at; Alexander Sturn - alexander.sturn@tugraz.at; Juergen Hartler - juergen.hartler@tugraz.at; Hubert Hackl - hubert.hackl@tugraz.at; Gernot Stocker - gernot.stocker@tugraz.at; Andreas Prokesch - andreas.prokesch@tugraz.at; Marcel Scheideler - marcel.scheideler@tugraz.at; Zlatko Trajanoski* - zlatko.trajanoski@tugraz.at

* Corresponding author †Equal contributors

Published: 18 April 2005

BMC Bioinformatics 2005, **6**:101 doi:10.1186/1471-2105-6-101

Received: 03 February 2005

Accepted: 18 April 2005

This article is available from: http://www.biomedcentral.com/1471-2105/6/101

Abstract

Background: Microarray analysis has become a widely used technique for the study of gene-expression patterns on a genomic scale. As more and more laboratories are adopting microarray technology, there is a need for powerful and easy to use microarray databases facilitating array fabrication, labeling, hybridization, and data analysis. The wealth of data generated by this high throughput approach renders adequate database and analysis tools crucial for the pursuit of insights into the transcriptomic behavior of cells.

Results: MARS (Microarray Analysis and Retrieval System) provides a comprehensive MIAME supportive suite for storing, retrieving, and analyzing multi color microarray data. The system comprises a laboratory information management system (LIMS), a quality control management, as well as a sophisticated user management system. MARS is fully integrated into an analytical pipeline of microarray image analysis, normalization, gene expression clustering, and mapping of gene expression data onto biological pathways. The incorporation of ontologies and the use of MAGE-ML enables an export of studies stored in MARS to public repositories and other databases accepting these documents.

Conclusion: We have developed an integrated system tailored to serve the specific needs of microarray based research projects using a unique fusion of Web based and standalone applications connected to the latest J2EE application server technology. The presented system is freely available for academic and non-profit institutions. More information can be found at http://genome.tugraz.at.

Background

Microarray analysis has become a widely used technique for the study of gene-expression patterns on a genomic scale [1,2]. Oligonucleotide and cDNA arrays have been utilized to study mRNA [3] and protein levels [4], to decipher protein-DNA interactions [5], to analyze the DNA copy number [6], to detect methylated sequences [7], and to analyze gene phenotypes in living mammalian cells [8]. Microarrays represent a very complex, multi step technique involving array fabrication, labeling, hybridization,

and data analysis. Currently, most laboratories are using either one labeled sample (Affymetrix microarrays) or two labeled samples (cDNA microarrays) for hybridizations, but several applications have been established were three color microarrays are used [9,10]. State-of-the-art microarrays can have from several hundred up to tens of thousands of elements annotated by dozens of parameters. Information on details of the bench work, typically kept in lab notebooks or scattered files, as well as information regarding spotting, reliable tracking of the spotted molecules, scanning, and image quantification settings, is important for the computational analysis and reproducibility of experiments. Every step generates a wealth of data spanning tens of megabytes and in each of them errors may occur or protocols might need optimization to improve results. Moreover, all these information must be archived according to accepted scientific standards, which allow scientists to share common information and to make valid comparisons among experiments. For this reason the Microarray Gene Expression Data Society (MGED) [11] is focusing on establishing standards for microarray data annotation and exchange, facilitating the creation of microarray databases and related software implementing these standards. MGED is heavily promoting the sharing of high quality, well annotated data within the life sciences community. Their initiatives – MIAME (Minimum Information About a Microarray Experiment) [12], MGED Ontology [13], and MAGE-ML (MicroArray Gene Expression Markup Language) [14] – maximize the value of microarray data by permitting greater opportunities for sharing information within scientific groups and thus for discovery. These will ultimately affect the description, analysis, and management of all high throughput biological data.

The 'list of genes' resulting from microarray analysis is not the end of a microarray experiment. The major challenge is to assign biological function and to generate new hypotheses. The simplest way to find genes of potential biological interest is to search the normalized data for the highly expressed ones. Additionally, identifying patterns of gene expression and grouping genes into expression classes can provide greater insight into their biological relevance. For this purpose several supervised or unsupervised clustering algorithms like support vector machines (SVM), hierarchical clustering, k-means, self organizing map (SOM), or principal component analysis (PCA) are in use. The annotation of genes or gene clusters can be achieved by mapping them to the Gene Ontology (GO) [15] in order to provide insights into relevant molecular functions, biological processes, and cellular components [16]. Another way to identify genes of biological interest is to map the normalized data or gene expression clusters [17] to known metabolic pathways as provided e.g. by KEGG [18] or BioCarta [19].

Several academic as well as commercial systems are available that address at least some of the needs such as laboratory information management systems (LIMS) [20], microarray databases [21-24] and repositories, normalization, clustering, pathway or GO mapping tools or expression analysis platforms [25]. However, freely available systems which integrate all the aspects mentioned above are rare and may lack important issues like usability, scalability, or standardized interfaces. Furthermore, for such integrated systems it is desireable to use a uniform and state-of-the-art software architecture in order to enhance setup, maintenance and further development.

We have therefore developed a Microarray Analysis and Retrieval System (MARS) using latest Java 2 Platform, Enterprise Edtition (J2EE) software technology. MARS provides modules mandatory for microarray databases:

• a laboratory information management system (LIMS) to keep track of information that accrues during the microarray production and biomaterial manipulation

• MAGE-ML export of data for depositing to public repositories e.g. ArrayExpress [26], GEO [27]

For these components already existing projects [21,23,26] have been evaluated. Their advantages as well as disadvantages have been taken into account for the design of MARS. Widely used concepts have been taken into consideration and accepted standard libraries like MAGE-STK [11]have been used whenever possible. Additionally, we extented this solid foundation and added novel features which can be highlighted as distinct advantages of the MARS system.

• a quality management application storing necessary quality control parameters indispensable for high-quality microarray data

• Web services to connect several well established tools such as normalization, clustering and pathway annotation applications

• applications for microarray normalization, gene expression clustering, and pathway exploration that are tightly integrated into the microarray analysis pipeline

• a novel, comprehensive, and Web based user management system to administrate institutes, groups, users, and their corresponding access rights

Implementation

Software architecture

MARS is based on a three tier architecture (Figure 1) using the Java 2 Platform, Enterprise Edition (J2EE), which

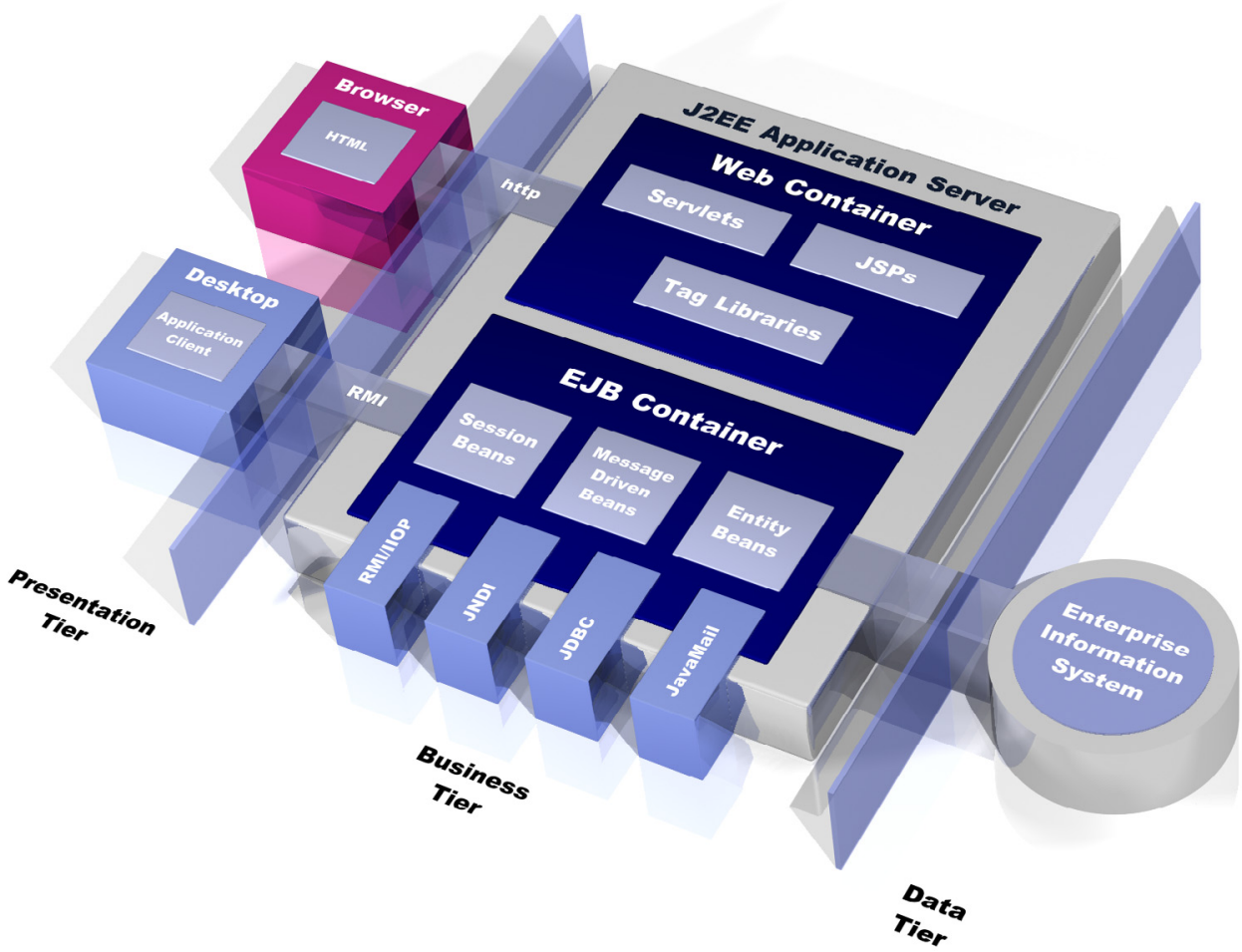


Figure 1
Three tier Java 2 Enterprise Edition software architecture. The J2EE platform simplifies the development of enterprise applications by providing standardized modular components like EJBs, JSP and Servlets. Furthermore it is providing a complete set of services to those components.

defines a standard for developing multi tier enterprise applications. The J2EE platform simplifies the development of enterprise applications using on standardized, modular components like Enterprise JavaBeans (EJB), Java Servlets, Java Server Pages (JSP), and XML technology.

A relational database (Oracle or PostgreSQL) builds the data- or Enterprise Information System tier. In the middle tier the J2EE compliant application server JBoss [28] is situated. It manages the access to the relational database as well as the interaction with the data. The Web server in conjunction with a servlet-container is responsible for the presentation tier. All the servlets and JSPs are executed to enable input and output of an application and to manage the applications workflow logic. An advantage of a multi tier architecture is that different tiers can be deployed to different servers, enabling load distribution as well as scalability.

Systems

The database schema, the business logic, and the Web interface can be subdivided into five major groups:

1. Microarray production

To address the needs of many laboratories which produce their own microarrays, MARS includes a generic array production LIMS. It manages data regarding the substances (clones) and their localization in microtiter plates, the array design spotted on the support, as well as single

arrays and array batches. The flexible and generic database design facilitates mapping of the steadily changing laboratory workflow. Additionally, each plate can be assigned to a library, which designates the organism and contains details about the cloning vector, forward and reverse primer and standard molecule annotations including gene name, accession number, UniGene number, and sequence. Substances stored in microtiter plates may undergo certain manipulations such as PCR amplification. Therefore a PCR amplification event can be assigned to a plasmid plate in order to generate a PCR plate in the database.

After entering the information necessary for spotting, a file is generated and prepared for download. This file is used by the spotting robot software to generate an array design file. After the spotting run has been completed, the array design file has to be uploaded into MARS. For each spotting run an array batch has to be created in MARS, and all slides spotted by this spotting run have to be assigned to this array batch. Additionally, important parameters regarding the spotting run such as temperature, duration, or humidity can be assigned to this array batch. Barcode tracking is employed for plates as well as for arrays to reduce possible input errors. Laboratories using commercial arrays have to upload the array design instead and define an array batch afterwards.

2. Sample preparation

Samples can be annotated in a user-customizable manner. MARS allows the annotation of biological descriptions such as the source and characteristics of a sample (e.g. tissue and disease), any genetic and chemical manipulation and stimulation. Performing such annotations in free text fields leads to large undefined vocabularies and makes them difficult to query. Thus, three different annotation types are provided: 1) enumeration enabling the usage of defined vocabularies or ontologies, 2) numbers to allow scoring and counting and 3) free text. Annotated samples will be linked to an extract, enabling a lab worker to annotate the extraction method, protocol, concentration, purity, and quantity. The labeled extract stores information on used extract quantity, the label and the labeling protocol.

3. Hybridization and raw data management

The hybridization page archives parameters regarding the hybridization tool and method and is linked to the used labeled extracts. In contrast to several other microarray databases MARS can handle any number of labeled extracts and thus allows the storage of multi color experiments. Resulting images from hybridized scanned slides can be uploaded to MARS and added to a hybridization record. It is noteworthy that a hybridization can have several image sets associated with images of different scanner settings. After analyzing the images several different raw datasets analyzed with different program settings can be uploaded and added to the appropriate image set.

4. Experiment annotation

A set of hybridizations forms an experiment. To store the experimental design these hybridizations can be divided into classes, paired, and flagged as a dyeswap hybridization. Additionally, an experiment can be annotated using MGED Ontology definitions (Figure 2) to specify the perturbational, methodological, and epidemiological design, as well as the biological properties. Transformed datasets can be added to classes and their corresponding raw dataset.

5. Quality management

To ensure high quality data and to allow the detection of possible sources of errors, a powerful quality management system has been integrated into MARS. This system is based on standard quality control procedures conducted during microarray production as well as during sample preparation, extraction and hybridization. In order to control the quality of PCR and purified PCR products generated during probe production, authorized users can upload gel images and analyze the bands according to a predefined schema (Figure 3). Based on this schema, PCR products can be identified later as a source of bad or missing spots on a slide. Quality annotation can be viewed by any user.

Slides can be scanned after fixation and/or after staining and parameters like spot walking or the number of missing spots are used to determine slide quality. In addition to array production quality controls, it is also necessary to check the quality of samples and its extracts. Data gained from an Agilent Bioanalyzer or gel images can be uploaded and analyzed either automatically (Bioanalyzer file) or manually (gel images) (Figure 4). Labeled extracts can be measured with a spectrophotometer to assess the efficiency of dye incorporation. Results of these measurements can be entered into MARS and the corresponding efficiency is calculated automatically. Finally, the quality of a hybridized slide is analyzed by extracting and displaying several statistical parameters from the raw data result file and by examining positive and negative controls printed onto a slide.

Data interfaces

One of the most important parts for the acceptance of a database is the data import interface. To allow the import of generic file formats, we have implemented a user definable parser that allows to read any tab delimited text file. The user has to define a file format where file columns are assigned to appropriate database fields. MARS allows to

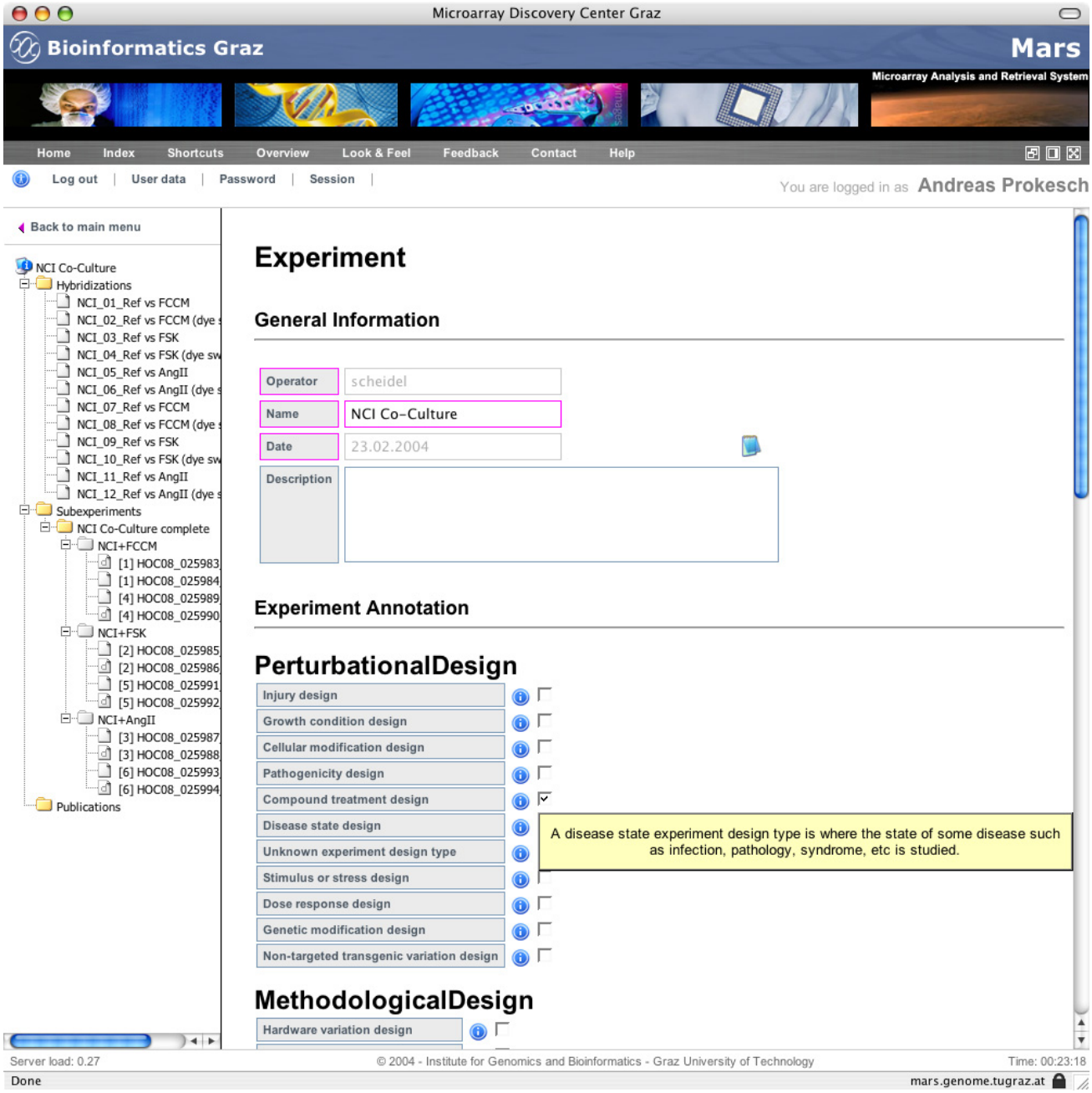


Figure 2
Experiment annotation. Web interface to define microarray experiments according to the MGED Ontology.

define file formats for importing plates, raw datasets, transformed datasets, and array designs.

Any file that has to be imported, linked, or used has to be uploaded to MARS at first. Afterwards these data can be analyzed by the users at their office desk without having to use another central storage system. Uploaded files are stored on the servers file system where MARS has been installed. Additionally, links to these files are maintained in the relational database to prevent the deletion of already imported, linked, or used files.

The implementation of other Web based applications and more important, the usage and correct linkage of their

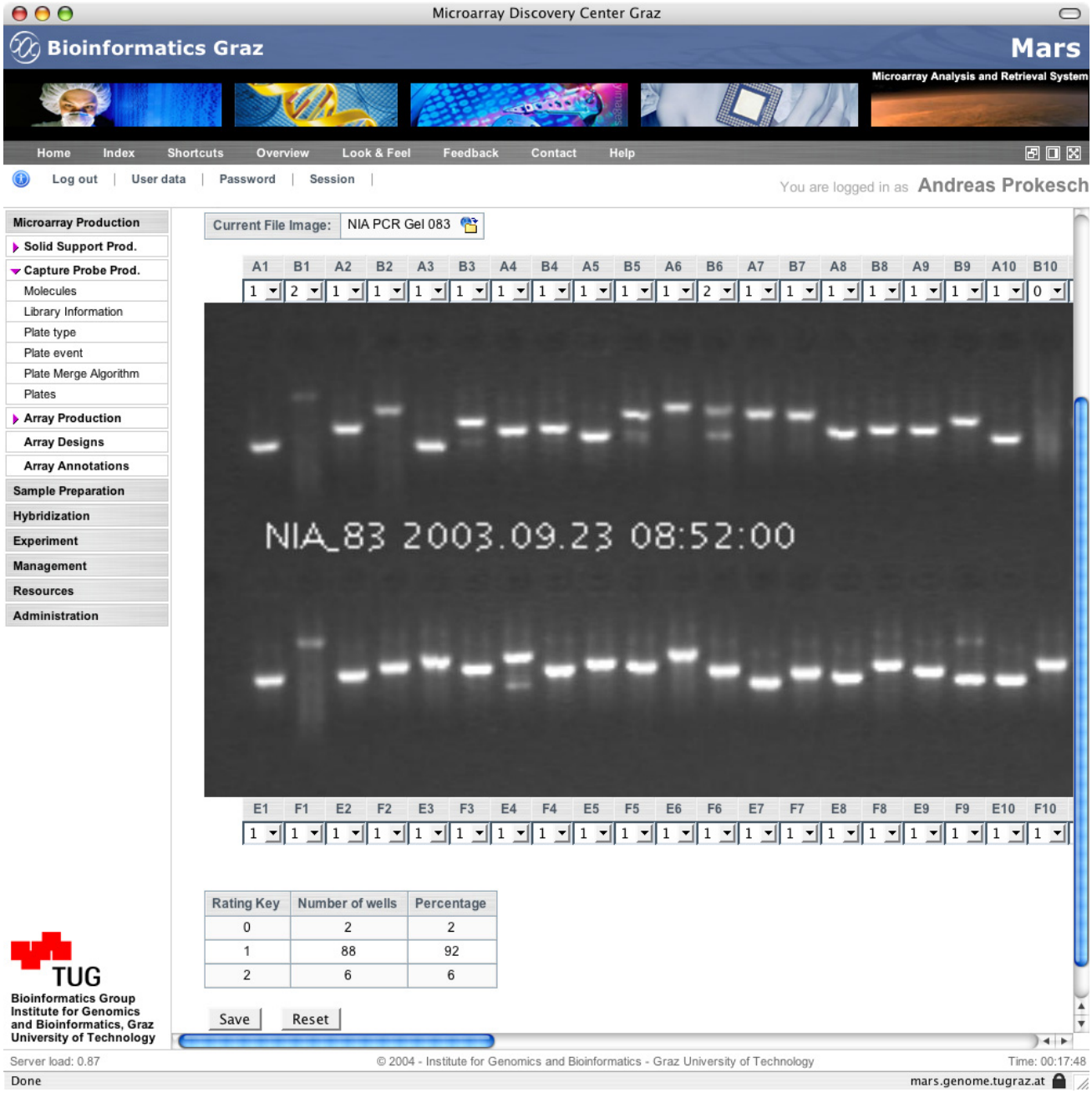


Figure 3
Quality control. A gel image from PCR products can be scored and associated to a plate.

stored data have been addressed by an External Application Connector Interface. Additional applications like supplementary quality checks can be added without any additional coding in MARS. The MARS user interface is dynamically displaying links to all former registered applications.

The Microarray Gene Expression Markup Language (MAGE-ML) has emerged as a language to describe and exchange information about microarray based experiments [29]. MAGE-ML is based on XML (eXtensible Markup Language) and can describe microarray designs, microarray manufacturing information, microarray

Figure 4
Quality control. Bioanalyzer analysis to check the RNA quality for a given RNA extract.

experiment setup and execution information, gene expression data, and data analysis results. By using the Java MAGE-STK (Mage Software Toolkit) [11] MARS is able to export samples, extracts, labeled extracts, arraydesigns, raw datasets, or whole experiments including several hybridizations.

Web service

In order to grant users access to MARS with software they are familiar with (e.g. BioConductor [30] or Matlab [31]), MARS provides a well defined Simple Object Access Protocol (SOAP) interface. SOAP is an XML-based communication protocol and encoding format for inter-application

communication. After minor software adaptions these interfaces allow to authenticate against MARS, to browse own and shared datasets, to download raw data, to filter the data, and to insert transformed datasets into MARS. To take advantage of the SOAP Web service we provide a Java library called MARSExplorer, that allows software developers to extend their programs with data access functionality to MARS. Additionally, if no firewall is located between the client software and MARS, the MARS API (Application Programming Interface) can be used to access public accessible methods via the RMI (Remote Method Invocation) interface.

Access control

To avoid unauthorized database access in a multi user environment the control of user access is a crucial criterion for the acceptance of any database managing functional genomic data. Furthermore, the definition of several fine grained user access levels that allow to visualize, edit or delete data (e.g. expression and sample data, protocols) based on the user rights is mandatory. Therefore we have developed an extensible and easy to use authentication and authorization system (AAS) which rests upon the same technology as MARS. In addition to its Web based management interface, the AAS provides software libraries that enable existing and new applications the integration of highly sophisticated authentication and authorization mechanisms. Moreover, the AAS provides single-sign-on to all its connected Web based applications. Since this AAS can also be used in various projects or institutions relying upon freely available software, MySQL has been choosen as database management system. If desired, this AAS can also manage Windows and Unix accounts using SAMBA [32] and LDAP (Lightweight Directory Access Protocol) [33]. For instance, at the Insitute for Genomics and Bioinformatics all Web based applications and user accounts are administrated by one single instance of the AAS.

Results

Database

All MARS user interfaces are providing a consistent look and feel and are very intuitive to use. In general, the Web based user interface can be divided into two types of user interaction pages: The first one is an input form, where a user can record required and optional data according to the MIAME standard. Required fields are marked in magenta and are validated for correct input. The second allows to list all stored records. To keep the information on a page simple, a user can hide unnecessary datafields. Furthermore it is possible to query for specific records (Figure 5) using the MARS report query tool. Because all Web pages are linked together, MARS permits to follow all conducted steps from the transformed data back to the corresponding well in a microtiter plate and to visualize the results of quality controls. The description of an experiment including hybridizations and their raw datasets is typically the starting point for further analysis.

Analytical pipeline

The usability of MARS and the functionality of the provided interfaces and APIs (Figure 6) are revealed by the integration of MARS into an analytical pipeline of microarray analysis, beginning with image analysis, normalization, gene expression clustering, and finally mapping of gene expression data onto biological pathways.

After entering all required information into MARS, the first step is to normalize the raw data gathered from the image analysis software in order to remove systematic and random errors inherent in the data. ArrayNorm [34], an application for visualization, normalization and analysis of two-color microarray data facilitates these essential steps. Raw data including the definition of experiment classes (biological conditions) and pairs (replicated or dye swapped slides) from whole experiments can be loaded from MARS into ArrayNorm. After visualization and applying different normalization methods like linear regression, LOWESS, or self-normalization, the transformed intensities can be written back to MARS, including the history of the applied methods. The next step in the analytical pipeline is usually gene expression cluster analysis to extract the fundamental patterns inherent in the data and to organize genes with similar expression patterns into biological relevant clusters. Normalized gene expression data can be loaded into Genesis [35]. Genesis allows to cluster the dataset using various similarity distance measurements and different clustering algorithms like hierarchical clustering, k-means, self-organizing maps, principal component analysis, correspondence analysis, and support vector machines. Moreover it is possible to perform one-way ANOVA to identify differentially expressed genes and to incorporate the Gene Ontology (GO) to map gene expression clusters to GO terms. Results can be written back into MARS.

Finally, the Pathway Editor [36] provides the opportunity to access MARS and to map data either from whole experiments or from gene expression clusters to specified pathways in order to get an overview of gene expression changes and their influencing factors. All aforementioned applications have integrated MARSExplorer to connect to MARS and to query, up- and download datasets.

Discussion

The database design, state-of-the-art software technology, well designed user interface, and its application interfaces make MARS a powerful tool for storing, retrieving, and analyzing multi color microarray data. The fusion of Web based and standalone applications provides researchers

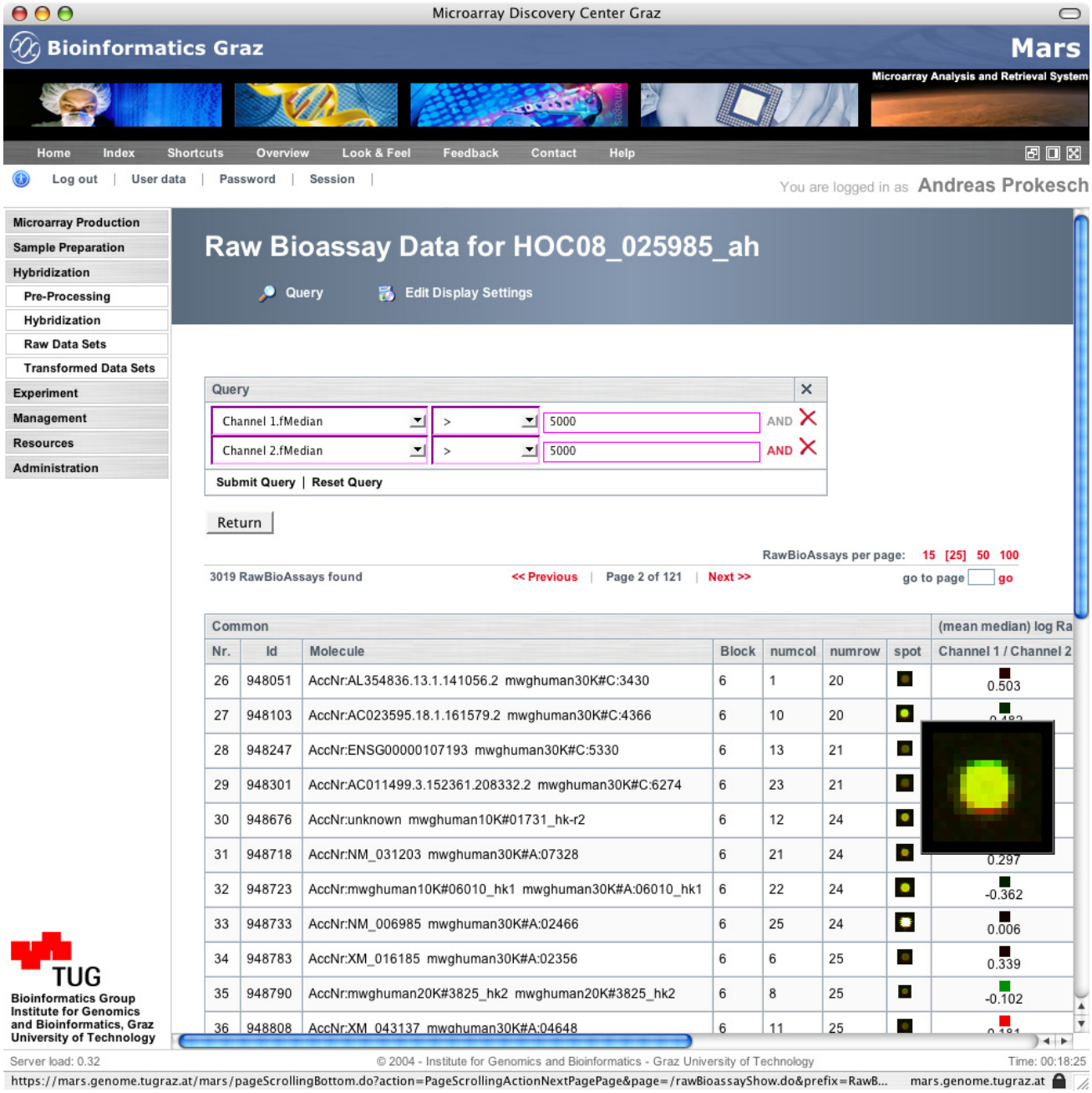


Figure 5
Typical MARS interface listing stored records. It allows to query for specific records using the user friendly query tool.

with an unique set of computational tools for genomic and transriptomic data.

The main strengths of MARS are:

1. Data interfaces

Fundamental for the acceptance of a database are the data interfaces. In principle two types of data interfaces for human computer interactions can be distinguished. Standalone applications allow better program-user interactions while having the drawback that several or even very

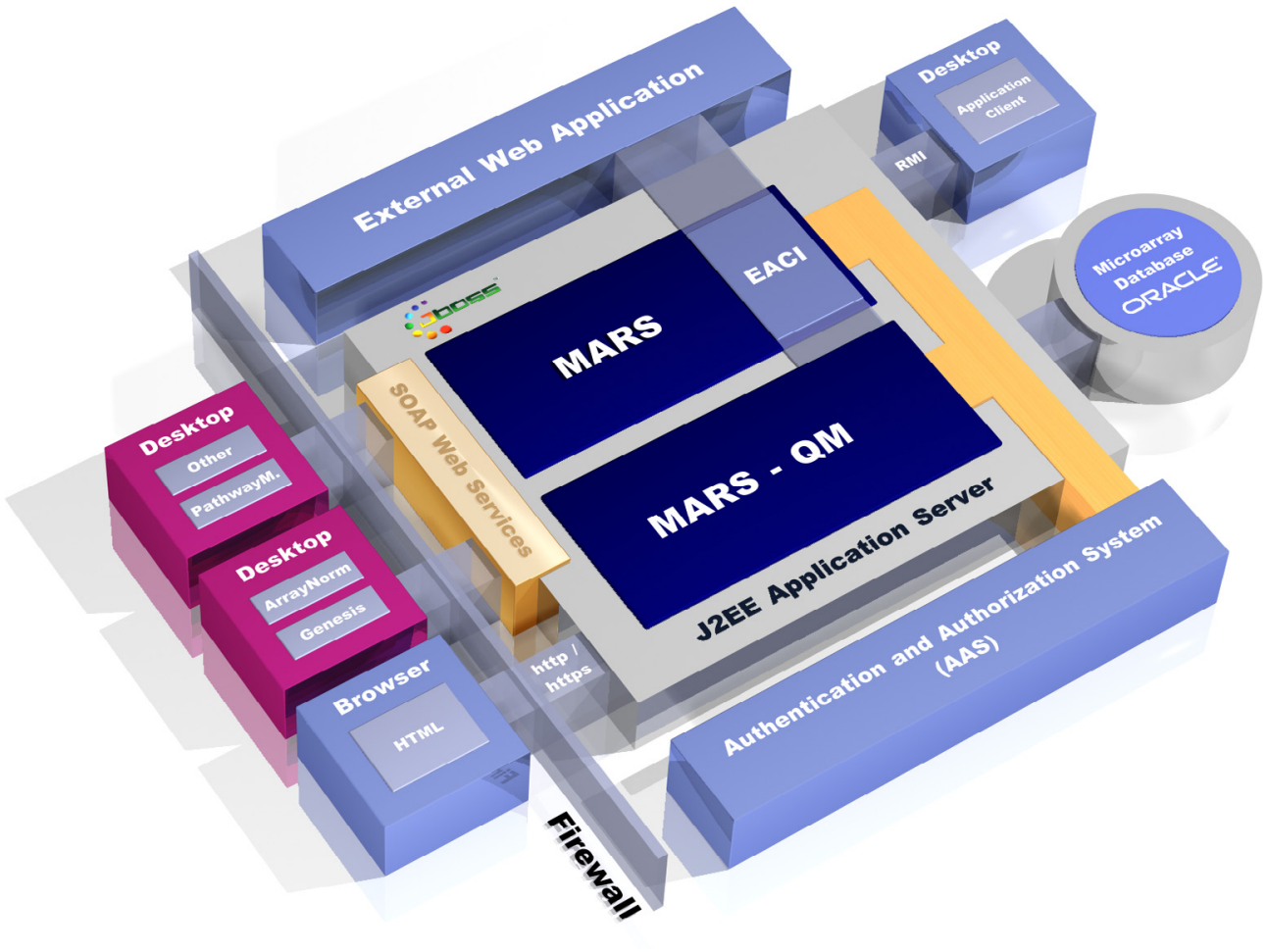


Figure 6
MARS system interactions. MARS and MARS-QM are deployed in a J2EE compliant application server. Interaction is possible either with a standard Web browser or an application supporting the SOAP or RMI protocol. The External Application Connector Interface (EACI) facilitates to connect to data from additional Web applications. SOAP and http/https enable MARS access also through firewalls.

old versions are in use. On the other hand Web based applications can be easily used on every computer without any installation effort and they provide the same and newest version to all users with the cost of limited user interaction. To ensure data integration and good usability we have developed the core data manipulation and storing functions using Web based technology and for data analysis we are using robust applications.

2. Application interfaces

Excellent usability does not only account for primely data interfaces. The ability to easily import data and the availability of well defined application interfaces are also crucial. Different institutions use diverse, mostly self tailored applications with proprietary and varying data formats. MARS provides several data and application interfaces. To import data we provide user definable and manageable parsers. When a user is uploading data, MARS tries to find an appropriate parser based on the file data or format header. Once the data is uploaded and stored, the data can be analyzed using the provided applications. For scientists who would like to analyze their data with other software, MARS provides also a Web service data interface. After some slight adaptations, users can authenticated and down- or upload data. Providing a Web service interface allows through its wide spread and platform independence to be implemented in all well-established programming languages and in tools like Matlab or BioConductor.

Existing Web applications can be plugged-in using the EACI that enables the linkage between data provided by the plugged-in application and data stored in MARS. Moreover it is possible to extend MARS without having to amend the MARS source code.

3. Quality management

In order to assure high-quality data and to understand or optimize lower value data it is important to be able to trace back all conducted quality control steps. MARS traces several quality measurements performed during the microarray production as well as during the sample preparation, extraction, and hybridization process. These quality checks are implemented as an additional application called MARS-QM, which is tightly integrated into MARS.

4. Data sharing and export

MARS enables users to share their datasets with other users. Supplementary to the user oriented data management an institution oriented level has been introduced. This amelioration allows several institutes to store their data into one data repository without having to share common settings and resources such as scanners, but offering the possibility to share the data among them.

Besides the sharing of microarray experiment data we provide the possibility to export hybridizations and experiments using the common exchange format MAGE-ML. This feature facilitates the easy sharing and publishing of high quality, well annotated data within the life sciences community by uploading the generated files to public repositories like ArrayExpress [26].

5. User management

Since microarray- as well as the corresponding quality control data may contain highly sensitive data, we have integrated our AAS into MARS to provide authentication and fine grained authorization mechanisms. The combination of AAS and External Application Connector Interface provides through a single-sign-on mechanisms and dynamic linkage of data the possibility to assemble heterogeneous Web applications to one powerful suite.

Because information attached to molecules is changing quickly, we are currently implementing the possibility to update and enhance the information tagged to a molecule. Changing this information on the molecule level may affect already existing results. In order to avoid such precarious alterations, a user should be able to update the molecule information for each experiment separately instead of replacing the initial molecule information. Further ongoing projects concentrate on the integration of Affymetrix GeneChip arrays into MARS and the improvement of MAGE-ML export capabilities in order to obtain approval from the ArrayExpress annotation team. Both features will be made available to the public in the next major release.

Conclusion

In summary, we have developed an integrated system consisting of a microarray database and a microarray quality control database, that has been tailored to serve the specific needs of microarray based research projects. Due to the unique fusion of using Web based and standalone applications connected to the latest J2EE application server technology, bioinformatics researchers receive the benefits of standards-based software engineering. The system can provide a model how to build up a similar platform for other emerging functional genomics technologies.

Availability and requirements

- Project name: MARS
- Project home page: http://genome.tugraz.at/Software/MARS/MARS.html
- Operating system: Solaris, Linux, Windows
- Programming language: Java, HTML
- Other requirements: Java JDK 1.4.x, Oracle 9i, MySQL 4.0.xx, Server with at least 1 GBytes of main memory
- License: IGB-TUG Software License
- Any restrictions to use by non-academics: no

Installation of MARS is not complicated and should be manageable within a few hours if necessary access rights especially to Oracle and MySQL are granted. Step-by-step instructions are provided at the projects Web site together with the files and scripts necessary for installation. The reference installation of MARS is running on a Sun Fire V880 server under Solaris 9 using Oracle 9i as Database Management System. Attached is a Storage Area Network (SAN) with 2 TBytes.

The production instance of MARS contains information from more than 1000 microtiter plates, 24 array batches, 232 hybridizations, and 312 rawbioassays with about 9,170,000 datapoints.

Authors' contributions

MM, RM, and AS designed and implemented the current version of MARS. They were responsible for the database design, the development of the business- as well as presentation logic. JH developed the quality management system and incorporated it into MARS. MM and JH were the lead developers of the AAS. HH, AP, GS, and MS have

been involved in the compilation of the user requirements document and contributed to the conception and design of the system. ZT was responsible for the overall conception and project coordination. All authors gave final approval of the version to be published.

Acknowledgements

The authors thank the staff of the Institute for Genomics and Bioinformatics for valuable comments and contributions. This work was supported by the Austrian Science Fund (Grant SFB Biomembranes F718) and the bm:bwk, GEN-AU BIN (Bioinformatics Integration Network) and GEN-AU GOLD (Genomics of Lipid-Associated Disorders). Michael Maurer, Robert Molidor and Juergen Hartler were supported by a grant from the Austrian Academy of Sciences.

References

1. Yang IV, Chen E, Hasseman JP, Liang W, Frank BC, Wang S, Sharov V, Saeed AI, White J, Li J, Lee NH, Yeatman TJ, Quackenbush J: **Within the fold: assessing differential expression measures and reproducibility in microarray assays.** *Genome Biol* 2002, **3:**RESEARCH0062.1-RESEARCH0062.12.
2. Schena M, Shalon D, Heller R, Chai A, Brown PO, Davis RW: **Parallel human genome analysis: microarray-based expression monitoring of 1000 genes.** *Proc Natl Acad Sci U S A* 1996, **93:**10614-10619.
3. Schena M, Shalon D, Davis RW, Brown PO: **Quantitative monitoring of gene expression patterns with a complementary DNA microarray.** *Science* 1995, **270:**467-470.
4. Haab BB, Dunham MJ, Brown PO: **Protein microarrays for highly parallel detection and quantitation of specific proteins and antibodies in complex solutions.** *Genome Biol* 2001, **2:**RESEARCH0004.1-RESEARCH0004.13.
5. Iyer VR, Horak CE, Scafe CS, Botstein D, Snyder M, Brown PO: **Genomic binding sites of the yeast cell-cycle transcription factors SBF and MBF.** *Nature* 2001, **409:**533-538.
6. Pollack JR, Perou CM, Alizadeh AA, Eisen MB, Pergamenschikov A, Williams CF, Jeffrey SS, Botstein D, Brown PO: **Genome-wide analysis of DNA copy-number changes using cDNA microarrays.** *Nat Genet* 1999, **23:**41-46.
7. Yan H, Park SH, Finkelstein G, Reif JH, LaBean TH: **DNA-templated self-assembly of protein arrays and highly conductive nanowires.** *Science* 2003, **301:**1882-1884.
8. Mousses S, Caplen NJ, Cornelison R, Weaver D, Basik M, Hautaniemi S, Elkahloun AG, Lotufo RA, Choudary A, Dougherty ER, Suh E, Kallioniemi O: **RNAi Microarray Analysis in Cultured Mammalian Cells.** *Genome Res* 2003, **13:**2341-2347.
9. Hessner MJ, Wang X, Khan S, Meyer L, Schlicht M, Tackes J, Datta MW, Jacob HJ, Ghosh S: **Use of a three-color cDNA microarray platform to measure and control support-bound probe for improved data quality and reproducibility.** *Nucleic Acids Res* 2003, **31:**e60-e60.
10. Tsangaris GT, Botsonis A, Politis I, Tzortzatou-Stathopoulou F: **Evaluation of cadmium-induced transcriptome alterations by three color cDNA labeling microarray analysis on a T-cell line.** *Toxicology* 2002, **178:**135-160.
11. **MGED – Microarray Gene Expression Data Society Home Page** [http://www.mged.org]
12. Brazma A, Hingamp P, Quackenbush J, Sherlock G, Spellman P, Stoeckert C, Aach J, Ansorge W, Ball CA, Causton HC, Gaasterland T, Glenisson P, Holstege FC, Kim IF, Markowitz V, Matese JC, Parkinson H, Robinson A, Sarkans U, Schulze-Kremer S, Stewart J, Taylor R, Vilo J, Vingron M: **Minimum information about a microarray experiment (MIAME)-toward standards for microarray data.** *Nat Genet* 2001, **29:**365-371.
13. Stoeckert CJ Jr, Causton HC, Ball CA: **Microarray databases: standards and ontologies.** *Nat Genet* 2002, **32(Suppl):**469-473.
14. Spellman PT, Miller M, Stewart J, Troup C, Sarkans U, Chervitz S, Bernhart D, Sherlock G, Ball C, Lepage M, Swiatek M, Marks WL, Goncalves J, Markel S, Iordan D, Shojatalab M, Pizarro A, White J, Hubley R, Deutsch E, Senger M, Aronow BJ, Robinson A, Bassett D, Stoeckert CJ Jr, Brazma A: **Design and implementation of microarray gene expression markup language (MAGE-ML).** *Genome Biol* 2002, **3:**RESEARCH0046.1-RESEARCH0046.9.
15. Gene C Ontology: **Creating the gene ontology resource: design and implementation.** *Genome Res* 2001, **11:**1425-1433.
16. Pasquier C, Girardot F, Jevardat dFK, Christen R: **THEA: ontology-driven analysis of microarray data.** *Bioinformatics* 2004, **20:**2636-2643.
17. Mlecnik B, Scheideler M, Hackl H, Hartler J, Sanchez-Cabo F, Trajanoski Z: **PathwayExplorer: web service for visualizing high-throughput expression data on biological pathways.** *Nucleic Acids Res* 2005 in press.
18. Kanehisa M, Goto S, Kawashima S, Nakaya A: **The KEGG databases at GenomeNet.** *Nucleic Acids Res* 2002, **30:**42-46.
19. **BioCarta – Charting Pathways of Life** [http://www.biocarta.com]
20. Kokocinski F, Wrobel G, Hahn M, Lichter P: **QuickLIMS: facilitating the data management for DNA-microarray fabrication.** *Bioinformatics* 2003, **19:**283-284.
21. Saal LH, Troein C, Vallon-Christersson J, Gruvberger S, Borg A, Peterson C: **BioArray Software Environment (BASE): a platform for comprehensive management and analysis of microarray data.** *Genome Biol* 2002, **3:**SOFTWARE0003.1-SOFTWARE0003.6.
22. Gardiner-Garden M, Littlejohn TG: **A comparison of microarray databases.** *Brief Bioinform* 2001, **2:**143-158.
23. Killion PJ, Sherlock G, Iyer VR: **The Longhorn Array Database (LAD): An Open-Source, MIAME compliant implementation of the Stanford Microarray Database (SMD).** *BMC Bioinformatics* 2003, **4:**32-32.
24. Gollub J, Ball CA, Binkley G, Demeter J, Finkelstein DB, Hebert JM, Hernandez-Boussard T, Jin H, Kaloper M, Matese JC, Schroeder M, Brown PO, Botstein D, Sherlock G: **The Stanford Microarray Database: data access and quality assessment tools.** *Nucleic Acids Res* 2003, **31:**94-96.
25. Theilhaber J, Ulyanov A, Malanthara A, Cole J, Xu D, Nahf R, Heuer M, Brockel C, Bushnell S: **GECKO: a complete large-scale gene expression analysis platform.** *BMC Bioinformatics* 2004, **5:**195-195.
26. Brazma A, Parkinson H, Sarkans U, Shojatalab M, Vilo J, Abeygunawardena N, Holloway E, Kapushesky M, Kemmeren P, Lara GG, Oezcimen A, Rocca-Serra P, Sansone SA: **ArrayExpress – a public repository for microarray gene expression data at the EBI.** *Nucleic Acids Res* 2003, **31:**68-71.
27. Edgar R, Domrachev M, Lash AE: **Gene Expression Omnibus: NCBI gene expression and hybridization array data repository.** *Nucleic Acids Res* 2002, **30:**207-210.
28. **JBoss.com:: The Professional Open Source Company** [http://www.jboss.org]
29. Quackenbush J: **Data standards for 'omic' science.** *Nat Biotechnol* 2004, **22:**613-614.
30. Dudoit S, Fridlyand J: **Bagging to improve the accuracy of a clustering procedure.** *Bioinformatics* 2003, **19:**1090-1099.
31. **The MathWorks – Matlab and Simulink for Technical Computing** [http://www.mathworks.com]
32. **Samba – opening windows to a wider world** [http://www.samba.org]
33. **OpendLDAP** [http://www.openldap.org]
34. Pieler R, Sanchez-Cabo F, Hackl H, Thallinger GG, Trajanoski Z: **ArrayNorm: comprehensive normalization and analysis of microarray data.** *Bioinformatics* 2004.
35. Sturn A, Quackenbush J, Trajanoski Z: **Genesis: cluster analysis of microarray data.** *Bioinformatics* 2002, **18:**207-208.
36. Trost E, Hackl H, Maurer M, Trajanoski Z: **Java editor for biological pathways.** *Bioinformatics* 2003, **19:**786-787.

BIOINFORMATICS

Vol. 21 Suppl. 2 2005, pages ii130–ii136
doi:10.1093/bioinformatics/bti1122

Microarrays

A fully Bayesian model to cluster gene-expression profiles

C. Vogl[1,*,†], F. Sanchez-Cabo[2,†], G. Stocker[2], S. Hubbard[3], O. Wolkenhauer[4] and Z. Trajanoski[2]

[1]Institute of Animal Breeding and Genetics, Veterinärmedizinische Universität Wien, 1210 Vienna, Austria, [2]Institute for Genomics and Bioinformatics and Christian Doppler Laboratory for Genomics and Bioinformatics, Graz University of Technology, Petersgasse 14, 8010 Graz, Austria, [3]Faculty of Life Sciences, University of Manchester, M60 1QD Manchester, UK and [4]Institute of Informatics, University of Rostock, 18051 Rostock, Germany

ABSTRACT

Motivation: With cDNA or oligonucleotide chips, gene-expression levels of essentially all genes in a genome can be simultaneously monitored over a time-course or under different experimental conditions. After proper normalization of the data, genes are often classified into co-expressed classes (clusters) to identify subgroups of genes that share common regulatory elements, a common function or a common cellular origin. With most methods, e.g. *k*-means, the number of clusters needs to be specified in advance; results depend strongly on this choice. Even with likelihood-based methods, estimation of this number is difficult. Furthermore, missing values often cause problems and lead to the loss of data.

Results: We propose a fully probabilistic Bayesian model to cluster gene-expression profiles. The number of classes does not need to be specified in advance; instead it is adjusted dynamically using a Reversible Jump Markov Chain Monte Carlo sampler. Imputation of missing values is integrated into the model. With simulations, we determined the speed of convergence of the sampler as well as the accuracy of the inferred variables. Results were compared with the widely used *k*-means algorithm. With our method, biologically related co-expressed genes could be identified in a yeast transcriptome dataset, even when some values were missing.

Availability: The code is available at http://genome.tugraz.at/BayesianClustering/

Contact: claus.vogl@vu-wien.ac.at

Supplementary information: The supplementary material is available at http://genome.tugraz.at/BayesianClustering/

1 INTRODUCTION

Gene-expression levels of essentially all genes can be monitored with cDNA or oligonucleotide chips over a time-course or under different experimental conditions. After appropriate filtering and normalization, a gene-expression matrix (GEM) is produced. Although the number of genes that can be monitored is generally impressive, the proportion of genes with missing values can be up to 50% in a two-color microarray experiment.

Clustering methods are widely applied to the GEM in order to identify subgroups of genes that share properties, e.g. common regulatory factors, a common function or a common cellular origin. Most of the traditional clustering algorithms are heuristic, e.g. *k*-means (Tavazoie *et al*., 1999), hierarchical clustering (Eisen *et al*., 1998) or self-organizing maps (SOM) (Tamayo *et al*., 1999). For some of these methods, the analyst needs to choose the number of clusters in advance. This choice affects clustering and is often a question that the microarray experiment is expected to address. Furthermore, these methods often have difficulties with missing values. Finally, most of the traditional clustering algorithms are deterministic, assigning a gene unequivocally to a particular cluster according to their similarity to other genes. A gene, however, might actually be grouped into several different clusters for biological reasons (Segal *et al*., 2003).

In contrast to these *ad hoc* methods, clustering algorithms based on a probabilistic model provide a consistent framework (Yeung *et al*., 2001a; Medvedovic and Sivaganesan, 2002; Dougherty and Brun, 2004). In particular, Gaussian mixture models have been widely applied to cluster gene-expression data owing to the roughly Gaussian distribution that these data present after appropriate transformations (Yeung *et al*., 2001a; Huber *et al*., 2002). Frequentist and Bayesian methods can then be used to infer the model parameters. Missing data pose no problem to both methods; with a Bayesian approach, it is even possible to estimate the distribution of the missing values. But inference of the number of clusters remains problematic: with frequentist methods, multiple log-maximum-likelihood values need to be compared, penalizing for increasing model complexity (e.g. the Bayesian Information Criterion, BIC, Schwarz, 1978). Alternatively, *N*-fold cross-validation or variants thereof can be employed (Yeung *et al*., 2001b). With a Bayesian approach, the number of clusters can be treated as a random variable and the posterior probabilities of different numbers of clusters can be evaluated. Unfortunately, the only implementation of such an approach applied to expression data up to date (Medvedovic and Sivaganesan, 2002) requires integration of the posterior distribution, which makes handling of missing values problematic.

In this paper, a fully probabilistic model is proposed and applied to cluster gene-expression profiles from microarray experiments. The sampling scheme is based on the Reversible Jump Markov Chain Monte Carlo (RJMCMC) (Green, 1995) applied to mixture models (Richardson and Green, 1997) to allow the number of clusters to

*To whom correspondence should be addressed.

†The authors wish it to be known that, in their opinion, the first two authors should be regarded as joint First Authors.

vary. This approach is fully Bayesian since all parameters of interest, including the missing values and the number of clusters, are treated as random variables and their posterior distribution is approximated with RJMCMC. The method forms clusters without the need of choosing the number of clusters in advance. It can also deal with replicate experiments. The model was first applied to simulated data, for which the true clusters were found with a high specificity and sensitivity. Our model also outperformed *k*-means clustering. When applied to experimental data on the yeast cell cycle (Spellman *et al*., 1998), the inferred clusters of genes were biologically sensible.

2 MATERIALS AND METHODS

2.1 Probabilistic model

Typically, a microarray dataset consists of the expression levels of N genes measured at T different biological conditions. In particular, the T biological conditions can be time points. Let i, $1 \leq i \leq N$ index the genes and t, $1 \leq t \leq T$ the time points; furthermore, let the variable x_{it} indicate the presence of a reliable expression level for the i-th gene at the t-th time point. Generalization to several replicates is straightforward but complicates notation. The algorithm accounts for both missing data as well as replicates. We aim to characterize the K stochastic processes (indexed by k, $1 \leq k \leq K$) that generated the N gene profiles $\vec{y}_i = (y_{i1}, \ldots, y_{iT})$ with $1 \leq i \leq N$. Missing data are allowed.

The goal of a Bayesian analysis is to determine the marginal posterior probability distribution for all parameters of interest. Since for complicated models this distribution cannot be obtained analytically, we will make use of a Markov Chain Monte Carlo (MCMC) simulation scheme (in particular, an RJMCMC to allow the jump between spaces of different dimensions). For reasons of speed, the algorithm is implemented in the C++ programming language.

2.1.1 General probabilistic model Let us suppose that K stochastic processes generated the N gene-expression profiles observed. We introduce the characteristic matrix $\mathbf{Z}_{N\times K}$ such that each row vector $\vec{z}_i = \{z_{i1}, \ldots, z_{iK}\}$ of $\mathbf{Z}$ has an entry of one for the cluster into which the i-th gene is currently classified and 0s for all other classes. Let ω_k be the weight of class k, with $\sum_k \omega_k = 1$. One might think of ω_k as the probability of the i-th gene to fall into the k-th cluster before knowing its expression data, i.e. $\omega_k = \Pr(z_{ik} = 1)$ for $k = 1, \ldots, K$.

We assume that the number of clusters K is unknown. Following earlier treatment of a similar model (Richardson and Green, 1997), the relationships between parameters conditional on the hyperparameters are formulated as follows (Fig. 1):

$$\begin{aligned}&\Pr(\mathbf{Y}, \mathbf{Z}, \vec{\mu}, \vec{\sigma}, \vec{\omega}, K \mid \lambda, \alpha, \vec{\mu}_0, \kappa_0, \nu_0, \tau_0)\\ &\quad = \Pr(\mathbf{Y} \mid \mathbf{Z}, \vec{\mu}, \vec{\sigma}) \Pr(\mathbf{Z} \mid K, \vec{\omega}) \Pr(\vec{\omega} \mid \alpha, K)\\ &\qquad \times \Pr(\vec{\mu} \mid \vec{\sigma}, K, \vec{\mu}_0, \kappa_0) \Pr(\vec{\sigma} \mid K, \nu_0, \tau_0) \Pr(K \mid \lambda)\,,\end{aligned}$$

where the data matrix is $\mathbf{Y} = (\vec{y}_1, \ldots, \vec{y}_N)$. The parameters are $\vec{\omega} = (\omega_1, \ldots, \omega_K)$, $\mathbf{Z} = (\vec{z}_1, \ldots, \vec{z}_N)$, $\vec{\mu} = (\vec{\mu}_1, \ldots, \vec{\mu}_K)$, $\vec{\mu}_0 = (\vec{\mu}_{01}, \ldots, \vec{\mu}_{0K})$ and $\vec{\sigma} = (\sigma_1, \ldots, \sigma_K)$. We assume a simple variance–covariance matrix, i.e. $\Sigma_k = \sigma_k^2 \mathbf{I}_{T\times T}$ different for each cluster k and correspondingly simple priors (ν_0, τ_0^2) for it. The hyperparameters are then, $\lambda, \alpha, \vec{\mu}_0, \kappa_0, \nu_0, \Lambda_0 = \tau_0^2 \mathbf{I}_{T\times T}$. The parameters and hyperparameters will be defined more thoroughly below.

2.1.2 Likelihood Conditional on z_{ik}, i.e. on knowing which mixture component generated the i-th gene expression profile, the i-th data vector $\vec{y}_i = \{y_{i1}, \ldots, y_{iT}\}$ is assumed to be normally distributed around the respective means of the k-th class with covariance matrix $\Sigma_k = \sigma_k^2 \mathbf{I}_{T\times T}$. For the sake of simplicity, we present here the detailed equations only for the complete data. Equations for the data with missing values can be found in the Supplementary material. The likelihood of the it-th data point conditional on $z_{ik} = 1$ is proportional to

$$\Pr(y_{it} | \mu_{kt}, \sigma_k^2) \propto (\sigma_k^2)^{-1/2} \exp\left(-\frac{(y_{it} - \mu_{kt})^2}{2\sigma_k^2}\right). \quad (1)$$

Let us now define $n_k = \sum_{i=1}^N z_{ik}$ (number of observations of genes in cluster k), $\bar{y}_{kt} = (\sum_i z_{ik} y_{it})/n_k$ (mean expression level of the genes in cluster k at condition t), and $\mathrm{SSQ}_k = \sum_i \sum_t z_{ik}(y_{it} - \bar{y}_{kt})^2$ (the sum of squares of expression levels of the genes in cluster k at all conditions t). Owing to the structure of the covariance matrix ($\Sigma_k = \sigma_k^2 \mathbf{I}_{T\times T}$) and assuming that the gene-expression profiles are independent and identically distributed (i.i.d.) within each subpopulation, the likelihood can be calculated by multiplying over genes and biological conditions. Whence,

$$\begin{aligned}&\Pr(\vec{y}_1, \ldots, \vec{y}_N \mid \mathbf{Z}, \vec{\mu}, \vec{\Sigma})\\ &\quad \propto \prod^K \Pr(\vec{y}_1, \ldots, \vec{y}_N \mid \mathbf{Z}, \vec{\mu}_k, \sigma_k)\\ &\quad \propto \prod^K (\sigma_k^2)^{-Tn_k/2} \exp\left(-\frac{\sum_t^T n_k(\bar{y}_{kt} - \mu_{kt})^2 + \mathrm{SSQ}_k}{2\sigma_k^2}\right). \quad (2)\end{aligned}$$

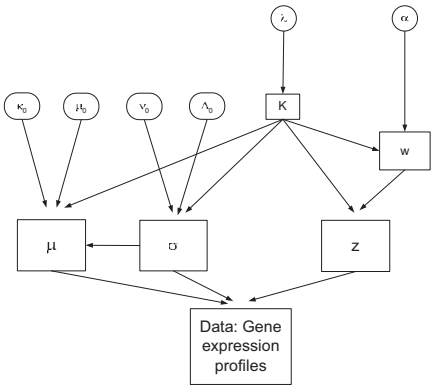


Fig. 1. Hierarchical mixture model. The parameters in circles are the hyperpriors.

2.1.3 Prior distributions Proper conjugate prior distributions were chosen that provide little information. Since empty clusters are possible in the model, non-informative priors cannot be used (Richardson and Green, 1997). Compared with the priors in Richardson and Green (1997), we use conjugate instead of independent priors for $\vec{\mu}_k$ and σ_k to reduce complexity in the multidimensional case. The same priors were used by Baldi and Long (2001) to detect differentially expressed genes from microarray experiments. Since the amount of data is typically large, the posterior is governed by the likelihood, such that the choice of the priors has little influence on the results.

In particular, we chose the following priors and hyperpriors:

- The weights ($0 \leq \omega_k \leq 1 \,\forall\, k$ and $\sum_k \omega_k = 1$) are assumed to be drawn from a symmetric Dirichlet prior with hyperparameter α set to one.
$$\Pr(\omega_1, \ldots, \omega_K \mid K, \alpha) \sim \mathrm{Dir}(\alpha, \ldots, \alpha)\,. \quad (3)$$
- The prior for K is a Poisson distribution with hyperparameter λ set to 10. A truncated Poisson could also be chosen.
- $\vec{z}_i$ is sampled from a multinomial generalization of a Bernoulli with $\Pr(z_{ik} = 1) = \omega_k \,\forall k$.
- For the simple variance–covariance structure we assume, we chose a conjugate prior distribution for each σ_k^2 independently. Conditional on σ_k^2, the prior distribution of $\vec{\mu}_k$ was chosen to be normal with a prior

mean of $\vec{\mu}_{0k}$ and a covariance matrix of Σ_k/κ_0 with $\Sigma_k = \sigma_k^2 \mathbf{I}_{T\times T}$ (Gelman *et al*., 1995):

$$\Pr(\sigma_k^2) \sim \text{Inv-}\chi^2(\nu_0, \tau_0)$$
$$\Pr(\vec{\mu}_k \mid \Sigma_k) \sim N(\vec{\mu}_0, \Sigma_k/\kappa_0)\,.$$

More complicated models that incorporate dependencies among the different time points (Ramoni *et al*., 2002) could also be used. The variance–covariance matrix we propose corresponds to the unequal volume model by Yeung *et al*. (2001a). Specifically, we set the prior value of the hyperparameters to $\kappa_0 = 1$, $\nu_0 = 2$ and $\tau_0 = 1$. If the data have been properly normalized, we can assume $\vec{\mu}_{0k} = 0\ \forall k$. The joint prior distribution of $\vec{\mu}_k$ and σ_k^2 is then

$$\Pr(\vec{\mu}_k, \sigma_k^2) \propto (\sigma_k^2)^{-(\nu_0/2+1)} \exp\left(-\frac{\nu_0\tau_0^2}{2\sigma_k^2}\right) \times \prod_t \left((\sigma_k^2)^{-1/2} \exp\left(-\frac{\kappa_0\mu_{kt}^2}{2\sigma_k^2}\right)\right). \quad (4)$$

2.1.4 Conditional posterior distributions For each gene i independently, the conditional posterior distribution of the vector $\vec{z}_i = \{z_{i1}, \ldots, z_{iK}\}$ is the multinomial generalization of the Bernoulli distribution:

$$\Pr(\vec{z}_i \mid \vec{y}_i, K, \vec{\omega}, \mu, \Sigma) \propto \prod_k (\Pr(\vec{y}_i \mid \mu_k, \sigma_k^2) \cdot \omega_k)^{z_{ik}}. \quad (5)$$

The conditional posterior distribution of the vector of weights is Dirichlet:

$$\Pr(\omega_1, \ldots, \omega_K \mid \mathbf{Z}) \sim \text{Dir}(\alpha + n_1, \ldots, \alpha + n_K)\,, \quad (6)$$

where $n_k = \sum_{i=1}^N z_{ik}$, i.e. the number of genes in the k-th cluster.

The conditional posterior distribution of the means and the variance of the k-th cluster is

$$\begin{aligned}\Pr(\vec{\mu}_k, \sigma_k^2 \mid \mathbf{Y}, \mu_0, \kappa_0, \nu_0, \tau_0) &\propto \Pr(\mathbf{Y} \mid \vec{\mu}_k, \sigma_k^2) p(\vec{\mu}_k, \sigma_k^2) \\ &\propto (\sigma_k^2)^{-((\nu_0+Tn_k)/2+1)} \\ &\quad \times \exp\left(-\frac{\nu_0\tau_0^2 + SSQ_k + (\kappa_0 n_k/\kappa_0 + n_k)\sum_t \bar{y}_{kt}^2}{2\sigma_k^2}\right) \\ &\quad \times \prod_t \left((\sigma_k^2)^{-1/2} \exp\left(-\frac{(\kappa_0 + n_k)(\mu_{n_k t} - \mu_{kt})^2}{2\sigma_k^2}\right)\right), \end{aligned} \quad (7)$$

where $\mu_{n_k t} = n_k \bar{y}_{kt}/(n_k + \kappa_0)$. Samples from the conditional joint posterior distribution of the means and variances can then be obtained as follows (Gelman *et al*., 1995): draw σ_k^2 from a scaled Inv-$\chi^2(\nu_{n_k}, \sigma_{n_k}^2)$, with $\nu_{n_k} = \nu_0 + Tn_k$ and $\nu_{n_k} \cdot \sigma_{n_k}^2 = \nu_0\tau_0^2 + \text{SSQ}_k + (\kappa_0 n_k/(\kappa_0 + n_k))\sum_t \bar{y}_{kt}^2$; draw $\mu_{kt}\ \forall\, t$ from a normal distribution $N(\mu_{n_k t}, \frac{\sigma_k^2}{n_k+\kappa_0})$.

2.1.5 RJMCMC sampling scheme The RJMCMC method proceeds by iterating through rounds of cyclically updating the variables in turn:

(1) updating the characteristic matrix $\mathbf{Z}$;

(2) updating the weights ω;

(3) updating the class means and variances $(\vec{\mu}_k, \sigma_k)$;

(4) adding or deleting an empty class.

For the first three moves, we use a Gibbs kernel, i.e. we sample from the respective conditional distributions. For the fourth move, we assume the number of classes to be drawn from a Poisson prior with mean λ. We suggest an additional class with probability $p_{(K+1)}$ and removal of a random class with probability p_K. For an addition, we create the new class $(K+1)$ as follows: we sample a new class variance and a new vector of class means $\vec{\mu}_{K+1}$ from the prior; we sample ϕ from a beta$(\phi \mid \alpha, K\alpha + N)$, and set the new class weight of the $(K+1)$-th class to $\omega_{(K+1)*} = \phi$ and that of the other K classes to $\omega_{k*} = (1-\phi)\cdot\omega_k$. Note that this choice of the proposal distribution simplifies the formulas compared with Richardson and Green (1997). The added class is always empty, such that the likelihood ratio is unity. We then accept the additional class with probability $\min\{1, A\}$, where

$$\begin{aligned} A &= \frac{\Pr(\omega_1^* \cdots \omega_{K+1}^* \mid \mathbf{Z})}{\Pr(\omega_1 \cdots \omega_K \mid \mathbf{Z})} \cdot \frac{\Pr(K+1)}{\Pr(K)} \cdot \frac{1}{\text{beta}(\phi \mid \alpha, K\alpha + N)} \cdot J \\ &= \frac{(\omega_1(1-\phi))^{\alpha+n_1-1} \cdots (\omega_K(1-\phi))^{\alpha+n_K-1}\phi^{\alpha-1}}{\omega_1^{\alpha+n_1-1} \cdots \omega_K^{\alpha+n_K-1}} \\ &\quad \times \frac{\Pr(K+1)}{\Pr(K)} \cdot \frac{1}{\phi^{\alpha-1}(1-\phi)^{K\alpha+N-1}} \cdot (1-\phi)^{K-1} \\ &= \frac{\lambda}{K+1}. \end{aligned} \quad (8)$$

For a deletion, we propose to randomly remove a class without genes. The acceptance probability for the delete move is $\min\{1, A^{-1}\}$.

2.1.6 Imputation of missing values The missing expression values can be imputed with the MCMC method at each step in the iteration by sampling from their conditional distribution given the class, say k,

$$\Pr(y_{it} \mid \ldots) \sim N(\mu_{kt}, \sigma_k^2)\,.$$

These imputed values can be treated as data. Over the course of the iteration, an approximation to the posterior distribution of the missing y_{it} can thus be obtained.

2.1.7 Interpretation of the posterior distributions Since jumping between spaces of different dimensions is allowed using the RJMCMC sampling scheme, it is difficult to follow the evolution of each cluster over the iterations. Hence, for each pair of genes, a similarity index was calculated as the number of iterations that appeared in the same cluster divided by the total number of iterations. The empirical distribution of all the pairwise similarity indexes will be used to determine a threshold; we chose the 95 percentile of the distribution. If the similarity index between two genes is higher than this limit, the genes are considered similarly expressed. In other words, a certain gene can be used as a 'seed' to generate a cluster of genes with similar profiles. Figure 2 shows the process in practice.

2.1.8 Implementation details The RJMCMC methods were implemented in C++. The post-Bayesian methods were implemented either in R (Ihaka and Gentleman, 1996) or, if performance was an issue, in C++. Code is available in the Supplementary material.

Parallel execution in a Beowulf linux cluster with 50 Intel Xeon 2.60 GHz CPUs considerably reduced the computational time. 10 000 iterations of the RJMCMC were performed; an initial 'burn-in' phase of 1000 iterations was discarded. See following section for a justification of the choice of these parameters.

3 RESULTS FOR THE SIMULATED DATA

Ten data sets simulating a typical microarray experiment with 6000 gene-expression profiles ($N = 6000$) measured across 10 different conditions ($T = 10$) were generated. The true number of clusters was 40. For each simulated dataset a user-specified proportion of data was substituted by missing values (10% in the datasets used in this paper). This way the performance of the method could be tested on datasets with missing values.

3.1 Convergence assessment

In order to guarantee the convergence and independence of the initial conditions when using iterative simulation, Gelman *et al*. (1995) suggest running several independent sequences with overdispersed starting conditions and monitoring the evolution of different independent parameters. The sequences will have reached convergence if the within-run variance roughly equals between-runs variation.

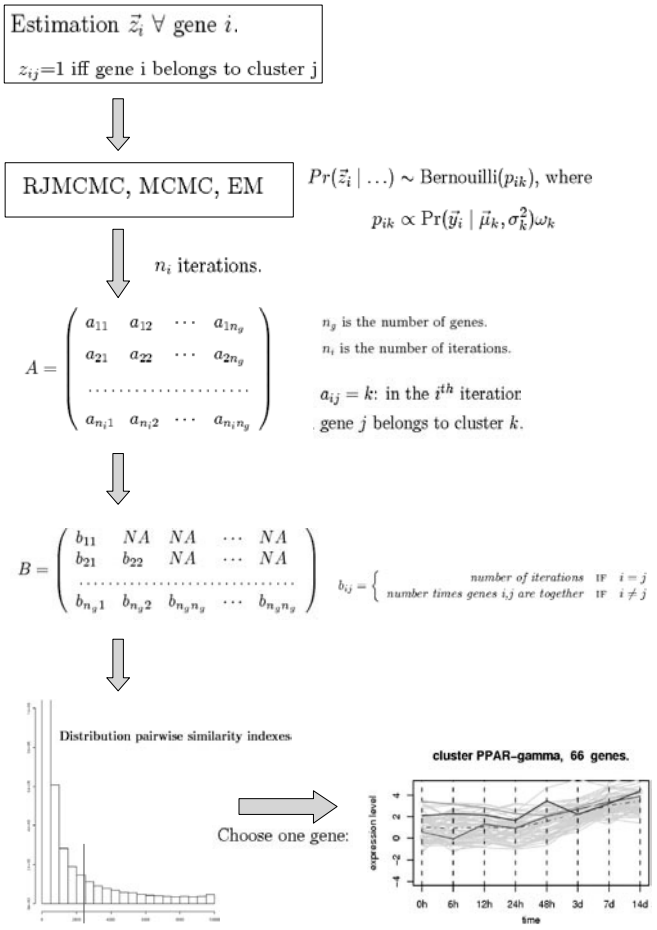


Fig. 2. RJMCMC-based clustering of gene-expression profiles: At each iteration, the different variables in the model are stored; using the indicator variable that assigns genes to the clusters, a matrix of similarity indices is built; given an important gene, others similar to it are found (threshold: the 95% percentile of the distribution).

They suggest to estimate the potential scale reduction as

$$\hat{R} = \sqrt{\frac{\text{var} + (\theta \mid y)}{W}}, \tag{9}$$

where $\text{var} + (\theta \mid y) = (n - 1/n)W + (1/n)B$, W is the within-run variance and B is the between-runs variance, and n is the length of the run. Gelman *et al*. (1995) recommend increasing the number of iterations if $\hat{R}$ is much bigger than 1.

When, for the same dataset, the starting number of clusters for 10 runs was varied between 10 and 100, rapid convergence was observed (Table 1). Hence, a burn-in period of 1000 iterations followed by 10 000 iterations seems sufficient for approximate convergence (Fig. 3).

3.2 Sensitivity and specificity

For the simulated data, one gene from each of the 40 true clusters was chosen at random. All genes with a similarity index higher than the threshold were selected. Subsequently, the sensitivity and specificity were calculated for each generated cluster. Sensitivity quantifies the

Table 1. Between-runs variability (B), within-runs variability (W), vâr and $\hat{R}$ as described by Gelman *et al*. (1995) based on the log likelihood

Iterations	B	W	var$^+$	$\hat{R}$
[1,100]	4 911 011	1 800 494	1 831 599	1.0086
[1,500]	3 096 544	515 818	520 980	1.0050
[1,1000]	2 051 000	276 125	277 900	1.0032
[1000,11 000]	282 591	13 637	13 663	1.0010
[1000,15 000]	167 101	13 679	13 690	1.0004
[1000,20 000]	224 146	13 512	13 523	1.0004

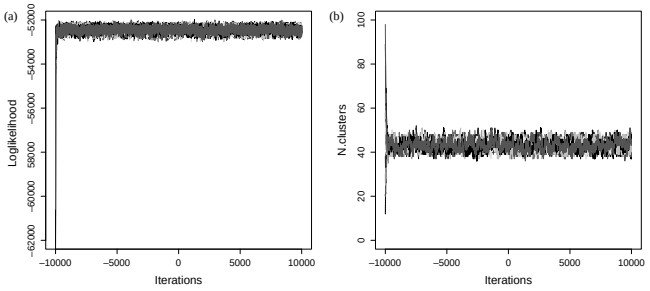


Fig. 3. Evolution of (**a**) the log likelihood and (**b**) the number of clusters in the course of 10 runs of the same dataset with overdispersed starting conditions. Color-coded figures are included in Supplementary Figure 2.

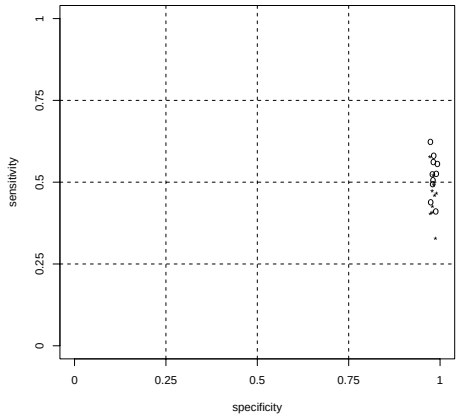


Fig. 4. Sensitivity and specificity of the RJMCMC-based algorithm. For each one of the 10 simulated datasets, the RJMCMC-based clustering was run over the complete dataset (without missing values) and over the dataset with missing values. The average sensitivity and specificity of the RJMCMC-based clustering are displayed without ('open circles') and with missing values ('asterisks').

capacity of a given test or algorithm to discriminate true positives; specificity quantifies the ability of a test to detect true negatives. The sensitivity of the algorithm is only ∼0.5 on average, whereas the specificity is high, i.e. the number of false positives is low (Fig. 4), irrespective whether data were complete or with 10% missing values. This is attributable to the way clusters are built (based on a 'seed' gene) which might result in the merge of more than one of the original clusters into one, depending on the profile of the selected gene.

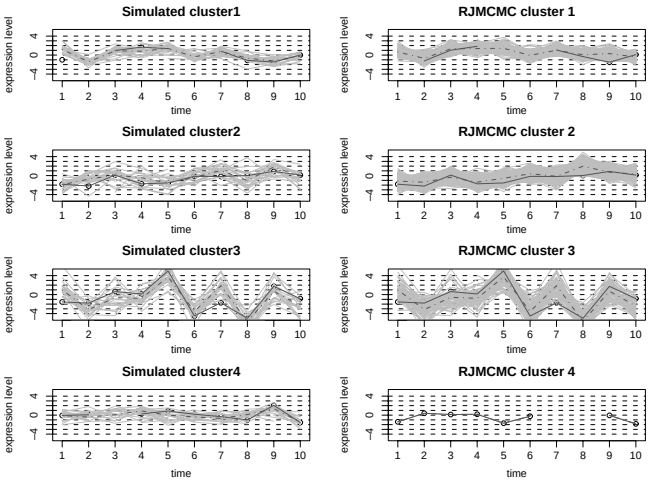


Fig. 5. Four true clusters (left) are compared with the RJMCMC-clusters (right) obtained using a random gene as seed. The dashed line represents the centroid of the cluster and the solid line the profile of the gene used as seed. Clusters 5–40 can be found in the Supplementary Figures 4–8.

3.3 Clustering of gene-expression profiles with missing values

The concordance between the clusters of the complete and the incomplete (0.1 missing) datasets is >75% on average (85.07, 90.64, 84.42, 89.16, 81.76, 83.29, 86.44, 80.84, 84.22 and 77.62%) for the 10 simulated datasets. See Figure 3 in the Supplementary material for the complete percentages per simulated cluster.

When a random gene is picked from the first four simulated clusters and the clusters are reconstructed from this seed, high concordance with the profiles of the true clusters is observed (Fig. 5 and Supplementary material).

3.4 Comparison with *k*-means clustering

The RJMCMC-based clustering algorithm has the advantage over other clustering methods, e.g. *k*-means, of allowing the clustering of gene-expression profiles without advance specification of the number of clusters. Methods proposed to estimate the number of clusters in a given dataset are often based on *k*-fold cross-validation and variants thereof, e.g. the figure of merit (FOM, Yeung *et al*., 2001b).

For the simulated data, the results using FOM proved inconclusive (see the Supplementary material). Thus the true number of clusters was supplied to the *k*-means algorithm using the academic software 'Genesis' (Sturn *et al*., 2002). Its sensitivity and specificity was compared with the RJMCMC-based clustering. Since real data almost always contain missing values, we used the dataset with missing values for comparison.

Although the number of clusters was given in advance to the *k*-means clustering algorithm, the RJMCMC-based clustering still showed greater sensitivity than the *k*-means clustering algorithm (Fig. 6 and Supplementary Figure 9), whereas the specificity was similar and high.

4 RJMCMC-BASED CLUSTERING OF CELL-CYCLE REGULATED GENES

In addition, we tested the performance of the method on data from one of the most widely used experiments to test algorithm performance,

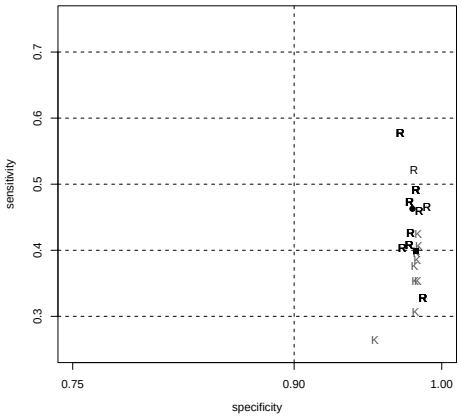


Fig. 6. Comparison of the performance of the *k*-means ('*K*') and the RJMCMC clustering algorithm ('*R*') based on the specificity and sensitivity. The medians are represented by a square and a bullet.

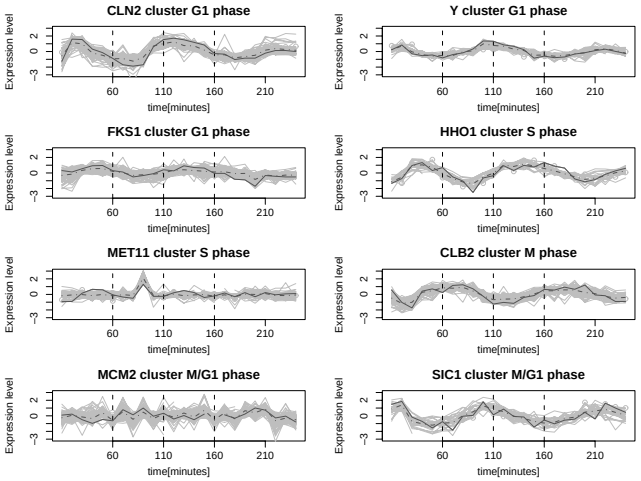


Fig. 7. Clusters of cell-cycle regulated genes found using the RJMCMC-based clustering method and based in the cdc15 experiment by Spellman *et al.* (1998).

the cdc15 experiment, which seems to be the most robust among those performed by Spellman *et al*. (1998). Based on hierarchical clustering and promoter analysis, Spellman *et al*. (1998) detected eight main clusters of cell-cycle regulated genes, peaking at different phases during the cell cycle. As a seed, we took either the most important gene in the cluster or, if no gene was biologically more relevant than the rest as in the *Y* or in the histones cluster, a random one. Figure 7 shows the expression profiles of the genes similar to them according to the RJMCMC-based clustering. The genes clustering together exhibit a very similar expression profile over the time-course, even though the whole dataset without prefiltering was used.

5 DISCUSSION

A paradigm of microarray studies is that clusters of co-expressed genes might share a common regulatory program or might be functionally related. The methods used to identify these clusters within a gene-expression dataset vary from the simple hierarchical clustering

(Spellman *et al*., 1998; Sorlie *et al*., 2003) to complicated probabilistic models (Golub *et al*., 1999; Zhou *et al*., 2005). The number of clusters hidden in the data may not be relevant by itself. But for many clustering algorithms, it is essential for the formation of the clusters and thus for determining which genes are co-expressed and, therefore, involved in the same biological process.

Herein, we present a hierarchical Bayesian model that infers clusters of genes co-expressed in replicated microarray experiments under different biological conditions, e.g. under different treatments or at different time points. The number of clusters does not need to be specified in advance, but is estimated from the data concurrently with the other parameters. Furthermore, missing values are imputed in the process. The posterior distribution is obtained through RJMCMC (Green, 1995) applied to mixture models (Richardson and Green, 1997).

Model-based clustering has already been applied extensively to gene-expression profiles (Yeung *et al*., 2001a, 2003; Ramoni *et al*., 2002; Medvedovic and Sivaganesan, 2002). The first authors implemented the *R* package `mclust` within the Bioconductor project. In the models by Yeung *et al*. (2001a) and Ramoni *et al*. (2002), the number of clusters needs to be specified in advance, although the use of a probabilistic model makes it possible to estimate the optimal number of clusters that maximizes the likelihood, or rather BIC, of the data. Medvedovic and Sivaganesan (2002) allow the number of clusters to vary, as in the present paper. But their method requires integration of the posterior distribution and is thus less flexible. It cannot, e.g. deal with missing data that occur often with microarrays. This flaw is shared with other methods. As a remedy, gene-expression profiles with missing values must either be excluded or missing values need to be imputed by other means, e.g. KNN (Troyanskaya *et al*., 2001), prior to clustering.

Generally, Bayesian inference can be applied to any problem for which a probabilistic model can be formulated. If the full likelihood (or posterior distribution) cannot be obtained in closed form, the full set of conditional distributions suffices. Hierarchical models, missing data and even variation of the number of parameters can easily be accommodated. Compared with other approaches, Bayesian inference is thus extremely flexible (Beaumont and Rannala, 2004).

Bayesian statistics were here used because they allow the number of clusters to vary and the integration of missing values. Both problems are pressing in the clustering of gene-expression data. In order to test the performance of our method, we generated simulated data similar to the yeast cell-cycle dataset by Spellman *et al*. (1998) with 6000 genes, 10 experimental conditions (e.g. time points) and 40 clusters (compare Medvedovic and Sivaganesan, 2002). Performance was assessed by running each dataset with different starting conditions, i.e. varying the initial number of clusters between 10 and 100. Convergence was rapidly reached after $<$1000 iterations. The number of clusters was also stable at $\sim$40 after this time (Fig. 3). Using overdispersed starting conditions (Gelman *et al*., 1995), we determined that running the RJMCMC for 10 000 iterations after a burn-in period of 1000 iterations gives sufficient accuracy and performance.

Each dataset was generated with and without missing values, and specificity and sensitivity of the RJMCMC algorithm were evaluated and compared with the *k*-means algorithm. The RJMCMC-based clustering has a high specificity irrespective of whether the data were complete or had 10% missing values. High specificity is important, since genes in the same cluster will be postulated as similar in function or involved in the same biological process and further analysis will be based on this assumption. However, the ability to detect true positives (sensitivity) was only $\sim$0.5. This was still much higher than that of the *k*-means algorithm, which showed a similar specificity.

More work is needed for the labeling problem (Richardson and Green, 1997) and for inference of clusters from the posterior. Our solution may not be optimal.

6 CONCLUSIONS

In this contribution, we account for multiple replicates, a variable number of clusters and missing data. Previous Bayesian clustering algorithms did not impute missing values within the probabilistic model (Yeung *et al*., 2003). Instead, external programs were used to impute the missing values, e.g. weighted *k*-nearest neighbors, KNNimpute, (Troyanskaya *et al*., 2001). Hence, we overcame the main two problems of most clustering algorithms: the existence of missing values and the need to fix the number of clusters in advance. In addition, no filtering of the 'irrelevant' genes is needed and the biologist has the opportunity to focus only on the profiles of interest, as shown with the cdc15 experiment (Spellman *et al*., 1998).

Although it is clear that some of the genes regulated by a common element or active in a certain pathway are likely to be co-expressed across the different biological conditions measured with a microarray experiment, the reciprocal may not be true. It is, hence, difficult to assess the effectiveness of a clustering algorithm on real data. For this reason the performance of the method was further tested in simulated data: specificity was very high, sensitivity was $\sim$0.5, much higher than that of the *k*-means algorithm. The method adjusted the right number of clusters dynamically and this did not influence the clusters formation. The main drawback of the method is still its computational expenses. Parallel computing could improve the performance.

ACKNOWLEDGEMENTS

This work was supported by the BBSRC, the Austrian GEN-AU project BIN (Bioinformatics Integration Network) and the EU Marie Curie Training Site grant Genomics of Lipid Metabolism.

Conflict of Interest: none declared.

REFERENCES

Baldi,P. and Long,A.D. (2001) A Bayesian framework for the analysis of microarray expression data: regularized *t*-test and statistical inferences of gene changes. *Bioinformatics*, **17**, 509–519.

Beaumont,M.A. and Rannala,B. (2004) The Bayesian revolution in genetics. *Nat. Rev. Genet.*, **5**, 251–261.

Dougherty,E.R. and Brun,M. (2004) A probabilistic theory of clustering. *Pattern Recogn.*, **37**, 917–925.

Eisen,M.B. *et al*. (1998) Cluster analysis and display of genome-wide expression patterns. *Proc. Natl Acad. Sci. USA*, **95**, 14863–14868.

Gelman,A. *et al*. (1995) *Bayesian Data Analysis*, Chapman and Hall, London.

Golub,T.R. *et al*. (1999) Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, **286**, 531–537.

Green,P.J. (1995) Reversible jump Markov Chain Monte Carlo computation and Bayesian model determination. *Science*, **82**, 711–732.

Huber,W. *et al*. (2002) Variance stabilization applied to microarray data calibration and to the quantification of differential expression. *Bioinformatics*, **18** (Suppl 1), S96–S104.

Ihaka,R. and Gentleman,R. (1996) R: a language for data analysis and graphics, *J. Comput. Graph. Stat.*, **5**, 299–314.

Medvedovic,M. and Sivaganesan,S. (2002) Bayesian infinite mixture model based clustering of gene expression profiles. *Bioinformatics*, **18**, 1194–1206.

Ramoni,M.F. *et al*. (2002) Cluster analysis of gene expression dynamics, *Proc. Natl Acad. Sci. USA*, **99**, 9121–9126.

Richardson,S. and Green,P.J. (1997) On Bayesian analysis of mixtures with an unknown number of components. *J. R. Stat. Soc.*, **59**, 731–792.

Schwarz,G. (1978) Estimating the dimension of a model. *Ann. Stat.*, **6**, 461–464.

Segal,E. *et al*. (2003) Decomposing gene expression into cellular processes. *Pac. Symp. Biocomput.*, 89–100.

Sorlie,T. *et al*. (2003) Repeated observation of breast tumor subtypes in independent gene expression data sets. *Proc. Natl Acad. Sci. USA*, **100**, 8418–8423.

Spellman,P.T. *et al*. (1998) Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomices cerevisae* by microarray hybridization. *Mol. Biol. Cell*, **9**, 3273–3297.

Sturn,A. *et al*. (2002) Genesis: cluster analysis of microarray data. *Bioinformatics*, **18**, 207–208.

Tamayo,P. *et al*. (1999) Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation. *Proc. Natl Acad. Sci. USA*, **96**, 2907–2912.

Tavazoie,S. *et al*. (1999) Systematic determination of genetic network architecture. *Nat. Genet.*, **22**, 281–285.

Troyanskaya,O. *et al*. (2001) Missing value estimation methods for DNA microarrays. *Bioinformatics*, **17**, 520–525.

Yeung,K.Y. *et al*. (2001a) Model-based clustering and data transformations for gene expression data. *Bioinformatics*, **17**, 977–987.

Yeung,K.Y. *et al*. (2001b) Validating clustering for gene expression data. *Bioinformatics*, **17**, 309–318.

Yeung,K.Y. *et al*. (2003) Clustering gene-expression data with repeated measurements. *Genome Biol.*, **4**, R34.

Zhou,X.J. *et al*. (2005) Functional annotation and network reconstruction through cross-platform integration of microarray data. *Nat. Biotechnol.*, **23**, 238–243.

W498–W503 *Nucleic Acids Research, 2006, Vol. 34, Web Server issue*
doi:10.1093/nar/gkl038

CARMAweb: comprehensive R- and bioconductor-based web service for microarray data analysis

Johannes Rainer[1,3], Fatima Sanchez-Cabo[1], Gernot Stocker[1], Alexander Sturn[1] and Zlatko Trajanoski[1,2,*]

[1]Institute for Genomics and Bioinformatics and [2]Christian-Doppler Laboratory for Genomics and Bioinformatics, Graz University of Technology, Petersgasse 14, 8010 Graz, Austria and [3]Tyrolean Cancer Research Institute, Innrain 66, 6020 Innsbruck, Austria

Received November 9, 2005; Revised December 23, 2005; Accepted January 10, 2006

ABSTRACT

CARMAweb (Comprehensive R-based Microarray Analysis web service) is a web application designed for the analysis of microarray data. CARMAweb performs data preprocessing (background correction, quality control and normalization), detection of differentially expressed genes, cluster analysis, dimension reduction and visualization, classification, and Gene Ontology-term analysis. This web application accepts raw data from a variety of imaging software tools for the most widely used microarray platforms: Affymetrix GeneChips, spotted two-color microarrays and Applied Biosystems (ABI) microarrays. R and packages from the Bioconductor project are used as an analytical engine in combination with the R function Sweave, which allows automatic generation of analysis reports. These report files contain all R commands used to perform the analysis and guarantee therefore a maximum transparency and reproducibility for each analysis. The web application is implemented in Java based on the latest J2EE (Java 2 Enterprise Edition) software technology. CARMAweb is freely available at https://carmaweb.genome.tugraz.at.

INTRODUCTION

Expression profiling using microarrays has become a widely used method for the study of gene-expression patterns. Different microarray technologies have become available, including the Affymetrix GeneChip platform (http://www.affymetrix.com), spotted two-color cDNA or oligo microarrays (1), or the ABI single-channel microarrays (Applied Biosystems, http://www.appliedbiosystems.com). All microarray platforms require analytical pipelines with modules for (i) data preprocessing including data normalization, (ii) statistical analysis for identification of differentially expressed genes, (iii) cluster analysis and (iv) Gene Ontology (GO) analysis. The module for normalization and data preprocessing is platform dependent and aims to reduce technical variability without altering the biological variance in the data. After data normalization, the selection of differentially expressed genes is often the main objective of a microarray experiment. Additionally, genes might be grouped into clusters according to the similarity of their expression patterns. Finally, genes can be mapped onto GO (2) terms in order to get an overview of the biological processes, cellular components or molecular functions for which the genes of interest might be involved.

In the past years, Bioconductor (3) (based on the statistical programming language R, http://www.R-project.org) has become the reference tool for the analysis of microarray data because it is based on the most complete set of up-to-date algorithms. However, for scientists without adequate programming experience, the command line usage of R and Bioconductor is too cumbersome. Moreover, the performance of laboratory desktop computers is often insufficient to analyze microarray data with tens of thousands of features. Therefore, many analysis tools with a graphical user interface and powerful computing servers have been developed, including web-based tools like GEPAS (4), ArrayPipe (5), MIDAW (6), RACE (7) or Expression Profiler (8). Of these, only GEPAS and Expression Profiler support both Affymetrix and two-color arrays. To the best of our knowledge, there is currently no web service available for the analysis of the increasingly popular ABI system. MIDAW and RACE use R and Bioconductor packages as analytical engines as well, but these web applications focus either on the analysis of two-color

*To whom correspondence should be addressed. Tel: +43 316 873 5332; Fax: +43 316 873 5340; Email: zlatko.trajanoski@tugraz.at

microarrays (MIDAW) or Affymetrix GeneChips (RACE). Presently, only Expression Profiler allows loading microarray data from the ArrayExpress database (9). ExpressionProfiler enables direct handling only for raw data from the Affymetrix platform, whereas for two-color microarrays external manipulation of the raw data files has to be performed. The raw data files derived from the image analysis software are usually large and difficult to handle, especially for inexperienced users. Thus, researchers working with two-color microarray data have to navigate several websites and transfer the data between the servers to complete their analyses.

We have therefore developed the web application CARMAweb (Comprehensive R-based Microarray Analysis web service) based on both the latest Java 2 Enterprise Edition (J2EE) software technology and R in combination with Bioconductor.

CARMAweb provides the following unique features:

- Support for Affymetrix, two-color and ABI microarrays,
- Import of raw data from a variety of imaging software tools for two-color microarrays (Agilent Feature Extraction, ArrayVision, BlueFuse, GenePix, ImaGene, QuantArray, SPOT or raw data files from the Stanford Microarray Database),
- A complete analytical pipeline for Affymetrix, two-color and ABI microarrays including modules for preprocessing, detection of differentially expressed genes, clustering and visualization, as well as GO mapping,
- Generation of comprehensive analysis report files.

METHODS

CARMAweb is designed as a multi-tier application based on the J2EE environment, including Java Server Pages and Servlets for the web tier, and Enterprise Java Beans for the middle tier. With the exception of the module for cluster analysis, visualization and classification, all calculations are performed in R using functions of the Bioconductor packages. The connection between Java and R is established through Rserve (http://stats.math.uni-augsburg.de/Rserve/). Each analysis is processed in R using the R function *Sweave* (http://www.ci.tuwien.ac.at/~leisch/Sweave). *Sweave* is a tool that allows embedding of R code into LaTeX documents. *Sweave* executes the R commands from the input file, which is created by the web application. Output from R, R commands and descriptive text are written into a LaTeX file. Thus, code, results and descriptions are presented in a consistent way. After the analysis the LaTeX file generated by *Sweave* is transformed into a pdf analysis report file. These analysis report files contain all R commands used to perform the analysis, together with descriptions for the various methods used. This guarantees a maximum of transparency and reproducibility of each analysis performed in CARMAweb. The CARMAweb user guide gives a short introduction to the various analysis methods available in the web application. Test datasets are provided for each microarray platform.

The current implementation of CARMAweb runs on a server equipped with two AMD Opteron (64 bit CPU) processors and 4 GB of physical memory. CARMAweb will be updated regularly to the newest R and Bioconductor releases. The current version of CARMAweb uses R version 2.2 and Bioconductor release 1.7.

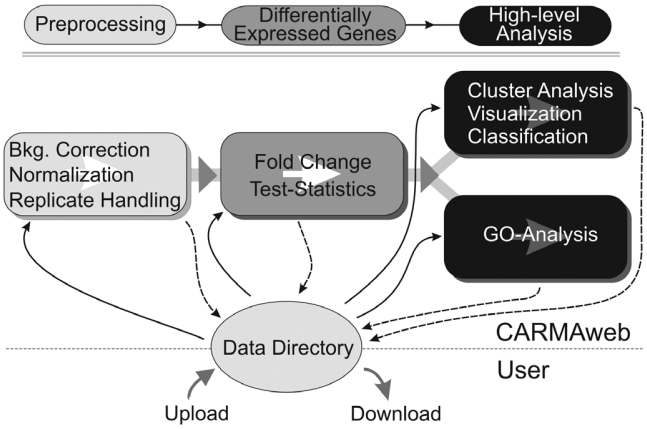


Figure 1. CARMAweb analysis workflow. The different modules of CARMAweb can either be used individually or in combination, resulting in an analytical pipeline. Analysis result files can be returned to the user's data directory and then be used as input for the other modules (e.g. the GO analysis module).

PROGRAM DESCRIPTION

The design and modular conception of CARMAweb allows the use of the different analysis modules either individually or combined into an analytical pipeline (Figure 1). After preprocessing of the raw data and identification of differentially expressed genes, cluster analysis, visualization and GO analysis can be performed. All analysis result files, i.e. tables with normalized expression values, differentially expressed genes or cluster analysis results can be returned to the users' data directory and subsequently used as input files for other analysis modules of CARMAweb or for other applications. Detailed descriptions and help texts for the various processing steps and methods of the different modules are provided as pop-up tool tips.

Data preprocessing

Data preprocessing is an essential step in the analysis of microarray data. The user has to choose an appropriate method from a wide range of available methods depending on the particularities of the data, i.e. their biological characteristics and the platform used.

Two-color microarrays. A large number of image analysis tools is available for two-color microarrays, and several features essential for the data preprocessing (i.e. flags determination, background estimates) differ between them. CARMAweb allows importing raw data files from Agilent Feature Extraction, ArrayVision, BlueFuse, GenePix, ImaGene, QuantArray, SPOT or raw data files from the Stanford Microarray Database. For background correction, CARMAweb allows several options (i) background subtraction, (ii) background subtraction followed by the minimum method (any intensity which is zero or negative after correction is set to half the minimum of the positive corrected intensities), (iii) the moving minimum method (background

estimates are replaced with the minimum of the backgrounds of the spot and its eight neighbors, and are then subtracted from the foreground) or (iv) the method described in (10). After background correction, methods like the median normalization, the loess or print tip loess normalization or the robust spline normalization (normalizes using robustly fitted regression splines and empirical Bayes shrinkage) are provided by CARMAweb to normalize within-array. Afterwards between arrays normalization can be performed using the median scaling or the quantile method. Additionally the variance stabilizing normalization (11), which combines both within and between array normalization, has also been included in CARMAweb. Most of these preprocessing methods are outlined in (12). The preprocessing of two-color microarrays is carried out in CARMAweb using functions from Bioconductors *limma* and *vsn* packages.

Affymetrix GeneChips. Preprocessing of Affymetrix GeneChips generally consists of the following steps: (i) background correction, (ii) normalization, (iii) correction for non-specific binding and (iv) summarization, where the measured probe intensities are averaged to one expression value per probe set. CARMAweb uses the *affy* package from Bioconductor for this purpose, and allows the usage of methods like the Affymetrix MAS5 algorithm or even more sophisticated methods like RMA (robust multi-array average) (13,14) or GCRMA (modified version of RMA that uses probe sequence information for the background correction) to perform the preprocessing. A comparison of the different Affymetrix preprocessing methods is outlined in (15). Additionally it is possible to define custom preprocessing methods by selecting different algorithms for each one of the preprocessing steps. For Affymetrix GeneChip analyses, Affymetrix raw data files (CEL files) are used as input files.

ABI microarrays. The module for the ABI microarray preprocessing supports tabulator-delimited text files, which can be exported from ABIs scanning software. These text files already contain background-corrected expression values from one or more microarrays. CARMAweb permits reading of microarray data from one or more of such exported text files, and allows the adjustment of raw (background-corrected) expression values across all microarrays of one experiment using quantile normalization. Alternatively, the assay-normalized signal provided by ABI might be used for the analysis (see Applied Biosystems 1700 Chemiluminescent Microarray Analyzer User's manual http://www.appliedbiosystems.com). Quality parameters (flags, signal to noise, cv) can be used to filter out poor quality spots.

Following the microarray preprocessing, replicated arrays can be averaged in an optional replicate handling step. This function also allows averaging of the replicated spots within each microarray, and its aim is to increase the quality of the microarray data by reducing the noise.

Detection of differentially expressed genes

The detection of differentially expressed genes can be performed in CARMAweb for microarray experiments with a small number of biological replicates using a simple fold change. Additionally CARMAweb allows ranking of genes according to the number of comparisons in which they were selected as differentially expressed.

In microarray experiments with a sufficient number of arrays, differentially expressed genes can be detected in CARMAweb using statistical tests like the Mann Whitney U test (16), the Student's *t*-test (16), the permutation (randomization) test (16), the moderated *t*-statistics (based on an empirical Bayes approach, the Bioconductors *limma* package) (17) or the significance analysis of microarrays (SAM, Bioconductors *siggenes* package) (18). Because microarray experiments generate large multiplicity problems in which thousands of hypotheses are tested simultaneously within one experiment (19), an adjustment of the calculated *P*-values should be performed. Bioconductors *multtest* package provides suitable methods to adjust *P*-values regarding this multiple hypothesis testing problem. Available adjustment methods are the procedure introduced by Benjamini and Hochberg (20) for strong control of the FDR (false discovery rate, expected proportion of false positives among the rejected hypotheses) or the method by Westfall and Young (21) to control the FWER (family-wise error rate, probability of at least one false positive). CARMAweb allows the use of all methods described in (19) for the adjustment of raw *P*-values.

To alleviate the loss of power from the formidable multiplicity of gene-by-gene hypothesis testing within a microarray experiment, a non-specific pre-filtering of the data can also be performed. This pre-filtering consists in the reduction beforehand of the number of genes to be tested, removing those that are either not relevant for the study in question or those expected to be unaltered through the experimental conditions. This can be achieved by focusing the analysis only on those genes for which variance across conditions is in the top $x\%$, where x is a user-defined value.

Cluster analysis, dimension reduction and visualization, and classification

For cluster analysis, dimension reduction and visualization, and for classification of microarray data, the module GenesisWeb can be used (Figure 2). This module is based on the cluster analysis suite Genesis (22), and uses its server (23) to perform the calculations. Cluster analysis and visualization requires intensive graphical user interaction that is not supported by R. The cluster analysis module of CARMAweb supports an interactive selection, coloring and export of clusters, and also displays other important information like the expression values or gene names both as tool tips and in the status bar of the browser when the user moves the mouse over the image.

Expression data can be adjusted beforehand with methods like mean or median centering, logarithmic transformation or division by the SD across samples or genes. Genes and/or samples can be grouped according to their expression similarity using the hierarchical clustering algorithm (HCL) (24), the *k*-means clustering method (KMC) (25) or self-organizing maps (SOM) (26). A wide range of distance-measurement methods is available to measure the similarity of gene or sample expression patterns (e.g. Euclidian distance, Pearson correlation, Spearman's rank or Kendall's tau). As mentioned before all result images are interactive, thus allowing the selection, coloring and export of clusters. Additional information, like gene names or expression values, is displayed both as

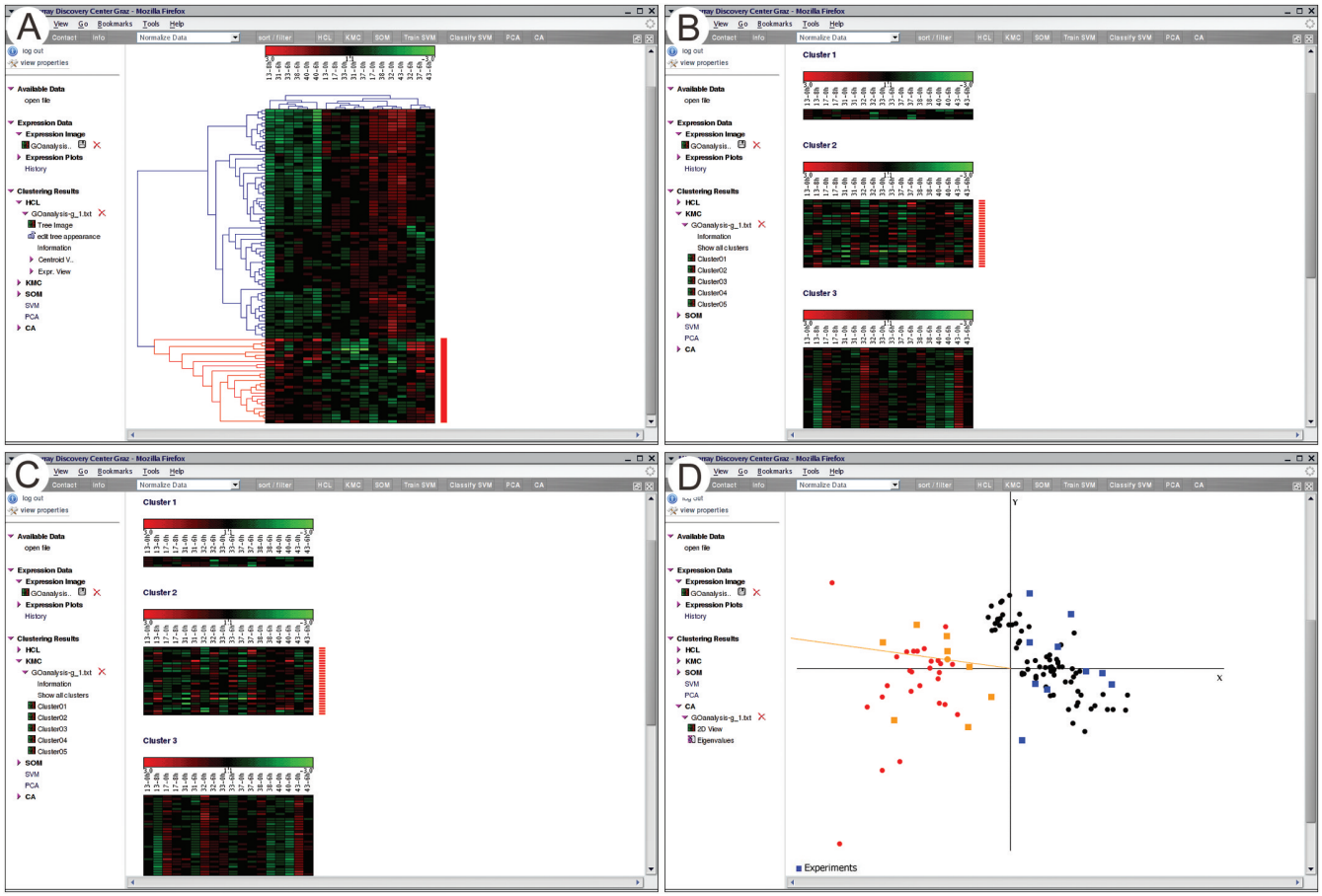


Figure 2. The cluster analysis module GenesisWeb offers interactive cluster selection. (**A**) Result from a hierarchical cluster analysis. (**B**) Result from *k*-means cluster analysis of the same dataset. (**C**) Result from SOM cluster analysis. (**D**) Visualization of a CA of the same dataset. Clusters interactively selected in any of the cluster analyses can be highlighted in further analyses (shown here as red labeled genes).

tool tips and in the status bar of the browser when the user moves the mouse over expression or cluster images.

The available dimension reduction and visualization methods are principal component analysis (PCA) (27) and correspondence analysis (CA) (28). Whereas PCA can be used to identify key variables (or combination of variables) in the datasets, CA allows simultaneous detection of dependencies between samples and genes in microarray data . Visualization tools available with the dimension reduction procedure also enable selection, coloring and export of genes that group together in the space spanned by the principal components.

Support vector machines (SVM) (29) can be used for classification of microarray data. The aim of this supervised classification method is to classify genes or samples by using the information gathered from the training on a dataset with known classification. For example, an SVM can learn in the training step what expression features are specific for a given functional group of genes specified by the researcher, and use this information to decide whether any given gene is likely to be a member of the group or not.

The tool can be used as a standalone web application at https://carmaweb.genome.tugraz.at/genesis, or in combination with CARMAweb, where it is possible to return cluster analysis results to the user's data directory. As input files, tabulator-delimited text files containing expression values (e.g. from an earlier analysis that detected differentially expressed genes, or from files uploaded by the user) are supported.

GO analysis

The Gene Ontology project (2) provides three independent ontologies for gene products. The three ontologies refer to the cellular component, biological process and molecular function of a gene product and allow its description in a hierarchical manner. The GO analysis aims to assist in the biological interpretation of the results by finding GO terms that are significantly often associated to genes in a given gene list. CARMAweb uses the *GOstats* and *GO* package from Bioconductor for the GO term analysis. The GO term analysis module of CARMAweb supports as input any tabulator-delimited text file that contains one column with EntrezGene (formerly LocusLink) identifiers of the genes of interest. This kind of input file can be either file uploaded by the user or result file from a previous analysis. The result of the GO analysis is a GO graph, and a table with GO terms and *P*-values calculated for over-representation of the genes in the corresponding GO terms. The GO graph is the collection of unique GO terms

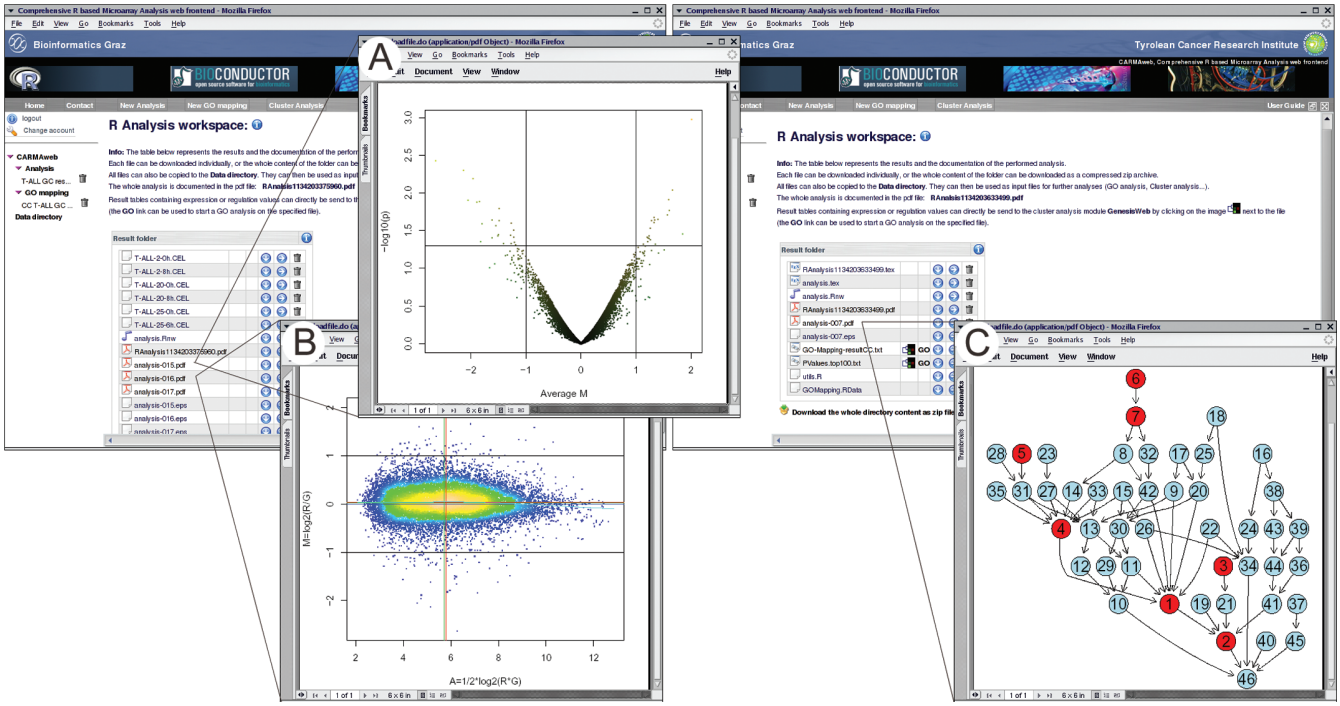


Figure 3. Result workspaces of a differentially expressed genes analysis (left) and a GO analysis (right). (**A**) Volcano plot displaying the mean differential expression against *P*-values (−log10 of the *P*-value) of all genes. (**B**) MA plot. Points are colored according to local point density with brighter colors coding for higher density. (**C**) The induced GO graph of the genes of interest. Red nodes represent over-represented GO terms.

that are associated with one or more of the genes of interest. In order to allow calculation of *P*-values, an additional file containing the EntrezGene identifiers of all genes that can be detected with the microarrays in use needs to be submitted. Affymetrix users can specify the GeneChip used in the analysis instead of submitting a file with all EntrezGene identifiers. Although some correction for multiple testing should be performed on the *P*-values, such tests are not independent and the sampling distribution is unclear (30), so CARMAweb at present does not perform any correction.

Output and analysis results

Each analysis is processed in its own workspace, which is accessible only to the user performing the analysis (Figure 3). The analysis result includes all raw data files, the analysis report file containing all commands and descriptions about the methods used, and all result tables and plots created during the analysis. Additionally the R workspace of each analysis step can be exported to an *RData* file, which can be used to continue the analysis in R on a local workstation. The result tables can comprise tables of normalized expression values for all genes in all arrays, tables with expression values of the subsets of differentially expressed genes, or tables containing the raw *P*-values and adjusted *P*-values using the various adjustment methods. In an Affymetrix GeneChip analysis all probe sets are annotated to the identifiers of the publicly available databases (GenBank (31), UniGene, EntrezGene) using the Bioconductor *annaffy* package. Analysis result files can be returned to the users' data directory and be used as input for further analyses.

Visualizations of the microarray data like MA plots, histograms, box plots or volcano plots are available as single files and are additionally embedded into the analysis report file. The content of each analysis workspace can be downloaded after completion as a single zip archive, or each file can be downloaded separately. The GO term analysis produces a directed acyclic graph of all GO terms to which the genes of interest are associated (Figure 3). Additionally a table containing all GO terms with the corresponding *P*-value is created. The *P*-values provide information about the over-representation of the genes of interest to the term compared with the total number of genes associated with it. The table contains the number of genes of interest that are mapped to the specific term and the total number of genes present on the microarray in use that are associated with the GO term.

Future development

The next release of CARMAweb will provide a complete SOAP (Simple Object Access Protocol) interface to its analysis facilities, thus allowing other applications to use the analysis and processing steps available in CARMAweb.

CONCLUSIONS

The web application CARMAweb that we have developed combines the advantages of an intuitive web-based graphical user interface with the wide range of state-of-the-art microarray normalization and analysis methods provided by Bioconductor. Owing to the modular structure of CARMAweb and the standards-based software engineering, extensions or new

functionalities can be implemented easily without complex and time-consuming alterations of the source code.

CARMAweb provides several unique features in a modular and flexible system for the analysis of microarray data. First, data from three platforms, namely Affymetrix GeneChip, two-color microarray and the ABI microarray platform can be analyzed. Second, a wide range of file formats for two-color microarray raw data is supported. Third, a complete analytical pipeline for the supported platforms is provided, including preprocessing, detection of differentially expressed genes, cluster analysis, dimensionality reduction and visualization, classification, and GO analysis. Fourth, data exploration is enhanced by analysis report files that include the parameters and commands used. The report files that are generated specifically for each analysis guarantee a maximum of transparency and reproducibility. Furthermore, these report files provide a unique way for the documentation of any analyses that have been performed by recording how and with which methods the analysis results have been derived. In sum, based on its flexibility in selecting different analysis steps, its possibility for customization and its comprehensive web-based graphical user interface, CARMAweb is a powerful tool for microarray data analysis.

ACKNOWLEDGEMENTS

We thank Reinhard Kofler and Stefan Schmidt from the Tyrolean Cancer Research Institute for discussions and providing test datasets available in CARMAweb. We also thank Hubert Hackl and Robert Molidor from the Institute of Genomics and Bioinformatics for discussions and comments, and James McNally from the National Cancer Institute, National Institutes of Health for comments on the manuscript. This work was supported by the bm:bwk project GEN-AU BIN (Bioinformatic Integration Network) and the Tyrolean Cancer Research Institute. Funding to pay the Open Access publication charges for this article was provided by bm:bwk project GEN-AU BIN.

Conflict of interest statement. None declared.

REFERENCES

1. Schena,M., Shalon,D., Davis,R.W. and Brown,P.O. (1995) Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, **270**, 467–470.
2. Gene Ontology Consortium. (2001), Creating the gene ontology resource: design and implementation. *Genome Res.*, **11**, 1425–1433.
3. Gentleman,R.C., Carey,V.J., Bates,D.M., Bolstad,B., Dettling,M., Dudoit,S., Ellis,B., Gautier,L., Ge,Y., Gentry,J. *et al.* (2004) Bioconductor: open software development for computational biology and bioinformatics. *Genome Biol.*, **5**, R80.
4. Herrero,J., Al-Shahrour,F., Diaz-Uriarte,R., Mateos,A., Vaquerizas,J.M., Santoyo,J. and Dopazo,J. (2003) GEPAS: a web-based resource for microarray gene expression data analysis. *Nucleic Acids Res.*, **31**, 3461–3467.
5. Hokamp,K., Roche,F.M., Acab,M., Rousseau,M.-E., Kuo,B., Goode,D., Aeschliman,D., Bryan,J., Babiuk,L.A., Hancock,R.E.W. *et al.* (2004) ArrayPipe: a flexible processing pipeline for microarray data. *Nucleic Acids Res.*, **32**, W457–W459.
6. Romualdi,C., Vitulo,N., Del Favero,M. and Lanfranchi,G. (2005) MIDAW: a web tool for statistical analysis of microarray data. *Nucleic Acids Res.*, **33**, W644–W649.
7. Psarros,M., Heber,S., Sick,M., Thoppae,G., Harshman,K. and Sick,B. (2005) RACE: Remote Analysis Computation for gene Expression data. *Nucleic Acids Res.*, **33**, W638–W643.
8. Kapushesky,M., Kemmeren,P., Culhane,A.C., Durinck,S., Ihmels,J., Korner,C., Kull,M., Torrente,A., Sarkans,U., Vilo,J. *et al.* (2004) Expression Profiler: next generation—an online platform for analysis of microarray data. *Nucleic Acids Res.*, **32**, W465–W470.
9. 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.
10. Edwards,D.E. (2003) Non-linear normalization and background correction in one-channel cDNA microarray studies. *Bioinformatics*, **19**, 825–833.
11. Huber,W., von Heydebreck,A., Sultmann,H., Poustka,A. and Vingron,M. (2002) Variance stabilization applied to microarray data calibration and to the quantification of differential expression. *Bioinformatics*, **18**, 96–104.
12. Smyth,G.K. and Speed,T.P. (2003) Normalization of cDNA microarray data. *Methods*, **31**, 265–273.
13. Irizarry,R.A., Hobbs,B., Collin,F., Beazer-Barclay,Y.D., Antonellis,K.J., Scherf,U. and Speed,T.P. (2003) Exploration, normalization, and summaries of high density oligonucleotide array probe level data. *Biostatistics*, **4**, 249–264.
14. Irizarry,R.A., Bolstad,B.M., Collin,F., Cope,L.M., Hobbs,B. and Speed,T.P. (2003) Summaries of Affymetrix GeneChip probe level data. *Nucleic Acids Res.*, **31**, e15.
15. Bolstad,B.M., Irizarry,R.A., Astrand,M. and Speed,T.P. (2003) A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics*, **19**, 185–193.
16. Motulsky,H. (1995) *Intuitive Biostatistics*. Oxford University Press, Oxford.
17. Smyth,G.K., Yang,Y.H. and Speed,T. (2003) Statistical issues in cDNA microarray data anlalysis. *Methods Mol. Biol.*, **224**, 111–136.
18. Tusher,V., Tibshirani,R. and Chu,G. (2001) Significance analysis of microarrays applied to the ionizing radiation response. *Proc. Natl Acad. Sci. USA*, **98**, 5116–5124.
19. Dudoit,S., Shaffer,J.P. and Boldrick,B.J. (2003) Multiple hypothesis testing in microarray experiments. *Stat. Sci.*, **18**, 71–103.
20. Benjamini,Y. and Hochberg,Y. (1995) Controlling the false discovery rate: a practical and powerful approach to multiple testing. *J. R. Stat. Soc. Ser.*, **57**, 289–300.
21. Westfall,P.H. and Young,S. (1993) *Resampling-Based Multiple Testing: Examples and Methods for P-Value Adjustment*. Wiley, NY.
22. Sturn,A., Quackenbush,J. and Trajanoski,Z. (2002) Genesis: cluster analysis of microarray data. *Bioinformatics*, **18**, 207–208.
23. Sturn,A., Mlecnik,B., Pieler,R., Rainer,J., Truskaller,T. and Trajanoski,Z. (2003) Client-server environment for high-performance gene expression data analysis. *Bioinformatics*, **19**, 772–773.
24. Eisen,M.B., Spellman,P.T., Brown,P.O. and Botstein,D. (1998) Cluster analysis and display of genome-wide expression patterns. *Proc. Natl Acad. Sci. USA*, **95**, 14863–14868.
25. Tavazoie,S., Hughes,J.D., Campbell,M.J., Cho,R.J. and Church,G.M. (1999) Systematic determination of genetic network architecture. *Nature Genet.*, **22**, 281–285.
26. Tamayo,P., Slonim,D., Mesirov,J., Zhu,Q., Kitareewan,S., Dmitrovsky,E., Lander,E.S. and Golub,T.R. (1999) Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic di erentiation. *Proc. Natl Acad. Sci. USA*, **96**, 2907–2912.
27. Yeung,K.Y. and Ruzzo,W.L. (2001) Principal component analysis for clustering gene expression data. *Bioinformatics*, **17**, 763–774.
28. Fellenberg,K., Hauser,N.C., Brors,B., Neutzner,A., Hoheisel,J.D. and Vingron,M. (2001) Correspondence analysis applied to microarray data. *Proc. Natl Acad. Sci. USA*, **98**, 10781–10786.
29. Brown,M.P., Grundy,W.N., Lin,D., Cristianini,N., Sugnet,C.W., Furey,T.S., Ares,M.J. and Haussler,D. (2000) Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proc. Natl Acad. Sci. USA*, **97**, 262–267.
30. Gentleman,R., Scholtens,D., Ding,B., Carey,V.J. and Huber,W. (2005) Case Studies Using Graphs on Biological Data. In Gentleman,R., Carey,V.J., Huber,W., Irizarry,R.A. and Dudoit,S. (eds), *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*. Springer, NY, pp. 375–378.
31. Benson,D.A., Karsch-Mizrachi,I., Lipman,D.J., Ostell,J. and Wheeler,D.L. (2005) GenBank. *Nucleic Acids Res.*, **33**, 34–38.