# BIOINFORMATICS PLATFORM FOR METABOLIC MODEL DEVELOPMENT

**STEPHAN PABINGER**


## DOCTORAL THESIS

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

Graz, March 2010

## EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am …….  ……..
(Unterschrift)

Englische Fassung:

## STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

…….  ……..
date  (signature)

# Abstract

The overall goal of systems biology is the simultaneous study of all processes and dynamic interactions at molecular level in order to draw conclusions which may not be apparent when only individual components are considered. Genome-scale modeling is a promising approach to systematically analyze complex cellular systems. Metabolic models have proven to be valuable for increasing the product yield, predicting the effect of gene deletions, improving gene annotation, and identifying regulatory mechanisms. As models are constructed based on annotated genomes, current advancements in next generation sequencing technologies will foster the development of new models.

Therefore, a bioinformatics platform for the management, storage, and development of metabolic models has been established. The web based *Me*tabolic *Mo*del *Sys*tem (MEMOSys) supports the development of new models by providing a built in version control system which offers access to the complete reconstruction history. Moreover, the integrated web board, the fine-grained authorization system, and the definition of user roles allow collaborations across departments and universities. Research on existing models is facilitated by a powerful search system, references to external databases, and a feature-rich comparison mechanism. MEMOSys provides customizable data exchange mechanisms using the SBML format to enable analysis in external tools. The web application is based on the Java EE framework and offers an intuitive user interface. It currently contains several well annotated and publicly available models.

In summary, the implemented bioinformatics platform provides researchers a novel application facilitating the management and development of metabolic models.

**Keywords:** Metabolic models, database, SBML, Java EE

# Publications

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

**Pabinger S**, Thallinger GG, Snajder R, Eichhorn H, Rader R, Trajanoski Z. QPCR: Application for real-time PCR data management and analysis. BMC Bioinformatics 2009, 10:268 PMID: 19712446

Rader R, **Pabinger S**, Ramsauer T, Specht T, and Trajanoski Z. Integrating bioprocess and gene expression data for optimizing industrial fermentation. submitted

**Pabinger S**, Rader R, Agren R, Nielsen J, Trajanoski Z. MEMOSys: Metabolic model research and development system. in preparation

# Contents

# Chapter 1

# Introduction

## 1.1 Systems Biology

With the assembly of the first whole genome sequences in the mid-1990s [Fleischmann et al., 1995], it became possible to identify all gene products involved in biological processes of an organism [Palsson, 2009]. Until then biological research was focused on the characterization of individual components - one at a time. This reductionistic approach shifted to large-scale studies which allow a systematical and concurrent analysis of multiple components [Snyder and Gallagher, 2009].

Systems Biology aims at simultaneously studying all processes and dynamic interactions at molecular level in order to define entire pathways and predict the behavior of investigated systems [Kay and Wren, 2009]. The molecular activity of cell components is strongly interconnected and needs to be investigated in a whole-system approach to explain physiological characteristics and dynamic behavior [Eils and Kriete, 2005a]. Large-scale analyses allow drawing meaningful conclusions which may not be apparent when only individual components are considered. Since the smallest known genome has about 450 highly connected genes [Fraser et al., 1995] and knock-out studies have shown that in an artificial organism approximately 375 of them are essential for life [Glass et al., 2006], it is of great importance to study the whole system instead of concentrating on a few components to explain physiological processes. Moreover, the genome of established model organisms such as *Saccharomyces cerevisiae* and *Escherichia coli* contain at least ten times more genes than the artificial minimal organism which makes a whole system approach even more essential [Westerhoff et al., 2009].

Advancements in several high-throughput technologies have given researchers the possibility

to investigate gene and protein expressions in large-scale studies. The recent introduction of next generation sequencing platforms considerably reduces the analysis time and financial effort which allows even small laboratories to sequence entire genomes. The task at hand is now to integrate the large amount of data about biological systems into models in order to gain novel insights into their interconnected functionality [Kay and Wren, 2009].

## 1.2 Genome-scale metabolic models

Genome-scale modeling is a promising approach to systematically analyze multiple high-throughput results of complex cellular systems [Selvarasu et al., 2010]. Based on the annotated genome sequence and biochemical data a metabolic network can be reconstructed which describes the relationship between genes and phenotypes. A metabolic network can be considered as the sum of all chemical reactions of a particular system [Palsson, 2006]. It tries to assess the physiological states of an organism and aims at identifying their biochemical implementation in terms of metabolic fluxes. The modeled system requires that production and consumption of all substrates and products are in balance and uses exchange fluxes to model the excretion and uptake of metabolites [Durot et al., 2009]. Due to the growing number of sequenced organisms, genome-scale metabolic models have been compiled for a large number of organisms including the model organisms *e. coli* and yeast, several different microorganisms, the human organism, and almost 20 bacterial species.

In order to analyze a compiled model, it needs to be converted into a mathematical representation where reversibility and rate limiting constraints can be employed (see figure 1.1). The resulting stoichiometric matrix contains all metabolic reactions and comprises coefficients representing time- and condition-invariant properties of the network [Palsson, 2006]. After the initial reconstruction step, manual fostering and systematic verification of the predictive capabilities are performed in order to correct model inconsistencies.

Since the initial reconstruction may contain an incomplete set of reactions, gap-filling methods [Osterman and Overbeek, 2003; Vongsangnak et al., 2008] have been developed to insert missing reactions or link unconnected metabolites to the network. These methods are essential for the creation of genome-scale models as they can explain differences between experimental data and computational predictions.

Mapping high-throughput data onto the curated genome-scale model can be useful to gain new biological insights as it provides a new way to analyze omics data against the knowledge about the target organism. Several studies map transcriptomic data, proteomic data, or tissue-specific expression-profiling data onto genome-scale models [Palsson, 2009]. The incorporation

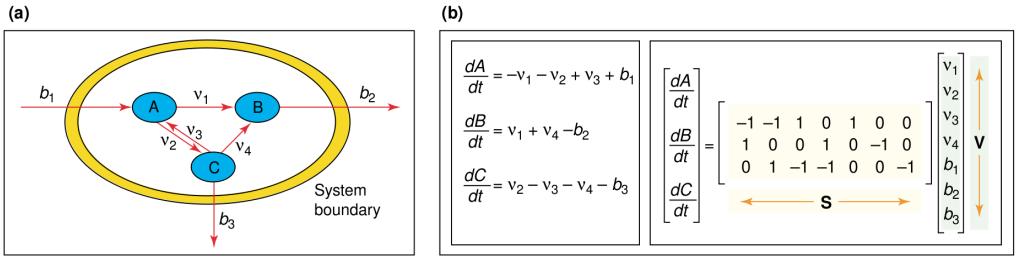


**Figure 1.1:** Figure (a) depicts a model system containing three metabolites (A, B, C), three exchange fluxes (b1, b2, b3), two unidirectional internal reactions (v1, v4), and one reversible reaction (v2, v3). (b) For each metabolite a mass balance equation is constructed which can also be represented in matrix form where **S** is the stoichiometric matrix and **V** is the matrix of all fluxes [Kauffman et al., 2003].

of additional constraints such as gene regulation, osmolarity, or electroneutrality will be a future step to further expand the scope of network-based analyses. Genome-scale metabolic models have already proven to be valuable for strain engineering which aims at improving production yield and stability [Lee et al., 2005; Nielsen and Jewett, 2008; Wendisch et al., 2006]. Their ability to predict the outcome of gene deletions and prognosticate the adaption of an organism to new nutritional environments makes them a useful instrument to determine the characteristics of alternative flux distributions [Papin et al., 2003]. Future applications of metabolic models will contribute to the understanding of bacterial genomes and may lead to better diagnostic tests and therapies of human diseases [Westerhoff and Palsson, 2004].

### 1.2.1 Simulation

The increasing number of metabolic models resulted in the development of numerous methods and toolboxes ([Becker et al., 2007; Heino et al., 2010; Hoops et al., 2006; Klamt et al., 2006; Sauro et al., 2003]) to analyze large-scale metabolic networks.

#### Elementary modes

Elementary modes are a set of vectors derived from the stoichiometric matrix that describe the smallest sub-networks which are still functional in steady state. They are calculated by using convex analysis and are helpful to understand cellular objectives of the metabolic model. Each network has a unique set of elementary modes where removing any reaction from an elementary mode would result in a non-functional unit [Schuster et al., 1999].

### Extreme Pathways

Extreme pathways represent the smallest unique set of pathways to accurately interpret the functional aspects of a metabolic network [Schilling et al., 2000].

Similar to elementary modes, extreme pathways are a set of convex basic vectors derived from the stoichiometric matrix. Each network possesses a unique set of pathways which are essentially subsets of elementary modes. The reversibility of reactions is treated differently by elementary modes and extreme pathways: the former uses a set of rules to include reversibility into the elementary modes whereas the latter decouples all internal reversible reactions into two separate, opposing reactions. As a result, certain elementary modes can be described as a linear combination of extreme pathways which correspond to the edges of the convex solution space in a biochemical network [Papin et al., 2004].

### Minimal metabolic behaviors

A recently introduced method for the analysis of metabolic networks termed “Minimal metabolic behaviors” characterizes a network by its minimal metabolic behaviors and the reversible metabolic space. Minimal metabolic behaviors are uniquely determined by the network and are based on an outer description of the solution space in contrast to the inner description used by elementary modes or extreme pathways. The method uses a refined classification of reactions according to their reversibility type, as the reversibility provides the key to explain reaction dependencies. On the one hand reversible reactions define the reversible metabolic space containing useful biological information, and on the other hand irreversible reactions completely characterize the minimal metabolic behaviors [Larhlimi and Bockmayr, 2009].

### Flux balance analysis

Growth is a fundamental principle of life that organisms fulfill by taking up nutrients and converting them to molecules through a set of chemical reactions called metabolism. Flux balance analysis (FBA) uses stoichiometric information about metabolic pathways to determine the theoretical capabilities of a metabolic system [Edwards et al., 2001]. It proposes a system where only metabolites that leave or enter the network are considered as exchange fluxes. Thermodynamic and capacity properties are included by constraining the flux value of both enzyme and exchange reactions. Since cell growth and process dynamics are typically substantially slower than metabolic changeovers, FBA assumes a steady-state of metabolic mass balances. It is therefore required that the production of every metabolite inside the network is equal to its consumption.

The hitherto specified properties describe a solution space that contains all feasible steady-

state flux vectors that satisfy the imposed constraints. In order to calculate a single solution, FBA uses linear programming to find the flux vector that maximizes or minimizes a proposed objective function. Usually the objective function is defined as the production of biomass which equals growth of an organism but may also be any other objective such as ATP production [Pramanik and Keasling, 1997]. FBA is able to incorporate additional information that add new constraints to the network to further reduce the size of the solution space. Since stoichiometric parameters can be defined from the annotated genome sequence, FBA is particularly applicable for post-genomic analysis.

The existence of a wide range of applications tremendously contributed to the success of FBA as an analysis tool for metabolic networks. Flux balance analysis can be used to assist the reconstruction of metabolic pathways by comparing predicted reaction fluxes to measured results. It facilitates metabolic engineering by calculating the most efficient pathway through the network in order to achieve a particular objective. Moreover, FBA is used to predict the outcome of gene deletion studies and to investigate metabolic capabilities of cellular systems.

Although widely used, the assumption, that the evolutionary history of an organism has selected the pathways with the highest biomass yield, is a source for potential errors. It is unclear whether the organism is actually capable of achieving that optimum, and external conditions that were present during its evolution could differ from the conditions defined by the researcher. Furthermore, it is questionable whether the criterion of maximum efficiency is at all important for the organisms under consideration [Westerhoff and Palsson, 2004].

### 1.2.2 Software tools

A crucial part in the construction of metabolic models based on annotated genome sequences is the determination of gene-enzyme and enzyme-reaction relationships [Ma and Zeng, 2003]. Enzymes are proteins that catalyze chemical reactions and link genes to specific metabolic reactions. In order to provide a unique and consistent representation of an enzyme, they are assigned to an Enzyme Commission classification number (EC number). Several enzyme databases have been published, such as the KEGG [Kanehisa, 1997], BRENDA [Chang et al., 2009], and ExPASy enzyme nomenclature database [Gasteiger et al., 2003]. In addition to basic information they contain descriptions about genes which are used to determine the gene-enzyme relationships.

Names of metabolites are often inconsistent between different models, which makes them nearly impossible to compare. The ChEBI database [Degtyarenko et al., 2008] and the KEGG Compound Database [Goto et al., 2002] assign unique identifiers to metabolites to facilitate the comparison of different models. Moreover, they list all known synonyms of a metabolite and provide structure and chemical information.

The reconstruction of a new model can be supported by performing homology comparisons with already existing networks. Several repositories have been created which provide access to established metabolic models (e.g.: BioModels [Novère et al., 2006], BIGG [BIGG, 2010]). Additionally, there are many databases available containing a wealth of information about specific organisms which are helpful for assembling a new network: EcoCyc [Karp et al., 2000] for *Escherichia Coli*; SGD [Engel et al., 2010] for *Saccharomyces cerevisiae*; MetaCyc [Caspi et al., 2010] and Dogan [Dogan, 2010] for multiple organisms.

Although a lot of databases and web applications allow researchers to download and query metabolic models, non of them provides support for the tedious construction process. Assembling the final version of a model is an iterative process involving many manual steps that generate several intermediate versions. The possibility to review all changes, extract previous versions, and use an innovative user interface to create new entries would greatly enhance the construction of new models.

## 1.3 Objectives

The aim of this thesis is to develop a web-based bioinformatics platform for managing, developing, and storing metabolic models of microbial organisms. Furthermore, it should facilitate the collaborative construction of new metabolic models and include a modular system for easy data exchange.

The specific goals are:

- design a database for all components of metabolic models
- build a user friendly web interface for the management of models
  - implement a flexible query system
  - support standardized data import and export
  - provide links to external databases
- implement features for the collaborative development of models
  - define fine grained user access levels
  - provide a supervision mechanism
  - implement an automatic version control system
- import available microbial models into the application

The software should be built using state-of-the-art software technologies to be scalable and extensible and allow an easy adaption and extension of future requirements.

# Chapter 2

# Methods

This chapter gives insight into various methods used to develop the presented application. The main focus has been laid on the description of frameworks, tools, and specifications of the Java programming language. Future versions and features of the Java platform are discussed by the Java Community Process (JCP) [JCP, 2009] where several different companies, organizations, and individuals participate. It defines formal Java Specification Requests (JSRs) to draft and outline new specifications. The chapter describes several parts of the Java programming language and concludes with the description of used programming libraries and development tools.

## 2.1 Java Enterprise Edition

### 2.1.1 Overview

The Java™ Platform, Enterprise Edition (Java EE) is an umbrella specification intended to facilitate the development of powerful, robust, distributed, and highly available applications [Goncalves, 2009].

The release of the current Java EE6 (Shannon [2009]) in December 2009 marked the tenth anniversary of the Java Enterprise Edition platform containing 28 Java Specification Requests (JSRs). The first version J2EE 1.2 was released by Sun in 1999 and contained only ten JSRs.

With the release of J2EE 1.4 in 2003 [Shannon, 2003] the platform became widely accepted in the Java community and was aimed at providing better support for application development and deployment. However, due to its heavyweight component model, J2EE 1.4 is difficult to test, deploy, and run and is usually considered as too complex and too complicated to learn.

“Simplify the development of web service applications” [Shannon, 2006] was the focus of Java

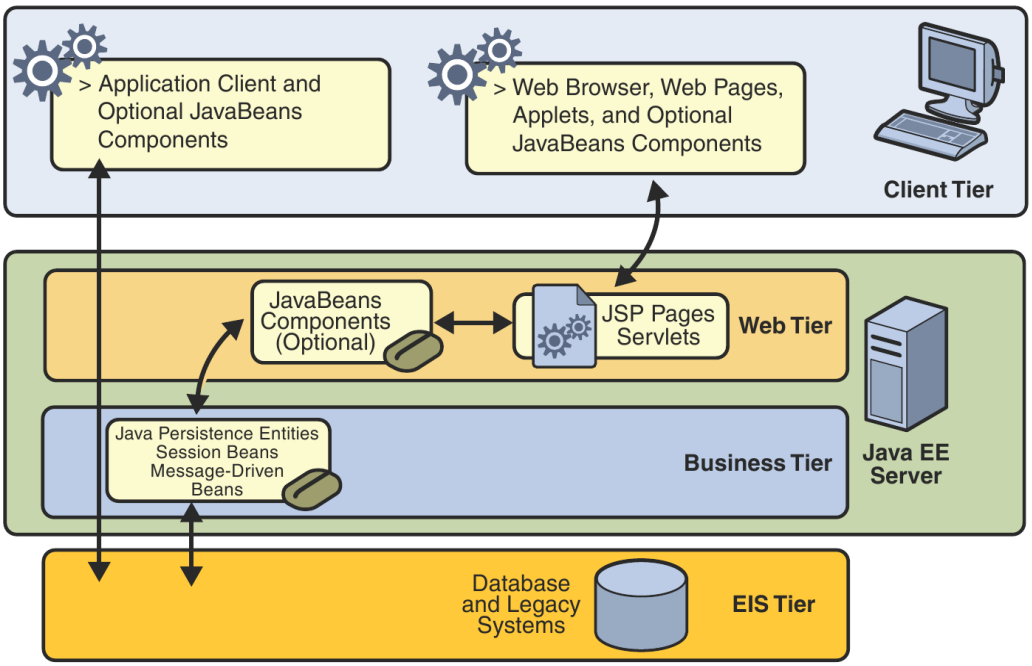


**Figure 2.1:** Overview of the multi-tiered platform Java EE (from Ball et al. [2006])

Enterprise Edition 5 (Java EE5) released in 2006. It represents a remarkable improvement over the previous version by refining several different aspects. The use of annotations moved the definition of metadata from external XML deployment descriptor files into the source code. A new application programming interface (API) for the easy creation of web services supported Service-Oriented Architecture (SOA) which is considered the “next phase in the evolution of business automation” [Erl, 2006]. Applying the “convention over configuration” paradigm reduces the configuration effort by implying intelligent default values. Therefore, the framework gains simplicity but still provides a high level of flexibility.

The current version Java EE 6 [Shannon, 2009] continues the “ease of development” approach by expanding the use of annotations. A major part of the new specification is the definition of profiles that include only parts of the platform in order to be adaptable to specific requirements. Additionally, the expert group has introduced a pruning process to remove or replace features that are not needed anymore: the EJB 2 persistent model has been replaced by JPA which was already introduced in Java EE5; JAX-RPC has been replaced by JAX-WS which is more robust and easier to use.

### 2.1.2 Architecture

Enterprise systems using the Java EE architecture are designed as distributed multi-tiered applications. Figure 2.1 depicts the different levels and shows their interactions. The specification defines the following tiers [Ball et al., 2006; Johnson, 2002]:

- **Client Tier** components are executed on the client machine by either a web client or an application client and handle the communication with the user.
- the **Java EE Tier** manages Java EE components which are deployed on the Java EE server. They are usually assigned to two sub-tiers:
  - the **Web Tier** handles communication with web clients.
  - the **Business Tier** manages business processes and access to the EIS tier.
- the **Enterprise Information System Tier (EIS)** manages data persistence by using Database Management Systems (DBMS).

The Java EE architecture organizes business logic into reusable components and uses containers for providing services for each component type. The following containers are defined:

- **Java EE server:** contains Enterprise JavaBeans and web containers.
- **Enterprise JavaBeans Container:** is compliant to the EJB specification and manages data and business model beans.
- **Web Container:** is responsible for the execution of web framework technologies (e.g.: JavaServer Faces, Java Servlet).
- **Application Client Container:** runs on the client side and manages the execution of client components.
- **Applet Container:** is responsible for the management of applets which are usually executed in a web browser using a Java plug-in.

The Java EE specification has been implemented by several Java application servers which differ in their consistency to the defined standard.

The most prominent representatives of application servers implementing the Java EE 5 standard are: the reference implementation Glassfish [Glassfish, 2010]; the fully certified and open source implementation Apache Geronimo [GeronimoAS, 2010], Java Open Application Server JOnAs [JOnAS, 2010], and JBoss Application Server [JBossAS, 2010]; the commercially available IBM WebSphere [WebSphere, 2010] and Oracle WebLogic [OracleAS, 2010]. Currently the

Java EE 6 specification is implemented only by the reference implementation Glassfish [Glassfish, 2010], but within the next year, several of the above mentioned application servers will be updated.

Java EE containers support developers by providing low level services like transaction management, resource pooling, and bean handing. The most common provided services and APIs are:

- **Java Transaction API** manages transactions for database connections, provides rollback support, and an auto commit mode.
- **Java Messaging Service (JMS) API** enables messaging and is used to create, send, receive, and read messages.
- **Java Mail API** is used to send mails.
- **Java API for XML Web Services (JAX-WS)** defines support for web services (see 2.1.4).
- **Java Persistence API (JPA)** [DeMichiel, 2009] manages persistence by using an object-relational mapping approach. The current version has been inspired by object relational mapping tools like Hibernate [Hibernate, 2010a] or TopLink [TopLink, 2010] and developed into a native Java API as a standard for object persistence. EJB 3 entities (see 2.1.3) are administered by an entity manager, which handles persistence and retrieval of objects.

  The Java Persistence API consists of three areas:

  – The Java Persistence core
  – The JPA Query Language (JPA QL) supporting aggregate functions, grouping, joins, and subqueries
  – Object/relational mapping metadata

  The JPA specification has been implemented by several different companies including Hibernate, EclipseLink [EclipseLink, 2010], and TopLink, which can be used as the persistence backend for Java EE applications. The current version JPA 2.0 adds several new features that include mapping of collections of simple data types, maintaining a persistence order, pessimistic locking, and orphan removal [Goncalves, 2009].
- **Java Naming and Directory Interface (JNDI)** is used to register and retrieve data, resources, and any type of Java objects.
- **Java Authentication and Authorization Service (JAAS)** provides user- or group-based authentication and authorization.

### 2.1.3 Enterprise JavaBeans

The Enterprise JavaBeans (EJB) [Saks, 2009] specification describes a framework which offers a model for building server-side components in distributed business applications [Monson-Hafael and Burke, 2006]. EJBs are portable components that encapsulate the business logic of an application. In order to fulfill their purpose they need to run inside an EJB container, which provides system-level services such as transactions, instance pooling, persistence, and security authorization. Since developers can rely on the container to provide all these services, the framework greatly simplifies the development of large, distributed applications.

The release of EJB 3.0 [DeMichiel and Keith, 2006] is targeted at reducing the complexity of the framework by providing a simplified API. Through the use of annotations as metadata it reduces the number of compulsory classes and interfaces and avoids the excessive use of deployment descriptors. Moreover, the design facilitates object orientated concepts by using annotated POJOs and dependency injection, and simplifies object persistence by specifying an object-relational mapping facility.

EJB 3.1 [Saks, 2009] introduces an embeddable API for running EJB components within a Java SE environment. It offers improved packaging, a simplified local view, enhanced EJB Timers, and allows the creation of Singleton session beans.

The current specification defines three main types of enterprise beans:

- **Entity Beans** are used for data persistence and retrieval in relational databases. Each Entity Bean is usually mapped to a database table where each record represents one persisted object. Since an Entity Bean is an annotated POJO, object orientated concepts can be applied and beans are able to exist outside the EJB container.
- **Session Beans** are server-side components that manage business processes and contain most of the business logic. They access other beans to retrieve data and call business methods.

  Two types of Session Beans can be distinguished:

  - **Stateful Session Beans** (SFSBs) maintain their conversational state across method calls. A SFSB is assigned to one client and keeps its state in memory until the client removes the bean or terminates. This allows caching of database or calculation intensive results to the cost of an increased memory consumption.
  - **Stateless Session Beans** (SLSBs) do not preserve a conversational state for the client. Each method call is completely independent from its predecessors and uses

only data passed by function call parameters. Since SLSBs are not assigned to specific clients, they offer great scalability for applications requiring a large number of clients.

- **Message Driven Beans** (MDBs) are enterprise beans which process messages asynchronously. They expose one public method which gets executed when receiving a message, and usually are invoked by JMS messages sent by any Java EE component. MDBs are useful when processing long lasting business jobs that should be decoupled from the user interface.

### 2.1.4 Web services

For several years Service-Oriented Architecture (SOA) has been part of day-to-day architectural life providing the ability to connect a wide variety of heterogeneous systems to achieve open interoperability [Pulier and Taylor, 2006]. It is an architecture based upon web service technology which can be implemented in most programming language using different technologies [Goncalves, 2009].

Web services describe business methods of applications which use interfaces to provide a standard way to connect diverse pieces of software over the Internet. JAX-WS [Kotamraju, 2006b] is a technology used in Java EE for building web services and clients that communicate over the web by using XML. Typically web service communication is encoded using an XML-based protocol such as the Simple Object Access Protocol (SOAP) [SOAP, 2003]. The SOAP specification defines an envelope structure, encoding rules, and conventions for representing web service invocations and responses. In order to hide the complexity of SOAP messages from the developer, JAX-WS converts the API calls and responses to and from SOAP messages using the Java Architecture for XML Binding (JAXB) [Kotamraju, 2006a]. JAXB provides a fast and convenient way to marshal and unmarshal Java object trees into XML documents.

With the rise of Web 2.0 RESTful web services have gained in popularity. Java EE 6 introduced REST with the Java API for RESTful Web Services (JAX-RS) [Hadley and Sandoz, 2009]. REpresentational State Transfer (REST) is an architectural style for distributed hypermedia systems based on the HTTP protocol. Data and functions are considered as resources and are accessed using Uniform Resource Identifiers (URIs).

### 2.1.5 JavaServer Faces

JavaServer Faces (JSF) [Burns, 2006b] is a user interface (UI) component framework for Java web applications. It is designed to make the development of applications, running on a Java application server, easier and faster. User interfaces can be easily constructed by using a set of

UI components, and custom UI components can be quickly built and re-used. JSF binds the UI components on a web page to a server-side representation which helps managing the UI state across server requests. Furthermore, it provides a simple mechanism to propagate events of web clients to the server. Since the JSF APIs are layered directly on top of the Servlet API it is easy to replace the view handler. The JavaServer Faces technology allows a clean separation of presentation and behavior which enables interface designers and code developers to concentrate on their respective parts.

JSF 2.0, published in July 2009 [Burns, 2009], improved many aspects of the previous specification. The most prominent ones are listed below:

- **Asynchronous JavaScript and XML (AJAX) support** - The request processing lifecycle has been expanded to be aware of AJAX and partial tree traversal has been introduced.
- **Partial Page Saving** - Instead of saving the complete state every time, JSF 2.0 stores only the state changes and restores the state by applying the modifications to the initial state. As a result, state saving is more efficient and easier to use.
- **Improved configuration** - JSF 2.0 introduces annotation-based configuration which reduces the amount of XML based configuration files.
- **Support for Representational State Transfer (REST)** [Hadley and Sandoz, 2009]
- **Bookmarkable URLs** - An improved GET support allows storing of generated URLs as fully functional bookmarks.
- **Facelets** - Usage of Facelets (see page 17) instead of JSP.

The JSF specification is implemented by Sun (reference implementation) [JSF RI, 2010] and the Apache MyFaces project [MyFaces, 2010].

#### Request processing lifecycle

JSF requests follow a well defined lifecycle which includes several phases (illustrated in figure 2.2):

1. **Restore View** - the tree of the UI components is built or restored if the page has already been requested.
2. **Apply Request** - components are updated to their current state using the information included in the current request. Moreover, queued events will be delivered to subscribed listeners.

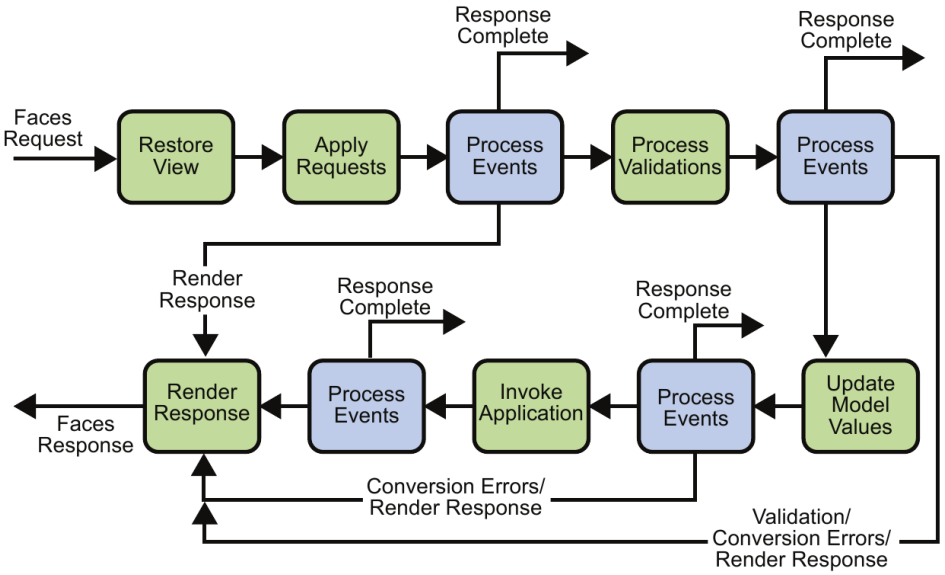


**Figure 2.2:** JSF request lifecycle (from Ball et al. [2006])

3. **Process Validation** - components with a registered validator are controlled for their correctness. If errors occur, JSF advances directly to the *render response phase*.
4. **Update Model Values** - JSF updates the server-side object properties with the submitted values.
5. **Invoke Application** - JSF executes application-level events that include form submissions, action calls, or page transitions.
6. **Render Response** - the state is saved for subsequent requests and the response is rendered and transferred to the client.

#### User interface component model

JavaServer Faces components are the basic elements for building a JSF user interface. Components are configurable and reusable building blocks ranging from simple buttons to complex multiple component compositions such as tables or tree views. They can be linked to corresponding objects in the data model of an application using value expressions (see page 16). On the server side JSF pages are represented as a component tree where each component can be accessed by a unique identifier. Moreover, JSF allows the definition of custom flexible user interface components which benefit from several helper APIs [Burns, 2006b]:

- **Converters** - Data in a web application is stored at two different locations: the server-side *model view* stores data as Java objects while the *presentation view* is limited to a String representation. A converter manages the conversion between these views. In addition to the existing standard converters (for most built-in Java data types), JSF provides the possibility to add user defined converters.
- **Events and Listeners** - The event and listener model is based on the design patterns outlined in the JavaBeans specification [JavaBeans, 2010]. Events sent by UI components are being queued by JSF and broadcasted to registered listeners.
- **Validators** - JSF supports a mechanism to validate client side data before the server-side model is updated. It allows the creation of pluggable support classes which ensure that the local value conforms to defined business rules. Validation failures generate error messages which can be displayed during rendering. Several standard validators are included in JSF to perform range and length checks. JSF 2.0 introduces support for Bean Validation [Bernard, 2010] that unifies validation from the persistence layer to the presentation layer.

#### Unified Expression Language

The Unified Expression Language allows the use of expressions in web pages in order to bind values, objects, and actions to components. JSF relies on the Expression Language (EL) which is part of the JavaServer Pages specification version 2.1 [Chung and Luehe, 2006]. The unified expression language represents a combination of the EL specified by JSP 2.0 [Pelegri-Llopart and Roth, 2003] and the EL created for JSF 1.0 [Burns, 2006a]. Since JSP has a simple lifecycle, the EL in JSP 2.0 supports only simple expressions to dynamically read data from JavaBeans. On the contrary, the multiphase JSF lifecycle supports conversion, update, validation, and event handling of its UI components. In order to meet the new requirements, the EL introduced in JSF 1.0 features the following functionality: (a) deferred evaluation of expressions, (b) the ability to get and set data, and (c) the ability to invoke methods.

The combination of existing ELs into one unified expression language enables mixing of JSP content with JSF tags, and allows new JSP-based technologies the use of the additional JSF features. The current version gives page authors the possibility to use expressions to perform the following tasks [Ball et al., 2006]:

- Dynamically read application data stored in JavaBean components, various data structures, and implicit objects
- Dynamically write data, such as user input, to JavaBean components

- Invoke arbitrary static and public methods
- Dynamically perform arithmetic operations

### Facelets

Facelets is a powerful open source web framework providing a highly performant JSF view handler. It is heavily inspired by the Tapestry framework [Tapestry, 2010] and uses concepts introduced by Apache Tiles [Tiles, 2010]. Facelets provides full support for all JSF UI components and builds its own component tree reflecting the view of a JSF application. Since all web pages must be valid XML documents, Facelets fully supports namespaces and is able to auto-wire additional artifacts to XML documents such as UIComponents, Validators, and Converters.

JSF 1.2 has been put on top of JSP and although both technologies have been improved to complement each other, they never achieved seamless interoperability [Bergsten, 2004]. Facelets uses JSF instead of the JSP API as the default view handler and supports templating for components and pages as a core feature. This allows an easy definition of layouts including header, footer, and navigation menus. Avoiding the need to compile pages into Servlets, as it is done in JSP, greatly improves the performance of JSF web applications. The dreadful search for errors in JSP pages has been improved by providing precise error reports that include line, tag, and attribute information. Facelets fully supports EL expressions and allows developers to create component compositions.

The current JSF specification 2.0 includes Facelets as the default view handler.

### Component libraries

Component libraries provide a set of reusable components for the JavaServer Faces framework. Since the release of JSF, a huge number of projects have been created [JSFMatrix, 2010]. Some of the most widely know and usually very advanced and professionally designed libraries have been created by Apache [Trinidad, 2010], ICEsoft [ICEfaces, 2010], and JBoss [RichFaces, 2010]. Component libraries allow developers to focus on top level UI design by providing support for different web browsers and applying a consistent behavior and style for all included components.

RichFaces [RichFaces, 2010] is an open source component library by JBoss which fully integrates into the JSF lifecycle. It adds AJAX capability into existing JSF components which allows modifying the view without the need to completely reload pages. The library provides extensive skinning support for all its components including windows, panels, data tables, and toggle controls.

## 2.2 Relational data persistence

Persisting data in an organized and structured way is one of the basic aims of business applications. In order to achieve this goal relational database management systems (RDBMS) are used as the persistence backend in enterprise applications. The principal ideas of RDBMS were already introduced in 1970 by Edgar F. Codd [Codd, 1970] and are still used in today's commercial and open source database systems. A RDBMS allows administrators to create and delete databases, define their structure, control data access, and manipulate the containing data. The basic idea is to map data to normalized tables where attributes are stored in table fields, and individual objects are persisted as entities identified by unique keys. Tables can be connected with each other by referencing their unique identifiers using foreign keys. This system allows the definition of one-to-one, one-to-many, and many-to-many relations. In order to guarantee data integrity and avoid redundancy and manipulation anomalies, the concept of normalization was introduced [Codd, 1970, 1990]. Over the course of several years, six *normal forms* have been stated which describe the mathematical rules for normalized relational databases. For common databases it is usually sufficient to apply the third normal form and informally these databases are considered as *normalized* [Rob et al., 2008]. In some cases the expanding number of tables, caused by the normalization process, may result in a considerable loss of system speed. Consequently, special circumstances force some degree of denormalization in order to increase the performance.

In 1986 the American National Standards Institute (ANSI) introduced the Structured Query Language (SQL) designed for managing data in a relational database. Since then the language has undergone several update revisions and is currently available as version SQL:2008. SQL defines three different sublanguages:

- **Data Definition Language (DDL)** - for defining data structures
- **Data Manipulation Language (DML)** - to create, insert, update, delete, and query data
- **Data Control Language (DCL)** - to control data access

Java provides access to a wide range of databases using the Java Database Connectivity API [JDBC, 2010]. It allows developers to query and manipulate data by calling SQL functions. Each supported database management system provides its own specific back-end implementation which is accessed by the JDBC interface. This allows an easy exchange of the underlying RDBMS and hides the complexity of the individual implementations from the developer.

### 2.2.1 Hibernate

Although JDBC provides a comfortable and standardized way to submit SQL commands in Java, it does not offer a mechanism to directly persist objects in database tables. Object-Relational Mapping (ORM) describes a method that solves the discrepancies between the object-oriented programming model and the relational database tables. Each ORM implementation needs to address object oriented concepts and has to specify ways to map Java objects to relational database tables [Johnson, 2002].

Hibernate [Hibernate, 2010a] is a powerful, widely spread open source ORM framework for the Java programming language. It allows developers to design object oriented persistent classes supporting inheritance, polymorphism, and composition. Hibernate can be configured using annotations or XML based descriptor files and creates, based on the configuration, RDBMS optimized SQL statements. It implements the JPA specification and can be used as a standalone or EJB 3 persistence provider. Due to its huge popularity in the Java community, it had strong influence on the development of the JPA specification.

Hibernate offers a wide range of query options and optimizes object loading using advanced fetching and caching mechanisms. In addition to supporting the retrieval of objects by their primary key the framework provides several query types:

- **Native SQL queries** - results are either managed entities or simple scalars.
- **Hibernate Query Language (HQL)** - is a fully object-oriented query language supporting inheritance, polymorphism, and association. HQL is an extension of the Java Persistence query language (JPQL) including additional features such as fetch joins, pagination, projection, and sorting.
- **Hibernate Criteria** - is a powerful and elegant alternative to HQL for dynamic query generation. The API uses the properties of a passed object to define the query parameters.

The framework is divided into several modules. *Hibernate Core* includes the persistence API and the mapping of metadata stored in XML. The *Hibernate Annotations* module allows developers to use annotation based configuration instead of XML files. The standard Java Persistence API is implemented by *Hibernate EntityManager* along with the Java Persistence Query Language. Hibernate Annotations, Hibernate EntityManager, and the versioning framework Envers (see page 20) will be part of Hibernate Core in the upcoming version 3.5. Validation of entity beans is supported by the *Hibernate Validator* module which will be based on the Bean Validation

specification [Bernard, 2010]. *Hibernate Search* provides full-text indexed search functionality, and *Hibernate Shards* allows the distribution of data across multiple databases.

### 2.2.2 Envers

Envers [JBoss, 2010], developed by Adam Warski, is a framework that allows easy versioning of persistent classes and is seamlessly integrated into Hibernate. In addition to allowing versioning of all mappings defined by the JPA specification, Envers supports mapping of various custom types and collections.

For each entity, which has been marked as versioned, the framework creates an additional table to store its revisions. Entities can be labeled as versioned using annotations or XML descriptor files. Envers uses a unique revision number for all entities which allows retrieving a view of the database at a certain revision. Moreover, it provides a rich API to search for archived data which is similar to Hibernate Criteria [Hibernate, 2010b]. A drawback of the current implementation is the missing support for relation traversing which inhibits the creation of complex queries.

Envers creates revisions of entities only at the end of successful transactions to ensure the integrity of the persisted data.

## 2.3 JBoss Seam

JBoss Seam [Seam, 2010] is an open source framework which tries to integrate different technologies for creating rich web applications in Java [Allen, 2009]. Based on the Java EE platform it ties together JSF, JPA, EJB 3.0 and Business Process Management (BPM) into a unified solution. Moreover, Seam provides useful additions to the JavaServer Faces lifecycle and the unified expression language. It tries to reduce the complexity of developing web-based applications by increasing the use of annotations and providing a rich set of UI components. In addition, Seam includes a testing framework where tests can be run without starting the application server [Yuan and Heute, 2007]. Many ideas implemented in Seam had great influence on the design of Java EE standards, especially on the *Java Context and Dependency Injection* specification [King, 2009].

The essence of Seam is a unified component model for all business logic in an application. The platform adds its own component model on top of the JSF and EJB models and collects all of the Java EE components into a central registry. Every class can be declared as a Seam component usually being an EJB bean, a JavaBean, or a JPA entity. Seam components have to be generated by the Seam framework, which attaches interceptors to objects in order to apply much of its implicit logic. Class annotations can be used to specify non-conventional behavior.

Seam enables JSF UI components to communicate directly with the EJB layer by allowing EJB 3 components to act as JSF beans and action listeners. It makes no distinction between presentation and business logic components and provides EJB 3 components access to web-tier scopes. On the one hand the design avoids the need for a managed bean facade reducing complexity and descriptor files but on the other hand loosens the stringency of the separation of concerns between the business and the presentation layer.

Each Seam component is assigned to a scope which defines the lifetime of the component. In addition to the application, session, and request scope specified by the Java Servlet API and the event and page scope defined by JSF, Seam introduces three new scopes: stateless, conversation, and business process context.

The *stateless* scope is a hint for the container to freshly instantiate the component on each call.

The *conversation* scope is applied for processes that span over several user interactions including page requests or database transactions. Start and end points of a conversation are defined by the developer using either method annotations or propagation attributes of JSF components. Seam is able to simultaneously handle multiple conversations which avoids inconsistencies between data view and session state when using multiple browser windows.

The *business process context* is used for long running business processes which are controlled by the Java Business Process Management (JBPM). It can span over multiple interactions and may last several days or weeks.

### 2.3.1 Dependency injection

Dependency injection (DI) is a key concept to achieve loose coupling of components. It is an appliance of the Inversion of Control (IoC) pattern where the framework instead of the developer is in control of the workflow. The principle of IoC is known as the “Hollywood Principle” where the delivery of the response is out of the requestor’s hands.

In DI, the container takes care of injecting dependent objects into the target class. This system eliminates the need for lookup logic and allows easy testing of components as injected services can be simply substituted with mock-ups. Dependent objects are defined using annotations or XML based descriptor files [Buckley, 2003]. Although the introduction of DI has been a great improvement over the previous lookup mechanisms it still lacks some important features. Injected objects are not able to propagate changes back to the context where they have been initially retrieved from.

#### Bijection

Seam enhances dependency injection by introducing a mechanism termed *bijection*. It describes a two-way injection and outjection interaction between Seam components and the Seam managed context [Yuan and Heute, 2007]. Instead of injecting dependencies only at the object instantiation, Seam uses method interceptors to continuously synchronize contexts with object variables which allows injecting the current state of a variable on each method call. *Outjecting* describes the process of propagating the value of a component to a context where it can be used again by other components. Components can be injected and outjected by using the @In and @Out annotations. Since bijection relies on method interceptors which generate additional workload on each method call, it can be disabled for components to increase the performance.

### 2.3.2 Events

In addition to the extended dependency injection mechanism, Seam provides an event handling method which allows components to pass messages to one another without direct interaction. The Event API is based on the observer pattern introduced by Gamma et al. [1995] using Seam as the messaging mediator. Events, that are generated during any page-related activity or raised by a component, are propagated to the Seam container which looks for and notifies registered observers. The event handling method allows the development of loosely-coupled components and supports asynchronous operations which help developers to remove tight coupling and facilitates component testing.

### 2.3.3 A framework within a framework

Seam provides a collection of component templates for a variety of programming requirements, a “framework within a framework”, which is often referred to as the Seam Application Framework [Allen, 2009]. The *Home* component manages the create, read, update, and delete (CRUD) operations on an entity instance and caches it for further usage. This allows entity classes to remain POJOs which maintains the separation of concerns by introducing a loose coupling between the domain object and the persistence framework. Since the *Home* component is an abstract class acting as a template, Seam is providing implementations for JPA and Hibernate which can be further extended by the developer to add additional features.

The *Query* component helps developers to manage results of performed queries. Similar to the *Home* component it is only a template where Seam provides implementations for JPA and Hibernate. The component includes several useful functions like the possibility to define restriction parameters and supports sorting and pagination of results.

#### Seam-gen

To get quickly started with a new project Seam provides a utility to set up a basic Seam project which includes the generation of actions, entities, and query stubs. Furthermore, the Ant [Ant, 2010] controlled generator is able to reverse engineer an existing database.

### 2.3.4 User authentication and authorization

Providing proper security mechanisms is a key criteria in a multiuser application and involves two functions [Goncalves, 2009]:

- **Authentication** describes the process of verifying the user's identity against an authentication system and assigning a role to the user
- **Authorization** checks if a user has the permission to access a particular resource or function

The security framework of Seam is based on the Java Authentication and Authorization Service (JAAS) hiding most of the JAAS's complexity and offers an own identity management system. Authorization is built either on the *rule* based system JBoss Drools or on a *role* and permission based mechanism.

## 2.4 Development libraries and tools

### 2.4.1 SBML

The Systems Biology Markup Language (SBML) [SBML, 2010] is a computer-readable language based on XML for representing models of biological processes. It provides a common intermediate format that can be used to describe models in regulatory networks, signaling pathways, gene regulation networks, and metabolic pathways. The use of SBML offers several benefits:

- one representation of a model can be used without modification in multiple tools
- models can be shared and published in a common, interdisciplinary format
- models stored in SBML are valid beyond the lifetime of the software used to create them

The first definition of SBML was assembled in 2000 when it became clear that Systems Biology (see 1.1) needs a common model format to enable data exchange between software tools. Since the publication of the first version in 2003 [Hucka et al., 2003], SBML has undergone several

revisions and the current stable version is available as SBML Level 2 Version 4 [Hucka et al., 2010].

Due to the loose definition of the language, several tools specified more stringent SBML formats. Amongst others, the COBRA toolbox [Becker et al., 2007], a widely spread analysis application, defines a distinct SBML format that is required by the software.

The format developed by the consensus yeast reconstruction group [Herrgård et al., 2008] tries to unify existing interpretations into one common format. It uses the resource description framework (RDF) [RDF, 2010] and the "Minimum information requested in the annotation of biochemical models" (MIRIAM) [Novère et al., 2005] notation to annotate species with external references. Each MIRIAM identifier is a single unique string, which unambiguously references an entity or object in an external resource. The consensus yeast format supports modeling of reactions catalyzed by two or more isoenzymes by using separate reactions for each complex. This allows researchers to map any Boolean combination of enzymes in the SBML format.

#### LibSBML

LibSBML [Bornstein et al., 2008] is an open-source programming library that allows developers to read, write, manipulate, translate, and validate SBML files. The library is implemented in C and C++ and provides wrappers for a number of software languages, including Java, Python, Perl, and MATLAB.

### 2.4.2 JFreeChart

JFreeChart [Gilbert, 2010] is an open source chart library for the Java platform. It supports the creation of a wide range of different chart types which can be exported to PNG and JPEG images. Moreover, the charts are highly customizable and can include tool tips and annotations. JFreeChart can be easily extended and allows the implementation of user defined datasets and chart types.

### 2.4.3 Development Tools

The development of applications can be tremendously improved by using the appropriate tools which help developers to increase productivity and quality of their code production. Amongst others the following tools have been used throughout the development of the application:

### Eclipse

Eclipse [Eclipse, 2010] is an open source community hosting the Eclipse Software Development Kit (SDK) which is an integrated development environment (IDE) for Java.

The core application provides different perspectives, editors, and views and supports multiple user workspaces. Additionally, it features syntax highlighting, code completion, debugging, search, and navigation functionality. A flexible plug-in mechanism allows easy integration of custom extensions which contributed to the huge success of Eclipse. The commercial Java EE extension MyEclipse [MyEclipse, 2010] provides several tools for the development of business applications including additional debuggers for browsers and JSPs, a database explorer, and a controlling mechanism for application servers. The plug-in Subclipse [Subclipse, 2010] allows accessing Subversion code repositories.

### Subversion

Subversion (SVN) [Subversion, 2010] is an open source version control system. It manages files and resources and allows developers to track changes between revisions and revert to previous versions. Revisions are identified by a unique number and contain additional information like timestamps and user comments. Subversion can considerably ease the collaboration between software developers when working on a common project. It features the definition of different development branches and version tags and is capable of moving, renaming, copying, and merging files with full revision history support.

### MagicDraw

The visual UML modeling and Computer-Aided Software Engineering (CASE) tool MagicDraw [MagicDraw, 2010] facilitates the analysis and design of object oriented systems and databases. It provides database schema modeling, Data Definition Language (DDL) generation, a code engineering mechanism, and reverse engineering facilities. As the tool is implemented in Java it is platform independent and can be installed on various environments.

### Oracle SQL Developer

The Oracle SQL Developer [SQLDeveloper, 2010] is a free tool for the management of databases. It allows developers to browse databases, format SQL statements, run SQL statements and scripts, and export query results. Additionally, the software provides reports of the database status and supports versioning using the source control system Subversion.

# Chapter 3

# Results

## 3.1 MEMOSys - Metabolic Model System

The main objective of this thesis is to design and implement an application for managing, developing, and storing metabolic models of microbial organisms. A metabolic model is a set of cross-linked reactions which describe the transition of metabolites from reactants to products. Reactions can be assigned to subsystems and contain information about their proteins and regulating genes.

In order to accomplish the outlined objectives (see 1.3) the **Me**tabolic **Mo**del **Sys**tem MEMOSys has been developed. It is a database centric Java EE application that includes a rich web front-end, a version-control system, comparison functionality, and sophisticated data exchange mechanisms. The following pages describe the main features of the application.

### 3.1.1 Data model

MEMOSys has been designed to map and store all properties of a metabolic model in a database. Reactions are the essential part of a model and are characterized by their reactants and their reversibility. Each reactant contains a metabolite, defines a stoichiometry, and is assigned to a compartment. A compartment is a closed part within a cell, usually surrounded by a membrane, which allows the cell to maintain special environmental properties (such as pH or enzyme systems) to carry out different metabolic activities. Compartments are arranged in a hierarchy which is mapped to the database using references to the parent compartment.

A reaction can be assigned to a subsystem which is a representation of a certain metabolic pathway. To support users in the creation or adaption of reactions, MEMOSys provides a bal-

ance check mechanism that validates the stoichiometry equilibrium of consuming and producing reactants (see figure 3.2).

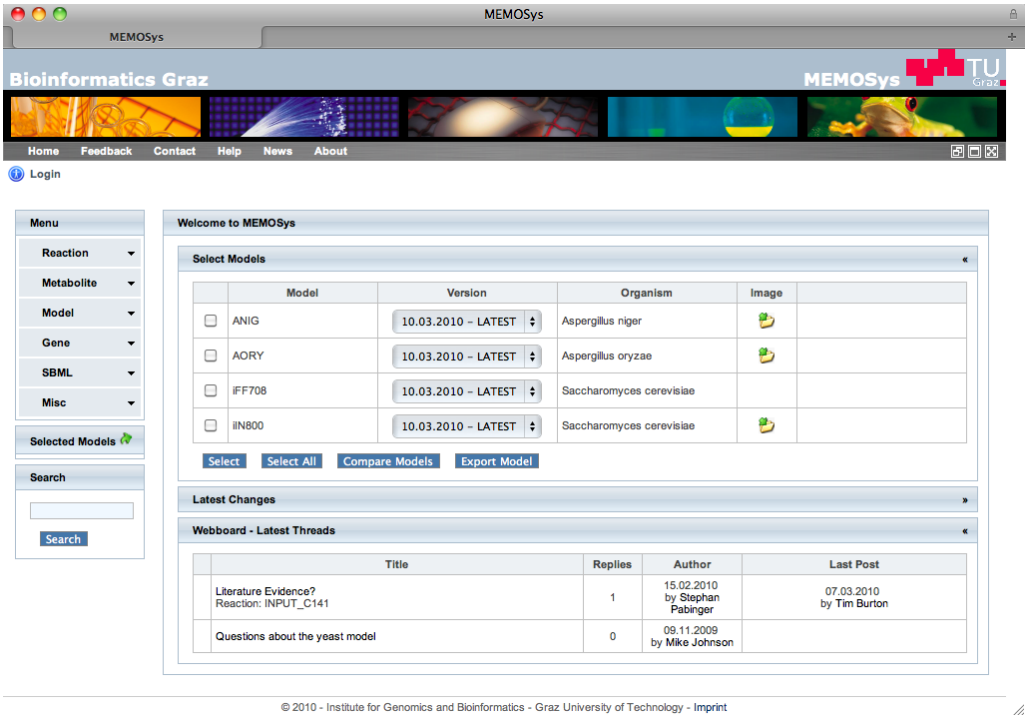


**Figure 3.1:** MEMOSys supports several different layouts which allow removing the image banner at the top and stretching the layout to use the full screen width. The depicted image shows the MEMOSys home screen where users can select models and versions to view and edit reactions. It displays the latest modifications of reactions and metabolites and shows the newest comments on the web board.

In addition to general properties like name and EC number, reactions can store citations to reference evidence in primary literature. Gene complex objects reflect the relationship of genes and reactions by using Boolean operators and hierarchical structures (e.g.: gene1 AND [gene2 OR gene3]). For genes having a reference to the UniProt database, the system provides a mechanism to download the amino acid sequence of the transcribed protein.

A model is defined by its reactions, specifies an organism, and may contain references to an image that graphically represents the metabolic map.

Compartments, metabolites, genes, and reactions provide several references to external databases using the MIRIAM notation. MEMOSys automatically transforms MIRIAM annotations to web addresses and displays links to the external references. Moreover, the application includes a mechanism to easily define additional external databases which can be referenced by components.

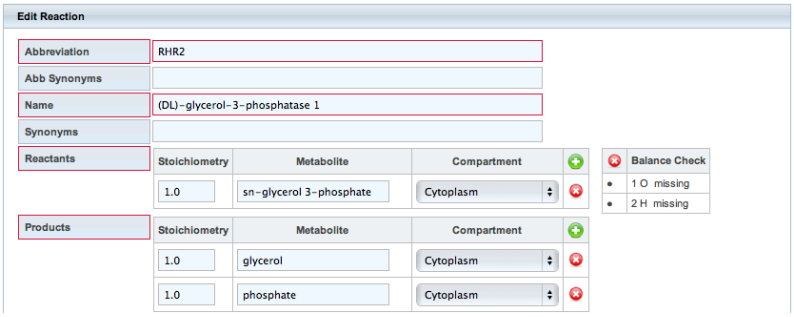


**Figure 3.2:** The balance check mechanism uses the stoichiometry of a reaction to check the equilibrium of production and consumption. It highlights missing or overproduced components which is helpful for correcting the reaction.

### 3.1.2 Web interface

This section describes the layout and presentation design used in the application.

#### Layout

The MEMOSys web interface layout consists of a header, a menu, an information panel, and a footer (see figure 3.1). The header contains an image strip and provides several useful links to display contact information, help pages, news, or feedback options. The information panel displays the main content and the footer contains copyright information and a link to the imprint.

MEMOSys allows users to customize the layout by providing three different options: (a) the standard layout includes header images and uses a fixed width, (b) the second layout hides the image strip, and (c) the full screen layout removes the width limitations and hides the images in the header. The layout selection is persisted across sessions for registered users and is restored as soon as the user logs in again.

#### Data presentation

MEMOSys uses reports and entity lists to present and query data stored in the database.

*Reports* are feature rich web views that display a list of queried objects and support sorting and pagination (see figure 3.3). The result list can be customized by specifying a set of search parameters to restrict the query and selecting the object attributes to be displayed. The use of AJAX avoids complete page reloads when browsing the result list, changing the sort direction, or updating the list to match new search parameters. As reports incorporate attributes from different tables in the result list, a detailed data representation can be displayed. MEMOSys uses reports to present data for metabolites, genes, and reactions. The implementation details of reports are explained in section 3.2.2.

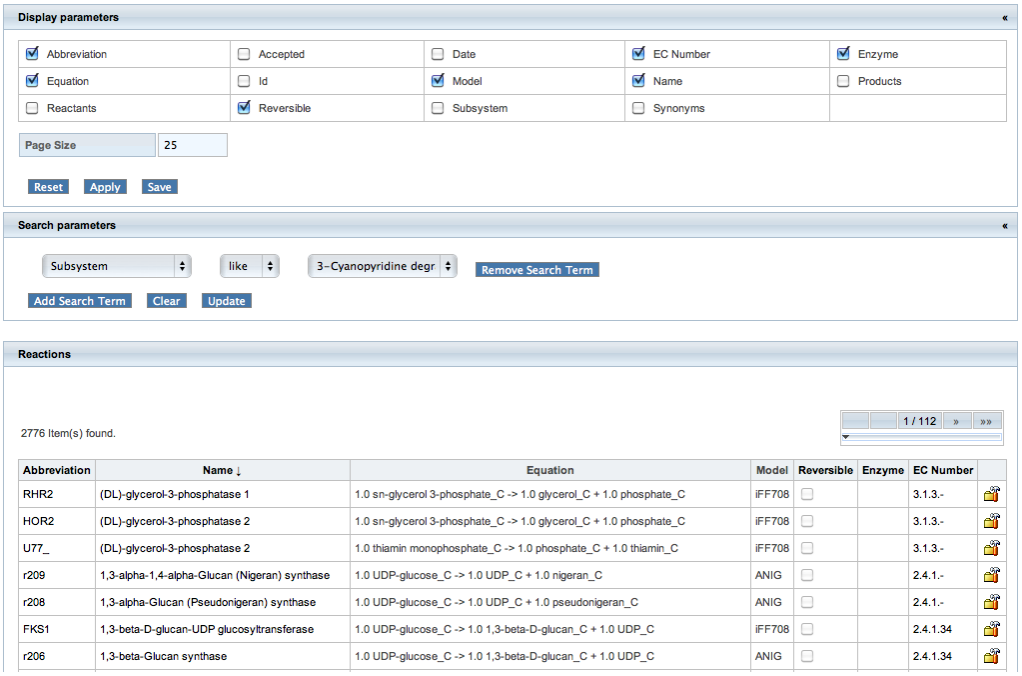


**Figure 3.3:** The reaction report allows users to browse reactions of one or more models. It displays amongst other attributes name, abbreviation, subsystem, equation, and links to external databases. All reports in MEMOSys are based on a common design. The *display parameters* panel allows the selection of attributes which are included in the result list and is used to define the number of items shown per page. Query parameters can be defined in the *search parameters* panel to apply restrictions on the result list. Both panels can be folded by clicking on their headers. Bold headers in the result list indicate that the list can be sorted according to the displayed attribute. Browsing in the result list is done by using the provided slider or navigation buttons. All reports make extensive use of AJAX to avoid whole page reloads when refreshing the result list.

*Entity lists* are used for displaying entities which do not require all features of reports. They use a fixed set of displayed attributes and provide a fast but simple search mechanism.

As all entities are highly connected with each other, MEMOSys displays links to referenced entities throughout the system and allows free navigation within and across all stored models. Moreover, entity pages list associated objects to provide information about their connectivity.

### 3.1.3 Application features

This section outlines general functionality and components of the developed application.

#### File zone

The file zone supports all file types and provides the possibility to upload, list, and download files. Uploaded files can be attached to components and all files are stored at a central repository to facilitate user access control and data backup. Currently, the SBML file used to import a model

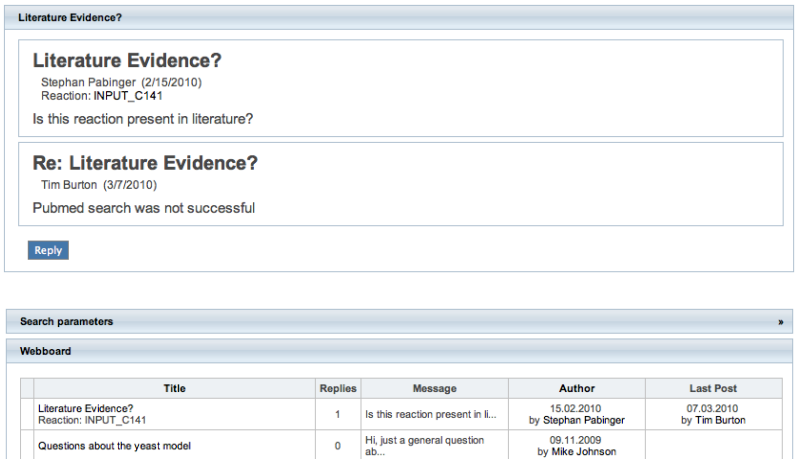


**Figure 3.4:** The webboard displays all general and specific discussions and allows users to easily create comments on existing discussions. Threads associated with a specific component provide direct links to the detail page of referenced entities.

and the image files containing the graphical representations of networks are stored in the file zone.

### Web board

The integrated web board contains general threads and allows researchers to simply attach discussions to stored entities (see figure 3.4). On every entity page a list of currently attached threads is shown, and new discussions can be added to the object. The latest comments of all discussions are displayed on the home screen to quickly update users about new topics.

Unregistered users need to fill in their name and pass a CAPTCHA [1] test to create or reply to discussions. Administrators are able to mark threads as sticky to permanently display them on top of the thread list.

### Model selection

The home screen (see figure 3.1) of the application displays a list of all available models and allows the selection of different versions of the models. The current selection is shown beneath the navigation menu on the left side and is persisted across sessions for registered users. Since each reaction belongs to a model, the system allows users to restrict reactions queries to currently selected models and examine reactions of only one model or several models at once. The model selection is taken into account by all reaction queries and the quick search functionality.

[1] CAPTCHA (**C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part) tests are used to ensure that form inputs are performed by humans and are not generated by computers.

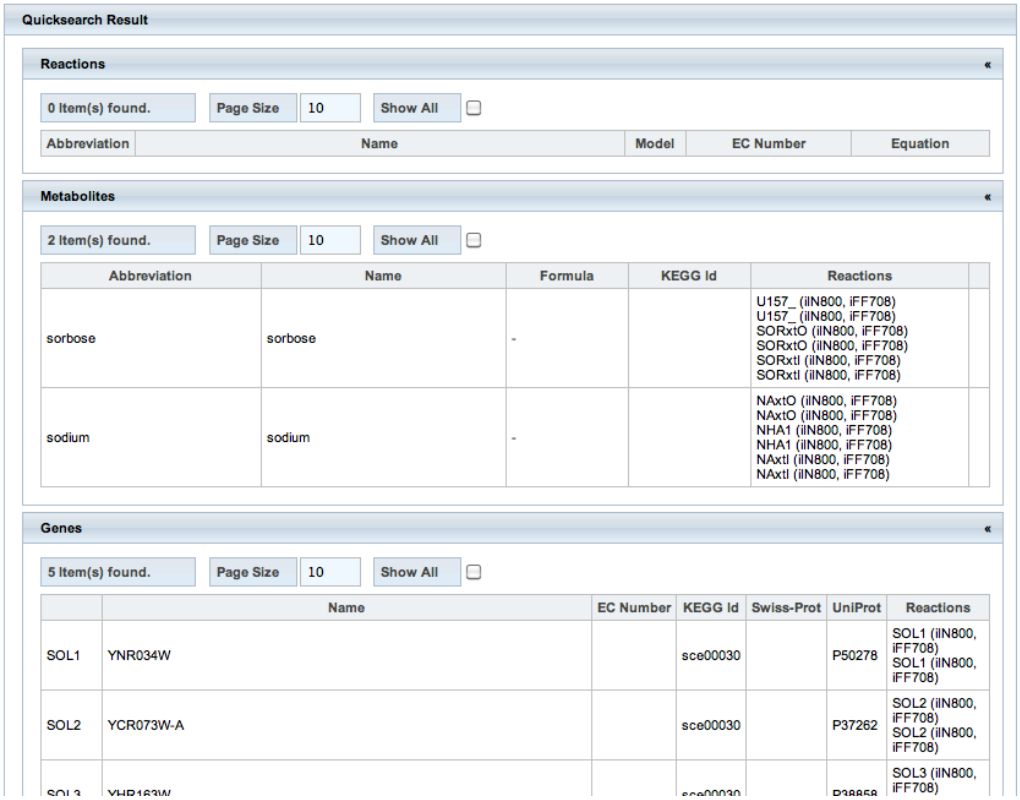


**Figure 3.5:** The image displays the result of a quick search (input *"so*"*). The system queries reactions (abbreviation, name, EC number, KEGG ID), metabolites (abbreviation, name, ChEBI ID), and genes (abbreviation, name). For each data type a list with matching entities is shown.

#### Quick search

The quick search field, located beneath the navigation menu, queries for several entities at once:

- **Reactions** are searched for abbreviation, name, EC number, and KEGG ID.
- **Metabolites** are searched for abbreviation, name, and ChEBI ID.
- **Genes** are searched for abbreviation, and name.

For each entity type the result page (see figure 3.5) displays lists which feature page navigation and page size adjustment. Moreover, the system provides links to referenced objects to quickly navigate to the detail page of the selected entity.

### 3.1.4 Version control

The development of a metabolic model is usually an iterative task that generates several intermediate versions until the final model is established. Therefore, the application includes an automatic

version control system that stores each modification of a model as a new revision. Except for static information like file uploads, all entities and their references are included in the version control system. This allows researchers to retrieve the complete model at any revision and query, compare, and export previous versions of a model. For all version-controlled entities MEMOSys provides a function to display the history of the object (see figure 3.6). It lists the modifications between each version allowing researchers to track the changes of an object.

The home screen (see figure 3.1) of the application lists the latest modifications for metabolites and reactions which gives users a first overview about performed updates.

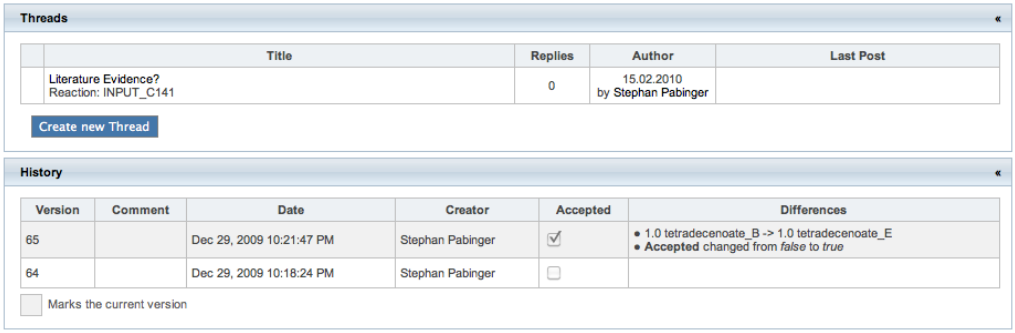


**Figure 3.6:** The top part of the figure lists the discussions of the selected reaction. The lower part shows its history including the latest modifications. For each version a comment, the timestamp, the user, and the differences from the last version are shown. The currently selected version (see 3.1.3) is marked using a gray background color.

### Supervision

Each modification of an object is at first marked as pending and needs to be confirmed by an administrator. The system provides a clear user interface to accept pending changes and every time an administrator approves modifications, a new version number is assigned to the model. In addition to the internal version number provided by the auditing system, models contain a user defined number which can be set by administrators.

MEMOSys differentiates between two access types for models controlled by administrators (see figure 3.7):

- **Public available** - the model is visible to all visitors of the web application and includes only accepted modifications.
- **Restricted** - the model is only visible to editors (see 3.1.7).

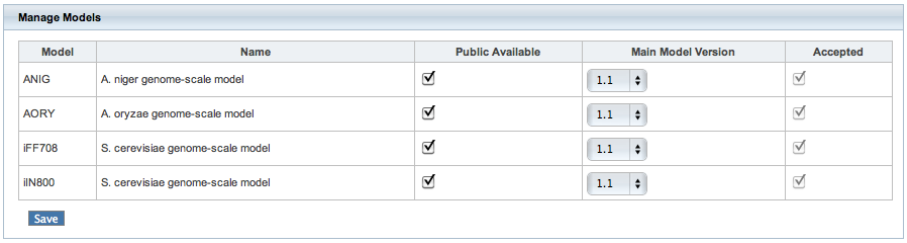


**Figure 3.7:** MEMOSys provides administrator an intuitive view to manage all models in the database. It allows them to the change the access type and the user defined model version and displays the current acceptance status of modifications.

### 3.1.5 Data exchange

MEMOSys features the import and export of metabolic models in the XML based format SBML. It uses the libSBML library (see 2.4.1) to read and write SBML files.

#### Import

The system supports the import of models that are compliant to the SBML format defined by the consensus yeast reconstruction group (see 2.4.1). Before importing a model into the application the file needs to be uploaded into the system (see page 29). The import process is performed asynchronously and a status bar informs the user about the current progress.

#### Export

The software architecture of the application allows exporting all available versions of a model. It supports restricting the exported reactions by (a) including only reactions that are in certain subsystems, or (b) using the result of a reaction query as input for the export mechanism. Moreover, the export functionality defines three different ways to assign reactions and metabolites to compartments (compartmentalization):

- **Completely Compartmentalized** - reactions and metabolites are assigned to compartments as they are stored in the database
- **Partially Decompartmentalized** - reactions and metabolites which are in sub-compartments of the Cytosol are assigned to the Cytosol. All other compartments are still present in the exported model.
- **Fully Decompartmentalized** - the exported model contains no compartments resulting in an unsegregated system.

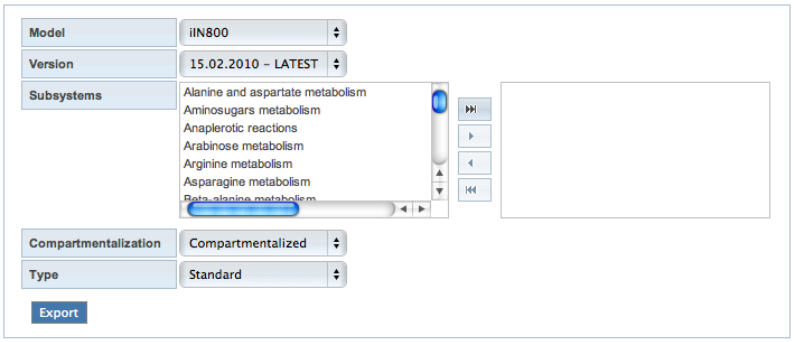


**Figure 3.8:** Depicted are the SBML export settings. They allow users to select the model and its version, restrict the export to a certain set of subsystems, and choose the level of compartmentalization. Currently, MEMOSys supports the export into two different SBML formats (*Type*) - the consensus yeast and the COBRA toolbox format.

Currently, MEMOSys supports exporting models into SBML files that are compliant with the consensus yeast format or with the COBRA toolbox (see figure 3.8).

In addition to the SBML export functionality, metabolite and reaction lists can be exported into Excel or PDF files for further usage.

### 3.1.6 Model comparison

Great attention has been paid to the implementation of a model comparison functionality. The application allows researchers to compare any version of two different models. Moreover, it is possible to compare two versions of the same model to identify development changes. The following entities of the models are compared:

- **Reactions** are compared based on their KEGG ID if defined by both reactions. Otherwise the system uses the equation (reversibility, metabolites, and their stoichiometry) for comparison.
- **Metabolites** are compared based on their ChEBI ID, KEGG ID, and their name.
- **Genes** are compared based on their UniProt ID, SGD ID, and their name.

The first part of the comparison result displays a summary of the detected differences (see figure 3.9) and shows area accurate Venn diagrams for reactions, metabolites, and genes. A new chart type was developed to create Venn diagrams in JFreeChart (see 2.4.2) which uses a numerical analysis approach to calculate the correct circle intersection distance.

The second part allows applying restrictions on the used models to limit the results to selected compartments and subsystems. The third part (see figure 3.10) displays detailed lists of equal and unique entities for each model and provides tabs to switch between reactions, metabolites,

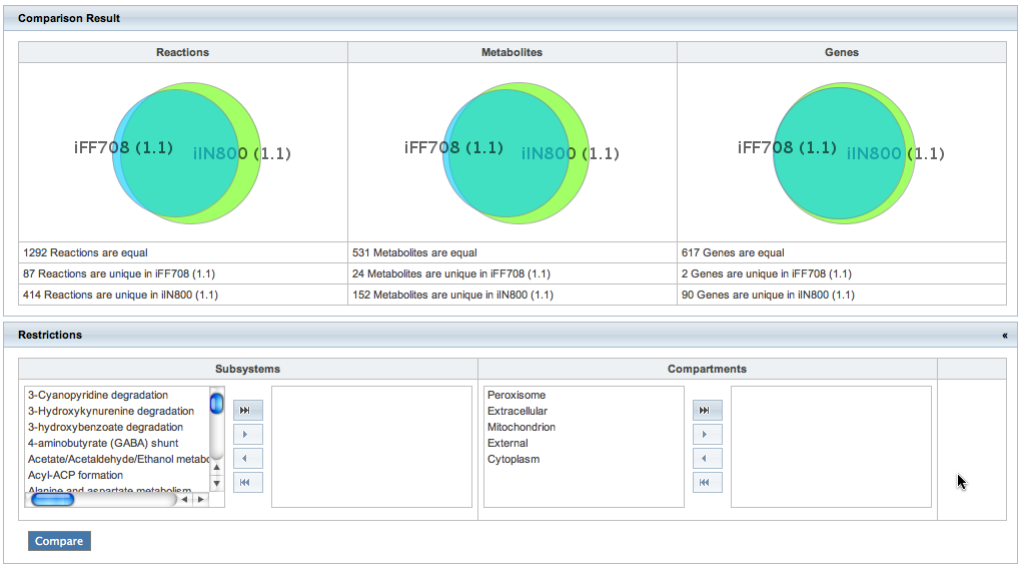


**Figure 3.9:** Displayed is the first part of a model comparison result. It shows the area accurate Venn diagrams for reactions, metabolites, and genes. Additionally, it provides a short quantitative description of the differences between the models. The next section is used to restrict the result to selected subsystems or compartments.

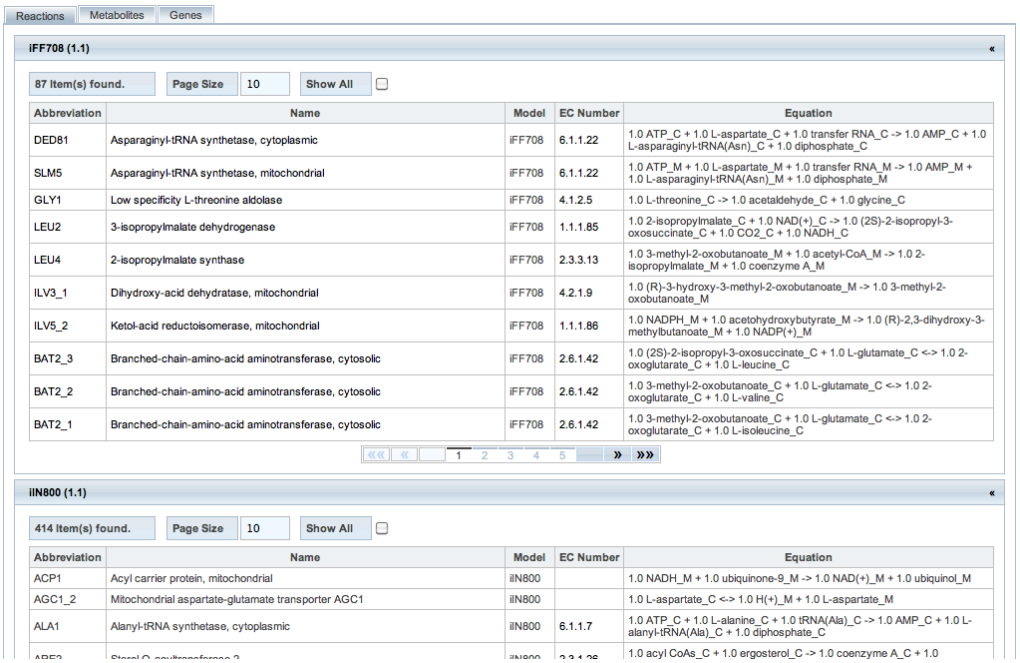


**Figure 3.10:** Shown is the comparison result list for one model. Not included in this picture are the result lists for the second model and the equal entities. The tabs on the top allow switching between reactions, metabolites, and genes.

and genes. In addition, reactions, metabolites, and genes are linked to the corresponding entities which display detailed information about the components. Each list is scrollable and allows adjusting the number of displayed entries per page.

### 3.1.7 User access

MEMOSys defines four different user classes to control data access:

- **Users without an account** - are allowed to view accepted model versions. Additionally, only models that have been made publicly available are visible to this user class. They are not allowed to create, update, or delete entities.
- **Visitors** - are granted the same rights as users without an account and certain user settings are persisted in the database to increase the usability of the application. Furthermore, administrators are able to grant individual visitors access to unpublished models.
- **Editors** - are allowed to create, update, and delete all entities in the database. Moreover, they have access to models which are not publicly available and always see the latest version of entities even if they have not yet been accepted. Editors are allowed to upload files to the web server and import SBML models into the database.
- **Administrators** - are editors, which are able to accept modifications of entities and change the public availability of models.

## 3.2 Implementation details

### 3.2.1 Application design and implementation

MEMOSys is a multi-tier application based on Java EE 5 and JBoss Seam. The presentation tier uses the JSF 1.2 reference implementation by Sun Microsystems in combination with JBoss RichFaces and Facelets. It employs institute guidelines to provide a corporate identity compliant web design and makes use of Web 2.0 technologies for dynamic content generation. RichFaces components are extensively used to guarantee a consistent web front-end and to provide support for all major web browsers. The Facelets templating mechanism has been extensively used to avoid unnecessary code redundancies.

The business tier contains *Home* (see 2.3.3) classes which manage entity beans to store and retrieve data. EJB session beans provide business functionality using services offered by the EJB container.

The persistence tier uses Hibernate as JPA implementation to store objects in a database and provides version control support based on Envers.

Data security is managed by an Authentication and Authorization System (AAS) [Zeller, 2003] which is attached to the Seam security framework. It allows the definition of fine grained user roles and stores users, user groups, and resource permissions. The security framework is used in the application to restrict method and page level access.

### 3.2.2 Seam

After finishing the definition of a database layout the reverse engineering mechanism of the Seam framework (see 2.3.3) has been used to create a first scaffold of MEMOSys. The resulting application components were used as a starting point to implement the required functionality. The Seam framework has been extended with the following features.

#### Reports

A core requirement of the application is to provide a powerful and flexible querying tool to retrieve desired information from the database. The Seam framework offers an EntityQuery component which creates Hibernate queries based on a criteria entity. It allows users to specify several criteria fields which are used as query parameters for database searches.

MEMOSys defines a new generic class used for data reports (see 3.1.2) which extends the EntityQuery class to include display, sort, and query parameters. Queries are built by concatenating a set of search terms which are translated to the *where* clause of a HQL query. Users can define an arbitrary number of search terms which use properties either from the main or from related entities of the report. According to the selected property the system presents input fields or automatically filled select boxes. The final search term consists of the selected property, a query operator (e.g: like, not like, $=$, $>$, $<$), and the user defined search value (see figure 3.3).

Display settings allow users to define which fields are shown in a report and are used to specify the number of items listed per page. For each registered user the settings are stored in the database using the user settings system (see 3.2.5).

All reports in MEMOSys are implemented as stateful session beans to maintain a consistent state throughout a session. This design supports the use of query results in other parts of the application.

#### Entity lists

In addition to data reports, MEMOSys uses stateless session beans based on the Seam Entity-Query class for lists that do not need to maintain a session state. The created components enhance the default functionality by supporting page scrolling and the definition of a default sort order.

#### Version control support

Reports, Entity lists, and Home classes were extended to include a version control mechanism and to provide support for user access roles and the public availability of models. Therefore, all Seam generated queries had to be redesigned to support the outlined functionality. Since the entity-manager provided by Envers did not offer all needed functionalities, Hibernate native queries are used to fetch versioned entities.

For each version of an entity the revision number, the timestamp, a freely definable comment, and the current user are stored in the database. Additionally, the object is marked as *pending* to inform the administrator about new modifications. All fields are automatically set by the implemented version-control system which allows an easy extension of the system.

### 3.2.3 Web board

The web board has been designed to allow simple attaching of discussions to entities. Therefore, the created thread stores the ID and the type of the connected entity and a generic system creates hyperlinks to the detail page of the referenced entity.

### 3.2.4 LibSBML

The libSBML library was modified and extended to be compliant to the consensus yeast format. The modifications included altering the C++ source code files of the SBML writer and adapting several SBML classes. The library was rebuild and integrated into the developed application.

### 3.2.5 User settings

MEMOSys offers users several settings - such as sort directions, theme layout, or model selection - to customize the application to their needs. As a user may access the application using different clients, these settings need to be stored in the database to persist them across sessions.

The user settings mechanism is implemented using interceptors and annotations. It allows developers to mark properties that should be persisted as user settings by simply applying an

annotation on the desired fields. In order to activate the method interceptor, a class level annotation has to be employed which scans the class for properties that are marked as user settings. The interceptor loads data from the database if the field has not been initialized and persists the setting when the value of the field is updated. If no value is stored in the database the system uses a defined default value. For each user setting the currently logged in user, the annotated class, and the name and value of the property are stored in the database.

#### Integration of Genome AAS in the Seam security framework

JBoss Seam provides a user security system that is fully integrated into the presentation and business tiers. The Institute for Genomics and Bioinformatics uses the *Genome Usermanagement* system which is a J2EE based Authentication and Authorization system (AAS) [Zeller, 2003]. In order to include the in-house AAS system into Seam, a custom identity manager was implemented which handles the communication between the two security systems.

## 3.3 Model integration

The developed application has been filled with several well-annotated reconstructions of metabolic models. Each reconstruction was manually reviewed in accordance with the model developers and has been improved to meet the standards of the SBML format defined by the consensus yeast reconstruction group.

The following models are currently stored in the system:

- **AORY** - Aspergillus oryaze [Vongsangnak et al., 2008]
- **ANIG** - Aspergillus niger [Andersen et al., 2008]
- **ANID** - Aspergillus nidulans [David et al., 2008]
- **iFF708** - Saccharomyces cerevisiae [Förster et al., 2003]
- **iIN800** - Saccharomyces cerevisiae [Nookaew et al., 2008]
- **ymn2_0** - Saccharomyces cerevisiae [Herrgård et al., 2008]

# Chapter 4

# Discussion

In this thesis a bioinformatics platform for the management, development, and storage of metabolic models has been developed. MEMOSys is a tool aimed at the metabolic research community to facilitate the study of existing metabolic models and ease the collaborative development of new networks. This chapter lists the main features of the application, outlines the differences to existing software tools, and discusses the used software technologies.

Blazeck and Alper [2010] and Risso et al. [2009] state that the development of new metabolic maps is an iterative process where the possibility to reproduce each development step is an important prerequisite. Therefore, MEMOSys features a built-in version control mechanism that automatically stores the complete history of a model. The system allows querying and displaying previous versions of a model and provides the possibility to display the history of each entity. Moreover, the system is able to list modifications of each object, including information about the user, the time-point of a modification, and an optional comment.

A common characteristic shared by all metabolic models is the heavy interdependency of their components, reflected for example by the larger number of reactions compared to the number of metabolites [Kauffman et al., 2003; Lacroix et al., 2008]. The MEMOSys data model takes this aspect into account and supports connections between all entities of a metabolic network and allows mapping of any possible combination of genes catalyzing a reaction. Furthermore, the application provides links to referenced entities to allow an easy navigation within and across models.

As more and more metabolic models are being generated, the future development of genome-scale networks will strongly rely on already existing reconstructions of related organisms. Hence,

a flexible and intuitive mechanism to assess the similarity between models will become more and more important. For this reason, MEMOSys offers a flexible comparison tool to quickly get an overview of the differences between two models. It features area accurate Venn diagrams which graphically illustrate the overlap of components. Additionally, the application displays detailed lists of equal or differing reactions, metabolites, and genes where each list entry is connected to the information page showing properties and relations to other components. Moreover, the comparison can be restricted to a selection of subsystems or compartments.

MEMOSys provides sophisticated data exchange mechanisms to import and export metabolic models. Specifically, it uses the SBML format outlined by the consensus yeast reconstruction group which laid special emphasis on annotating components with external references using the MIRIAM format. Due to historical reasons, different names that describe the same component are used in biomolecular models. MIRIAM annotations allow the unique identification of components in metabolic models which is a prerequisite for scientific collaborations and model comparability [Krause et al., 2010].

The user interfaces for data import and export have been designed to be as lean as possible while still providing flexibility. The system is able to export either all reactions of a model or only a set of reactions which is useful when just a specific part of a model should be exported.

Additionally, the export mechanism supports several different ways of compartmentalization, including partially or fully decompartmentalized models (see 3.1.5). Therefore, the exported models can be directly used in analysis tools that do not support a fine-grained distribution of reactions into different compartments.

New biological insights drive the development of metabolic models and therefore lead to various versions of a particular network. As certain use cases demand access to specific versions of a model, the MEMOSys export mechanism is fully integrated into the version control system and provides researchers access to the complete history of a model. As outlined in section 1.2.1, numerous methods are available to analyze a metabolic model. In addition to the format defined by the consensus yeast reconstruction group, MEMOSys supports the proprietary file format of the COBRA toolbox. Furthermore, reaction and metabolite lists can be exported into Excel or PDF files.

MEMOSys offers a feature rich state-of-the-art web interface which provides quick search, query, and comparison mechanisms. It stores user preferences within and across sessions, includes searchable and customizable data reports, and provides automatically filled suggestion boxes, thus relieving users from repetitive tasks and increasing usability. As data security is an

important issue in the research community, special attention has been laid on the implementation of an adjustable and solid security system supporting the definition of fine grained access rights. MEMOSys defines several user types which, amongst others, allow visitors to make use of the implemented user settings system and guarantees editors that unpublished data is only visible within a specific group. The application includes a supervision system which enforces that modifications are approved by a key researcher to maintain a high model quality.

Oberhardt et al. [2009] stated that the reconstruction of a genome-scale metabolic model is a four-step process: (i) assembly of an initial reconstruction based on annotated gene data and information from external databases, (ii) curation of the initial model by including references from primary literature and conversion into a mathematical representation, (iii) validation of the model using analysis methods to compare the predictions to the phenotypic data, and (iv) refining the model by modifying its components and relations to narrow the gap between predicted and phenotypic data.

MEMOSys supports researchers in all of the above mentioned steps. It includes external references to each component of a model using the MIRIAM annotation, and provides a query mechanism which allows a quick look-up of synonyms of a particular component to avoid inconsistencies in the reconstruction. In addition, the system supports model curation by storing references to primary literature and allows users to export models into formats used by analysis tools. Due to the included version control system, all refining steps are stored in the database to enable a transparent and traceable reconstruction process. As the refinement process involves the comparison of predictions to experimental data, a lot of additional files need to be stored and organized by the researchers. MEMOSys allows the upload of arbitrary files into the system and attaching them to models in order to collect experimental data at a central repository.

The main purpose of the developed application is the management and development of metabolic models. The system is not intended to analyze models and therefore does not include tools to assess their capabilities. However, the application features a highly customizable export mechanism which can be easily extended to include file formats for new analysis tools. MEMOSys is not restricted to a particular organism or group and is therefore able to manage any metabolic model. It currently contains several well annotated models which have been manually curated to be compliant with the specification defined by the consensus yeast reconstruction group (see 2.4.1).

The integrated web board facilitates the collaborative development of metabolic models. Researchers can create either global threads to discuss general topics, or attach threads to individual

objects to debate specific issues of a model or its components.

To the best of our knowledge this is the first bioinformatics platform which is attached to a highly sophisticated user management system and offers a built in version control system for all components. Current existing systems like the BiGG Database [BIGG, 2010] or the BioModels database [Novère et al., 2006] do not support the reconstruction of novel metabolic models by providing an automatic auditing system and a customizable authentication and authorization system. Moreover, the developed application features a state-of-the-art web front-end and allows researchers to perform complex queries on the stored models. MEMOSys offers free academic use, support of data exchange standards, and has been tested at the Institute for Genomics and Bioinformatics, Graz University of Technology.

### Technology

The application has been implemented in Java which guarantees platform and database independence. It is based on the Java EE framework, which is a great improvement over its predecessor J2EE introducing dependency injection and facilitating testing and bean creation. A high test coverage is a key quality measurement in today's software development as it guarantees correct functionality of components when modifying or extending an application. The use of Java EE made it possible to easily test business and persistence functions, the developed version control system, and complex user access scenarios.

The well established Object Relational Mapping (ORM) tool Hibernate is used as the persistence framework which provides good integration into the Java EE framework. The client tier is managed by the state-of-the-art presentation framework JSF which features a component based design and allows the creation of custom, reusable JSF templates.

JBoss Richfaces has been favored over other component libraries as it provides a rich component set, great AJAX functionality and offers good community support. It includes a skinning mechanism which is useful to adapt the design to corporate guidelines. The use of established component libraries guarantees that their components are well tested and provide the same design and functionality on all major browsers.

JBoss Seam has been used in addition to Java EE and JSF, which offers a feature rich component model tying together the different layers of the technology stack. The bijection mechanism of Seam allows developers to inject and outject values from context variables into component attributes and vice versa which poses a very flexible programming model. However, bijection may be the reason for performance issues as it needs to inject and outject resources on each method call, and should therefore not be used in heavily accessed components. Seam offers a reverse engineering tool which allows the rapid creation of a prototype, including business and presenta-

tion code, based on an existing database schema. This feature relieves the developer of many tedious coding tasks such as providing CRUD operations for entities or implementing simple lists or detailed views of objects. As all used technologies follow the “convention over configuration” principle, most standard properties do not need to be specified which eases the development and improves source code readability.

Envers has been used as the basis framework for the version control system. It was originally designed as an extension to Hibernate, but will be included into the core module of the upcoming version. This tight integration guarantees great usability and continued improvements in the next releases. As Envers currently does not support traversing relations in queries, workarounds had to be implemented to provide the required functionality. Moreover, the entity manager of Envers was not able to execute specific, advanced queries, which instead had to been modeled using Hibernate native queries. However, Envers provides all functions for the version management of entities and offers active community support.

### Outlook

Applications in the bioinformatics field can never be considered as finished as the ongoing development of biological and information technologies continuously change the requirements of computational environments.

One of the next steps in the development of MEMOSys will be the integration of gene expression data into the system. As the application already stores annotated graphical model representations, transcriptomic data can easily be mapped onto the model using existing tools such as ReMapper [Andersen, 2008] or Pathway Tools [Karp et al., 2010]. The detailed map will be enriched with transcription information that for example displays gene values as color-coded boxes next to the reactions. This will allow researchers to study all gene expression profiles in context with metabolic maps at once to get an overview of the investigated organism.

The development of a functionality to browse the provided graphical representation of a network could be realized within a reasonable time frame. The integration of an image browser will make the use of additional tools unnecessary and will enhance the usability and consistency of the developed application. As all included images are vector graphics, unlimited scalability without quality loss is provided.

Focus will be placed on the improvement of the version control system. The tight integration of Envers with Hibernate in the upcoming version will facilitate the development of new features and increase the performance of the system. Therefore, it will be possible to include additional queries which will support an even more detailed search for components in current and archived versions.

Another step will be the improvement of the comparison mechanism which will allow researchers to compare not only models but any current or versioned entity of a particular type. The output should highlight the differences of their attributes and provide links to affected associated entities.

### Conclusion

In conclusion, the implemented application MEMOSys provides researchers a tool to store, manage, and develop metabolic models. The rich web interface and the included version control system greatly facilitate the development of new metabolic models and support the study of existing networks. Furthermore, the fine-grained authorization system and the clear definition of user roles allow collaborations across departments and universities. Due to the flexible and modular software architecture additional tools and new methods can be easily integrated into the application.

# Appendix A

# Bibliography

## Article references

[Andersen et al., 2008] Mikael Rørdam Andersen, Michael Lynge Nielsen, and Jens Nielsen. Metabolic model integration of the bibliome, genome, metabolome and reactome of Aspergillus niger. *Mol Syst Biol*, 4:178, 2008. doi: 10.1038/msb.2008.12. URL http://dx.doi.org/10.1038/msb.2008.12.

[Becker et al., 2007] Scott A Becker, Adam M Feist, Monica L Mo, Gregory Hannum, Bernhard Ø Palsson, and Markus J Herrgard. Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox. *Nat Protoc*, 2(3):727–738, 2007. doi: 10.1038/nprot.2007.99. URL http://dx.doi.org/10.1038/nprot.2007.99.

[Blazeck and Alper, 2010] John Blazeck and Hal Alper. Systems metabolic engineering: Genome-scale models and beyond. *Biotechnol J*, Feb 2010. doi: 10.1002/biot.200900247. URL http://dx.doi.org/10.1002/biot.200900247.

[Bornstein et al., 2008] Benjamin J Bornstein, Sarah M Keating, Akiya Jouraku, and Michael Hucka. LibSBML: an API library for SBML. *Bioinformatics*, 24(6):880–881, Mar 2008. doi: 10.1093/bioinformatics/btn051. URL http://dx.doi.org/10.1093/bioinformatics/btn051.

[Caspi et al., 2010] Ron Caspi, Tomer Altman, Joseph M Dale, Kate Dreher, Carol A Fulcher, Fred Gilham, Pallavi Kaipa, Athikkattuvalasu S Karthikeyan, Anamika Kothari, Markus Krummenacker, Mario Latendresse, Lukas A Mueller, Suzanne Paley, Liviu Popescu, Anuradha Pujar, Alexander G Shearer, Peifen Zhang, and Peter D Karp. The MetaCyc database of metabolic pathways and enzymes and the BioCyc collection of pathway/genome databases.

*Nucleic Acids Res*, 38(Database issue):D473–D479, Jan 2010. doi: 10.1093/nar/gkp875. URL http://dx.doi.org/10.1093/nar/gkp875.

[Chang et al., 2009] Antje Chang, Maurice Scheer, Andreas Grote, Ida Schomburg, and Dietmar Schomburg. BRENDA, AMENDA and FRENDA the enzyme information system: new content and tools in 2009. *Nucleic Acids Res*, 37(Database issue):D588–D592, Jan 2009. doi: 10.1093/nar/gkn820. URL http://dx.doi.org/10.1093/nar/gkn820.

[Codd, 1970] E. F. Codd. A Relational Model of Data for Large Shared Data Banks. *Commun. ACM*, 13(6):377–387, 1970.

[David et al., 2008] Helga David, Ilknur S Ozçelik, Gerald Hofmann, and Jens Nielsen. Analysis of Aspergillus nidulans metabolism at the genome-scale. *BMC Genomics*, 9:163, 2008. doi: 10.1186/1471-2164-9-163. URL http://dx.doi.org/10.1186/1471-2164-9-163.

[Degtyarenko et al., 2008] Kirill Degtyarenko, Paula de Matos, Marcus Ennis, Janna Hastings, Martin Zbinden, Alan McNaught, Rafael Alcántara, Michael Darsow, Mickaël Guedj, and Michael Ashburner. ChEBI: a database and ontology for chemical entities of biological interest. *Nucleic Acids Res*, 36(Database issue):D344–D350, Jan 2008. doi: 10.1093/nar/gkm791. URL http://dx.doi.org/10.1093/nar/gkm791.

[Durot et al., 2009] Maxime Durot, Pierre-Yves Bourguignon, and Vincent Schachter. Genome-scale models of bacterial metabolism: reconstruction and applications. *FEMS Microbiol Rev*, 33(1):164–190, Jan 2009. doi: 10.1111/j.1574-6976.2008.00146.x. URL http://dx.doi.org/10.1111/j.1574-6976.2008.00146.x.

[Edwards et al., 2001] J. S. Edwards, R. U. Ibarra, and B. O. Palsson. In silico predictions of Escherichia coli metabolic capabilities are consistent with experimental data. *Nat Biotechnol*, 19(2):125–130, Feb 2001. doi: 10.1038/84379. URL http://dx.doi.org/10.1038/84379.

[Engel et al., 2010] Stacia R Engel, Rama Balakrishnan, Gail Binkley, Karen R Christie, Maria C Costanzo, Selina S Dwight, Dianna G Fisk, Jodi E Hirschman, Benjamin C Hitz, Eurie L Hong, Cynthia J Krieger, Michael S Livstone, Stuart R Miyasato, Robert Nash, Rose Oughtred, Julie Park, Marek S Skrzypek, Shuai Weng, Edith D Wong, Kara Dolinski, David Botstein, and J. Michael Cherry. Saccharomyces Genome Database provides mutant phenotype data. *Nucleic Acids Res*, 38(Database issue):D433–D436, Jan 2010. doi: 10.1093/nar/gkp917. URL http://dx.doi.org/10.1093/nar/gkp917.

[Fleischmann et al., 1995] R. D. Fleischmann, M. D. Adams, O. White, R. A. Clayton, E. F. Kirkness, A. R. Kerlavage, C. J. Bult, J. F. Tomb, B. A. Dougherty, and J. M. Merrick. Whole-genome

random sequencing and assembly of Haemophilus influenzae Rd. *Science*, 269(5223):496–512, Jul 1995.

[Fraser et al., 1995] C. M. Fraser, J. D. Gocayne, O. White, M. D. Adams, R. A. Clayton, R. D. Fleischmann, C. J. Bult, A. R. Kerlavage, G. Sutton, J. M. Kelley, R. D. Fritchman, J. F. Weidman, K. V. Small, M. Sandusky, J. Fuhrmann, D. Nguyen, T. R. Utterback, D. M. Saudek, C. A. Phillips, J. M. Merrick, J. F. Tomb, B. A. Dougherty, K. F. Bott, P. C. Hu, T. S. Lucier, S. N. Peterson, H. O. Smith, C. A. Hutchison, and J. C. Venter. The minimal gene complement of Mycoplasma genitalium. *Science*, 270(5235):397–403, Oct 1995.

[Förster et al., 2003] Jochen Förster, Iman Famili, Patrick Fu, Bernhard Ø Palsson, and Jens Nielsen. Genome-scale reconstruction of the Saccharomyces cerevisiae metabolic network. *Genome Res*, 13(2):244–253, Feb 2003. doi: $10.1101/\mathrm{gr}.234503$. URL `http://dx.doi.org/10.1101/gr.234503`.

[Gasteiger et al., 2003] Elisabeth Gasteiger, Alexandre Gattiker, Christine Hoogland, Ivan Ivanyi, Ron D Appel, and Amos Bairoch. ExPASy: The proteomics server for in-depth protein knowledge and analysis. *Nucleic Acids Res*, 31(13):3784–3788, Jul 2003.

[Glass et al., 2006] John I Glass, Nacyra Assad-Garcia, Nina Alperovich, Shibu Yooseph, Matthew R Lewis, Mahir Maruf, Clyde A Hutchison, Hamilton O Smith, and J. Craig Venter. Essential genes of a minimal bacterium. *Proc Natl Acad Sci U S A*, 103(2):425–430, Jan 2006. doi: $10.1073/\mathrm{pnas}.0510013103$. URL `http://dx.doi.org/10.1073/pnas.0510013103`.

[Goto et al., 2002] Susumu Goto, Yasushi Okuno, Masahiro Hattori, Takaaki Nishioka, and Minoru Kanehisa. LIGAND: database of chemical compounds and reactions in biological pathways. *Nucleic Acids Res*, 30(1):402–404, Jan 2002.

[Heino et al., 2010] Jenni Heino, Daniela Calvetti, and Erkki Somersalo. Metabolica: A statistical research tool for analyzing metabolic networks. *Comput Methods Programs Biomed*, 97(2): 151–167, Feb 2010. doi: $10.1016/\mathrm{j.cmpb}.2009.07.007$. URL `http://dx.doi.org/10.1016/j.cmpb.2009.07.007`.

[Herrgård et al., 2008] Markus J Herrgård, Neil Swainston, Paul Dobson, Warwick B Dunn, K. Yalçin Arga, Mikko Arvas, Nils Blüthgen, Simon Borger, Roeland Costenoble, Matthias Heinemann, Michael Hucka, Nicolas Le Novère, Peter Li, Wolfram Liebermeister, Monica L Mo, Ana Paula Oliveira, Dina Petranovic, Stephen Pettifer, Evangelos Simeonidis, Kieran Smallbone, Irena Spasic, Dieter Weichart, Roger Brent, David S Broomhead, Hans V Westerhoff, Betül Kirdar, Merja Penttilä, Edda Klipp, Bernhard Ø Palsson, Uwe Sauer, Stephen G Oliver,

Pedro Mendes, Jens Nielsen, and Douglas B Kell. A consensus yeast metabolic network reconstruction obtained from a community approach to systems biology. *Nat Biotechnol*, 26(10): 1155–1160, Oct 2008. doi: 10.1038/nbt1492. URL http://dx.doi.org/10.1038/nbt1492.

[Hoops et al., 2006] Stefan Hoops, Sven Sahle, Ralph Gauges, Christine Lee, Jürgen Pahle, Natalia Simus, Mudita Singhal, Liang Xu, Pedro Mendes, and Ursula Kummer. COPASI–a COmplex PAthway SImulator. *Bioinformatics*, 22(24):3067–3074, Dec 2006. doi: 10.1093/bioinformatics/btl485. URL http://dx.doi.org/10.1093/bioinformatics/btl485.

[Hucka et al., 2003] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, A. A. Cuellar, S. Dronov, E. D. Gilles, M. Ginkel, V. Gor, I. I. Goryanin, W. J. Hedley, T. C. Hodgman, J-H. Hofmeyr, P. J. Hunter, N. S. Juty, J. L. Kasberger, A. Kremling, U. Kummer, N. Le Novère, L. M. Loew, D. Lucio, P. Mendes, E. Minch, E. D. Mjolsness, Y. Nakayama, M. R. Nelson, P. F. Nielsen, T. Sakurada, J. C. Schaff, B. E. Shapiro, T. S. Shimizu, H. D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, J. Wang, and S. B. M. L. Forum. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, Mar 2003.

[Kanehisa, 1997] M. Kanehisa. A database for post-genome analysis. *Trends Genet*, 13(9):375–376, Sep 1997.

[Karp et al., 2000] P. D. Karp, M. Riley, M. Saier, I. T. Paulsen, S. M. Paley, and A. Pellegrini-Toole. The EcoCyc and MetaCyc databases. *Nucleic Acids Res*, 28(1):56–59, Jan 2000.

[Karp et al., 2010] Peter D Karp, Suzanne M Paley, Markus Krummenacker, Mario Latendresse, Joseph M Dale, Thomas J Lee, Pallavi Kaipa, Fred Gilham, Aaron Spaulding, Liviu Popescu, Tomer Altman, Ian Paulsen, Ingrid M Keseler, and Ron Caspi. Pathway Tools version 13.0: integrated software for pathway/genome informatics and systems biology. *Brief Bioinform*, 11(1): 40–79, Jan 2010. doi: 10.1093/bib/bbp043. URL http://dx.doi.org/10.1093/bib/bbp043.

[Kauffman et al., 2003] Kenneth J Kauffman, Purusharth Prakash, and Jeremy S Edwards. Advances in flux balance analysis. *Curr Opin Biotechnol*, 14(5):491–496, Oct 2003.

[Kay and Wren, 2009] Emily Kay and Brendan W Wren. Recent advances in systems microbiology. *Curr Opin Microbiol*, 12(5):577–581, Oct 2009. doi: 10.1016/j.mib.2009.08.007. URL http://dx.doi.org/10.1016/j.mib.2009.08.007.

[Klamt et al., 2006] Steffen Klamt, Julio Saez-Rodriguez, Jonathan A Lindquist, Luca Simeoni, and Ernst D Gilles. A methodology for the structural and functional analysis of signaling and

regulatory networks. *BMC Bioinformatics*, 7:56, 2006. doi: 10.1186/1471-2105-7-56. URL http://dx.doi.org/10.1186/1471-2105-7-56.

[Krause et al., 2010] Falko Krause, Jannis Uhlendorf, Timo Lubitz, Marvin Schulz, Edda Klipp, and Wolfram Liebermeister. Annotation and merging of SBML models with semanticSBML. *Bioinformatics*, 26(3):421–422, Feb 2010. doi: 10.1093/bioinformatics/btp642. URL http://dx.doi.org/10.1093/bioinformatics/btp642.

[Lacroix et al., 2008] Vincent Lacroix, Ludovic Cottret, Patricia Thébault, and Marie-France Sagot. An introduction to metabolic networks and their structural analysis. *IEEE/ACM Trans Comput Biol Bioinform*, 5(4):594–617, 2008. doi: 10.1109/TCBB.2008.79. URL http://dx.doi.org/10.1109/TCBB.2008.79.

[Larhlimi and Bockmayr, 2009] Abdelhalim Larhlimi and Alexander Bockmayr. A new constraint-based description of the steady-state flux cone of metabolic networks. *Discrete Applied Mathematics*, 157(10):2257 – 2266, 2009. ISSN 0166-218X. doi: DOI:10.1016/j.dam.2008.06.039. URL http://www.sciencedirect.com/science/article/B6TYW-4TDBM62-7/2/8f90eb51c8f65d680c8da49ca177a04e. Networks in Computational Biology.

[Lee et al., 2005] Sang Yup Lee, Dong-Yup Lee, and Tae Yong Kim. Systems biotechnology for strain improvement. *Trends Biotechnol*, 23(7):349–358, Jul 2005. doi: 10.1016/j.tibtech.2005.05.003. URL http://dx.doi.org/10.1016/j.tibtech.2005.05.003.

[Ma and Zeng, 2003] Hongwu Ma and An-Ping Zeng. Reconstruction of metabolic networks from genome data and analysis of their global structure for various organisms. *Bioinformatics*, 19(2):270–277, Jan 2003.

[Nielsen and Jewett, 2008] Jens Nielsen and Michael C Jewett. Impact of systems biology on metabolic engineering of Saccharomyces cerevisiae. *FEMS Yeast Res*, 8(1):122–131, Feb 2008. doi: 10.1111/j.1567-1364.2007.00302.x. URL http://dx.doi.org/10.1111/j.1567-1364.2007.00302.x.

[Nookaew et al., 2008] Intawat Nookaew, Michael C Jewett, Asawin Meechai, Chinae Thammarongtham, Kobkul Laoteng, Supapon Cheevadhanarak, Jens Nielsen, and Sakarindr Bhumiratana. The genome-scale metabolic model iIN800 of Saccharomyces cerevisiae and its validation: a scaffold to query lipid metabolism. *BMC Syst Biol*, 2:71, 2008. doi: 10.1186/1752-0509-2-71. URL http://dx.doi.org/10.1186/1752-0509-2-71.

[Novère et al., 2005] Nicolas Le Novère, Andrew Finney, Michael Hucka, Upinder S Bhalla, Fabien Campagne, Julio Collado-Vides, Edmund J Crampin, Matt Halstead, Edda Klipp, Pe-

dro Mendes, Poul Nielsen, Herbert Sauro, Bruce Shapiro, Jacky L Snoep, Hugh D Spence, and Barry L Wanner. Minimum information requested in the annotation of biochemical models (MIRIAM). *Nat Biotechnol*, 23(12):1509–1515, Dec 2005. doi: 10.1038/nbt1156. URL http://dx.doi.org/10.1038/nbt1156.

[Novère et al., 2006] Nicolas Le Novère, Benjamin Bornstein, Alexander Broicher, Mélanie Courtot, Marco Donizelli, Harish Dharuri, Lu Li, Herbert Sauro, Maria Schilstra, Bruce Shapiro, Jacky L Snoep, and Michael Hucka. BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acids Res*, 34(Database issue):D689–D691, Jan 2006. doi: 10.1093/nar/gkj092. URL http://dx.doi.org/10.1093/nar/gkj092.

[Oberhardt et al., 2009] Matthew A Oberhardt, Bernhard Ø Palsson, and Jason A Papin. Applications of genome-scale metabolic reconstructions. *Mol Syst Biol*, 5:320, 2009. doi: 10.1038/msb.2009.77. URL http://dx.doi.org/10.1038/msb.2009.77.

[Osterman and Overbeek, 2003] Andrei Osterman and Ross Overbeek. Missing genes in metabolic pathways: a comparative genomics approach. *Curr Opin Chem Biol*, 7(2):238–251, Apr 2003.

[Palsson, 2009] Bernhard Palsson. Metabolic systems biology. *FEBS Lett*, 583(24):3900–3904, Dec 2009. doi: 10.1016/j.febslet.2009.09.031. URL http://dx.doi.org/10.1016/j.febslet.2009.09.031.

[Papin et al., 2003] Jason A Papin, Nathan D Price, Sharon J Wiback, David A Fell, and Bernhard O Palsson. Metabolic pathways in the post-genome era. *Trends Biochem Sci*, 28(5):250–258, May 2003.

[Papin et al., 2004] Jason A Papin, Joerg Stelling, Nathan D Price, Steffen Klamt, Stefan Schuster, and Bernhard O Palsson. Comparison of network-based pathway analysis methods. *Trends Biotechnol*, 22(8):400–405, Aug 2004. doi: 10.1016/j.tibtech.2004.06.010. URL http://dx.doi.org/10.1016/j.tibtech.2004.06.010.

[Pramanik and Keasling, 1997] J. Pramanik and J. D. Keasling. Stoichiometric model of Escherichia coli metabolism: Incorporation of growth-rate dependent biomass composition and mechanistic energy requirements. *Biotechnol Bioeng*, 56(4):398–421, Nov 1997. doi: 3.0.CO;2-J. URL http://dx.doi.org/3.0.CO;2-J.

[Risso et al., 2009] Carla Risso, Jun Sun, Kai Zhuang, Radhakrishnan Mahadevan, Robert DeBoy, Wael Ismail, Susmita Shrivastava, Heather Huot, Sagar Kothari, Sean Daugherty, Olivia

Bui, Christophe H Schilling, Derek R Lovley, and Barbara A Methé. Genome-scale comparison and constraint-based metabolic reconstruction of the facultative anaerobic Fe(III)-reducer Rhodoferax ferrireducens. *BMC Genomics*, 10:447, 2009. doi: 10.1186/1471-2164-10-447. URL http://dx.doi.org/10.1186/1471-2164-10-447.

[Sauro et al., 2003] Herbert M Sauro, Michael Hucka, Andrew Finney, Cameron Wellock, Hamid Bolouri, John Doyle, and Hiroaki Kitano. Next generation simulation tools: the Systems Biology Workbench and BioSPICE integration. *OMICS*, 7(4):355–372, 2003. doi: 10.1089/153623103322637670. URL http://dx.doi.org/10.1089/153623103322637670.

[Schilling et al., 2000] C. H. Schilling, D. Letscher, and B. O. Palsson. Theory for the systemic definition of metabolic pathways and their use in interpreting metabolic function from a pathway-oriented perspective. *J Theor Biol*, 203(3):229–248, Apr 2000. doi: 10.1006/jtbi.2000.1073. URL http://dx.doi.org/10.1006/jtbi.2000.1073.

[Schuster et al., 1999] S. Schuster, T. Dandekar, and D. A. Fell. Detection of elementary flux modes in biochemical networks: a promising tool for pathway analysis and metabolic engineering. *Trends Biotechnol*, 17(2):53–60, Feb 1999.

[Selvarasu et al., 2010] Suresh Selvarasu, Iftekhar A Karimi, Ghi-Hoon Ghim, and Dong-Yup Lee. Genome-scale modeling and in silico analysis of mouse cell metabolic network. *Mol Biosyst*, 6 (1):142–151, Jan 2010. doi: 10.1039/b912865d. URL http://dx.doi.org/10.1039/b912865d.

[Snyder and Gallagher, 2009] Michael Snyder and Jennifer E G Gallagher. Systems biology from a yeast omics perspective. *FEBS Lett*, 583(24):3895–3899, Dec 2009. doi: 10.1016/j.febslet.2009.11.011. URL http://dx.doi.org/10.1016/j.febslet.2009.11.011.

[Vongsangnak et al., 2008] Wanwipa Vongsangnak, Peter Olsen, Kim Hansen, Steen Krogsgaard, and Jens Nielsen. Improved annotation through genome-scale metabolic modeling of Aspergillus oryzae. *BMC Genomics*, 9:245, 2008. doi: 10.1186/1471-2164-9-245. URL http://dx.doi.org/10.1186/1471-2164-9-245.

[Wendisch et al., 2006] Volker F Wendisch, Michael Bott, and Bernhard J Eikmanns. Metabolic engineering of Escherichia coli and Corynebacterium glutamicum for biotechnological production of organic acids and amino acids. *Curr Opin Microbiol*, 9(3):268–274, Jun 2006. doi: 10.1016/j.mib.2006.03.001. URL http://dx.doi.org/10.1016/j.mib.2006.03.001.

[Westerhoff and Palsson, 2004] Hans V Westerhoff and Bernhard O Palsson. The evolution of molecular biology into systems biology. *Nat Biotechnol*, 22(10):1249–1252, Oct 2004. doi: 10.1038/nbt1020. URL http://dx.doi.org/10.1038/nbt1020.

[Westerhoff et al., 2009] Hans V Westerhoff, Catherine Winder, Hanan Messiha, Evangelos Simeonidis, Malgorzata Adamczyk, Malkhey Verma, Frank J Bruggeman, and Warwick Dunn. Systems biology: the elements and principles of life. *FEBS Lett*, 583(24):3882–3890, Dec 2009. doi: 10.1016/j.febslet.2009.11.018. URL `http://dx.doi.org/10.1016/j.febslet.2009.11.018`.

## Book references and manuals

[Allen, 2009] Dan Allen. *Seam in Action*. Manning Publications, 2009.

[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`.

[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.

[Eils and Kriete, 2005] Roland Eils and Andres Kriete. *Computational Systems Biology*. Academic Press, 2005a.

[Erl, 2006] Thomas Erl. *Service-Oriented Architecture (SOA): Concepts, Technology, and Design*, volume 6. Prentice Hall PTR, 2006.

[Gamma et al., 1995] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1995.

[Goncalves, 2009] Antonio Goncalves. *Beginning Java™ EE 6 Platform with GlassFish™ 3: From Novice to Professional*. Apress, 2009.

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

[Monson-Hafael and Burke, 2006] Richard Monson-Hafael and Bill Burke. *Enterprise JavaBeans 3.0*. O'Reilly, 2006.

[Palsson, 2006] Bernhard O. Palsson. *Systems Biology: Properties of Reconstructed Networks*. Cambridge University Press, 2006.

[Pulier and Taylor, 2006] Pulier and Taylor. *Understanding Enterprise SOA*. Manning Publications, 2006.

[Rob et al., 2008] Peter Rob, Carlos Coronel, and Keeley Crockett. *Database Systems. Design, Implementation and Management*. Cengage Learning Services, 2008.

[Yuan and Heute, 2007] Michael Yuan and Thomas Heute. *Jboss®Seam: Simplicity and power beyond Java™ EE*. Prentice Hall Press, Upper Saddle River, NJ, USA, 2007. ISBN 013241273X.

## Specifications

[Bernard, 2010] JSR 303: Bean Validation. Last visited on 2010-02-11, 2010. URL `http://jcp.org/en/jsr/detail?id=303`.

[Buckley, 2003] JSR 175: A Metadata Facility for the Java™Programming Language. Last visited on 2010-02-18, 2003. URL `http://jcp.org/jsr/detail/152.jsp`.

[Burns, 2006] JSR-127: JavaServer Faces. Last visited on 2010-01-19, 2006a. URL `http://www.jcp.org/jsr/detail/127.jsp`.

[Burns, 2006] JSR-252: JavaServer Faces 1.2. Last visited on 2010-01-19, 2006b. URL `http://www.jcp.org/en/jsr/detail?id=252`.

[Burns, 2009] JSR-314: JavaServer Faces 2.0. Last visited on 2010-01-12, 2009. URL `http://www.jcp.org/en/jsr/detail?id=314`.

[Chung and Luehe, 2006] JSR 245: JavaServer™Pages 2.1. Last visited on 2010-01-19, 2006. URL `http://jcp.org/jsr/detail/245.jsp`.

[DeMichiel, 2009] JSR 317: Java™Persistence 2.0. Last visited on 2010-01-15, 2009. URL `http://jcp.org/jsr/detail/317.jsp`.

[DeMichiel and Keith, 2006] JSR 220: Enterprise JavaBeans™3.0. Last visited on 2010-02-18, 2006. URL `http://jcp.org/jsr/detail/220.jsp`.

[Hadley and Sandoz, 2009] JSR-311: JAX-RS: The Java™API for RESTful Web Services. Last visited on 2010-02-14, 2009. URL `http://jcp.org/jsr/detail/311.jsp`.

[King, 2009] JSR 299: Contexts and Dependency Injection for the Java™EE platform. Last visited on 2010-01-27, 2009. URL `http://jcp.org/en/jsr/detail?id=299`.

[Kotamraju, 2006] JSR 222: Java™Architecture for XML Binding (JAXB) 2.0. Last visited on 2010-02-09, 2006a. URL `http://jcp.org/en/jsr/detail?id=222`.

[Kotamraju, 2006] JSR 224: Java™API for XML-Based Web Services (JAX-WS) 2.0. Last visited on 2010-02-09, 2006b. URL http://jcp.org/en/jsr/detail?id=224.

[Pelegri-Llopart and Roth, 2003] JSR 152: JavaServer™Pages 2.0. Last visited on 2010-02-18, 2003. URL http://jcp.org/jsr/detail/152.jsp.

[Saks, 2009] JSR 318: Enterprise JavaBeans™3.1. Last visited on 2010-01-27, 2009. URL http://jcp.org/jsr/detail/318.jsp.

[Shannon, 2003] JSR-151: Java™2 Platform, Enterprise Edition 1.4 (J2EE 1.4) Specification. Last visited on 2010-02-21, 2003. URL http://jcp.org/jsr/detail/151.jsp.

[Shannon, 2006] JSR-244: Java™Platform, Enterprise Edition 5 (Java EE 5) Specification. Last visited on 2010-01-13, 2006. URL http://jcp.org/jsr/detail/244.jsp.

[Shannon, 2009] JSR-316: Java™Platform, Enterprise Edition 6 (Java EE 6) Specification. Last visited on 2010-01-12, 2009. URL http://jcp.org/jsr/detail/316.jsp.

## Unpublished references

[Andersen, 2008] Mikael Rørdam Andersen. *Systems Biology Studies of Aspergilli - From Sequence to Science*. PhD thesis, Technical University of Denmark, November 2008.

[Bergsten, 2004] Hans Bergsten. Improving JSF by Dumping JSP. URL http://www.onjava.com/pub/a/onjava/2004/06/09/jsf.html. Last visited on 2010-01-21, 09 2004.

[Zeller, 2003] Dieter Zeller. Design and development of a user managment system for molecular biology database systems. Master's thesis, Graz University of Technology, 2003.

## Web link references

[Ant, 2010] Apache Ant. Last visited on 2010-01-26, 2010. URL http://ant.apache.org/.

[BIGG, 2010] BIGG Database. Last visited on 2010-02-09, 2010. URL http://bigg.ucsd.edu/.

[Dogan, 2010] Database Of the Genomes Analyzed at NITE. Last visited on 2010-03-09, 2010. URL http://www.bio.nite.go.jp/dogan/.

[Eclipse, 2010] Eclipse Integrated Java Development Environment. Last visited on 2010-01-27, 2010. URL http://www.eclipse.org/.

[EclipseLink, 2010] Eclipse Persistence Services Project (EclipseLink). Last visited on 2010-01-18, 2010. URL http://www.eclipse.org/eclipselink/.

[GeronimoAS, 2010] Apache Geronimo Application Server. Last visited on 2010-01-18, 2010. URL http://geronimo.apache.org/.

[Gilbert, 2010] JFreeChart. Last visited on 2010-01-29, 2010. URL http://www.jfree.org/jfreechart/.

[Glassfish, 2010] Glassfish. Last visited on 2010-01-18, 2010. URL http://glassfish.dev.java.net/.

[Hibernate, 2010] Hibernate Persistence Framework. Last visited on 2010-01-22, 2010a. URL http://www.hibernate.org/.

[Hibernate, 2010] Hibernate Criteria. Last visited on 2010-01-14, 2010b. URL http://docs.jboss.org/hibernate/stable/core/reference/en/html/querycriteria.html.

[Hucka et al., 2010] SBML Level 2 Version 4. Last visited on 2010-01-27, 2010. URL http://sbml.org/Documents/Specifications/SBML_Level_2/Version_4.

[ICEfaces, 2010] ICEsoft Technologies Inc. ICEfaces. Last visited on 2010-01-21, 2010. URL http://www.icefaces.org.

[JavaBeans, 2010] JavaBeans. Last visited on 2010-01-20, 2010. URL http://java.sun.com/javase/technologies/desktop/javabeans/index.jsp.

[JBoss, 2010] Envers. Last visited on 2010-03-03, 2010. URL http://www.jboss.org/envers/.

[JBossAS, 2010] JBoss Application Server. Last visited on 2010-01-18, 2010. URL http://www.jboss.org/jbossas/.

[JCP, 2009] Java Community Process. Last visited on 2010-02-18, 2009. URL http://jcp.org/.

[JDBC, 2010] JDBC. Last visited on 2010-01-22, 2010. URL http://java.sun.com/javase/technologies/database/.

[JOnAS, 2010] JOnAS Application Server. Last visited on 2010-01-18, 2010. URL http://jonas.objectweb.org/.

[JSF RI, 2010] JSF Reference Implementation. Last visited on 2010-01-19, 2010. URL http://java.sun.com/javaee/javaserverfaces/.

[JSFMatrix, 2010] JSF AJAX Component Library Feature Matrix. Last visited on 2010-01-21, 2010. URL http://www.jsfmatrix.net/.

[MagicDraw, 2010] MagicDraw, UML modeling and CASE tool from. Last visited on 2010-01-27, 2010. URL http://www.magicdraw.com.

[MyEclipse, 2010] MyEclipse - Java EE plugin for Eclipse. Last visited on 2010-01-27, 2010. URL http://www.myeclipseide.com/.

[MyFaces, 2010] Apache MyFaces Project. Last visited on 2010-01-19, 2010. URL http://myfaces.apache.org/.

[OracleAS, 2010] Oracle Application Server. Last visited on 2010-01-18, 2010. URL http://www.oracle.com/appserver.

[RDF, 2010] RDF/XML Syntax Specification. Last visited on 2010-02-11, 2010. URL http://www.w3.org/TR/REC-rdf-syntax/.

[RichFaces, 2010] JBoss RichFaces. Last visited on 2010-01-21, 2010. URL http://www.jboss.org/jbossrichfaces/.

[SBML, 2010] Systems Biology Markup Language. Last visited on 2010-01-26, 2010. URL http://sbml.org/.

[Seam, 2010] The Seam Framework. Last visited on 2010-01-25, 2010. URL http://seamframework.org/.

[SOAP, 2003] W3C Specification for SOAP Version 1.2. Last visited on 2010-03-03, 2003. URL http://www.w3.org/TR/soap12.

[SQLDeveloper, 2010] Oracle SQL Developer. Last visited on 2010-01-27, 2010. URL http://www.oracle.com/technology/products/database/sql_developer/.

[Subclipse, 2010] Subclipse. Last visited on 2010-01-27, 2010. URL http://subclipse.tigris.org/.

[Subversion, 2010] Subversion Code Version Management System. Last visited on 2010-01-27, 2010. URL http://subversion.tigris.org/.

[Tapestry, 2010] Apache Tapesty Framework. Last visited on 2010-01-21, 2010. URL http://tapestry.apache.org.

[Tiles, 2010] Apache Tiles Framework. Last visited on 2010-01-21, 2010. URL http://tiles.apache.org/.

[TopLink, 2010] Oracle TopLink. Last visited on 2010-01-18, 2010. URL http://www.oracle.com/technology/products/ias/toplink/index.html.

[Trinidad, 2010] Apache Myfaces Trinidad. Last visited on 2010-01-21, 2010. URL http://myfaces.apache.org/trinidad/.

[WebSphere, 2010] WebSphere Application Server. Last visited on 2010-01-18, 2010. URL http://www.ibm.com/software/websphere/.

# List of Figures

# Appendix B

# Glossary

| | |
|---|---|
| AAS | Authentication and Authorization System |
| AJAX | Asynchronous JavaScript and XML |
| ANSI | American National Standards Institute |
| API | Application Programming Interface |
| AS | Application Server |
| CAPTCHA | Completely Automated Public Turing test to tell Computers and Humans Apart |
| CASE | Computer-Aided Software Engineering |
| ChEBI | Chemical Entities of Biological Interest |
| CRUD | Create, Read, Update, and Delete |
| CSV | Comma Separated Value |
| DBMS | Database Management Systems |
| DCL | Data Control Language |
| DDL | Data Definition Language |
| DML | Data Manipulation Language |
| DI | Dependency Injection |
| Dogan | Database Of the Genomes Analyzed at NITE |
| EC-Number | Enzyme Commission classification number |
| EIS | Enterprise Information System |
| EJB | Enterprise Java Beans |
| EL | Expression Language |
| ERP | Enterprise Resource Planning systems |
| ExPASy | Expert Protein Analysis System |

| | |
|---|---|
| FBA | Flux balance analysis |
| 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 |
| Java EE | Java Enterprise Edition |
| JAXB | Java Architecture for XML Binding |
| JAX-RPC | Java API for Remote Procedure Calls |
| JAX-RS | Java API for RESTful Web Services |
| JAX-WS | Java API for XML-Web Services |
| JBPM | Java Business Process Management |
| JCP | Java Community Process |
| JDBC | Java Database Connectivity |
| JMS | Java Messaging Service |
| JMX | Java Management Extension |
| JNDI | Java Naming and Directory Interface |
| JPA | Java Persistence API |
| JPQL | Java Persistence query language |
| JSF | Java Server Faces |
| JSP | JavaServer Pages |
| JSR | Java Specification Request |
| JTA | Java Transaction API |
| KEGG | Kyoto Encyclopedia of Genes and Genomes |
| MEMOSys | Metabolic Model System |
| MIRIAM | Minimum information requested in the annotation of biochemical models |
| MDB | Message Driven Bean |
| ORM | Object/Relational Mapping |
| POJO | Plain Old Java Object |
| QL | Query Language |
| RDBMS | Relational Database Management System |
| RDF | Resource Description Framework |

| | |
|---|---|
| REST | Representational State Transfer |
| RI | Reference Implementation |
| RMI | Remote Method Invocation |
| SFSB | Stateful Session Bean |
| SGD | Saccharomyces Genome Database |
| SLSB | Stateless Session Bean |
| SOA | Service-Oriented Architecture |
| SOAP | Simple Object Access Protocol |
| SQL | Structured Query Language |
| SSO | Single Sign On |
| UI | User Interface |
| UML | Unified Modeling Language |

# Appendix C

# Acknowledgments

This work was supported by the Christian Doppler Research Association (CDG). I would like to thank my supervisor, Zlatko Trajanoski, for his support, scientific guidance, vision, and belief in me.

Further thanks go to the members of the “Institute for Genomics and Bioinformatics” for their fruitful discussions, help, and friendship. Furthermore, I want to thank Jens Nielsen and the members of his lab for their helpful input and for giving me the opportunity to spend some time in Gothenburg. Special thanks go to Rasmus Ågren, Liming Liu, Intawat Nookaew, and Wanwipa Vongsangnak for providing the metabolic models and for their help with the database design. I would also like to acknowledge Timo Hardiman, Rudolf Mitterbauer, and Thomas Specht of Sandoz corporation for their conceptional input and testing effort.

I am indebted to my family for their unfailing support, encouragement, and understanding.

Stephan Pabinger

Graz, Austria, March 2010

# Appendix D

# Publications

**Pabinger S**, Thallinger GG, Snajder R, Eichhorn H, Rader R, Trajanoski Z. QPCR: Application for real-time PCR data management and analysis. BMC Bioinformatics 2009, 10:268 PMID: 19712446

BMC Bioinformatics

BioMed Central

Software

Open Access

# QPCR: Application for real-time PCR data management and analysis

Stephan Pabinger[1,2], Gerhard G Thallinger[1], René Snajder[1], Heiko Eichhorn[3], Robert Rader[2] and Zlatko Trajanoski*[1,2]

Address: [1]Institute for Genomics and Bioinformatics, Graz University of Technology, Petersgasse 14, 8010 Graz, Austria, [2]Christian Doppler Laboratory for Genomics and Bioinformatics, Petersgasse 14, 8010 Graz, Austria and [3]Development Anti-Infectives Microbiology, Sandoz GmbH, Biochemiestrasse 10, 6250 Kundl, Austria

Email: Stephan Pabinger - stephan.pabinger@tugraz.at; Gerhard G Thallinger - gerhard.thallinger@tugraz.at; René Snajder - rene.snajder@tugraz.at; Heiko Eichhorn - heiko.eichhorn@sandoz.com; Robert Rader - robert.rader@tugraz.at; Zlatko Trajanoski* - zlatko.trajanoski@tugraz.at

* Corresponding author

Published: 27 August 2009

*BMC Bioinformatics* 2009, **10**:268 doi:10.1186/1471-2105-10-268

Received: 2 March 2009
Accepted: 27 August 2009

This article is available from: http://www.biomedcentral.com/1471-2105/10/268



## Abstract

**Background:** Since its introduction quantitative real-time polymerase chain reaction (qPCR) has become the standard method for quantification of gene expression. Its high sensitivity, large dynamic range, and accuracy led to the development of numerous applications with an increasing number of samples to be analyzed. Data analysis consists of a number of steps, which have to be carried out in several different applications. Currently, no single tool is available which incorporates storage, management, and multiple methods covering the complete analysis pipeline.

**Results:** QPCR is a versatile web-based Java application that allows to store, manage, and analyze data from relative quantification qPCR experiments. It comprises a parser to import generated data from qPCR instruments and includes a variety of analysis methods to calculate cycle-threshold and amplification efficiency values. The analysis pipeline includes technical and biological replicate handling, incorporation of sample or gene specific efficiency, normalization using single or multiple reference genes, inter-run calibration, and fold change calculation. Moreover, the application supports assessment of error propagation throughout all analysis steps and allows conducting statistical tests on biological replicates. Results can be visualized in customizable charts and exported for further investigation.

**Conclusion:** We have developed a web-based system designed to enhance and facilitate the analysis of qPCR experiments. It covers the complete analysis workflow combining parsing, analysis, and generation of charts into one single application. The system is freely available at http://genome.tugraz.at/QPCR

## Background

Amongst other high throughput techniques like DNA microarrays and mass spectrometry, qPCR has become important in many areas of basic and applied functional genomics research. Due to its high sequence-specificity, large dynamic range, and tremendous sensitivity it is one of the most widely used methods for quantification of gene expression. Moreover, due to the adoption of robotic

pipetting stations and 384-well formats, laboratories generate a huge amount of qPCR data demanding a centralized storage, management, and analysis application.

Most software programs provided along with the qPCR instruments support only straightforward calculation of quantification cycle (Cq) values from the recorded fluorescence measurements. However, in order to get biological meaningful results these basic calculations need to undergo further analyses such as normalization, averaging, and statistical tests [1].

To this end, a variety of different methods have been published describing the normalization of Cq values. The simplest model (termed ΔΔ-Cq method) was developed by Livak and Schmittgen [2] which assumes perfect amplification efficiency by setting the base of the exponential function to 2 and uses only one reference gene for normalization. The model proposed by Pfaffl [3] considers PCR efficiency for both the gene of interest and a reference gene and is therefore an improvement over the classic ΔΔ-Cq method. Nevertheless, it still uses only one reference gene which may not be sufficient to obtain reliable results [4]. Hellemans *et al.* [5] proposed an advanced method which considers gene-specific amplification efficiencies and allows normalization of Cq values with multiple reference genes based on the method proposed by Vandesompele *et al.* [4]. It should be noted that these methods could differ substantially in their performance, because of the different assumptions they are based on.

Available software tools often cover only single steps in the analysis pipeline compelling researchers to use multiple tools for the analysis of qPCR experiments [5-8]. However, these tools do not share a common file format making it difficult to analyze the experimental data. Additionally, no standardization of methodology has been established that would be needed for relatable comparison between laboratories [9]. Recently, the Minimum Information for Publication of Quantitative Real-Time PCR Experiments (MIQE) guidelines [10] were published which are intended to describe the minimum information necessary for evaluating and comparing qPCR experiments. Based on a subset of these guidelines the XML-based Real-Time PCR Data Markup Language (RDML) [11] was proposed which tries to facilitate the exchange of qPCR data and related information between qPCR instruments, analysis software, journals, and public repositories. These efforts could allow a more reliable interpretation of qPCR results if they were accepted in the qPCR community.

The lack of complete or partial assessment of error propagation throughout the whole analysis pipeline may result in an underestimated final error and could therefore lead to incorrect conclusions. Moreover, the analysis of experiments using tools that make invalid biological assumptions can cause significantly wrong results as reported in [8].

To the best of our knowledge, there is no single tool available which integrates storage, management, and analysis of qPCR experiments. Hence a system enabling comparison of results and providing a standardized way of analyzing data would be of great benefit to the community. We have therefore developed QPCR, a web-based application which supports: a) technical and biological replicate handling, b) the analysis of qPCR experiments with an unlimited number of samples and genes, c) normalization using an arbitrary number of reference genes, d) inter-plate normalization using calibrators, e) assessment of significant gene deregulation between sample groups, f) generation of customizable charts, and g) a plug-in mechanism for easy integration of new analysis methods.

## Implementation

The QPCR system was implemented in Java, a platform independent and object-oriented programming language [12]. The application is based on the Java 2 Enterprise Edition (J2EE) three-tier architecture consisting of a presentation-, business -, and database-layer. A relational database (PostgreSQL or Oracle) is used as the persistence backend. The business layer consists of Enterprise Java Beans (EJB) and is deployed on a JBoss [13] application server. The presentation layer is based on the Model-View-Controller (MVC) framework Struts [14] and uses Java Servlets and Java Server Pages.

In order to enhance usability current web technologies have been extensively used in this application. AJAX functionality has been incorporated into the application using the open-source library DWR [15]. This technology allows asynchronous loading of data without the need to reload the page thus providing a desktop like application behavior. Multiple JavaScript libraries (Prototype [16], JQuery [17]) have been used that allow executing functions on the client side and therefore remarkably improve the usability of the application. Charts are generated using the open-source Java library JFreeChart [18] and all charts are created either in the lossless PNG format or as a scalable vector graphic (SVG).

All algorithms, calculation methods, and data file parsers used by the application are integrated through a plug-in mechanism which allows simple extension with additional qPCR data formats and analysis approaches. For each class that uses the plug-in mechanism a specific interface needs to be implemented in order to support another vendor or implement an additional analysis method. The

new Java classes are then automatically detected by the QPCR application.

Currently the data file parsers support files generated by Applied Biosystems (ABI 7000, ABI 7500, ABI 7900) and Roche LightCycler (LightCycler 2.0, LightCycler 480) [19] systems as well as a generic file format based on comma separated values (CSV). Since not all fluorescence measurements can be extracted from data files created by the qPCR instrument systems, additional export files are required to parse all relevant data.

Analysis methods that calculate Cq and amplification efficiency values are computationally expensive and are therefore executed asynchronously and do not interfere with the QPCR web interface. They are designed to operate on a per well basis and report the current progress of the calculation. Normalization methods and statistical tests are not time consuming processes and are therefore executed in real time.

The QPCR application has been designed using the Unified Modeling Language (UML) [20]. The use of a UML representation improves maintainability as the application architecture is outright visible and provides an important part of the system documentation. We used the AndroMDA framework [21] to create basic EJB and presentation tier source code as well as configuration files based on the UML model. AndroMDA minimizes repetitive coding tasks, allows to easily extend or edit the architecture of the application, and helps maintaining the consistency between design and implementation.

The stored data is secured by a user management system which allows the definition of several fine grained user access levels and offers data sharing and concurrent access in a multi-centric environment [22]. Moreover, the application provides two configurations which assign the ownership of objects either to the submitter or to the submitter's institute. The latter setup provides the possibility to edit and analyze experiments by all users of an institute without the need to explicitly share objects.

## Results

QPCR is an application which integrates storage, management, and analysis of qPCR experiments into one single tool. Implemented as a web application it can be accessed by a web browser from every network connected computer and therefore supports the often decentralized work of biologists. It parses files generated by qPCR instruments, stores data and results in a database, and performs analyses on the imported data. Moreover, it allows conducting of statistical tests and provides several ways to visualize and export the calculated results (Figure 1).

### *Parsing files and calculation of Cq/efficiency values*

Data files are uploaded into the application using a single file upload dialog or an integrated Java applet which supports uploading of multiple files at once. An upload zone lists all available files and allows querying and downloading of data previously uploaded. All files are stored in a user defined directory facilitating the backup of project critical files.

After uploading the exported files into the QPCR application, a list of all files which have not yet been processed is shown. The user can select single or multiple files for parsing. Moreover, Cq and amplification efficiency values can be automatically calculated after the files have been parsed using one or several different methods.

During parsing all relevant data is extracted, including plate setup, fluorescence measurements, and qPCR instrument specifications and stored in the database. In contrast to many available analysis tools the application is able to import qPCR data files without the need for additional file manipulations and therefore reduces error-prone and cumbersome manual work. In addition to the already existing data file parsers the application can be easily extended to support other vendors due to the modularity of the platform and the used plug-in mechanism.

Once the data is parsed and stored in the database, Cq and amplification efficiency values are calculated based on the fluorescence measurements. Several published and widely-used algorithms were implemented; two different algorithms to calculate Cq together with efficiency values, three different algorithms to calculate solely the amplification efficiency, and one method to calculate the Cq value are available (see Table 1).

The progress of all active parser or analyzer background tasks is displayed on a view that automatically updates the current status. As soon as a process has finished a message is shown at the top of the page. For each process a log file is created which informs the user about the outcome of the performed job. A color scheme helps to quickly identify the jobs that have not finished successfully.

During parsing of uploaded files a *Run* is created in the application which is a direct representation of the performed qPCR run. It stores information about the hardware, software, thermocycler profile, and category.

Each *Run* contains a plate which consists of multiple wells that store information about the sample, target, passive reference, task, and omitted status. The plate layout can be displayed in a list and each well can be edited to correct inconsistencies or to omit it from further analysis.

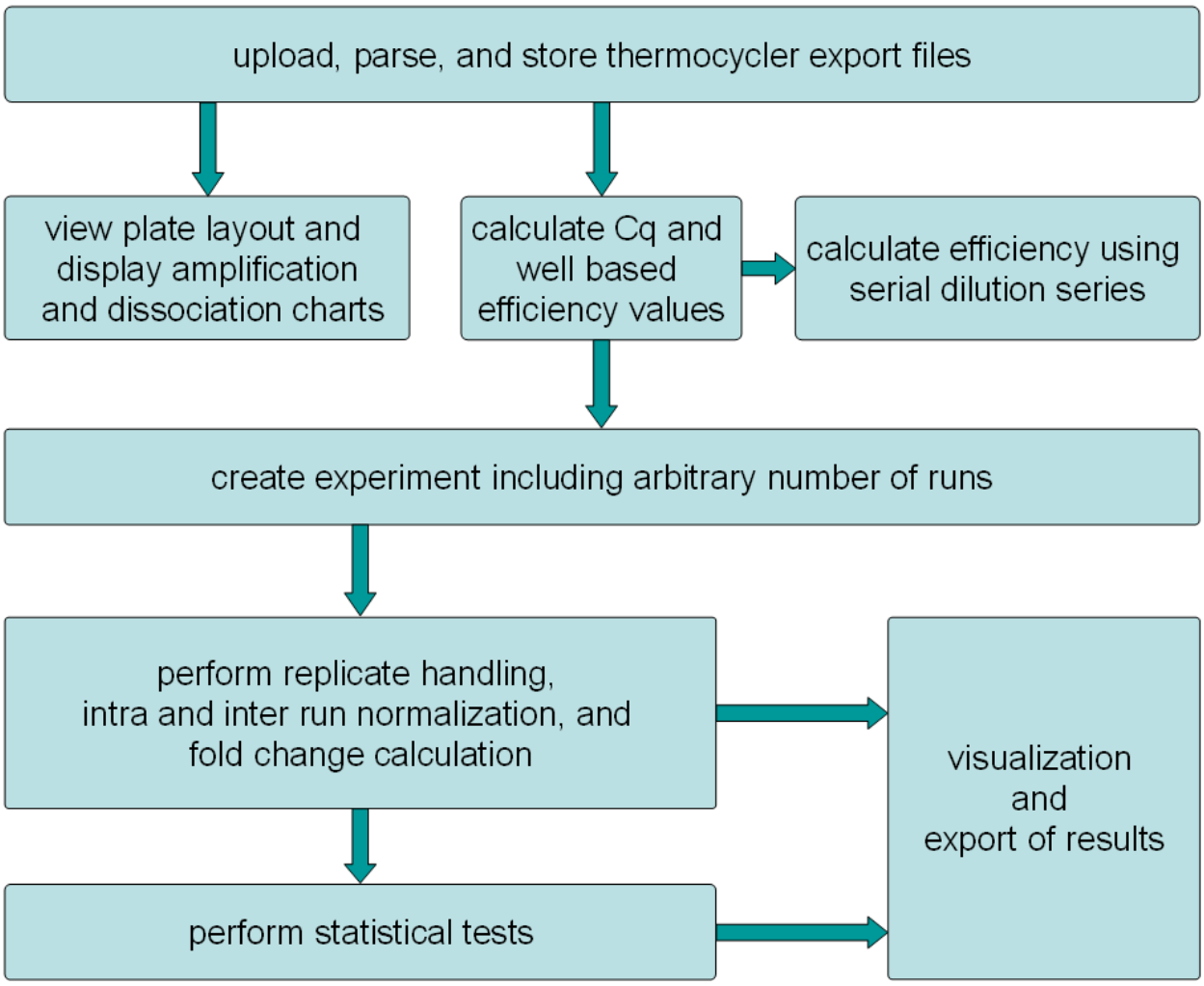


**Figure 1**
**Analysis pipeline**. This figure illustrates the analysis pipeline implemented in the QPCR application.

Additionally, QPCR provides a graphical representation of the plate layout by showing a grid which displays sample, target, and status information of each well. By selecting an arbitrary number of wells, charts of amplification (raw and background subtracted) and dissociation (raw and derivative) curves are displayed (Figure 2). This view is helpful to evaluate the performance of the PCR for each well and is useful to perform a quick quality check of the conducted qPCR run.

### *Analysis of experiments*

After Cq and efficiency values have been determined, experiments consisting of one or multiple runs are subjected to subsequent analysis steps. Several plates can be combined into one experiment. In order to support a flexible and adaptable analysis of experiments, the application allows selecting of specific samples and genes to be used in subsequent analysis steps. Moreover, the Cq calculation method, the efficiency method, and the reference genes can be defined.

Four different ways to consider amplification efficiencies in the analysis have been implemented: (1) setting a single efficiency value for all targets, (2) manually defining the efficiency for each target, (3) using efficiencies derived from dilution series for each target, and (4) using calculated efficiencies for each well. Several different efficiencies values for a target, calculated by serial dilution series, can be stored in the database.

Normalization of experiments is based on a method proposed by Hellemans *et al.* [5] and includes averaging of

**Table 1: Algorithms used in the QPCR web application**

| Name | Description | Author | Cq | Efficiency |
|---|---|---|---|---|
| Cy0 | The method operates on raw fluorescence data and fits a five parameter Richard's curve. Next it calculates the tangent of the inflection point and intersects it with the abscissa axis which results in the Cq value. | Guescini *et al.* [27] | ✓ | - |
| LinReg | The method operates on background corrected data and uses log transformed values to construct a slope with the highest correlation to the original curve. The parameters of this slope are used to calculate the amplification efficiency. | Ramakers *et al.* [8] | - | ✓ |
| Miner | The method operates on raw fluorescence data and fits a four-parameter logistic curve to determine the Cq value by using the second derivative maximum. The efficiency is calculated by using a weighted average of a fitted exponential curve. | Zhao *et al.* [28] | ✓ | ✓ |
| RutledGene | The method operates on raw fluorescence data and fits a four-parametric sigmoid function to calculate efficiency values. | Rutledge *et al.* [29] | - | ✓ |
| SoFar | The method operates on raw fluorescence data and fits exponential or sigmoid function on smoothed data to calculate Cq and efficiency values. | Wilhelm *et al.* [30] | ✓ | ✓ |
| TAQ | The method operates on raw fluorescence data and performs a linear regression on log transformed data to determine the efficiency values. | Ostermeier *et al.* [31] | - | ✓ |

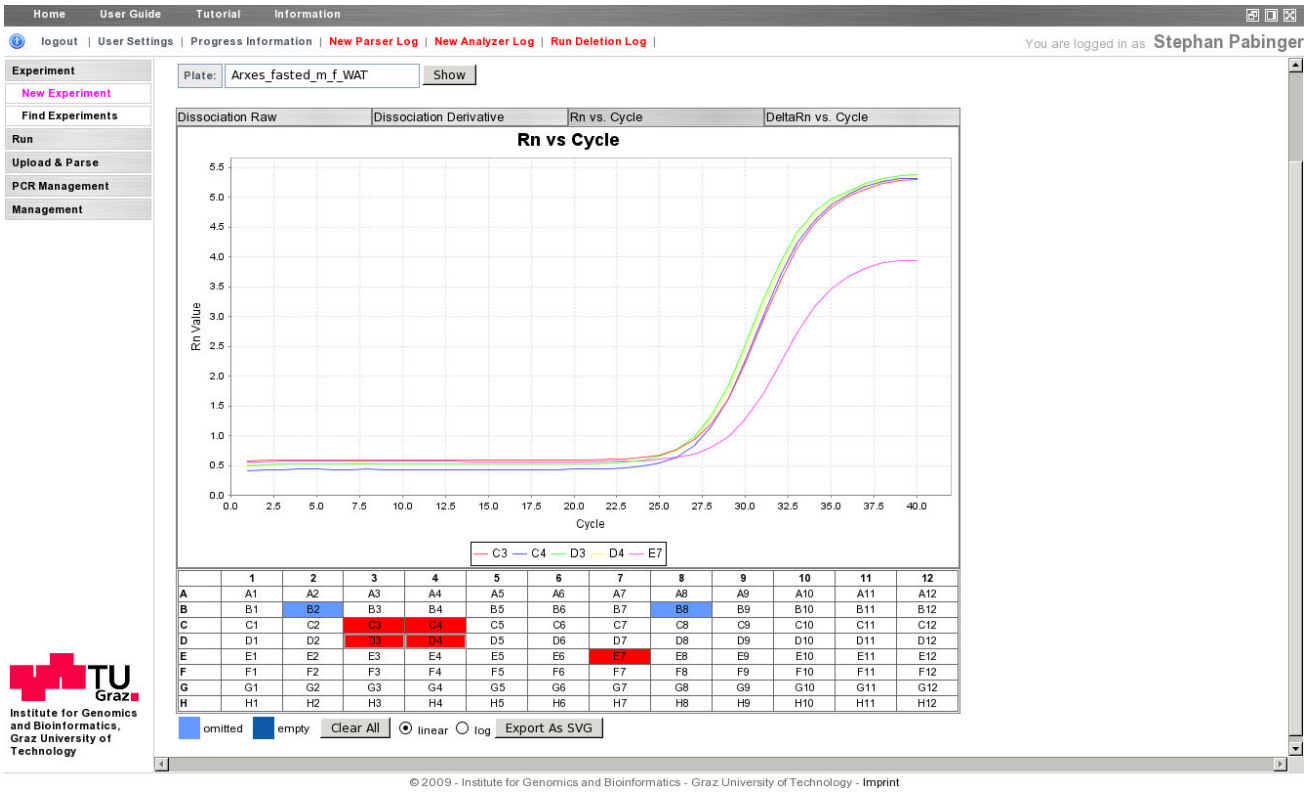


**Figure 2**
**Graphical representation of the plate layout**. The tabbed bar at the top is used to switch between different chart types. The chart itself features tool tips and provides a legend. Beneath the chart is a representation of the plate layout that is adapted to the plate size (96/384 wells, linear layout). Selected wells are colored in red, omitted and empty wells in blue.

technical replicates, normalization against reference genes, inter-run calibration, and calculation of quality control parameters. Technical replicates are averaged either within one plate or over all plates of the experiment depending on the analysis setting. In the next step all samples of one gene are referenced to the arithmetic mean Cq value across all samples for this gene. Thereafter the user selected type of efficiency is considered for each target and the samples are normalized to the selected reference genes. If reaction specific efficiency has been selected the efficiency is averaged for each target. Depending on the analysis setting the application supports spreading of reference genes across multiple runs or uses reference genes for each run independently. Finally, inter-run calibrators are automatically detected and are used to normalize results between different qPCR runs.

Quality control parameters for reference genes are calculated based on a method described by Vandesompele *et al.* [4]. When multiple reference genes are selected the coefficient of variation and the gene stability value M are calculated. These parameters are helpful for selecting and evaluating reference genes. Additionally, QPCR performs outlier detection by calculating the difference in quantification cycle value between technical replicates and allows highlighting those that have a larger difference than a user defined threshold. Moreover, quality control checks are performed to test if a no template control (NTC) is present for each target.

Fold change ratios of the calculated normalized Cq values can be calculated by referencing them to one or multiple samples. All analysis setup parameters are automatically stored in the database and are loaded when the experiment is analyzed again. Additionally, each analysis setup can be stored under a user defined name. Throughout the whole analysis process proper error propagation is performed using methods described in [5,23].

During the development of the QPCR application special attention was laid on the accurate and user-friendly visualization of calculated results. Therefore, the application allows to display and export results of every important analysis step. The generated figures are highly customizable and are designed to be usable in publications without further manipulation. Among other parameters QPCR allows to define color, labeling, sort sequence, and data type to be used in histogram charts. Cq values normalized by reference genes and calibrators are presented as histograms displaying results of one gene or multiple genes at once (Figure 3). Every result throughout the analysis pipeline can be exported in tab-delimited or spreadsheet format (txt, csv, xls) to be used in external applications.

#### *Conducting statistical tests*

The final step in the analysis pipeline is the comparison of samples using statistical tests (e.g.: biological replicates, samples of a time series). The application allows to group samples into an arbitrary number of classes which are tested for their significant difference against one defined reference class. QPCR includes several statistical tests to compute p-values such as ANOVA, student's t-test, and a permutation based test which makes no assumption on the distribution of the data. Tests can be conducted on either untransformed or log2 transformed values. The application allows adjusting the calculated p-value by supporting several established correction methods for multiple testing [24].

Calculated test results are displayed for each class and can be exported for further analysis. Moreover, the fold changes of samples are displayed in histogram charts in which samples of each class are grouped together. Every class is assigned to a specific user defined color or shape that is used in different shades to group the samples of one class (Figure 4).

#### *General data entry and query*

The application provides views of every entity to (1) manually enter data and (2) list available items. Entry views consist of mandatory and optional fields and use drop down selection lists to specify references to other entities. Entered data is checked for validity and the user is informed about erroneous inputs. List views present the data in tabular form and support paging, sorting, and querying for any combination of the available attributes. Moreover, queries can be stored in the database for later use.

## Discussion

We have developed an integrated platform for the analysis and management of qPCR experiment data using state-of-the-art software technology. The uniqueness of the application is defined by the support of various qPCR instruments, multiple data analyzers, and statistical methods, as well as the coverage of the complete analysis pipeline including proper error propagation. Moreover, it provides a flexible plug-in mechanism to incorporate new parsers and methods and allows generation of highly customizable charts. A comparison of features between QPCR and several other popular qPCR analysis tools is provided in Table 2.

The capability to import and parse data without the need for further file manipulations is an integral part of the application which avoids errors during the analysis and reduces the time to analyze the experimental data. As most of the available qPCR software tools rely on special formatted input files it was a prerequisite of the platform

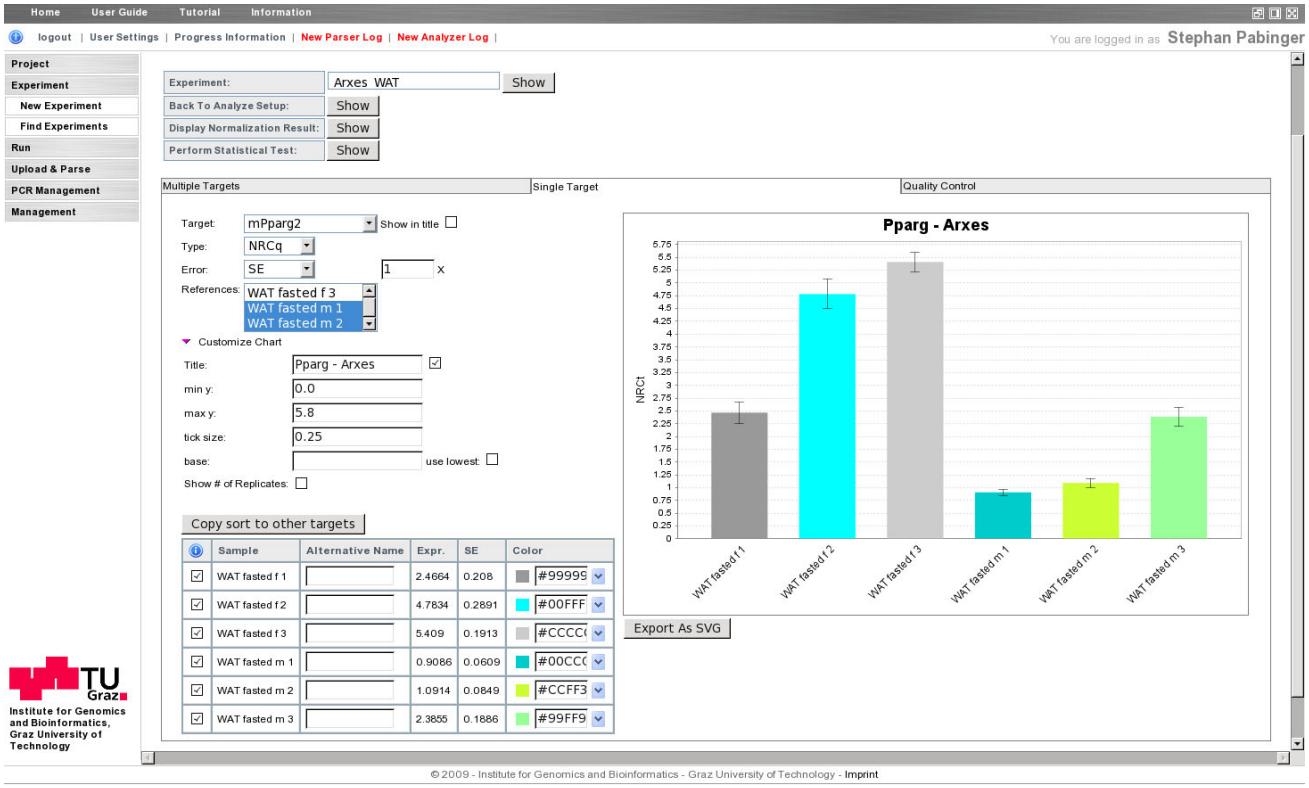


**Figure 3**

**Visualization of normalized relative quantities**. The tabbed bar is used to switch between views that display multiple targets at once, one target at a time (displayed), or quality control parameters. On the left side the user can define various parameters including the displayed target, the specific result, the presented error, and the reference samples. The list of displayed samples can be reordered using drag and drop, samples may be excluded from the chart, and for each sample an alternative name and an individual color can be assigned.

to be able to directly parse files generated by the qPCR instruments software suits. Moreover, the system is not confined to a specific manufacturer and can therefore be used in laboratories equipped with qPCR instruments from different vendors.

QPCR includes established and widely used methods for the calculation of Cq and amplification efficiency values and supports an easy integration of new algorithms. This framework does not limit the researcher to one specific approach and allows incorporation of newly developed analysis methods. Furthermore, it is of great value as different experimental situations need to be considered separately and it remains up to individual researches to identify the method most appropriate for their experimental conditions [25]. QPCR allows to store several different analysis settings for each experiment and calculates quality control parameters which help to evaluate the performed analysis. Incorporating several different methods to include the amplification efficiency enhances the flexibility of the application and allows adapting the analysis to the experimental conditions or laboratory practices. Particularly, supporting the widely used calculation of efficiency based on serial dilution series increases the acceptance in the qPCR community.

An often underestimated drawback of using multiple tools to analyze qPCR experiments is the lack of support for assessment of error propagation. Therefore the final error is often based solely on the standard deviation of biological replicates which can lead to false biological interpretations. The QPCR application addresses this problem and includes assessment of error propagation throughout the whole analysis pipeline covering technical replicate handling, normalization, inter-run calibration, referencing against samples, and biological replicate handling. The implemented method is based on Taylor series expansion which allows direct calculation of the full probability distribution and is in contrast to Monte Carlo based methods computationally inexpensive [26].

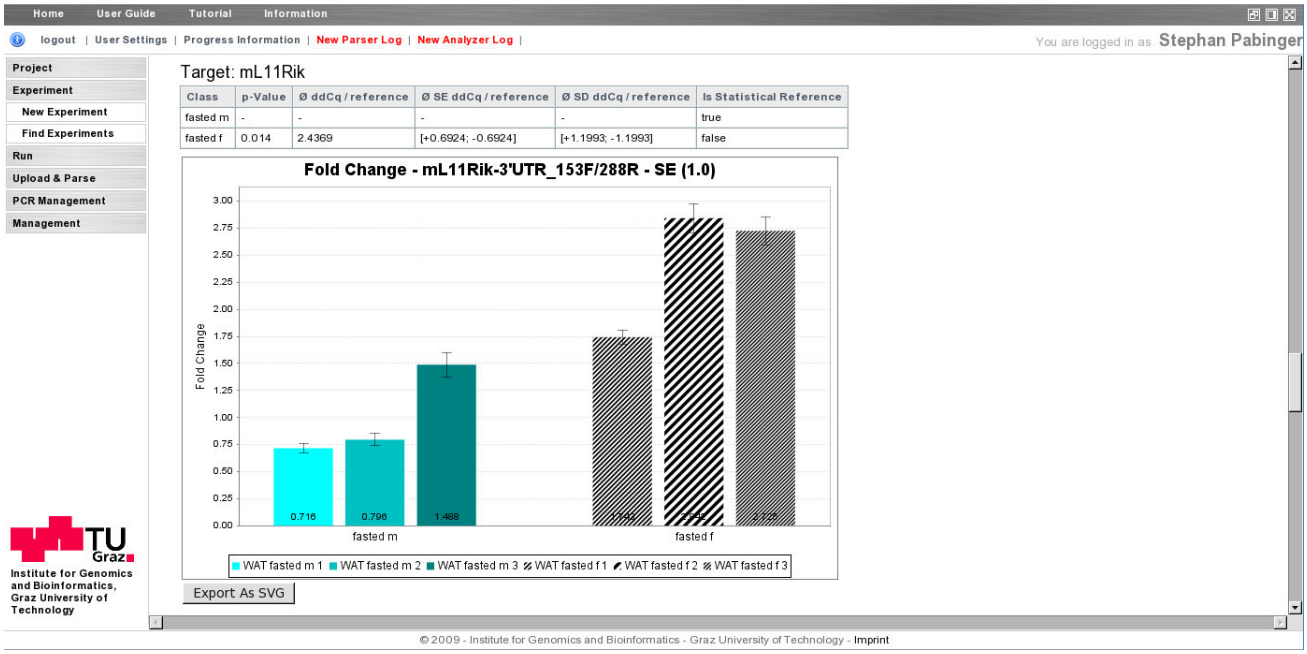


**Figure 4**

**Visualization of a statistical test result**. The statistical test was used to test two classes of biological replicates for their significant differences, whereas class "*fasted m*" was used as reference class. The table shows the calculated p-Value and parameters. Samples of each class are grouped together and marked in different colors or shapes.

Special focus was laid on the presentation of analysis results. QPCR provides an interface which uses state-of-the-art software technologies to generate highly customizable charts that are designed to be ready for publication. Since many available tools do not provide a suitable graphical representation of the calculated results, Microsoft Excel is often used to create figures which require manual import and/or conversion of data. QPCR combines the calculation and presentation of results into one single tool which reduces analysis time and avoids additional potential error-prone steps. A flowchart displaying each analysis step and its suggested method is included into the user guide.

The recent developments of data exchange formats (RDML) and guidelines describing the minimum information about qPCR experiments (MIQE) could become an important part in standardizing qPCR experimental data. QPCR already integrates the suggested nomenclature and RDML support will be implemented as soon as the relevant Java libraries are available. Once established in the qPCR community these initiatives will allow a standardized exchange of data between software tools and facilitate the comparison of qPCR experiments.

Using three-tier software architecture that separates the presentation, the business, and the database layer enables not only easy maintenance but also allows distribution of the computing load to several servers. As more and more data needs to be analyzed this design may be very valuable in the future.

The use of a database allows easy querying and comparing of data and guarantees data integrity. The implemented plug-in framework, which is used for including data file parsers, analysis methods, and statistical algorithms, ensures that the application is adaptable to new developments and allows the effortless integration of innovative scientific methods.

## Conclusion

We have developed QPCR, a system for the storage, management, and analysis of qPCR data. It integrates the complete analysis workflow, ranging from Cq determination over normalization and statistical analysis to visualization, into a single application. The analysis time is significantly reduced and complex analyses can now be compared within a single or across multiple laboratories. Optimal usability has been ensured by involving biologists throughout the entire development process and by extensive tests in a laboratory setting. Given the incorporation of several analysis methods and the flexibility due to the use of standard software technology and plug-in mechanism, the developed application could be of great interest to the qPCR community.

**Table 2: Comparison of qPCR analysis tools**

| Feature | QPCR | StatMiner 3.0 (Integromics) [32] | qBase 1.3.5 [5] | qBasePlus 1.1 (Biogazelle) [33] | GenEx 4.4.2 (MultiD) [34] | qPCR-DAMS [6] |
|---|---|---|---|---|---|---|
| Import (Applied Biosystems, Lightcycler, BioRad, Excel) | ✓[1]/✓[2]/-/✓ | ✓/-/-/✓ | ✓/✓/✓/✓ | ✓/✓/✓/✓ | -/-/-/✓ | -/-/-/✓ |
| Absolute/relative quantification | -/✓ | ✓/✓ | -/✓ | -/✓ | ✓/✓ | ✓/✓ |
| RDML compliant | -[3] | - | - | ✓ | - | - |
| Calculate Cq | ✓ | - | - | - | - | ✓ |
| Calculate reaction specific efficiency | ✓ | - | - | - | - | - |
| Calculate target specific efficiency | ✓ | ✓ | ✓ | ✓ | ✓ | - |
| Includes geNorm, Normfinder | ✓/- | ✓/✓ | ✓/- | ✓/- | ✓/✓ | -/- |
| Normalization | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Inter-run calibration | ✓ | - | ✓ | ✓ | - | ✓ |
| Calculate fold change ratios | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Quality control | ✓ | ✓ | ✓ | ✓ | - | ✓ |
| Statistical tests | ✓ | ✓ | - | - | ✓ | - |
| Clustering | - | ✓ | - | - | ✓ | - |
| Correlation | - | - | - | - | ✓ | - |
| Error propagation | ✓ | - | ✓ | ✓ | - | - |
| 2D, 3D, Scatter, Bar plots | ✓/-/-/✓ | ✓/-/-/✓ | -/-/-/✓ | -/-/-/✓ | ✓/✓/✓/✓ | -/-/-/- |
| Multi-user functionality | ✓ | - | - | - | - | - |
| Database storage | ✓ | ✓ | - | - | - | ✓ |
| Web/Standalone/Plugin | ✓/-/- | -/✓/- | -/-/✓[4] | -/✓/- | -/✓/- | -/-/✓[5] |
| Freely available | ✓ | - | ✓ | - | - | ✓ |

(1) ABI 7000, ABI 7500, ABI 7900; (2) LightCycler 2.0, LightCycler 480; (3) Uses the suggested nomenclature; (4) Microsoft Excel based tool: (5) Microsoft Access based tool

## Availability and requirements

- Project name: QPCR
- Project home page: http://genome.tugraz.at/QPCR
- Operating system: Solaris, Linux, Windows, Mac OS X
- Programming language: Java
- Other requirements: Java JDK 1.6.x, Oracle™ 9i or PostgreSQL™ 8.0.x, a server with at least 1 GB of main memory (2 GB are recommended) available to the application

• License: IGB-TUG Software License

• Any restrictions to use by non-academics: IGB-TUG Software License

Installation of the application is provided through an installer and should be completed within one hour provided the necessary database access rights are granted. We recommend installing the application on a central server by a system administrator. Step-by-step instructions are provided at the projects web site together with the installer file. The reference installation of QPCR is running on a SUN Fire™ X4600 M2 6 × dual core Opteron server (Sun Microsystems Ges.m.b.H, Vienna, Austria) with 24 GB of memory running Solaris and using a dedicated Oracle 10 g database server. Attached is a Storage Area Network (EVA 5000, Hewlett-Packard Ges.m.b.H., Vienna, Austria) with 9.5 TBytes net capacity.

## Authors' contributions

SP designed the application and drafted the manuscript. He was responsible for implementation of the database, the development the data presentation and many parts of the business logic. GGT contributed to conception and design of the application and helped drafting the manuscript. RS improved the data file parsers and analysis methods. HE gave valuable input regarding the usability of the platform. RR participated in the design and implementation of the application and helped drafting the manuscript. ZT was responsible for the overall project coordination. All authors gave final approval of the version to be published.

## Acknowledgements

This work was supported by the Austrian Ministry of Science and Research, GEN-AU program (project Bioinformatics Integration Network) and the Christian-Doppler Society. We thank Anne Krogsdam and Andreas Prokesch for valuable discussions and Roman Fiedler for implementing the initial file parser.

## References

1. Wong ML, Medrano JF: **Real-time PCR for mRNA quantitation.** *Biotechniques* 2005, **39:**75-85.
2. Livak KJ, Schmittgen TD: **Analysis of relative gene expression data using real-time quantitative PCR and the 2(-Delta Delta C(T)) Method.** *Methods* 2001, **25:**402-408.
3. Pfaffl MW: **A new mathematical model for relative quantification in real-time RT-PCR.** *Nucleic Acids Res* 2001, **29:**E45.
4. Vandesompele J, De P, Pattyn F, Poppe B, Van R, De P, Speleman F: **Accurate normalization of real-time quantitative RT-PCR data by geometric averaging of multiple internal control genes.** *Genome Biol* 2002, **3:**RESEARCH0034.
5. Hellemans J, Mortier GR, De P, Speleman F, Vandesompele J: **qBase relative quantification framework and software for management and automated analysis of real-time quantitative PCR data.** *Genome Biol* 2007, **8:**R19.
6. Jin N, He K, Liu L: **qPCR-DAMS: a database tool to analyze, manage, and store both relative and absolute quantitative real-time PCR data.** *Physiol Genomics* 2006, **25:**525-527.
7. Simon P: **Q-Gene: processing quantitative real-time RT-PCR data.** *Bioinformatics* 2003, **19:**1439-1440.
8. Ramakers C, Ruijter JM, Deprez RH, Moorman AF: **Assumption-free analysis of quantitative real-time polymerase chain reaction (PCR) data.** *Neurosci Lett* 2003, **339:**62-66.
9. Bustin SA: **Quantification of mRNA using real-time reverse transcription PCR (RT-PCR): trends and problems.** *J Mol Endocrinol* 2002, **29:**23-39.
10. Bustin SA, Benes V, Garson JA, Hellemans J, Huggett J, Kubista M, Mueller R, Nolan T, Pfaffl MW, Shipley GL, Vandesompele J, Wittwer CT: **The MIQE Guidelines: Minimum Information for Publication of Quantitative Real-Time PCR Experiments.** *Clin Chem* 2009, **55:**611-622.
11. Lefever S, Hellemans J, Pattyn F, Przybylski DR, Taylor C, Geurts R, Untergasser A, Vandesompele J: **RDML: structured language and reporting guidelines for real-time quantitative PCR data.** *Nucleic Acids Res* 2009, **37:**2065-2069.
12. Gosling J, Joy B, Steele G, Bracha G: *The Java(TM) Language Specification* 3rd edition. Boston: Addison-Wesley Professional; 2005.
13. JBoss Group: **JBoss Application Server.** 2008 [http://www.jboss.org/jbossas/].
14. Apache Software Foundation: **Apache Struts.** 2006 [http://struts.apache.org/].
15. Getahead: **DWR: Easy AJAX for JAVA.** 2008 [http://directwebremoting.org].
16. Prototype Core Team: **Prototype: JavaScript Framework.** 2009 [http://www.prototypejs.org/].
17. John Resig and jQuery Team: **jQuery.** 2009 [http://jquery.com/].
18. Gilbert David: **The JFreeChart Class Library.** 2008 [http://www.jfree.org/jfreechart/].
19. Wittwer CT, Ririe KM, Andrew RV, David DA, Gundry RA, Balis UJ: **The LightCycler: a microvolume multisample fluorimeter with rapid temperature control.** *Biotechniques* 1997, **22:**176-181.
20. Booch G, Rumbaugh J, Jacobson I: *The Unified Modeling Language User Guide* 2nd edition. Boston, MA, USA, Addison-Wesley Professional; 2005.
21. AndroMDA Core Team: **AndroMDA.** 2007 [http://www.andromda.org/].
22. Maurer M, Molidor R, Sturn A, Hartler J, Hackl H, Stocker G, Prokesch A, Scheideler M, Trajanoski Z: **MARS: microarray analysis, retrieval, and storage system.** *BMC Bioinformatics* 2005, **6:**101.
23. Larionov A, Krause A, Miller W: **A standard curve based method for relative real time PCR data processing.** *BMC Bioinformatics* 2005, **6:**62.
24. Dudoit S, Shaffer JP, Boldrick J: **Multiple Hypothesis Testing in Microarray Experiments.** *U C Berkeley Division of Biostatistics Working Paper Series Working Paper 110* 2002 [http:www.bepress.cocgviewcontent.cgi?article=1014&context=ucistat].
25. Bustin SA, Benes V, Nolan T, Pfaffl MW: **Quantitative real-time RT-PCR – a perspective.** *J Mol Endocrinol* 2005, **34:**597-601.
26. Gerards BM: *Error Propagation In Environmental Modelling With GIS* Bristol, PA, USA, Taylor & Francis; 1998.
27. Guescini M, Sisti D, Rocchi MB, Stocchi L, Stocchi V: **A new real-time PCR method to overcome significant quantitative inaccuracy due to slight amplification inhibition.** *BMC Bioinformatics* 2008, **9:**326.
28. Zhao S, Fernald RD: **Comprehensive algorithm for quantitative real-time polymerase chain reaction.** *J Comput Biol* 2005, **12:**1047-1064.
29. Rutledge RG: **Sigmoidal curve-fitting redefines quantitative real-time PCR with the prospective of developing automated high-throughput applications.** *Nucleic Acids Res* 2004, **32:**e178.
30. Wilhelm J, Pingoud A, Hahn M: **SoFAR: software for fully automatic evaluation of real-time PCR data.** *Biotechniques* 2003, **34:**324-332.
31. Ostermeier GC, Liu Z, Martins RP, Bharadwaj RR, Ellis J, Draghici S, Krawetz SA: **Nuclear matrix association of the human beta-globin locus utilizing a novel approach to quantitative real-time PCR.** *Nucleic Acids Res* 2003, **31:**3257-3266.
32. Integromics: **RealTime StatMiner.** 2009 [http://www.integromics.com/StatMiner.php].
33. Biogazelle: **qBasePlus.** 2009 [http://www.biogazelle.com/site/products/qbaseplus].
34. MultiD: **GenEx.** 2009 [http://www.multid.se/genex.html].