Sample Annotation of MS Experiments

Andreas Scheucher

DIPLOMARBEIT

eingereicht am
Fachhochschul-Diplomstudiengang

Bioinformatik

in Hagenberg

im August 2006

Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die aus anderen Quellen entnommenen Stellen als solche gekennzeichnet habe.

Hagenberg, am 19th September 2006

Andreas Scheucher

Contents

Acknowledgments

First of all, I want to thank MSc Jürgen Hartler for the great help during design and implementation of the extensions for MASPECTRAS, and the helpful proofreading and comments on the thesis's text.

Special thanks to Prof. Zlatko Trajanoski for giving me the possibility to do my diploma thesis at the Institute for Genomics and Bioinformatics - TU Graz.

Then, I want to thank my supervisor Mag. Dr. Jacques Colinge from the Upper Austria University of Applied Sciences in Hagenberg for his advices and proofreading of this paper.

Finally, I would like to thank my family, especially my parents, for supporting me during my studies and my diploma thesis.

Kurzfassung

Die Proteomik beschäftigt sich mit der Erforschung des Proteoms, der Gesamtheit der Proteine, und zählt zu den am stärksten vorangetriebenen Forschungsgebieten der vergangenen Jahre. Durch den technologischen Fortschritt entstanden neue immer komplexere experimentelle Methoden und Protokolle. Das führte dazu, dass publizierte Experimente kaum oder nur mit größter Anstrengung reproduzierbar sind. Daher wird der Ruf nach Standards für die sorgfältige Beschreibung von Proteomikexperimenten immer lauter.

Am Institut für Genomik und Bioinformatik wurde das MAss SPECTRometry Analysis System (MASPECTRAS) entwickelt. Es handelt sich hierbei um eine web basierte Softwareplattform die zum Management und der Analyse von liquid chromatography tandem mass spectrometry (LC-MS/MS) gewonnen Daten dient. MASPECTRAS basiert auf dem relationalen Datenbankschema des Proteome Experimental Data Repository (PEDRo) und folgt den Richtlinien (Minimum Information About a Proteomic Experiment MIAPE) der von der Human Proteome Organisation (HUPO) ins Leben gerufenen Proteomics Standards Initiative (PSI). Damit eine lückenlose Beschreibung der Probenaufbereitungsdaten ermöglicht wird, wurde die bestehende Applikation im Rahmen dieser Arbeit erweitert und an die neuesten MIAPE Richtlinien im Bereich Probenaufbereitung angepasst. Die Erweiterung besticht durch ihre benutzerfreundliche und doch flexible Verwaltung der Daten über eine Baumstruktur, die sowohl das Splitten der Proben als auch die sequentielle Eingabe der einzelnen Aufbereitungsschritte in beliebiger Zusammenstellung ermöglicht. Durch die Integration des am Institut entwickelten Usermanagementsystems und die integrierte Exportfunktion erfolgt eine einfache und kontrollierte Verbreitung der Experimente. Die Entwicklung der Web-Applikation mit Datenbank Backend erfolgte mit Java, Java 2 Enterprise Beans, Struts und einen AndroMDA Codegenerator.

Die entwickelte Applikationserweiterung von MASPECTRAS stellt ein nützliches Hilfsmittel dar, welches die Konsistenz der LC-MS/MS Probenaufbereitungsdaten garantiert, und die Verwaltung und Verbreitung vereinfacht.

Abstract

Proteomics is the study about the Proteome, the collectivity of all proteins. In recent years proteomics became one of the hottest areas in research. Driven by advances in technology in recent years, new and complex methods and protocols emerged. However, published experiments were nearly not or very hard reproducible. Therefore there is a urgent need for a standardised description.

The Institute for Genomics and Bioinformatics developed the Mass SPECTRometry Analysis System (MASPECTRAS). MASPECTRAS is a web-based software platform for the management and analysis of proteomic LC-MS/MS data, which adopted and extended the relational database scheme of the Proteome Experimental Data Repository (PEDRo). It fulfils the MIAPE (Minimum Information About a Proteomic Experiment) standard of the Proteomics Standards Initiative (PSI), which is part of the Human Proteome Organisation (HUPO). To provide a complete description of the sample pre-processing, the existing application has been extended and adopted to the newest guidelines of MIAPE. The developed sample pre-processing part of system features an intuitive and flexible module for the administration of the data via a tree-structure. The flexible design permits splitting of probes as well as the arbitrary assembly of sample pre-processing steps. The sharing and dissemination of the data is easy due to the integrated features from the user management system of the institute and the export module. The developed web-application with database backend is based on Java, Java 2 Enterprise Beans, the struts technology, and an AndroMDA code generator

The developed extension of MASPECTRAS represents a useful tool which guarantees the integrity of LC-MS/MS sample pre-processing data and eases the administration and dissemination of proteomic mass spectrometry experiments.

Chapter 1

Introduction

1.1 Biological Background Information

This chapter gives an overview over the synthesis, localisation, and analysis of proteins and some necessary basics to see it in the biological context.

1.1.1 Deoxyribonucleic Acid - DNA

The Deoxyribonecleic Acid (DNA) is the place where the genetic information is stored and given from cell to cell during cell proliferation. It consists of huge amount of nucleotides and has the shape of a double helix. The sequence of the nucleotides are coding the genetics information in genes. This genes are read by the polymerase, which produces Messenger Ribonucleic Acid (mRNA).

1.1.2 Ribonucleic Acid - RNA

The Ribonucleic Acid (RNA) is like the DNA a long polynucleotide. Only one different nucleotide is used to code the sequence. There are several different kinds of RNA existing in the cell (e.g. tRNA, mRNA,...). The mRNA, which was produced as negative copy of the DNA, is used as template for proteins.

1.1.3 Proteins

Proteins are the building blocks and workhorses of the cell. They have either structural or enzymatic function. The proteins with structural function are used comparable to bricks to build cellular compartments. Enzymatic proteins trigger and direct chemical reactions in the cell. They are biological catalysts.

To understand how proteins work, we have to know how they are build. To get a picture of the spatial shape, we inspect them from four different

points of view [7]:

Primary structure of an protein is the sequence of amino acids (AAs). This sequence defines the folding possibilities, but does not define the folding itself.

Secondary structure defines the occurrence of well known shapes. The most often occurring are the alpha helix and the beta sheet. This structures can only occur at certain AA sequences.

Tertiary structure is describing spatial relation between the elements of the secondary structure. It determines the angle between them. The cause of this structure are the chemical properties of the AAs lying in the region, building hydrophobic cores, or salt bridges.

Quaternary structure is the shape of protein complexes, which are build by more than a single protein. This structure is acting often as one unit.

The folding of the proteins often is supported by helper proteins, which are called chaperons. They are involved in the folding of new translated proteins and they repair damaged ones.

1.1.4 Central Dogma of Biology

For long time there was the theory of the *Central Dogma of Biology* accepted to be true. It says that, DNA generates RNA generates protein. After several years it was found out, that it has to be adopted, because also RNA can make DNA (see figure 1.1). The main production of biological macromolecules follows the central dogma.

But there are several possibilities of modifications from DNA to AA sequence, where modifications are possible. For example alternative splicing of the mRNA generates different AA sequences then would be expected as direct product of the DNA. These modifications are the reasons, why it is necessary to examine proteins and it is not enough to inspect the DNA and deduce from the gained information to the properties of the proteins finally translated on the end of the translation chain.

1.2 Treatments of Protein Samples

To process and identify proteins it is necessary to undergo a chain of treatments. First, proteins are extracted and purified to get rid of the unwanted particles. After the proteins had been purified standard methods to separate and identify them are applied.

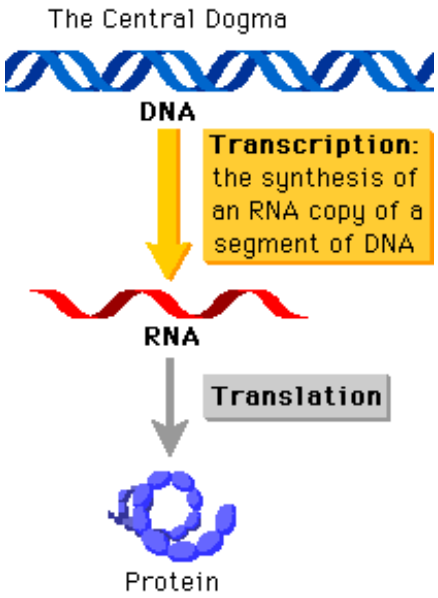


Figure 1.1: The central dogma of biology. It explains in rough lines the main synthesis paths of proteins. I fact in the transcription from DNA to mRNA variance can occur. This is called the alternative splicing. It leads finally to other protein AA sequence than you could expect from reading directly the DNA sequence. This is one reason why proteomics experiments are important. (taken from [28])

1.2.1 Gel Electrophoresis

Electrophoresis is one of the the main techniques for molecular separation in today's cell biology. Because it is such a powerful technique, and yet reasonably easy and inexpensive, it has become commonplace. In spite of many physical arrangements for the apparatus, and regardless of the medium through which molecules are allowed to migrate, all electrophoretic separations depend upon the charge distribution of the molecules being separated [14].

Electrophoresis can be one dimensional (i.e. one plane of separation) or two dimensional [14].

1D Gel Electrophoresis

One dimensional electrophoresis is used for most routine protein and nucleic acid separations [14]. "When the detergent SDS (sodium dodecyl sulfate) 2 is used with proteins, all of the proteins become negatively charged by their attachment to the SDS anions. When separated on a polyacrylamide gel, the procedure is abbreviated as SDS–PAGE (for Sodium Dodecyl Sulfate

PolyAcrylamide Gel Electrophoresis). The technique has become a standard means for molecular weight determination“ [14].

“Polyacrylamide gels are formed from the polymerization of two compounds, acrylamide and N,N-methylene- bis-acrylamide. Bis is a cross-linking agent for the gels. The gels are neutral, hydrophillic, three-dimensional networks of long hydrocarbons cross linked by methylene groups“ [14]. The separation of molecules within a gel is determined by the relative size of the pores formed within the gel.

“In electrophoresis, proteins are separated on the basis of charge. In case of using SDS it is negatively charged. In normal operation, a column of gel is partitioned into three sections, known as the Separating or Running Gel, the Stacking Gel and the Sample Gel. Electrodes are attached to the ends of the column and an electric current passed through the partitioned gels” [14]. Negatively charged proteins are carried by the current towards the positive electrode. The smaller a protein is, the faster it travels. This splits them up by their molecular size and creates bands of proteins with the same mass.

2D Gel Electrophoresis

2D gel electrophoresis is an orthogonal separation technique Proteins are separated by two different physicochemical principles. First, they are separated on the basis of their (pH-dependent) net charges by isoelectric focusing (IEF) [13]. Isoelectric focusing is a technique whereby proteins are concentrated based upon their isoelectric point (pI). The isoelectric point for a protein corresponds to the pH at which the protein has no net charge. This isoelectric point is a function of the protein's primary structure as well as the spatial orientation of the protein side groups [15]. In the second step the proteins in the gel is separated on the basis of their molecular masses by electrophoresis in the presence of SDS [13] in the second dimension, like described for 1D Gel Electrophoresis.

1.2.2 Liquid Chromatography

A nice definition of chromatography was given by the Union of Pure and Applied Chemistry (IUPAC) [6]:

> Chromatography is a physical method of separation in which the components to be separated are distributed between two phases, one of which is stationary (the stationary phase), while the other (the mobile phase) moves in a definite direction. A mobile phase is described as “a fluid which percolates through or along the stationary bed in a definite direction”. It may be a liquid, a gas or a supercritical fluid, while the stationary phase may be a solid, a gel or a liquid. If a liquid, it may be distributed on a solid, which may or may not contribute to a separation process.

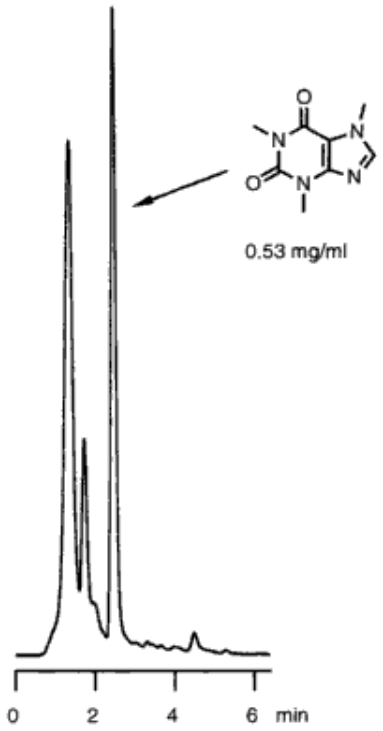


Figure 1.2: HPLC separation of coffee. Conditions: column, 15cm x 2mm i.d., stationary phase, YMX 120 ODS-AQ, 3μm; mobile phase, 0.3ml $min^{-}1$ wateracetonitrile (8:2); UV detector 272nm [19](taken from [19])

A chromatographic instrument consists of a device for sample introduction, a mobile phase, a stationary phase and a detector. In Liquid Chromatography the sample has to be introduced into the mobile phase without any discrimination effects. The separation of a sample in chromatography occurs, when the different components interact with the mobile and the stationary phase to a different extent. Due to this interaction, the components take different time to travel through the column. On the end of the column a detector analyses the liquid phase (see figure 1.2).

Liquid Chromatography is often linked with mass spectrometry. The main disadvantage of liquid chromatography is its inability to get an unequivocal identification of a component of a sample. Therefore Mass Spectrometry (MS) is used to identify the components in the fractions. In a first step liquid chromatography is used to separate the sample mixture into the components. These fractions are identified in a second step with the mass spectrometer.

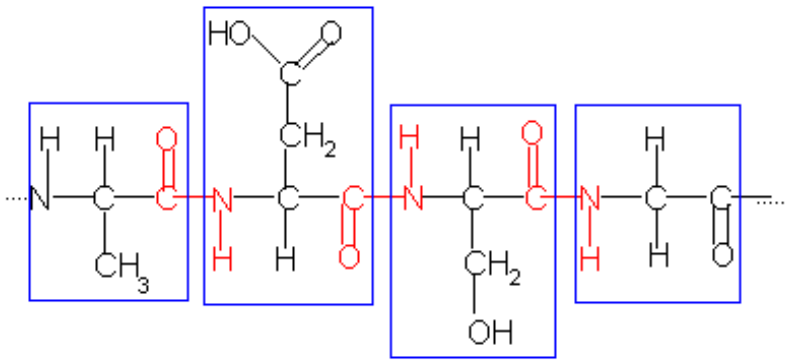


Figure 1.3: A short amino acid sequence of a protein. In the picture you can see four amino acids and in red you can see the atoms connected to a peptide bond. (taken from [27])

1.3 Mass Spectrometry

Gel electrophoresis and liquid chromatography are good methods to separate probes but the results can be interpreted in a very wide range. This is the point where Mass Spectrometry (MS) comes into view. In MS it is possible to identify the exact atomic mass of the molecules in the probe. This is a very discriminating information for molecule identification.

1.3.1 Proteins

To understand what is happening in MS, we have to know more about proteins. As already mentioned proteins are macromolecules build of a long chain of amino acids (AS) connected with each other by peptide bonds. In the picture 1.3 four amino acids are linked together. The atoms in the blue box are one amino acid each. The connection between them in read are showing peptide bonds, which are able to rotate freely. Hence, they have the flexibility to achieve different properties, which are mainly given by the spatial shape, the side chains hydrophilicity, hydrophobicity, and the positive or negative charge of the amino acids (either heading inside or showing outside of the protein).

1.3.2 Identification of Proteins with Mass Spectrometry

Analysis in mass spectrometry are done in high vacuum with an array of instruments. The basic principle is always the same. “First, an ion source produces gaseous ions of the molecule of interest, then a mass analyzer differentiates the ions according to their mass-to-charge ratio and finally,

a detector measures the ion beam current and produces a mass spectrum, which contains information about the ions that have been formed in the ion source" [16]. I will introduce the procedures and instruments used in LC-MS/MS which are used for evaluation by MASPECTRAS.

Digestion of Proteins

Proteins can be up to more than 1000 amino acids long. This is too large to be detected with an mass spectrometer. Hence, there is used an enzyme, in most cases trypsin, to cut the proteins into short peptides[1] with an overage length of 12 amino acids, by breaking the peptide bonds. This is called digestion (tryptic digestion in the case of trypsin). This peptides are short enough to be measured in the mass spectrometer.

Liquid Chromatography

As first step to separate the proteins is a Liquid Chromatography (LC). For details on LC see section 1.2.2. The result of the LC are fractions with one or only some few peptides in it. These fractions are analysed by mass spectrometry.

Ionisation Methods

There are several methods known for ionising a probe. In the 80s two new methods had been discovered, which allowed to work with such big molecules like proteins [16]. These had been Matrix Assisted Laser Desorption/Ionization (MALDI) and ElectroSpray Ionisation (ESI).

MALDI The digested probe is mixed with a detergent called MALDI matrix. This mixture is then crystallized and shot by a laser. The molecules in the probe absorb the energy, are desorbed, ionised in a gaseous state [16].

ESI The probe is dissolved and moved trough a capillary. On the end of the capillary a high volt is applied. This high voltage leads to explosion of the drops coming out of the capillary. The probe is also ionised [16].

MS Detectors

Also for the mass detection there are several different detectors available. Two of the most important ones are the ion trap and the Time Of Flight (TOF) detector.

[1] Short amino acid oligopeptides up to about 100 AS are called peptides. Only longer AS oligomeres are called Proteins

TOF The ions are accelerated in an electrical field depending on their charge and size/weight. Small and light ions are accelerated more than long and heavier ions. The time of flight of the ions through the TOF MS, which is depending on the mass/charge ration, is measured. This results are not very precise, because of the initial speed of the ion after the laser shot. To extend the accuracy of the mass spectrometer, the ionised peptides are reflected in their trajectory. This eliminates the influence of the different initial speed and extends the precision up to 50-100 ppm[2] [9].

Ion Trap A Ion Trap, in the most of the cases a Quadrupole, is build "of three hyperbolic electrodes: 'the ring electrode, the entrance end cap electrode, and the exit end cap electrode. These electrodes form a cavity in which it is possible to trap and analyze ions. Both end cap electrodes have a small hole in their centres through which the ions can travel. The ring electrode is located halfway between the two end cap electrodes. Ions produced from the source enter the trap through the inlet focusing system and the entrance end cap electrode. Various voltages and radio frequency potentials are applied to trap and/or eject ions according to their mass to charge ratios" [16].

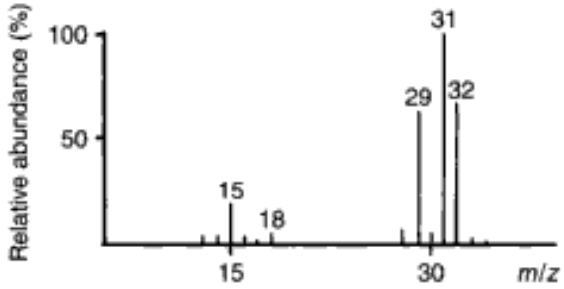
LC-MS/MS

In Liquid Chromatography - Mass Spectrometry / Mass Spectrometry (LC-MS/MS) also called Tandem Mass Spectrometry, the peptides are separated three times during one run. The first step is done with a Liquid Chromatography (LC) to separate the proteins from each other. The chromatograph is connected to the ionisation unit. For example an ESI. There the peptides are ionised and introduced into the first mass spectrometer and the fractions are transferred into it. Most likely it is a quadrupole ion trap. There, the three or four most abundant peptides are selected and transferred into a Fraction Chamber, where the peptides are fractionated again. These fractionated peptides are then introduced into the second mass spectrometer. This can be again an quadrupole or an TOF instrument. There again the ions are analyzed and a mass spectrum is generated.

Mass Spectra

The result of the measurement is then shown in the mass spectrum. On the x axis the mass per charge m/z is plotted and on the y axis the measured intensity of a given mass to charge ratio is displayed. As idealized result, you would see a peak for every peptide in the picture. But the chemical elements of peptides can consist of isotopes, therefore commonly an identified peptide

[2] ppm stands for parts per million.



m/z	Relative abundance (%)	m/z	Relative abundance (%)
12	0.33	28	6.3
13	0.72	29	64.0
14	2.4	30	3.8
15	13.0	31	100.0
16	0.21	32	66.0
17	1.0	33	0.73
18	0.9	34	~0.1

Figure 1.4: The mass spectrum of methanol. It shows for a simple molecule the relative abundance for given m/z values graphically and textual. A list of relative abundances is written into a file by the mass spectrometer, the so called peak list which is then used for computational analysis. (taken from [11])

consists of more than one peak in the MS spectra. One for the monoisotopic, two for the peptides with two isotopes, and so on. This can complicate the identification of the proteins, as the peaks for certain peptides can overlap. In the picture 1.4 you can see a simple example for methanol. It has been chosen for its simplicity. For computer based further analysis, the result is available as so called peak list. This is a long list, with m/z values and their abundance in the probe. The peak lists can be used for different computer based analysis.

Interpretation of Results / MS Result Search Engines

The MS spectrum delivers the abundance of mass per charge. This information is not very helpful. So there are methods needed to link this information with protein identifications.

As result of long analysis of the scientific community, there are big protein databases available for a lot of different species, including the human one. Databases store information about the sequence, the structure, and a lot more about the examined proteins. For detecting proteins in our probe

we can take advantage of this information, since we are able to calculate the theoretical masses for all of this peptides and then can compare them with our spectras.

The comparison of experimental determined mass spectra with theoretical mass spectra is done by MS result search engines like SEQUEST, Mascot, Spectrum Mill, X! Tandem, and OMSSA. They work all based on the same principle, roughly outlined here. They iterate over every sequence in the database and work off the following steps for each:

- Find out, where in the theoretical protein sequence (out of the database) the enzyme for digestion would cut to get all of the possible peptides in our probe. As we know the sequence pattern of cut sites for the enzymes, this is no problem.
- A theoretical fragmentation of the peptides has to be done next to simulate the backbone break of the peptides in the fragmentation cell. As the backbone can break at every bond and also side chains of amino acids can break, the decision has to be made which fragmentation in calculation have to be considered to limit the calculation costs.
- Now the theoretical mass of the theoretical peptides we have found for the protein are calculated. Here it depends how much effort one want to take. Isotopes, modifications like methylation, phosphorylation and others can be considered.This increases the quality of search results, but also the calculations costs.
- After achieving the theoretical masses for a protein sequence, they can be compared with the experimental mass spectrum gotten from the probe analysis. This is done for every sequence in the database. It is ranked by the quality of the similarity. For this comparison there are several different algorithms available. They base on different approaches, often on probabilistic ones.

The result, is a list of best matches against a given protein database. Based on the several parameters (e.g. matched ion count, score, correlation factors) it has to be distinguished, whether the best result is a solid hit and really identifies the examined protein in the probe or the peptides match accidentally.

1.4 Scopes of MASPECTRAS

This section should give the reader an overview of the MASPECTRAS application, which this thesis work is a part of.

1.4.1 Overall MASPECTRAS Project Goals

The Institute for Genomics and Bioinformatics has "developed MAss SPECTRometry Analysis System (MASPECTRAS), a platform for management and analysis of proteomic LC-MS/MS data. MASPECTRAS is based on the Proteome Experimental Data Repository (PEDRo) relational database scheme and follows the guidelines of the Proteomic Standard Initiative (PSI)." [18] PEDRo is an suggestion for implementing a data repository for proteomic experiments allowing to store all processing steps, with related information, for the probe treatment down to the mass spectrometry experiment. MASPECTRAS supports the following main features as explained in [18]:

- Import and parsing of MS searches in the following search engines: SEQUEST, Mascot, Spectrum Mill, X! Tandem, and OMSSA.
- Peptide validation.
- Clustering of proteins based on Markov Clustering and multiple alignments.
- Quantification using the Automated Statistical Analysis of Protein Abundance Rations algorithm (ASAPRation).

Analysis is not the only purpose of MASPECTRAS. MASPECTRAS was designed "as web-based platform for management and analysis of proteomic LC-MS/MS data. It was developed using stat-of-the-art software technology and enables data import form various search engines. Analytical modules are provided along with visualization tools and PRIDE[3] export." [18]

1.4.2 Thesis Project Goals

Proteomics experiments are widely used and often reported in scientific papers. As the sample treatment is complex and the reproduction of the experiments often is not successful, because of missing information in the papers, there is put a lot of effort in finding a standard, to overcome this problem of missing reproducibility. The Proteomics Standard Initiative (PSI), which is part of The Human Proteome Organization Proteomics (HUPO), is the driving force pushing further the standards for storing data in proteomics. The arising standard is the Minimum Information About a Proteomic Experiment (MIAPE).

MIAPE Compliant Experiment Tracking

MIAPE defines all the necessary attributes to be able to reproduce protein identification mass spectrometry experiments. This starts from the beginning of the sample treatment and covers all steps to the mass spectrometry.

[3] PRoteomics IDEntifications database (PRIDE).

To ensure that MASPECTRAS current database schema based on PEDRo is compliant to the latest MIAPE standards, the current situation of the database schema has to be analyzed and if necessary it has to be updated to fulfill the MIAPE standards.

Data Maintenance Web Interface

The second step is to design and implement an intuitive web-interface for an user friendly and easy input of the sample processing data. As lab work often is done on different places, a web application is the method of choice, as it allows the data feed to be done on every computer with Internet connection.

XML Export of Data

One strong argument against an application is the inaccessibility of data if it is stored internally (whether in files in a proprietary format or in a database which can not be accessed). This can prevent the use of the data later on in a lot of circumstances. No matter whether the software will not be supported anymore, or the data is needed for further analysis, which can not be performed in the application it is stored in, the inaccessibility of the data can be a good reason not to use an application. To avoid this situations and to make the MASPECTRAS future proof for the users a XML export of the sample processing data will be implemented. This will support the PSI' *.xml formats, as well as the PRIDE *.xml [21, 10-13] format.

Chapter 2

Methods

This chapter gives an insight in the methods, tools and libraries used during the thesis.

2.1 Used Standards

In this chapter the most essential standards used in the project will be introduced. Most of the standards have been defined by the Object Management Group (OMG). The OMG [34] is a non-profit consortium that produces computer industry specifications for interoperable enterprise applications.

> OMG™is an open membership, not-for-profit consortium that produces and maintains computer industry specifications for interoperable enterprise applications. Our membership includes virtually every large company in the computer industry, and hundreds of smaller ones.

2.1.1 Model Driven Architecture (MDA)

The Model Driven Architecture (MDA) [35] approach of development was introduced by the Object Management Group (OMG). The main aspect of MDA is the ability to address the whole life cycle in development, covering analysis and design, programming, testing, deployment and maintenance [26].

MDA was not a complete new development, rather it builds on well known OMG standards as Unified Modelling Language™(UML®), the Meta Object Facility (MOF™) and the Common Warehouse Metamodel (CEM™). It also includes the UML Profile for Enterprise Distributed Object Computing (EDOC), including its mapping to EJB, and the CORBA Component Model (CCM) [26].

The main focus of MDA is to decouple the design of the implementation for a specific infrastructure.

2.1.2 Unified Modeling Language (UML®)

UML®is an other standard designed by the OMG group. It was invented to model applications before coding to ensure it is flexible, scalable and modular enough to fulfill the given requirements [3].

> Modeling is the designing of software applications before coding. Modeling is an Essential Part of large software projects, and helpful to medium and even small projects as well. A model plays the analogous role in software development that blueprints and other plans (site maps, elevations, physical models) play in the building of a skyscraper. Using a model, those responsible for a software development project's success can assure themselves that business functionality is complete and correct, end-user needs are met, and program design supports requirements for scalability, robustness, security, extendability, and other characteristics, before implementation in code renders changes difficult and expensive to make. Surveys show that large software projects have a huge probability of failure - in fact, it's more likely that a large software application will fail to meet all of its requirements on time and on budget than that it will succeed. If you're running one of these projects, you need to do all you can to increase the odds for success, and modeling is the only way to visualize your design and check it against requirements before your crew starts to code.

The UML 2.0 standard defines several different diagrams (each to fulfill a specified task), divided into three main categories:

Structure Diagrams The main Diagram in this category is the Class Diagram. It defines the responsibility of the different Classes and their connection. Other Diagrams are the Object Diagram, the Component Diagram, the Composite Structure Diagram, the Package Diagram and the Deployment Diagram.

Behavior Diagrams includes the Activity Diagram, the the State Machine Diagram and the Use Case Diagram. It is used during the summarizing of the requirements to identify all the interactions done by the user or interacting systems with the new application.

Interaction Diagrams, all derived from e more general Behavior Diagram, include the Sequence Diagram, Communication Diagram, Timing Diagram, and Interaction Overview Diagram. They show how The parts of the application (e.g. Classes, Objects, Objects on different Servers) interact with each other.

2.1.3 XML Metadata Interchange (XMI)

An other standard defined by the OMG is the XML Metadata Interchange (XMI). It was designed to interchange object information with the Extensible Markup Language (XML). It can be used for several kind of objects like software (e.g. Java, C#, C++), components (e.g. Enterprise Java Beans) or databases (e.g. CWM). The most common use is the easy data interchange between different UML-based modelling tools. This is also the use it was initially purpose of XMI.

The XMI specification specifies how to

- represent objects in term of XML elements and attributes,
- link objects,
- uniquely identify objects,
- reference other objects,
- validate an XMI document using DTDs and XML Schemes.

The intention of the standard was not reached fully. There are a lot of incompatibilities between the different tools, as a lot of them use their own XMI implementations. This breaks up the information interchange. Hence, a lot of tool offer their own import/export methods to interchange information.

The first version of XMI defines the main purpose. The most recent version v2.1

2.2 Java 2 Enterprise Edition (J2EE)

Java Enterprise Edition (J2EE) is an extension to Java[1]. The Java 2 Platform, Enterprise Edition technology provides a component-based approach to the design, development, assembly, and deployment of enterprise applications [12]. It uses a multitiered distributed application model. This means that application logic is divided into components according to its function. These components can be installed on different machines corresponding to their J2EE tier they belong to.

The J2EE defines four different components:

- Client components running on the client machine.
- Web tier components running on the J2EE server.
- Business tier components running on the J2EE server.

[1] An object oriented, operating system independent programming language. For details referee to http://java.sun.com

- Enterprise information system (EIS) tier software running on the EIS/-database server.

According to the definition, there are four tiers, but J2EE applications are usually referred to as three-tiered, because mostly the web and the business tier are running under the same application server on the same machine. (see fig. 2.1).

There are some very strong arguments to develop high performance and mission critical applications with J2EE [29]:

Open standards J2EE is developed based on open standards.

Standardised technologies J2EE uses standardised technologies like JDBC, JNDI, JMS, JCA, JSP, EJBs,...

Portability there is no need to use a certain hardware or operating system other than there has to be a Java Virtual Machine (JVM) available.

Heterogeneous environment possible So it can be used in heterogeneous server environments.

Scalability On increasing load on the application it can be easily scaled by adding new servers with an application server on it. Then the components of the applications can be spread over the servers to share the load.

2.2.1 J2EE Components

J2EE applications are made out of components. These are self containing software units, interacting witch each other. There are three types of components defined by the J2EE specifications:

- Application clients and Applets. They run on the client machine.
- Java Servlets and Java Server Pages (JSP) are web components and run on the server.
- Enterprise Java Beans (EJB) components are business components (which contain the application logic) that run on the server

2.2.2 The Client Tier

The client Tier is the interface to the user of the system. It offers ways of interaction with the server application. There are two different types of clients available. The application clients and the more often used web clients.

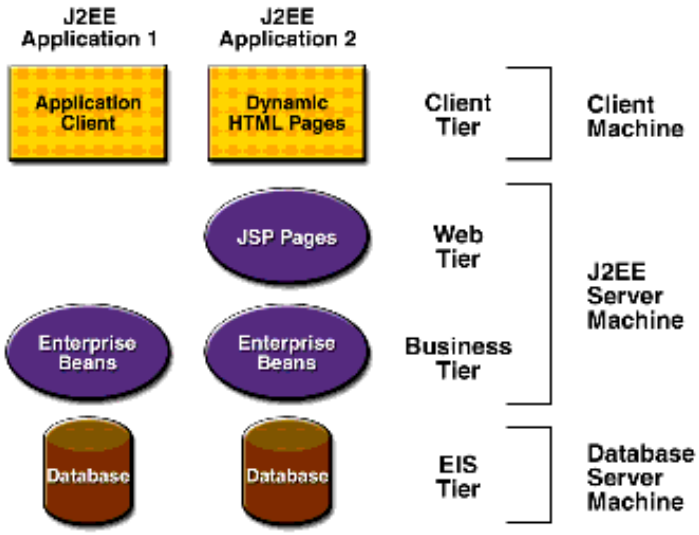


Figure 2.1: This picture shows the typical multitiered J2EE application. It consists of the ESI tier for data storage. It typically consists of an database system, the Enterprise Beans, which contain the business logic and the JSP pages for producing the HTML response for the Client tier, which then renders the HTML as user interface (taken from [12]).

Application Clients

Application clients are mainly used if the requirements are too complex to fulfill them by using a markup language. Typically there are AWT/Swing or other GUI tool kits used to build these clients. These clients access the business beans on the web tier over protocols like SOAP or HTTP [12].

Web Clients

Web clients are most likely web browsers. The only task of an web client is to show the output - most likely HTML - of the web tier and to interact with the web tier to trigger user input on the server. These clients are called thin clients, because they do not take any responsibilities other than user interactions. They show the generated information and do not communicate with databases, concern about business logic, user management and demanding issues.

2.2.3 The Web Tier

The web tier responsibility is to interact with the client tier. In J2EE applications it reacts on clients HTTP request and interacts with the business

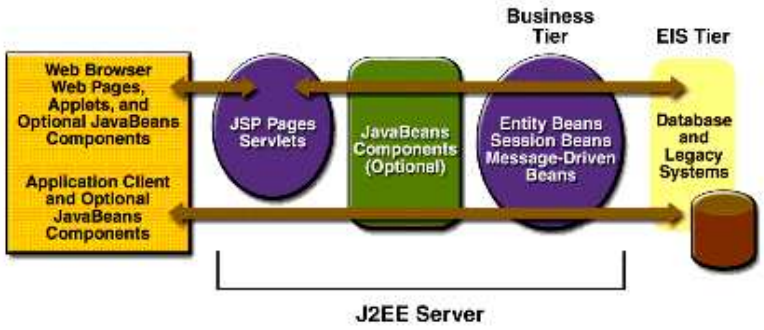


Figure 2.2: More detailed view of the three tier application logic (taken from [12])

logic to generate the response to the client tier. Typically it produces HTML or XML content, although the web tier can generate and serve any content type. The most common web components are either Servlets or Java Server Pages (JSP). But also other technologies are existing here (e.g. XSLT generated pages out of XML sources or Tapestry).

2.2.4 Application Tier

Business code solves or meets the needs of a particular business domain as banking, finance, or like in our case protein sample processing handling, is handled by enterprise beans in the business tier [12]. These enterprise beans are called business components.

Enterprise Java Beans

"Enterprise beans serve as the building blocks of distributed enterprise applications" [36]. "Enterprise beans are components which are used as parts of distributed enterprise applications. Each enterprise bean encapsulates a part of the business logic of the application" [36]. At run time, an enterprise bean resides in an EJB container. It provides the deployment and run time environment for enterprise beans, including such services as remote access using standard distributed protocols, security, transaction, deployment, concurrency, and instance life-cycle management [36].

The EJB architecture defines three types of enterprise beans:

Entity Beans Entity Beans represent information about the business domain. In the case of MASPECTRAS an sample or an experiment would be such an entity. The job of an entity bean is to hold the information and the changes happening during the processing of the requests coming from the front end to store it later to the EIS (e.g.. the database)

again. Therefore they offer interfaces for database access. They also can be handed around between the different session beans for information transfer.

Session Beans Session Beans are extensions of the client application that manage processes or tasks [8]. They are not persistent and not managed in the case of a restart of the server. Dependent on the needs you can use either stateless or statefull session beans. Stateless beans do not remember any state of a transaction. On each request they are virgin and have always the same behavior. They are light weighted and can be used by several clients in parallel. In comparison, statefull session beans remember the state of the last action. Both of them are addressed in synchronous way which means the user has to wait for them to finish until he gets the response.

Message Drive Beans Similarly, message-driven beans coordinate tasks involving other session and entity beans. Message-driven beans and session beans differ primarily in the way they are accessed. While a session bean provides a remote interface that defines which methods can be invoked, a message-driven bean listens for messages. It responds by processing the message and managing the actions that other beans take [8]. In opposite to Session Beans, Message Driven Beans are asynchronous.

2.2.5 Enterprise Information System Tier (Data Tier)

The Enterprise Information System Tier (EIS) includes information systems used in the enterprise, like database systems, enterprise resource planning (ERP) systems, user management systems, and a lot of others. The main responsibility of the EIS tier is to make the enterprise resources available to the business logic. In the MASPECTRAS application it consists mainly out of the database and the user management system. The database can be either a Oracle, Postgresql, or a MySQL database.

2.2.6 A More Detailed View of the Web Tier

Web components are typically either Servlets or JSP pages (JSPs) although there are also other technologies available. Servlets are Java classes, which are accessed by the client, and build dynamical content and sends it back to the client. Java Server Pages are written in HTML with extended functionality for dynamic content which is written down as XML tags. These tags are defined in tag libraries which can be integrated into the JSP. In fact, the J2EE server translates the JSPs to Servlets, which are executed by the server and generates HTML for the client. This means, JSP is an language to ease the design of servlets for humans.

The web tier typically performs the following functions in a J2EE application:

- manages interactions between web clients and application business logic.
- generates dynamic content in entirely arbitrary data formats, including HTML, images sound and video.
- translates HTTP GET and POST actions into a form that the business logic understands and presents results as web content.
- usually determines which page has to be to displayed next.
- has a simple, flexible mechanism for the accumulation of data for transactions and for interaction context over the lifetime of a user session.
- can implement business logic in web-only and low volume applications. In this way the separation between the presentation and the business tier is broken and it is more difficult to maintain the application. Therefore, enterprise applications normally implement the business logic in enterprise beans.

Lets have a closer look at the parts of the web tier:

Servlets

"Servlets provide a simple but powerful API for generating web pages dynamically. Although Servlets can be used for many different request-response protocols, they are predominantly used to process HTTP requests for web pages" [8] and respond by writing to an output stream.

Java Server pages (JSP)

"Java Server Pages is an extension of the Servlet component model that simplifies the process of generating HTML dynamically. JSPs essentially allow the incorporation of Java directly into an HTML page as a scripting language," [8] which is not best practice. It is better to use the set of tags defined by the JSP standard. The JSP pages are then translated and compiled into Java Servlets, which are then run in a web server just like any other Servlet.

JSP features:

Plain Text Like in HTML pages it is possible to write plain text to be displayed on the client.

HTML Also like in HTML pages pure HTML is supported, which is interpreted and rendered by the client.

Directives are instructions to tell the translator how the Servlet should be generated. Directives include the definition of additional Java classes to be imported or the definition which custom tag libraries are used. Directives have to be defined inside a '<%@ %>' Tag.

Scriptlets are sections of Java code in an JSP. If the '<% %>' tags are used, the code is directly taken over in the Servlet, if the '<%=%>' tag is used, the statement is evaluated into an expression and written to the response.

Standard actions are written by using JSP tags. These tags allow to perform certain tasks, like instantiating objects or accessing Java Beans, without requiring Java coding.

Custom tags add the possibility to create custom actions. They are the triggered by the usage of custom tags. This eases the construction of whole tag libraries to fulfill certain needs of functionality. It is easier to separate application logic from the presentation layer. The maintenance of the presentation layer is simplified afterwards.

The typical way of presenting data is to generate a Java Bean in the application to store the information encapsulated. This Java Bean is then forwarded to the JSP, which is reading the information out of the Java Bean and includes its content in the response. It has to be remarked, these Java Beans are unrelated to Enterprise Java Beans! Although they are also part of a component model, they only fulfill certain naming rules for accession of the data they store, construction (has to have an default constructor without parameters) and behaviour (the class should be serializable).

2.2.7 J2EE Patterns

"A Design Pattern is a general solution to a commonly-occurring problem. The J2EE Patterns provide a collection of tested and proven answers for J2EE-based problems. Amongst others," [5] in the following patterns had been specified:

Business Delegate decouples the business logic from the presentation layer. It delegates every call from to the *Business Delegate* to the Business Object responsible for the action to take.

Session Facade manages and abstracts the underlying business objects and provides a service layer that exposes only the required interfaces. Thus the whole functionality of an application is accessible through one bean. It decouples the client from the application and offers a well defined interface.

Transfer Objects - also known as *Value Objects* - are used to encapsulate related attributes. On one hand the entities stored in the database are represented and on the other hand the costs of remote calls are reduced, as not every attribute has to be transferred separately and produces communication overhead.

Value List Handler is used to access collection data on distant components. It avoids the direct access of every single object in the collection. Instead it uses a local copy of Transfer Objects.

2.2.8 JBoss

MASPECTRAS is hosted on a JBoss Server. This is an open source J2EE-based application server. The server includes a web server(Servlet/JSP container , and HTML server), Enterprise Java Beans (EJB) container, Java Message Service (JMS), Java Mail, and Java Transaction API/Java Transaction Service(JTA/JTS). Normally it is shipped with the Apache Tomcat Servlet engine, but there is also a version using an embedded Jetty Servlet engine.

2.3 Struts

The classical JSP approach had been to implement the logic into JSPs as Scriptlets. "This turned out to be a very dangerous way, and several projects tended to get very complex and a lot of code was recopied from JSP to JSP or it tended to end in very complex and very soon unmaintainable inclusion trees of Scriptlet pages. As solution to this problem an modified Model View Controller (MVC), the Model 2 architecture, was introduced" [25].

2.3.1 The MVC Pattern

The original MVC Pattern separates the application into the three sections:

Model The *Model* models the examined part of the reality into objects to hold the attributes. During the lifetime of the application the values of the attributes in the model can change permanently.

View The view is responsible for the data representation. To change the data, enter new data, or do some calculation on the data, the *Controller* is triggered to change the data in the *Model*.

Controller The *Controller* is responsible to deliver the data from the *Model* to the *View* and to do updates on the *Model* triggered by the *View*.

To be able to do recognize changes in the Model and to update the view immediately, the *Observer Pattern* is used. In this Design Pattern a *Listener*

registers to an observed Object. If the state of an observed object changes (e.g. an value change of an attribute), the *Listener* is notified and can take actions to fetch the changed data. The data is not sent directly, so every *Listener* can decide whether it needs an update or not, to avoid unnecessary update costs.

2.3.2 The Model 2 architecture

Web applications have to deal with the fact, that the HTTP protocol is not a push but only a pull protocol. Hence, it is not possible to implement the *Observer Pattern* in Web application without using tricks or other technologies. This is the main reason why the architecture implemented in Struts is called the Model 2 architecture. It is very similar to the MVC pattern, but does not support the automatic update of the View in case of a change in the Model [25].

Model

The way to implement the model can vary depending on which technology is used to implement the MVC pattern. In the case of J2EE applications the model most likely will be implemented in Enterprise Java Beans. Often Java Beans are generated and returned from session beans and used in the web tier. The remote call of entity beans would cause a performance overhead and result in a performance loss. These Java Beans are referred to as *data transfer objects* or *value objects* and are used in the view to build the dynamic content.

View

The most common technology to construct the view are Java Server Pages (JSPs), but a lot of other technologies are available for this part as well. Actually Java Server Faces (JSF) are the state of the art and are overtaking JSPs.

To build the view components, Struts offers a big library of additional tags. They are separated in four sections:

bean The bean tags are useful in defining new beans (in any scope) from a variety of possible sources, as well as a tag to render a particular bean (or bean property) to the output response.

html The HTML tags are used to create input forms, as well as other tags generally useful in the creation of HTML-based user interfaces. The output is HTML 4.01 compliant or XHTML 1.0 in XHTML mode.

logic The logic tags that are useful in managing conditional generation of output text, looping over object collections for repetitive generation of output text, and application flow management.

nested The nested tags extend the base Struts tags to allow them to relate to each other in a nested nature. The fundamental logic of the original tags doesn't change, except in that all references to beans and bean properties will be managed in a nested context [32].

Other essential features of Struts for building web applications are:

Automatic Form Population using the HTML tag library struts fills in the current values of the form bean into the HTML form input elements.

Form Validation of form fields input is implemented in struts. If there are validation rules are given for a form field, then the entered data is validated automatically and in case of an error, the user is forwarded to a defined page to reenter the input or to show him an error message.

Internationalisation is supported using Message Resources. Struts supports resource bundles treated like a database. It allows to request message strings for a certain language setting to be shown. To support a certain language, only a resource bundle has to be defined, that has an ISO language code suffix to its name (e.g. _de for German).

Controller

The controller is the main part of the Struts framework. It maps a request Uniform Resource Identifier (URI) to an Action Class. This Action Class then prepares everything for the JSP to be shown afterwards. Loading data from the database, doing calculations, and what more has to be done to fulfill the business rules. After preparing,again the controller class is asked to where the application should be forwarded to. The prepared information is most commonly displayed by a JSP. The advantage of this concept is to have one single point to decide, how the application logic path should look like. To ease the control of the application, the rules are defined in a central XML file, the struts-config.xml.

2.4 XDoclet

XDoclet is an open source code generation engine. It enables Attribute-Oriented Programming for java. This means that adding meta data (attributes) to your java sources gives more significance to the code. This is done in special Java Doc tags [2]. XDoclet will parse the source files and generate java source code, controlled by the meta tags. This can be used in

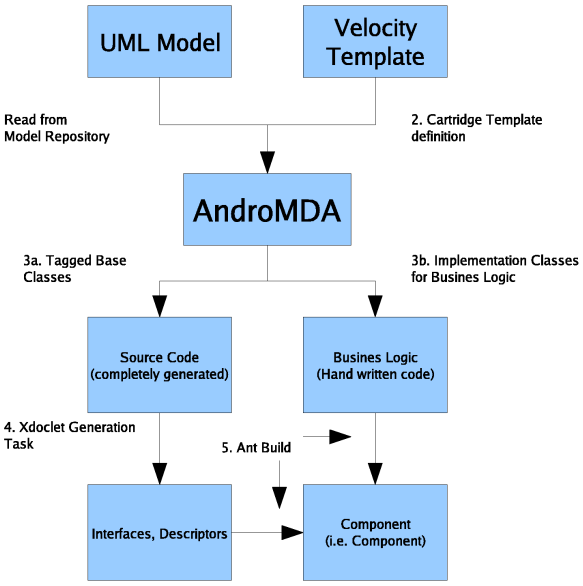


Figure 2.3: Schematic overview of AndroMDA (taken from [24])

J2EE applications, where one Enterprise Java Bean object consist typically out of seven or more files. With XDoclet only one is maintained and the rest is generated. This takes away a lot of error prune work from the developer.

2.5 AndroMDA

"AndroMDA is an extensible generator framework that adheres to the Model Driven Architecture (MDA) paradigm. Models from UML tools will be transformed into deployable components for your favorite platform (J2EE, Spring, .NET)" [33]. "Unlike other MDA tool kits, AndroMDA comes with a lots of ready-made cartridges that target today's development tool kits like Axis, jBPM, Struts, JSF, Spring and Hibernate. AndroMDA also contains a toolkit for building your own cartridges or customise existing ones - the meta cartridge. Using it, you can build a custom code generator using your favorite UML' tool" [33].

2.5.1 Cartridges

In the case of code generation for MASPECTRAS, cartridges had been written fitting the special needs for J2EE projects. These cartridges are

written as Velocity[2] templates. They interpret the UML diagram and write the code into the source files with tags for XDoclet for further processing. The cartridges for this project had been written by MSc Thomas Truskaller and had been enhanced by MSc Jürgen Hartler and MSc Robert Rader. The cartridges create among other things entity-, session, message drive beans, value objects and JSPs automatically. Take a look at picture 2.3 for a schematic overview of AndroMDA.

2.6 Tools

This section will introduce the most important tools used during the development of the MASPECTRAS application.

2.6.1 UML Tools

UML tools are software programs to create/view/edit UML diagrams to design software. On the market there are a lot of free and commercial products. For the development of the MASPECTRAS project *Magic Draw* has been used. "MagicDraw is a visual UML modeling and CASE tool with teamwork support. Designed for Business Analysts, Software Analysts, Programmers, QA Engineers, and Documentation Writers, this dynamic and versatile development tool facilitates analysis and design of Object Oriented (OO) systems and databases" [1]. The ease of use and the rich palette of features makes it to one of the most used modelling tools. Its most important features are:

- Drawing of class diagrams.
- Creation of models out of existing source from various programming languages.
- Generation of diagrams out of the class diagram (e.g. static structure, package dependency, and hierarchy).
- Team working support.
- Code generation.
- And important for this project, the support of the XMI format to feed AndroMDA with.

[2]Velocity is a script language and part of the Apache project.

2.6.2 Eclipse

Eclipse is an open source Integrated Development Environment (IDE) mainly for the Java field. What makes the Eclipse Project noteworthy is the fact, it is completely modular and extensible. It is build as an open plugin system, where everybody who has the need for, can develop a plugin. Even the basic functionality is provided by plugins. It is available for a lot of operating systems like Windows, MacOS X, Linux, Solaris, other *nixes and as source bundle to be compiled on every other system. Also a lot of Hardware Platforms are supported (e.g. x86, x86_64, ia64, SPARC, PPC, HP9000).

In this project Eclipse was chosen, because of its flexibility and extensibility. There had been used plugins for.

- editing velocity templates - Veloeclipse.
- sql editing.
- JSP editing.
- HTML editing.
- struts configuration file editing.

These editors had been part of the commercial MyEclipse Plugin.

MyEclipse

MyEclipse is a Eclipse Plugin, which is delivering rich support for Web development. It offers Editors for JSF, JSP, HTML, configuration files, XML. Hibernate development tools, database explorer, Tapestry support and a lot of other features.

2.7 Genome Usermanagement

The Genome Usermanagement [10] is a inhouse developed and used institute wide for all web applications. It was developed by Dieter Zeller and consists of a server and a client part. The server is configurable by a web front end. The client part has been implemented as custom JSP tags which integrate fully in the Struts environment. Among other features it includes the definition of users, user groups, applications, resources and access control lists. In the JSP the custom permission tags can be used to restrict parts within the page to one or a certain group of users. Alternatively other parts (for example error messages) can be displayed to users without access rights. The use of the permission tags gives you full freedom of control in your JSP. This allows the access control to data on user and on group level.

2.8 Java Prize Tags

The Prize Tags JSP Tag Library is a collection of tag libraries for Internet/intranet applications. The table below lists the tags in the Prize Tags collection [31]. The Prize Tags collection contains the following tags:

- AJAX Tags
- Tree Tag
- Tabbed Pane Tag
- Alternate Tag
- Calendar Tag
- Icon Tag
- Template Tag
- ... and several other, smaller tag libs.

In MASPECTRAS the Tree Tag was used to show the sample processing details 4.3.2.

2.8.1 Tree Tag

The Tree Tag is a JSP custom tag capable of the generation of dynamic tree controls in JSP pages, similar to the tree control found in the Windows explorer, and many other applications. The Tree Tag was designed to support the following cases [30]:

- Trees that are build dynamically, based on records in a database.
- Very large tree models, where only the visible part should be loaded.
- The information to each node is fully customisable.
- The looks of each tree node is fully customisable.

To achieve this goals the Tree Tag classes have been divided using the Model View Control (MVC) design pattern. The library consists of the JSP tags for the view and an tree model Application Programming Interface (API) to be used in the controller.

Chapter 3

Requirements and Design

For building stable and user friendly applications it is essential to first determine the user requirements which are clearly restricted to the user needs and then to make a stable design, which is based on user requirements.

3.1 Detailed Requirements

The main goal of this thesis was to extend MASPECTRAS, an application for tracking sample processing details for LC-MS/MS mass spectrometry experiments. It should give an easy way to enter and maintain the sample processing data from several places. There is also the need, to be able to publish the the information to a wide field of interested scientists, to be able to reproduce the experiments done to verify the results.

The most important requirements are:

Web based application, to ensure full flexibility in entering and maintaining the data. This is an very important point, as biologists doing the sample preparation and processing steps, as well as the mass spectrometry experiments usually have to work at different places and laboratories. So it is very important to give them easy access to the application from every computer which is connected to the Internet. As protein analysis require a lot of calculation power it is advantageous to outsource it on a high performance server. So there is no need for high performance client machines.

Full freedom in linking the experiments It is theoretical possible to use the result of every sample treatment as basis of another treatment. Hence, an kind of recursive approach of data representation and maintenance is useful. What this means in detail you can see in picture 3.1.In reality not all of the combinations will make sense, but for flexibility it is better not to limit the possibilities here.

Data export To ensure the possibility to exchange the data with other MIAPE (see 1.4.2) compliant applications, a data export is planned. This should support the PSI *.xml (see 1.4.1), as well as the PRIDE *.xml (see 1.4.1) format.

The model driven architecture gives a lot of advantages in maintaining and building software project (see 2.1.1). Therefore this approach was used for MASPECTRAS to ease the extension of the existing application with the required functionality.

3.2 Workflow of Protein MS Experiments

For a mass spectrometry experiment there is a typical work flow common. One or more samples exist for a given experiment, which has to be examined by mass spectrometry. There are loads of different proteins in a sample, therefore the sample has to be treated to separate the proteins before the mass spectrometry can be done. Otherwise the complexity of the sample would be to high and the amount of data gained from the MS experiment is too big. It would not be possible to evaluate such data. This work flow can vary very strong in the separation of the proteins. We will show now one typical but simple work flow for sample treatment:

1. Do a 2D gel electrophoresis to separate the sample. This results in a big amount of spots with only one or a small count of Proteins each.
2. Extract one spot of interest, soluble it, and do a Tryptic digestion to peptides of the protein(s) in the spot.
3. Do a mass spectrometry experiment with the digestion products.

The sample processing steps are represented in a tree structure. This gives full possibility to allow as many steps to follow the successor step without limiting flexibility. Because of this fact, an implementation with tree navigation for the sample processing data was chosen for the user interface.

3.3 UML Diagram

Based on the requirements, the modelled work flow,and the already implemented PEDRo [4] model the following UML 6.1 had been taken over from the PEDRo model. It had to be refined to fulfill the MIAPE standards.To take a look at the final UML diagram for the relational schema take a look at picture 6.1 in the Appendix.

3.3.1 Entities

In the entity UML diagram 6.1 you can see all entities which stand for the different sample treatments and related information to them describing details. The types supported for the sample annotation are:

- *Gel1d* with *Bands* as results.
- *Gel2d* with *Spots* as results.
- *LcColumn* with *Fractions* as results.
- *Chemicaltreatment* with *Treatedanalyte* as results.
- *OtheranalytePs* with *Otheranalyte* as results.

To be able to represent the recursive sample processing information, there is an self reference over two levels introduced (see picture 3.1. We can distinguish two different entity types, lets say Sample Processing Steps (SPS)) (e.g. gel 1f, gel 2d, lc column,. . .) and Sample Processing Step Results (SPSR) (e.g. band, spot, fraction,. . .). Because of the sample, as the root element of the tree, has the potential of having several sample processing steps beneath it, it can be handled like an Sample Processing Step Result (although it does not describe a processing itself).

This means in the general data structure we have an SPSR related with an SPS with a cardinality of 1:*. To allow the use of any result as basis of any other treatment there was the need of a relational entity to model a generic relation. The relations between the SPS and the SPSR is easier to handle, because there always a combination of two related elements (e.g. Gel1d and Band, or LcColumn and Fraction). This relation has also the cardinality of 1:* [4].

To encapsulate this complexity of the SPSR - SPS relation, the *Analyteprocessingstep* was introduced in the entity diagram [4]. It is a relation entity which stores the type of the upper and lower entity and the references to it.

3.3.2 Reports

ReportBeans are able to generate Reports for list pages for one entity. They support configurable amount of items, page turning, and free definable filtering, and sorting. The reports are generated directly in the database to avoid unnecessary network traffic [20]. This ReportBeans had been developed by the team of the Institute for Genomics and Bioinformatics of the Technical University Graz.

To generate a ReportBean for a certain entity, they are modeled in the UML diagram and then generated by the responsible Velocity scripts of AndroMDA. In the figure 3.2 there is shown the model for one ReportBean as example.

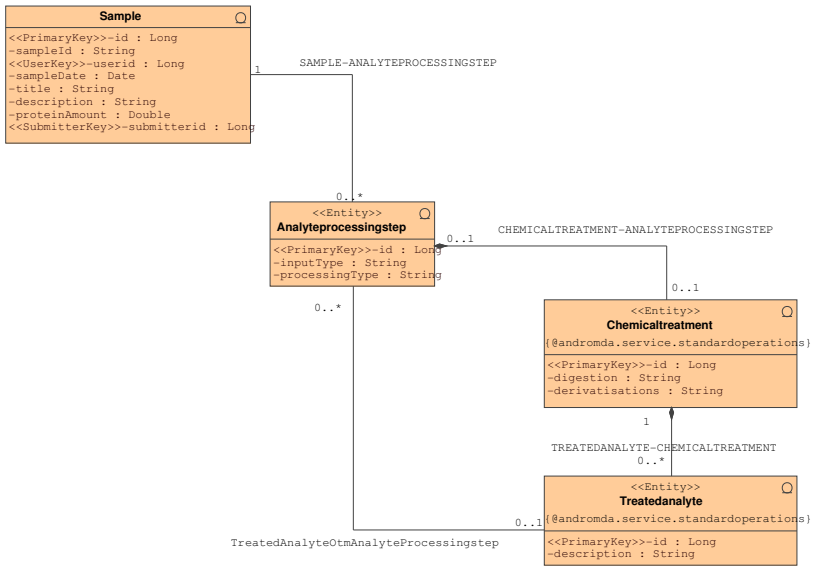


Figure 3.1: In the sample processing annotation entity model there are two different kind of entities under the sample. For every sample there can be done several kind of treatments (e.g. Gel 1D, Gel 2D, Liquid Chromatography,. . .). Lets call this treatments Sample Processing Steps (SPS). Each of this steps can have several pieces of result information stored related to them. This are bands for gel 1ds, spots for gel 2ds, and Fractions for LcColumns. Lets call them Sample Processing Steps Results (SPSR). Every of this SPSR entities can be basis for the next processing step. This means every type of SPSR can be connected with every type of SPS. For a better overview only one type of the SPS and SPSR are shown in this picture. To ease the modeling of this relation the *Analyteprocessingstep* was introduced in the PEDRo model. It is able to store the input type, the processing type (corresponds to the output type), and references to the instances of them.

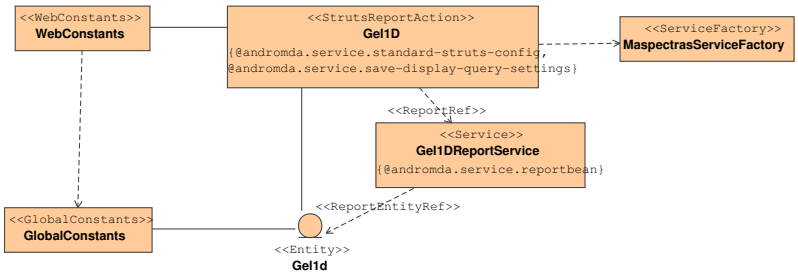


Figure 3.2: The full model of an ReportBean for one example entity.

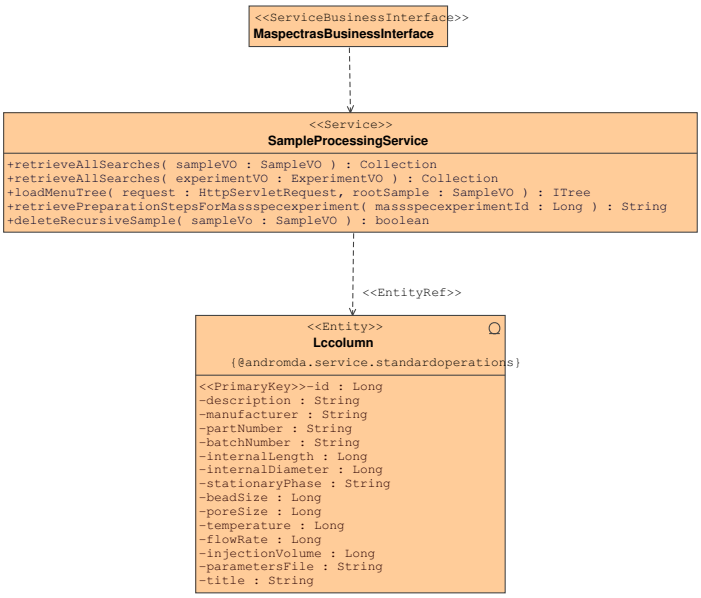


Figure 3.3: The *SampleProcessingService* offers Create/Update/Delete (CRUD) methods for all the entities it has a ≪*EntityReference*≫ to. Additional it offers data access methods for time critical data access.

3.3.3 Services

In J2EE applications, every Enterprise Java Bean (EJB) offers database access methods via their Utility class. The access of them is not straightforward. To avoid problems by using this methods, the *Servcie* classes are generated by AndroMDA. For example the *SampleProcessingService* (see picture 3.3) was designed to offer methods to access the Util classes of the EJBs and hide the complexity. It offers Create Read Update Delete (CRUD) methods for all entities it is associated with.

3.4 Schema Redesign for MIAPE Compliance

3.4.1 Substandards Taken Into Account

The MIAPE standard is a set of different sub standards covering the different areas of proteomics. There is the basic Functional Genomics Experiment (FuGE) object model, which is used as a basis for other standards. For the sample processing part of MASPECTRAS the following standards had been taken into account [23]:

FuGE The basic Functional Genomics Experiment object model.

Sample Processing There are three standards existing to describe sample processing, all covered by the sample processing Markup Language (spML).

- **MIAPE CC** is covering reporting requirements for column chromatography.
- **MIAPE CE** is covering reporting requirements for capillary electrophoresis.
- **MIAPE SP** is covering reporting requirements for sample preparation and handling.

MIAPE GE reporting requirements for gel electrophoresis covered by Gel Markup Language (GelML).

To fulfill all of the basic MIAPE standards, all attributes out of the spML and the GelML data format had been checked against the implemented PEDRo schema. It turned out that in the current version the spML is nicely covered by the PEDRo schema, but for the GelML some adaptions have to be done.

3.4.2 Necessary adaptions

Localisation of Spots and Bands

The first necessary adaption was the ability of localising bands and spots on the gel pictures. For that there are three approaches to consider to fulfill the GelML standards:

Circles To localise spots *circles* are introduced in the schema. They store the X/Y coordinates of the circle center and the radius.

Rectangles To localise bands on gels the *rectangles* are introduced. They store the X/Y coordinates of the upper left corner and the x and y size in pixel.

Boundary point "*BoundaryPoint* can be used to define an irregularly shaped location, giving X/Y coordinates of all the points surrounding a location" [22].

Boundary chains "*BoundaryChain* is an alternative method for specifying the outline of a location on an image. The X/Y coordinates specify the starting point and then a chain is described consisting of direction/step pairs. The direction is given as an integer from 0-7 corresponding to the angle (E=0, NE=1, N=2 and so on). The step is also given as integer for the number of pixels traversed in the same direction" [22].

Additional Entities

Inter Dimension Action 2D Gel electrophoresis is done in two steps. One for each dimension. To be able to describe treatments to the gel done between this two steps the *InterDimensionAction* was introduced in the schema. It stores a protocol which describes the treatment.

Buffer Description To describe the buffer attributes used during the gel electrophoresis the tables *GelBuffer* and *Buffer* had been added.

Gel Separation Method The gel separation method describes how the separation is done, which physicochemical separation range the gel has, which dimension is described, the property range, and the distribution.

Gel Substance Each gel consists of several substances. Do describe this the *Gelsubstance* was introduces. It stores the name of the substance by a controlled vocabulary and whether it is a main substance or not.

Chapter 4

Implementation

The implementation of the sample processing interface was done in the typical two step approach of the model drive development. In the first step the code, configuration files, and JSPs as basic structure are generated. In the second step, the generated code, configuration files, and the JSPs are modified and adopted to cover the design.

The section 4.1 will explain the generation with AndroMDA and the most important functionality generated.

The sections 4.3 will explain the adaptions which had been necessary for the detail pages.

4.1 Standard JSPs and Server Side Elements Generated by AndroMDA

MASPECTRAS is designed as J2EE web based application. For the web tier JSP with Struts as application navigation layer has been chosen to stay consistent with the existing project. Also the already existing AndroMDA cartridges are generating JSP pages. For the persistence layer in production an Oracle database is used, but can be changed by a simple configuration to PostgreSQL or MySQL. The MASPECTRAS application itself is deployed on a JBoss application server.

The implementation of new functionality is done in a two step approach. First, the UML model is extended or modified and code is generated with AndroMDA. This steps includes the generation of:

- the Value Objects.
- the J2EE Interfaces which are part of the Java Enterprise Beans (e.g. LocalHome interface, RemoteHome interface,...).
- JSPs for detail pages and report pages for the entities.

- the Struts Action and Form classes for request handling and data handling (see 2.3.2).
- standard mappings in the struts-config.xml for application navigation configuration.
- entries in Application resource properties for English implementation (for internationalisation).

In the second step the developer customizes the generated classes and creates new ones to add or modify the functionality. Not all of the source generated initially is overwritten on every generation run, only parts of it to ensure manual changed code is not lost. Due to this, it is possible to refine the UML diagram later on and regenerate without loosing customisations. The extension of the generated classes with user modified code is done by inheritance. The super class is generated and the sub class is doing the specific implementation by overwriting methods.

4.1.1 Reports

Beside showing single entries it is important to view lists of available entries of an entity. As this is the recurring need for this lists, report functionality was implemented in the AndroMDA cartridges to generate them [20]. The standard functionality consists of configurable amount of items, page turning, and free definable filtering, and sorting of the items in the list. A report consists out of a JSP, a Struts Action Class and a Struts Form Class, the standard struts navigation configuration, and a Session Bean.

As we will see later on, in the case of the sample processing data, not all of the features will be needed directly because of the amount of items in the lists will not be that high. But nevertheless the functionality will remain in the Struts Action classes to be available later on if it becomes necessary later.

4.1.2 Generated JSPs

During the code generation, two JSPs are created for every entity modeled with the report functionality:

Detail JSP The detail pages are used for creation, viewing, editing, and deleting of single entity entities.

List JSP The list pages are able to show a widely configurable report of all existing objects of one entity. For the full feature list of the reports see 4.1.1.

These Pages are of big use for further development, but have to be reworked heavily to implement the tree view.

4.1.3 Struts Action and Form

For each entity with modelled report functionality there is a Struts Action and Form class generated.

Action Classes

The Action class is derived from the *LookupDispatchAction* class which allows to map different methods configured by a configuration file. This allows to pack different functionality for one entity in one class.

Every generated action class supports a set of functions for retrieving entity information from EIS layer for detail and list view. In detail view retrieving, editing, deleting entity information, and retrieving lists for pull down boxes is handled. For list view loading of entity elements, sorting them, page turning for long lists, display configuration (show or hide certain columns of the list), filtering of the information in the list is done by the Action class.

The most important ones are:

createEdit This method is called for creation, editing, and deletion of entity elements in the data base.

viewEdit It is used to view retrieve the entity elements value and the lists to be shown in the detail JSP.

findAll It creates a list of entity elements of one entity owned by an user or group.

display Shows the "Edit Display Settings" column selection table in the detail JSP.

hide Hides the "Edit Display Settings" column selection table in the detail JSP.

Form class

The form class derives from the *ActionForm* class and is for data feed, editing, validation, and storing display settings for list view of entities. Each form class consists of:

- an attribute for every entity attribute and it's related getter and setter methods.
- a validation method which checks the values of the form to fulfill the input requirements.
- an attribute for every data attribute (with the related getter and setter methods) which tells the JSPS whether it should be shown in the JSPs list view or not.

4.2 MIAPE Compliance

To fulfill the MIAPE compliance the UML diagram was extended with the following entities:

- *InterDimensionAction*
- *GelBuffer*
- *Buffer*
- *GelSeparationMethod*
- *GelSubstance*
- *Circle*
- *Rectangle*
- *BoundaryChain*
- *BoundaryPoint*

The entity classes, database tables, and related J2EE classes had been generated by AndroMDA. Also the standard JSPs for detail and list view had been generated by modeling the report functionality in the UML diagram. The report classes had been deleted out of the model again, as the report functionality is not used for this entities. Only the JSPs had been preserved for manual inclusion in the sample processing detail pages.

4.3 Sample Processing Data Web Interface Implementation

To view, edit and delete the sample processing data the standard JSPs and list JSPs together with the report classes can not be used as they are generated. Customisation is necessary to be able to include them into the application because of the structure of the recursive sample processing data hierarchy.

4.3.1 Structure of the Data Presentation

As explained already in the design section (see 3.3.1) the sample processing data is structured in two levels, where they are related in a 1:* cardinality. For this reason and the fact of limited count of SPSR objects per SPS object, there was made the decision to generate one single JSP to show the details of an SPS object and a list with the SPSR objects linked with the SPS object.

As mentioned already, because of this recursive schema the data navigation was designed as tree model.

For the presentation of the data a framset consisting of two frames was chosen. In the left small frame the navigation tree is displayed and in the right frame the detail information of the node is shown.

4.3.2 Realisation with PrizeTags Tree Tag

There are several different public available tree libraries for JSPs available. The decision to use the Jenkov Prize Tags Tree Tag [17] was done, because it is based on a very flexible object model in background. This gives the possibility to create the whole tree model on the server and then forward it to the JSP. The JSP contains tree tags, which render the tree based on the model. For flexible use there are features like event listeners and the view of the tree is completely freely definable.

The Tree Model

To demonstrate the possibilities of the tree model a short example code snippet is given (see 4.1).

Listing 4.1: Creation of an example tree object model

```
Itree tree = new Tree();
                //(node id, node name, node type)
ITreeNode root          =
        new TreeNode("1","Sample","sample_type");
ITreeNode gel1d         =
        new TreeNode("2","Gel_1D","gel1d_type");
ITreeNode lcColumn      =
        new TreeNode("3","Lc_Column","lc_column_type");

root.addChild(gel1d);
root.addChild(lcColumn);
tree.seatRoot(root);

session.setAttribute("tree.model", tree);
```

Of course in MASPECTRAS the tree is not generated hard coded like in the example. There is a method in the *SampleProcessingServiceBean* which loads the tree. This Method is called by the *showTree* method of the *ProcessingStepAction* class, which is handling the SPSR entities and the tree. As the expected tree is not very big, the whole tree is loaded when it has to be shown first time together with the value objects of the sample processing entities (see picture 4.1). This means to every tree node the related value object is linked over an information object, the *AnalyteStepTreeStore*

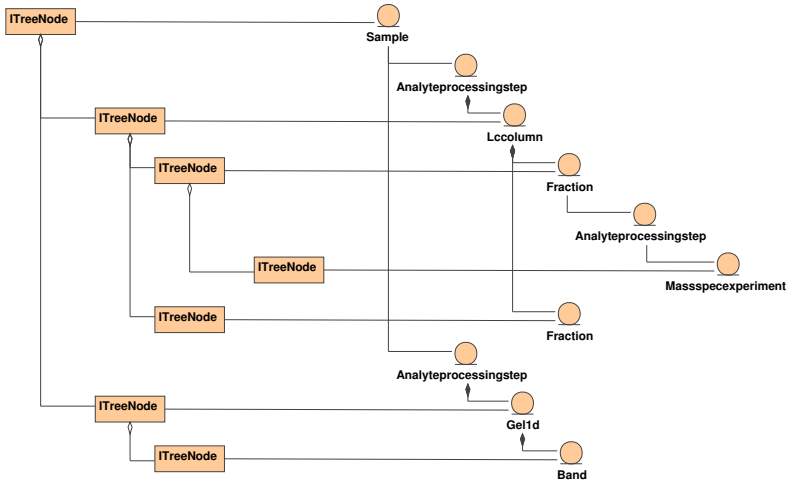


Figure 4.1: The tree model on the left side of the picture is build by *TreeNode* classes, which implement the *ITreeNode* interface. The sample processing data tree on the right side shows some sample treatments. For each of the sample processing step entities, there is one related tree node existing. Only the *AnalyteprocessingstepsVO* do not need a tree node, as they are relational objects not representing data. The linking class *AnalyteStepTreeStore* is omitted in this diagram for simplicity.

object. This object stores additional information, like ids, forwarding paths for the application to show the related detail page in the right frame, and some more. As the AndroMDA cartridges do not support more complex navigation logic the forwarding path is generated on creation of the tree model by the *loadMenuTree* method and stored in the *AnalyteStepTreeStore* object.Finally the whole tree model is stored in the HTTP session to be accessible in the related JSP.

For deleting sample processing steps from any position in the tree, a recursive deletion method was implemented. It iterates down to the leafs of the tree and deletes the sample processing steps and the related tree nodes. Only the mass spectrometry sample processing steps are not deleted, because they are reachable through other views and have to be kept.

The Tree JSP

After the tree has been generated it is rendered by the SampleDetailTree.jsp. It is shown in the left frame of the frameset. The tree tag in the page is iterating over the whole tree model and renders the open nodes of the tree depending on subtags. This subtags are trying to match against the state of a tree node. If the conditions (e.g. open, selected, last in branch,. . .) match,

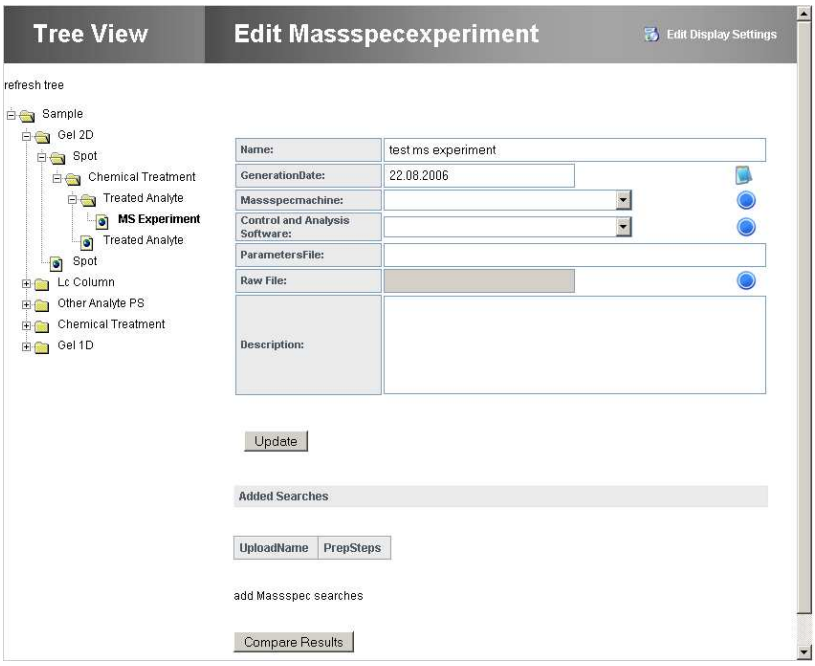


Figure 4.2: The tree JSP in the left frame is responsible to show the navigation tree. In the right frame the details of an mass spectrometry experiment are shown. On this page as SPSR elements mass spectrometry database searches can be added. The page in the right frame was not part of these thesis but developed by MSc Jürgen Hartler.

it renders the HTML in the tag body. For navigation the JSP contains a selection listener, which is triggered every time a node of the tree is selected. This selection listener triggers then the loading of the detail page in the right frame. For an example see picture 4.2.

4.3.3 The Detail Pages

Because of the same cardinality structure of the relations between all of the sample processing entities, the detail pages are build in the same design for all of the SPS entities. For all the SPSR entity detail pages and for the sample the same JSP is used. As there was the need for reuse of the entity lists on several places the HTML table of the generated list JSPs had been extracted into an own List JSP for every entity (see picture 4.4).

The Sample and SPSR Detail Page

The Sample JSP and SPS JSP have a navigation and creation functionality for SPS elements. It is a modular page which includes a list for each possible

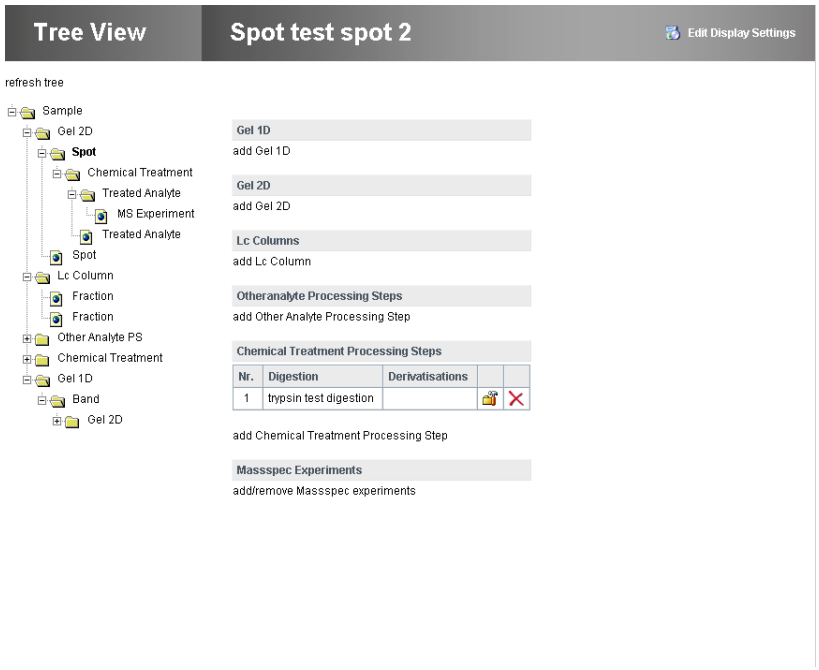


Figure 4.3: The Detail page of SPSR elements and for the Sample. In this case shown for a Spot. For every possible SPS entity type there is a list of the existing samples below the actual shown one and a link for generating new ones.

SPS type to appear under a SPSR element which is shown if at least one of the type is existing. For every entity therefore the List JSP has been included. For creation of new sub elements a link for every type is existing.

To build this page, the extracted tables from the list JSPs generated by AndroMDA are included into one JSP. This JSP is dynamically showing the name of the displayed processing step in the title line and offers configuration of the viewed columns for all of the lists. This is done with a menu, which is displayed after selecting the "Edit Display Settings".

Navigation by clicking one of the links in the list shows the details of the entity entry. Selecting one of the *create* links on the page,the detail page for the related SPS entity is shown with empty attribute fields for entering data to create a new entry. In both cases the same JSP is shown. Attributes in the session activate the needed elements in the JSP (see picture 4.3).

The SPS Detail Page

As an other logical consequence of the data model structure and the requirement to see the SPS details and an list of the SPSR data at once, the detail page generated by AndroMDA was modified. Therefore the List JSP for the

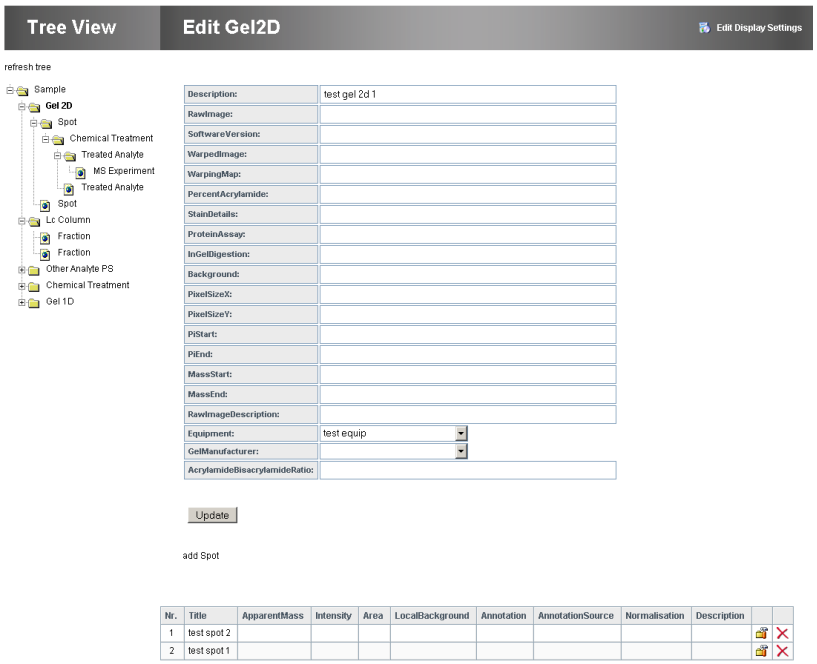


Figure 4.4: As example for an Sample Processing Step SPS entity the gel 2D detail page with the included spot list is shown. The list of the Sample Processing Step Results SPSR, in this case Spots, are included JSPs.

SPSR entity was included into the related SPS JSP (e.g. SpotList.jsp into Gel2d.jsp).

In the first version of the Detail pages page turning, sorting and filtering of the SPSR lists is not used, as there are not that many entries expected per SPS (see picture 4.4).

For the implementation of the detail pages of the SPS the generated detail page has been modified to show together with the SPS details the list of related SPSR entries. Therefore the generated table has been extracted from the SPSR JSP together with the menu for column selection into a separate JSP. This one is then included in the SPS table.

Modifications of the Action Classes for the Tree View

Due to the need of additional handling of the tree model in action classes several adoptions had been necessary. The generated methods have been reworked to deal with the tree node objects of the tree model. Therefore on every creation of a new sample processing step a new tree node is generated to build the parallel tree model as explained (see 4.1). This tree nodes has to be deleted as well on sample processing step deletion. For SPS entity objects

also the *AnalyteprocessingstepVO* has to be taken in account on creation and deletion.

The *prepareForTreeViewEdit* method has been added to the SPS Action classes. This method loads the list of the SPSR entities for the SPS entity (e.g. *Fractions* for *LcColumns*) . Then it forwards to the viewEdit method to load the SPS detail information.

Chapter 5

Discussion

The goal of this thesis was the extension of the MASPECTRAS application. The main tasks of the extension had been to ensure the MIAPE conformance of the implemented PEDRo database schema and the implementation of an web frontend.

The MIAPE conformance of the MASPECTRAS application will ensure the possibility of reproduction of protein mass spectrometry experiments. This will give big value to proteomics scientists using it for documenting their experiments.

MASPECTRAS is implemented as J2EE web based application. Regarding the requirements of the sample processing data web interface relatively little amount of data is entered. Hence, fast persistence is no issue in this part of the application Container Managed Persistence (CMP) is used.

5.1 XML Sample Processing Data Export

The data export of the sample processing data, which originally would have been part of this thesis, finally had been done by Robert Rader, an member of the Institute of Genomics and Bioinformatics of the Technical University Graz.

There was published a paper by MSc Jürgen Hartler et al. about MASPECTRAS in the terminal phase of this thesis. As MASPECTRAS is the first proteomics platform fulfilling the MIAPE standard and the sample processing data export is an interesting feature, the data export got high priority to be finished before the paper was released. To be able to report about the export functionality and the limited time frame for finishing this thesis had been the reason to hand over the work.

5.2 Improvements and Future Development

Due to the close time limit, the web interface of several side tables had not been implemented yet. To fully support the MIAPE standard it will be the next step to implement the interface for following tables:

- Interdiemnsionaction
- Gelbuffer
- Buffer
- GeSseparationMethod
- Digegel
- GeSubstance
- Mobilephasecomponent
- Gradientstep
- Percentx
- Assaydatapoint
- The location information tables: Circle, Rectangle, Boundarypoint, Boundarychain.
-

The implementation of these missing table will be easier than for the tables in the tree structure, because the generated JSPs and code can be used directly and has not to be customized that heavily like in the tree view. Also the navigation is done in the same frame, which simplifies the application navigation logic.

For future versions there is the possibility to implement the sorting and filtering of the SPSR lists in the tree view. This can turn out to be interesting in case of use of more SPSR elements than expected in the design phase.

As the MIAPE standard is still under development, many changes in it will upraise in future. This means ongoing updating and refinement of the MASPECTRAS application to stay in phase with the standard. Because of the MDA approach of the development this can be done relatively easy.

Chapter 6

Appendix

6.1 UML Diagrams

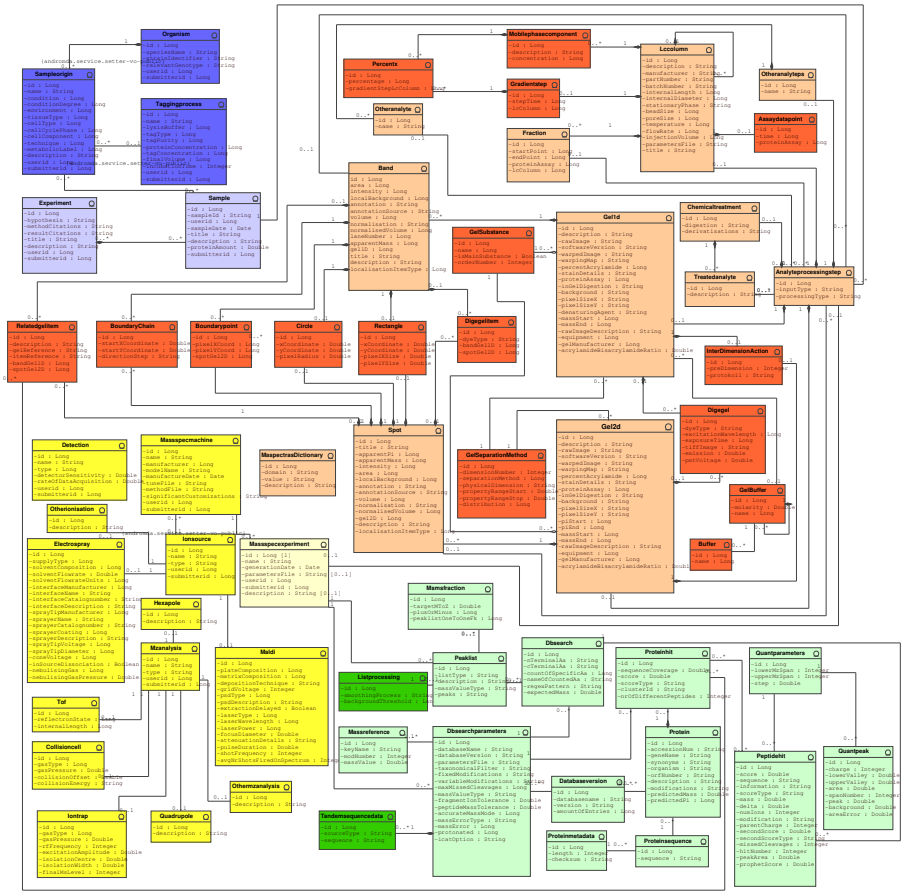


Figure 6.1: The UML diagram of the schema of MASPECTRAS. The blue part is describing the sample generation. The yellow part is describing the mass spectrometry. The green part is describing the MS results analysis. The pink and red part is describing the sample processing, which is the most important part for this thesis. The entities colored in pink are the main entities describing the sample treatment and the result of the treatments. The entities colored in red are holding the additional information for the main entities.

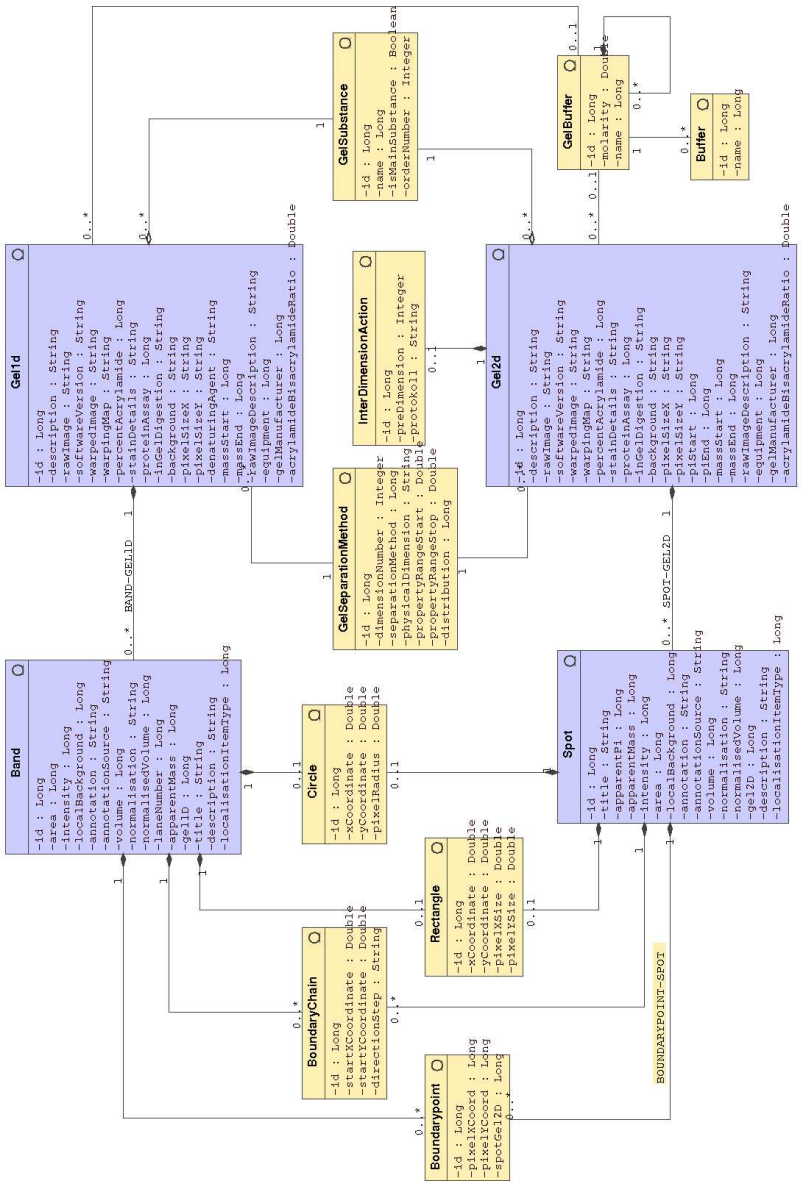


Figure 6.2: The redesigned sample processing UML model for MIAPE compliance. For explanation see 3.4. Only the affected classes are shown in this diagram.

Bibliography

[1] *Introducing MagicDraw*. URL: http://www.magicdraw.com.

[2] *Welcome! What is XDoclet?*. URL: http://xdoclet.sourceforge.net/xdoclet/index.html.

[3] *Introduction to OMG's Unified Modeling Language*. URL: http://www.omg.org/gettingstarted/what_is_uml.htm, July 2005.

[4] AL., C. F. T. ET: *A systematic approach to modeling, capturing and disseminationg proteomics experimental data*. Nature Biotechnology, 21:247–254, March 2003.

[5] ALUR, D., D. MALKS and J. CRUPI: *Designing Enterprise Applications–with the Java 2 Platform, Enterprise Edition*. Prentice Hall PTR, New Jersey, 2001.

[6] ARDREY, R. E.: *Liquid Chromatography Mass Spectrometry*. John Wiley and Sons, 2003.

[7] B. ALBERTS, D. BRAY, A. J. J. L. M. R. K. R. P. W.: *Lehrbuch der Molekularen Zellbiologie*. Wiley-vch, Second edition ed., 1998.

[8] BILLÂ BURKE, SACHAÂ LABOUREY, R.-H.: *Enterprise JavaBeans, 4th Edition*. O'Reilly, June 2004. SBN: 0-596-00530-X.

[9] COLINGE, J.: *Computational Proteomics*, 2005. Script for 'Computational Proteomics' for at the University of Applied Sience Hagenberg.

[10] E., Z.: *Design and Development of a User Management System for Molecular Biology, Master Thesis*. Master's thesis, Graz University of Technology, 2003.

[11] EDMOND DE HOFFMANN, V. S.: *Mass Spectrometry*. John Wiley and Sons, Oktober 2001. ISBN 0471485667.

[12] ERIC ARMSTRONG, JENNIFER BALL, S. B. D. B. C. I. E. D. G. K. H. E. J.: *The J2EE 1.4 Tutorial*. URL: http://java.sun.com/j2ee/1.4/docs/tutorial/doc/, 2001.

[13] Garfin, D. E.: *Two-dimensional gel electrophoresis: an overview*. Trends in Analytical Chemistry, Vol. 22(No. 5):11, 2003.

[14] Heidcamp, W. H.: *Lap Book*. url, URL-http://homepages.gac.edu/ cellab/index-1.html.

[15] Hodges, W. A.: *Characterization of the Recombinant Human Factor VIII Expressed in the Milk of Transgenic Swine*. Master's thesis, Virginia Polytechnic Institute and State University, 2001.

[16] Hofmann, A.: *Automatic annotation of mass spectra by example of glycosylated proteins*. Master's thesis, Center for Bioinformatics, Saarland University, 2005.

[17] Jenkov: *Jenkov Prize Tag - Tree Tag*. URL: http://www.jenkov.com.

[18] Jürgen Hartler, Gerhard G. Thallinger, G. S. A. S. T. R. B. E. K. A. S. R. R. K. M. and Z. Trajanoski: *MASPECTRAS: a plattform for management and analysis of proteomic LC-MS/MS data*. p. 17, 2006. not yet published.

[19] Meyer, V. R.: *Practical High-Perfomance Liquid Chromatogrphy*. John Wiley and Sons, 2004.

[20] Molidor, R.: *DESIGN AND DEVELOPMENT OF A BIOINFORMATICS PLATFORM FOR CANCER IMMUNOGENOMICS*. PhD thesis, TU-Graz, 2004.

[21] Ochard, S., H. H. a. R.: *The proteomics standards initiative*. Proteomics, 3:1374–1376, 2003.

[22] PSI GPS Working Group: Andrew Jones, University of Manchester; Frank Gibson, N. U. C. T. E. N. W. P. U. o. M.: *gelML: Gel Markup Language*. URL: http://psidev.sourceforge.net/gel/, Dezember 2005. Draft.

[23] PSI GPS Working Group: Norman W. Paton, Andrew Jones, C. T.: *spML: Sample Processing Markup Language*. URL: http://psidev.sourceforge.net/sp/, 2005.

[24] T., T.: *Data integration into a Gene Expression Database*. Master's thesis, Graz University of Technology, 2003.

[25] Ted Husted, Cedric Dumoulin, G. F. D. W.: *Struts in Action*. Manning, April 2003. ISBN 1-930110-50-2.

[26] Truyen, F.: *The Fast Guide to Model Driven Architecture, The Basics of Model Driven Architecture*. URL: http://www.omg.org/mda/presentations.htm, January 2006. Whit Paper.

[27] TUT, A. E: *Structure of Proteins*. URL: http://www.ausetute.com.au/proteins.html.

[28] UNKNOWN AUTHOR: *Concept 1: Overview: The Central Dogma*. URL: http://www.phschool.com/science/biology_place/biocoach/transcription/overview.html.

[29] UNKNOWN AUTHOR: *J2EE - Was ist denn das?*. URL: http://www.j2ee-develop.de/basics/.

[30] UNKNOWN AUTHOR: *Price Tags: Tree Tag user Guide*. URL: http://www.jenkov.com/prizetags/documentation.tmpl.

[31] UNKNOWN AUTHOR: *Prize Tags - Introduction*. url, URL-http://www.jenkov.com/prizetags/introduction.tmpl.

[32] UNKNOWN AUTHOR: *Welcome to Struts Taglib*. URL: http://struts.apache.org/1.x/struts-taglib/index.html.

[33] UNKNOWN AUTHOR: *What is AndroMDA?*. URL: http://galaxy.andromda.org/index.php?option=com_content&task=blogcategory&id=0&Itemid=42.

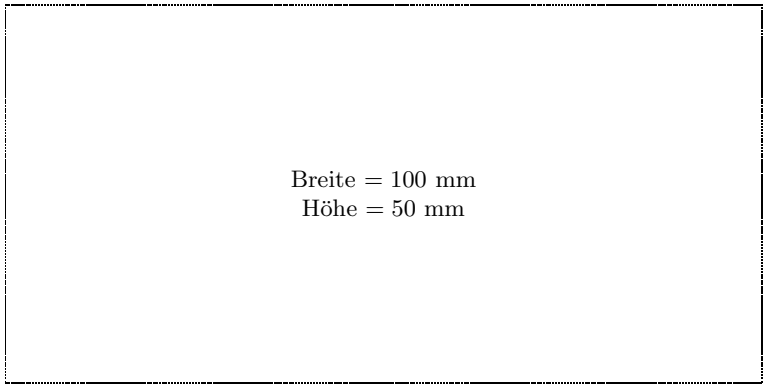
[34] UNKNOWN AUTHOR. URL: http://www.omg.org, August 2006.

[35] UNKNOWN AUTHOR: *OMG Model Driven Architecture*. URL: http://www.omg.org/mda, August 2006.

[36] VLADAÂ MATENA, SANJEEVÂ KRISHNAN, L. B.: *Applying Enterprise JavaBeans: Component-Based Development for the J2EE Platform*. Addison Wesley, Second Edition ed., May 2003. ISBN: 0-201-91466-2.

Messbox zur Druckkontrolle

— Druckgröße kontrollieren! —



— Diese Seite nach dem Druck entfernen! —