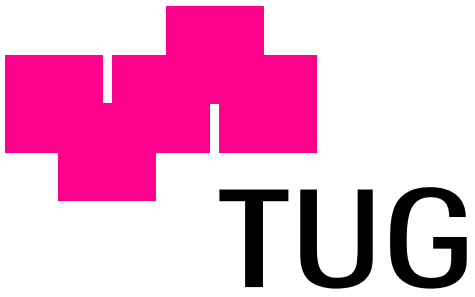
Christoph Thumser

Quality Control for Microarray Production

Master Thesis



Institute for Biomedical Engineering

Graz University of Technology

Krenngasse 37, A - 8010 Graz

Head of Institute:

Univ.-Prof. Dipl.-Ing. Dr.techn. Gert Pfurtscheller

Supervisors:

Dr.rer.nat. Marcel Scheideler

Dipl.-Ing. Gerhard Thallinger

Evaluator:

Univ.-Prof. Dipl.-Ing. Dr.techn. Zlatko Trajanoski

Graz, September 2003

Für meine Eltern

Abstract

Laboratory Information and Management Systems (LIMS) and databases for medical and biological use have been available for the last 20 years but the importance is increasing every day by better stability, scalability and usability. Although suitable software tools exist custom applications covering special needs have to be implemented. At the Institute of Biomedical Engineering the MARS (Microarray Analysis and Retrieval System) application was developed to manage all data related to microarray production and analysis.

A quality control component for MARS was developed based on a domain model in Unified Modelling Language (UML). This model was used to generate the business logic [Enterprise Java Beans (EJB)] and the web presentation with a generative programming framework. The generated code was extended to meet all requirements and provide the necessary usability. Java Server Pages (JSP) and additional business logic were implemented.

The compliance of Java Database Connectivity (JDBC) driver implementations with the JDBC specification was tested for Oracle, Microsoft SQL Server, MySQL and Cloudscape. Based on the test results a guideline for database development with Java was proposed.

Keywords: LIMS, microarray, MARS, MDA, AndroMDA, UML, JDBC, JDBCCTS

Kurzfassung

Laborinformations- und Managmentsysteme (LIMS) und Datenbanken für medizinische und biologische Anwendungen verwendet man schon seit 20 Jahren. Da diese Systeme immer robuster, skalierbarer und benutzerfreundlicher werden, wird die Bedeutung immer wichtiger. Obwohl entsprechende Software-Anwendungen in diesem Bereich vorhanden sind, sollten dafür angepasste Anwendungen mit speziellen Anforderungen realisiert werden. In der Arbeitsgruppe Bioinformatik vom Institut für Biomedizinische Technik entstand 2003 eine Datenbank namens MARS (Microarray Analysis and Retrieval System), die die Daten der Microarray Produktion erfasst und verarbeiten kann.

Es wurde eine Qualitätskontrolle für MARS basierend auf einem Domain-Modell mit Unified Modeling Language entwickelt. Aus dem Modell wurde die Steuerungslogik [Enterprise Java Beans (EJB)] und die Web-Präsentation mit Hilfe von "Generativen Software Enwicklungsmethoden" erzeugt. Der generierte Code wurde entsprechend den Anforderungen erweitert um ein funktionierendes System zu erstellen. Dafür wurden Java Server Pages (JSP) und weitere Steuerungslogik erstellt und eingebunden.

Um einen Leitfaden für die Erstellung von Datenbanken mit Java zu Verfügung zu stellen, wurde die Java Database Connectivity (JDBC) auf Verträglichkeit von unterschiedlichen JDBC-Treibern beim Zugriff auf die Datenbanken Oracle, Microsoft SQL Server, MySQL und Cloudscape getestet.

Schlüsselwörter: LIMS, Microarray, MARS, MDA, AndroMDA, UML, JDBC, JDBCCTS

Contents

List of Figures

List of Tables

Glossary

API The Application Programmer Interface is a formalised set of software calls and routines that can be referenced by an application program.

cDNA The Complementary DNA is a form of DNA prepared in the laboratory using messenger RNA (mRNA) as template, i.e. the reverse of the usual process of transcription in cells; the synthesis is catalysed by reverse transcriptase. cDNA thus has a base sequence that is complementary to that of the mRNA template; unlike genomic DNA, it contains no non-coding sequences (introns).

DBMS The DataBase Management System is a software system that facilitates the creation and maintenance of a database.

DDL The Data Definition Language is a subset of SQL allowing the creation and deletion of tables in a database.

DML Data Manipulation Language is a subset of SQL including the syntax for complex queries as well as for updates, insertions and deletions of data records.

DNA The DeoxyriboNucleic Acid is a long linear polymer, composed of four kinds of deoxyribose nucleotides that is the carrier of genetic information.

EJB The Enterprise Java Bean is a business data object specification developed in java technology.

ERP System The Enterprise Resource Planning System is a complete enterprise wide business solution that attempts to integrate all departments and functions in a company into a single computer system.

Gnumake is a build program of the Free Software Foundation Gnu which is a project to provide a freely distributable replacement for Unix.

IDE The Integrated Development Environment is an editor integrating all functions to manage, compile and build files or develop a software project.

JavaDoc Javadoc is a tool that parses the declarations and documentation comments in a set of source files and produces a set of HTML pages describing the classes, inner classes, interfaces, constructors, methods and fields.

JDBC The Java Database Connectivity is a Java API and specification for accessing databases.

JMS The Java Message Service is an API for accessing enterprise messaging systems from Java programs. Java Message Service, part of the J2EE suite, provides standard APIs that Java developers can use to access the common features of enterprise message systems.

MARS The Microarray Analysis and Retrieval System is an application to provide data integration for microarray experiments.

MIAME The Minimum Information About a MicroArray Experiment is a standard describing the minimum information of a microarray experiment. MIAME outlines the content and not the annotation format for an array experiment.

Middleware connects two or more network services into an integrated platform. Such connection should allow to exchange information in ways that are feasible to the user.

ODBC The Open DataBase Connectivity is a Microsoft API for accessing databases.

PCR The Polymerase Chain Reaction is a technique for selectively replicating a given stretch of DNA in vitro to produce a large amount of a part of the gene. Starting material is genomic DNA which renders subcloning unnecessary.

PIM The Platform Independent Model is a view of a system from the platform's independent viewpoint. A PIM exhibits a specified degree of a platform independence compatible with a number of different platforms of similar type.

PSM The Platform Specific Model is a view of a system from the platform specific viewpoint. A PSM combines the specifications in the PIM with details that determine how the system uses a particular type of a platform.

QC Quality Control is a management function whereby quality of raw materials, its assemblies, final production and its components, as well as human resources related to production and management are ascertained. This procedure should prevent production of defective material or the rendering of faulty services.

RNA The RiboNucleic Acid is a linear single stranded polymer composed of ribose nucleotides that is synthesised by transcription of DNA or by copying of RNA.

SQL The Structured Query Language is a non procedural language for accessing relational databases.

SQL92 is a specific version of SQL adopted as an ISO standard in the year 1992

XML The eXtensible Markup Language is an open standard of the World Wide Web Consortium (W3C) designed as a data format for structured document interchange.

Chapter 1

Introduction

Although Laboratory Information and Management Systems (LIMS) or databases for medical and microbiological use have been available for the last 20 years, they become more important every day. In many instances laboratory technicians are still working with calculator, paper and spreadsheet analysis. This may lead to mistakes in acquired data and problems of loosing data of valuable experiments. Although excellent software tools are available custom made programs or web application tools are needed to cover special needs. First of all the number of web applications are increasing and are implemented in many variations. The main advantage of this technology is that the user is not forced to install a stand-alone application. A user only needs a web browser to access his application and consequently is independent of the operating system.

An important component of such an application is the quality management to assure consistent quality and to avoid and detect errors caused by human and technical failures.

1.1 Laboratory Information Management Systems (LIMS)

A LIMS should be a system in which all users can store and retrieve data related to the laboratory work. It can be separated into a management system and an information system. The main aspects of a management system are:

- Maintenance, ordering and management of laboratory items such as ingredients and laboratory equipment
- Financial budgeting and accounting
- Automatic documentation
- Data backup
- Scheduling of human and equipment resources
- Quality management
- User management

The information system represents following purposes:

- Data and documents should be instantly accessible
- Provide a data analysis
- Graphical representation of data
- Experiment modelling and documentation

A LIMS should provide an organised system for all laboratory workers to attain a more efficient workflow and increased productivity. A LIMS should also be extendable for any system and equipment which could be integrated in the laboratory. Therefore it should support a wide range of international standards [6] in all interfaces like:

- DICOM Digital Imaging and Communications in Medicine [7]
- HL7 Health Level Seven [12]

A LIMS should contain various interfaces to different laboratory equipment like barcode scanners, barcode printers and photometers. Further it should have interfaces to Enterprise Resource Planning (ERP) systems like SAP (Systems, Application, Products)[28] which are installed in many companies and universities.

In general a complete LIMS is very extensive and consequently will take about one year to integrate it depending on the size of the laboratory and the number of developers creating the custom made system. The main concern should rely on the usability which should be as easy as possible. More about the implantation and documentation of LIMS can be found on the web page of LIMSource.[22]

1.1.1 Quality Management in a LIMS

Quality management is an integrative component in a LIMS which should not only manage but also maintain the quality of all laboratory work and equipment. The most important quality management standards are described below:

- Total Quality Management (TQM): Total Quality Management refers to a management process and set of disciplines that are coordinated to ensure that an organisation consistently meets and exceeds customer requirements. TQM engages all divisions, departments and levels of an organisation. It provides systematic management of data of all procedures to eliminate waste, pursue continuous improvement, prevent problems and errors in a long term orientation.[30]

- ISO 9000 is a family of quality management standards from the International Organization for Standardization (ISO) and can be split up in:

  - ISO 9001: These are representing the requirements all the way from design and development to production, installation and servicing.
  - ISO 9002: These are the requirements for the production, installation and servicing only.
  - ISO 9003: Basically uses inspection and testing to ensure that final products and services meet specified requirements.

  The details of the revised version of ISO 9000 called ISO 9000:2000 can be found on the web page. [15]

- Good Laboratory Practice (GLP): The requirements of the GLP [11] have been defined to protect the environment and generate good quality and validity of data from any laboratory experiment. The GLP provides a guideline to plan, perform, monitor, record, archive and report such experiments. Further it should avoid duplicate experiments, protect the environment and the animal welfare and minimise costs for governments and the industry.

- 21 CFR Part 11[1]: The 21. Code of Federal Regulations Part 11 is a regimentation of the Food and Drug Administration (FDA)[9] of the United States of America for computer-systems to handle electronic documents and signatures. The documentation is facilitated for the biological and pharmaceutical industry because all experiments had to be documented on paper before. Many LIMS providers claim to support the 21 CFR Part 11 requirements.

1.2 Microarray Production

DNA (DeoxyriboNucleic Acid) arrays (microarrays) have become the preferred method for large-scale gene expression measurement. The obtained expression ratios levels in a number of different tissues and different experimental conditions provide a first step towards functional characterisation of new entities revealed by DNA sequencing. Although the basic principle of measurement is based on hybridisation of a mixed probe derived from tissue RNA (RiboNucleic Acid) to large sets of DNA fragments representing many genes, a number of different forms of implementation of this principle are at hand. They are briefly described and compared, emphasizing sensitivity and sample requirements as well as accessibility of the methods to academic scientists.[41]

Microarrays are glass slides which are coated to provide a hydrophobic and positively charged surface for the spotting process. Microarrays can be grouped into oligo arrays and cDNA arrays depending on the type of probe used for microarray production. Since the QC application covers only cDNA microarrays oligo arrays are not described further. The cDNA microarray production can be classified in a cDNA library production and a cDNA amplifica-

tion part. The first step of creating a cDNA library is to isolate RNA from the cells, whereby most isolation methods are already available in kit format. The RNA is then reverse transcribed into cDNA (complementary DNA). The cloning involves ligation of the vector and DNA insert catalysed by ligase and introducing the joined recombinant plasmid into a suitable bacterial host (transformation) for stable storage and further multiplication. The following steps include multiplication with bacteria and the amplification of DNA by the PCR (Polymerase Chain Reaction).[33] The PCR products can be verified by gel electrophoresis, if necessary purified and analysed again. If the PCR products are conforming the quality standards they will be concentrated and pipetted in different plates used for the spotting process.

1.2.1 Gel Electrophoresis of PCR Amplification

The Electrophoretic methods [40] are separating the molecules depending of their size. Electrical charges are carrying molecules through an electrical field whereby the molecules (DNA fragments) are negatively charged. Due to the negative charge of the DNA provided by phosphate groups the molecules are moving towards the positive electrode. In biological laboratories

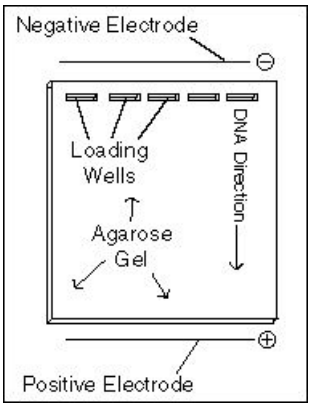


Figure 1.1: Principle of gel electrophoresis [10]. The molecules are moving in a agarose gel from the negative to the positive electrode due to their negative charge.

electrophoresis is used to separate and analyse molecules on a polymer matrix ranges in pore size from 1 to 1000 nanometers. The gel provides a certain resistance to the molecules moving

in the electric field. The following criteria influence the distance of separation:

- the distance between the particles
- the electric field strength
- the size of the molecules
- the conformation of the molecules

Usually two forms of gel are used:

- Agarose gel is purified Agar which is a natural long-chain polysaccharide produced from seaweed. The agarose gel is produced by heating agarose powder in a buffer for gel-electrophoresis, pouring it in a gel chamber and cooling it down.
- Polyacrylamide gel consists of the small synthetic acrylamide molecules and polymerizes to long chains by the action of a catalyst. The pores are smaller than in the agarose gel and is therefore used for the separation of proteins or also smaller nucleic acids.

After the DNA fragments have run through the gel an analysis is performed. Many different labelling methods are available, but due to the specific application only the bond of fluorescence dyes is described. Most of the laboratories use flourescent labelling with ethidium bromide which intercalates between the bases of DNA/RNA so that these molecules can be detected by ultraviolet radiation. The molecules absorb the ultraviolet radiation (302nm) and emit the radiation in the visible range (560nm) which can be further photographed. In general these pictures will be taken with a digital camera, saved and accordingly managed by a LIMS.

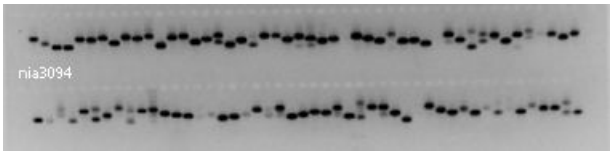


Figure 1.2: Example of a gel image with two rows of 48 lanes

1.2.2 Microarray Database

The microarray database should manage all the data obtained from the workflow in the laboratory. Here is a sample microarray workflow:

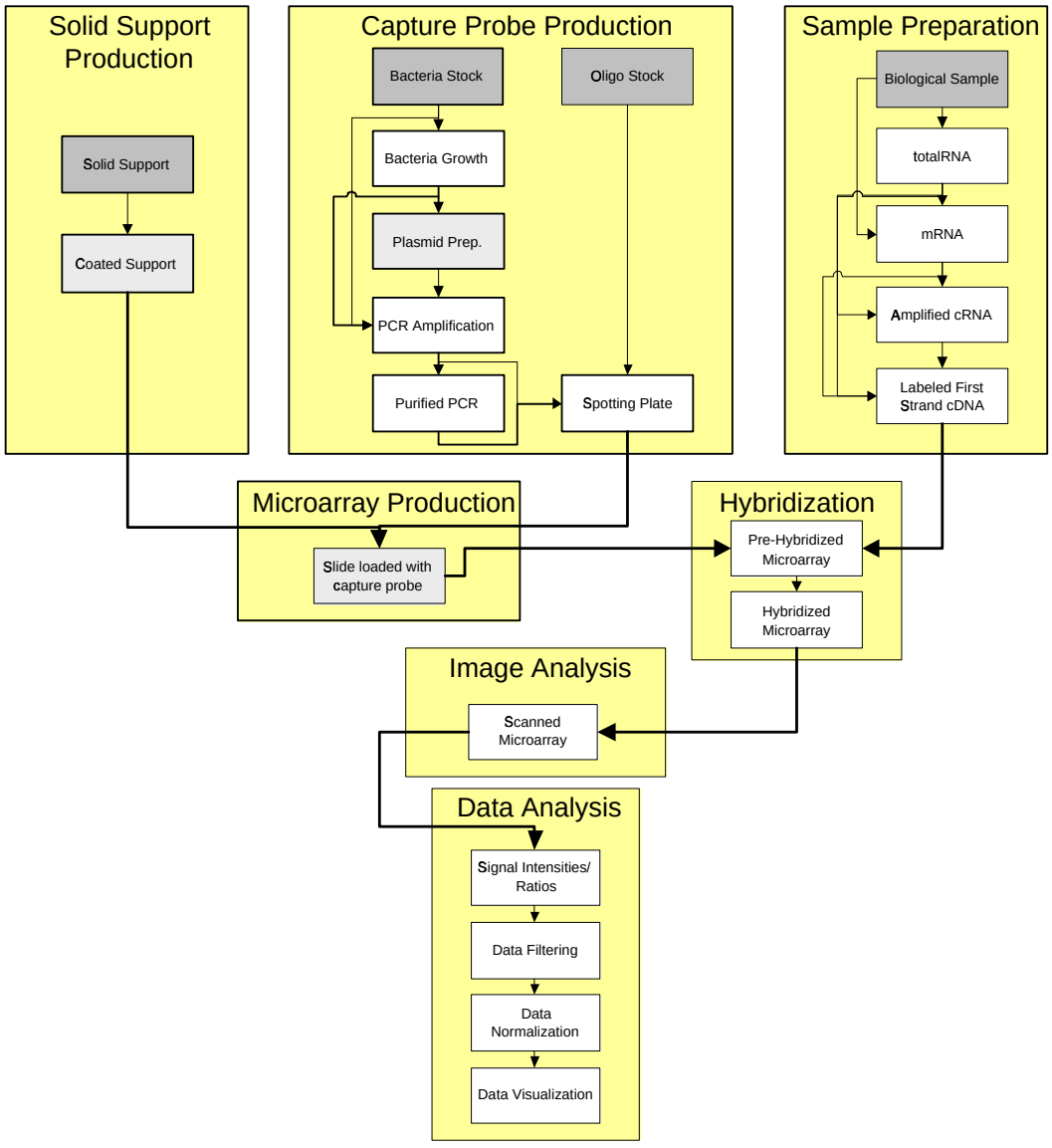


Figure 1.3: The workflow of the microarray production, hybridisation and analysis

This workflow is realised with the database MARS (Microarray Analysis and Retrieval System). It supports the MIAME (Minimum Information About a Microarray Experiment) stan-

dard which is a nomenclature to describe all steps of microarray production. The MARS is a 3-tier web application constructed with EJB (Enterprise Java Beans)(see subsection 3.5.1) and the Jakarta Struts Framework (see subsection 3.6.2) whereby the data is stored in an Oracle database. Since it is a web-based system, only a web browser and an user account is necessary to use it. To completely support the microarray workflow, analysis tools and quality control components have to be added to MARS.

Chapter 2

Objectives

This thesis contains two objectives. One is the integration of a QC component MARS; the other one is to investigate JDBC driver compatibility.

2.1 Quality Control for Microarray Production

The aim was to develop a quality control (QC) application for the MARS. The application should be a stand-alone component to be accessed by MARS. The requirements are as follows:

- preparing a flexible configuration environment for different gel images, gel chambers and pipettes
- retrieving the gel image and PCR plate files from MARS
- providing an easy evaluation handling for the samples
- calculating an evaluation matrix from the evaluated wells
- generating standard plates

For this purpose a framework [42] should be used which supports easy application modelling with EJBs (see subsection 3.5.1) and the Struts Framework (see subsection 3.6.2). This QC application should be integrated into the MARS (see subsection 1.2.2) without altering the

MARS. The QC application should be a stand-alone web application in the application server which should be accessed by the MARS. First a model should be created of the QC application with a UML/CASE tool. A framework provided from Thomas Truskaller [42] which is based on AndroMDA (see subsection 3.2.1) is used to create the EJBs and the appropriate Java Server Pages (JSP) (see subsection 3.6.1) from the UML model. Further business logic and a web application should be developed where PCR QC entities can be created, modified and saved in a web browser and the quality of the gel images can be evaluated.

2.2 JDBC Driver Compatibility Study

The aim of this work was the investigation of the implementation differences of various Java Database Connectivity (JDBC) drivers (see subsection 3.7.1).

JDBC drivers provided from different DBMS vendors should be tested for their conformity to the SUN Microsystems JDBC API specification. [19] A guideline for the database access implementation should be provided. For this purpose the JDBC Compatibility Test Suite (JDBCCTS)(see subsection 3.7.3) from SUN Microsystems should be used.
The guidlines should help to implement applications, which do not depend on the features of a specific DBMS. In the future it should be possible to change a database engine without worrying about incompatibilities of the different driver implementations.

Chapter 3

Methods

In this chapter different programming tools are being discussed that are used to develop the QC for MARS (see subsection 1.2.2). Furthermore different database access methods are described with special emphasis on the database access with Java.

3.1 Application Development using Models

Enterprise applications must be structured in a way that enables scalability, security and robust execution and their architecture has to be documented thoroughly. These programs must be designed to work perfectly in many areas. Design time is the best occasion to structure an application as a collection of self-contained modules or components. One of the benefits of this structure is that it enables code reuse.

Modelling is the design of software applications before coding. A model plays the analogue role in software development which blueprints and other plans (site maps, elevations and physical models) play in the building of a skyscraper. Modelling is the only way to visualise the design and check it against requirements before the application will be coded.

The Object Management Group (OMG)[26] is an open membership nonprofit consortium that produces and maintains computer industry specifications for interoperable enterprise applications. One of the most important specifications of OMG is the multi-platform Model Driven Architecture (MDA)[24] which is based on following modelling specifications:

- MOF (Meta-Object Facility)
- UML (Unified Modeling Language) (see subsection 3.1.1)
- XMI (XML Metadata Interchange) (see subsection 3.1.2)
- CWM (Common Warehouse Metamodel)

3.1.1 Unified Modelling Language (UML)

The UML helps to specify, visualise and document models of software systems including their structure and design in a way that meets all of their requirements.
It is possible to model any type of application running on any type and combination of hardware, operating system, programming language and network. Its flexibility allows modelling of distributed applications that use any middleware on the market. Built upon the MOF metamodel, which defines class and operation as fundamental concepts, it is a natural fit for object-oriented languages and environments such as C++, Java, and the recent C#. But UML can be used also to model non-OO (object-oriented) applications as well.
Some tools also analyse the source code and reverse engineer it into a set of UML diagrams. Others execute, test and verify UML models. Some of the most important and most common tools are listed below:

- Poseidon for UML from Gentleware[14]
- Rose from Rational Software (now IBM)
- Together from TogetherSoft (now Borland)

The webpage of javaskyline[38] provides a good overview of UML software.
With its joining techniques and middleware independence UML forms the foundation of OMG's Model Driven Architecture (MDA).
UML defines twelve types of diagrams divided into three categories: Four diagram types represent static application structure, five represent different aspects of dynamic behaviour and three represent ways to organise and manage the application modules.

- Structural Diagrams include the Class Diagram, Object Diagram, Component Diagram, and Deployment Diagram.
- Behaviour Diagrams include the Use Case Diagram (used by some methodologies during requirements gathering), Sequence Diagram, Activity Diagram, Collaboration Diagram, and State-chart Diagram.
- Model Management Diagrams include Packages, Subsystems and Models.

A UML model can be either platform-independent or platform-specific. The MDA development process uses both of these forms: Every MDA standard or application is based normatively on a Platform-Independent Model (PIM) which represents its business functionality and behaviour very precisely but does not include technical aspects. From the PIM, MDA-enabled development tools follow OMG-standardised mappings to produce one or more Platform-Specific Models (PSMs).
The PSM contains the same information as an implementation but in the form of a UML model instead of executable code. In the next step a code generator creates the executable code from the PSM along with other necessary files (including interface definition files if necessary, configuration files, make files and other file types). After giving the developer an opportunity to hand-tune the generated code a deployable final application can be build.

3.1.2 XMI (XML Metadata Interchange)

The main purpose of XMI is to enable easy interchange of metadata between modelling tools (based on the OMG-UML) and metadata repositories (OMG-MOF based) in distributed heterogeneous environments. XMI integrates three key industry standards:

- XML - eXtensible Markup Language, a W3C standard
- UML - Unified Modeling Language, an OMG modelling standard
- MOF - Meta-Object Facility, an OMG metamodelling and metadata repository standard

The integration of these three standards into XMI merges the best of OMG and W3C metadata. Modelling technologies allowing developers of distributed systems to share object models and other metadata over the Internet. XMI together with MOF and UML form the core of the OMG metadata repository architecture.

3.2 Generative Programming

Generative Programming is not a new technique because also a compiler generates a program binary from the program implementation in a high level language. Many wizards are implemented in different Integrated Developer Environments (IDEs)(Jbuilder, Visual Studio,...) which create the different linked applications automatically.
Model driven applications like CASE/UML tools use code generation to generate classes with plug-ins. The code generation has two main advantages:

- Complete software projects can be generated.
- The code can always be regenerated.

The code generation tool itself used to generate the source files for the QC application including some other tools is outlined below.

3.2.1 AndroMDA

AndroMDA [37] is an open source code generation tool creating application classes and deployable components based on a UML model, specific cartridges and custom made Velocity templates.[31] Velocity is a Java-based template engine. It permits anyone to use the simple yet powerful template language to reference objects defined in the Java code. AndroMDA is based on the MDA and therefore the UML model has to be in a XMI format.

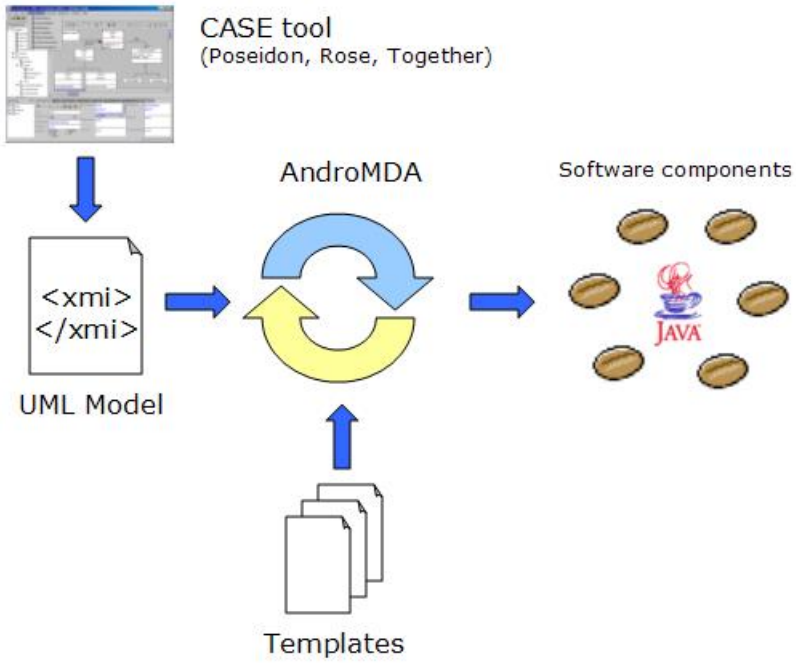


Figure 3.1: Generative Programming with AndroMDA.[37] AndroMDA builds from the UML model and adapted templates diverse software components like Java classes, EJBs and JSPs.

AndroMDA has a core module that allows to plug in "cartridges". A cartridge consists of a set of template files that determine what to generate.
AndroMDA provides a set of standard cartridges which have to be modified for a specific application. The following basic cartridges are available from AndroMDA:

- andromda-java – a cartridge that generates general Java source code.
- andromda-ejb – generates Enterprise Java Beans (see section 3.5.1).
- andromda-hibernate – generates persistent classes for the Hibernate Object/Relational mapper.
- andromda-struts – generates web pages, forms and action classes for use with Jakarta Struts (see subsection 3.6.2).

The cartridges are defined in `andromda-cartridge.xml` describing how all Velocity templates are generated. The Velocity Engine integrated in AndroMDA creates specific source files from the different Velocity template files.

3.2.2 XDoclet

The files generated by AndroMDA (see subsection 3.2.1) have to be transformed into deployable files for a specific applications using XDoclet.[32] XDoclet is another code generation engine which enables Attribute-Oriented Programming for Java. With special JavaDoc tags which are embedded in the Java sources XDoclet generates Java source code and XML deployment descriptors. These files can be deployed on the application server (see subsection 3.3).

3.2.3 Jakarta Ant

AndroMDA and XDoclet are integrated into the build environment provided by Ant which is a platform independent, Java-based build tool.
Ant consists of different items called Ant Tasks which are executing specific tasks. Besides the core Ant tasks for low level work like copying, making directories and zipping and optional tasks for more complex work it is also possible to define own tasks by extending the Ant `build.xml` file. The sequence of processing the Ant tasks can also be determined in the `build.xml` file. Both AndroMDA and XDoclet are tasks and subtasks in the Ant `build.xml` - file which are executed before Ant compiles the Java source files, builds the jar-, war- and ear-files and verifies all deployable components.

3.3 Application Server

An application server makes it possible for a server to generate a dynamic or customized response to a client's request. The middleware or application server must meet the requirements of the J2EE specification. An overview of the different application server implementations can be found in [4]. For the QC application as well as the MARS the open source application server

JBoss [18] is used. The generative programming framework integrates XDoclet-tags (see subsection 3.2.2) for the JBoss server although they could be prepared for any other application server too.

3.4 Web Applications

If one intends to build a usable web-based application not only the user interface is important. A web application has to be reliable, highly available, fault-tolerant and scalable. These needs are not different from the needs of any other enterprise application. To provide such an enterprise application a multitiered application model is used. The middleware is implemented as an application server, whereby the back-end is an enterprise information system (EIS)(i.e. a database).

Java 2 Enterprise Edition (J2EE) and Microsoft .NET are evolutions of the existing application server technology used to build such enterprise applications.
Microsoft .NET is a product suite that enables organisations to build smart enterprise-class web applications. It is important to take note of the difference between these two technologies: .NET is a product strategy whereas J2EE is a standard to which products are constructed. Only the J2EE web architecture which is used for the QC control application is discussed in detail.

3.5 Java 2 Enterprise Edition (J2EE)

To reduce costs and fast track application design and development J2EE provides a component-based approach to the design, development, assembly and the deployment of enterprise applications. The J2EE platform offers a multi-tiered distributed application model, reusable components, a unified security model, flexible transaction control and web applications support through integrated data interchange on Extensible Markup Language (XML)-based open standards and protocols.[16]
J2EE component technologies include Enterprise Java Beans (EJBs)(see subsection 3.5.1), servlets and Java Server Pages (JSP)(see subsection 3.6.1). J2EE applications include access to standard

network protocols, database systems and messaging systems. J2EE components are executed in a web or EJB container. A container is an environment that provides components with the low level services such as security, deployment and threading. EJBs contain the business logic for the application while web components such as servlets and JSPs contain the presentation logic for the applications.

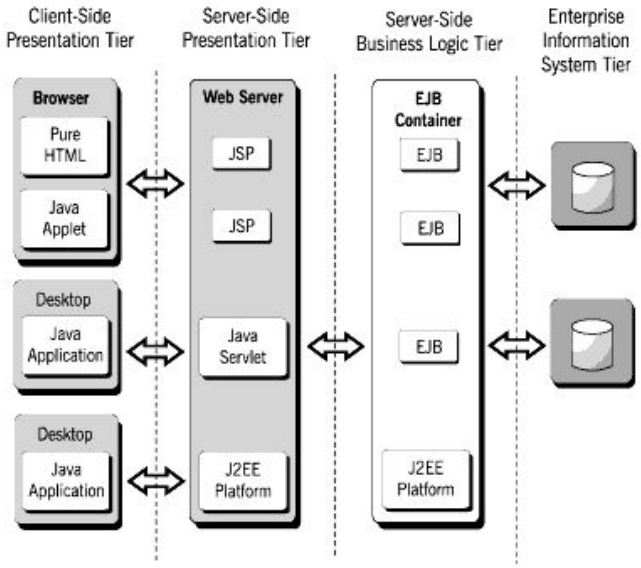


Figure 3.2: The tiered architecture of J2EE presents the client-side, the server-side(web container and EJB contanier) and the Enterprise Information System (i.e. database). [29]

Following items present the different tiers of J2EE [29]:

- Client-Side Presentation Tier: This tier contains end-user clients which access services provided by server-side components. Clients can include HTML pages and applets running in browsers, Java applications and other applications that can interoperate with Java.
- Server-Side Presentation Tier: This tier provides dynamic information from the server-side to clients. It contains components such as Java Server Pages (JSP) and Java servlets. These components run in a J2EE-conformant web container.
- Server-Side Business Logic Tier: This tier provides the business logic that drives the enterprise application. It provides services that JSPs and servlets can access to fulfill client

requests. Components in the business logic tier are EJBs and run in a J2EE-conformant EJB container.

- Enterprise Information System Tier: This tier contains resources such as databases and other information systems that EJB components access to implement business tasks.

3.5.1 Enterprise Java Beans (EJBs)

An Enterprise Java Bean [8] is a server-side component (EJB Container) that encapsulates the business logic of an application (the code that serves the purpose of the application). The EJB technology simplifies the development of large distributed applications:

- The EJB container handles services like transaction management and security authorisation for the EJB, so the Bean developer can focus on the business logic of the EJB.
- The client developer can concentrate on the presentation and therefore avoids coding business logic and access to databases in the client applications whereby the client application can also run on small devices.
- The existing beans can be reused on different J2EE compliant servers because they are portable components.

Three fundamental types of EJBs exist:

- Entity bean: An entity bean represents an item of persistent data stored in a back-end database and provides operations that allow clients to access and manipulate this data. For example an entity bean could represent a piece of data stored in a database. Multiple clients can access an entity bean concurrently – the container uses transactions to control access to the data. They can be divided into two subcategories:
  - Container-managed persistence (CMP): The EJB container automatically provides the database access, therefore it must be a relational database. The necessary persistent attributes of the entity bean must be entered in the descriptor file.

  - Bean-managed persistence (BMP): Here the database access must be executed by the bean itself, the bean developer is responsible to make the bean persistent. The advantage is that the database access can be optimised.

- Session bean: A session bean is a transient object that acts as a server-side agent for a client carrying out actions on the client's behalf. Session beans typically represent a client during a single session between the client and the server-side application logic. There are two types of session beans:

  - Stateful session beans: A stateful session bean holds information about the state of a communication session between the client and the application. A stateful session bean normally exists while a client is using it. Nevertheless the EJB server can inactivate it or make it invalid for example after a timeout period.

  - Stateless session beans: A stateless session bean keeps no state information between calls but supports multiple clients and can offer better scalability for applications for many clients.

- Message Driven Beans: It operates as JMS (Java Message Service) message listener and allows the J2EE applications to process messages asynchronously.

3.6 Web Development

For the presentation of the web pages the Server-Side Presentation Tier as well as the Client-Side Presentation Tier is responsible. In the Server-Side Presentation Tier the client's requested web pages are dynamically created from the Java Server Pages (JSP).

3.6.1 Java Server Pages

The Java Server Pages (JSP) technology facilitates the creation of web pages in a fast, platform-independent and dynamic way. JSPs contain the HTML tags as well as specific XML-related tags which account for the dynamic generated part. The JSP technology separates the user

interface from content generation, enabling designers to change the overall page layout without altering the underlying dynamic content.
The servlet engine compiles the JSPs to servlets which then are launched in the web container to perform their demanded tasks. Servlets are platform-independent 100% pure Java server-side modules that fit into a Web server framework and can be used to extend the capabilities of a Web server with minimal overhead, maintenance and support. Servlets invoke the applications of the application server or access a database. The Servlet engine sends back dynamically generated HTML pages which are displayed by the client's browser.

3.6.2 Jakarta Struts Framework

Struts [3] is an application development framework that is designed for and used with the J2EE platform (see section 3.5). The Struts framework shortens the development process and makes developers more flexible in the design of applications as it offers a series of tools and by providing components to build applications with. Struts is non-proprietary and works with virtually any J2EE compliant application server. It is a subproject of the Apache Software Foundation's Jakarta project.[2] By accelerated development process Struts minimises the costs of large software projects. Additionally it helps to improve the quality and the maintainability of the application. Struts is based on the Model-View-Controller (MVC) design pattern. The MVC pattern is known as being among the most well-developed and mature design patterns in use. By using the MVC design pattern processing is splitted into three distinct sections aptly named the Model, the View and the Controller. Those are described in the following listing:

- Model components: They provide a "model" of the business logic or data behind a Struts program. It is very common for model components to provide interfaces to databases or back-end systems. Model components are generally standard Java classes. There is no specific format required for a model component so the Java code written for other projects might be reused.

- View components: View components are those pieces of an application that present information to users and accept input. In Struts applications these correspond to Web pages.

They are used to display the information provided by model components. Usually there will be one or more view components for each Web page in a Struts application. They are generally built using Java Server Page (JSP) (see subsection 3.6.1) files. Struts provides a large number of "JSP Custom Tags" (sometimes referred to as Struts Tags) which extend the normal capabilities of JSP and simplify the development of view components.

- Controller components: They coordinate activities in the application. They are updating a database through a model component by inserting the user's data. Furthermore they are detecting an error condition with a back-end system and are directing the user through special error processing. Controller components accept data from the users, decide which model components need to be updated and then they decide which view component needs to be called to display the results.

  One of the major contributions of controller components is that they allow the developer to remove much of the error handling logic from the JSPs. (After all if errors in processing occur the controller component forwards them to an error processing view component but not the primary results view component.) This can significantly simplify the logic in the pages and makes them easier to be developed and maintained.

  Controller components in Struts are Java classes and must be built according to specific rules. They are usually referred to as "Action classes".

3.7 Database Access

The database is the heart of an enterprise system and it is used to store the data of shared business objects that make up an enterprise system. Database architectures can be classified in:

- Relational database: Relational Databases make it possible to relate data of two dimensional tables with primary keys and foreign keys. The object-relational mapping has to be done by the developer.

- Object-oriented database: This does not attempt to separate object data from the behaviour and can be seen as a permanent storage of objects. The object-oriented encapsu-

lation of data however makes it difficult to relate data as done in a relational database.

- Object-Relation Database: This combines a "best of both world" advantage by providing both object and relational means of accessing data.

The overwhelming majority of today's databases application uses relational databases. The most important options to access relational databases are discussed below.
SQL (Structured Query Language) is an ANSI (American National Standards Institute) standard language for accessing and manipulating database systems. SQL statements are used to retrieve and update data in a database. There are two possibilities in accomplishing data manipulations with SQL: firstly commands which return demanded datasets defined in the data manipulation language (DML) and secondly commands which alter the database's internal structures using the data definition language (DDL). Although SQL should be vendor independent all database vendors proprietarily extended SQL to enhance the performance of their own product. Consequently the specific SQL syntax cannot be used for different DBMS.

To simplify Java database access, SUN Microsystems developed the JDBC (Java Database Connectivity) API (see section 3.7.1) in 1997. It should provide an easier access to different databases and enable the developer to replace the underlying DBMS without changes in the application. A Java programmer should not worry about varying SQL commands anymore.
What JDBC is to the Java world ODBC (Open Database Connectivity) is for the Windows world. It is a client-side cross-platform data communication standard developed by Microsoft and is used predominantly on the Windows platform.
The Enterprise JavaBeans Query Language (EJB QL) allows database access via the finder and select methods of an entity bean with container-managed persistence.

3.7.1 Java Database Connectivity (JDBC)

A buzzword in the today's programmer world is platform independence. Java, which has been designed having platform independence in mind, provides an API to access databases: JDBC (Java Database Connectivity)[17] which is based on the structured query language (SQL). It is called an SQL-level API because JDBC allows to construct SQL statements and embed them

into Java API calls. JDBC is the glue between databases and the Java world. The results from the database query for instance are returned as Java objects and access problems are thrown as exceptions.[35]
The three main aspects for SUN Microsystems to develop JDBC were:

- JDBC should be an SQL-level API
- JDBC should capitalise on the experience of existing database API's
- JDBC should be simple

Even if the JDBC API meets these requirements the DBMS vendors are responsible for the implementation of JDBC. Therefore they are constructing JDBC drivers mapping functions and datatypes to the JDBC specification.[19] Alternative to the JDBC drivers provided by the database vendors a JDBC driver can also be purchased from many software specialised companies providing their drivers on the internet.[21]

3.7.1.1 JDBC Driver Types

The JDBC drivers are classified according to their type:

- Type 1: These drivers use a bridging technology to access a database. The JDBC-ODBC bridge is a good example for this kind of drivers. It provides a gateway to the ODBC API. Implementations of this API in turn do the actual database access. Bridge solutions generally require software to be installed on the client systems.
- Type 2: Type 2 drivers are native API drivers indicating that these means the driver calls native C or C++ methods provided by the individual database vendors. These solutions also require software to be installed on the client systems.
- Type 3: The JDBC driver on the client uses sockets to call middleware application on the server which in turn translates the client requests into an API specific to the desired driver. This is extremely flexible since a single driver can actually provide access to multiple databases.

- Type 4: Using network protocols built into the database engine type 4 drivers talk directly to databases using Java sockets. Since they are implemented in pure Java, they are portable across all platforms. This type of driver will almost always be provided from the database vendor because these proprietary network protocols are almost never documented.

3.7.1.2 Connecting to a Database

Basically two parameters have to be known to connect from Java to a database via JDBC:

- The class name of the JDBC driver
- The JDBC URL to access the database

For example:

```
Class.forName("oracle.jdbc.OracleDriver()");
Connection connection = DriverManager.getConnection
("jdbc:oracle:thin:@10.2.0.6:1521:taranis","cts1","cts1");
connection.close();
```

3.7.2 EJB Query Language (EJB QL)

The Enterprise Java Beans Query Language (EJB QL) is a subset of SQL92 that specifies queries for finder and select methods of an entity bean with CMP (see subsection 3.5.1). EJB QL has extensions that allow navigation over the relationships defined in an entity bean's abstract scheme. The scope of an EJB QL query spans the abstract schemes of related entity beans that are packaged in the same EJB JAR file.
EJB QL queries have to be scripted in the deployment descriptor of the entity bean which are translated by the J2EE container at runtime into the specific JDBC commands. The developer has to configure the specific JDBC driver in a XML file describing the datasource. The entity beans are consequently portable and not tied to a specific database.

3.7.3 JDBC Compatibility Test Suite (JDBC CTS)

The JDBC Test Suite 1.3.1[20] was prepared by SUN Microsystems in 2002 to provide a tool for JDBC driver vendors to test their drivers. If they meet the specification of the JDBC API they can get the compliance certificate from SUN that their driver is JDBC compliant.

The test suite generates a text file where all results are presented. The JDBC Test Suite executes the same JDBC commands in different contexts as listed below:

- Enterprise Java Bean (EJB)(see subsection 3.5.1)
- Servlet (see subsection 3.6.1)
- Java Server Page (JSP)(see subsection 3.6.1)
- Application client

For the first three test modes an J2EE compliant application server is required to run the test. The main differences are that the application clients run on a client platform whereas EJB and Web components (JSPs, Servlets) run on a server platform. The three types of J2EE modules (see section 3.5) are:

- Enterprise JavaBeans components contain class files for enterprise beans and an EJB deployment descriptor. EJB modules are packaged as JAR files with a `.jar` (Java Archive) extension.
- Web components contain JSP files, class files for servlets, GIF and HTML files and a Web deployment descriptor. Web modules are packaged as JAR files with a `.war` (Web Archive) extension.
- Application client modules access the database directly. They contain class files and an application client deployment descriptor. Application client modules are packaged as JAR files with a `.jar` extension

Chapter 4

Results

This chapter will present the domain model and the database tables for the implemented QC component. Further the business logic and the web presentation of the QC entities will be described. Finally the results of the JDBC Driver Compatibility Study are presented.

4.1 Database Model

In a first briefing with natural scientists the exact requirements were discussed and some implementation proposals where outlined.

The first step was the creation of project files using the project wizard and the altered `wizard.properties`-file from the framework. Afterwards the data model was designed using the CASE/UML tool called "Poseidon for UML 1.6 Community Edition"[14], which is available free of charge.

Basically the used model components can be listed by their stereotype:

- Services will be converted to Session Beans.
- Entitys will be converted to Entity Beans.
- SessionBeanUtils will retrieve home interfaces of Session Beans in another application.
- Exceptions will be converted to Java exceptions.

- StandardStrutsActions will be implemented as Struts actions.
- ValueObjects will be realised as Value Object classes.

The following listing should represent the approach of the model design:

- First the PCR QC entities had to be grouped to small units and assigned to management entities and real data entities to get the best flexible solution for the implementation.
- A service was created to maintain, modify and retrieve data for entities and action classes of the web application.
- The entities had to be linked according to their relations in the database.
- The service as well as the entities had to get their appropriate stereotype which is most important for AndroMDA (see subsection 3.2.1) to transform the XMI file using the templates to specific source code.
- All the entities had to get their attributes representing the columns of the table in the database.
- The attributes had to be set to their specific datatypes for Java and the database.
- The tagged values were assigned to their corresponding operation defined in the cartridges of AndroMDA, whereby only the `andromda.service.standardoperations` were used. Further it is possible to define other tagged values like the table name for the database with the tag `@ejb.persistence`.

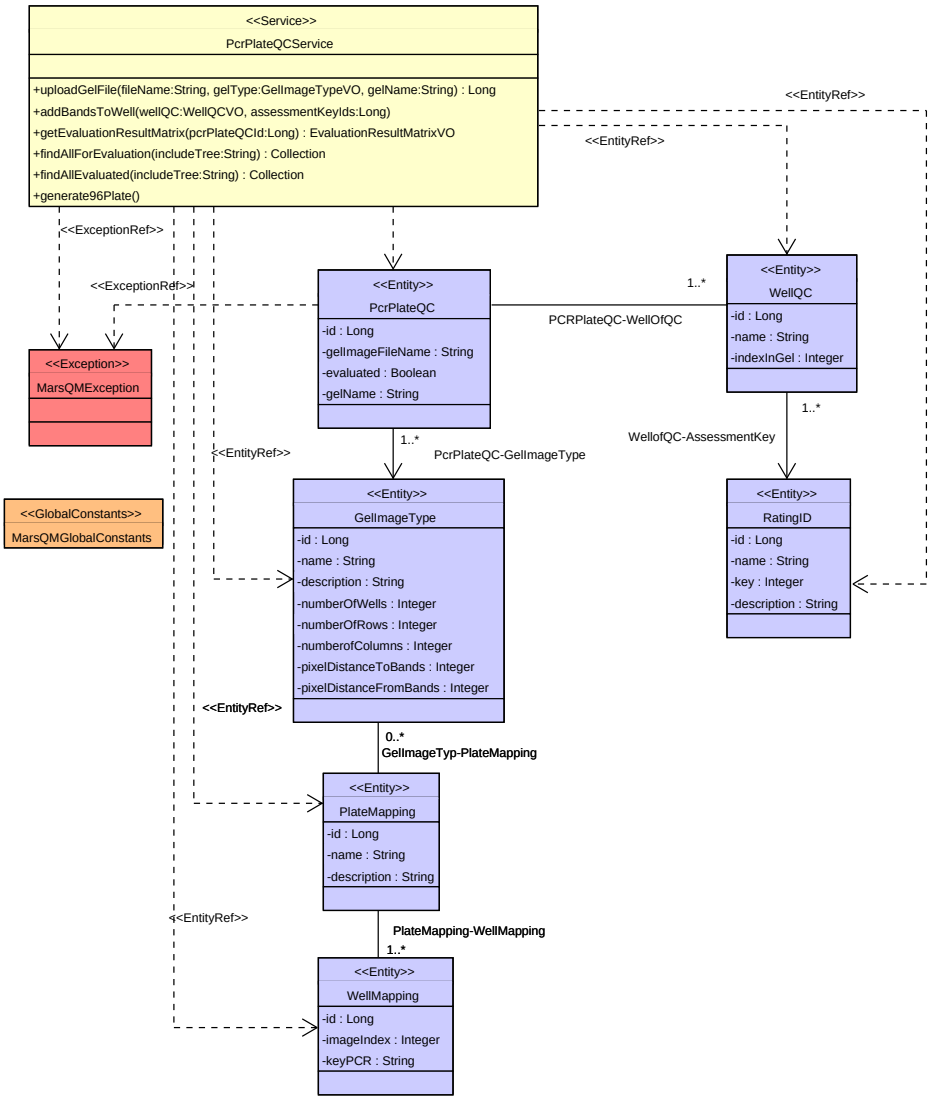


Figure 4.1: The Domain Model of MarsQM, consisting of services, entities and exceptions

To complete the UML model, action classes, value objects and utility classes have to be added.

For every entity a corresponding value object has to be defined.

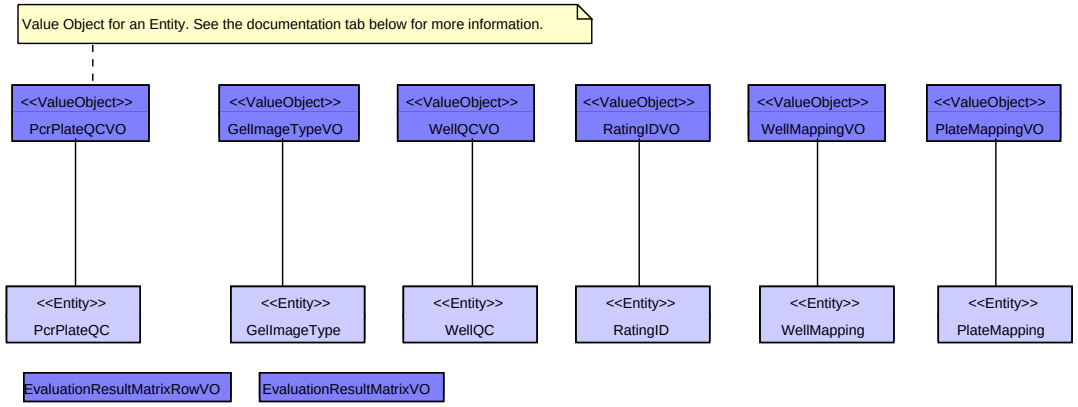


Figure 4.2: The Value Objects Model of MarsQM links the entities to the Value Objects.

Session Bean Utils were implemented to receive the home interfaces of specific Session Beans from the MARS:

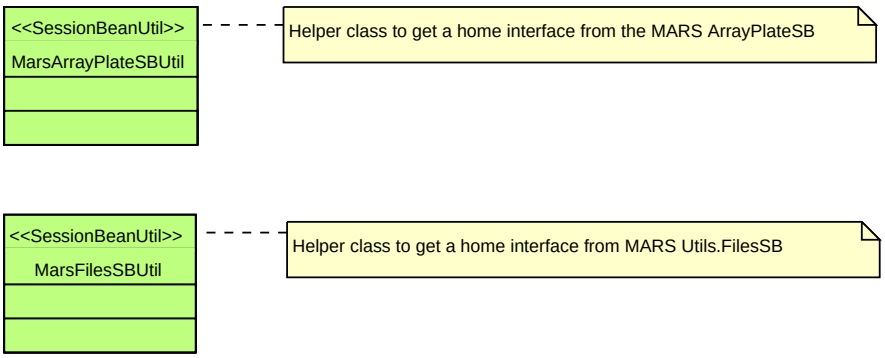


Figure 4.3: The Session Bean Utils classes of MarsQM

Action components are linked to the TemplateServiceFactory:

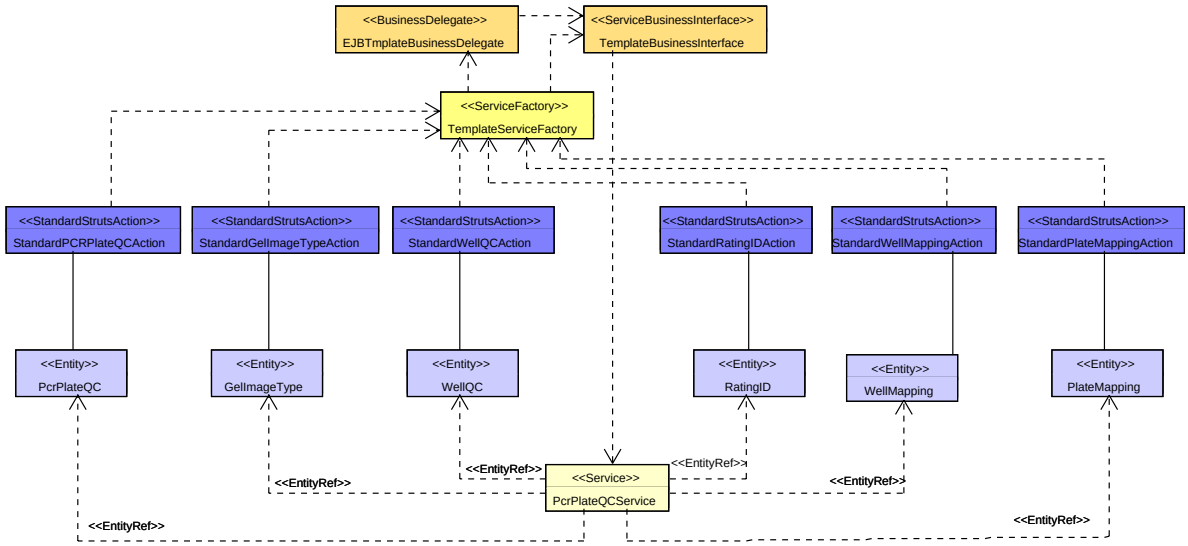


Figure 4.4: The Action Model of MarsQM is including the links from the entities to actions. Actions are linked to the ServiceFactory and this further to the BusinessDelegate and accordingly to the ServiceBusinessInterface

The ServiceFactory entirely separates the Web application from the data layer and enables the instanciation of different BusinessDelegate classes. These classes in turn have to implement a ServiceBusinessInterface which has to provide the required business methods.

4.1.1 Quality Control Management Entities

The management tables have been created in order to support different plate layouts, pipetting instructions and gel image layouts in the QC component. Four different entities describe the adjustments:

- well-mapping: This entity determines mapping of a well to the gel image which allows the use of different pipetting schemes, microtiter plates and gel layouts.
- plate-mapping: The plate-mapping defines the mapping of a complete plate to the gel chamber; each well of the plate is represented by a well-mapping

- gel-image-type: The gel-image-type specifies the plate-mapping, the size of gel image in pixels, offset of the first and last band to the image border, the number of wells, the number of rows and number of columns.
- rating ID: With this entity it is possible to assess a PCR product displayed in the specific lane of the gel image.

4.1.2 Quality Control PCR plate specific entities

The PCR plate related entites consists of two items::

- PCR-QC-plate: This entity defines the name of the PCR-QC-plate in the QC application. Furthermore the gel-image-type and the relation to the gel image is specified.
- well: The well defines the well-mapping, the relation to the PCR-QC-plate and the rating ID.

4.1.3 QC Service

The service is generated as a session bean (see subsection 3.5.1) and provides altering the existing application with additional business logic.
In the implementation class of the session bean all following methods were inserted:

- creating a PCR-QC-plate with the gel image and the gel-image-type
- adding ratingID to wells of the QC
- creating the evaluation matrix
- finder method to get all evaluated PCR-QC-plates
- finder method to get all PCR-QC-plates for the evaluation
- generating the standard PCR-QC management entity with 96 wells

4.2 Code Generation and Additional Modifications

The UML model is saved in XMI format and the build tool Ant prepared with the necessary Ant tasks (see subsection 3.2.3) is executed. The first Ant task AndroMDA (see subsection 3.2.1) generates all classes and files defined in the XMI format (EJB-, ValueObject-, SessionBean-Util- classes and accordingly the Standard-action-classes and Standard-JSPs). The second Ant task XDoclet (see subsection 3.2.2) generates the deployable files for the application server. Finally the classes are compiled, zipped, deployed and verified.

The project generated by the framework was loaded into the JBuilder IDE, where the further modifications were implemented. These methods are defining the necessary business logic for the application to simplify the handling of the MarsQM project.

4.3 Web Interface

For the web application different JSPs, action-classes and configuration files were designed with the Struts framework. The whole application was deployed on the application server JBoss.[18] The standard-menu generated by the framework contains an "Add New" and "Find All" button for every entity.

Here are described a few steps to create a PCR-QC-plate:
First all QC management entities have to be configured as like well-mapping, plate-mapping and gel-image-type. The menu has additional items like "Generate standard plate" to add a standard PCR-QC-plate. The submenu offers the possibility to create a 96 well plate by clicking on "Generate 96 plate". Now the plate definition is generated automatically with the preferred parameters for well-mappings, the plate-mapping and the gel-image-type.

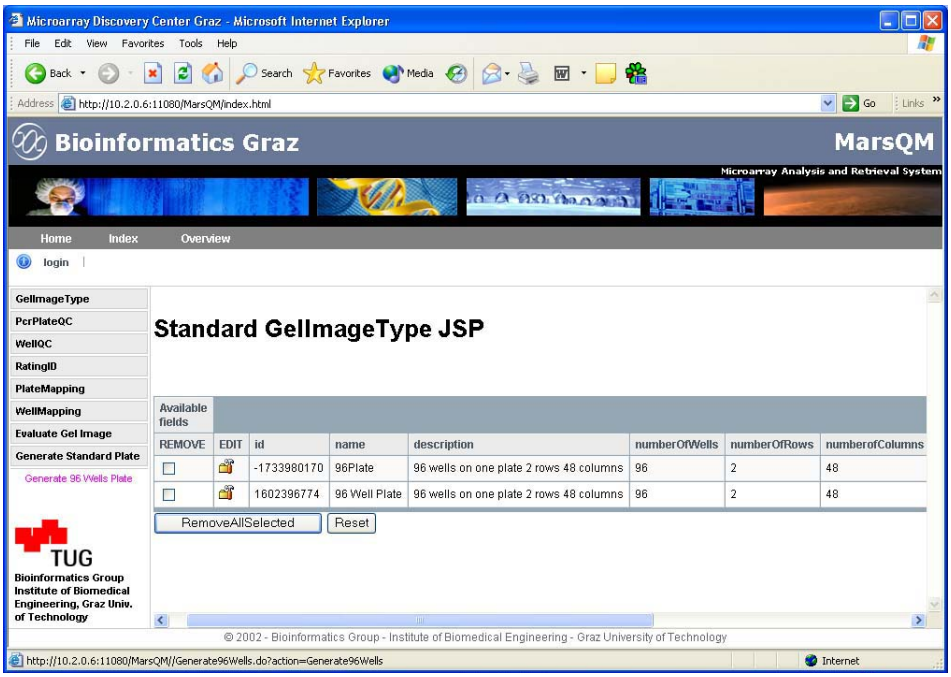


Figure 4.5: The "Generate Standard 96 Plate" web page

Additionally the ratingIDs had to be defined as well.

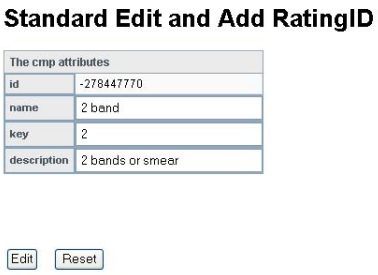


Figure 4.6: Adding an RatingID

By then a generation of PCR-QC-plates was possible. Clicking in the MARS application on the QC button of a specific PCR plate links to the web page of the MarsQM application for generating a PCR-QC-Plate.

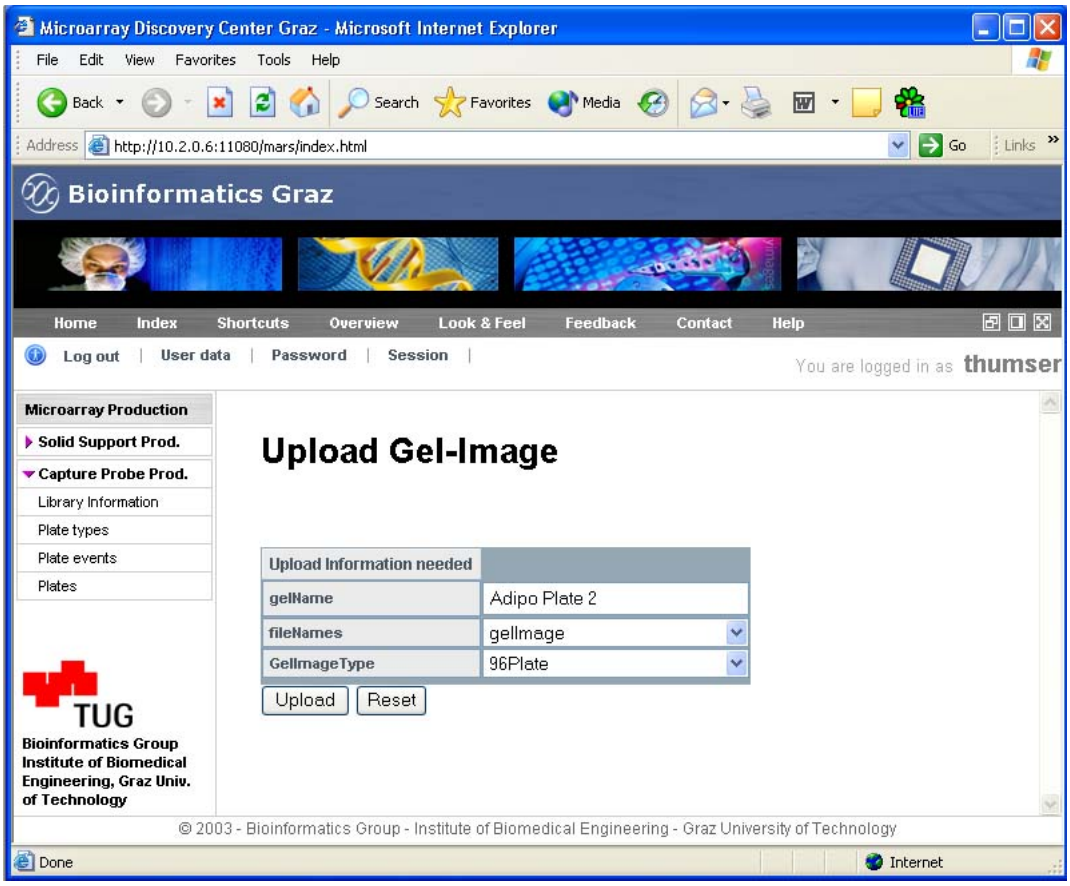


Figure 4.7: Generating the PCR-QC-plate

If the PCR-QC-plate already exists the MarsQM application links directly to the Evaluation web page which displays a description of the ratingIDs, the gel image and the evaluation matrix. The evaluation matrix shows the number of lanes which have one, two or no bands. The percentage of these are calculated in the special case of 96 wells.

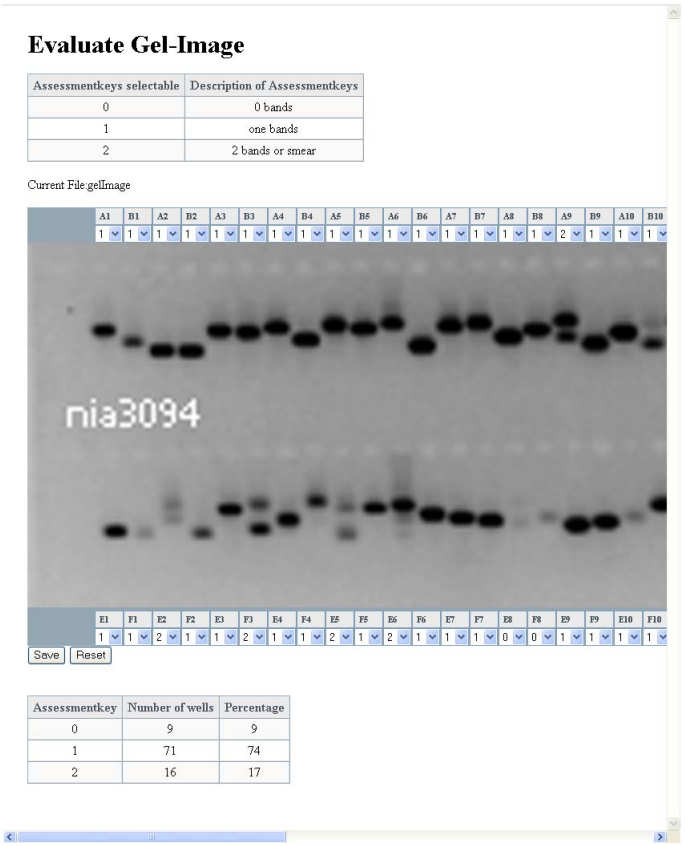


Figure 4.8: The evaluated PCR-QC-plate

4.4 JDBC Driver Compatibility Study

The first topic in this section is the classification of the tests from the test suite followed by a guideline to avoid problems when changing a DBMS and a chart of the main differences between the datatypes. Furthermore there are sections describing the differences of the tested databases. (see section B). Detailed results from the JDBC driver tests are available in an Excel sheet.

4.4.1 Setup of the JDBC CTS

The JDBC Test Suite is platform-independent although additionally Gnumake has to be installed which is fundamental to build all files. Therefore it would be better to run the testsuite on a UNIX system where Gnumake is already installed in the most cases. After the Test Suite is set up[20] the test suite has to be configured, by adapting the environment variables. The correct JDBC driver for the specific DBMS (see section A) has to be chosen. After setup of the database scheme with the file "initdb" the application server can be started. When the database and the application server are started the Test Suite is ready for execution.

4.4.2 Classification of the Tests

The test groups of the JDBC CTS are classified according to the JDBC specification[17]:

- DatabaseMetadata
- ResultSetMetadata
- ResultSet
- Batchupdates
- Prepared Statement
- Scalar Functions
- Time-Date-Functions
- Batch-Update-Exceptions
- SQL-Exceptions
- Statements
- Connection
- Callable Statements

4.4.2.1 DatabaseMetadata

The DatabaseMetaData class provides a lot of information concerning the structure and settings of a database and its Connection object. It should return a ResultSet otherwise it throws a SQL-exception. Typical connections are performing for example following methods:

- Are the inserts/updates/deletes visible for yourself/others
- What username is being used for this connection?
- What are the primary keys for a table?
- ...

4.4.2.2 ResultSetMetadata

The ResultSetMetaData class returns details about a ResultSet like

- How many columns are in a result set?
- From which table did a given column name come from?
- ...

4.4.2.3 ResultSet

The class which is checked here represents a result set of a database. It provides an application with access to database queries one row at a time. During query processing a ResultSet maintains a pointer to the current row being manipulated. There are many methods included which are "getting or updating" different objects.

4.4.2.4 BatchUpdates

In the test section BatchUpdates the processing batches are tested for add/clear/execute/continue.

4.4.2.5 Prepared statement

The Prepared Statement interface enables a SQL statement to contain parameters like a function definition. With this feature a single statement can be executed repeatedly with changing values for those parameters.

4.4.2.6 Scalar Functions

The Scalar Functions section checks the database for support of mathematical, string, time and date related functions and all kinds of outer-joins.

4.4.2.7 Time-Date-Functions

This part of the JDBCCTS checks whether the JDBC driver converts the date, time and timestamp values of the database accurately.

4.4.2.8 Batch-Update-Exceptions

One test gets the number of the actual batch updates. Others verify the BatchUpdateException Object which is created.

4.4.2.9 SQL-Exceptions

The test in this section checks if a SQL-exception is created dependent on a special case. Also the methods getErrorCode, get- and setNextExceptions, and getSQLState are tested.

4.4.2.10 Statements

In the Statement section there are all essential features concerning a statement tested like:

- Executing Queries/Updates
- Getting Resultsets/Resultsettypes
- Setting maximal Fieldsize/Rows

- ...

4.4.2.11 Connection

Here all different methods concerning the initial settings and creating Statements/Prepared Statements/Callable Statements are checked.

4.4.2.12 Callable Statements

A Callable Statement is very similar to a Prepared Statement but a Callable Statement is stored first in the database. A Callable Statement can be called a stored procedure as well. Stored procedures have the following advantages:

- They can be executed much faster because they are much faster than a dynamic SQL which must be interpreted each time it is issued.
- Syntax errors can be caught at compile time and not at runtime.
- It is only important to know the procedure, its inputs and outputs. It is irrelevant how the procedure is implemented and the tables are accessed or structured.

Disadvantages:

- They require more work to create and debug them (especially for a SQL newbie).
- There is no universal language for stored procedures (PL/SQL for Oracle and T-SQL for SQL Server).

4.4.3 Tested Database Management Systems

The choice of the databases has been made for logic and economic considerations. The JDBC drivers have been retrieved directly from the DBMS vendors and not from any commercial JDBC driver provider. More detailed information about the used DBMS and JDBC drivers is available in the appendix A.

4.4.3.1 Oracle

Oracle[27] is a high-performance reliable DBMS used for big business solutions, which is also in use at the Bioinformatics Group of the Institute for Biomedical Engineering[5].

4.4.3.2 Microsoft SQL Server

Microsoft SQL Server 2000[23] is also a very fast high performance DBMS which is available for a single-user as MSDE (Microsoft Developer Engine).

4.4.3.3 MySQL

The MySQL database[25] is an open source database which can also be used on small systems because the requirements are containing the resources.

4.4.3.4 Cloudscape

Cloudscape[13] is an IBM DBMS included in SUNs J2EE reference implementation.

4.4.4 Guidelines for JDBC Programming

This guideline is extracted from the results of the JDBC CTS, literature[39][36][34] and a number of web pages. To avoid problems when changing the DBMS, the following items should be observed:

- "INT" should be used instead of BOOLEAN.
- Attention should be paid to the use of DATE, TIME, TIMESTAMP, (LONG)(VAR)BINARY and LONGVARCHAR.
- Converting datatypes in the database is not supported.
- For quotes use *"*.
- The use of "multiple Resultsets" should be avoided.

- Full-outer-joins should not be used although normal outer-joins are supported from all DBMS.
- The use of "schemas" and "catalogs" should be avoided.
- Positioned delete and updates are not supported.
- There should not be open cursors/statements across commit or rollback.
- The best way is to use ResultSets from type TYPE_SCROLL_INSENSITIVE.
- The ResultSetsCurrency should only be from type TYPE_SCROLL_INSENSITIVE and the concur-type CONCUR_READ_ONLY.
- All functions concerning "Visible" and "Detected" should not be used.
- The Transaction-Isolation-Level should be TRANSACTION_SERIALIZABLE and TRANSACTION_READ_COMMITTED.
- Addbatch and Continuebatch should be avoided although Executebatch is supported in special cases.
- Take care of causing a classcast-exception with prepared statements and callable statements (mainly from the Oracle JDBC driver and the MySQL JDBC driver).
- Date-time methods and also the exception-testing methods should function well for the use in the different DBMS.
- Numeric, String and Time-Date functions are not well supported.
- Stored procedures should not be used.

4.4.5 Datatypes

This table should present the different datatypes of the DBMS. It should provide an overview to replace the datatypes changing the DBMS. These datatypes are extracted from literature [39][36][34] combined with the datatypes used in the JDBCCTS:

JDBC	JAVA	MS SQL Server 2000	Oracle 9i	MySQL	Cloudscape
BIGINT	long	BIGINT	NUMBER(38)	BIGINT	LONGINT
BINARY	byte[]	VARBINARY(24)	RAW(24)	VARBINARY(24)	BIT(192)
BIT	boolean	BIT	SMALLINT	BIT	BOOLEAN
CHAR	String	CHAR(n)	CHAR(n)	CHAR(n)	CHAR(n) or NATIONAL CHAR(n)
DATE	Date	DATETIME	DATE	DATE	DATE
DECIMAL	BigDecimal	DECIMAL(P,S)	DECIMAL(P,S)	DECIMAL(P,S)	DECIMAL(P,S)
DOUBLE	double	DOUBLE PRECISION	DOUBLE PRECISION	DOUBLE PRECISION	DOUBLE
FLOAT	double	FLOAT(30)	FLOAT	FLOAT(30)	FLOAT
INTEGER	int	INTEGER or INT	INTEGER	INTEGER or INT	INTEGER
LONGVARCHAR	String	VARCHAR(n)	LONG VARCHAR	VARCHAR(n)	LONG VARCHAR or LONG NVARCHAR
LONGVARBINARY	byte[]	VARBINARY(50)	LONG RAW	VARBINARY(50)	LONG VARBINARY
NUMERIC	BigDecimal	NUMERIC(P,S)	NUMERIC(P,S)	NUMERIC(P,S)	NUMERIC(P,S)
REAL	float	REAL	REAL	REAL	REAL
SMALLINT	short	SMALLINT	SMALLINT	SMALLINT	SMALLINT
TIME	Time	DATETIME	DATE	TIME	TIME
TIMESTAMP	Timestamp	DATETIME	DATE	DATETIME or TIMES-TAMP	TIMESTAMP
TINYINT	byte	TINYINT	NUMBER(4) or NUMERIC	TINYINT	TINYINT
VARBINARY	byte[]	VARBINARY(48)	RAW(48)	VARBINARY(48)	BIT VARYING(384)
VARCHAR	String	VARCHAR(n)	VARCHAR(n)	VARCHAR(n)	VARCHAR(n)

Chapter 5

Discussion

In this thesis a QC control component for MARS was developed, supporting the evaluation of a gel image created by gel electrophoresis of PCR products. Therefore also some management items are constructed to configure the elements of the QC component. It is possible to use different microtiter plates as well as a different pipetting schemes. Also the gel image can be configured due to a specific gel electrophoresis. Although the QC is very flexible it is possible to create quickly standard plates.

A guideline was prepared for the usage of JDBC to allow changing a DBMS without altering the code. Hence a table for the conversion of datatypes is presented, if it is intended to change a DBMS.

5.1 Future Aspects of Application Modelling

Application modelling will be the future technology to produce diverse scalable, reliable applications in a short period. Application modelling will simplify programming but a software engineer will not be freed of writing the templates and specific or extended cartridges. In the future AndroMDA will also provide template generation. It will be possible to include AndroMDA in the UML/CASE tool Poseidon, so it will be faster to generate the specific source code. A nice-to-have feature would also be to reverse engineer existing code for the UML tool because changes in the source code will not be reflected in the UML model.

5.2 Extension of the Quality Management

This QC application was created to provide a application for the use in laboratories but still other QC applications for the MARS have to be implemented:

- Quality control of Bacterial Growth plate
- Quality control of amplified RNA
- Qualtiy control of Oligo Stock plate

In the future a quality management application supporting the entire workflow will be created. The implemented QC application for PCR gel images could be extended and would be a part of the whole quality management. Therefore the UML model must be extended with the relevant entities and consequently the core of the quality management has to be considered.

5.3 Differences between EJB QL and JDBC

First question is: Why performing a JDBC Compatibility test although the business applications in a J2EE platform are using the EJB QL?

The EJB QL does not cover the complete SQL language and it is limited to the SQL92 standard. For performant database access JDBC would be much faster than the EJB QL. Furthermore EJB QL has a few other restrictions:

- Comments are not allowed.
- Date and time values are in milliseconds and use a Java "long" datatype. A date or time datatype should be an integer datatype. To generate a millisecond value the java.util.Calendar class should be used.
- Currently container-managed persistence does not support inheritance. For this reason two entity beans of different types cannot be compared.

But the EJB QL will always be the faster implementation of a database access and the programmer does not have to worry about the implementation of the database access.

5.4 Effects of changing a DBMS

Changing a DBMS for an application can cause serious problems if SQL commands are used carelessly. The main concern should be the different technologies supported by the different DBMS like "stored procedures". Further more attention should be paid to the differences between datatypes used by the DBMS. If the aim is to use different DBMS for an application a reduced set of SQL commands should be used although there is no warranty that all the functions will be supported.

Acknowledgements

First I want to thank all my supporters especially my supervisors Marcel Scheideler, Gerhard Thallinger and Zlatko Trajanoski. Further also my colleagues Bernhard Mlecnik and Thomas Truskaller appertain my thanks who supported me with their knowledge about the techniques and the framework.

Further I want to thank Michael Maurer, Robert Molidor, Christine Paar, Andi Prokesch, Alexander Sturn and my sister Regina Thumser who have been consulted for this thesis.

Thanks also go to my colleagues Roland Pieler and Johannes Rainer.

Finally I want to express my sincere gratitude to my family for supporting me with all their love and faith through out my entire studies.

Bibliography

[1] 21 CFR Part 11. WWW, Aug. 2003. http://www.21cfrpart11.com.

[2] Apache Software Foundation. WWW, Sep. 2003. http://www.apache.org/.

[3] Apache Struts Framework. WWW, Aug. 2003. http://jakarta.apache.org/struts/.

[4] Application Server. WWW, Aug. 2003. http://www.javaskyline.com/serv.html.

[5] Bioinformatics Group of the Institute for Biomedical Engineering. WWW, Sep. 2003. http://genome.tugraz.at/.

[6] Clinical Data Interchange Standards Consortium (CDISC). WWW, Sep. 2003. http://www.cdisc.org/standards/index.html.

[7] Digital Imaging and Communications in Medicine (DICOM). WWW, Sep. 2003. http://medical.nema.org/.

[8] Enterprise JavaBeans Technology. WWW, Sep. 2003. http://java.sun.com/products/ejb/.

[9] FDA 21 CFR Part 11. WWW, Aug. 2003. http://www.fda.gov/ora/compliance_ref/part11/.

[10] Gel Electrophoresis. WWW, Aug. 2003. http://brainarray.mhri.med.umich.edu/gelintro.html.

[11] Good Laboratory Practice. WWW, Aug. 2003. http://www.oecd.org/env/glp.

[12] Health Level Seven (HL7). WWW, Sep. 2003. http://www.hl7.org/.

[13] IBM Cloudscape. WWW, Sep. 2003. http://www-3.ibm.com/software/data/cloudscape/.

[14] Information about Poseidon Community Edition. WWW, Aug. 2003. http://www.gentleware.com/products/poseidonCE.php3.

[15] ISO 9000:2000. WWW, Aug. 2003. http://www.iso.ch/iso/en/iso9000-14000/iso9000/qmp.html.

[16] Java 2 Platform, Enterprise Edition (J2EE). WWW, Sep. 2003. http://java.sun.com/j2ee/.

[17] Java Database Connectivity. WWW, Sep. 2003. http://java.sun.com/products/jdbc/.

[18] Jboss Application Server. WWW, Sep. 2003. http://www.jboss.org/index.html.

[19] JDBC API 3.0 Specification. WWW, May. 2003. http://java.sun.com/products/jdbc/download.html#corespec30.

[20] JDBC Compatibility Test Suite. WWW, Sep. 2003. http://java.sun.com/products/jdbc/jdbctestsuite-1_3_1.html.

[21] JDBC Drivers. WWW, Sep. 2003. http://servlet.java.sun.com/products/jdbc/drivers.

[22] LIMSource. WWW, Aug. 2003. http://www.limsource.com/.

[23] Microsoft SQL Server. WWW, Sep. 2003. http://www.microsoft.com/sql/.

[24] Model Driven Architecture. WWW, Aug. 2003. http://www.omg.org/mda/.

[25] MySQL. WWW, Sep. 2003. http://www.mysql.com/.

[26] Object Management Group. WWW, Aug. 2003. http://www.omg.org/.

[27] Oracle Corporation. WWW, Sep. 2003. http://www.oracle.com/.

[28] SAP - Systems, Applications, Products. WWW, Aug. 2003. http://www.sap.com/.

[29] The tiered architecture of J2EE. WWW, Sep. 2003. http://www.iona.com/support/docs/e2a/asp/5.1/j2ee/aspintroduction/WhatIs3.html.

[30] Total Quality Management. WWW, Aug. 2003.
http://www.performance-appraisals.org/Bacalsappraisalarticles/articles/tqm1.htm.

[31] Velocity. WWW, Sep. 2003. http://jakarta.apache.org/velocity/.

[32] XDoclet. WWW, Sep. 2003. http://xdoclet.sourceforge.net/.

[33] Vierstraete A. PCR. WWW, Aug. 2003.
http://allserv.rug.ac.be/ avierstr/principles/pcr.html.

[34] Bales D. *Java Programming with Oracle JDBC*. O'Reilly, 1th edition, 2001.

[35] Reese G. *Database Programming with JDBC and JAVA*. O'Reilly, 2nd edition, 2000.

[36] Shapiro J R. *SQL Server 2000 The Complete Reference*. Osborne / McGraw-Hill, 1th edition, 2001.

[37] Bohlen M. AndroMDA. WWW, Sep. 2003. http://www.andromda.org/index.html.

[38] Jeckle M. UML Tools. WWW, Sep. 2003. http://www.jeckle.de/umltools.htm.

[39] Garcia M F. *Microsoft SQL Server 2000 Das Handbuch*. Microsoft Press, 1th edition, 2000.

[40] Martin R. *Elektrophorese von Nucleinsäuren*. Spektrum, 1th edition, 1996.

[41] Granjeaud S, Bertucci F, and JordanB R. Expression profiling:DNA arrays in many guises. WWW, 1999.
http://www.esil.univ-mrs.fr/ thieffry/Genomics/granjeaud_1999.pdf.

[42] Truskaller T. Data Integration into a Gene Expression Database. Master's thesis, TU-Graz, 2003.

Appendix A

JDBCCTS Setup details

A.1 MS SQL Server

A.1.1 JDBC Driver - Version:

SQLServer 2.2.0022

Filename: mssqlserver.jar, msbase.jar, msutil.jar

Creation date: 04.12.2002

A.1.2 Database - Version:

Microsoft SQL Server 2000 - 8.00.534 (Intel X86)

Nov 19 2001 13:23:50

Copyright (c) 1988-2000 Microsoft Corporation

Desktop Engine on Windows NT 5.1 (Build 2600: Service Pack 1)

A.1.3 URL:

```
jdbc:microsoft:sqlserver://hostname:port(usually 1433);
SelectMethod=cursor;DatabaseName=dbname(here:tempdb)",
"username", "password"
```

A.1.4 JDBC Driver - Classname:

```
com.microsoft.jdbc.sqlserver.SQLServerDriver
```

A.2 Oracle

A.2.1 JDBC Driver - Version:

Oracle JDBC driver 9.2.0.1.0

Filename: classes12.jar

Creation date: 18.08.2001

A.2.2 Database - Version:

Oracle9i Enterprise Edition Release 9.0.1.0.0 - Production With the Partitioning option JServer Release 9.0.1.0.0 - Production

A.2.3 URL:

```
jdbc:oracle:thin:@hostname:port(usually 1521):dbname(here:
oralin), "username", "password"
```

A.2.4 JDBC Driver - Classname:

```
oracle.jdbc.OracleDriver
```

A.3 MySQL

A.3.1 JDBC Driver - Version:

MySQL-AB JDBC Driver 3.1.0-alpha (Date: 2003/02/17 17:11:07 , Revision: 1.27.4.5)

Filename: mysql-connector-java-3.1.0-alpha-bin.jar

Creation date: 17.02.2003

A.3.2 Database - Version:

MySQL 4.1.0-alpha-max-debug

A.3.3 URL:

```
jdbc:mysql://hostname:port(usually 3306)/dbname(here:cts1)
?user=&password=
```

A.3.4 JDBC Driver - Classname:

```
com.mysql.jdbc.Driver
```

A.4 Cloudscape

A.4.1 JDBC Driver - Version:

Cloudscape Embedded JDBC Driver 4.0

Filename: cloudscape.jar, cloudutil.jar, cloudclient.jar, RmiJdbcDriver.jar

Creation date: 17.01.2002

A.4.2 Database - Version:

DBMS:cloudscape 4.0.6

A.4.3 URL:

```
jdbc:cloudscape:rmi://hostname:port(usually 1099)/dbname
(here:CloudscapeDB); user=; password=
```

A.4.4 JDBC Driver - Classname:

```
com.mysql.jdbc.Driver
```

Appendix B

JDBC Driver Compatibility Study - Details

B.1 Microsoft SQLServer 2000

B.1.1 DatabaseMetadata

At first a few characteristics of the MS SQLServer 2000 database engine are described. The following datatype conversions are not supported:

		source	
		DOUBLE	LONGVARBINARY
target	VARCHAR	X	X
	INTEGER	X	

Table B.1: Microsoft SQLServer 2000 datatype conversion marked with X is not supported.

Further the database does not support "table correlation names" and "specifying a LIKE escape clause". It supports "ANSI92 EntryLevelSQL" but not "ANSI92 IntermediateSQL" and "ANSI92 FullSQL" and also not "Extended-SQL-Grammar". There are only "Outer Joins" but no "Full Outer Joins" provided. "Commit/Rollback" across Open Cursors are not supported. The "data definition statement" within a transaction does not force the transaction to commit it. The JDBC driver does not get any Procedures, ProcedureColumns, Tables, Columns, ColumnPrivileges, TablePrivileges, BestRowIdentifier, VersionColumns, CrossReference, IndexInfo

and no primary/imported/exported keys as it should.
The database neither supports Resultsets from the Type "TYPE_SCROLL_SENSITIVE". Also the concurrency of a Resultset cannot be performed if the database is in "TYPE_SCROLL-_SENSITIVE". There is no function of all own/others inserts/deletes/updates and no detection of inserts/deletes/updates. The Transaction with the Transactionlevel "TRANSACTION_NONE" is not supported.

B.1.2 ResultSetMetadata

ResulsetMetaData does not return TableName and SchemaName like in MySQL or Cloudscape.

B.1.3 Batchupdates

The ContinueBatch method is not supported.

B.1.4 Scalar Functions

Many scalar functions are not supported:
Insert, Replace, User, Truncate, Curdate and Database, Left-, Right- and Fullouterjoins. But instead of "user" also "user_name" can be used. In the case "mod"(modulo) was a SQL-Exception which has been generated because it can only be used for the datatype "INTEGER" and not like used in the JDBCCTS as "NUMERIC". Also the function TimestampDiff01 which calculates the difference of two timestamps in the hundredth of a second caused a SQL-Exception. The problem here is that the difference is too big. If there would be a smaller difference the test would work correct.

B.1.5 Time-Date Functions

In the dateTime-Section all the functions have performed well except GetTime, GetTimestamp and GetDate when used with a SQL Null and the Calendar-Object.

B.1.6 Callable Statements

Further the JDBC driver has a problem with the scaling with this command `registerOutParameter(1,java.sql.Types.NUMERIC,15);` . The scaling is used for defining the decimal places. The JDBC driver converts number 999999999999999 into 0,999999999999999. Alternatively this command can be used like this:
`registerOutParameter(1,java.sql.Types.NUMERIC)`
Sometimes problems can occur when comparing some dates because the database or JDBC-driver changes the value although the date is regarded as correct.
But if the method java.sql.DATE.compareTo(java.sql.DATE) is used there is a difference between the two dates. Indeed a Timestamp can be taken and not a date but this is an error and therefore should be avoided.
There have been also some ClassCastExceptions in the test but these are thrown by the JDBC driver which creates the following error message: `[Microsoft][SQLServer 2000 Driver for JDBC]Value cannot be converted to requested type.`
The following conversion cannot be fulfilled:

		target and source	
		java.sql.Types.DATE	java.sql.Types.TIME
source	java.util.Date	X	X
source and target	Time		X
source and target	java.sql.Date	X	

Table B.2: Microsoft SQLServer 2000 datatype to Java datatype conversion marked with X is not supported.

Finally it is very important that if there is a use of boolean values it should be taken "0" and "1" as values and not "true" and "false".

B.2 Oracle 9i

B.2.1 DatabaseMetadata

First of all not all "Procedures" are callable and accordingly not all tables are selectable. Further the database supports altering a table with dropping a column and all the converts of datatypes do not work. There cannot be multiple ResultSets performed. The Oracle database cannot understand ANSI92 Intermediate SQL and ANSI92 Full SQL. It does not support catalogues in ProcedureCalls, TableDefinitions, IndexDefinitions and PrivilegeDefinitions. Also PositionedDelete, PositionedUpdate cannot be executed. The commit/rollback across OpenCursors/OpenStatements are not supported. GetProcedures, GetProcedureColumns, GetTables, GetCatalogs and GetColumns do not return anything. All other inserts/updates/deletes are not visible except updates if they are executed in TYPE_SCROLL_SENSITIVE-mode. Also own inserts are not visible and own deletes/updates in the "TYPE_FORWARD_ONLY"-Mode. Inserts/updates/deletes are not detected. The Transaction with the Transactionlevel "TRANSACTION_NONE", "TRANSACTION_READ_UNCOMMITTED",
"TRANSACTION_REPEATABLE_READ" is not supported. There are no return and following "get"-methods getting TablePrivileges, VersionColumns, PrimaryKeys, ImportedKeys, ExportedKeys, CrossReference and IndexInfo. The getBestRowIdentifier functions only for "bestRowTransaction" with and without nullable columns.

B.2.2 ResultSetMetadata

The getScale method returns a negative value although the other databases return 0 and getTableName does not return anything like the MS SQLServer 2000. A batch cannot be added.

B.2.3 Prepared & Callable Statements

In the section callable statements or procedures following JDBC commands can not be well performed: The getTimestamp-method returns a SQL-exception since it is called with a Calender object as a parameter.

A big difference compared to the other databases is that "0" or "0.0" is returned if the value is number and no text or character. The MS SQL Server 2000 and Cloudscape returning "null".
ClassCastExecptions: In principle it can be differentiated between two JDBC commands: setObject and getObject. Whereby setObject wants to cast the specific datatypes to the desired datatypes before send updating/inserting the specific value. The other ClassCastException is obtained because first the object is retrieved from the database and secondly it will be casted to a specific datatype which is used later.
getObject: There cannot be casts of the following datatypes:

		target		
		Double	BigDecimal	Integer
source	java.sql.Types.DOUBLE	X		
	java.sql.Types.REAL	X		
	java.sql.Types.FLOAT	X		
	java.sql.Types.DECIMAL		X	
	java.sql.Types.SMALLINT			X

Table B.3: Java datatype to Oracle datatype conversion marked with X is not supported.

setObject: There can not be casts of following datatypes:

		source									
		String	BigDecimal	Boolean	Integer	Long	Float	Double	Date	Time	Timestamp
target	java.sql.Types.TINYINT	X		X							
	java.sql.Types.SMALLINT	X		X							
	java.sql.Types.INTEGER	X		X							
	java.sql.Types.BIGINT	X		X							
	java.sql.Types.REAL	X		X							
	java.sql.Types.FLOAT	X		X							
	java.sql.Types.DOUBLE	X		X							
	java.sql.Types.DECIMAL	X		X		X	X				
	java.sql.Types.NUMERIC	X		X							
	java.sql.Types.DATE	X									X
	java.sql.Types.TIME	X									X
	java.sql.Types.TIMESTAMP	X							X		
	java.sql.Types.CHAR		X	X	X	X	X	X	X	X	X
	java.sql.Types.VARCHAR		X	X	X	X	X	X	X	X	X
	java.sql.Types.LONGVARCHAR		X	X	X	X	X	X	X	X	X

Table B.5: Oracle datatype to Java datatype conversion marked with X is not supported.

B.2.4 Scalar Functions

The Oracle DBMS does not support the following scalar functions: Insert, Left, Right, Repeat, Space, IfNull, Cot, Dayname, Dayofweek, Dayofyear, Week, Monthname, Quarter, Degrees, Radians, Rand, Difference, TimestampAdd and TimestampDiff. Also "Database" is not performed well because the result is "null". Further following SQL statements which are supported are not well performed by the JDBC driver: Lcase, Locate, Ucase, Char and Truncate.

B.2.5 Connection

In the section connection the method getCatalog does not work well because of returning "null".

B.3 MySQL

Generally there is a big difference to the other databases because MySQL cannot work with stored procedures and that is a big disadvantage of the DBMS.

B.3.1 DatabaseMetadata

Like the Oracle database the tables in the MySQL database are not selectable. It does not use a local file for a table as well. The Identifier-Quote-string for this database is *‘* although the other databases take *"*.
MySQL does not order or group by "unrelated". Multiple Resultsets are not supported as well full-outer-joins. Like Microsoft SQL Server 2000 this database only understands the ANSI92 Entry Level SQL.
GetSchemaName and GetProcedureName deliver no return compared to the other databases. There are no "Schemas" supported as well also no "Catalogues" except "Catalogues in Data - Manipulation". It is also not allowed to make a "positioned" delete/update and no "select for update". Further it is not supported to make correlated subqueries and open cursors/open statements across commit/rollback. It is not allowed to make DataDefinition- and DataManipulation-Transactions. MySQL only supports ResultSetType TYPE_SCROLL_INSENSITIVE. It does

not support visible-functions and also not detected-function like MS SQL Server 2000. Also no "User Defined Types" and no Transaction with the Transactionlevel "TRANSACTION_NONE" are supported. There are no results for Table/Columns Privileges and Best-Row-Identifiers as well for Version-Columns, Cross-Reference and all Primary/Imported/Exported keys.Even if the common function "supportsConvert" reports that this is not supported some specific conversions worked well. Some datatype conversions which are not supported are listed below:

		source	
		VARCHAR	INTEGER
target	ARRAY	X	
	BIT	X	X
	BLOB	X	
	CLOB	X	
	DISTINCT	X	
	JAVA_OBJECT	X	
	REF	X	
	STRUCT	X	
	DATE		X

Table B.6: MySQL datatype conversion marked with X is not supported.

B.3.2 ResultSetMetadata

The Resulset-Meta-Data is not "currency" and not signed and the columns are not nullable. Like the other databases it returns no Schema-Name and the ColumnTypeName returns "UNKNOWN" although the other databases deliver a result for this method. But compared to the other databases writing on columns is "definitely writable".

B.3.3 ResultSet

A conversion from a String to a Float should only be made with attention to the precision of the float-value that can be stored in the MySQL database. If there is a value which is stored as a float in the database this should be retrieved from MySQL with getInt. This value should only have a certain size.

B.3.4 Prepared Statements

Prepared Statement: Calling the GetMetaData causes a SQL-Exception because this method is not implemented.

ClassCastExceptions:
getObject: There cannot be casts of following data-types:

		target					
		BigDecimal	Double	byte[]	java.util.Date	java.sql.Time	java.sql.Timestamp
source	BigDecimal(String)	X					
	Double		X				
	byte[]			X			
	java.sql.Date				X		
	java.sql.Time				X		
	java.sql.Timestamp						X
	java.sql.Types.FLOAT		X				
	java.sql.Types.DOUBLE		X	X			
	java.sql.Types.DECIMAL	X					
	java.sql.Types.NUMERIC	X					
	java.sql.Types.BINARY			X			
	java.sql.Types.VARBINARY			X			
	java.sql.Types.LONGVARBINARY			X			
	java.sql.Types.DATE				X		
	java.sql.Types.TIME					X	
	java.sql.Types.TIMESTAMP						X

Table B.8: MySQL datatype to Java datatype conversion marked with X is not supported.

B.3.5 Scalar Functions

Following ScalarFunctions are not supported: IfNull, Sign, Difference, TimestampAdd and TimestampDiff. The functions "Hour", "Minute" and "Second" are also not correct implemented in the JDBC driver because the results are "null".

B.4 Cloudscape

B.4.1 DatabaseMetadata

A Search-String-Escape and also a Extra-Name-Characters could not be retrieved from Cloudscape. Further it does not support "altering tables with dropping a column", "expression in order by" and "order by unrelated". The use of "Multiple Resultsets" is not provided and the database only understands "Minimum SQL Grammar". Like in MS SQL Server 2000 and MySQL in the Cloudscape cannot perform "Full outer joins". The getCatalogSeparator does not deliver a value which is in the MS SQL Server 2000 and MySQL ".". Cloudscape does not support any catalogues. It allows "open statements across a commit" although it does not at a "rollback". Further it is not supported to have open cursors at commit or rollback. The definition of data causes no TransactionCommit. Procedures, ProcedureColumns and Tables with GetProcedures, GetProcedureColumns and GetTables could not be retrieved. Further no catalogs and no columns can be obtained. There cannot be a Resultset from type "TYPE_SCROLL_SENSITIVE" and a ResultSetConcurrency can only be from Type "TYPE_FORWARD_ONLY, CONCUR_READ_ONLY" or "TYPE_SCROLL_INSENSITIVE, CONCUR_READ_ONLY". All inserts/updates/deletes are not visible for own changes but are visible from others. But there a no inserts/deletes/updates detected. The Transaction-Isolation-Level "TRANSACTION_NONE" and "TRANSACTION_READ_UNCOMMITTED" are not supported. Upon the test there was no return for Column-Privileges, TablePrivileges BestRowIdentifiers, VersionColumns, Primarykeys, Imported Keys and Exported Keys, Crossreference and Indexinfos.

B.4.2 ResultSetMetadata

It is not "currency" and not "definitely writeable". Further a schema-name cannot be retrieved.

B.4.3 Batchupdates

The continuation of a batch is not supported from Cloudscape.

B.4.4 Scalar Functions

Most of the Scalar functions like mathematical and time-date-functions are not supported by Cloudscape. Here are only those which can be used: Length, Locate, Ltrim, Rtrim, Substring, User, Abs, Sqrt and Curdate.

B.4.5 Connection

The Catalog in the section connection could not be retrieved.