
Biotechnology researchers increasingly rely their research on the large amount of data stored in multiple heterogeneous bioinformatics databases. Typically, they need to access these databases in a sequential and/or a parallel manner. Thus, an integrated environment and a unified interface are critical for them. This thesis addresses this requirement by introducing a Web-Services based architecture for integrating bioinformatics databases and discusses an implementation that integrates the PDB database, the ProTherm database and the MuSE software.

The integrated system has a multiple-tier architecture centered with Web-Services. In addition to use Web-Services as the interface to access databases, the system uses JSP for presentation, Servlet as the controller to provide an extensible and flexible integration management component, and Databases as the backend for data storage. The interface of the integrated system uses multiple frames and provides a unified interface to access integrated system as well as other databases and software.

Headings:

Bioinformatics Databases – Integration

Information System – Design

System Structure – Multiple-tier architecture

Web-Services
A WEB-SERVICES APPROACH
FOR INTEGRATION OF HETEROGENEOUS
BIOINFORMATICS DATABASES
AND SOFTWARE

By
Jun Zhang

A Master’s paper submitted to the faculty
of the School of Information and Library Science
of the University of North Carolina at Chapel Hill
in partial fulfillment of the requirements
for the degree of Master of Science in
Information Science.

Chapel Hill, North Carolina
April, 2003

Approved by:

_________________________
Advisor
# Table of Contents

Chapter 1  Introduction ........................................................................................................ 1

Chapter 2  Background Review ......................................................................................... 6

  2.1. Services–Oriented Architecture .............................................................................. 6
  2.2. Database Integration Methods ............................................................................... 8
  2.3. Related Bioinformatics Databases ....................................................................... 11
     2.3.1. PDB Database ......................................................................................... 12
     2.3.2. ProTherm Database ............................................................................... 15
     2.3.3. MuSE Software ..................................................................................... 17

Chapter 3  System Requirement and Functionality ......................................................... 18

  3.1. User Scenario ........................................................................................................ 18
  3.2. System Requirements .......................................................................................... 19
  3.3. System Functionalities ........................................................................................ 21

Chapter 4  System Design and Architecture ................................................................... 23

  4.1. Design Strategy .................................................................................................... 23
  4.2. System Architecture .......................................................................................... 24
  4.3. System Design .................................................................................................... 27
     4.3.1. Interface Design ..................................................................................... 27
     4.3.2. Middle Tier Design ............................................................................... 29
4.3.3. Web-Services Design. ................................................................. 31

4.3.4. ProTherm Database Design .................................................. 36

4.4. System Development Environment ............................................. 39

Chapter 5    User Guides ................................................................. 41

5.1. Guide for End-User ................................................................. 41

5.1.1. Overview. ................................................................. 41

5.1.2. ProTherm Search Page .................................................. 41

5.1.3. ProTherm Query Results Page ....................................... 45

5.1.4. MuSE Mutation Prediction Page ....................................... 46

5.1.5. MuSE Prediction Results .............................................. 46

5.2. Guide for Developers ............................................................... 46

5.2.1. Overview ................................................................. 47

5.2.2. ProTherm Web-Services API Documentation .................. 48

5.2.3. XML Format ............................................................. 50

5.2.4. Procedure for Generating Stub and Client ....................... 52

Chapter 6    Conclusion and Future Work ........................................ 54

6.1. Conclusion ................................................................. 54

6.2. Future Work ................................................................. 55

Bibliography ................................................................................. 59

Appendix A Screen Shots of Interface ........................................... 62

Appendix B WSDL of ProTherm Web-Services ......................... 69
Chapter 1: Introduction

Over the last decades, biotechnology researchers have accumulated a large amount of data. This data has been stored in various databases by different research groups. Quite often, many of these databases were not initially designed as databases, but become one during the research process. Because of the nature of the creation of these databases and the lack of standard formats for data storage, each of these databases has quite a different organization and interface (Ellis & Attwood, 2001). Moreover, many of these databases contain information on the same topic with different emphases. For example, “The Molecular Biology Database Collection: 2003 update” has listed key biological databases including 64 genomic databases, 15 DNA sequence databases, 32 gene identification and structure databases, 63 protein databases and 40 structure databases (Andreas 2003).

Biotechnology researchers rely on data in these valuable databases to conduct effective research (Ellis & Attwood, 2001). One typical example is that researchers need to get some specific information from a database, e.g. the thermodynamic data of proteins from the ProTherm database. They then analyze the information and use the result to further gather the structure information of proteins in a protein structural database such as Protein Data Bank (PDB). In this case, they want to be able to dynamically issue a sequence of queries. The first query will be issued against one database (or a selected set
of databases). Its results will be used as the second query to query the second database (or a second set of databases), and so on. Another example is that researchers who study a gene will need to gather all the relevant information about that gene. That information may include a paper in MEDLINE that describes the sequencing of that gene, the gene sequence that is stored in GenBank, and the gene’s mapping information that is stored in a genome mapping database. Also, the gene sequence codes a protein product whose sequence is stored in the protein sequence database and its structure may be found in a protein structural database. Thus, in order to gather all these types of information about a specific gene, researchers need to conduct searches on multiple related databases. Ideally, they would like to have a unified interface to easily access all of those databases. They would also like to be able to specify which databases to search and with one query they can automatically obtain all relevant information from these various databases.

In both examples discussed above, access to multiple heterogeneous databases is required. Normally, each of these databases has a web interface and is located on different computer networks that are connected to the Internet. Moreover, different research groups need to access different sets of databases and many of these databases have different formats and interfaces. Therefore, it is necessary to develop technologies to integrate various heterogeneous databases to allow easily access to them according to users’ requirements. Furthermore, the integration technologies have to take into account how to handle disparate data formats and how to handle various network communications protocols.
Several integration approaches have been proposed over the last several years, which notably include the data warehousing approach (Davidson, et. al. 2001), the multiple database tools approach (Haas, et. al. 2001; Goble, et. al. 2001; Markowitz & Chen, 1999) and the distributed component-based approach (Siepel, et. al., 2001a; Siepel, et. al., 2001b; Hu, Mungall, Nicholson & Archibald, 1998; Dickson, 2001). The data warehousing approach creates a central repository to incorporate data from different databases. It has the advantage of enforcing data consistency, providing uniform data representation and encouraging coordinated development. However, the high cost of building and maintaining such a system is formidable. It is also necessary to synchronize the data in the data warehouse with the individual databases. Another obstacle is lack of consensus on data representation format.

The multiple database tools approach designs a set of tools to present the information on behalf of various underlying databases. The tools usually include a query component. However, the tools need to know the protocol to access the underlying databases. They also need to know the data format of the different databases and have a unified presentation, which itself is an un-solved problem.

The distributed component-based approach defines a standard application programming interface (API). Individual databases will be wrapped with this interface and allow others to access the databases through this standard interface. Common Object Request Broker Architecture (CORBA) and Java Remote Method Invocation (RMI) (Harold, 1998) are normally employed as supporting architectures.
Even though each of these approaches has its pros and cons, many researchers, such as Siepel, et. al. (2001a), have argued in favor of the distributed approach over the other two approaches. However, the distributed approach proposed in that article and other articles have been strongly influenced by the CORBA technology. CORBA is the technology based on the research work on distributed programming in the early 1990s. As computer technologies advance, it is valuable to explore whether new technologies can be used to design better architecture for system integration.

We believe that the recent proposed Web-Services (Rommel, 2001) and XML (Achard, Vaysseix, & Barillot, 2001) technologies are a much better successor of COBRA for service-oriented system design and integration. The key advantages of Web-Services include: an Internet friendly open standard, an XML based open system and a wider industry support. Using Web-Services, we can build a more flexible and extensible integrated bioinformatics database system.

This thesis will exploit the idea of using web-services as the core technology for integrating and utilizing different bioinformatics databases and software. Specifically, we will focus on integrating the PDB database, the ProTherm database and the MuSE analytical software. This integrated system has a multiple-tier architecture centered with the use of Web-Services. This Web-Services tier provides a Web-Services based interface to access bioinformatics databases. In other tiers, it uses JSP for the presentation, a Servlet as the controller for providing an extensible and flexible integration management
component, and Databases as the backend for the data storage. This architecture results in a flexible and extensible system that can independently grow in the Web-Services tier, Servlet tier and other tiers. The interface of the integrated system uses multiple frames and provides a unified interface to access other databases without leaving the system. More importantly, the user requirements for integrating the example databases and software are coming from a research group at University of North Carolina at Chapel Hill (UNC-CH). Our solution thus not only has theoretical values to prove the concept of integrating bioinformatics database by web-services, but also directly addresses a practical need from bioinformatics research communities.
Chapter 2: Background Review

There are three different areas that are related with this research. The first area is technologies that are used or will be used by integration methods. These technologies include CORBA and Web-Services. The second area is integration methods, specifically, a distributed approach of integrating different bioinformatics databases. The third area includes the specific databases and software that will be used in our integrated system.

2.1. Services-Oriented Architecture

CORBA (Stevens & Miller, 1999) is a distributed object-oriented architecture proposed by the Object Management Group (OMG) in the early 1990s. Later in the 1990s, CORBA was accepted as an industry standard distributed object model by many of the computer companies except, notably, Microsoft. Microsoft has its own standard distributed object model called DCOM. CORBA has the following two major advantages. First, CORBA allows objects to be written in different languages and run on different machines, these objects will work together in the CORBA framework. Second, CORBA provides an interface definition language (IDL). A CORBA object exposes its function through IDL and thus hides implementation details. This separation of interface from implementation makes it possible to improve a system without affecting client applications.
Different CORBA objects communicate with each other through an Object Request Broker (ORB). ORB has a standard specification; however, different vendors can compete on the implementation of ORB. In addition to the communication protocol, CORBA also defines a set of services including object life cycle services, event services and security services.

Web-Services (Rommel, 2001) were first introduced in the later 1990s. Similar to CORBA, Web-Services also define a distributed object-oriented architecture. However, Web-Services is a more loosely coupled architecture than CORBA. In Web-Services architecture, each component is regarded as a service. The Application Programming Interface (API) of a service exposes the functions of the Web-Services. A Web-Services can be described, published, located and invoked over a network, generally, the World-Wide-Web. All of these capabilities are defined by the following set of standards:

- Simple Object Access Protocol (SOAP). This protocol uses XML to transfer messages between different applications.
- Web-Services Description Language (WSDL). WSDL provides a XML format to describe the functions of a Web-Services.
- Universal Description Discovery and Integration (UDDI). UDDI allows both the registration and discovery of a Web-Services by querying the UDDI registry.

Thus, the Web-Services can be used dynamically.
As pointed out in Gisolfi (2001), Web-Services has the following advantages when compared with CORBA:

- Uses HTTP to be firewall friendly;
- Employs XML as an encoding schema that is more widely adopted than CORBA’s presentation language;
- Can be deployed in a “free” HTTP/SOAP server environment (the management system of Web-Services is maintained by third parties. On the other hand, CORBA implementation requires an ORB to be maintained by the integration system);
- Uses the pervasive internet concept of URLs to address object identifications;
- Offers more than a promise of interoperability.

Finally, traditionally competitive vendors, including Microsoft, Sun and IBM, all embrace the concept.

2.2. Database Integration Methods

As mentioned in the introduction, there are three primary approaches to integrate different bioinformatics databases. However, this review will focus on the distributed approach. The distributed approach can be further classified into two sub-categories. One directly uses CORBA as the basic architecture. The other category uses CORBA as a protocol to communicate with different bioinformatics databases.
Using CORBA as the basic architecture, several researchers have taken different approaches. Hu et. al. (1998) defined an IDL for the genome mapping system. They used this IDL to write wrappers for several genomic databases and used ORB to link the different databases together. To demonstrate the use of the architecture, they built a prototype and added a client component to the prototype system. This client component allows web users to access the integrated system. This client uses a Java applet to talk with the CORBA server. The Java applet provides the system with the flexibility to include a gene map viewer into the prototype. The prototyping method used in this research is valuable in proving the concept of the CORBA-based approach. The definition of the IDL is an important attempt to design a common object model for representing the genome mapping information. However, the paper did not discuss the implications when integrating other bioinformatics databases. The paper also did not address the issue whether the bioinformatics databases can be located on different machines on the Internet.

Dickson (2001) discussed the implications when integrating other bioinformatics databases which were not addressed by Hu et. al. (1998). His paper proposed a CORBA-Java based framework for drug discovery, which needs to use heterogeneous bioinformatics databases. The paper discussed several different approaches to use CORBA to integrate different databases. It also discussed a workflow type of query that may need to access data from one database and then use the results to access another database. The flexibility of integrating different databases using this CORBA-Java framework was discussed. However, most of the discussions in this paper were at an
abstract level. No practical data and experiments were presented to support the arguments.

Jungfer et. al. (1999) reviewed the European Bioinformatics Institute (EBI)’s efforts using CORBA to integrate different bioinformatics databases. In addition to a discussion on defining IDL for the sequence data, the paper discussed several issues due to the use of the CORBA wrappers. For example, it looked at how the system states should be managed in a workflow type of query and how to map an IDL object definition to the data in a relational database. It also briefly reviewed several other prototypes built using CORBA as the basic framework. The paper shows the vast interests and the feasibility in using CORBA as the basic framework to integrate multiple bioinformatics databases. However, no quantitative data is presented to support the integration methods. There were also no discussions on communication issues over the Internet.

In addition to using CORBA as the basic integration architecture, several other researchers (Siepel, et. al., 2001a; Siepel, et. al., 2001b) have used CORBA as a communication protocol to link different databases. Those researchers point out that client side integration using different tools to take advantage of different databases is also very important. Interestingly, several of these client side integration approaches used the CORBA ORB concept as the core architecture. Integrated System (ISYS) (Siepel, et. al., 2001a) is a notable example. ISYS proposed a client platform based on CORBA’s service registration concept. Each client application, such as a SequenceViewer, can register with the ISYS platform as a service and thus can be used by end users. Each server
application, including different server side bioinformatics databases, can also register with the ISYS platform as a service. Those databases will be wrapped with CORBA IDL and communicate with the ISYS platform using the CORBA protocol. ISYS has been a very successful research project. It started as a prototype and now has been used by several research organizations to integrate various bioinformatics tools and databases.

Several concerns with ISYS are as follows: It defines its own proprietary interfaces for both client tools and server databases. Thus, to use a client tool, the client application has to write a specific ISYS wrapper that uses the ISYS interfaces. This wrapper cannot be used by other integration frameworks. This is also an issue for integrating server side databases. The IDL is defined to be consistent with the ISYS server proxy interface. Another problem with ISYS is that the contents to be transferred between the ISYS platform and the server IDL wrapper are not discussed in detail and are not built using open standards.

2.3. Related Bioinformatics Databases

The PDB database, the ProTherm database and the MuSE analytic software have been selected for the integration. Following is a brief review about them.

2.3.1. PDB

The PDB is the single international repository for public data on the three-dimensional structures of biological macromolecules (PDB Annual Report 2002).
As stated in the PDB website, the primary goals of the PDB database are:

- Enable users to locate structures of interest;
- Perform simple analyses on one or more structures;
- Act as a portal to additional information available on the Internet;
- Enable you to download information on a structure, notably the Cartesian atomic coordinates, for further analysis.

The PDB is an integrated system of heterogeneous databases that store, organize and distribute the structural data and related information of biological macromolecules such as proteins, nucleic acids, and protein-nucleic acid complexes. Currently, it consists of five major components as described by Berman et al. (2000):

- The core relational database managed by Sybase provides the central physical storage for the primary experimental and coordinate data. The core PDB relational database contains all deposited information in a tabular form that can be accessed across any number of structures.
- The final curated data files (in PDB and mmCIF formats) and data dictionaries are the archival data and are present as ASCII files in the ftp archive.
- The POM (Property Object Model)-based databases, which consist of indexed objects containing native (e.g., atomic coordinates) and derived properties (e.g., calculated secondary structure assignments and property profiles).
- The Biological Macromolecule Crystallization Database (BMCD) is organized as a relational database within Sybase and contains three general categories of literature derived information: macromolecular, crystal and summary data.
The Netscape LDAP server is used to index the textual content of the PDB in a structured format and provides support for keyword searches.

Figure 1. The integrated query interface to the PDB

(Courtesy of Berman et. al., 2000)

As described by Berman et. al (2000), communication among these five databases has been accomplished by using the Common Gateway Interface (CGI). An integrated Web interface submits a query to the appropriate database(s), which then execute the query. Each database returns the PDB identifiers that satisfy the query, and the CGI program integrates the results. A variety of output options are then available for use with the final list of selected structures.

Berman et. al (2000) believed that the CGI approach permits integration of other databases into this system. For example, data on different protein families, NMR data
found in the BMRB or data found in other community databases could be included into the system.

In the future, a CORBA-based approach will be used to implement the PDB system. The PDB is adopting the CORBA Macromolecular Structure Specification. This specification, closely aligned with the mmCIF standard, opens the door to more seamless and specific access to PDB data by providing a standard Application Programming Interface (API). The standard API will allow direct access to the binary data structures of the PDB by remote programs. Investigators will be able to retrieve a single data item from an entire PDB file for use in a local application, without having to download the entire file (Berman et. al., 2000).

The PDB site provides three query interfaces for users to query the data in PDB. These three interfaces are ‘Status Query’, ‘SearchLite’ and ‘Search-Fields’.

The SearchLite interface provides a single form field for simple text-based search and returns one or more structures which contain text that matches the query string. All textual information in the PDB files including dates and some experimental data are searchable via simple or structured queries.

The SearchFields interface is a customizable query form. This interface allows searching over many different data items including PDB ID, compound, citation authors, sequence and release date or deposition date.
There are two interfaces for query result display. They are ‘Query Result Browser’ and ‘Structure Explorer’.

The ‘Query Result Browser’ interface provides some general information and allows access to more detailed information in tabular format. It also provides the options to download whole sets of data files for result sets consisting of multiple PDB entries, to refine the query and to review the query and so on.

The ‘Structure Explorer’ interface provides summary information about individual structures and serves as the entry point to find more information on structure. It provides options to view structure, to download and display files, to access the structural neighbors and geometry of the structure and so on.

2.3.2. ProTherm

The Thermodynamic Database (ProTherm) for Proteins and Mutants contains 13140 entries of proteins and mutants as of March 17, 2003. Each entry includes numerical data of several thermodynamic parameters, sequence and structural information, experimental methods and conditions, and literature information (Gromiha, et. al., 2002).

The sequence and structural information includes information about the source of each protein, secondary structure, identification codes for SWISS-PROT, Protein Data Bank (PDB), and other relevant databases.
ProTherm is cross-linked with sequence, structural, functional database (3DinSight) and other databases (EC, BRENDA, PubMed), and the mutant sites and surrounding residues are automatically mapped on the structure (Gromiha, et. al., 2002).

The form-based interface of the ProTherm database enables users to search data based on various criteria such as the PDB code, number of states and reversibility. The search results can be displayed with different sorting options.

**Figure 2. Organization of ProTherm: Content of data and links to other databases**

*(Courtesy of Gromiha, et. al., 2002)*

In version 3.0, ProTherm is implemented into 3DinSight. 3DinSight is a relational database system for holding structural, functional and property information of biological molecules. Figure 2 illustrates the organization of ProTherm (Gromiha, et. al., 2002).
2.3.3. MuSE

MuSE is an analytical software system that uses the protein structure data in the PDB format and incorporates with the Simplicial Neighborhood Analysis of Protein Packing (SNAPP) evaluation to predict virtual mutagenesis of a protein.

The MuSE interface can accept PDB structure data stored at a local machine. It provides a browse function to allow the selection of the PDB data sets. It also can accept the PDB ID of a protein. It will then use this PDB ID to download the structure data from the PDB website. The downloaded data are in turn used by the analysis tool of MuSE.

In addition to the functions discussed above, the MuSE interface allows users to specify several parameters. Those parameters will be used when MuSE analyzes the submitted data.
Chapter 3: System Requirement and Functionality

3.1. User Scenario

The following user scenario is obtained from interviewing a UNC research team and experimenting with their existing systems.

In a typical data analysis process, the research scientists in that team first accesses the ProTherm database and searches for the thermodynamic data of proteins. The resulting data sets are then examined and the PDB IDs of interest are then selected. Those PDB ids are used as inputs to a query for searching the PDB database and to obtain protein structural data. Third, the resulting protein structural data are downloaded and submitted to the MuSE analytical software for predicting virtual mutagenesis of the proteins with incorporated SNAPP evaluation. Finally, the correlation between the unfolding Gibbs free energy change of the protein and their calculated Native Score or Difference will be analyzed using other selected statistical software.

There are several issues involved with the above process. First, ProTherm, PDB and MuSE are provided and maintained by three different organizations located at three different locations. Each system has its own web interface and a user may have to open
three web browsers to access them. Second, the query results from searching the first
database could contain hundreds or thousands of interesting data sets. The user has to
manually submit these results into the second database (PDB) or the MuSE system. The
task to submit these results individually to the subsequent database and software is
painstaking and inefficient. Third, the ProTherm database and MuSE do not have output
functions such as “Save Results”. This makes storing query results for further analysis
very difficult.

The scientists would like to have an integrated system that can help them to effectively
conduct their research work.

3.2. System Requirements

Our system is designed to server two purposes: one is to address the user needs as
described in the previous section. In other words, it is to provide a useful system to the
biotechnology research communities to serve their data retrieval and analysis needs.
Another is to prove the concept of Web-Services for integrating bioinformatics databases.
There are two potential user groups that are targeted by this system. One is the end user
of the Integrated System; the other is the application developer who is interested in
integrating bioinformatics databases. The end user often has a certain degree of the
domain knowledge and is more interested in the ease of use and the functionalities of the
system. The application developer is more interested in using the architecture developed
by this thesis to integrate other bioinformatics databases or to deploy our solutions in other environments.

With these two types of users in mind, it is necessary to design a system that is not only easy for the end user to efficiently access ProTherm, PDB and MuSE system, but also flexible and extensible to allow the integration of other databases into the system. To achieve these goals, the system needs to satisfy the following three primary requirements.

First, it should provide a unified interface and a central access point for users to access ProTherm and PDB, and MuSE. It should also allow the use of the query results from the first database as the input query to be submitted to the second database in a batch. The results from the second database can be submitted to the MuSE analytical software for further analysis.

Second, the system should use Web-Service architecture to wrap the bioinformatics databases and thus allow other Web-Services clients to access databases through Web-Services interface remotely. The Web-Service should have a clearly defined WSDL and XML return structure and thus can be used by other Web-Services clients in an open manner.

Third, the system should use a multiple-tier architecture to decouple the systems into a distributed environment. The system should have an extensible controller to allow other
Web-Services ready databases or HTTP ready databases to be easily integrated into the system.

3.3. System Functionalities

According to the system requirement, the integrated system will be designed to have a unified interface to provide a central access point for users to access ProTherm, PDB and MuSE. It should also have the following functions.

The system should allow access to individual ProTherm database, PDB database and MuSE software directly through the web interface of the integrated system.

The system should allow access to ProTherm, PDB and MuSE sequentially in the following manner:

- Users initiate a search at the ProTherm query search page to get results from the ProTherm database based on various conditions. Users have options to view brief and detailed displays.

- At the ProTherm search results page, users can perform the following actions: select multiple entries and submit their PDB_ID to the PDB database for exploring and downloading the structural data; or select multiple entries and submit their PDB_ID to MuSE system for predicting virtual mutagenesis of the proteins with incorporated SNAPP evaluation; or select multiple entries and save the results as text file for analysis using other statistical analysis software.
• At the PDB search result page, users can use all functions provided by the PDB database.

• At the MuSE submitting page, users can reselect the entries and provide additional parameters for predicting virtual mutagenesis of the proteins in a batch mode.

• Finally, users can save the results from MuSE as a text file for further analysis by using other statistical analysis software.

The integrated system should allow adding other databases into the system with a minimum integration effort. These databases may either have a Web-Services interface or a HTTP based interface.

The integrated system should serve as an example and also provide a guideline to allow other systems to access the ProTherm database through its Web-Services interface.
Chapter 4: System Design and Architecture

4.1 Design Strategy

The fundamental design consideration is how to use Web-Services, XML and Java Servlets as the core technologies to design a flexible and extensible system that can integrate different bioinformatics databases.

Our first design strategy is to develop a proxy or wrapper on the top of the existing bioinformatics database. This wrapper is a Web-Services end-point that can be accessed from a Web-Services client or other applications. The output of this wrapper is represented as an array of Strings with each String in the XML format. Thus, the outputs can be easily consumed by Web-Services clients.

Another strategy is to design the application based on the Model-View-Control architecture as shown in Figure 3. The Model will be various bioinformatics databases wrapped by Web-Services. The controller can talk with different Web-Services wrapped databases and forward the result to the different Viewers to present. This design allows the potential additions of other Web-Services wrapped databases into the system. Furthermore, the controller can also talk with bioinformatics databases through other
communication approaches such as direct HTTP requests to other bioinformatics databases.

The implementation of our system demonstrated the capability and flexibility of the designed system. A Web-Service interface has been developed for the ProTherm database and the HTTP protocol has been used to directly talk with PDB and MuSE by using their CGI interfaces.

**Figure 3. Model-View-Control Architecture**

![Model-View-Control Architecture Diagram](image)

### 4.2 System Architecture

The system employs a multiple-tier architecture as shown in Figure 4.
The client tier is the User Interface component (UI). It uses JSP, HTML and JavaScript as the primary implementation technologies.

The middle tier is the integration component. It is responsible for tying the system together by communicating with the client and the Web-Services or other backend systems. This tier is implemented as a web-based application using Java Servlet. This tier has two layers. The top layer is implemented as a Servlet and is served as a controller. It accepts client's initial requests and passes them to specific action handlers for processing. It also accepts the processing results returned from the action handlers and passes them to the client. Those action handlers compose the second layer. They process the specific user actions and know how to communicate with different backend systems. They are either Web-Services clients or HTTP clients. The Servlet is running under the Tomcat Java Server.

The Web-Services tier is the wrapper to the backend data storage system – the ProTherm database. It uses SOAP and WSDL as the main implementation technologies. It talks with the ProTherm databases through a generic Java JDBC API. It expresses its functions through Web-Services interface, for example, WSDL. This Web-Services is deployed in the Tomcat Java Server with the plug in of the Axis Soap Engine.

The backend data storage and analysis tier consist of the ProTherm database, the PDB database and the MuSE software. The ProTherm database is a local copy of the production version of the ProTherm database. This local copy is implemented using
MySQL relation database management system. There is no local copy of the PDB database. We use its CGI interface to access and retrieve desired data. The MuSE software also provides a web-based API. We use that API to serve the analysis request for the data obtained from the use of other part of the system.

This multiple-tier architecture results in a very flexible system. Each tier can be developed and improved independently. Each tier can run on completely different machines. These different machines can be located on different part of the Internet and can even be located behind companies’ firewalls.

Figure 4. System Architecture
4.3 System Design

4.3.1 Interface Design

The primary user interface of our system consists of two frames. The top frame is accessible from every page in the system. It provides the entry points to access the integrated system and to two other databases and the MuSE analysis software. The bottom frame, also called the content frame, is used to display the functions and the contents of the integrated system. It also displays the contents of other databases and software that can be directly accessed from the system. This design of the user interface provides a unified view for users to access several related databases and software, and also the integrated system. The access to other databases and software are straightforward: access their URLs and direct the results into the content frame. Thus, the rest of the discussion will be focused on the interface design of the integrated system.

The integrated system has four major pages. Each page is designed to be similar to the original interface of the databases and software. For example, the first page is used to specify the search criteria for searching the ProTherm database. The layout of this interface is designed to be similar to the search interface of the ProTherm database. The reason for having these similarities is to allow the users who are familiar with the original systems to quickly use the integrated system. However, we have made several significant changes and added many useful features as discussed below.
**Color scheme.** We have used consistent colors and fonts across the system. These colors and fonts also match the color and font scheme used in the original interface of PDB, ProTherm and MuSE.

**Much-improved look-and-feel.** For example, the layout of the MuSE page is reorganized to have a better layout and allow multiple selections. The results from the MuSE analytic software are now displayed in a table format with each set of results in a separate row.

**Simplified interface.** For example, the original ProTherm interface is overwhelming and hard to use. I have simplified that interface to make it more user-friendly.

**Useful features.** Several useful features have been introduced. For example, one can “Select All” and “Clear All” of one’s selections with one single click. The action buttons are displayed at the top and the bottom of the pages for the ease of use.

The interface is implemented by the use of JSP, JavaScript and HTML. JSP is a very powerful and flexible server-side technology. It is used to generate both dynamic contents and some client-side control functions. Specifically, JSP is used to retrieve information from JavaBean populated by the Servlet and to build the HTML for the presentation. JSP is also used to generate JavaScript functions. Some of these JavaScript functions are static and some of them are dynamic. For example, some values in those JavaScripts are initialized according to the amount of contents being retrieved.
JavaScript is critical for building the interface with a richer user experience. For example, select and de-selected a set of contents and submit requests to different resources in the same page are implemented by JavaScript functions.

Several screen shots of the user interface are included in Appendix A.

4.3.2 Middle Tier Design
As we have pointed out in the section of the system architecture, the middle tier is the controller that brings the various parts of the system together. It ties the system together by communicating with the client and the Web-Services or other backend systems. As we also pointed out, this tier has two layers. The top layer servlet is responsible for the overall control that accepts the client’s initial requests and passes them to a specific action handler for processing. It also accepts the processing results returned from action handlers and passes them to the presentation tier in the form of JavaBeans. The current implementation of the second layer includes the following four major action handlers.

**ProTherm query handler.** This handler communicates with the ProTherm Web-Services by acting as a Web-Services client. It gets appropriate parameters from the HTTP request and passes them into the Web-Services method call. These method calls are implemented using the WSDL-based stub-skeleton approach. It also uses JDOM API to parse the XML document that is returned from Web-Services. The results from this
parsing process are used to populate corresponding presentation Java Bean objects. It then returns those objects to the controller servlet.

**PDB query handler.** This handler communicates with the PDB database through PDB’s open CGI API, for example [http://www.rcsb.org/pdb/cgi/import.cgi?pdbIds=xxxx.xxxx](http://www.rcsb.org/pdb/cgi/import.cgi?pdbIds=xxxx.xxxx), where “xxxx” are the PDB ID in which a user may be interested. The system invokes that API through a Java URL Connection interface. The returned PDB query results, which are in HTML format, will be forwarded to the controller servlet and further presented without any changes.

It should be noted that this CGI API has a limitation on the number of IDs that can be passed in one request. The limitation is due to the constraints on the number of characters that can be sent by the HTTP’s “Get” method. That value is either 512 or 1024 dependent on the web-server that the PDB database is using.

**Predict action handler:** This handler communicates with MuSE software through MuSE’s CGI API, for example, [http://mmlsun4.pha.unc.edu/cgi-bin/psw/mutagenesis.cgi?pdbid=selectedIds[i]&chain=chain[i]&r1=r1[i]&r2=r2[i]&r3=r3[i]&r4=r4[i]&r5=r5[i]&r6=r6[i]&a1=a1[i]&a2=a2[i]&a3=a3[i]&a4=a4[i]&a5=a5[i]&a6=a6[i]](http://mmlsun4.pha.unc.edu/cgi-bin/psw/mutagenesis.cgi?pdbid=selectedIds[i]&chain=chain[i]&r1=r1[i]&r2=r2[i]&r3=r3[i]&r4=r4[i]&r5=r5[i]&r6=r6[i]&a1=a1[i]&a2=a2[i]&a3=a3[i]&a4=a4[i]&a5=a5[i]&a6=a6[i]); where similar to the PDB action handler, it uses a Java URL Connection interface to invoke MuSE’s CGI API. However, instead of sending the results to the display directly, the handler will parse and reverse engineering the resulting HTML document to produce an internal representation. This representation is used to populate the
corresponding presentation Java Bean. That bean is returned to the controller servlet to be used by the JSP for presentation. The reverse engineering is required in this case for two reasons. One is that the current MuSE’s CGI API can only accept one request at a time. However, our requirements require us to be able to submit a set of requests to MuSE and present all of the analysis results in a user-friendly manner. Thus, the action handler will need to send a set of requests to MuSE in a sequential manner and assemble the results from each request. Another reason is that we are not satisfied with the current presentation of the MuSE results. To change the presentation, we have to understand the data of the results and separate them from their original HTML representation.

“Save Results” action handler. This handler provides a downloading functionality. The handler creates a “tab” separated text file according to the contents selected by users. It then returns the content of this file to the browser. In returning the file content, it sets the content type in the HTTP response header to be "application/unknown". This will make sure that the user gets the opportunity to save the returned file into his/her local file systems.

4.3.3 Web-Services Design

The Web-Services tier is the wrapper to the backend database systems, for example, ProTherm database. The design of this tier follows the blueprint of Web-Services, a service-oriented architecture.
A service-oriented architecture consists of three roles including a service provider, a service registry and a service requestor. A service provider is responsible for creating a service description, publishing that service description to the service registry, and receiving service invocation messages from the service requestor. A service requestor is responsible for finding a service description published to the service registry and is responsible for using service descriptions to bind to (or invoke) services hosted by the service provider. A service registry is responsible for advertising service descriptions and for allowing service requestors to search the collection of service descriptions. Three operations (publish, find, and bind) are the contracts between these roles. In the web-services architecture, those three roles are implemented by the technologies described below.

**Describing services:** Web-Services uses the Web Services Description Language (WSDL) to describe the services. WSDL is an XML-based interface definition language and is platform independent.

**Discovering services:** Web-Services uses Universal Description, Discovery and Integration (UDDI) for its standard service registration and discovery.

**Binding services:** Web-Services uses Simple Object Access Protocol (SOAP) to access remote objects. SOAP is an XML-based protocol and supports different languages and platforms by using an XML-based SOAP envelope. SOAP supports HTTP and other
protocols in the transport layer. This is well suited for web-based application developments.

Based on these standard technologies, many frameworks, including tools and environments, have been developed to facilitate the development of Web-Services applications. When using these frameworks, the design and development work are greatly simplified, since tools provided by the framework normally can automatically generate many of the Web-Services specific functions. For example, with the appropriate tools, WSDL can be automatically generated from the source programs. Discovering Services are handled similarly. Thus, in designing our application, our focus has been on deciding what functions should be provided by this tier of application, how to expose these functions, how to interact with the backend databases, and what framework should be used and how to design this tier to fit into the selected framework.

In the high level, the functions provided by this tier are straightforward: to allow searching the contents in the database (Other functions such as adding, updating and deleting records in the database may be considered in the future according to the practical needs). Thus, a search API is designed, which includes what XML format should be used to return the search results. To search the contents of the database, we will use JDBC to talk with the database. This provides a flexible mechanism to talk to potentially different database management systems. In selecting the framework, we have decided to use Axis, an open source implementation based on Java technologies. A Java-based framework is selected over a .net-based framework since other tiers of our system are built on Java.
technologies. Also, Axis can run in a Tomcat environment. Thus the Web-Services tier can be in the same Tomcat used by other tiers of our system. Furthermore, Axis can generate WSDL from the Java code and it provides flexible ways to facilitate the client and server side programming. Using a Java-based framework such as Axis can greatly simplify the development, deployment and third party integration process. Figure 5 shows a high-level architecture of this Web-Services layer of our implementation.

**Figure 5. Architecture of Web-Services**

In our ProTherm Web-Services implementation, the Web-Services API takes a set of input parameters that represents the search criteria. The meaning and the possible relationships among input variables are pre-defined. For example, the first and the second arguments together represent the range of entries. The complete interpretation of the input parameters is described in Chapter 5.
The results of this search method is an array of XML strings. Here is an example XML string.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<item>
  <entry>1</entry>
  <pdb_wild>_1BP2</pdb_wild_>
  <protein_>PHOSPHOLIPASE A2</protein_>
  <source_>BOVINE</source_>
  <mol_weight>14536.1</mol_weight>
  <mutation>WILD</mutation>
  <state>2</state>
  <dG_H2O>9.5</dG_H2O>
  <ddG>0.0</ddG>
  <pH>8.0</pH>
  <buffer_name>borate</buffer_name>
  <measure>CD</measure>
  <method>GdnHCl</method>
  <reversibility>UNKNOWN</reversibility>
</item>
```

The complete XML document and its interpretation are described in Chapter 5.

With the selection of using Axis as our Web-Services development framework, our interfaces will be first coded as Java methods. The tools in the Axis are used to generate WSDL. After the WSDL is generated, there are two approaches to access these interface. First, any Web-Services client, either based on Java or .net, can access the service interface. The second option is to use Axis tools to generate a Java-based object-oriented client/server program. The tool generates the client stub and server skeleton classes from WSDL. The server side will then use the skeleton to communicate with the Java implementation code. The client side uses the stub to write their implementation in an object-oriented way. Our current implementation uses the second approach: the middle-
tier uses the methods in the stub to communicate with the skeleton of the Web-Services and the Web-Service tier uses the skeleton to call the Java implementation methods.

4.3.4 ProTherm Database Design

The ProTherm database stores the information about the thermodynamic data of Proteins and Mutants. Each entry includes numerical data of several thermodynamic parameters, sequence and structural information, experimental methods and conditions, and literature information. The thermdata table (Table 1) is designed to contain the above information.

The current system design does not talk with the production version of the ProTherm databases directly. The main reason is that the ProTherm developer team was not very responsive to our request to develop a Web-Services interface for the ProTherm. Certainly, it is possible for us to use their web APIs to access their database and reverse-engineer the returned HTML representation of results. However, we found that the complexity of the returned HTML data prevents us from conducting reverse-engineering effectively. We also found that several queries in the current ProTherm interface do not return the expected results. Fortunately, it is possible to download the entire database records of ProTherm. Further, we believe that if we build a local copy of the ProTherm database using downloaded database records and then write a Web-Services interface for it, we can better demonstrate the idea of this thesis. Thus, we decided to implement a local copy of the ProTherm database. In our implementation of the ProTherm database, we collected the sample data from the production version of the ProTherm database at http://www.rtc.riken.go.jp/jouhou/protherm/protherm_search.html
It should be noted that we can provide our Web-Services wrapper to the ProTherm developer team and they will be able to present a Web-Services interface for the production version of the ProTherm database. If they decide to do so in the future, there will be no changes to our middle tier (except to point to their web-server) and presentation code.

Table 1: The thermdata table

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTRY</td>
<td>int (5)</td>
<td>PK</td>
<td>The id for each entry.</td>
</tr>
<tr>
<td>PDB_WILD</td>
<td>Varchar (10)</td>
<td></td>
<td>The PDB code for the native protein.</td>
</tr>
<tr>
<td>PDB_MUTANT</td>
<td>Varchar (10)</td>
<td></td>
<td>The PDB code for the mutant protein.</td>
</tr>
<tr>
<td>PROTEIN</td>
<td>Varchar (30)</td>
<td></td>
<td>The name of the protein.</td>
</tr>
<tr>
<td>SOURCE</td>
<td>Varchar (30)</td>
<td></td>
<td>The source of the protein.</td>
</tr>
<tr>
<td>MOL_WEIGHT</td>
<td>Float</td>
<td></td>
<td>The molecular weight of the protein.</td>
</tr>
<tr>
<td>MUTATION</td>
<td>Varchar (10)</td>
<td></td>
<td>The details about the mutant including residue in wild, residue number and residue in mutant protein.</td>
</tr>
<tr>
<td>MUTATION_NUM</td>
<td>int (4)</td>
<td></td>
<td>The type of the mutation including wild type (0), single (1), double (2) and multiple (3) mutation.</td>
</tr>
<tr>
<td>AMINO_LENGTH</td>
<td>Float</td>
<td></td>
<td>The length or amino acid numbers of the protein.</td>
</tr>
<tr>
<td>E_C_NUMBER</td>
<td>Varchar (20)</td>
<td></td>
<td>The enzyme commission number.</td>
</tr>
<tr>
<td>PMD_NO</td>
<td>Varchar (10)</td>
<td></td>
<td>The protein mutant database accession number.</td>
</tr>
<tr>
<td>SEC_STR</td>
<td>Varchar (10)</td>
<td></td>
<td>The secondary structure information including Helix, Strand, Turn and Coil.</td>
</tr>
<tr>
<td>ASA_PERCENT</td>
<td>Float</td>
<td></td>
<td>The accessible surface area (ASA) of the residue in wild type (computed using the program ASC, Analytical Surface Calculation) expressed in percentage.</td>
</tr>
<tr>
<td>ASA</td>
<td>Float</td>
<td></td>
<td>The accessible surface area (ASA) of the residue in wild type (computed using the program ASC,</td>
</tr>
<tr>
<td>STATE</td>
<td>int (4)</td>
<td>The number of transition states</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
<td>---------------------------------</td>
<td></td>
</tr>
<tr>
<td>dG_H2O</td>
<td>Float</td>
<td>The Free Energy of unfolding in water, determined by denaturant (urea; GdnHCl; GSSG/GSH; GdnSCN) denaturation of proteins and extrapolation of the data to zero concentration of denaturant [kcal/mol]</td>
<td></td>
</tr>
<tr>
<td>ddG_H2O</td>
<td>Float</td>
<td>dG_H2O(mutant) - dG_H2O(wild) [kcal/mol]</td>
<td></td>
</tr>
<tr>
<td>dG</td>
<td>Float</td>
<td>1) Free energy of unfolding at a certain concentration of denaturant in the case of denaturant denaturation methods</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2) Free energy of unfolding obtained for extrapolation temperature T using dCp in the case of thermal denaturation method [kcal/mol]</td>
<td></td>
</tr>
<tr>
<td>ddG</td>
<td>Float</td>
<td>dG(mutant) - dG(wild) [kcal/mol] Free energy of unfolding obtained with Schellman equation (ddG = dTm.dS) in the case of thermal denaturation method [kcal/mol]</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>float</td>
<td>In the case of denaturant denaturation methods, T is the temperature used in the experiment. In the case of thermal denaturation methods, T is the temperature at which thermodynamic quantities such as unfolding free energy (DG), unfolding free-energy change due to mutation (DDGG) and sometimes unfolding enthalpy change (DH) were calculated. [degree Celsius]</td>
<td></td>
</tr>
<tr>
<td>Tm</td>
<td>float</td>
<td>The midpoint temperature of the thermal unfolding for thermal denaturation methods [degree Celsius]</td>
<td></td>
</tr>
<tr>
<td>dTm</td>
<td>float</td>
<td>Tm(mutant) - Tm(wild) [degree Celsius]</td>
<td></td>
</tr>
<tr>
<td>dHvH</td>
<td>float</td>
<td>van't Hoff enthalpy change of denaturation (enthalpy obtained from the temperature dependence of the denaturation equilibrium constant)</td>
<td></td>
</tr>
<tr>
<td>dHcal</td>
<td>float</td>
<td>The calorimetric enthalpy change of denaturation (enthalpy measured by calorimetry)</td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>float</td>
<td>The slope of dG on denaturant concentration [dG vs. urea/GdnHCl] [kcal/mol/M]</td>
<td></td>
</tr>
<tr>
<td>Cm</td>
<td>float</td>
<td>The concentration of denaturant at which 50% of the protein is unfolded [M]</td>
<td></td>
</tr>
<tr>
<td>dCp</td>
<td>float</td>
<td>The heat capacity change of denaturation [kcal/mol/K]</td>
<td></td>
</tr>
<tr>
<td>pH</td>
<td>float</td>
<td>The pH value.</td>
<td></td>
</tr>
<tr>
<td><strong>BUFFER_NAME</strong></td>
<td>varchar (50)</td>
<td>The name of the buffer used in the experiment</td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------</td>
<td>-----------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>BUFFER_CONC</strong></td>
<td>float</td>
<td>The concentration of the buffer.</td>
<td></td>
</tr>
<tr>
<td><strong>ION_NAME</strong></td>
<td>varchar (50)</td>
<td>The name of the added ion.</td>
<td></td>
</tr>
<tr>
<td><strong>ION_CONC</strong></td>
<td>float</td>
<td>The concentration of the ion.</td>
<td></td>
</tr>
<tr>
<td><strong>PROTEIN_CONC</strong></td>
<td>float</td>
<td>The concentration of the protein when the experiment has been performed.</td>
<td></td>
</tr>
<tr>
<td><strong>MEASURE</strong></td>
<td>varchar (30)</td>
<td>The experiments performed to measure the thermodynamic parameters including Fluorescence spectroscopy, Circular Dichroism, Differential Scanning Calorimetry, Absorbance, NMR, etc. Keywords: Fl, CD, DSC, Abs, NMR, others</td>
<td></td>
</tr>
<tr>
<td><strong>METHOD</strong></td>
<td>varchar (30)</td>
<td>The experimental method of denaturation including Thermal, Urea, GdnHCl etc.</td>
<td></td>
</tr>
<tr>
<td><strong>REVERSIBILITY</strong></td>
<td>varchar (10)</td>
<td>The reversibility of denaturation. If the reversibility is mentioned in the paper, &quot;Yes&quot; or &quot;No&quot; is listed, otherwise this field is set to blank.</td>
<td></td>
</tr>
<tr>
<td><strong>ACTIVITY</strong></td>
<td>float</td>
<td>The specific activity of each mutant (in percentage)</td>
<td></td>
</tr>
<tr>
<td><strong>ACTIVITY_Km</strong></td>
<td>float</td>
<td>The Michaelis-Menten constant (substrate concentration at half maximal rate, Vmax/2) [mM].</td>
<td></td>
</tr>
<tr>
<td><strong>ACTIVITY_Kcat</strong></td>
<td>float</td>
<td>The Michaelis-Menten constant (rate constant) [1/s]</td>
<td></td>
</tr>
<tr>
<td><strong>ACTIVITY_Kd</strong></td>
<td>float</td>
<td>The dissociation constant [micro molar]</td>
<td></td>
</tr>
<tr>
<td><strong>KEY_WORDS</strong></td>
<td>text</td>
<td>A list of keywords used for the specific protein in the article.</td>
<td></td>
</tr>
<tr>
<td><strong>REFERENCE</strong></td>
<td>longtext</td>
<td>The complete reference of the article with a link to NCBI database</td>
<td></td>
</tr>
<tr>
<td><strong>AUTHOR</strong></td>
<td>text</td>
<td>The name of the authors.</td>
<td></td>
</tr>
<tr>
<td><strong>REMARKS</strong></td>
<td>longtext</td>
<td>Some specific comments</td>
<td></td>
</tr>
</tbody>
</table>

### 4.4 System Development Environment

The system development environments are briefly described in following table (Table 2):

<p>| <strong>Table2: System Development Environment Table</strong> |</p>
<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware</td>
<td>Dell GX1 PC with PentiumIII Processor and 384M memory</td>
</tr>
<tr>
<td>Operating System</td>
<td>Windows 2000 Professional</td>
</tr>
<tr>
<td>JDK Version</td>
<td>j2sdk1.4.1</td>
</tr>
<tr>
<td>Web Server / Servlet Container</td>
<td>Jakarta Tomcat 4.1.18</td>
</tr>
<tr>
<td>Soap Engine</td>
<td>Axis-1.1RC</td>
</tr>
<tr>
<td>XML Parser</td>
<td>JDOM Beta 8</td>
</tr>
<tr>
<td>Extended Library</td>
<td>Servlet.jar, xerces.jar, mm.mysql-2.0.2-bin.jar</td>
</tr>
<tr>
<td>Database System</td>
<td>MySQL 3.23.55 Window Version</td>
</tr>
</tbody>
</table>
Chapter 5: User Guides

5.1. Guide for End-User

5.1.1. Overview
The interface of the current implementation not only allows users to access the integrated system, but also allows users to directly access the ProTherm database, the PDB database and the MuSE software. A user accesses the individual ProTherm, PDB databases and MuSE respectively by clicking “Search PDB only”, Search ProTherm only”, or “Access MuSE only” buttons on the main navigation frame. This will bring the individual website to the bottom content frame of our user interface. Similarly, a user accesses the integrated system by clicking “Integrated System” in the navigation frame. Since each individual database has its own user guide on how to use and search the corresponding database, this user guide will focus on explaining how to use the integrated system.

5.1.2. The ProTherm Search Page
The ProTherm Search page of the Integrated System is the home page of the system. It provides a form-based interface that allows users to select different query criteria by selecting or filling up the corresponding fields such as entry id, PDB id, protein name, and reversibility fields, etc. If none of the fields are selected or filled up, the system
returns the first 50 entries. If multiple fields are used for a search, the system returns a list of entries satisfying all of the specified field values. Table 3 describes the meaning of each field in the user interface.

### Table 3. ProTherm Search Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry</td>
<td>Entry number. This option can be used for getting data from a particular entry or a range of entries.</td>
<td>3012, 10107-10365</td>
</tr>
<tr>
<td>Protein</td>
<td>Name of the protein. Multiple words can be entered with spaces.</td>
<td>Gene V, Phospholipase A2</td>
</tr>
<tr>
<td>PDB Code</td>
<td>Protein Data Bank code for the native protein</td>
<td>1BP2, 2RN2</td>
</tr>
<tr>
<td>Mol-weight</td>
<td>Molecular weights were calculated as follows: The amino acid sequence corresponding to a protein was taken from PIR database, and leader (signal) regions were removed. Then, the molecular weights were calculated according to the amino acid sequence. Amino and Carboxy terminals were taken into account. Any mutations to the amino acid sequence were taken into account. Post-translational modifications were NOT taken into account.</td>
<td>14536.1, 17611.0</td>
</tr>
<tr>
<td>Mutation</td>
<td>Details about the mutation: residue in wild type, residue number and residue in mutant protein</td>
<td>A 123 G, D 94 E</td>
</tr>
<tr>
<td>Sec.Structure</td>
<td>Secondary Structural Information (H -- Helix, S -- Strand, T -- Turn and C --Coil)</td>
<td>H, S, C</td>
</tr>
<tr>
<td><strong>Accessibility</strong></td>
<td>Accessible surface area (ASA) of the residue in wild type (computed using the program ASC, Analytical Surface Calculation) expressed in units of A**2. Accessibility (%) is defined as the ASA of the residue (X) in its parent protein, computed with ASC divided by the ASA of the residue in an extended tripeptide Ala-X-Ala conformation.</td>
<td></td>
</tr>
</tbody>
</table>
| Example: | - 69.12  
- 39.13 |
| **Measure** | The experiments performed to measure the thermodynamic parameters including: Fluorescence --- Fluorescence spectroscopy, CD --- Circular Dichroism, DSC --- Differential Scanning Calorimetry, Abs --- Absorbance, NMR, etc. |
| Example: | - Check checkbox at left side corresponding method |
| **Method** | Experimental method of denaturation including: Thermal, Urea, GdnHCl etc. (activity: 50% relative remaining activity of the enzyme after the heat treatment) |
| Example: | - Check checkbox at left side corresponding method |
| **PH** | The pH value. |
| Example: | - 8.0  
- 6.8 To 9.0 |
| **dTm** | dTm = Tm(mutant) - Tm(wild) [degree Celsius] |
| Example: | - 5.7  
- -1.6 To 3.2 |
| **Tm** | Midpoint temperature of the thermal unfolding for thermal denaturation methods [degree Celsius] |
| Example: | - 49.8  
- 23.2 To 46.0 |
| **T** | In the case of denaturant denaturation methods, T is the temperature used in the experiment. In the case of thermal denaturation methods, T is the temperature at which thermodynamic quantities such as unfolding free energy (DG), unfolding free-energy change due to mutation (DDG) and sometimes unfolding enthalpy change (DH) were calculated. [degree Celsius] |
| Example: | - 49.8  
- 23.2 To 46.0 |
| **dHvH** | van't Hoff enthalpy change of denaturation (enthalpy obtained from the temperature dependence of the denaturation equilibrium constant) |
| Example: | - 102.4  
- 90.4 To 98.6 |
| **dCp** | Heat capacity change of denaturation [kcal/mol/K] |
| **dG** | (1) Free energy of unfolding at a certain concentration of denaturant in the case of denaturant denaturation methods  
(2) Free energy of unfolding obtained for extrapolation temperature T using dCp in the case of thermal denaturation method [kcal/mol]  
Example:  
- 2.5  
- 2.5 To 2.8 |
| **dG_H2O** | Free Energy of unfolding in water, determined by denaturant (urea; GdnHCl; GSSG/GSH; GdnSCN) denaturation of proteins and extrapolation of the data to zero concentration of denaturant [kcal/mol]  
Example:  
- 8.3  
- 8.3 To 9.1 |
| **ddG** | ddG = dG(mutant) - dG(wild) [kcal/mol], Free energy of unfolding obtained with Schellman equation (ddG = dTm.dS) in the case of thermal denaturation method [kcal/mol]  
Example:  
- -0.2  
- -0.2 To 2.1 |
| **ddG_H2O** | ddG_H2O = dG_H2O(mutant) - dG_H2O(wild) [kcal/mol]  
Example:  
- 0.13  
- 0.13 To 0.74 |
| **STATE** | Number of transition states (space denotes the two state transition)  
Example:  
- 2  
- 3 |
| **Reversibility** | Reversibility of denaturation. If the reversibility is mentioned in the paper, "Yes" (and % of reversibility if described) or "No" is listed, or otherwise this field is set to blank.  
Example:  
- Yes  
- No |
| **Key_words** | List of keywords used for the specific protein/article. Multiple words can be entered with spaces.  
Example:  
- Hydrophobic  
- Electrostatic |
| **Author** | Name of the authors. Multiple words can be entered with spaces. No comma after last name  
Example:  
- SERRANO L.  
- MATOUSCHEK A. |
There are two options for displaying results. One is a short display. Another is a detailed display. The default display is a short display. The default number of records to be displayed is 50. A user can change this default value by modifying the “Display hit list” fields. If any of the “Display hit list” fields are not filled up with values, the system will use the default display number.

The system displays the search results in the tabular format for both display options.

5.1.3. The ProTherm Query Result Page

At the ProTherm query result page, there are checkboxes at the first column for each record. A user can use them to select or deselect records of interest and submit them to the subsequent database or software. There are three submit buttons, “Import to PDB”, “Import to MuSE” and “Save Results” at both the top and the bottom of the content frame. The system’s validation process will prevent the submission of the request if none of the records are selected.

The “Saved Results” button allows a user to save the selected result sets to the local disk drive. The saved file is a text file with a tab-delimited format.

The “Import to PDB” button allows a user to submit the PDB_IDs of selected entries as inputs to search the PDB database. There are two types of result display interface provided by the PDB database. These are “Query Result Browser” and “Structure Explorer”. The “Query Result Browser” interface is used if multiple structures are
returned from the query. The “Structure Explorer” interface is used if a single structure is
returned. For a detailed description of how to interpret these search results, users can
refer to the user guide of PDB at http://www.rcsb.org/pdb/help-results.html.

The “Import to MuSE” button allows a user to submit the PDB_IDS of selected entries to
the MuSE system. After clicking this button, the MuSE mutation prediction page will be
displayed.

5.1.4. The MuSE Mutation Prediction Page

At the MuSE mutation prediction page, a user can use the checkboxes on the left hand
side of each row to select the entries for predicting the Virtual Mutation of the proteins.
The user also needs to provide several parameters that are necessary for conducting the
prediction. The default values of these parameters in some fields can be reset if needed.

5.1.5. The MuSE Prediction Results Page

The predicted results by the Virtual Mutagenesis are displayed in a tabular format. At this
result page, a user can select the result sets of interest and save them to the local disk
drive. The saved file is in the tab-delimited format (.txt). For a detailed description on
how to fill up the necessary parameters and how to interpret the results, a user should

5.2. Guide for Application Developers
5.2.1. Overview

As we have discussed in the previous chapter, the ProTherm Web-Services interface is implemented by using the Axis, an open source SOAP implementation. As you would expect, the function of this Web-Service can be found in its WSDL (See Appendix B – WSDL of ProTherm Web-Services). Specifically, this WSDL definition describes what the service does, how it can be accessed and where it is located. Application developers can also dynamically access the WSDL of our deployed ProTherm Web-Services at http://yellow.ils.unc.edu:8080/axis/services/Protherm?wsdl. We have used this WSDL definition to automatically generate the server side skeleton that interacts with the deployed Web-Services implementation object and the client side stub that communicates with the skeleton. The main function of this Web-Services is to allow searching the contents in the database.

Application developers who want to talk with this ProTherm Web-Services interface should download the WSDL. Using this WSDL, you can use Axis tools to automatically generate a client side stub. That stub can communicate with the skeleton that is generated and used by our implementation. Application developers can then write the WSDL client to use the generated stub to access the methods provided by the ProTherm Web-Services. The methods provided by the ProTherm Web-Services have a set of parameters, which will be discussed in the next section. Developers should be familiar with this ProTherm Web services API documentation to understand the specifics on how to form valid search method calls and what values will be returned in what formats.
5.2.2. ProTherm Web-Services API Documentation

The following table lists the Java API documentation of the implementation class. This API contains the same kind of parameters that you can find in the corresponding WSDL method description. However, we think that this API gives more detailed description about the parameters and the developers may be more comfortable with Java API documentation than WSDL.

<table>
<thead>
<tr>
<th>Class Protherm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java.lang.object ---- Protherm</td>
</tr>
<tr>
<td>public class Protherm extends Java.lang.object</td>
</tr>
</tbody>
</table>

Methods:

- **public String[] search();**
  
  Search the ProTherm database with no criteria.

  Parameters:
  
  None

  Returns:
  
  An array of XML string, each xml string represents a record in database.

- **public String[] search(String count1, String count2);**
  
  Search the ProTherm database with limiting the number of records returned.

  Parameters:
  
  count1 --- the start index of returned records.
  
  count2 --- the end index of returned records

  Returns:
  
  An array of XML string, each xml string represents a record in database.

- **public String[] search(String entry1, String entry2, String pdb_wild, String protein, String source, String weight_from, String weight_to, String muta0, String muta1, String muta2, String muta3);**
String muta3, String sec0, String sec1, String sec2, String sec3, String asa1, String asa2,
String asa3, String measure0, String measure1, String measure2, String measure3, String
measure4, String measure5, String method0, String method1, String method2, String
method3, String ph1, String ph2, String dtm0, String dtm1, String dtm2, String dh0,
String dh1, String dh2, String ddg0, String ddg1, String ddg2, String stat0, String stat1,
String stat2, String revs, String key0, String key1, String key2, String author0, String
author1, String author2, String count1, String count2);

Search the ProTherm database with various criteria.

Parameters:

entry1 --- the start id value of entries.
entry2 --- the end id value of entries.
pdb_wild --- the pdb id.
protein --- the name of protein.
source --- the name of source.
weight_from --- the start value of amino weight.
weight_to --- the end value of amino weight.
muta0 --- the value of check box for search wild type mutation. Its value is “Wild Type” or null.
muta1 --- the value of check box for search single mutation. Its value is “Single” or null.
muta2 --- the value of check box for search double mutation. Its value is “Double” or null.
muta3 --- the value of check box for search multiple mutation. Its value is “Multiple” or null.
sec0 --- the value of check box for search secondary structure. Its value is "Helix" or null.
sec1 --- the value of check box for search secondary structure. Its value is "Sheet" or null.
sec2 --- the value of check box for search secondary structure. Its value is "Turn" or null.
sec3 --- the value of check box for search secondary structure. Its value is "Coil" or null.
asa1 --- the start value of Accessible Surface Area.
asa2 --- the end value of Accessible Surface Area.
asa3 --- the unit of Accessible surface area. Its value is “0-%” or "1-A**2".
measure0 --- the value of check box for search measure type of experiments. Its value is
"Absorbance" or null.
measure1 --- the value of check box for search measure type of experiments. Its value is "CD"
or null.
measure2 --- the value of check box for search measure type of experiments. Its value is "DSC"
or null.
measure3 --- the value of check box for search measure type of experiments. Its value is
"Fluorescence" or null.
measure4 --- the value of check box for search measure type of experiments. Its value is
"NMR" or null.
measure5 --- the value of check box for search measure type of experiments. Its value is
"Others" or null.
method0 --- the value of check box for search experiments method. Its value is "Thermal" or
null.

method1 --- the value of check box for search experiments method. Its value is "GdnHCl" or null.

method2 --- the value of check box for search experiments method. Its value is "Urea" or null.

method3 --- the value of check box for search experiments method. Its value is "Others" or null.

ph1 --- the start value of pH

ph2 --- the end value of pH

dtm0 --- the option for search dTm, Tm or T. its value are “0-dTm”, “1-Tm” and “2-T”.

dtm1 --- the start value of dTm or Tm or T

dtm2 --- the end value of dTm or Tm or T

dh0 --- the option for search dH, dCp, dG or dG_H2O. Its value are "0-dH", "1-dCp", "2-dG_H2O" and "3-dG".

dh1 --- the start value of dH, dCp, dG or dG_H2O.

dh2 --- the end value of dH, dCp, dG or dG_H2O.

ddg0 --- the option for search ddG or ddG_H2O. Its value are "0-ddG" and "1-ddGH2O".

ddg1 --- the start value of ddG or ddG_H2O.

ddg2 --- the end value of ddG or ddG_H2O.

stat0 --- the value of check box for search state. Its value is "2" or null.

stat1 --- the value of check box for search state. Its value is "3" or null.

stat2 --- the value of check box for search state. Its value is "&gt;3" or null.

revs --- the option for search reversibility. Its value are "0-Any", "1-Yes" and "2-No"

key0 --- the value is "OR" or “AND”.

key1 --- the keyword to search for.

key2 --- the keyword to search for.

author0 --- Its value is "OR" or “AND”.

author1 --- the author name.

author2 --- the author name.

count1 --- the start index of returned records.

count2 --- the end index of returned records

**Returns:**

An array of XML string, each xml string represents a record in database.

---

**5.2.3. XML Format**
As we have mentioned in the previous section, an array of XML string will be returned by the search method. Each XML string represents a record in database and has the following format:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<item>
  <entry>1</entry>
  <pdb_wild_>1BP2</pdb_wild_>
  <pdb_mutant />
  <protein_>PHOSPHOLIPASE A2</protein_>
  <source_>BOVINE</source>
  <mol_weight>14536.1</mol_weight>
  <mutation>WILD</mutation>
  <mutation_num>0</mutation_num>
  <amino_length>130.0</amino_length>
  <e_c_number>EC 3.1.1.4</e_c_number>
  <pmd_no>A930651</pmd_no>
  <sec_str />
  <asa_percent>0.0</asa_percent>
  <asa>0.0</asa>
  <state>2</state>
  <dG_H2O>9.5</dG_H2O>
  <ddG_H2O>0.0</ddG_H2O>
  <dG>0.0</dG>
  <ddG>0.0</ddG>
  <T>30.0</T>
  <Tm>0.0</Tm>
  <dTm>0.0</dTm>
  <dHvH>0.0</dHvH>
  <dHcal>0.0</dHcal>
  <m>1.47</m>
  <Cm>6.9</Cm>
  <dCp>0.0</dCp>
  <pH>8.0</pH>
  <buffer_name>borate</buffer_name>
  <buffer_conc>0.0</buffer_conc>
  <ion_name />
  <ion_conc>0.0</ion_conc>
  <protein_conc>0.0</protein_conc>
  <measure>CD</measure>
  <method>GdnHCl</method>
  <reversibility>UNKNOWN</reversibility>
  <activity>0.0</activity>
  <activity_km>1.4</activity_km>
  <activity_kcat>675.0</activity_kcat>
  <activity_kd>0.0</activity_kd>
  <key_words>CATALYTIC(TRIAD; CONFORMATIONAL STABILITY;)</key_words>
  <author>LI Y. &amp; TSAI M.-D.</author>
  <remarks>ADDITIVE : EDTA(0.1 MM)</remarks>
</item>
```
As you can see from above example XML string, we have taken a simple approach to organize the return values. In the future, we may impose a hierarchical structure to group related concepts together.

5.2.4. Procedure for Generating Stub and Client

This section summarizes the detailed steps on how to use the WSDL to generate a stub, and write the WSDL stub-skeleton based client in the Tomcat and Axis development environment.

1. Copy or download the WSDL of ProTherm Web-Services.
   
   You can either copy the WSDL of the ProTherm Web-Services from Appendix B or download it form [http://yellow.ils.unc.edu:8080/axis/services/Protherm?wsdl](http://yellow.ils.unc.edu:8080/axis/services/Protherm?wsdl).
   
   Either way, you should save it into a file named Protherm.wsdl

2. Generate Skeleton and Stub.

   Axis provides a utility to generate the skeletons and stubs from the WSDL. An example invocation of this utility is as follows:

   ```
   java org.apache.axis.wsdl.WSDL2Java --server-side --skeletonDeploy true
   Protherm.wsdl
   ```

   This example assumes that the WSDL file has been saved as Protherm.wsdl and is accessible to the JVM. The generated files will appear in a subdirectory of the current directory.

3. Write WSDL client based on generated stub and skeleton.

   The WSDL-based stub-skeleton approach is based on RPC- or RMI-style access. In the above step of generating stub and skeleton, the Axis tools also generate the ServiceLocator object from WSDL for the ProTherm services. The locator is
used to get the stub for that service. Once the stub object is obtained, the client code can use that object to call the desired method of the service. This approach abstracts out the details of the remote invocation to the Web-Services located in the remote Axis and Tomcat environments. From the client programmer’s point of view, it gets a stub object and then calls the methods on that object. That is all the client code needs to do. It is stub and other Axis objects’ responsibility to complete the communication with remote services.

Here is an example code of WSDL client:

```java
ProthermService service = new ProthermServiceLocator();
edu.unc.ils.yellow.axis.services.Protherm.Protherm port = service.getProtherm();
String[][] items = port.search();
```
Chapter 6: Future Work and Conclusion

6.1. Conclusion

One of the challenges for biotechnology researchers is how to effectively access the large amounts of information stored in multiple databases. As the amount of information from the mapping of the human genome increases, the ability to access multiple databases becomes even more critical. However, the existing approaches for the integration of different databases have various limitations. Notably, most of the distributed integration approaches are based on CORBA technology, which has gradually lost support from the leading computer software vendors.

This thesis introduced a multiple tier integration architecture that uses the newly introduced Web-Services as its core technology. In this architecture, Java Servlet and JSP technologies are used to provide an extensible and flexible integration management component. Web-Services is used to create a reusable and open wrapper for the ProTherm database. The JDBC technology is used to access our version of the ProTherm database built with MySQL. This JDBC technology provides the capability to implement the database that is built with other database management systems without code changes. Last, the interface is designed with JSP, JavaScript and multiple HTML frames. This
interface design provides a unified view to the integrated system and other related databases and a dynamic user experience.

Our implemented prototype system integrated the PDB and the ProTherm databases and the MuSE analytic software. The user requirements are gathered from interviewing a UNC research team. The core functionalities of the integrated system are as follows:

1. Allow searching PDB and ProTherm databases and using MuSE analysis system.
2. Allow searching results from ProTherm to be used to search PDB and as well as to be imported to the MuSE system.
3. Allow saving the results in the various stages of the integrated system.
4. Allow analysis of multiple items conducted by MuSE and present the results in a user-friendly manner.

This example implementation not only proves our concept of Web-Services integration for bioinformatics databases and software, but also provides a feasible solution for the UNC research team and others who are interested in using ProTherm, PDB and MuSE in a similar manner. The complete system can be accessed using a web browser at the following URL: http://yellow.ils.unc.edu:8080/junzhang/jsp/. This URL is valid until July 31, 2003.

6.2. Future Work and Implications for Other Situations
Many interesting research and implementation topics can be pursued along the line of the idea proposed in this thesis.

First, the current implementation can be applied to the production version of the ProTherm database. Our implemented version of ProTherm is a subset of the production version. One way to make our current implementation practically usable is to contact the ProTherm developer team, provide the ProTherm Web-Services implementation in this thesis and set up a Web-Services interface for the production version of ProTherm. With the methodologies and instructions described in this thesis, this implementation is a straightforward task.

Second, apply this idea to integrate other databases and software such as PDB and MuSE using Web Services. The current implementation of integrating PDB and MuSE uses HTTP to talk with those systems directly. Even though this implementation reflects the flexibility of our middle tier, it does pose the challenges to consistently display the search results. For example, we decided to display the returned PDB results without making any changes to it, however, we re-engineered the returned MuSE HTML results. When doing this, we lose control on how the PDB results should be displayed. However, we do have the option to produce a more consistent presentation for the MuSE results.

Third, design standard XML representations for data returned from Web-Services responses. One key benefit of using web-services is the open XML representation of search results. The openness of XML makes it possible that all of the databases can be
accessed and the results can be processed by different applications as long as they can parse the standard XML documents. There is much work underway to develop various standard data representations. Any effort along this line will increase the benefits of developing Web-Service based solutions.

Fourth, quantitatively analyze the flexibility of the integration approach. There are no quantitative analyses to show the effectiveness of the various integration approaches. The implemented system in this thesis has demonstrated the flexibility of Web-Services in integrating different bioinformatics databases and software. However, a quantitative analysis of the system can give better insight into the effectiveness of the integration approach. Thus, our implemented system should be further analyzed and evaluated in this fashion. There is much research about the quantitative analysis of the effectiveness and the flexibilities of the architecture of various software systems (Bengtsson & Bosch, 1999; Kazman et al., 1996; Lassing et. al., 1999). Such quantitative methods can be used to evaluate our implemented integration system. This evaluation will help us understand how effective the proposed integration approach is for accessing heterogeneous bioinformatics databases.

Last, there are various implementation specific issues that can be addressed. To name a few: allow saving files in other popular data formats such as Microsoft Excel; integrate statistical software into the system; provide query and result manipulation functions; develop a program to automatically update and synchronize our implemented version of ProTherm with the production version; use CCS to control presentation; and conduct
usability tests. Addressing these problems may become indispensable if we want to further enhance the system to become a production quality product.
References


Protein Data Bank (PDB): http://www.rcsb.org/pdb/

Rommel, J. (2001, August). Will web services jump-start the software slump?. JavaWorld


ProTherm Thermodynamic Database for Proteins and Mutants:  
http://www.rtc.riken.go.jp/jouhou/protherm/protherm_search.html
Appendix A

Screen Shots of Interface

Figure 6. The ProTherm Search Page
Figure 7. The ProTherm Query Result Page

The Integrated Bioinformatics Databases System

ProTherm Search Results:

<table>
<thead>
<tr>
<th>Entry</th>
<th>PDB_ID</th>
<th>Protein</th>
<th>Source</th>
<th>dG_H2O</th>
<th>dG_H2O</th>
<th>T</th>
<th>Mem</th>
<th>Mi</th>
<th>dHcal</th>
<th>m</th>
<th>CmpH</th>
<th>Measure</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1BP2</td>
<td>PHOSPHOLIPASE A2</td>
<td>BOVINE</td>
<td>9.5</td>
<td>0.0</td>
<td>0.0</td>
<td>30.0</td>
<td>0.0</td>
<td>2.1</td>
<td>0.0</td>
<td>1.47</td>
<td>6.9</td>
<td>8.0</td>
</tr>
<tr>
<td>2</td>
<td>1BP2</td>
<td>PHOSPHOLIPASE A2</td>
<td>BOVINE</td>
<td>6.5</td>
<td>-3.0</td>
<td>0.0</td>
<td>30.0</td>
<td>0.0</td>
<td>1.2</td>
<td>0.8</td>
<td>1.2</td>
<td>5.4</td>
<td>8.0</td>
</tr>
<tr>
<td>3</td>
<td>1BP2</td>
<td>PHOSPHOLIPASE A2</td>
<td>BOVINE</td>
<td>8.9</td>
<td>0.6</td>
<td>0.0</td>
<td>30.0</td>
<td>0.0</td>
<td>1.34</td>
<td>0.0</td>
<td>1.34</td>
<td>6.6</td>
<td>0.0</td>
</tr>
<tr>
<td>4</td>
<td>1BP2</td>
<td>PHOSPHOLIPASE A2</td>
<td>BOVINE</td>
<td>6.4</td>
<td>-3.1</td>
<td>0.0</td>
<td>30.0</td>
<td>0.0</td>
<td>1.02</td>
<td>0.0</td>
<td>1.02</td>
<td>6.3</td>
<td>8.0</td>
</tr>
<tr>
<td>5</td>
<td>2RN2</td>
<td>RIBONUCLEASE HI</td>
<td>ESCHERICHIA COLI</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>49.8</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>3.0</td>
<td>CD</td>
<td>Thermal</td>
</tr>
<tr>
<td>6</td>
<td>2RN2</td>
<td>RIBONUCLEASE HI</td>
<td>ESCHERICHIA COLI</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>52.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>5.5</td>
<td>CD</td>
<td>Thermal</td>
</tr>
<tr>
<td>7</td>
<td>2RN2</td>
<td>RIBONUCLEASE HI</td>
<td>ESCHERICHIA COLI</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>49.8</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>3.0</td>
<td>CD</td>
<td>Thermal</td>
</tr>
<tr>
<td>8</td>
<td>2RN2</td>
<td>RIBONUCLEASE HI</td>
<td>ESCHERICHIA COLI</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>52.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>5.5</td>
<td>CD</td>
<td>Thermal</td>
</tr>
</tbody>
</table>
Figure 8. The MuSE Mutation Prediction Page
Figure 9. The MuSE Prediction Results Page

The Integrated Bioinformatics Databases System

Virtual Mutagenesis Predicted Results

<table>
<thead>
<tr>
<th>PDB ID</th>
<th>Native Score</th>
<th>Geometry Z Scores</th>
<th>Res score</th>
<th>Contacts</th>
<th>Mutations</th>
<th>Difference</th>
<th>ΔZZVol</th>
<th>ΔZZArea</th>
<th>ΔZZTot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1BR2</td>
<td>37.882</td>
<td>27.201</td>
<td>35.241</td>
<td>33.115</td>
<td>1.4025</td>
<td>Res 99</td>
<td>2</td>
<td>E99A</td>
<td>0.410</td>
</tr>
<tr>
<td>1BO8</td>
<td>39.365</td>
<td>40.349</td>
<td>46.908</td>
<td>36.109</td>
<td>1.796</td>
<td>Res 99</td>
<td>8</td>
<td>F99A</td>
<td>-2.289</td>
</tr>
<tr>
<td>1VQB</td>
<td>16.247</td>
<td>11.899</td>
<td>12.554</td>
<td>21.598</td>
<td>-0.5688</td>
<td>Res 99</td>
<td>1</td>
<td>M99A</td>
<td>0.087</td>
</tr>
<tr>
<td>1BNI</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>-0.5688</td>
<td>Res 99</td>
<td>1</td>
<td>99A</td>
<td>0.000</td>
</tr>
</tbody>
</table>
**Figure 10. Search PDB Only**

The Integrated Bioinformatics Databases System

**PDB Mirrors**

- "Please bookmark a minor site"  
  San Diego Supercomputer Center  
  Rutgers University  
  National Institute of Standards and Technology  
  Cambridge Crystallographic Data Centre, UK  
  National University of Singapore  
  Osaka University, Japan  
  Universidad Federal de Minas Gerais, Brazil  
  Max Delbrück Center for Molecular Medicine, Germany

**Search the Archive**

Enter a PDB ID or keyword

- query by PDB ID only
- match exact word
- search sequence homologs

Search by keyword search form with examples
Search Fields customizable search form
Status Search find entries awaiting release

**Did you find what you wanted?**

<table>
<thead>
<tr>
<th>OTHER SITES</th>
</tr>
</thead>
<tbody>
<tr>
<td>RCSB partner</td>
</tr>
</tbody>
</table>

In citing the PDB please refer to:

- H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat,  
  H. Weissig, I.N. Shindyalov, P.E. Bujol. The Protein Data Bank.  
Figure 11. Search ProTherm Only
Figure 12. Access MuSE Only
Appendix B

WSDL of ProTherm Web-Services

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions
  targetNamespace="http://yellow.ils.unc.edu:8080/axis/services/Protherm"
  xmlns="http://schemas.xmlsoap.org/wsd/"
  xmlns:apachesoap="http://xml.apache.org/xml-soap"
  xmlns:impl="http://yellow.ils.unc.edu:8080/axis/services/Protherm"
  xmlns:intf="http://yellow.ils.unc.edu:8080/axis/services/Protherm"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsd/"
  xmlns:wsdlsap="http://schemas.xmlsoap.org/wsd/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <wsdl:types>
    <wsdl:types>
      <schema
        targetNamespace="http://yellow.ils.unc.edu:8080/axis/services/Protherm"
        xmlns="http://www.w3.org/2001/XMLSchema">
        <import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
        <complexType name="ArrayOf_xsd_string">
          <complexContent>
            <restriction base="soapenc:Array">
              <attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:string[]" />
            </restriction>
          </complexContent>
        </complexType>
      </schema>
    </wsdl:types>
  </wsdl:types>
  <wsdl:message name="searchResponse">
    <wsdl:part name="searchReturn" type="impl:ArrayOf_xsd_string" />
  </wsdl:message>
  <wsdl:message name="searchRequest">
    <wsdl:part name="in0" type="xsd:string" />
    <wsdl:part name="in1" type="xsd:string" />
    <wsdl:part name="in2" type="xsd:string" />
    <wsdl:part name="in3" type="xsd:string" />
    <wsdl:part name="in4" type="xsd:string" />
    <wsdl:part name="in5" type="xsd:string" />
    <wsdl:part name="in6" type="xsd:string" />
    <wsdl:part name="in7" type="xsd:string" />
  </wsdl:message>
</wsdl:definitions>
```
<wsdl:part name="in8" type="xsd:string" />
<wsdl:part name="in9" type="xsd:string" />
<wsdl:part name="in10" type="xsd:string" />
<wsdl:part name="in11" type="xsd:string" />
<wsdl:part name="in12" type="xsd:string" />
<wsdl:part name="in13" type="xsd:string" />
<wsdl:part name="in14" type="xsd:string" />
<wsdl:part name="in15" type="xsd:string" />
<wsdl:part name="in16" type="xsd:string" />
<wsdl:part name="in17" type="xsd:string" />
<wsdl:part name="in18" type="xsd:string" />
<wsdl:part name="in19" type="xsd:string" />
<wsdl:part name="in20" type="xsd:string" />
<wsdl:part name="in21" type="xsd:string" />
<wsdl:part name="in22" type="xsd:string" />
<wsdl:part name="in23" type="xsd:string" />
<wsdl:part name="in24" type="xsd:string" />
<wsdl:part name="in25" type="xsd:string" />
<wsdl:part name="in26" type="xsd:string" />
<wsdl:part name="in27" type="xsd:string" />
<wsdl:part name="in28" type="xsd:string" />
<wsdl:part name="in29" type="xsd:string" />
<wsdl:part name="in30" type="xsd:string" />
<wsdl:part name="in31" type="xsd:string" />
<wsdl:part name="in32" type="xsd:string" />
<wsdl:part name="in33" type="xsd:string" />
<wsdl:part name="in34" type="xsd:string" />
<wsdl:part name="in35" type="xsd:string" />
<wsdl:part name="in36" type="xsd:string" />
<wsdl:part name="in37" type="xsd:string" />
<wsdl:part name="in38" type="xsd:string" />
<wsdl:part name="in39" type="xsd:string" />
<wsdl:part name="in40" type="xsd:string" />
<wsdl:part name="in41" type="xsd:string" />
<wsdl:part name="in42" type="xsd:string" />
<wsdl:part name="in43" type="xsd:string" />
<wsdl:part name="in44" type="xsd:string" />
<wsdl:part name="in45" type="xsd:string" />
<wsdl:part name="in46" type="xsd:string" />
<wsdl:part name="in47" type="xsd:string" />
<wsdl:part name="in48" type="xsd:string" />
<wsdl:part name="in49" type="xsd:string" />
<wsdl:part name="in50" type="xsd:string" />
</wsdl:message>
<wsdl:portType name="Protherm">
<wsdl:operation name="search" parameterOrder="in0 in1 in2 in3 in4 in5 in6 in7 in8 in9 in10 in11 in12 in13 in14 in15 in16 in17 in18 in19 in20 in21 in22 in23 in24 in25 in26 in27 in28 in29 in30 in31 in32 in33 in34 in35 in36 in37 in38 in39 in40 in41 in42 in43 in44 in45 in46 in47 in48 in49 in50">
<wsdl:input message="impl:searchRequest" name="searchRequest" />
</wsdl:operation>
</wsdl:portType>
<wsdl:output message="impl:searchResponse" name="searchResponse" />
  </wsdl:operation>
  </wsdl:portType>
<wsdl:binding name="ProthermSoapBinding" type="impl:Protherm">
  transport="http://schemas.xmlsoap.org/soap/http" />
<wsdl:operation name="search">
<wsdlsoap:operation soapAction="" />
<wsdl:input name="searchRequest">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  namespace="http://yellow.ils.unc.edu:8080/axis/services/Protherm"
  use="encoded" />
</wsdl:input>
<wsdl:output name="searchResponse">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  namespace="http://yellow.ils.unc.edu:8080/axis/services/Protherm"
  use="encoded" />
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="ProthermService">
<wsdl:port binding="impl:ProthermSoapBinding" name="Protherm">
<wsdlsoap:address
  location="http://yellow.ils.unc.edu:8080/axis/services/Protherm" />
</wsdl:port>
</wsdl:service>
</wsdl:definitions>