Robert Rader

Master Thesis

Design and Development of a Database for Protein-Protein Interaction in Crystals

Institute of Genomics and Bioinformatics,
Graz University of Technology
Petersgasse 14, 8010 Graz, Austria
Head: Univ.-Prof. Dipl.-Ing. Dr.techn. Zlatko Trajanoski

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

Supervisor:
Dipl. Ing. Gerhard Thallinger

Graz, December 2005

Abstract

German

Das Kristallisieren von Proteinen zur Röntgenstrukturanalyse kann ein langwierig und aufwändiger Prozess sein. Viele Parameter, wie der pH Wert oder die Temperatur und Konzentration von Zusatzstoffen, werden variiert, um einen Kristall zu züchten. Die hier entwickelte Applikation erlaubt es, für Proteine mit bereits bekannter Struktur Oberflächeneigenschaften und Protein-Protein Kristallkontakte zu berechnen, zu speichern, zu betrachten und zu analysieren, mit dem Ziel ein besseres Verständnis für Protein Kontakte und ihrer kristallographischen und biologischen Funktion zu erlangen. Dieses Verständnis kann in weiterer Folge einerseits dazu benutzt werden, Protein Mutationen zu konzipieren, die bessere Kristallisationseigenschaften aufweisen und andererseits die Suche nach biologischen Protein-Protein Kontakten zu unterstüzten.

Bei der entwickelten Software handelt es sich um eine drei schichtige Java Enterprise Applikation, die mit Hilfe von Model Driven Architecture erstellt wurde. Das Datenbankschema basiert auf dem der Protein Data Bank. Erweiterungen am Schema erlauben das Speichern von Oberflächen, Kontaktregionen und ihrer Eigenschaften, wie der Hydrophobizität oder des elektrostatischen Potenzials. Es wurde ein detailliertes Interface zum Integrieren von Import-, Exportfunktionen, Berechnungen und Analysen in die Applikation definiert. Die volle Funktionalität der Anwendung ist über ein Web Interface verfügbar, das auf dem Model-View-Controller Framework Struts basiert. Als Datanbank wurde Oracle verwendet und als Applikationsserver JBoss.

Stichwörter: Protein-Protein Kontakte, Kristallkontakte, Datenbank, J2EE

Abstract

English

The crystallisation of proteins for X-Ray diffraction can be a long lasting and exhausting task. Usually many parameters like the pH, the temperature and the concentration of the additives are varied to grow a crystal. The here developed application allows to calculate, store, view and analyse surface attributes and protein-protein crystal contacts for proteins with a known structure and should lead to a better understanding of protein contacts and their crystallographic and biological function. This understanding can later be used for the design of protein mutants with better crystallisation premises or the search for biological protein-protein interaction partners.

The three tiered Java enterprise application was developed using model driven architecture. The database schema is based on the Protein Data Bank. It has been enhanced by tables to store protein surfaces, contact patches and various properties including hydrophobicity and electrostatic potential of surface points and contact regions. Detailed interfaces and an underlaying task management have been designed for import functions, calculations and analyses. All business methods can be accessed through a Web interface based on the Model-View-Controller framework Struts. Oracle was used as database server and JBoss as application server.

Keywords: protein-protein interaction, crystal contacts, database, J2EE

Contents

Figures, Tables and Listings

List of Figures

List of Tables

Listings

Glossary

API Application Programming Interface defines how an application accesses the functionality of an operating environment or another application.

Asymmetric Unit is a crystallographic term for the smallest piece of a crystal that gets translated to form the unit cell.

Biological Molecule see biological unit.

Biological Unit is the macromolecule, that has been shown to be or is believed to be functional.

CASE Computer Aided Software Engineering

CGI Common Gateway Interface is a standard for communication between external applications and information servers, such as HTTP or Web servers.

CMP Container Managed Persistence describes an Enterprise JavaBean, whose persistence is managed by the underlaying container.

CORBA Common Object Request Broker Architecture is the OMG vendor-independent architecture and infrastructure, which computer applications use to work together over networks.

CWM Common Warehouse MetaModel is an extensive OMG standard meta-model for data warehousing, data mining, data modelling and other aspects of data storage and manipulation.

Database Schema is the logical and physical definition of a database structure.

DBMS Data Based Management System manages the data files in the database and may provide independence of programs and data.

Design Pattern is a reusable, tested and proven to function solution for a reoccurring problem in software development.

EBI European Bioinfomatics Institute

EIS Enterprise Information System

EJB Enterprise JavaBean is a server-side component of the J2EE platform that encapsulates the business logic of an application.

EMBL European Molecular Biology Laboratory

ERP Enterprise Resource Planning describes a business management system that integrates many aspects of the business, including planning, manufacturing, sales, and marketing.

Functional Unit see biological unit.

GUI Graphical User Interface

Hetero Component in a protein structure describes an atom belonging to a small molecule cofactor rather than to a polymer chain.

IUCr International Union of Crystallography

IUPAC International Union of Pure and Applied Chemistry

J2EE Java 2 Platform, Enterprise Edition defines a platform for the development and deployment of multi-tiered, web-based enterprise applications.

JDBC Java Database Connectivity is the standard API for accessing relational databases.

JNDI Java Naming Directory Interface is the standard API for multiple naming and directory services.

JMS Java Messaging Service is the standard API for sending and receiving messages.

JSP JavaServer Pages are text pages using a combination of HTML or XML tags, JSP tags, and code written in the Java programming language.

mmCIF Macromolecular Crystallographic Information is a file format to store macromolecular structure data.

MDA Model Driven Architecture is an OMG standard for system development, that provides a means for using models to direct the course of understanding, design, construction, deployment, operation, maintenance and modification.

Metamodel is a special kind of model that specifies the abstract syntax of a modelling language.

MOF Meta-Object Facility is an OMG specification standardizing a metamodel and repository for metadata.

MVC Model View Controller is a design pattern separating business logic from presentation and request processing.

OMA Object Management Architecture is OMG's formal architecture organizing standard objects that provide standard services for assignment object-oriented applications.

OMG Object Management Group is an open membership, not-for-profit consortium, that produces and maintains computer industry specifications for interoperable enterprise applications.

OpenMMS Toolkit contains software for parsing protein and nucleic acid macromolecular structure data stored in the mmCIF format.

PDB Protein Data Bank is the central database for protein structures.

PPIX Unit is an additional biological unit defined in the PPIX application.

PPIX Partition is a complete assignment of all asymmetric units to PPIX units.

PPIX Protein Protein Interaction (in crystals)

PQS Protein Quaternary Structure provides likely quaternary states for structures contained in PDB.

Report in this thesis is an automatically generated web page, that represents a list of data and provides search and browsing functionality.

Spacegroups are symmetry operations applied to points arranged on a crystal lattice. There are 230 unique crystallographic space groups, which describe the rotational and translational symmetry elements present in the structure.

Unit Cell is the basic building block of a crystal, repeated indefinitely in three dimensions. Characterized by three vectors and the angles between the vectors.

UML Unified Modelling Language is OMG's standard language for the analysis and design of applications, specifying the structure and behavior of systems. UML is defined as an underlying abstract syntax and an overlying graphical concrete representation.

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

XMI XML Metadata Interchange is an OMG specification, expressed as XML DTDs and schemas plus production rules for XML DTDs and schemas, for interchange of metamodels and models as XML documents.

Chapter 1

Introduction

1.1 Proteins

Proteins are the workhorses of the cell, performing nearly every function required for life. Within the last decade a large number of proteins has been analysed and an increasing insight has been gained into the dynamics at the macromolecular level.

A gene is a functional and inheritable element in the genome, that usually codes for a protein. The protein coding sequence of most genes is interrupted many times by blocks of noncoding sequences. Following the Central Dogma of Molecular Biology [2] the DNA is replicated, then transcribed into RNA. After a processing step (which among others includes splicing), the RNA is translated into proteins. (see fig. 1.1)

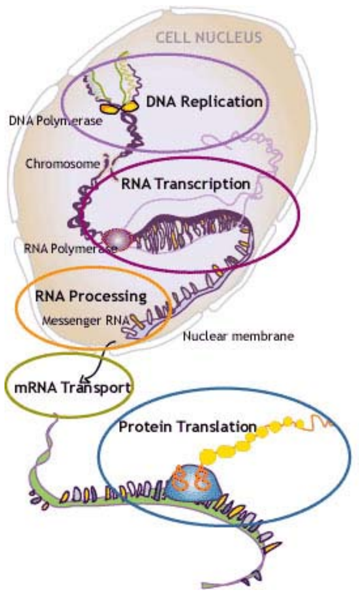


Figure 1.1: Central Dogma of Molecular Biology (taken from [1])

Within the last years the need to modify the dogma arose due to the discovery of microRNA (miRNA) [3], which play a critical role in regulating the timing and rate of protein translation [4] [5].

1.1.1 Protein Structure

Proteins are biological macromolecules. They consist of chains of amino acid sequences, which fold into unique structures. The structure of a protein is determined by four aspects (see table 1.1).

Primary Structure	is the amino acid sequence of the polypeptide chain(s), without regard to spatial arrangement.
Secondary Structure	is the local spatial arrangement of its main-chain atoms without regard to the conformation of its side chains or to its relationship with other segments.
Tertiary Structure	is the arrangement of all its atoms in space, without regard to its relationship with neighboring molecules or subunits.
Quartenery Structure	is the arrangement of its subunits in space and the ensemble of its intersubunit contacts and interactions, without regard to the internal geometry of the subunits.

Table 1.1: Definitions of Primary, Secondary, Tertiary, and Quaternary Structure by IUPAC-IUB Commission on Biochemical Nomenclature (CBN) [6].

1.2 Analysing protein structures

The most common method to analyse protein structures at the moment is X-ray diffraction. It is a technique in the field of crystallography, which uses the diffraction pattern produced by X-rays to determine the atom structure of a crystal. The second technique is nuclear magnetic resonance (NMR) spectroscopy. For the first method the protein has to be available in its crystallized form. The task of crystallising a protein can be a very enduring and exhausting task, which does not always achieve success. Many parameters like the pH, the temperature or the concentration of the additives are usually considered and varied. NMR determines structures of proteins in solution, but is limited to molecules not much greater than 30 kDa. NMR is often used for small molecules, because it reveals the locations of water molecules, which do not scatter X-rays and cannot be dedected using X-ray diffraction. During the first phase of the PPIX project only X-ray determined structures will be analysed.

1.2.1 Asymmetric and Biological Units

An **asymmetric unit** is a crystallographic concept. It defines the smallest portion of a crystal structure to which crystallographic symmetry can be applied to generate the unit cell. The symmetry operations most commonly found in biological macromolecular structures are rotations, translations, and screws (combined rotation and translation). The unit cell is the smallest unit in a crystal that, when translated in three dimensions, makes up the entire crystal [7].
The asymmetric unit is used by the crystallographer to refine the structure against experimental data and does not necessarily represent a biologically functional molecule.

A **biological unit** is the macromolecule that has been shown to be or is believed to be functional.
An asymmetric unit may contain one biological molecule, a portion of a biological molecule or multiple biological molecules. One biological unit on the other hand may be one copy of the asymmetric unit, multiple copies of the asymmetric unit or a portion of the asymmetric unit.

When analysing proteins using X-ray diffraction protein-protein contacts can either be biological (meaning, that they also exist in solvent and are biological functional) or contacts artificially created during crystallisation. Crystallisation contacts are of no functional relevance.

1.3 Existing protein databases

1.3.1 Protein Data Bank (PDB)

The Protein Data Bank [8] is a computer-based archival database for macromolecular structures. The database was established in 1971 by Brookhaven National Laboratory, Upton, New York, as a public domain repository for resolved crystallographic structures. Since 1999 the Protein Data Bank is operated by Rutgers, The State University of New Jersey and the San Diego Supercomputer Center at the University of California, San Diego – two members of the Research Collaboratory for Structural Bioinformatics (RCSB) [9] [10].

The Research Collaboratory for Structural Bioinformatics is a non-profit consortium dedicated to improve the understanding of the function of biological systems through the study of the 3-D structure of biological macromolecules. RCSB members work cooperatively and equally through joint grants and subsequently provide free public resources and publications to assist others and further the fields of bioinformatics and biology [11].

At the moment[1] the PDB contains 33585 structures, 28665 of them have been determined using X-ray crystallography. For the last two years more than 5000 new structures have been added into the database.

1.3.2 Probable Quaternary Structure (PQS)

The Probable Quaternary Structure uses an empirically derived weighted score to estimate whether protein-protein contacts that occur in crystals are specific oligomer interfaces or crystal artifacts. For those deemed specific oligomers, the oligomer is provided. When multiple copies of the molecule occur in the asymmetric unit, single copies are provided [12] [13].

1.4 Protein structure file formats

1.4.1 PDB file format

PDB file entries consist of records of 80 characters each [14]. Using the punched card analogy, columns 1 to 6 contain a record-type identifier, the columns 7 to 80 contain data. In older entries, columns 71 to 80 are normally blank, but may contain sequence information added by library management programs. The first four characters of the record identifier are sufficient to identify the type of record uniquely, and the syntax of each record is independent of the order of records within any entry for a particular macromolecule.

1.4.2 mmCIF file format

The Crystallographic Information File (CIF) is the International Union of Crystallography (IUCr) standard for presentation of small molecules. The macromolecular Crystallographic Information File (mmCIF) [15] is based on the Self Defining Text Archival and Retrieval (STAR) [16] format and is intended as the replacement for the fixed-field PDB format for presentation of macromolecular structures. The PDB format cannot express adequately the large amount of data associated with a single macromolecular structure and the experiment from which it was derived in a consistent way that permits direct comparison with other structure entries [17]. Detailed information on

[1]as of 15-11-2005

the file formats used in the PDB can be found at the information site of the PDB (http://www.rcsb.org/pdb/info.html#File_Formats_and_Standards).

```
PDB File Format Header:

HEADER    PLANT SEED PROTEIN               11-OCT-91   1CBN

becomes in the mmCIF File Format:

_struct.entry_id                 '1CBN'
_struct.title                    'PLANT SEED PROTEIN'

_struct_keywords.entry_id        '1CBN'
_struct_keywords.text            'plant seed protein'

_database_2.database_id           PDB
_database_2.database_code         1CBN

_database_PDB_rev.num                    1
_database_PDB_rev.date_original 1991-10-11
```

Listing 1.1: Excerpts of the protein 1CBN from the PDB file and the mmCIF file. The PDB file HEADER (identifier) line with fixed width column properties gets divided into several name-value pairs.

1.5 Objectives

As this thesis is only a part of the PPIX project, this section will give a short overview of the overall project goals and then point out the specific thesis aims.

1.5.1 Overall PPIX project goals

The project goal is to gain further insight into protein-protein interactions. For this purpose a scientific application will be designed, implemented and used. This will include a database, a web front end, a structure viewer, a number of calculations programs and analysis. Special attention will be laid on the surface of the crystal structure. The system should be able to store the protein surfaces and surface properties, protein-protein interaction patches and their properties.
The application will be used to run analysis to gain a better understanding for

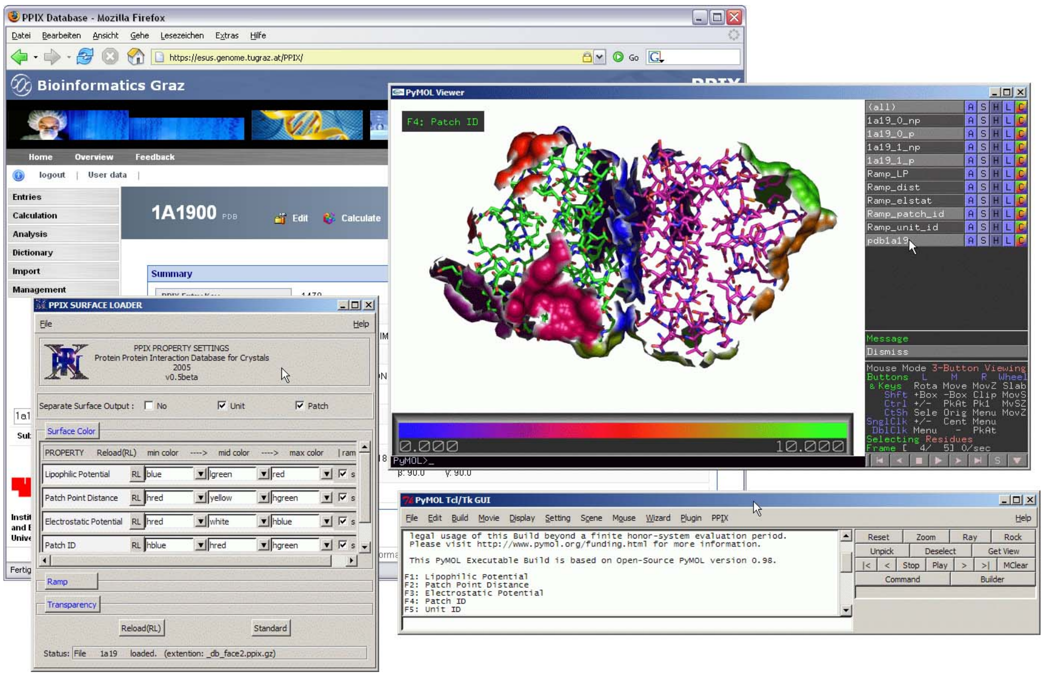
crystal-, biological- and transient contacts (an intermediate type of contact) in proteins.
The crystal patches will then be analysed and compared among each other in order to find patches, which will lead to improved protein crystallisation. On base of this analyses suggestions should be given, how to create protein mutants that enable crystallisation of proteins, which up to now have not been successfully crystallised.

1.5.2 Thesis project goals

For a detailed summary of the goals of the Protein Protein Interaction Database project see appendix A. The main goals are:

- the design of a database schema based on the Protein Data Bank, which should support surfaces points, patches, patch interaction and their properties. It should easily be possible to add additional properties later in the development.
- the implementation of an import function for mmCIF files. It should be possible to import a structure by either entering the PDB Id or by uploading a self generated file.
- the definition of an application interface to add and run analyses and calculations.
- the development of a web application, that allows to view the structures in the database and to start the import of PDB structures, analyses and calculation of surfaces, patches and their properties.

The visualisation of patches and patch properties is not part of this thesis, but will be integrated in the Web interface. The posibillity to view the calculated surfaces, surface patches and surface point parameters was implemented by Georg Steinkellner in form of a plugin for the molecular visualization system PyMol [18] [19].



PPIX PROPERTY SETTINGS
Protein Protein Interaction Database for Crystals
2005
v0.5beta

Separate Surface Output : ☐ No ☑ Unit ☑ Patch

PROPERTY	Reload(RL)	min color	---->	mid color	---->	max color
Lipophilic Potential	RL	blue		lgreen		red
Patch Point Distance	RL	hred		yellow		hgreen
Electrostatic Potential	RL	hred		white		hblue
Patch ID	RL	hblue		hred		hgreen

Status: File 1a19 loaded. (extention: _db_face2.ppix.gz)

```
 legal usage of this Build beyond a finite honor-system evaluation period.
 Please visit http://www.pymol.org/funding.html for more information.

 This PyMOL Executable Build is based on Open-Source PyMOL version 0.98.

F1: Lipophilic Potential
F2: Patch Point Distance
F3: Electrostatic Potential
F4: Patch ID
F5: Unit ID
```

Figure 1.2: Viewing the PPIX structure including its patches and surface point parameters with PyMol.

Chapter 2

Methods

This chapter will give a brief introduction of the technologies and tools used during the thesis.

2.1 Used Standards

In this section some basic standards will be shortly presented. All of the standards have been published by the Object Management Group (OMG). 'The OMG is an open membership, not-for-profit consortium that produces and maintains computer industry specifications for interoperable enterprise applications' [20].
The OMG is not a development organisation. All the development work of the published standards is done by its members, which include almost every major company in the computer industry. The focus of the OMG is to create standards, that are used commercially, and by that way foster the development of reusable and interoperable software.
All specifications of the OMG can be downloaded at the online 'Catalog of OMG Modelling and Metadata Specifications' [21].

2.1.1 Model Driven Architecture (MDA)

The Model Driven Architecture (MDA) [22] is the flagship specification of the OMG. The OMG Model Driven Architecture addresses the complete life cycle of designing, deploying, integrating, and managing applications. The three primary goals of MDA are portability, interoperability and reusability through architectural separation of concerns [23].
MDA is an approach to use models in software development. It is based on

the OMG standards Unified Modelling Language (UML), Meta-Object Facility[1] (MOF), XML Metadata Interchange (XMI) and Common Warehouse Metamodel[2] (CWM) and tries to separate business and application logic from underlying platform technologies.

2.1.2 Unified Modelling Language (UML)

Modelling is an abstract way to describe the architecture of a software project before coding. A model plays the analogous role in software development that blueprints play in the building of a skyscraper. Modelling is a way to visualise a design and check it against requirements like scalability, robustness, security or extendibility before any line of code is written. In this way a lot of errors might be prevented before they even happen [26].

UML 2.0 defines thirteen types of diagrams, divided into three different categories.

Structure Diagrams include among others the Class Diagram, the backbone of every object oriented software design. Other diagrams are the Object Diagram, the Component Diagram, the Composite Structure Diagram, the Package Diagram, and the Deployment Diagram.

Behavior Diagrams include the Use Case Diagram (used during requirements gathering), the Activity Diagram, and the State Machine Diagram.

Interaction Diagrams all derived from the more general Behavior Diagram, include the the Sequence Diagram, the Communication Diagram, the Timing Diagram, and the Interaction Overview Diagram.

2.1.3 XML Metadata Interchange (XMI)

The XML Metadata Interchange (XMI) is an OMG standard for exchanging object information via the Extensible Markup Language (XML). XMI can be applied to different kinds of objects like software (e.g. Java), components (e.g. Enterprise JavaBeans) or databases (e.g. CWM). The most common use and the original purpose of XMI is to enable easy interchange of metadata

[1]The OMG Meta-Object Facility (MOF) bridges the gap between dissimilar metamodels by providing a common basis for metamodels [24].

[2]The Common Warehouse Metamodel (CWM) is a specification that describes metadata interchange among data warehousing, business intelligence, knowledge management and portal technologies [25].

between UML-based modeling tools.
The XMI specification defines how to

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

There are two versions of the XMI specification. Version 1.x defines the main purpose of XMI. The most recent XMI specification (v2.0) uses XML Schemas for a better document validation [27].
As every UML tool has its own XMI implementation, there are a lot of incompatibilities between the generated XMI files. Good tools have some import/export functions to support diagrams created by other tools.

2.2 Java 2 Enterprise Edition (J2EE)

The Java 2 Platform, Enterprise Edition (J2EE[3]) provides a component-based approach to the design, development, assembly, and deployment of enterprise applications[4]. Application logic is divided into components according to its function. The various application components that make up a J2EE application are installed on different machines depending on the tier the application component belongs to [28][29].

The J2EE standard defines following components:

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

[3]The next version will be called Java 5 EE or just Java EE.

[4]An enterprise application in general is a highly complex systems. It is a business application supposed to be scalable, distributed, component-based, and mission-critical. It may be deployed on a variety of platforms across corporate networks, intranets, or the Internet. It is data-centric, user-friendly, and must meet stringent requirements for security, administration, and maintenance.

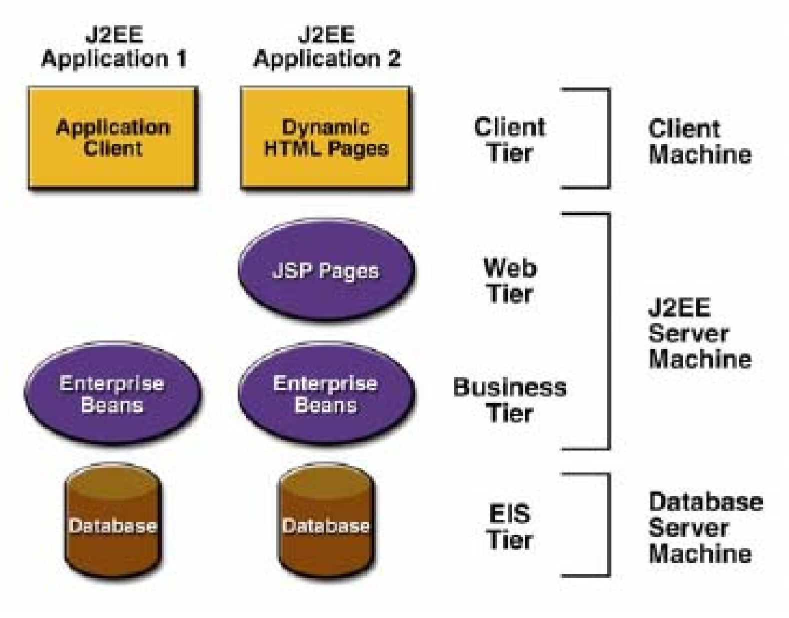


Figure 2.1: Multi-tiered J2EE Application (taken from [28])

- Enterprise Information System (EIS) software runs on the database server.

Although there are four different tiers, J2EE applications are usually referred to as three-tiered, because they run on three different machines (see fig. 2.1).

In addition to the components J2EE also defines underlying services in form of containers for every component type. They handle low level details like transaction and persistence handling, multithreading, security or naming. Container types are the Enterprise JavaBeans- (EJB), the Web-, the Application- and the Applet container (see fig. 2.2). Before a component can be executed it has to be assembled into a J2EE module and deployed into a container. Although the low level services do not have to be implemented, they have to be configured using XML files, called deployment descriptors.

The J2EE standard includes several API specifications for example Java Database Connectivity (JDBC), Java Transaction (JTA), Java Message Service (JMS), client-side applets, Remote Procedure Call (RPC), Common Object Request Broker Architecture (CORBA), and defines how to coordinate them (see fig. 2.2). J2EE products are application servers that provide a complete implementation of the EJB, Servlet, and Java ServerPages (JSP) technologies.

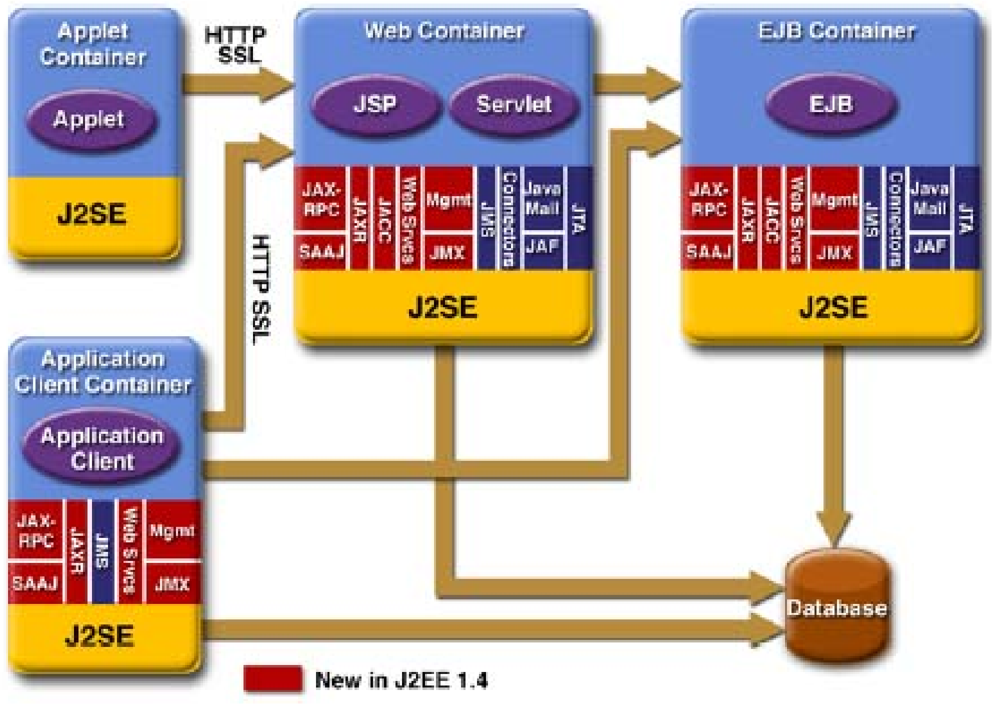


Figure 2.2: J2EE Platform API (taken from [28])

2.2.1 Enterprise Information Tier (Data Tier)

The enterprise information system (EIS) tier handles enterprise information system software and enterprise infrastructure systems like database systems, enterprise resource planning (ERP) systems (e.g. SAP) and other legacy information systems. J2EE application components for example might need access to enterprise information systems for database connectivity. In the PPIX application an Oracle Database Server is used.

2.2.2 Application Tier

Although the application and the Web tier are two different tiers, they are usually referred to as one, because generally they run on the same server.
The application tier is also referred to as the Enterprise JavaBeans (EJB) tier. It hosts application-specific business logic and provides system-level services such as transaction management, concurrency control, and security.

Enterprise JavaBeans

Enterprise JavaBeans specifies a server-side component model. Using a set of classes and interfaces from the javax.ejb package, developers can create, assemble, and deploy components that conform to the EJB specification. EJB components are a fundamental link between presentation components hosted by the Web tier and data maintained in the enterprise information system tier. EJB components run in the EJB container of the application server [30][31].

There are three different types of EJBs:

Entity beans (EBs) represent objects, that are usually persistent records in a database.

Session beans (SBs) are extensions of the client application that manages a process or taskflow. Session beans are not persistent. There are two kinds of session beans. A stateful session bean keeps its conversational state[5] in memory across method calls. The state is retained while a client carries on a conversation with an enterprise bean, and is lost when the conversation ends.
Stateless session beans do not maintain a conversational state for a client. They have two advantages over stateful session beans. One bean can service more than one client and no memory has to be spent on instance variables.

Message driven beans (MDBs) also coordinate tasks accessing other session and entity beans. The most important difference is the way they are accessed. They listen to messages and upon receiving one, they process it asynchronously.

2.2.3 Web Tier

In the Web tier HTTP requests are processed. In a J2EE application, the Web tier usually manages the interaction between Web clients and the business logic. The Web tier typically produces HyperText Markup Language (HTML) or XML content, though the Web tier can generate and serve any content type. Web components are either Servlets or pages created using JSP technology.

[5]the state of an object consists of the values of its instance variables.

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

- manages interaction 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 present results as Web content.
- usually determines which 'screen' (that is, which page) is to be to display next.
- has a simple, flexible mechanism for accumulating 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 seperation between the presentation and the business logic tier is broken and it is more difficult to maintain the application. This is why enterprise applications normally implement the business logic in enterprise beans.

Servlet

A Java Servlet is a platform-independent Web application component that is hosted in the Web container. Servlets cooperate with Web clients by means of a request/response model.

JavaServer Pages

JavaServer Pages are an extension to the Java Servlet technology. They can be referred to as 'inside-out Servlets', because instead of writing the HTTP response inside a print statement, everything in a JSP file is written to the response except of special tags and scriptlets. When a JSP page is accessed for the first time it gets translated into a Java Servlet. In this way Plain HTML pages can be easily enhanced with dynamic content.

The features of JSP include

Plain text is written directly to the response. In most cases this is HTML code.

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

Scriptlets are pieces of Java code. Either the code is copied into the Servlet using '<% %>' tags or the statement is evaluated into an expression and written to the response using '<%= %>' tags.

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

Custom tags add the possibility to create custom actions.

Especially the feature to add custom defined tag libraries[6] supports the effort to separate the application logic from the presentation and to simplify the maintenance of the presentation layer.

Many times, JSP tags work hand-in-hand with JavaBeans. The application sends a JavaBean to the JSP, which include its content in the response. Although JavaBeans are a component model like Enterprise JavaBeans they are completely unrelated. While JavaBeans are for intraprocess purposes, Enterprise JavaBeans are for distributed components. JavaBeans are Java classes, which obey certain conventions about method naming (for the access to its properties), construction (there should be an empty default constructor) and behavior (the class should be serializable) [32] [33].

2.2.4 Client Tier

A client can either be a Web client or an application client. Application clients are programs running on the client machine and access enterprise beans running on the business tier. Web clients display content generated by the Web tier. The content usually is some kind of markup language like HTML.

2.2.5 J2EE Patterns

Design Patterns are reusable, tested and proven to function solutions for reoccurring problems in software development. Some of the utilized patterns are

[6]A tag library (or short taglib) is a collection of custom tags

shortly described in this section. Detailed information about J2EE patterns can be found in Core J2EE Patterns [34].

Business Delegator decouples business components from the code that uses them. Thus a change in a component does not affect the application code.

Transfer Objects (Value Objects) group related attributes and are often used as a return type of remote business methods. The client can then access the objects locally and update the attributes by an update business method.

Session Facade provides clients with a single interface for the functionality of an application or application subset. It also decouples lower-level business components from one another, making designs more flexible and comprehensible.

Data Access Object Factory adapts specific data resource's access APIs to a generic client interface. It allows data access mechanisms to change independently of the code that uses the data by separating the data resource's client interface from its data access mechanisms.

Service Locator centralises distributed service object lookups, provides a single point of control, and may act as a cache that eliminates redundant lookups.

Value List Handler provides a client with an iterator for a virtual list that resides in another application tier. The iterator typically accesses a local ordered collection of Transfer Objects.

Model View Controller decouples data access, business logic, data presentation and user interaction. (see sec. 2.3)

2.2.6 JBoss Application Server

The PPIX project has been develeoped and is currently running on a JBoss application server. It is a popular, open source and platform-independent J2EE compliant application server. JBoss includes a Web server (Servlet/JSP container, and HTML server), Enterprise JavaBeans (EJB) container, Java Message Service (JMS), JavaMail, and Java Transaction API/Java Transaction Service (JTA/JTS). JBoss is usually shipped with an Apache Tomcat Servlet engine, although there exists a version that uses an embedded Jetty Servlet engine.

2.3 Struts

There are two JSP Models. The first one proposes a direct connection from a Web client to the JSP container. The second model has an intermediate front controller Servlet, that receives all requests and determines which JSP page to display next. The second approach called JSP Model 2 architecture tries to take advantage of the strength of Servlets and JSPs using a Servlet for request processing and JSPs for presentation. The separation from business logic, presentation and request processing is also referred to as a Model-View-Controller pattern. The open source framework Struts [35] implements this pattern and includes some useful features like JSP custom tag libraries.

2.3.1 The Model

Depending on the type of architecture the model portion of the MVC pattern can take many different forms. In a complex enterprise application the model portion of the MVC pattern will be implemented in Enterprise JavaBeans. In many cases JavaBeans are returned from session beans and used within the Web tier, because of a performance loss when calling an entity bean remotely. These JavaBeans, commonly referred to as data transfer objects or value objects, are used within the views to build the dynamic content.

2.3.2 The View

The view portion of a Struts-based application is most often constructed using JavaServer Pages. Struts includes a set of custom tag libraries that facilitate creating user interfaces that are fully internationalized and interact gracefully with beans. There are taglibs for bean handling, for HTML rendering and for logic functions.

Some features of Struts and its tag libraries include

Internationalization[7] is supported using Message Resources. Struts treats a set of resource bundles like a database, and allows to request a particular message string for a particular Locale. To support a certain language only a resource bundle has to be defined, that has an ISO suffix ('_xx' where xx is the ISO language code) added to its name.

Automatic Form Population Using the HTML tag library Struts automatically fills in the current form values into the HTML form input elements.

Validation of form fields is integrated into the Struts framework by the Validator plugin. If validation rules exist for a form element, Struts will automatically validate the user input using either some predefined functions for validating date, number and email formats or a self written function. Beside the server-sided validation routines Struts provides also a meachanism for performing client-sided validation using JavaScript versions of the validation routines.

2.3.3 The Controller

The Controller Servlet is the backbone of the Struts Framework. Its primary function is to map a request Uniform Resource Identifier (URI) to an action class. The Servlet receives all client requests and calls a user defined action class. The action class processes the call and returns a value, which tells the controller what page to display next. The use of this single point of access facilitates for example the user authentication, which otherwise would have to be done within each page.

2.4 XDoclet

Writing and maintaining deployment descriptors for EJBs is a labour-intensive and error-prone job. Detailed customisation of an enterprise application can lead to some large and complex files. The XDoclet code generation engine can write deployment descriptors as well as EJB interfaces classes interpreting Javadoc-style attribute tags inside the Bean code [36].

```
/**
 *
 * @ejb.bean name="AtomType"
 *   type="CMP"
 *   cmp-version="2.x"
[...]
 */
public abstract class AtomTypeBean
        implements javax.ejb.EntityBean
```

Listing 2.1: Example of XDoclet tags for an entity bean

2.5 AndroMDA

AndroMDA [37] is an open source code generation framework that follows the Model Driven Architecture paradigm. It evolved from uml2ejb, a project by Matthias Bohlen that was originally implemented to generate Enterprise Java Beans from UML Diagrams. It is constructed modularly consisting of a core and additional cartridges, which generate the code. The standard scripting language used is called Velocity and is part of the Apache project. Apache Ant [38] was used in this project as a build tool. The Java-based Ant uses XML-based make files. It has been used with subtasks for running AndroMDA and XDoclet, for compiling the Java code and for building application archives (.jar, .war, .ear files).
Figure 2.3 shows how Ant generates the project using AndroMDA and XDoclet.

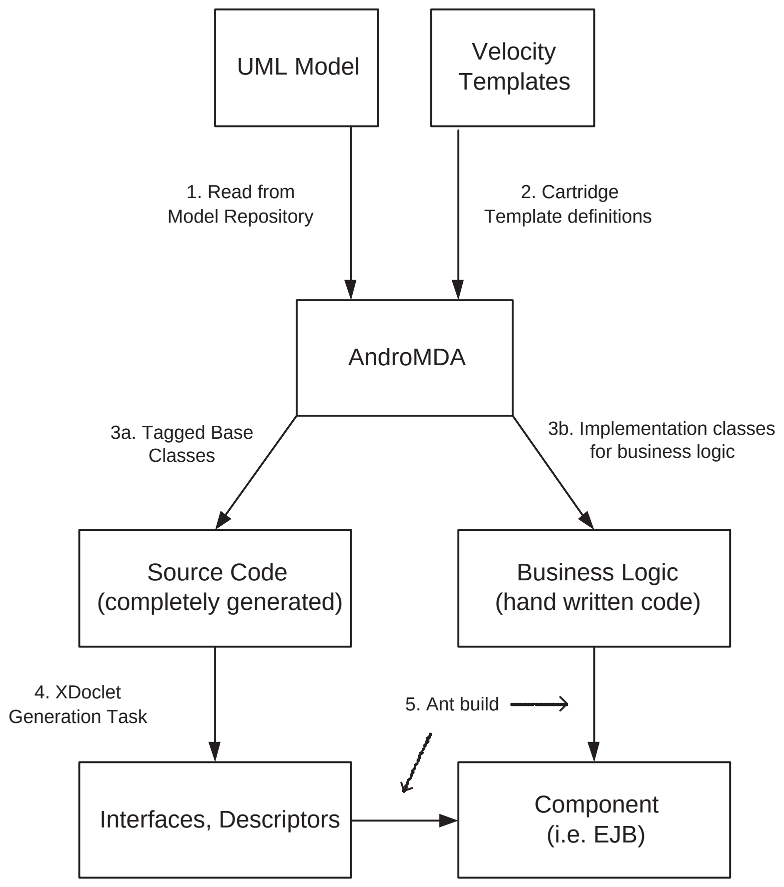

UML Model
Velocity Templates
1. Read from Model Repository
2. Cartridge Template definitions
AndroMDA
3a. Tagged Base Classes
3b. Implementation classes for business logic
Source Code (completely generated)
Business Logic (hand written code)
4. XDoclet Generation Task
5. Ant build
Interfaces, Descriptors
Component (i.e. EJB)

Figure 2.3: Flowchart of a UML based build process using Ant, AndroMDA and XDoclet (taken from [39])

2.5.1 Cartridges

The cartridges define what files are generated out of which UML element and what content is written into the files. The cartridges used within this project have been developed by Thomas Truskaller [39] and enhanced by Robert Molidor and Jürgen Hartler. Some adaptations have been made during this project (see sec. 3.6). The cartridges allow among other things to create entity-, session-, message driven beans, value objects and report Web pages (see sec. 3.4) automatically.

2.5.2 Schema2XMI

Schema2XMI is a generator that reads a database schema and writes it into an XMI model. It was written by Chad Brandon to facilitate the migration from already existing applications or database schemas to AndroMDA based applications. The program accesses and inspects database using the java.sql.DatabaseMetaData interface of the JDBC-API and generates an XMI file using the org.omg.uml package [40].

Basic features include:

- Associations will contain correct multiplicities based on foreign key relations.
- Correct multiplicities on attributes that are nullable vs. non-nullable (0..1 and 1).
- Composite aggregation will be created on relationships marked with cascading deletes.
- Configurable stereotypes will be added to classes and identifiers.
- Optional configurable tagged values for column names and table names can be added to attributes and tables respectively.

During the project an extended version including a graphical user interface was developed (see sec. 3.5).

2.6 Tools

This section will introduce the main development tools used during the thesis.

2.6.1 UML Tools

UML Tools are software programs for creating UML Diagrams. There exist a lot of free and commercial UML tools, that differ in their features and handling. A good starting point for searching an adequate tool can be found at the UML Homepage [41]. During the development of the thesis project two tools have been used.

Rational Rose

Rose [42] is a CASE[8] tool, that was originally developed by General Electric and later purchased by Rational. Under the lead of Rationals Chief Scientist Grady Booch [43] (one of the brains in the development of UML) Rose became one of the first CASE tools to support UML. In 2003 Rational was acquired by IBM. Rose is one of the completest and most powerful UML tools, but also one of the most complex to handle.

MagicDraw

MagicDraw is a visual UML modelling 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 systems and databases [44].
MagicDraw supports the complete UML 1.4 notation and all diagram types. The current version 10.0 is supposed to support UML 2.0 and XMI Version 2.1. MagicDraw is quite popular for its feature and ease of use. As it is written completely in Java, it is platform independent. The MagicDraw Addon RConverter provides a way to convert the Rational Rose Model file format to MagicDraw supported file format.

2.6.2 JBuilder

JBuilder is an Integrated Developer Environment (IDE) that was used for building the project. Running the ant build task or the application server from within the IDE have been rather slow on the development computer, thus this tasks have been started in an own shell.

[8]**CASE** = **C**omputer **A**ided **S**oftware **E**ngineering

2.7 Genome Usermanagement

The Genome Usermanagement [45] [46] is a Java-based usermanagement and authentication system, that was developed by Dieter Zeller. It consists of a usermanagement server and a client part. The server has a Web front end for configuration. The client part integrates fully into Struts and provides an API as well as its own tag library to query the server.
It includes among other features the definition of users and user groups, applications, resources and access control lists. In JSPs permission tags of the tag library can be used to display certain sections only for users with appropriate rights (see listing 2.2). In the Java code the usermanagement API can be used to control if a user is allowed to execute a certain task.

```
<permission:noPermission resourceKey="comp" accessLevel="R">
<table><tr>
  <td class="StandardTableCellEven">
    <html:img align="center" src="images/stop.png"/>
  </td>
  <td colspan="2" align="center">
    insufficient rights to get this information
  </td>
</tr></table>
</permission:noPermission>
```

Listing 2.2: Custom JSP Tag of the Genome Usermanagement. The section between the permission tags will only be written to the response stream, if the current user has the accessLevel 'R' (=Read) for the resource 'comp'.

2.8 OpenMMS

The OpenMMS (Open Macromolecular Structure) Toolkit, released by the non-profit Research Collaboratory for Structural Bioinformatics (RCSB) consortium, allows researchers to use the PDB more efficiently with the Common Object Request Broker Architecture (CORBA) standard. Out of the complete toolkit only the mmCIF parser was used for the project. The toolkit can create a mmCIF parser based on the mmCIF dictionary [15], which gets regularly updated. This allows a generated parser to be able to read all mmCIF categories and items [47].

Chapter 3

Results

In the course of this thesis a WEB application to calculate, store, view and analyse surface attributes and protein-protein crystal contacts for proteins with a known structure has been developed. It is currently used by the 'Institut für Physikalische Chemie (Subgruppe Strukturbiologie)' [48] at the Karl-Franzens University Graz. At the moment the application is used for internal research and analyses. In long term it is planned to open parts of the application for public access.

This chapter describes the Web application developed during the work on the diploma thesis. It is divided into five sections. The first section describes the development of the schema of the PPIX Database. The second outlines the functionality of the business logic included in the application. The third shows the Web front end of the application.
The last two sections present Schema2XMI, a tool developed and used for creating the UML model of the application, and extensions to the AndroMDA cartridges.

The implementation of the application is based on a user requirement definition, which is the result of several intensive discussions between the users at the Structural Biology Division of the Institut for Chemistry and the developers. It has been edited by Markus C. Jorde and can be found as an appendix (see app. A).

3.1 PPIX Database Structure

The database is based on the OpenMMS Database schema. A diagram of the schema was created using Rational Rose. In a number of meetings with members of the Structural Biology Division of the Institut for Chemistry a lot of tables and attributes, which are not necessary for the purpose of the PPIX application have been removed from the schema [19]. Additional tables for protein surfaces, contacts and patches have been designed. An overview of all PPIX tables can be found as Appendix C.

3.1.1 Adopted PDB tables

A total of 42 out of 176 tables in the OpenMMS toolkit have been selected during the design phase. In the main table of a protein structure (ENTRY) a numeric primary PPIX key has been added. In addition to the PDB Id a version number field has been created to support more than one version of a PDB structure.
As the OpenMMS schema is based on the mmcif flat file format, it has a star like design. Every table includes a PDB Id field. The schema has the advantage that search operations are quick, because not many JOIN operations are needed, but there is a lot of redundancy and also the relations between the tables are not very clear. Pieces of information are stored in more than on place within the same file. Furthermore identical information is stored in each structure file, like a list of chemical components present in each structure. In the data definition language (DDL) files foreign keys are not explicitly declared, are not always unambiguous and do not contain the primary key of the foreign table, but an alphanumeric Id.
An effort was made to remedy this shortcomings by explicitly defining foreign keys for primary keys of foreign key tables. The search and referencing of foreign keys is done during the import (see sec. 3.4.5) of a structure (see lst. 3.1).

Four tables are very important for all calculations, because they contain data needed by many of the calculations, like the calculation of the surface or the surface properties. The chemical component table (CHEM_COMP) and its atom table (CHEM_COMP_ATOM) store information about chemical components and their atom attributes like the radius, the hydrophobicity or the electrostatic potential. The atom type table (ATOM_TYPE) stores general information about atom types like the radius, which can be used as estimation parameters when a chemical component is not present or has no attributes defined. The space group (SPACEGROUP) table stores all crystallographic space groups. These tables serve as *dictionary tables* and are referenced by each structure.

```
loop_
_atom_site.group_PDB
_atom_site.id
_atom_site.type_symbol
_atom_site.label_atom_id
_atom_site.label_alt_id
_atom_site.label_comp_id
_atom_site.label_asym_id
_atom_site.label_entity_id
_atom_site.label_seq_id
_atom_site.pdbx_PDB_ins_code
_atom_site.Cartn_x
_atom_site.Cartn_y
_atom_site.Cartn_z
_atom_site.occupancy
_atom_site.B_iso_or_equiv
_atom_site.Cartn_x_esd
_atom_site.Cartn_y_esd
_atom_site.Cartn_z_esd
_atom_site.occupancy_esd
_atom_site.B_iso_or_equiv_esd
_atom_site.auth_seq_id
_atom_site.auth_comp_id
_atom_site.auth_asym_id
_atom_site.auth_atom_id
_atom_site.pdbx_PDB_model_num
ATOM   1    N N     . VAL A 1 11  ? 36.141 33.321
 51.604 1.00 26.00 ? ? ? ? ? 11   VAL A N   1
ATOM   2    C CA    . VAL A 1 11  ? 36.606 32.459
 52.727 1.00 22.64 ? ? ? ? ? 11   VAL A CA  1
```

Listing 3.1: Excerpt of the atom site description of the 1KM4 mmCIF file including the decription header and the first two value lines. During the import into the PPIX foreign key references are made for the type_symbol (ATOM_TYPE), comp_id (CHEM_COMP), asym_id (STRUCT_ASYM), the sequence (STRUCT_SEQ) and the enitity_id (PENTITY) table. The combination of the atom_id and the comp_id is used to reference an atom of a chemical component (CHEM_COMP_ATOM).

3.1.2 PPIX Tables

As already mentioned in the introduction one of the main reasons to develop this database is to store the protein surfaces, the surface patches and the mapped surface point properties (e.g. hydrophobicity or electrostatic potential). To be able to include these attributes some additional

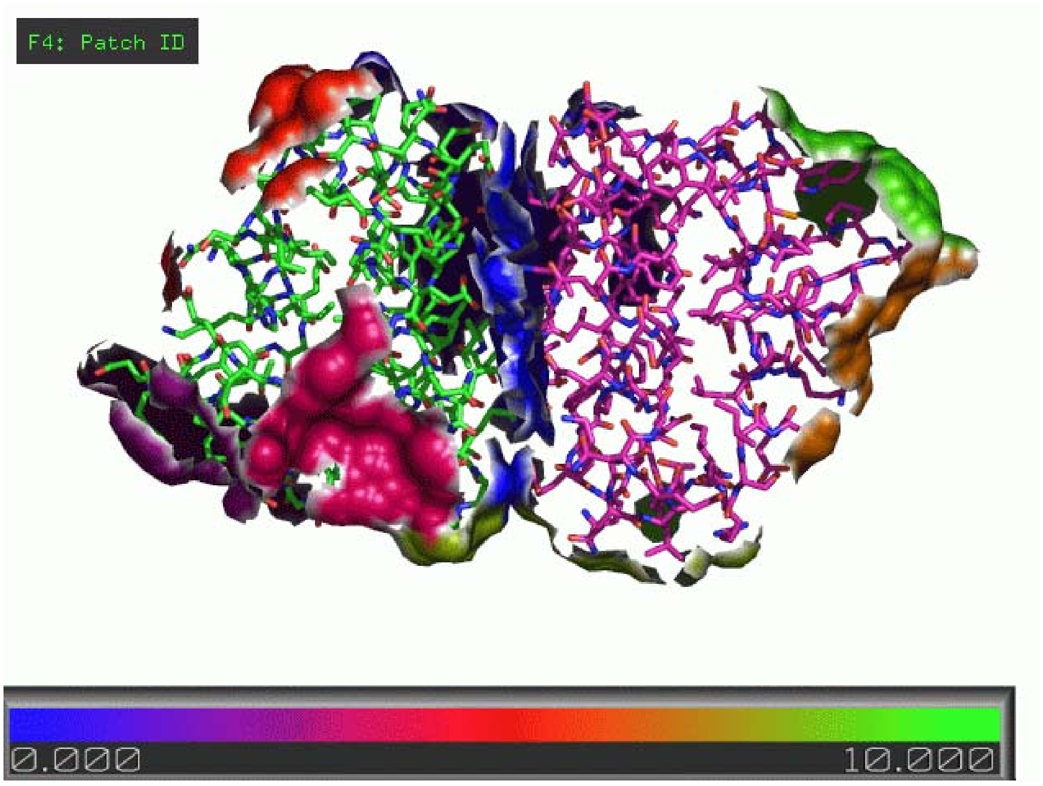
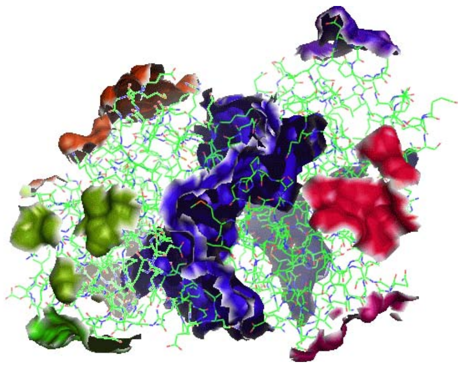


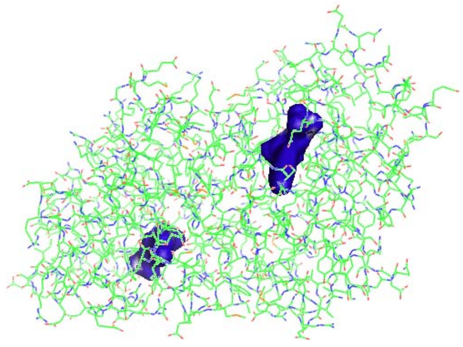
Figure 3.1: Display of the protein 1a19 with colour annotated surfaces patches. The colour schema gives the distance between the interacting patch surface points in Angstrom.

database tables have been introduced. Asymmetric units[1] (struct_asym) can be combined to form PPIX units (UNIT). PPIX units are biological units, that may differ from biological units stored in the PDB. For these units surfaces (UNIT_SURFACE) with different resolutions can be stored. A surface consists of triangles (TRIANGULATION) of surface points (SURFACE_POINTS). In addition each surface point is assigned to one atom (ATOM_SITE).
A combination of all available asymmetric units into PPIX units is called partition (UNIT_PARTITION) (based on the mathematical concept of a set). Surfaces and patches (PATCH) will be calculated for units. One patch can be between a unit A and a unit B or also between a unit A and unit A (with a different symmetry operation). Figure 3.2 shows an example of different unit configurations for the structure 1KM4.

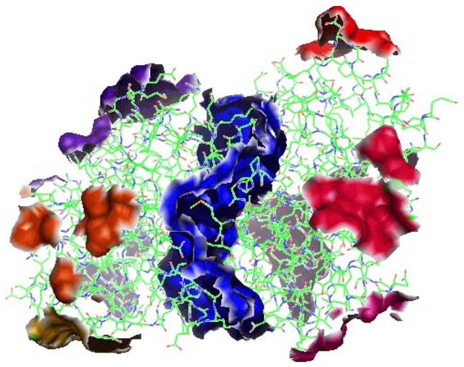
[1]Asymmetric units include chains and hetero components.



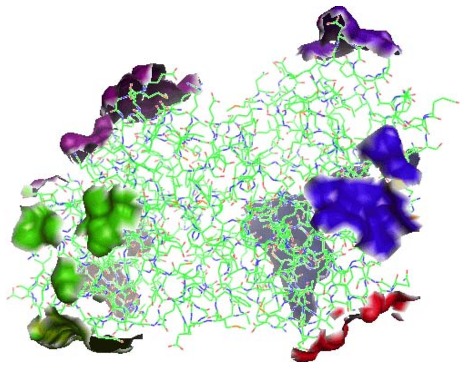
(a) Patches between the four defined units (the original PDB chain and hetero component and two units generated by a symmetry translation)



(b) Patches of the hetero components. They are completely embedded in the chains.



(c) Patches between the two units (Chain A + Hetero C and Chain B + Hetero D)



(d) Contact patches for a unit containing all asymmetric structures (Chain A + Chain B + Hetero C + Hetero D). These patches are only crystallisation contacts.

Figure 3.2: Example of a PPIX unit configuration proccess. The figures show the structure of the protein 1KM4 in pyMOL with colour annotated contact regions. It is a dimer with four asymmetric units, two chains and two hetero components. At the moment when an new structure gets imported a basic PPIX unit partition is created, in which each PPIX unit contains a single asymmetric unit (a). As each hetero component is completely covered by a chain (b), these two complexes can be defined as two units (c). As the structure is dimeric, it forms one biological unit (d). This result corresponds to the information from the PQS.

The patch interaction table (PATCH_INTERACTION) stores how these patches interact. It also contains the symmetry operation to obtain the patch.

3.2 Creating the UML Diagram

Based on the design of the database schema the UML diagram of the application was designed. To create a base model an extended version of schema2xmi (see sec. 3.5) was used. The created entities in the model were then divided into different domains to simplify the design. The domain definition is based on the category definition of the mmCIF dictionary [15]. A list of Entities in each domain can be found in figure 3.3.

The *domains* are:

Atom includes all tables containing information about atom positions.

Contact includes all contact specific entities (including most of the added tables).

Entity includes all entities containing information about chemical entities.

Others includes all entities, which do not fit into the other domains and are too few to form an own domain.

Struct includes all entities containing structure information.

System includes the main entity for one entry and a system table for database information.

Later in the application development additional domains for calculations, analyses, imports and the task management have been added.

For all these domains diagrams in the model have been created containing the entities (tables), a Session Bean and one exception class. Business functions that belong to this domain have been specified in the Session Bean (UML Service) element.

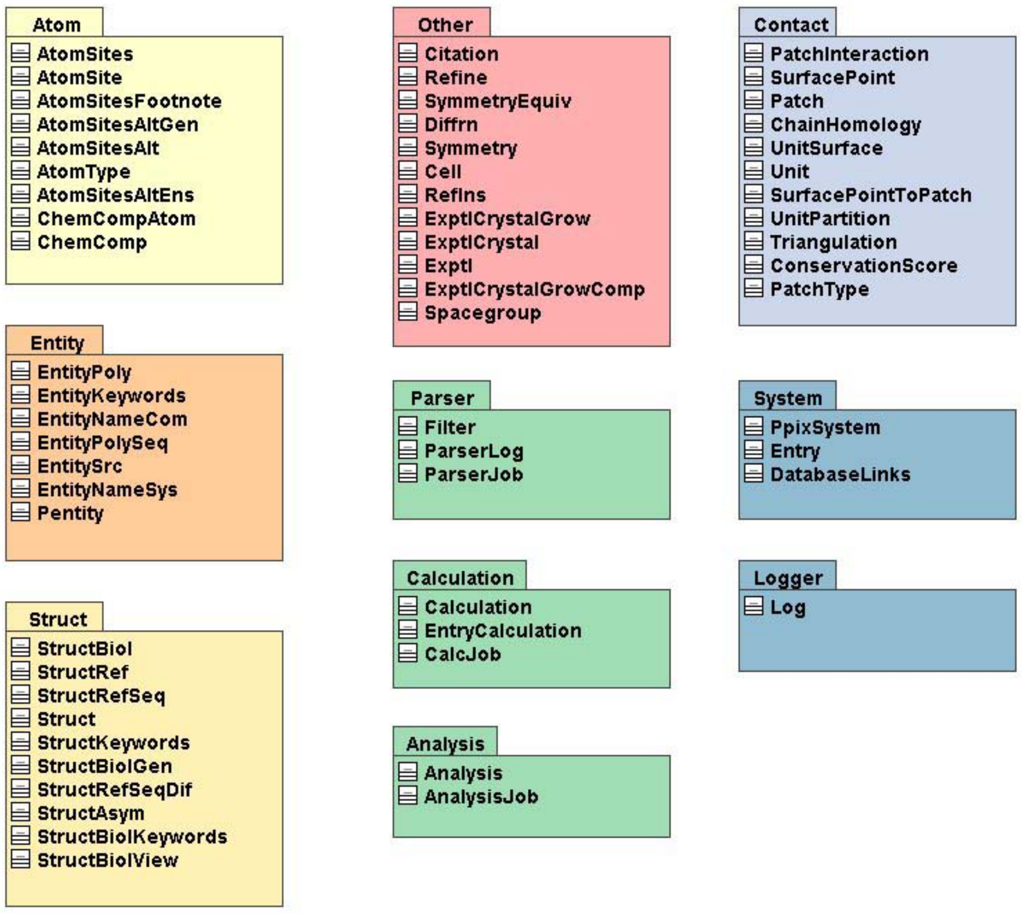


Figure 3.3: Diagram of all PPIX entities sorted by their domains. Yellow to red domains include entities taken from the OpenMMS schema. The light blue Contact domain contains all entities for storing surfaces, surface patches and unit partitions. The three green domains contain task managment entities. The darker blue domains contain System entities.

3.3 Business Logic

The business logic of the application is encapsulated in Enterprise JavaBeans. Many of the basic business functions (e.g. creating, removing or finding entries) are automatically created and implemented by AndroMDA. Additional in the model defined business functions are generated with a dummy return value and a Javadoc '@todo' tag in 'implementation' classes, which do not get overwritten, when rebuilding the application from the UML model.

The following *Session Beans* have bean defined:

AnalysisService provides service functions to start and stop analyses.

CalulationService provides service functions to start and stop calculations.

ExporterService provides service functions to retrieve a list of available exporters and to export structures.

PPIXAtomService provides service functions for all entities in the Atom domain.

PPIXContactService provides service functions for all entities in the Contact domain, including methods for defining, changing and retrieving unit partitions.

PPIXEntityService provides service functions for all entities in the Entity domain.

PPIXLogService provides service functions for the log table.

PPIXOthersService provides service functions for all entities in the Other domain.

PPIXParserService provides service functions to start and stop the import of PDB structures and to import and update components.

PPIXStructService provides service functions for all entities in the Struct domain.

PPIXSystemService provides service functions for all entities in the Atom domain, including methods to get and verify structure lists given in the VersionID[2] format.

[2]The PPIX VersionId consists of the PDB Id and a two digit version number.

Further Session Beans have been defined for all Report Web pages (see sec. 3.4.1) in the application. These provide search functions and result list handling mechanisms.

Following *Reports* have been defined:

- Analysis
- AnalysisJob
- AtomType
- DatabaseLinks
- Calculation
- CalcJob
- ChemComp
- ChemCompAtom
- EntryView
- Log
- ParserJob
- Spacegroup

Beside the Session Beans Message Driven Beans have been defined for running imports, calculations and analyses.

3.3.1 PDB Importer

The import of PDB structures is done by importing mmCIF files into the database in a two step process. In the first step the CIF file is parsed into a Java object using the OpenMMS (see sec. 2.8) mmCIF parser. In the second step foreign keys are linked (for an example see lst. 3.1, dictionary references are made and only then the data is written into the database.

Between step one and two there is also a *filter* function, which filters unwanted structures. At the moment only X-Ray diffracted structures are inserted. Another filter criteria checks the existence of a transformation matrix for the crystal. A complete list of filtering conditions can be found in the

User Requirement Document (see app. A). Does one of the criterias match the

The importer is called from within a message driven bean. That means an import job can be started by sending a message to a JMS[3] queue. The message type has to be an object message that contains a UserParserMessage Object. It includes a repository type, a repository location and a list of PDB IDs, which should be imported.

Supported *repository types* are:

local means, that the file is stored on the local file system.

web means, that the files have to be accessed over the Internet.

There is also a compression type field in the message. The importer tries to access a file, which is named PDB Id + "." + compression type. The compression type should be given, otherwise all compression types have to be tried until an accessible file is found.

Supported *compression types* are:

.cif an uncompressed CIF file

.cif.gz a gzip compressed CIF file

.cif.Z a unix compressed CIF file

.cif.zip a windows compressed CIF file

For a detailed description of the UserParserMessage class have a look at the interface definition document (see app. B).

The dictionaries that need to be accessed during the linking of each atom are loaded into hash tables before the first PDB structure is imported. This causes a delay of about one minute before the first mmCIF file is parsed, but it significantly increases the speed for importing additional files.

The Parser Message Driven Bean is a *Singleton Bean Managed MDB*. All incoming messages on the queue will be handled by only one instance of the message bean. In this way it is guaranteed, that the large dictionary tables are only loaded once and that unnecessary database transaction locking is

[3]JMS...**J**ava **M**essaging **S**ervice

avoided.
The bean is also defined as 'Bean Managed', meaning that not the bean container controls the database transactions, but the bean itself. One transaction includes the import of one entry. If there are any errors during the import, the transaction will be rolled back, and the import will continue with the next given structure.
The database access itself is done by direct JDBC calls using prepared statements to increase the speed of the imports. Compared to using entity beans to store a structure a speedup of the factor five has been achieved.

Component Importer

Beside the mmCif structure file importer also an importer for the mmCIF component file[4] has been implemented. The component file contains all chemical components and their atoms used in the PDB.

Unfortunately there are no attributes like the atom radius or the hydrophobicity stored in the component file. To add these attributes without the need to change each of the components manually, a component update method has been defined. The only method parameter is a file. This file has to be a CSV [5] file formatted as the example in listing 3.2.

```
chem_comp_id;atom_id;Radius_MSMS;Hydrophobicity_Ghose_Crippen
ALA; C  ;1,7;-0,1002
ALA; CA ;2,37;0,3663
```

Listing 3.2: Example for component update CSV file.

- The first two columns must always be the chemical component id and the atom id.
- All other columns can correspond to any *numerical* attribute in the ChemCompAtom table.

3.3.2 Unit configuration

To define which asymmetric units form a biological unit the unit partition of a structure can be set. One partition has to include all chains and hetero

[4]The component file can be found at
ftp://ftp.rcsb.org/pub/PDB/data/monomers/components.cif
[5]CSV...**C**omma **S**eperated **V**alues

components. Each asymmetric unit has to belong to a biological unit and each biological unit has to have at least one asymmetric unit. A simple way to change the partition of a structure is provided by the changePartition function of the PPIXStructService Session Bean.

The partition string parameter holds the new partition configuration. The unit and the chain separators are defined as global constants. The characters used for the unit separator is a space (' '). The characters used for the chain separator is a semicolon (';'). For the example in figure 3.2 the definition of the partitions are given by 'A B C D' (4 units), 'A;C B;D' (2 units) and 'A;B;C;D' (1 unit).

3.3.3 Calculation starter

All time consuming calculations, like the calculation of the surface points and their properties, are going to be executed on a computer cluster. The interaction with the cluster will be done through an already existing Java API developed by G. Stocker. A calculation job is started by sending a message to the calculation message queue. The message is received and processed by the calculation message driven bean. It has to be an object message containing a UserCalculationMessage object. (see app. B)

A UserCalculationMessage contains a list of structure Ids and a list of calculation jobs. The message can contain one or more jobs. One calculation job includes a calculation key, a parameter string and a list of entry keys, the calculations should be performed on.

A calculation entity has fields for a name, default parameters and a class name. An object of the the given class name will be created and its run function will be executed using Java reflection [32].

The implementation of the interaction between the application server and the cluster using the cluster API is part of M. Jorde's Master thesis [49]. The sketched calculation interfaces may therefore be changed if needed.

3.3.4 Analysis

Analyses management is structured similar to calculation management. There is an analysis table containg all defined analyses, an analysis job table and an analysis user message class. Analyses will be implemented in the course of another Master thesis [49].

3.3.5 Job management

Up to now there are three kinds of jobs defined in the application. *Parser-*, *Calculation-* and *Analysis Jobs*. All of them are started by sending an object message to a message driven bean. For each of this three different jobs there is also a job table. In this job table the status and the progress of the job is stored.

Possible status values are:

scheduled job is scheduled for later processing

running job is currently processed

abort job is marked for abortion by a user

aborted job has been aborted by a user

finished job has finished

Each of the three tasks has a Session Bean defined, which has methods to start a job by sending a user message to the message driven bean. Furthermore they have a method to abort a job.

The abortion of the job is done by setting the status to 'abort'. The working function of the job has to regularly check for the status and stop the job, if it has been marked for abortion. The working function also has to update the progress of the job. In terms of the import and the calculation job this is defined by the number of the structures scheduled and number of the entries finished. In case of an analysis it can be expressed as a percentage. Required table fields for the progress and the status of jobs are present in each job table. The working function or the message driven bean has to update the job status, when it starts, succeeds, fails or aborts.

In the importer job the update of the progress and status is done by the message driven bean.

3.3.6 Exporter / Importer

Importer and exporter functions are also necessary for calculation results and calculation inputs. These have to be specified for each calculations. Exporters may be used externally and not only for a calculation. The exporter Session Bean provides a method to retrieve all defined exporters.

The exporters to be found by this function have to be part of the package at.tugraz.genome.PPIX.exporter.functions. (see app. B)

Because of the large amount of data, which will be exported and imported into the database, it is strongly recommended to access the database using JDBC and not through entity beans. A simple example of an exporter, that creates an input file for a surface calculation program called Maximal Speed Molecular Surface (MSMS) [50], has been implemented.

3.3.7 Logging facility

Besides the application server log, which uses log4j [51] there is also a PPIX logging facility defined. This logger first uses the log functionality of log4j and then writes the line in a database log table using an entity bean.
Inserting the log message into the database could also be done by log4j, but the log table stores some extra information. It has foreign keys to the three job tables (see sec. 3.3.5) and an additional field for the PDB Id related to the log entry. These are the only tasks that should use this logging facility.
The foreign keys to the job tables are marked as 'on cascade delete', thus the log entries of a certain job get deleted by deleting the job.
The log level can be set by the global constant

```
PPIX_PARSER_LOGLEVEL = org.apache.log4j.loglevel.INFO
```

independently from the log4j log level in the GlobalConstants.java file. The levels correspond to the log4j log levels. The default value is 'info'. The log level should not be set to debug, because it will seriously slow down the application due to heavy database access.

3.4 PPIX Web Application

The PPIX Web front end is based on an AndroMDA template project, that has evolved from the MARS [6] application [46]. This includes the style sheets, the menu and the usermanagement. The PPIX Web interface makes heavy use of so called Report Beans, which generate a browse-, search-, and editable web view for a given table. In addition to the JSP pages, also the code behind is generated in form of Struts Action Classes and Report Session Beans.

[6]MARS...**M**icroarray **A**nalysis and **R**etrieval **S**ystem

3.4.1 Extended Search

A Web page that has been generated by a Report Bean is the 'Extended Search' page (see fig. 3.4). It is based on an Oracle View, which combines the most important attributes of a protein structure.
A report has a header (1) with a caption and task icons. Among these icons following tasks[7] are defined:

Query (2) displays an additional query table (4) between the result list and the header.

Edit Display Setting (3) displays an additional table (5) under the query table, where the user can select which fields should be displayed from all in the report available fields.

Refresh refreshes the current page.[8]

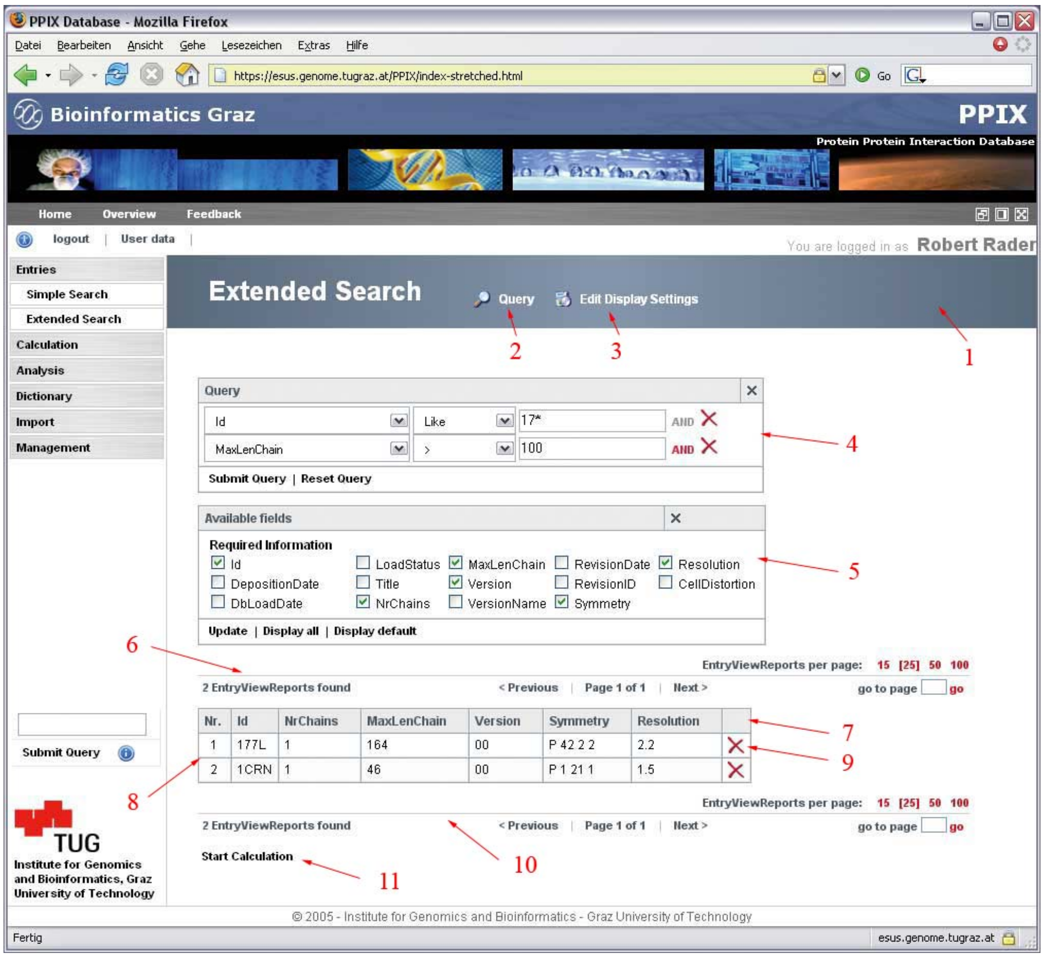
Below the 'Display Settings' table (5) there is a navigation header (6), that contains navigation information and links. The information includes the number of the found items, the current page number, the number of total pages and the number of items displayed by page.

Below the navigation header there is a list header (7), that describes which field is in which column. The fields in the list can be selected in the 'Select Display Settings' table. By clicking on a column header, a user can change the field and the direction that the list is sorted by.

The list itself (8) includes the selected fields and at the very right additional task symbols to delete (9), view or edit the entries.
Below the list there is a navigation footer (10), that looks the same as the navigation header.
On the extended search page there is an additional button (11) to start calculations (see sec. 3.3.3) for all currently found structures.

When a user visits the page using the menu, no search criteria are defined and all additional tables are hidden. As no search criteria are defined, all structures of the database will be found and the default display settings will be used. A user can then open the additional tables, change the displayed fields and search for different parameters.

[7]not all tasks are available for every report
[8]only present at reports about job tables


PPIX Database - Mozilla Firefox

Datei Bearbeiten Ansicht Gehe Lesezeichen Extras Hilfe

https://esus.genome.tugraz.at/PPIX/index-stretched.html

Bioinformatics Graz PPIX

Protein Protein Interaction Database

Home Overview Feedback

logout | User data |

You are logged in as Robert Rader

Entries
- Simple Search
- Extended Search

Calculation
Analysis
Dictionary
Import
Management

Extended Search — Query — Edit Display Settings

Query

Id	Like	17*	AND
MaxLenChain	>	100	AND

Submit Query | Reset Query

Available fields

Required Information

✓ Id	☐ LoadStatus	✓ MaxLenChain	☐ RevisionDate	✓ Resolution
☐ DepositionDate	☐ Title	✓ Version	☐ RevisionID	☐ CellDistortion
☐ DbLoadDate	✓ NrChains	☐ VersionName	✓ Symmetry	

Update | Display all | Display default

EntryViewReports per page: 15 [25] 50 100

2 EntryViewReports found < Previous | Page 1 of 1 | Next > go to page go

Nr.	Id	NrChains	MaxLenChain	Version	Symmetry	Resolution
1	177L	1	164	00	P 42 2 2	2.2
2	1CRN	1	46	00	P 1 21 1	1.5

EntryViewReports per page: 15 [25] 50 100

2 EntryViewReports found < Previous | Page 1 of 1 | Next > go to page go

Start Calculation

Submit Query

TUG Institute for Genomics and Bioinformatics, Graz University of Technology

© 2005 - Institute for Genomics and Bioinformatics - Graz University of Technology

Fertig esus.genome.tugraz.at

Figure 3.4: PPIX Extended Search Page with opened 'Query' and 'Select Display Settings'. This Report Page contains a header (1), a 'Query' (2) and a 'Edit Display Settings' (3) task link, opened 'Query' (4) and 'Display Settings' (5) tables, and the item list(8) with a navigation header(6) and footer (10), a list header (7) and symbols to delete an item (9).

The query tab allows to search for database fields using different relational operators.[9] Different queries can be logical 'and' combined. Logical 'or' combinations can be done using the 'in' and 'not in' operators. In the search field asterisk characters are allowed, like '?' for one character or '*' for any number of consecutive characters.
In the extended search page there are currently 25 different parameters defined, which can be searched for. Among these are the PDB Id, the title, the

[9]The currently specified operators are: 'like, 'not like', >, >=, <>, <=, <, between, not between, in, not in'.

number of chains, the structure type, the cell constants and many more. For a full list of searchable fields have a look at the URD (see app. A).

Technically database queries are done using the JDBC API. The primary keys of the matching database entries are stored in a list. The corresponding entity beans will be fetched only if the item is currently displayed on the Web page.

3.4.2 Quick / Simple Search

Quick Search is an easy to use additional search possibility. The user enters one search phrase, that will compared to the VersionId[10] format. The search can be accessed either from any page using the field right under the menu on the left (see fig. 3.4), or by selecting the 'Simple Search' menu item.

The result will be displayed in an extended search page.

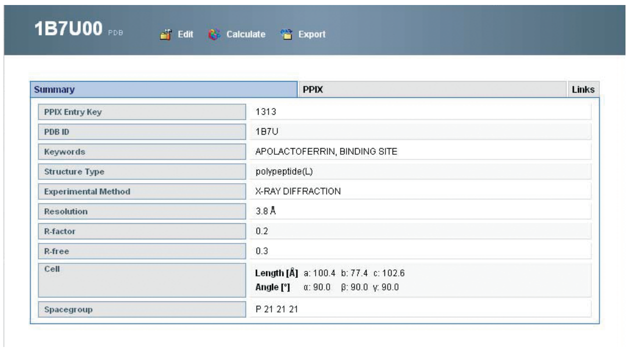
3.4.3 Protein View

The main view of one protein is divided into three tabs. All of them have a common header, which contains the PDB Id including a two digit PPIX version suffix. Right beside the Id there is a link to structure information at the PDB. Then there are three task links. The first to edit the structure, the second to start a calculation and the third to export it. By pressing the export link a list of existing exporters is displayed. When one of the exporters is selected, the export is done on the fly and the result can be saved on the local computer.

[10]The PPIX VersionId consists of the PDB Id and a two digit version number.

Main View

In the main view the most important general information of the protein is displayed. This includes the Id, the title, the structure type, keywords, the experimental method, r-factor, the resolution, the cell constants and the spacegroup (see fig. 3.5).

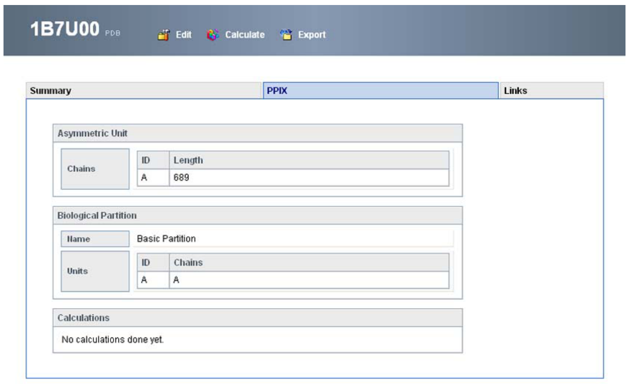


PPIX Entry Key	1313
PDB ID	1B7U
Keywords	APOLACTOFERRIN, BINDING SITE
Structure Type	polypeptide(L)
Experimental Method	X-RAY DIFFRACTION
Resolution	3.8 Å
R-factor	0.2
R-free	0.3
Cell	**Length [Å]** a: 100.4 b: 77.4 c: 102.6 **Angle [°]** α: 90.0 β: 90.0 γ: 90.0
Spacegroup	P 21 21 21

Figure 3.5: Summary View of a protein structure

PPIX View

The second tab named 'PPIX' holds information about asymmetric units, chains, hetero components and their partitioning into biologic units. In addition all calculations, that have been performed for this structure, are displayed (see fig. 3.6).



Asymmetric Unit

Chains

ID	Length
A	689

Biological Partition

Name: Basic Partition

Units

ID	Chains
A	A

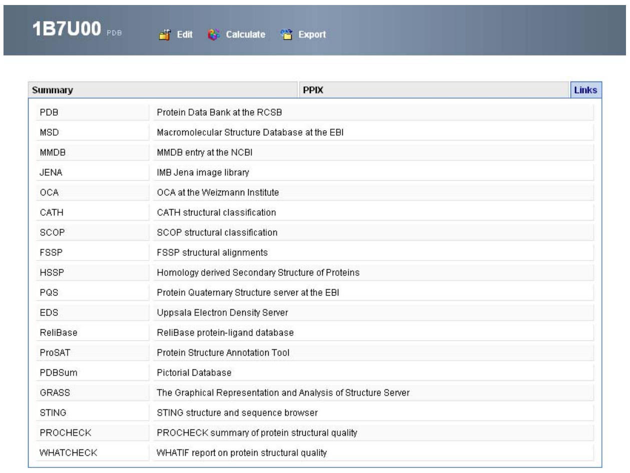
Calculations

No calculations done yet.

Figure 3.6: PPIX View of a protein structure

Links

The 'Links' tab contains a number of links to other databases. The links do not target at the main page of the databases, but at the information for the currently viewed structure (see fig. 3.7).



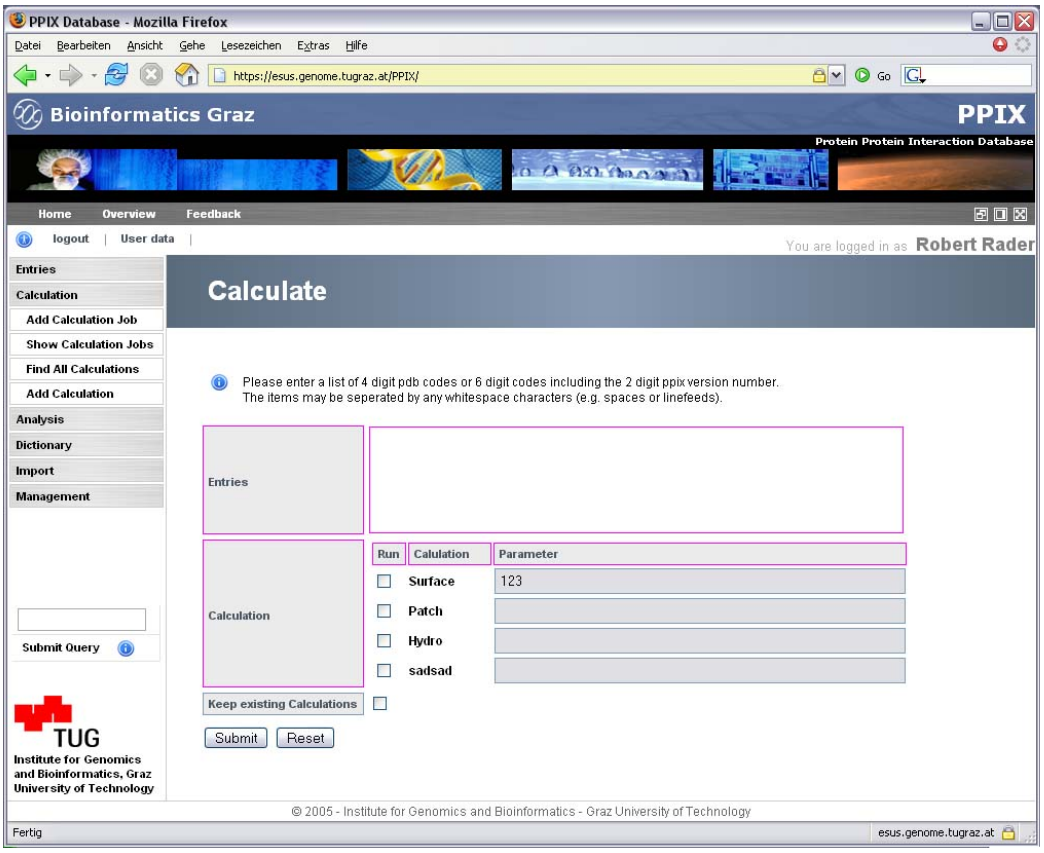
PDB	Protein Data Bank at the RCSB
MSD	Macromolecular Structure Database at the EBI
MMDB	MMDB entry at the NCBI
JENA	IMB Jena image library
OCA	OCA at the Weizmann Institute
CATH	CATH structural classification
SCOP	SCOP structural classification
FSSP	FSSP structural alignments
HSSP	Homology derived Secondary Structure of Proteins
PQS	Protein Quaternary Structure server at the EBI
EDS	Uppsala Electron Density Server
ReliBase	ReliBase protein-ligand database
ProSAT	Protein Structure Annotation Tool
PDBSum	Pictorial Database
GRASS	The Graphical Representation and Analysis of Structure Server
STING	STING structure and sequence browser
PROCHECK	PROCHECK summary of protein structural quality
WHATCHECK	WHATIF report on protein structural quality

Figure 3.7: Links for a protein structure

Edit View

There are only three protein attributes that can be changed manually for a structure. The first is the version name. There can be more than one version of a protein in the PPIX Database. To distinguish the versions a version suffix is added to the PDB Id. The version name should help to define what is different between the versions.

The second editable entity is the unit partition. Later in the development of the application the unit partition should be computed automatically. This page gives the possibility to change it manually. The changePartition function (see sec. 3.3.2) is called with the given string.
As a last option a remark about the crystallisation conditions for the structure can be changed. It is a free text field that can be used to store additional information about the experiment used to crystallise the protein.


PPIX Database - Mozilla Firefox
Datei Bearbeiten Ansicht Gehe Lesezeichen Extras Hilfe
https://esus.genome.tugraz.at/PPIX/
Bioinformatics Graz
PPIX
Protein Protein Interaction Database
Home Overview Feedback
logout | User data |
You are logged in as Robert Rader
Entries
Calculation
Add Calculation Job
Show Calculation Jobs
Find All Calculations
Add Calculation
Analysis
Dictionary
Import
Management
Submit Query
TUG
Institute for Genomics and Bioinformatics, Graz University of Technology
Calculate
Submit
Reset
Fertig
esus.genome.tugraz.at

Please enter a list of 4 digit pdb codes or 6 digit codes including the 2 digit ppix version number.
The items may be seperated by any whitespace characters (e.g. spaces or linefeeds).

Entries			
Calculation	Run	Calulation	Parameter
	☐	Surface	123
	☐	Patch	
	☐	Hydro	
	☐	sadsad	
Keep existing Calculations	☐		

© 2005 - Institute for Genomics and Bioinformatics - Graz University of Technology

Figure 3.8: Add Calculation Job Webpage

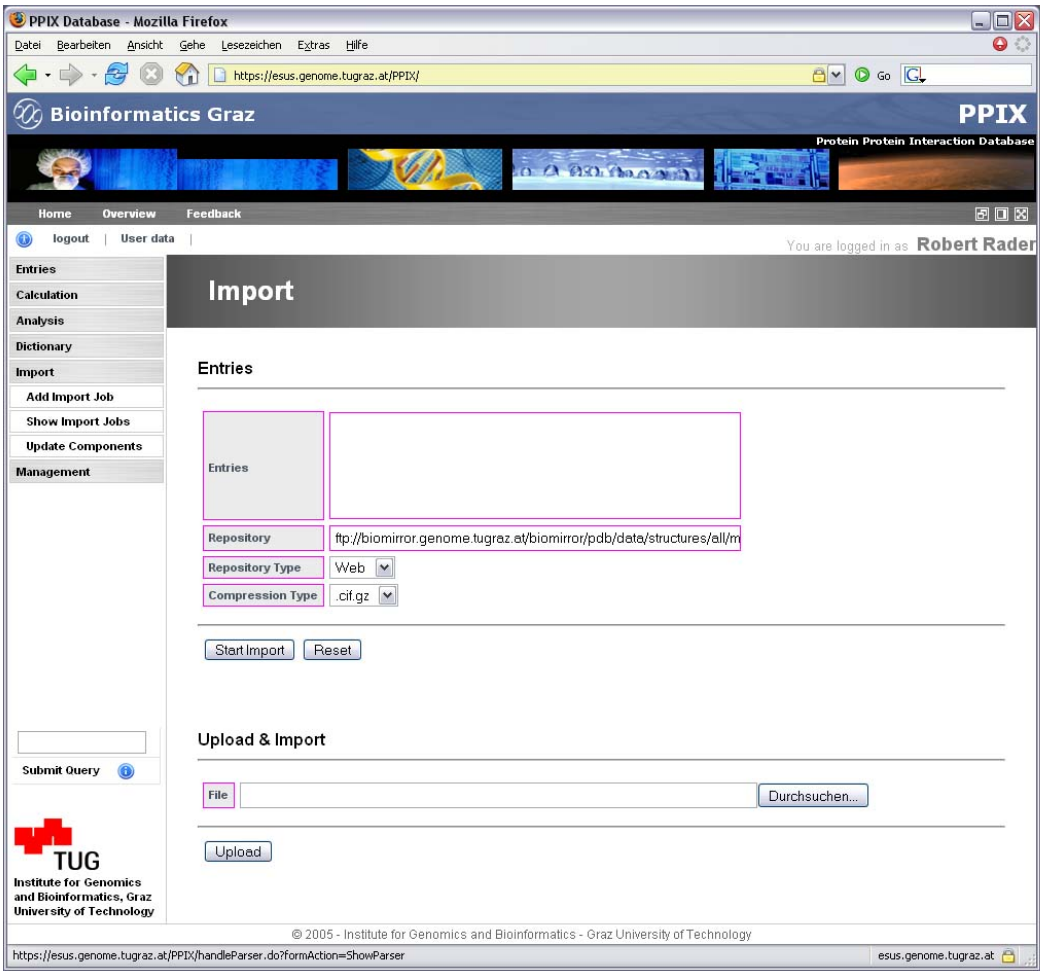
3.4.4 Calculation

The next top label menu item named calculation contains four subitems. These items allow to start calculations, view and edit calculation definition and to watch the progress of running and scheduled calculations.

Add Calculation Job

Calculations can either be started by the calculation task bar link in one of the three protein views, by the 'start calculation' button on the 'Extended Search' view or by the 'Add Calculation Job' menu item. The last option simple opens the add calculation job page, the seconds automatically writes the VersionId[11] into the list of IDs, for which a calculation should be done. The third option adds all proteins to the list which match the current search options. If the number of found items is higher than 50, instead of the whole

[11]The PPIX VersionId consists of the PDB Id and a two digit version number.



PPIX Database - Mozilla Firefox
Datei Bearbeiten Ansicht Gehe Lesezeichen Extras Hilfe
https://esus.genome.tugraz.at/PPIX/
Bioinformatics Graz
PPIX
Protein Protein Interaction Database
Home Overview Feedback
logout | User data |
You are logged in as Robert Rader
Entries
Calculation
Analysis
Dictionary
Import
Add Import Job
Show Import Jobs
Update Components
Management
Submit Query
TUG
Institute for Genomics and Bioinformatics, Graz University of Technology
Import
Entries
Start Import
Reset
Upload & Import
Durchsuchen...
Upload
© 2005 - Institute for Genomics and Bioinformatics - Graz University of Technology
https://esus.genome.tugraz.at/PPIX/handleParser.do?formAction=ShowParser
esus.genome.tugraz.at

Entries	
Repository	ftp://biomirror.genome.tugraz.at/biomirror/pdb/data/structures/all/m
Repository Type	Web
Compression Type	.cif.gz

File	

Figure 3.9: Import Web Page

list, only the number of selected items is displayed.
The user can define the calculations and their parameters as well as whether already calculated results should be overwritten or not.

3.4.5 Import

Under the menu option 'Import' there are two pages which start a parser job or an update. In both cases the action class validates the data entered by the user and accesses, if everything is correct, the Parser Session Bean using the BusinessInterface to start the import.

PDB Import

This page is a front end for the functionality specified in the PDB Importer business logic (see sec. 3.3.1). The data entered on the Web page (see fig. 3.9) is validated before being processed. The list is checked to be a whitespace separated four digit list. The filename given in the 'Upload & Parse' section is checked for existence. As default repository the local Biomirror [52] instance at the TU Graz [53] is used.

Components Update

This page gives simple access to the component update function (see sec. 3.3.1).

3.4.6 Dictionaries

There are four tables that get referenced by every structure during the import. These tables are ChemComp, ChemCompAtom, Spacegroup and AtomTypes. Reports have been generated to view and edit these tables.

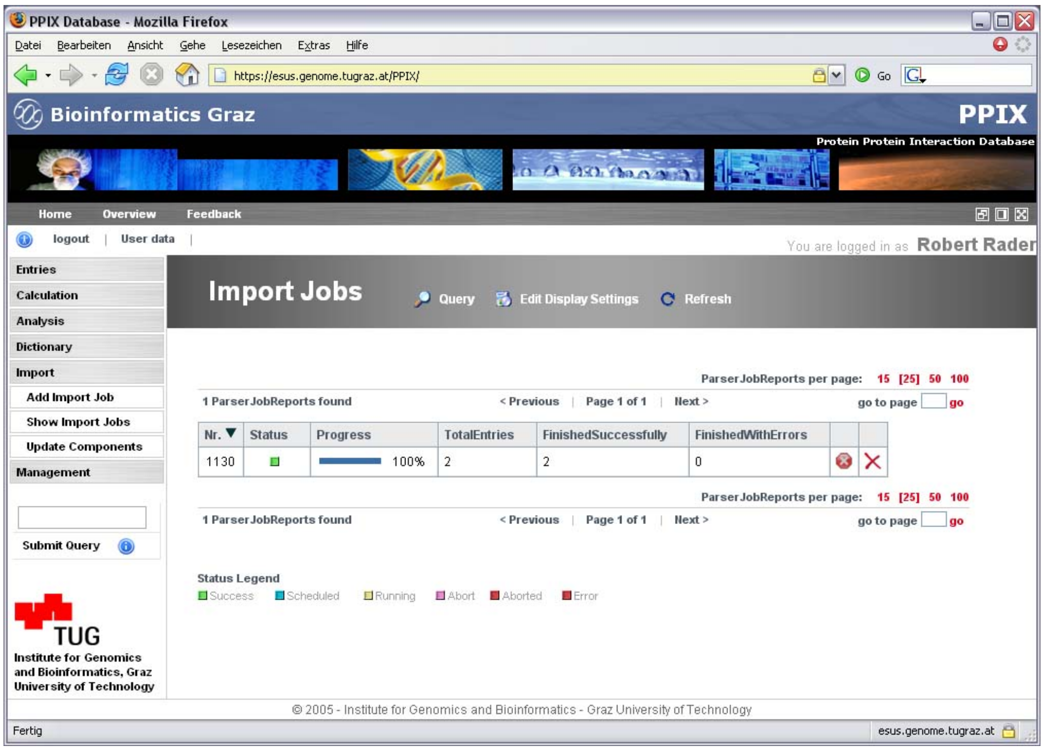
The generated reports are

Chemical Components By clicking on a chemical component, the user gets to a report on the atoms of the selected component.

Spacegroups gives access to all defined spacegroups and their attributes.

Atom Types is a simple list of atom types.

Database Links stores the databases that are linked for each structure (see sec. 3.4.3).


PPIX Database - Mozilla Firefox
Datei Bearbeiten Ansicht Gehe Lesezeichen Extras Hilfe
https://esus.genome.tugraz.at/PPIX/
Go
Bioinformatics Graz
PPIX
Protein Protein Interaction Database
Home Overview Feedback
logout | User data |
You are logged in as Robert Rader
Entries
Calculation
Analysis
Dictionary
Import
Add Import Job
Show Import Jobs
Update Components
Management
Submit Query
TUG
Institute for Genomics and Bioinformatics, Graz University of Technology
Import Jobs
Query
Edit Display Settings
Refresh
Fertig
esus.genome.tugraz.at

ParserJobReports per page: 15 [25] 50 100

1 ParserJobReports found   < Previous | Page 1 of 1 | Next >   go to page go

Nr. ▼	Status	Progress	TotalEntries	FinishedSuccessfully	FinishedWithErrors		
1130	🟩	100%	2	2	0	⊗	✕

ParserJobReports per page: 15 [25] 50 100

1 ParserJobReports found   < Previous | Page 1 of 1 | Next >   go to page go

Status Legend

🟩 Success 🟦 Scheduled 🟨 Running 🟪 Abort 🟥 Aborted 🟥 Error

© 2005 - Institute for Genomics and Bioinformatics - Graz University of Technology

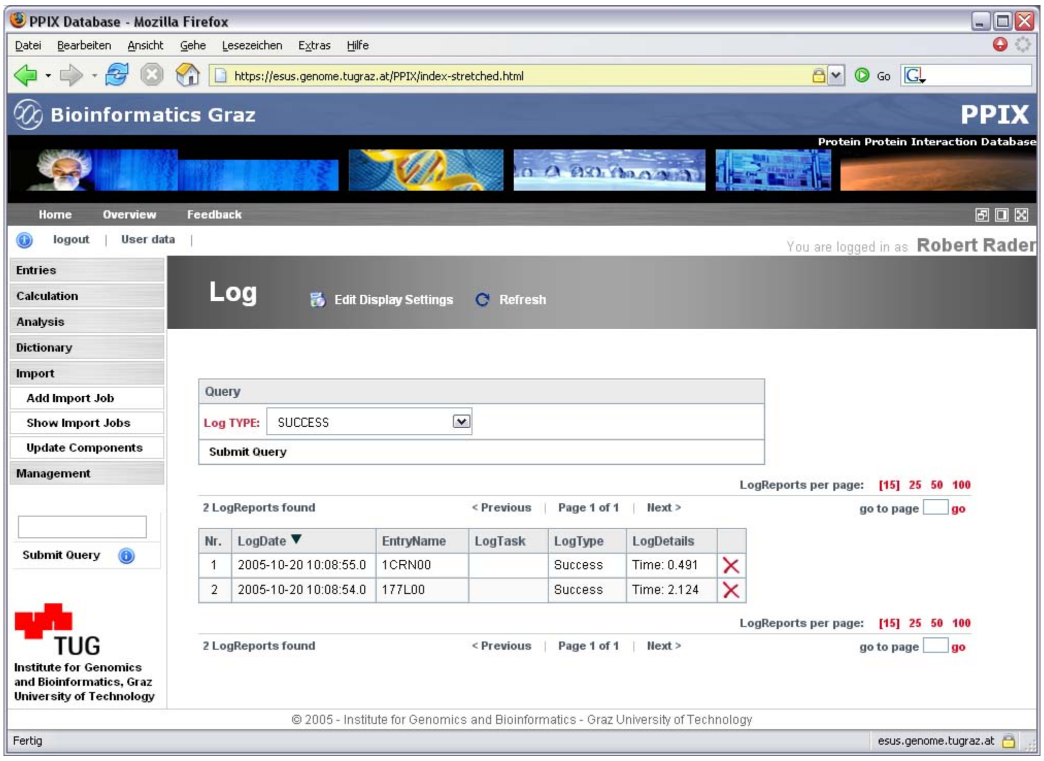
Figure 3.10: Job Viewer Web Page

3.4.7 Job & Log Viewer

The status and the log of a task can be viewed on the Job View page. For each of the three defined tasks (import of structures, calculations and analyses) a Report has been generated. These reports have been enhanced by a status icon and a progress bar (see fig. 3.10). By clicking on one job, the user gets to the log report for the selected job. The only accessible search option in the log report is the log level (see fig. 3.11). The full log report without a preselected job and with all search options enabled can be viewed by an application administrator using the menu item Management / Logs.

3.4.8 Overview

The page gives an overview about two different topics. One contains a statistic about the database contents, the other one informs about the status of the application server. The status of the database includes the number of structures, units, chains, atoms, chemical components, surface points and patches. For each defined calculation the number of structures, which the calculation has been performed on, is given.


PPIX Database - Mozilla Firefox
Datei Bearbeiten Ansicht Gehe Lesezeichen Extras Hilfe
https://esus.genome.tugraz.at/PPIX/index-stretched.html
Go
Bioinformatics Graz
PPIX
Protein Protein Interaction Database
Home Overview Feedback
logout | User data |
Entries
Calculation
Analysis
Dictionary
Import
Add Import Job
Show Import Jobs
Update Components
Management
Submit Query
TUG
Institute for Genomics and Bioinformatics, Graz University of Technology
Log
Edit Display Settings
Refresh
Fertig
esus.genome.tugraz.at

You are logged in as **Robert Rader**

Query

Log TYPE: SUCCESS

Submit Query

LogReports per page: [15] 25 50 100

2 LogReports found < Previous | Page 1 of 1 | Next > go to page go

Nr.	LogDate ▼	EntryName	LogTask	LogType	LogDetails	
1	2005-10-20 10:08:55.0	1CRN00		Success	Time: 0.491	✗
2	2005-10-20 10:08:54.0	177L00		Success	Time: 2.124	✗

LogReports per page: [15] 25 50 100

2 LogReports found < Previous | Page 1 of 1 | Next > go to page go

© 2005 - Institute for Genomics and Bioinformatics - Graz University of Technology

Figure 3.11: Log Viewer Web Page

3.5 Schema2XMI Graphical User Interface

The development of the application was based on the existing OpenMMS database schema. To avoid the time-consuming and error-prone work of defining all tables manually in the UML diagramm, Schema2XMI (see sec. 2.5.2) was used.
To ease the use of the tool, a graphical user interface (GUI) was implemented. In addition to creating the GUI, the code was extended to facilitate the work with the AndroMDA cartridges developed by T. Truskaller. Currently supported databases include Oracle, MySQL and PostgreSQL. All databases that provide a JDBC driver can be supported by the tool. Beside the driver also a XML data type mapping file needs to be defined, that describes how to convert database data types to Java data types.

3.5.1 Extensions to Schema2XMI

The extensions include the possibilities to

- create a value object for each created entity.
- create a dependency stereotyped as an 'EntityRef' to a default session service for each entity.
- create a dependency stereotyped as an 'ExceptionRef' to a default exception for each entity.
- tag each table attribute (@ejb.persistence tag) and all foreign key associations (jboss.relation.fk-column tag) to keep their database names.
- create a foreign key for all tables, which do not have one or have a combined primary key.

A small change was also made to the header of the generated XMI file to avoid a warning by MagicDraw 8.0.

Using this changes it is, with a few restrictions (as not all features of the cartridges have been adopted), possible to access the database as a J2EE persistence source without any coding work.

The necessary steps are:

- take an existing database schema[12].

[12]The schema has to be installed on a database server.

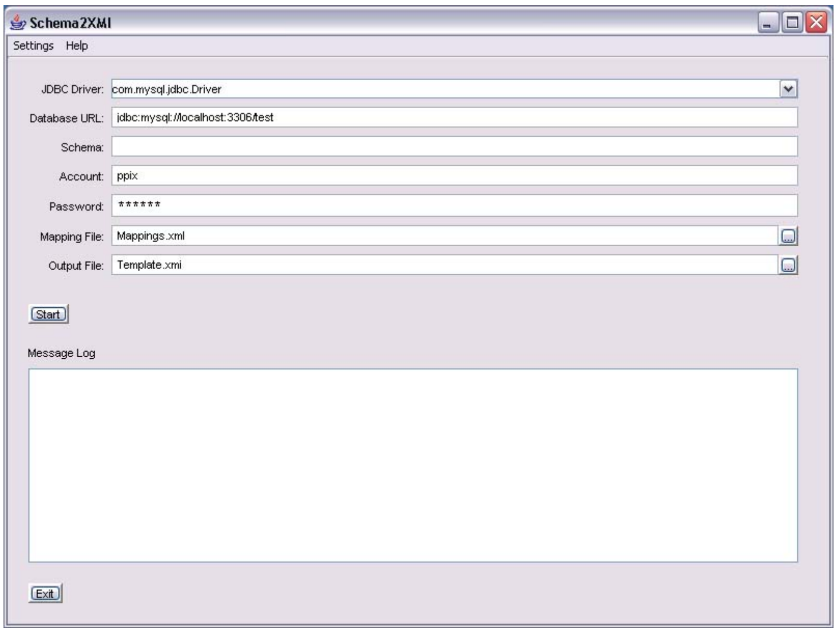
- run schema2xmi to get a UML model.
- generate the source with AndroMDA and XDoclet.
- deploy the generated project on a JBoss server.

The tool can also help in the reengineering of existing projects to AndroMDA based UML applications.

3.5.2 Graphical User Interface

The graphical user interface (see fig. 3.12) is kept rather simple. Only the most important configuration features are made accessible. All other necessary parameters are configured for the template project and stored as global constants. At the bottom half of the application a log window is displayed, which uses a log4j appender to write log messages of the schema transformer to the panel.

In addition to the database connection driver name, URL, user name and password, the user has also to specify the mapping file, that is used to map the database fields to Java data types and the template XMI file, that should be used as a UML base model.



JDBC Driver:	com.mysql.jdbc.Driver
Database URL:	jdbc:mysql://localhost:3306/test
Schema:	
Account:	ppix
Password:	******
Mapping File:	Mappings.xml
Output File:	Template.xmi

Figure 3.12: Graphical User Interface for Schema2XMI

3.6 Extentions to the AndroMDA cartridges

During the development of the PPIX application some issues arose, that were not supported by the AndroMDA cartridges. Some of the issues were solved by implementing the needed functionality in the cartridges.

The support for specifying *database attribute names for foreign keys* was added. The cartridges generate names for foreign key attributes based on names of the linked table. This would break the naming schema of a present database. The possibility to define the foreign key attribute name was implemented using the tagged value '@jboss.relation.fk-column' in the Velocity file generating the Entity Bean classes. The tag is then processed by XDoclet to specify the foreign key attribute name in the JBoss configuration files. This was used to keep the database table and attribute names of the OpenMMS schema. This also applies when an application already in use with an existing and populated database should be adapted to an AndroMDA based project.

The support for *more than one foreign key between two tables* has been added by a tagged value named '@nameSuffix' for association in the UML element. This suffix is added to the Entity Bean and Value Object method and attribute names. Without this change two foreign key references between the same tables would lead to duplicated method names. This change affects the Velocity files for the Value Object and the Entity Bean generation.

An extended sorting feature for Report Beans has been implemented that allows to sort ascending and descending for every displayed column. It also supports the definition of a primary sort column, which is used to sort the results when the report page is opened. To implement this feature changes have been made to the GlobalConstants, the 'ReportAction' and the Report JSP Velocity files.

A function to open a report page with a predefined query has been added to the 'ReportAction' Velocity file . This feature is used for example by the 'Log Report' to access log entries for a specific job (see sec. 3.4.7).

A feature to define the container configuration of a message driven bean has been implemented. The support of a '@jboss.container-configuration' tag was added to the message driven bean Velocity file. This is used to specify the the 'import message driven bean' (see sec. 3.4.4) as a Singelton class.

Chapter 4

Discussion

During this thesis the core of the PPIX application was developed. The core consists of the database, the Web front end, the basic business logic including an import function for mmCIF files and interfaces for calculations and analyses.

The application has to serve some rather diverse requirements. It has to be able to contain a vast amount of data[1], but it should also be fast (e.g. for search operations), robust, failsafe, portable, extendable and simple.
For a failsafe and comfortable access to the database, especially by a web client, entity beans are used. For fast data retrieval, like the export of structures or the search queries, JDBC is used. Also the import uses direct JDBC access to the database.
To implement the application the state of the art software technologies J2EE and MDA have been used, which makes the project easy to maintain and extend. Task management has been implemented for the import of PDB structures, for calculations and for analyses. These are often long lasting tasks, which run asynchronously and only update their status in a job table. For heavy weight calculations an interface to a computing cluster is used. The application is currently used by the 'Institut für Physikalische Chemie (Subgruppe Strukturbiologie)', where further calulations and analyses are currently developed.

Schema2XMI is a helpful tool that facilitates creating the basic UML document from an existing database schema or application with a database persistence. A good case tool (but sadly in most cases not in the free version) should be able to perform most tasks of Schema2XMI, the import of a database schema, the automatic tagging of entity tables or the batched gen-

[1]as estimated in the user requirement definition it will be about 125GB, depending on the resolution used to calculate the surface.

eration of associations to service entities. Nevertheless using Schema2XMI relieves the application designer from a lot of working steps, which otherwise have to be done manually. It integrates well with the existing AndroMDA environment utilized at the Institute.

4.1 Improvements and Future Development

The very next step in the PPIX project will be the integration of already developed calulations through the cluster interface by M. Jorde.

As seen in the testing phase the deletion of a structure takes a lot of time. When deleting one structure, a lot of foreign relation have to be resolved and the data in the foreign tables has to be removed. Although this is done on the database side, it takes about 20 seconds for a structure with no calculated surface. This twenty seconds are on the edge of a user's patience and it might make sense to run the deletion of structures asynchronously.

The functionality of reports could be extended to support the selection of multiple rows / entries. As currently implemented a task (e.g. a calculation) can only be started for one structure, for a manually given list of structures or for a list returned by a query. It would be a nice feature, if items could be selected from the list and then tasks like calculations and analysis could be performed for the selected subset.

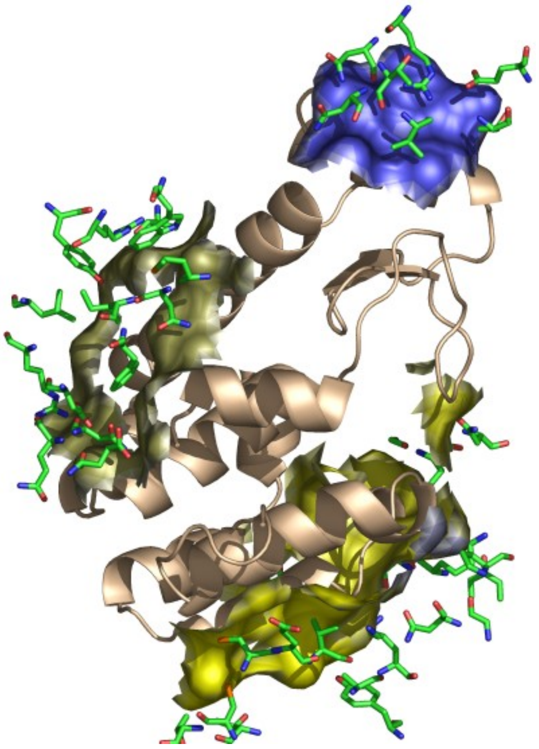
Appendix A

User Requirement Document

PPIX

Protein-Protein Interaction in Crystals

User Requirements



Version 0.7 (2005-04-18)

Markus C Jorde
mjorde@sbox.tugraz.at

1 Introduction

1.1 Project Purpose

The purpose of the PPIX – Protein-Protein Interaction in Crystals – system is to aid in the exploration of the interaction between protein-protein contacts in crystal structures.

Ultimately the understanding of the structural requirements for the formation of crystal contacts might help to design protein mutants with significantly improved crystallization properties.

1.2 Project Aims

Its aim is to design and implement a (database) system which provides the basic functionality to store and manage all relevant data pertaining to all proteins whose crystal structure is known.

The existing protein structures will be imported (from the PDB) into the system and based on their data a number of calculations will be performed to generate missing protein properties relevant for subsequent analyses. Such properties include the protein surface and the various properties specific to surface points, e.g. hydrophobicity, electrostatic potential, amongst numerous others.

The system will also calculate the contact (=interaction) surfaces – patches – between proteins, which will constitute the primary element of the later analyses.

Once the required data has been generated, the system will be able different analyses on the stored data. These analyses will include the assessment of the relevance of the different surface properties for the formation of protein-protein contacts. Attention will also be directed to the statistical evaluation of the resulting data.

The system will provide functionality to visualize the data and the results of these analyses, including the 3-dimensional (interactive) display of structures with different user-selectable annotations, such as the coloration of the contact surface points with respect to their various properties.

To achieve these aims existing applications will be integrated into the system, if the required functionality is already available. Otherwise software tools will be developed for each of the specific tasks and integrated into the system.

Due to the nature of this project and its scientific research background, certain requirements and aspects of the system, and the specification of its aims can and likely will change during the progression through the various development stages.

As such the systems architecture has the to be designed with this in mind, in a flexible and open way to allow for non-resource-intensive adaptations of the system.

2 User Requirements (URD)

2.1 Database

2.1.1 Database Structure

The base of the PPIX system is the database which stores the protein structures and all their associated information, including the results of the calculations which are performed by the system.

2.1.2 Database Size

The database should be able to store (at least) the number of entries currently contained in the PDB. There are currently approx. 10000 structures of interest (which currently excludes all NMR and duplicate similar structures) in the PDB.

The main storage capacity will be required for the positional data of the structures' atoms and the surface point data. Partitions will be stored as additional versions. The average number of versions will be 3 per structure.

The average number of atoms per entry is 6000 and the estimated average number of surface points per surface (with a resolution of 1.0 points per sq angstrom) is 30000.

Additionally the data for the estimated 30000 surface triangles per structures has to be stored.

2.1.2.1 Detailed Memory Requirements

Based on the above average values and the following database type size assumptions:

- 4 Byte per Integer/Number
- 8 Byte per Float and
- 1 Byte per Char

the database will have a storage requirement of approximately 125 GB.

This value does not include any database specific overheads, including but not limited to table indeces.

2.1.3 Database Import

There are 2 different methods to import structures into the system. Before the structure is imported into the system, certain checks are made to test if the structures meets the import requirements.

2.1.3.1 Import (based on 4 letter structure identifiers)

The system can be instructed to import (new) structures by specifying a single or multiple 4 letter structure identifiers. The system will automatically retrieve the necessary mmCIF file(s).

2.1.3.2 mmCIF Upload

The user can upload a new structure by providing a compatible mmCIF file. These structures have to be namend using non-used PDB entries codes. Codes should begin with 'x' to avoid accidental entry name collision with existing PDB entries.

This upload provides the possibility to import new/unpublished or modified structures into the system.

2.1.4 Database Import Requirements

Before a structure is imported into the system, it has to meet a number of requirements.

In the current version of the system the following requirements are enforced:

1. Only X-Ray diffraction structures (No NMR)
2. Only protein structures
3. No structures of RNA/DNA complexes
4. No incomplete entries
5. No entries with a 'strange' transformation matrix (old entries)

Later versions of the system will also consider entries of RNA/DNA complexes.

Detailed Requirements:

1.	The experimental method field [EXPLT.method] has to contain the Text "X-Ray"
2.	The field [CHEM_COMP.type] must contain the text "L-Peptide"
3.	The field [CHEM_COMP.type] must not contain either • "RNA" • "DNA" • "TNA"
4.	The [CHEM_COMP.ID] field must not contain either • "UNK" • "---" • "..." The spacegroup field [SYMMETRY.space_group_name_H-M] must contain a value. (Incorrect values will be filtered at a later stage) The fields [CELL.length_*] and [CELL.angle_*] must contain valid entries.
5.	The transformation vector fields ([ATOM_SITE.frac_transf_vector*]) have to be [0/0/0].

2.2 User Levels

The system is able to differentiate between 3 different user levels. This provides the functionality to restrict certain functions to specific user levels.

2.2.1 Web User (limited functionality)

The Web User will not require to login but will only have access to a limited number of functions. The Web User will not be able to run resource-consuming analysis or calculations.

2.2.2 User (standard functionality)

This is the stanard user level, which will be used for the main scientific research. The user has access to all functions execpt the dedicated administrative ones.

The user has access to all research functionality including all available search and query functions and resource consuming analyses.

2.2.3 Administrator (administrative functionality)

The administrator level allows the use of all available functions, including database management functions, such as deleting entries or calculations, modifying entries.

2.3 User Interface

The system provides a user interface which is accessible through any standards-compliant web browser. The compatibility of the application with Internet Explorer 6 and Firefox 1.0 has to be checked. The user interface is used to interact with the system und will provide a number of different functions.

2.3.1 Web User Functions

2.3.1.1 Search for Entries

Functionality to search for structures in the database based on the different criteria. Searches return a list of corresponding structure identifiers. Searches will be based on the following fields (which can be combined in a AND/OR fashion):

Query	**Definition**	**Example**
PDB ID	PDB Code or PDB Code list	17rg [17rg,177l,4req]
PPIX ID	PPIX generated Structure Code and Versions	17 2123 12
Titel	Titel of PDB	Hydrolase , Lysozym Mutant

Number of Chains	How many chains in structure	>2, 0 <2
Length of longest Chain	Length of the longest Chain in the structure	<200 >200
Date of PPIX entry	The date at which the structure has been imported into the PPIX system	19.12.2004
Unit Cell: a	Cell dimension length a	<100 1 >200
Unit Cell: b	Cell dimension length a	<100 1 >200
Unit Cell: c	Cell dimension length a	<100 1 >200
Unit Cell: alpha	Cell dimension angle alpha	<60 >60 40
Unit Cell: beta	Cell dimension angle beta	<60 >60 40
Unit Cell: gamma	Cell dimension angle gamma	<60 >60 40
Unit Cell: longest site/ shortest site	Gives a feeling of how "warped" the Cell is (one site is (x) times smaller than one other.	1 >50 >100
Resolution	Resolution of the PDB File in Angstrom	<2.2 > 5.0
Symmetry	H_M symbol of PDB File	P 21 21 21 C 1 2 1
Chemical Comp: Yes	Returns structures that contain these components	001 [001, GLU, HEM] HEM
Chemical Comp: No	Returns structures that do not contain these components	001 [001, GLU, HEM] HEM
Calculation performed	Returns structures for which the calculations has been performed.	
Calculation not performed	Returns structure for which the calculation has not been performed yet.	

2.3.1.2 Visualization

See section 2.4 Visualization

2.3.2 Standard User Functions

2.3.2.1 Database Statistics/Summary

Display a summary of the currently stored entries:

- Total Number of Structures, Amino Acids, Atoms, Surface Points, Patches
- Average Number of Atoms, Surface Points, Patches per Structure
- For each Calculation:
  - Number completed
  - Number outstanding
  - Number scheduled

2.3.2.2 Display summary for a single entry

1. General Information (Date of Entry etc...)
2. Chain Information: (for each chain) Name, Length
3. Space Group Symmetry, Cell Constants
4. Type of Entry
5. Number of Chains, Amino Acids, Atoms, Surface Points, Patches
6. Calculations completed
7. Calculations outstanding

2.3.2.3 Initiate Calculations

This part is used to initiate calculations for a single or multiple structures in the database. The system will automatically take care of the scheduling of the calculation application and the distribution on the computing cluster.

The user can initiate calculations using the following options:

1. select the 'All Entries' option
   or
   specify a single or multiple PDB structure codes
2. select from the available calculations (or select 'All')
3. select whether to delete and recalculate existing properties or to keep them

A list of the calculations which the system will be able to perform in the current is in section 2.5.

Certain conditions have to be checked before a calculation can be performed for an entry. (E.g. to calculate the hydrophobicity values, the surface of the structure has to be calculated first.)

2.3.2.4 Calculation Log

The user interface will provide access to the log of the calculations which will show the following information:

1. The currently active calculations
2. Their current status/progress
3. Potentially failed calculations and the possible reason for this
4. (Recently) completed calculations

2.3.2.5 Edit/Modify attributes

This function allows the user to modify certain attributes of a structure manually.

The following properties can be modified through the user interface:

1. Partitions
   The user can specify the partitions associated with a specific structure. In later versions of the system the partitions will be calculated automatically and manual modification will only be necessary in complicated cases.
2. Crystallisation Conditions (currently as free text)

   The crystallization conditions are not available for most entries in the PDB, and therefore have to be added manually. Future versions of the system will provide functionality to automatically retrieve the crystallization conditions from the BMCD (Bio-Molecular Crystallization Database).

   The analysis of the crystallization conditions and the similarities between certain types of conditions and their capability to facilitate crystallization can be used to increase the chances of the crystallization for certain structures to occur. Additionally a correlation between the crystallization conditions and other attributes of the structure, such as the number of patches, might exist.

2.3.2.6 Export of Entries

The user can export complete entries or parts of (such as surface only) in a number of different formats from the system database. Additional options can be specified to customize the export.

The current version of the system will support the following 2 types of export formats:

1. A PDB-type format that will contain (only) the atom coordinates.
2. A CSV type format that contains the surface points and their stored/calculated property values.

   [surface coordinates: x,y,z; hydrophobicity; electrostatic potential; etc.]

2.3.3 Administrative Functions

The administrative level user can administer the database through the user interface and perform the following tasks:

2.3.3.1 Delete Entries

This function is used to remove a complete entry from the database, including any calculated and manually added or modified attributes.

2.4 Visualization

The system provides means to visualize the protein structures with different (selectable) annotated (colored) surfaces. Annotation and color correspond to the values of various properties of the surface points such as hydrophicity or electrostatic charge.

The visualization is provided through browser plugins and/or external programs. In the current version of the system, only a PyMol export will be supported. In future versions additional visualization options will be supported, such as X3D/VRML2.

2.4.1 PyMol

The visualization in PyMol will work with via a custom export file format.

It will provide the complete structure with color annotated surfaces for each of the units.

The surfaces will be be annotated with the the following properties:

1. Hydrophobicity
2. Electrostatic Potential
3. Patch Number
4. Curvature

For complete support of the export format, a PyMol plug-in will be needed. This plug-in will enable additional functionality which are currently not provide through PyMol directly, such as the handling of the more efficient and flexible export format.

The details of the PyMol export format will be defined later. It roughly consist of the following groups of data: a PDB file like structure for the atom site coordinates, a surface data file with the annotated/calculated properties.

2.5 Calculations

The user can initiate a number of calculation to calculate additional properties for each of the structures and their elements.
The calculations are implemented through external applications, which are executed by the system and automatically provided with the required input data. Upon completion the system processes the output and feeds the results back into the system database.

The following properties can be calculated by the system, partly by using existing applications:

1.	Surface	MSMS
2.	Patches	HydroCalc (inhouse)
3.	Partitions (simple version)	to be developed
4.	Hydrophobicity	HydroCalc (inhouse)
5.	Electrostatic Potential	Delphi, APBS

2.5.1 Cluster

Due to the large number of elaborate and complex calculations which will have to carried out, a cluster will be used to parallelize the calculations and distribute the work load. The parallelization should be transparent to the calculation applications – i.e. the applications are not aware of the parallelization – and the scheduling to the different cluster nodes is performed by the system automatically.

2.6 Analyses

The system provides the functionality to run a number of analyses on the available data.

More complex analyses which require a substantial amount of time to complete, will be scheduled by the system. Upon completion the user will be notified by the system (via Email), and can then view the results online.

2.6.1 Analyses Log

2.6.2 Distribution of Amino Acids in Patches vs. Non-Patches

The aim is to compare the relative distribution of each of 20 different amino-acid in areas of patches and non-patches.

2.6.3 Distribution Amino-Acid Interaction Pairs in Patches

Analyze the interaction partner of each amino-acid which appears in a patch and its respective patch "partner".

3 Future Development

This section lists and describes features and functions of the PPIX system which will be of interest for future version of the system.

3.1 Database Import - AutoUpdate from PDB Server

The system is able to check the PDB Server for new structures in regular intervals and import these automatically. Structures which have been updated and already exist in the system database will be handled differently.

There are 3 types of PDB updates:

- New Entries
  These entries are simply imported into the system.
- Obsolete Entries
  Obsolete structures have been removed from the PDB, for various reason, e.g. They have been replaced by new entries. Obsolete structures will be removed from the system.
- Modified Entries
  These entries already exist in the PDB, but have been modified. Usually these modifications are minor. Care has to be taken not to overwrite user modified attributes such as crystallisation conditions. In ambigious update conditions the system administrator will be notifiied of a modified entry but the structure will not be updated automatically.

3.2 Calculations - Reduction of the crystal symmetry

A number of oligomers build their biological unit through a crystallographic symmetry operation. This can lead to incorrect results in later analyses. Therefore it is necessary to identify the corresponding symmetry operation and create the biological unit through this operation and reduce symmetry accordingly before any further processing is done.

3.3 Calculations - Conservation Score

The conservation score indicates how conserved the alignment of a structure's aminoacids is compared to homologous proteins.

Conserved aminoacids are stable and are unlikely to change due to functionaly reasons. The conservation score might be used to differentiate between biologcial and crystalographic contacts.

3.4 Calculations – Curvature

3.5 Calculations - Patch Properties

3.6 Calculations – Improve Partitioning

3.7 Database Import - Automated Import of BMCD

The system will be able to automatically import data from the BMCD (Crystallisation Conditions) and assign them to the corresponding entries in the PPIX system.

3.8 Visualization - Export: Conservation Score

3.9 Visualization - X3D (VRML 2)

3.10 Analyses - Patch Classification

This analysis will classify patches based on the following criteria:

1. Geometry (Size)
2. Correlation Analysis
3. Properties Mapped on Surface

4 Glossary

Asymmetric Unit (ASU)	The asymmetric ->unit contains the atoms which are necessary to form the whole crystal structure using a given crystal-symmetry operations.
Biological Contacts	->Oligomer contacts (protein-protein contacts) that also occur in solution and not only as a result of the crystallization.
BMCD	Biological Macromolecule Crystallization Database (http://wwwbmcd.nist.gov:8080/bmcd/bmcd.html)
Chime	Browser Plugin for viewing molecular structures in 3D
Component	Amino Acid-Chains or chemical (hetero-component)
Crystal Contact	Protein-Protein contacts which only occur in a crystal as the result of the protein crystallization; these also called non-biological contacts.
DelPhi	Application used to calculate the electrostatic potential of the surface points
Entry	Entry of a molecule in the ->PDB
Entry Code	A 4 Letter Code (e.g. 177l, 4req) which uniquely identifies every ->Entry to the ->PDB.
mmCIF	The new file format for ->PDB entries
MSMS	Maximal Speed Molecular Surface Application used to calculate the surface of a ->unit.
Partition	A partition contains all separate ->units, which are either defined by the user or done by the system automatically. (Additional atom coordinates can appear in units, if there is a special handling for crystallographic symmetry necessary, which can be caused of forming a crystallographic symmetry related molecule which is in an biological contact)(case2)
Patch	Surface Region of inter-protein contact. Symmetry created surface points with contact to other surface points (distance cut-off) after symmetry calculation. (for each symmetry + txtytz), or surface points between different units (symmetry= 0 tx=0 ty=0 tz=0)

PatchCalc	->Patch Calculation Program
PDB	The Protein DataBase (http://www.rcsb.org/pdb/)
PyMol	Application used to display molecular structure in 3D (Provides a Python programming interface)
Standard Partition	The standard Partition is defined as the separated Units defined by dividing chains and components in the stored PDB file.(each chain and each component is a separate unit)(case1) A pre- selection can be defined as standard partition as well (e.g. if an component is fully integrated in a other Unit (chain)) (case1b)
Surface (PPIX)	MSMS calculated surface (points) of a unit.
Symmetry Operation	Matrix operation to create symmetry equivalent atoms or coordinates from the asymmetric unit.
Symmetry Reduction	The Some ->oligomeres build the biological unit new and reduce the symmetry accordingly by removing the corresponding symmetry operation.
Unit	A Unit is a selection of separated components (chains or hetero components). All defined units together form a Partition which must be composed of all atoms in a structure.
Unit-Cell	The Unit-Cell contains all the Atoms which are necessary to form the crystal-structure by translation of Unit-cell in x,y,z.
VRML	Description Language for (Web) Interactive 3D Enviroments
X3D	VRML 2.0
Oligomere	A structure consisting of 2 or more chains

Appendix B

Interface Definition

PPIX

Protein-Protein Interaction in Crystals

Interface definition between the web and calculations, analyses and export functions

Robert Rader (rrader@sbox.tugraz.at)

April 25, 2005
Version 0.4

1 Calculations

1.1 Webpages

- Calculation Status Page:
  Displays all running jobs and their status. (accesses Calc_Jobs table)
- Calculation Start Page:
  Either select all entries, or specify entries in a text field. Select Calculations, and select whether to keep already calculated properties. (accesses Calculations table / sends message)
- Job details
  Display log entries for a given job. Either display only errors or all logged messages.
- Entry view
  Display which property has been calculated and which has not.
  Display calculation results.

1.2 Entitiy Beans (Database tables):

1.2.1 CALCULATIONS

The Calculation tables stores all available tables.

- calc_key
  Primary Key
- calc_name
  Name of the Calculation
- class_name
  java class name of the calculation, that will be called from the message bean
- description
  description of the calculation for use in the web interface.
- default_parameters
- required_calcs (fk)
  is a foreign key to another calculation, that must be completed before this calculation can be started.

1.2.2 ENTRY_CALCULATION

Is a many to many table between the calculation and the entry table, to store which calculation has already been run for which entry.

- entry_calc_key
  Primary Key
- entry_key (fk)
- calculation_key (fk)
- calc_date
- user_id

- parameters
  parameters used for this calculation
- prog_version

1.2.3 CALC_JOBS

The CALC_JOBS table stores all started / running and finished jobs.

- job_key
- calculation_key (fk)
- entries_scheduled (nr. of)
- finished_successfully (nr. of entries)
- finished_errors (nr. of entries)
- status (scheduled, running, finished, abort, abortDelete, aborted)

scheduled:	not yet sent to the cluster
running:	sent to the cluster
abort:	job aborted by user, but not yet stopped
abortDelete:	same as abort, but delete calculation results
aborted:	job aborted by user and stopped

- user_id (from Usermanagement)

1.2.4 CLUSTER_JOBS

This table stores the job information for each cluster job. It is not of interest for the web.

- cluster_job_key (pk)
- job_key (fk)
- entry_key (fk)
- calculation_key (fk)
- cluster_job_id
  job id from the cluster
- status (not scheduled, scheduled, failed, completed successfully)

1.2.5 CALC_LOG

Logs all messages returned by a calculation.

- log_key
- job_key (fk)
- entry_key (fk)
- type (success / error)
- text

1.3 Services

1.3.1 Session Bean:

The **Calculation service** is a single service bean for accessing the entities:

Calculation
EntryCalculation
CalcLog
CalcJobs

(standardservices created by andromda)

The **abortCalc (job_key)** method sets the status of the job to 'abort'. The CalcJob has to check regularly for this status and abort it.
The **abortDeleteCalc (job_key)** method sets the status of the calc_jobs to 'abortDelete'. The CalcJob has to check regularly for this status.

The **UserCalcMessage calculate(Collection entries, Collection calculations, boolean keep_existing)** method creates a calculation_job for each calculation and returns a UserCalcMessage, which is later sent to the CalcMDB as an ObjectMessage.

1.3.2 Message Driven Beans:

The **CalcMDB** receives an ObjectMessage with a UserCalcMessage. It creates a CalcJob and calls the run method.

1.4 Java Classes

A UserCalcMessage is a simple bean with one constructor, setter, getter and adder methods. It is used to pass the user information to the CalcMDB.
A CalcJob is main class to perform the calculations. It is responsible that all calculations are run in the correct order. It has to update the CalcJob table, which is read by the calculation status page.

```
package at.tugraz.genome.ppix.calculation;

public class UserCalcMessage implements java.io.Serializable (

  /**
   * Constructs a UserCalcMessage
   *
   * @param entries Collection of entries (pks)
   * @param calcs Collection of calculations (pks)
   * @param keep_existing keep or recalculated existing properties
   */
public void UserCalcMessage (   java.util.Collection entries,
                                java.util.Collection jobs,
                                boolean keep_existing);
```

```
package at.tugraz.genome.ppix.calculation;

public class CalcJob (

  /**
   * Constructs a CalcJob
   * @param UserCalcMessage calculation job definitions
  */
public void CalcJob(UserCalcMessage ucm)

  /**
   * runs the User Job
   */
public void run();
```

2 Analyses

2.1 Webpages

- Analyses Status Page:
  Displays all running analyses and their status. (accesses ANALYSES_JOBS table)
- Analyses Start Page:
  Either select all entries, or specify entries in a text field. Select which analyses to run and specify parameters if necessary. (accesses Analyses table / sends message to AnalaysisMDB)

2.2 Entity Beans (Database tables)

2.2.1 ANALYSES

The Calculation tables stores all available tables.

- analysis_key
  Primary Key
- analysis_name
  Name of the Calculation
- class_name
  java class name of the analysis
- description
  description of the calculation for use in the web interface.
- default_parameters

2.2.2 ANALYSES_JOBS

The ANALYSES_STATUS table stores all started / running and finished jobs.

- job_key
- analysis_key (fk)
- progress (0 … 100 %)
  can be updated by the analysis. Serves as information for the user in the analysis status page. Should be set to NULL if no information is available.
- parameters (text)
- status (running, finished, abort, aborted)
- user_id (from Usermanagement)

2.3 Services

2.3.1 Session Bean

The **Analysis Service** is the single service bean for accessing the analyses and the analyses_jobs tables. (standard service operations are created by andromda).

The **abortAnalysis (job_key)** method sets the status of the analysis job to 'aborted'. Long lasting analyses have to check for this status regularly.

The **UserAnalysisMessage calculate(Collection entries, Long analysis, String parameters)** method creates an analysis_job and returns a UserAnalysisMessage, which is later sent to the AnalysisMDB as an ObjectMessage.

2.3.2 Message Driven Beans:

The **AnalysisMDB** receives an ObjectMessage with a UserAnalysisMessage object, creates an analysis and calls the run method.

2.4 Java Classes

The calculate method is called after a user has entered data in the analysis start page. The returned UserAnalysisMessage is then passed to the AnalysisMDB inside an ObjectMessage, where an AnalysisJob is created and the run method is called.
An analysis my update the progress attribute in the analysis job table.

```
package at.tugraz.genome.ppix.analysis;

public class UserAnalysisMessage implements java.io.Serializable (

  /**
   * Constructs a UserAnalysisMessage
   *
   * @param entries Collection of entries (pks)
   * @param analysisJob analysis job (pk)
   */
public void UserAnalysisMessage (   java.util.Collection entries,
                                    Long analysisJob);

package at.tugraz.genome.ppix.analysis;

public abstract class AnalysisJob (

  /**
   * Constructs an Analysis
   *
   * @param uam UserAnalysisMessage includes all necessary
   * information for one analysis
   */
  public void Analysis( UserAnalysisMessage uam);

  /**
   * runs the Analysis
   */
  public void run();
```

3 Exporter

3.1 Webpages

- Entry View
  Exports of entries can only be done one by one from the Entry View page.

3.2 Services

One ExportService Session Bean has one method for each available export function.

3.3 Java Classes

An Exporter has a constructor getting the primary key of an entry as a parameter and a method export that exports the entry and returns a stream.

```
package at.tugraz.genome.ppix.exporter;

public abstract class Exporter {

private Long entry_;

  /**
   * Exporter
   *
   * @param entry primary key of the entry to be exported
   */
  public Exporter(Long entry) {
    entry_ = entry;
  }

  /**
   * exports one entry
   *
   * @return exported entry as a stream
   */
  public abstract (java.io.OutputStream) export();
}
```

Appendix C

PPIX Database Tables

Table C.1: List of PPIX Tables. The second column shows if they have been adopted from the OpenMMS database schema. In the last column the modifications from the OpenMMS tables are noted. For all tables pdbx attributes have been removed.

Table Name	**PPIX**	**Modifications**
ATOM_SITE	X	
ATOM_SITES	X	
ATOM_SITES_ALT	X	
ATOM_SITES_ALT_ENS	X	
ATOM_SITES_ALT_GEN	X	
ATOM_SITES_FOOTNOTE	X	
ATOM_TYPE	X	
CELL	X	
CHAIN_HOMOLOGY		
CHAIN_TO_UNIT		
CHEM_COMP	X	
CHEM_COMP_ATOM	X	added hydrophobicity attributes
CITATION	X	added author and editor fields
CONSERVATION_SCORE		
DATABASE_LINKS		
DIFFRN	X	
ENTITY_KEYWORDS	X	
ENTITY_NAME_COM	X	

Continued on next page

Table C.1 – continued from previous page

Table Name	MMS	Modifications
ENTITY_NAME_SYS	X	
ENTITY_POLY	X	
ENTITY_POLY_SEQ	X	
ENTITY_SRC	X	combined most important attributes of ENTITY_SRC_GEN and ENTITY_SRC_NAT
ENTRY	X	renamed from MMS_ENTRY
EXPTL	X	
EXPTL_CRYSTAL	X	added a free text field for additional experimental information
EXPTL_CRYSTAL_GROW	X	
EXPTL_CRYSTAL_GROW_COMP	X	
PATCH		
PATCH_INTERACTION		
PATCH_TYPE		
PENTITY	X	renamed from Entity, because of problems in source generation
PPIX_SYSTEM	X	renamed from MMS_SYSTEM
REFINE	X	
REFLNS	X	
SPACEGROUP		
STRUCT	X	
STRUCT_ASYM	X	
STRUCT_BIOL	X	
STRUCT_BIOL_GEN	X	
STRUCT_BIOL_KEYWORDS	X	
STRUCT_BIOL_VIEW	X	
STRUCT_KEYWORDS	X	
STRUCT_REF	X	
STRUCT_REF_SEQ	X	
STRUCT_REF_SEQ_DIF	X	

Continued on next page

Table C.1 – continued from previous page

Table Name	MMS	Modifications
SURFACE_POINT		
SURFACE_POINT_TO_PATCH		
SYMMETRY	X	
SYMMETRY_EQUIV	X	
TRIANGULATION		
UNIT		
UNIT_PARTITION		
UNIT_SURFACE		

Bibliography

[1] DNA-RNA-Protein Introduction.
http://nobelprize.org/medicine/educational/dna/
November 24th, 2005

[2] Crick F. Central Dogma of Molecular Biology. *Nature*, 227:561–563, 1970.

[3] Ambros V. microRNAs: tiny regulators with great potential. *Cell*, 107(7):823–826, 2001.

[4] Mattick J. Challenging the dogma: the hidden layer of non-protein-coding RNAs in complex organisms. *Bioessays*, 25(10):930–939, 2003.

[5] Dennis C. The brave new world of RNA. *Nature*, 418(6894):122–124, 2002.

[6] IUPAC-IUB. Abbreviations and Symbols for the Description of the Conformation of Polypeptide Chains. *Arch. Biochem. Biophys.*, 145:405–421, 1971.
http://www.chem.qmul.ac.uk/iupac/misc/ppep1.html.

[7] Introduction to Biological Units.
http://www.rcsb.org/pdb/biounit_tutorial.html
November 24th, 2005

[8] Protein Data Bank (PDB).
http://www.rcsb.org/pdb/
November 24th, 2005

[9] Koetzle TF, Bernstein FC et al. The Protein Data Bank. A computer-based archival file for macromolecular structures. *Eur J Biochem*, 80(2):319–324, 1977.

[10] PDB Documentation and Information.
http://www.rcsb.org/pdb/info.html
November 24th, 2005

[11] Research Collaboratory for Structural Bioinformatics.
http://www.rcsb.org/index.html
November 24th, 2005

[12] Henrick K, Thornton JM. PQS: a protein quaternary structure file server. *Trends Biochem Sci*, 9:358–361, 1998.

[13] PQS Protein Quaternary Structure Query Form at the EBI.
http://pqs.ebi.ac.uk/
November 24th, 2005

[14] PDB Format Description Version 2.2.
http://www.rcsb.org/pdb/docs/format/pdbguide2.2/guide2.2_frame.html
November 24th, 2005

[15] mmCIF.
http://mmcif.pdb.org/
November 24th, 2005

[16] Hall SR. The STAR file: a new format for electronic data transfer and archiving. *Journal of Chemical Information and Computer Sciences*, 31(2):326–333, 1991.

[17] Berman HM, Bourne PE et al. The Macromolecular Crystallographic Information File (mmCIF). *Methods in Enzymology*, 277:571–590, 1997.

[18] DeLano WL. The PyMOL Molecular Graphics System.
http://www.pymol.org
November 24th, 2005

[19] Steinkellner G. *PPIX - Protein Protein Interactions in Crystals - Database and Tools.* PhD thesis, Graz Karl-Franzens University, 2006.

[20] Object Management Group (OMG).
http://www.omg.org
November 24th, 2005

[21] Catalog of OMG Modeling and Metadata Specifications.
http://www.omg.org/technology/documents/modeling_spec_catalog.htm
November 24th, 2005

[22] Model-Driven Architecture (MDA).
http://www.omg.org/mda/
November 24th, 2005

[23] Miller J, Mukerji J. MDA Guide Version 1.0, May 2003.
http://www.omg.org/docs/omg/03-06-01.pdf
November 24th, 2005

[24] Meta-Object Facility (MOF).
http://www.omg.org/cwm/
November 24th, 2005

[25] Common Warehouse Metamodel (CWM).
http://www.omg.org/cwm/
November 24th, 2005

[26] What is UML.
http://www.omg.org/gettingstarted/what_is_uml.htm
November 24th, 2005

[27] XMI Specifications.
http://www.omg.org/technology/documents/formal/xmi.htm
November 24th, 2005

[28] Ball J, Armstrong E et al. The j2ee 1.4 tutorial.
http://java.sun.com/j2ee/1.4/docs/tutorial/doc
November 24th, 2005

[29] Singh I, Stearns B, Johnson M. *Designing enterprise applications with the J2EE platform.* Addison-Wesley, Reading, second edition, 2002.

[30] Monson-Haefel R, Burke B, Labourey S. *Enterprise JavaBeans.* O'Reilly Media, Inc., Sebastopol, fourth edition, 2004.

[31] Roman E. *Mastering Enterprise JavaBeans and the Java 2 Platform, Enterprise Edition.* John Wiley & Sons, New York, London, Sydney, 1999.

[32] Ullenboom C. *Java ist auch eine Insel.* Galileo Press, Bonn, 4. aufl. edition, 2005.

[33] JavaBeans. `http://java.sun.com/products/javabeans/index.jsp`.

[34] Alur D, Malks D, Crupi J. *Core J2EE Patterns: Best Practices and Design Strategies.* Prentice Hall PTR, Englewood Cliffs, 2001.

[35] Apache Struts.
http://struts.apache.org
November 24th, 2005

[36] XDoclet - Attribute Oriented Programming.
http://xdoclet.sourceforge.net/xdoclet/index.html
November 24th, 2005

[37] AndroMDA.
http://www.andromda.org/
November 24th, 2005

[38] Apache Ant.
http://ant.apache.org
November 24th, 2005

[39] Truskaller T. Data Integration into a Gene Expression Database. Master's thesis, Graz University of Technology, 2003.

[40] Schema2XMI.
http://www.andromda.org/andromda-schema2xmi/
November 24th, 2005

[41] UML Tools.
http://www.uml.org/#Links-UML2Tools
November 24th, 2005

[42] Rational Rose.
http://www-306.ibm.com/software/rational/
November 24th, 2005

[43] Grady Booch.
http://www-306.ibm.com/software/rational/bios/booch.html
November 24th, 2005

[44] MagicDraw.
http://www.magicdraw.com/
November 24th, 2005

[45] Zeller D. Design and Development of a User Managment System for Molecular Biology. Master's thesis, Graz University of Technology, 2003.

[46] Maurer M, Molidor R, Sturn A et al. MARS: microarray analysis, retrieval, and storage system. *BMC Bioinformatics*, 6(1):101, 2005.

[47] OpenMMS Toolkit.
http://openmms.sdsc.edu/
November 24th, 2005

[48] Institut für Physikalische Chemie (Subgruppe Strukturbiologie) at the Karl-Franzens University Graz.
http://strubi.uni-graz.at
November 24th, 2005

[49] Jorde MC. High throughput Analysis of Protein Surface Properties. Master's thesis, Graz University of Technology, 2006.

[50] Sanner MF, Olson AJ, Spehner JC. Reduced surface: an efficient way to compute molecular surfaces. *Biopolymers*, 38(3):305–320, 1996.

[51] Apache Log4j.
http://www.apache.org/log4j.html
November 24th, 2005

[52] Ugawa Y, Gilbert D et al. Bio-mirror project for public bio-data distribution. *Bioinformatics*, 20(17):3238–3240, 2004.

[53] Biomirror at the Institute for Genomics and Bioinformatics at the University of Technology Graz.
http://biomirror.genome.tugraz.at/
November 24th, 2005